



# UNIVERSIDAD DE EXTREMADURA

## Escuela Politécnica

Ingeniería Informática

Proyecto Fin de Carrera

“VDSL: un DSL para la generación automática  
de aplicaciones de video vigilancia”

Mario A. Corchero Jiménez

Junio, 2012



# UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Ingeniería Informática

Proyecto Fin de Carrera

“VDSL: un DSL para la generación automática de  
aplicaciones de vídeo vigilancia”

Autor: Mario A. Corchero Jiménez

Fdo.:

Director: Juan Hernández Núñez

Fdo.:

Director: José María Conejero Manzano

Fdo.:

## Tribunal Calificador

Presidente:

Fdo.:

Secretario:

Fdo.:

Vocal:

Fdo.:

## Calificación:

Fecha:



## Prólogo

El proyecto descrito a continuación pretende aunar dos áreas de aplicación diferentes como son el desarrollo dirigido por modelos y la video-vigilancia, aprovechando las ventajas de la primera para la creación de un sistema de control y supervisión de video-vigilancia.

El objetivo es crear un sistema fácilmente configurable y ampliable, que permita al usuario definir su instalación y se genere, a través de un modelo, una aplicación totalmente personalizada para él, pudiendo ver los dispositivos que tiene, realizar grabaciones, programar y gestionar eventos, o incluso crear nuevas herramientas que pueden ser añadidas al sistema a través del modelo. Este sistema, debe ser accesible tanto directamente a través de la aplicación generada como a través de internet por medio de un WebSite, lo cual dotaría al usuario del total control de su instalación, independientemente de su ubicación.

Para cumplir los objetivos, es necesario que el software cumpla los siguientes requisitos:

- El sistema debe ofrecer información acerca de la organización de los dispositivos y su estado.
- La seguridad del sistema debe ser alta y sólo se puede acceder a él autenticándose con un usuario y contraseña.
- Deben establecerse roles y privilegios que distingan a los diferentes tipos de usuarios y restringir las acciones que éstos pueden realizar sobre el sistema.
- El sistema debe ofrecer un modo de recoger los eventos de las cámaras y otros dispositivos para tratarlos dentro del sistema.
- El sistema debe ofrecer un modo de informar a los usuarios con los eventos ocurridos en éste.
- El usuario debe tener la posibilidad de añadir funcionalidad externa al sistema de una forma sencilla.
- Debe crearse un sistema de recepción, tratamiento de eventos complejo que sea configurable por el usuario.

Para alcanzar estos objetivos, el proyecto se encuentra dividido en cuatro partes fundamentales:

- Gestor de eventos.
- Aplicación servidor.
- Cliente.
- DSL (Domain Specific Language) para el modelado del sistema y generación de código final.

En esta documentación se describen en detalle las diferentes partes anteriores, así como la arquitectura y tecnologías utilizadas en el desarrollo de VSDSL.

# Índice

<b>1. Introducción .....</b>	<b>10</b>
1.1    Vídeo-Vigilancia .....	10
1.2    Arquitecturas Distribuidas .....	12
1.2.1    Características .....	12
1.3    Seguridad y Encriptado .....	12
1.4    Desarrollo de Software Dirigido por Modelos .....	13
1.5    Alcance y objetivos del proyecto .....	13
<b>2. Definición del problema .....</b>	<b>14</b>
2.1    Sistema Existente.....	14
2.1.1    Tecnologías .....	15
2.1.2    Análisis de debilidades .....	15
2.2    Software Axis .....	16
2.2.1    Axis Camera Station .....	16
2.2.2    Axis Camera Companion .....	17
2.2.3    Conclusión .....	17
2.3    Otro Software disponible.....	17
2.4    Especificación de requisitos .....	18
<b>3. Estudio de Viabilidad .....</b>	<b>19</b>
3.1    Conexión a la base de datos .....	19
3.1.1    Driver PostgreSQL .....	19
3.1.2    Windows ODBC.....	20
3.2    Implementación del servidor .....	20
3.2.1    C# .....	20
3.2.2    WPF.....	21
3.2.3    WCF.....	21
3.2.4    .NET Framework 4.....	21
3.3    Cámaras Axis.....	21
3.4    Notificación al usuario .....	22
3.4.1    SMS Esendex .....	22
3.4.2    Envío de Correo .....	22
3.5    Cadena de Eventos .....	23
3.5.1    Delegados y Sistema de notificación de eventos .....	23
3.6    Comunicación Cliente-Servidor .....	24
3.6.1    Web Services .....	24
3.6.2    JSON .....	25
3.7    WebSite .....	25

3.7.1	<i>IIS</i> .....	25
3.7.2	<i>HTTPS y SSL</i> .....	26
3.7.3	<i>HTML5</i> .....	26
3.7.4	<i>JavaScript y JQuery</i> .....	26
3.7.5	<i>AJAX</i> .....	27
3.8	Comunicación Servidor-Cliente .....	28
3.8.1	<i>Tecnologías Push</i> .....	28
3.8.2	<i>WebSockets</i> .....	28
3.8.3	<i>Long Polling / Comet</i> .....	30
3.8.4	<i>Server Side Events</i> .....	31
3.8.5	<i>Pusher.com</i> .....	32
3.9	Modelado.....	33
3.9.1	<i>EMF</i> .....	34
3.9.2	<i>GMF</i> .....	34
3.9.3	<i>OCL</i> .....	34
3.9.4	<i>XText</i> .....	34
3.9.5	<i>JET</i> .....	34
3.10	Creadores .....	35
3.11	Hardware Disponible .....	35
3.11.1	<i>Ordenador Servidor</i> .....	35
3.11.2	<i>Cámaras Axis</i> .....	35
4.	<b>Análisis y Diseño.....</b>	<b>38</b>
4.1	Arquitectura.....	38
4.1.1	<i>Aplicación Servidor</i> .....	38
4.1.2	<i>Cliente</i> .....	39
4.1.3	<i>Modelado y generación de código</i> .....	39
4.2	Análisis y diseño del gestor de eventos .....	39
4.2.1	<i>Visión general</i> .....	40
4.2.2	<i>Ejemplo</i> .....	42
4.2.3	<i>Casos de uso</i> .....	43
4.2.4	<i>Diagrama de Secuencia</i> .....	43
4.2.5	<i>Diagrama de clases</i> .....	43
4.2.6	<i>Jerarquía de eventos</i> .....	45
4.2.7	<i>Servicios Disponibles</i> .....	45
4.3	Análisis y diseño del Servidor.....	51
4.3.1	<i>Visión General</i> .....	51
4.3.2	<i>Diagramas de Casos de Uso</i> .....	52

4.3.3	<i>Servicios Web</i> .....	52
4.3.4	<i>Base de datos</i> .....	58
4.3.5	<i>Conexión a la base de datos</i> .....	60
4.3.6	<i>Gestión de contraseñas</i> .....	60
4.3.7	<i>Archivo de configuración</i> .....	61
4.3.8	<i>Usuarios y Roles</i> .....	61
4.3.9	<i>Diagramas de clases Conceptuales</i> .....	62
4.3.10	<i>Diagrama de Clases</i> .....	63
4.3.11	<i>Estructura del proyecto</i> .....	64
4.4	Análisis y diseño del cliente Web.....	65
4.4.1	<i>Visión General</i> .....	65
4.4.2	<i>Diagramas de Casos de Uso</i> .....	65
4.4.3	<i>Diagrama de secuencia</i> .....	66
4.4.4	<i>Seguridad</i> .....	66
4.4.5	<i>Árbol de navegación</i> .....	67
4.4.6	<i>Script</i> .....	67
4.4.7	<i>Notificaciones desde el servidor</i> .....	67
4.4.8	<i>Conexión a las cámaras</i> .....	68
4.4.9	<i>Estructura del proyecto</i> .....	68
4.4.10	<i>Soporte</i> .....	69
4.5	DSL .....	71
4.5.1	<i>Meta modelo</i> .....	72
4.5.2	<i>Restricciones OCL</i> .....	76
4.5.3	<i>Decisiones generales del modelo</i> .....	78
4.5.4	<i>Editores de modelo</i> .....	83
4.5.5	<i>Transformaciones M2T (JET)</i> .....	86
4.5.6	<i>Workspace</i> .....	88
5.	<b>Manual de usuario</b> .....	89
5.1	Manual de la aplicación servidor .....	89
5.1.1	<i>Interfaz de usuario</i> .....	89
5.1.2	<i>Configuración de la base de datos</i> .....	92
5.1.3	<i>Instalación</i> .....	93
5.1.4	<i>Configuración de las cámaras</i> .....	94
5.2	Manual del servidor web .....	95
5.2.1	<i>Descripción de las páginas</i> .....	95
5.2.2	<i>Instalación</i> .....	98
5.3	Manual del DSL .....	103

5.3.1	<i>Editor en árbol</i> .....	103
5.3.2	<i>Editor Grafico</i> .....	107
5.3.3	<i>Editor Textual</i> .....	109
5.3.4	<i>Generación de código</i> .....	109
5.3.5	<i>Despliegue del sistema</i> .....	111
5.3.6	<i>Diseño de los servicios</i> .....	112
5.3.7	<i>Implementar un nuevo servicio</i> .....	118
6.	<b>FAQ del sistema</b> .....	120
7.	<b>Conclusiones</b> .....	122
7.1	Requisitos alcanzados .....	122
7.1.1	<i>Integración del vídeo de las cámaras en el sistema</i> .....	122
7.1.2	<i>Acceso al sistema a través de internet</i> .....	122
7.1.3	<i>Funcionalidad de respuesta de eventos configurable</i> .....	122
7.1.4	<i>Integración de cámaras Axis en el sistema</i> .....	122
7.1.5	<i>Notificación de los eventos a los usuarios</i> .....	122
7.1.6	<i>Garantizar la seguridad del sistema</i> .....	123
7.1.7	<i>Soporte para múltiples usuarios de forma simultánea</i> .....	123
7.1.8	<i>Posibilidad de añadir funcionalidad externa al sistema</i> .....	123
7.1.9	<i>Personalización del layout del sistema</i> .....	123
7.1.10	<i>Integración de otro tipo de dispositivos en el sistema</i> .....	123
7.1.11	<i>Acceso al sistema optimizado para dispositivos móviles</i> . .....	124
7.2	Aprendizaje Adquirido .....	124
7.3	Opinión sobre sistema desarrollado .....	124
8.	<b>Ampliaciones</b> .....	125
8.1	Integración de las grabaciones en la web.....	125
8.2	Generación de un eclipse propio .....	125
8.3	Desarrollo para tablets .....	125
8.4	Integración del modelado en el servidor .....	126
8.5	Configuración del sistema desde el móvil .....	126
8.6	Mayor soporte para otros tipos de dispositivo.....	126
8.7	Mejora del modelado .....	126
8.8	Nuevos servicios por defecto.....	126
8.9	Incremento de la personalización del aspecto de las aplicaciones generadas .....	127
9.	<b>Agradecimientos</b> .....	128
10.	<b>Referencias</b> .....	130
	<b>Anexo I: Utilización Pusher</b> .....	132
	<b>Anexo II: Despliegue de servicios web tipo REST sobre HTTPS en una aplicación no web</b> .....	135

---

Servidor.....	135
Cliente .....	137
<b>Anexo III: Despliegue de WebSockets.....</b>	<b>138</b>
Servidor.....	138
Cliente.....	139



# 1. Introducción

Este capítulo proporciona una visión genérica de las diferentes áreas en las que se enmarca el proyecto: video-vigilancia, sistemas distribuidos, seguridad y desarrollo dirigido por modelos.

## 1.1 Vídeo-Vigilancia

La vídeo vigilancia, tal y como se encuentra descrito en [1], ha estado evolucionando desde 1960, cuando las primeras cámaras fueron instaladas en las calles de Reino Unido. Fue en 1970 cuando se empezó a combinar CCTV (Circuito de Televisión Cerrado) y los sistemas de grabación, punto en el que se puede establecer el nacimiento de la vídeo-vigilancia en sí.



En 1990 comenzó el desarrollo de la vídeo vigilancia “de hogar” en la que los usuarios empezaron a ser particulares y no únicamente empresas de seguridad. Esta área se vio notablemente mejorada gracias a la ayuda de sistemas informáticos, realizando en un inicio el procesado y la grabación del vídeo en ordenadores en lugar de grabadores analógicos. Esto permitió ahorrar en infraestructuras ya que al ser procesado el vídeo a través de software antes de ser almacenado se conseguía un gran ahorro de capacidad. Con estas últimas mejoras la vídeo vigilancia se convirtió en un área consolidada que empezaba a cobrar importancia en la sociedad.

En la actualidad, la vídeo vigilancia está avanzando hacia vídeo sobre IP, es decir, toda la instalación se realiza mediante sistemas informáticos, siendo las cámaras también conectadas a la red sobre TCP/IP. Esto ha traído las siguientes mejoras:

- Las cámaras pueden traer un sistema empotrado con funcionalidad como detección de movimiento, envío de correo, etc...
- Puede desarrollarse un sistema distribuido de cámaras sin necesidad de tener un servidor central para pequeñas instalaciones.
- El usuario tiene acceso a las cámaras y a su configuración a través de la red, sin necesidad de manipularlas manualmente.
- Al no ser un circuito cerrado, el usuario puede acceder desde cualquier parte del mundo a las cámaras a través de internet.
- Incremento de la calidad del vídeo al no realizarse perdidas en la transformación de analógico a digital como en los sistemas anteriores.

Además, en caso de que deseemos una instalación de circuito cerrado (dado que es más segura), puede crearse una red privada TCP/IP sin conexión a internet, ofreciendo una arquitectura parecida a la de la siguiente figura.



Por último, tal y como se muestra en la arquitectura de la figura anterior, puede introducirse en el sistema un decodificador a la red para transformar la señal digital en analógica, en caso de que se tenga instalado sistemas con esta tecnología.

## 1.2 Arquitecturas Distribuidas

Diversos estilos de arquitecturas se han diseñado a lo largo de los años con el desarrollo de otras tecnologías. En concreto, el desarrollo de microprocesadores, que permitió reducir el tamaño y costo de los ordenadores haciéndolos accesibles a más personas, y la evolución de las comunicaciones permitió la creación de un tipo de arquitecturas, las arquitecturas distribuidas.

### 1.2.1 Características

Un sistema distribuido, debe tener las siguientes características:

- Compartición de Recursos. Los recursos en un sistema distribuido están encapsulados y sólo pueden ser accedidos por el software que gestiona estos recursos, como ocurre con el flujo de vídeo de las cámaras o los datos almacenados en una base de datos. Estos recursos son utilizados a través de sus gestores.
- Concurrencia. En un sistema distribuido, la ejecución del sistema se realiza en diferentes máquinas, cada una tiene su función. Además, puede existir concurrencia dentro de cada componente en sí.
- Escalabilidad. El software del sistema no debe cambiar en función de la escala de este, no debe afectarse ya haya dos o cien componentes.
- Tolerancia a fallos. Si un componente falla, sólo debe afectar a los usuarios que utilicen este componente, y no a todo el sistema.
- Transparencia. El usuario no tiene necesidad de conocer la tecnología que se utiliza detrás de su aplicación.

## 1.3 Seguridad y Encriptado

La seguridad de los sistemas informáticos siempre ha sido un punto delicado en el desarrollo de todo proyecto en el que se trate información delicada de los usuarios. Es por ello que un sistema debe asegurar todos los puntos sensibles que tenga, ya que la seguridad de un sistema es tan alta como la de su punto más débil.

Es muy importante proteger el sistema contra ataques externos de hackers y otros tipos de usuarios maliciosos. Para hacernos una idea de la importancia que tiene la lucha contra el ciber-crimen, el

coste total en Reino Unido en lucha contra este tipo de crimen es de veintisiete millones de libras anuales[2].

Para asegurar la información la herramienta es el cifrado, tanto en el envío como en el almacenamiento de información. Así, existen herramientas como SSL, que nos permite cifrar conexiones HTTP a través de internet, o MD5, un método de cifrado muy utilizado para almacenar información.

## 1.4 Desarrollo de Software Dirigido por Modelos

En 2001 el Object Management Group (OMG) adoptó las arquitecturas dirigidas por modelos (MDA) como una aproximación para la utilización de modelos en el desarrollo de software con tres objetivos: portabilidad, interoperabilidad y reusabilidad mediante la separación de los problemas a los que hay que enfrentarse en el desarrollo de software [3].

El Desarrollo de Software Dirigidopor Modelos, basado en las ideas extraídas de MDA, presenta una nueva forma de desarrollar software que busca la separación entre la especificación del sistema y la implementación final. En el DSDM la especificación completa y precisa del sistema, así como de su funcionalidad se realiza a través de modelos, los cuales pueden ser traducidos al código que definirá el sistema.

## 1.5 Alcance y objetivos del proyecto

En el marco definido se busca la creación de una herramienta utilizando desarrollo de software dirigido por modelos para la generación de aplicaciones de video vigilancia.

El objetivo es crear sistemas distribuidos totalmente individuales, adaptado a las necesidades del usuario y proporcionando toda la seguridad posible dado que la información que se tratará es altamente delicada. Los principales dispositivos a utilizar son cámaras Axis IP. El sistema deberá configurarse y generarse a través de un DSL.

Es importante permitir el acceso al sistema a través de internet, para ofrecer a los usuarios la posibilidad de gestionar sus instalaciones de forma remota. Debe cuidarse siempre los aspectos de seguridad, para asegurar que cada componente en la arquitectura distribuida que se va a desarrollar proteja la información a la que tenga acceso de accesos no deseados.

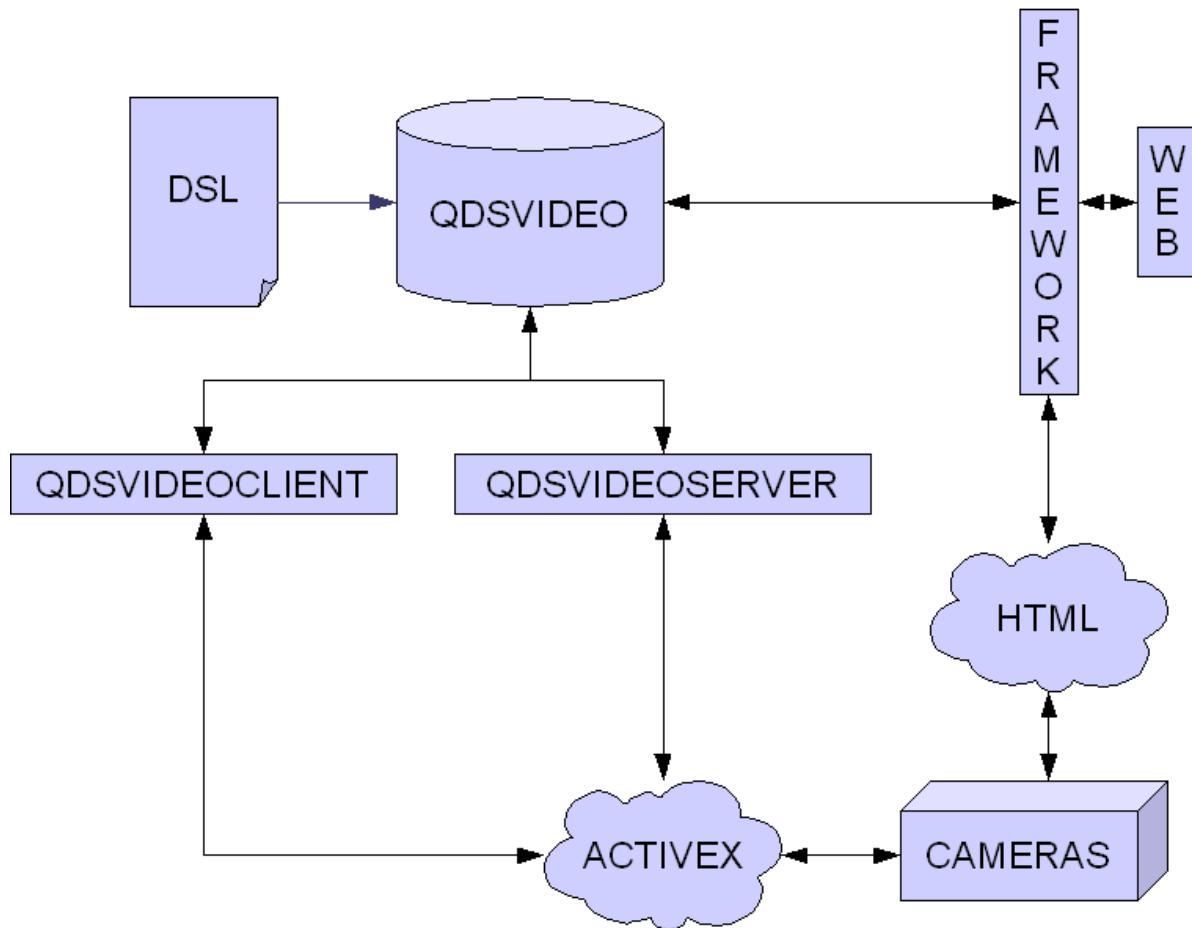
El servidor del sistema se orienta a una plataforma Windows pero se busca total portabilidad para el acceso de los clientes.

## 2. Definición del problema

A continuación se explica la necesidad de crear un nuevo sistema de vídeo vigilancia configurable detallando los sistemas que se pueden encontrar en la actualidad.

### 2.1 Sistema Existente

El proyecto nace con la idea de mejorar un sistema existente, QDSVideo, un sistema de vídeo vigilancia que contaba con una aplicación Servidor, un cliente en C# y un WebSite en Flash. La configuración de este sistema se extraía de una base de datos cuyo contenido era generado por un DSL.



Éste fue el sistema sobre el que se partió en un principio, debido a ciertas debilidades que este presentaba, se decidió comenzar con un sistema nuevo que fuera más flexible, robusto y escalable (usando la experiencia tomada en éste último).

### 2.1.1 Tecnologías

El sistema QDSVideo utiliza las siguientes tecnologías explicadas brevemente:

- DSL: Se disponía de un pequeño editor textual que permitía la descripción del sistema. Con éste se generaba código SQL que debía ser ejecutado por el usuario en la base de datos para realizar la configuración.
- C#: Todo el código del servidor y de la aplicación estaba escrito en C#.
- PostgreSQL: Se utilizaba PostgreSQL como base de datos, conectada a través de ODBC.
- Controlador Axis: La conexión a las cámaras se realizaba a través del controlador que ofrecía Axis.
- Flex: La aplicación web estaba escrita en Flash y se conectaba directamente a la base de datos.

### 2.1.2 Análisis de debilidades

A continuación se detallan las principales debilidades que sería recomendable mejorar o sustituir ya que hicieron que no se pudiera continuar con el avance del proyecto y en su lugar se creara un sistema nuevo.

- Código poco flexible: El código de las aplicaciones C# era en general poco flexible para los requerimientos del sistema al que se quería avanzar, imposibilitando características como la agregación de funcionalidad a través de un modelo.
- Funcionalidad limitada en el servidor: La aplicación servidor permitía únicamente ver las cámaras del sistema, no era posible configurar ninguna característica en ésta ni saber qué eventos existían programados.
- Arquitectura ficticia cliente/Servidor: La arquitectura real del sistema no era totalmente cliente/servidor, no existía interacción ninguna entre el cliente y servidor, ambos se conectaban directamente a la base de datos pero no existía comunicación entre ellos.
- Dificultosa gestión de usuarios: Cuando se deseaba crear un usuario nuevo en el sistema, éste tenía que definirse también en la base de datos y en cada una de las cámaras que tuviera el sistema.
- Limitada configuración del sistema: El DSL se utilizaba para definir las cámaras existentes, grabaciones programadas, usuarios en el sistema y el envío de correos y mensajes de texto. Estas características sólo podían configurarse modificando la base de datos una vez el sistema estuviera desplegado, lo que hacía muy difícil tareas tan comunes como añadir un usuario.

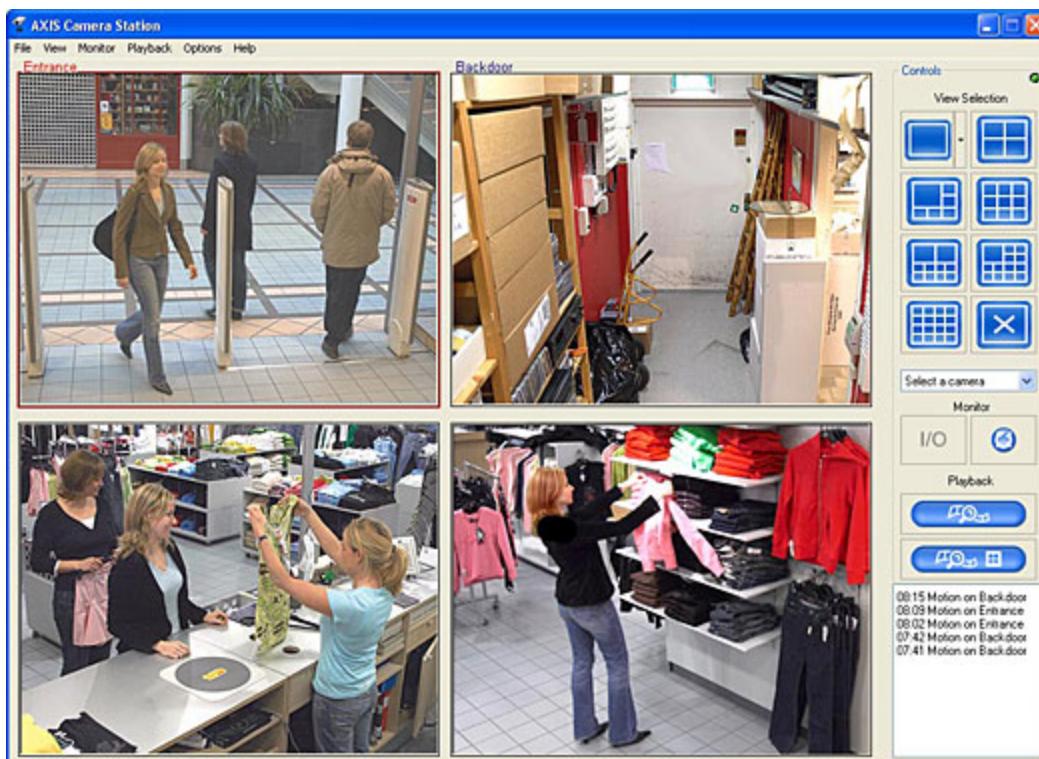
- Sistema incompatible con iOS: Debido a que el cliente estaba desarrollado en C# y el cliente web en Flash, los dispositivos Apple no tenían forma de acceder al sistema al no poder utilizar ninguna aplicación.

## 2.2 Software Axis

En la web de Axis podemos encontrar una colección de software que pretende ser utilizados para las tareas de vídeo vigilancia como los que se describen a continuación:

### 2.2.1 Axis Camera Station

Esta aplicación permite la visualización y grabación manual de vídeo de las cámaras Axis. Este software está indicado para instalaciones medianas-grandes. Puede ser recomendable si únicamente se quiere un software que te permita visualizar múltiples cámaras y realizar grabaciones a mano para cámaras Axis.



#### 2.2.1.1 Análisis de debilidades

A pesar de que este software es muy intuitivo y robusto, presenta las siguientes debilidades:

- Está totalmente sujeto a cámaras Axis y a sus servicios, por lo que no se puede utilizar ningún otro dispositivo ni usar funcionalidad que no esté incluida en Axis.
- No existe una cadena de eventos configurables por lo que el usuario se ve muy limitado a la hora de configurar cómo se debe reaccionar a los eventos que pueda producir una cámara.

- Software privativo: Al no ser código abierto no se puede realizar ninguna modificación a la aplicación por lo que no podría ampliarse en ningún momento.
- La gestión de usuarios se realiza directamente en las cámaras.

### 2.2.2 Axis Camera Companion

Esta solución, indicada para instalaciones pequeñas, se trata de una solución fácil y rápida para visualizar las cámaras que una instalación tiene. Ofrece una buena calidad de imagen(tan alta como la cámara permita) y acceso desde iPhone,Android y iPad desde aplicaciones nativas. Esta solución no proporciona una aplicación servidor sino que en todos los dispositivos que se necesite se instala un cliente que accede directamente a las cámaras.

#### 2.2.2.1 Análisis de Debilidades

- Esta totalmente sujeto a cámaras Axis y a sus servicios, por lo que no se puede utilizar ningún otro dispositivo ni usar funcionalidad que no esté incluida en Axis.
- No existe una cadena de eventos configurables por lo que el usuario se ve muy limitado a la hora de configurar cómo se debe reaccionar a los eventos que pueda producir una cámara.
- Software privativo: Al no ser código abierto no se puede realizar ninguna modificación a la aplicación por lo que no podría ampliarse en ningún momento.
- La gestión de usuarios se realiza directamente en las cámaras.
- No es un sistema que se gestione de una forma centralizada.

### 2.2.3 Conclusión

Como hemos podido ver, Axis ofrece diferentes soluciones que pueden ser interesantes para un usuario busque únicamente la visualización de las cámaras, pero ninguna de estas soluciones alcanza el nivel de funcionalidad que se espera que ofrezca el sistema a desarrollar.

## 2.3 Otro Software disponible

En la web de Axis podemos encontrar otro software de vídeo vigilancia que ofrece gestión de eventos y comunicación de éstos al usuario, pero este software viene con una funcionalidad totalmente prestablecida, obligando al usuario a utilizar un sistema que no cumple las necesidades que necesita y no es fácilmente configurable.

Además, este software tiene un proceso de instalación dificultoso y que en la mayoría de los casos debe ser realizado por un experto, puesto que cada instalación de vídeo vigilancia tiene unas necesidades diferentes y el sistema informático debe responder a ellas.

## 2.4 Especificación de requisitos

A continuación se detallan los requisitos que se plantearon para el desarrollo de este proyecto, diferenciando los requisitos mínimos y los requisitos que sería deseable alcanzar.

### Requisitos Mínimos:

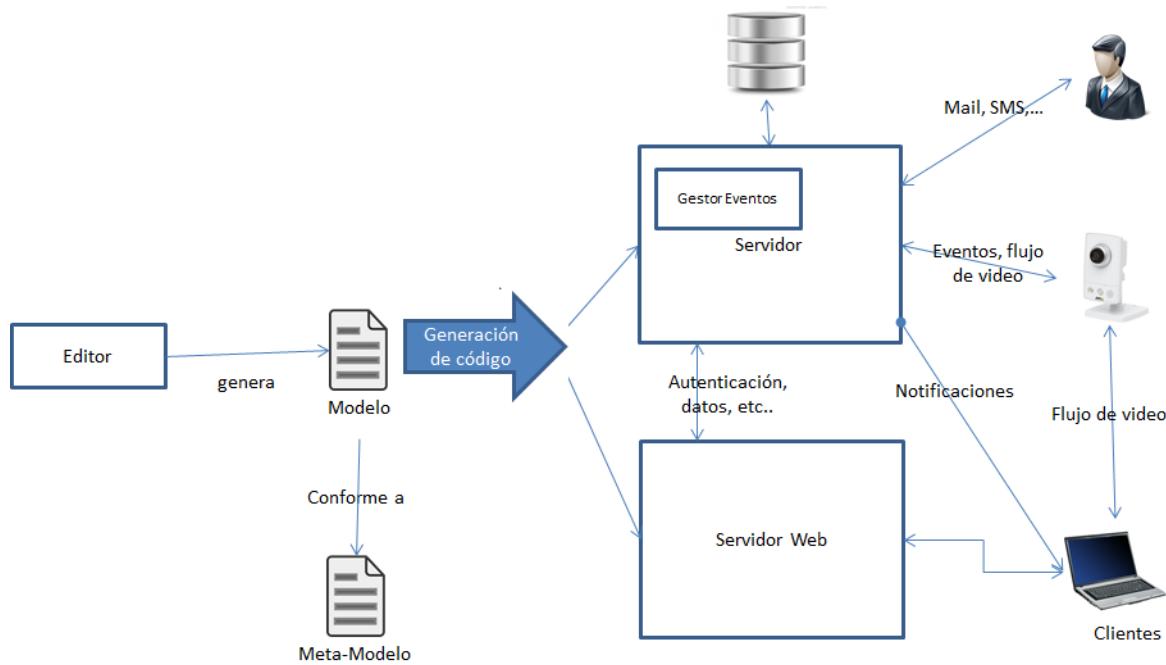
- Integración del vídeo de las cámaras en el sistema.
- Acceso al sistema a través de internet.
- Funcionalidad de respuesta a eventos configurable.
- Integración de cámaras Axis en el sistema.
- Notificación de los eventos a los usuarios.
- Garantizar la seguridad del sistema.
- Soporte para múltiples usuarios de forma simultánea.

### Requisitos Deseables:

- Posibilidad de añadir funcionalidad externa al sistema.
- Generación de clientes totalmente portables.
- Personalización del layout del sistema.
- Integración de otro tipo de dispositivos en el sistema.
- Acceso al sistema optimizado para dispositivos móviles.

### 3. Estudio de Viabilidad

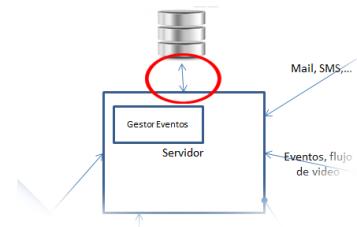
Para cumplir los requisitos establecidos y ofrecer un sistema robusto y flexible, se busca implementar la siguiente arquitectura:



Para poder construir cada parte de esta arquitectura es necesario utilizar una serie de tecnologías, cada una con sus ventajas e inconvenientes que deben ser elegidas con especial cuidado. A continuación se detalla el análisis y una breve explicación de estas tecnologías.

#### 3.1 Conexión a la base de datos

Para la conexión del servidor con la base de datos se estudiaron dos opciones principales explicadas a continuación, incluyendo finalmente ambas.



##### 3.1.1 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, por las siguientes características:

- Licencia BSD: Permiso total para utilizar el software de la base de datos.
- Soporte profesional de la comunidad y de empresas privadas.
- Muy alta fiabilidad, estabilidad y rendimiento.
- Herramientas gráficas para la creación y gestión de la base de datos.

Así, la utilización de este driver ofrece una respuesta mucho más eficiente que el driver ODBC, aunque menos flexible a cambios.

### 3.1.2 Windows ODBC

Windows Open Database Connectivity es un 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 ODBCCad. Así, en la aplicación sólo 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.

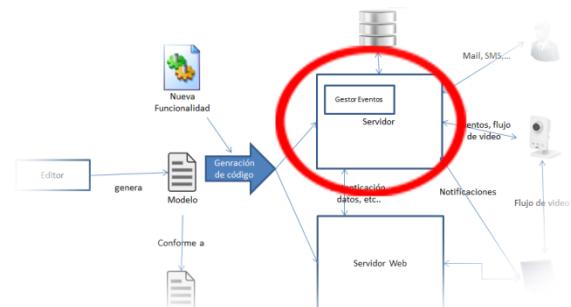
De esta forma, podremos separar totalmente la conexión de la base de datos de la aplicación, permitiendo que un cambio no sólo de ubicación de la base de datos, sino del gestor de base de datos que se utilice repercuta únicamente en cambiar una línea desde un asistente en Windows.

## 3.2 Implementación del servidor

A continuación se muestran las tecnologías, detallando sus características favorables y negativas, que se utilizan en la implementación de la aplicación servidor.

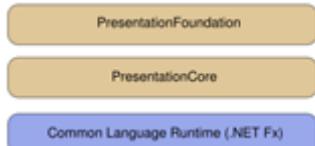
### 3.2.1 C#

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 más llevadera. La última versión de C#, 4.0, añade aún más facilidades para el programador y es que aunque 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 esté a la altura en eficiencia de C o C++[4]. Debido a esto se ha decidido 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.



### 3.2.2 WPF

Windows PresentationFoundation es una tecnología incluida en .Net 3.0 conocida anteriormente como Avalon. Ésta es una opción mas flexible y potente que Windows Forms[5] ya que permite separar totalmente la interfaz de usuario de la “lógica de negocio”, permitiendo que éstas se definan de forma separada, una característica más que deseable para nuestro proyecto, razón por la que se eligió esta tecnología de Microsoft.

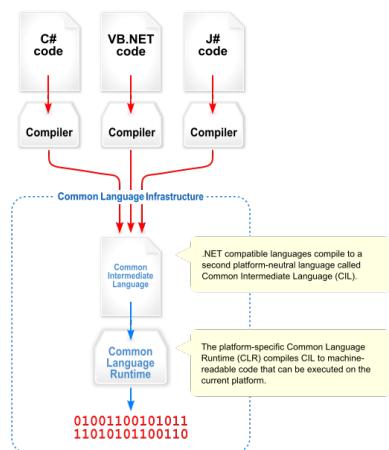


### 3.2.3 WCF

Windows ComunicationFoundation, anteriormente conocido como Indigo, es una tecnología de Microsoft que proporciona de una manera rápida y sólida 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 la utilidad WebServiceHost, el cual permite desplegar servicios web REST en una aplicación de carácter principal no web como en nuestro caso.

### 3.2.4 .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 a utilizar en un lenguaje concreto.

## 3.3 Cámaras 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, ofreciendo vídeo sobre IP, una característica muy deseable para este sistema. El acceso a las cámaras se puede realizar tanto sobre HTTP como sobre HTTPS, incrementando la seguridad.

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

Estas cámaras serán el dispositivo principal del sistema.

### 3.4 Notificación al usuario

Para realizar la notificación de eventos desde el servidor a los usuarios se han estudiado e incluido las tecnologías que se describen a continuación.



#### 3.4.1 SMS Esendex

Esendex es una empresa que proporciona un 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.



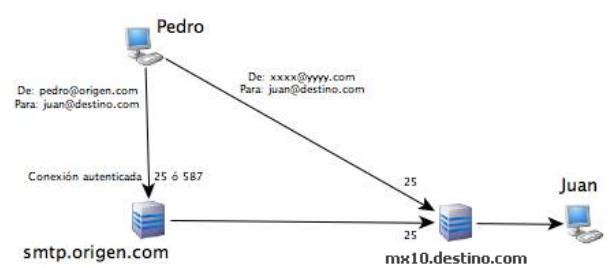
Disponen de dos modalidades de pago:

- Pre pago: 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.

#### 3.4.2 Envío de Correo

Dado que lo más común para este tipo de servicio es utilizar un servidor SMTP disponible en la web, se ofrecerá al usuario la posibilidad de utilizar uno a su elección. De esta forma, se realizará una conexión desde el servidor a implementar con el servidor de correo elegido. Hay dos puertos destinados al envío, el 25 y el 587.



Para el envío de correos desde el servidor se puede utilizar cualquier servidor de correo SMTP. El proyecto utiliza la librería System.SMTP.Mail de .Net, en concreto “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.

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

Ej: “smtp.gmail.com”

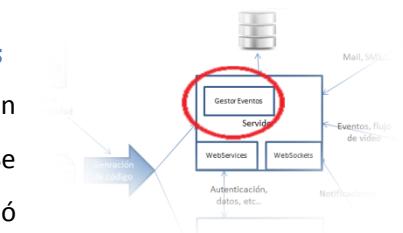
### 3.5 Cadena de Eventos

#### 3.5.1 Delegados y Sistema de notificación de eventos

Para implementar la configuración entre los servicios se buscó un sistema de notificación que fuera adecuado para el sistema. Se pensó en CORBA, Servicios Web y otras librerías, pero se terminó por utilizar el sistema asíncrono de delegados de C#. Un delegado es un tipo que hace referencia a un método. Cuando se asigna un método a un delegado, éste se comporta exactamente como el método. El método delegado se puede utilizar como cualquier otro método, con parámetros y un valor devuelto. Esto nos permite asociar varios métodos a un delegado e invocarlos 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 conectadas terminaran, provocando que si dos dispositivos lanzan un evento, éstos 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 se trata finalmente de un servicio de notificación en el cual unos consumidores se suscriben a los productores de eventos para que sean notificados cuando éstos ocurran sin bloqueo.

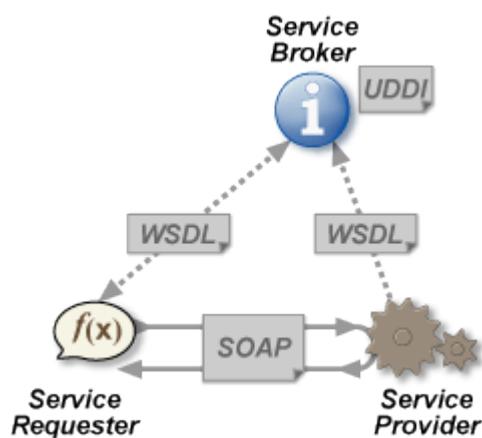


### 3.6 Comunicación Cliente-Servidor

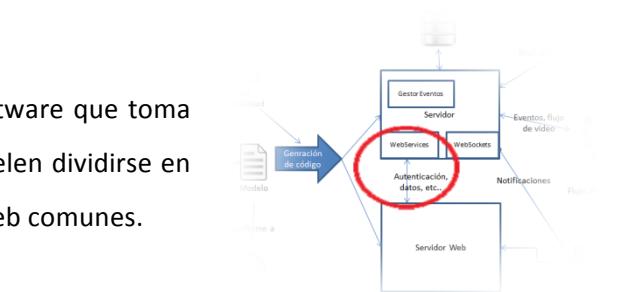
#### 3.6.1 Web Services

“Los servicios web son un componente de software que toma una entrada y produce una salida”[6] estos suelen dividirse en dos tipos, servicios web tipo REST y servicios web comunes.

Estos dos tipos responden a los siguientes esquemas:



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[7]. 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 código JavaScript. En cambio, al utilizar servicios web REST, el cliente conoce la ubicación y los métodos del proveedor de servicio, accediendo directamente a los servicios ofrecidos 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...Se ha decidido dejar al usuario la libertad de

elegir el certificado que se instalará, es por ello que éste debe ser añadido a través de IIS. Se puede obtener un certificado de entidades como VeriSign o firmar uno el mismo usuario (IIS permite la creación de un certificado de forma sencilla).

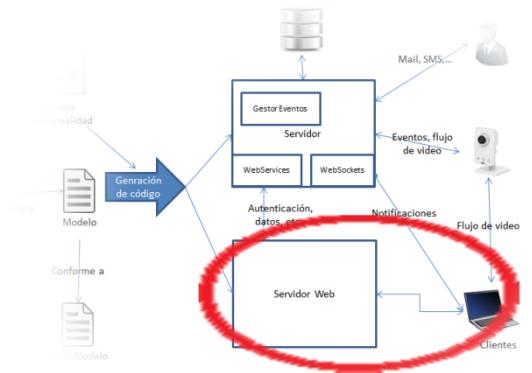
### 3.6.2 JSON

JavaScript ObjectNotation es un formato de intercambio de datos estándar independiente del lenguaje y basado en texto[8]. Este formato es muy conveniente para nuestra aplicación web que utiliza JavaScript ya que es soportado directamente por el lenguaje.

## 3.7 WebSite

### 3.7.1 IIS

Internet InformationServices 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, éste buscará automáticamente los certificados en IIS.

Es importante elegir una versión correcta de IIS (la 7 en nuestro caso), debido a que en versiones anteriores IIS fue catalogado como totalmente inseguro por dos razones:

- Vulnerabilidades que hacían al sistema fácil de atacar.
- Instalación en equipos con copias ilegales de Windows que no podían ser actualizados.

Esta situación ha cambiado totalmente en las versiones 6 y 7 ya que se ha aumentado más que considerablemente la seguridad que ofrece este software y ahora las actualizaciones de seguridad están disponibles para todos los usuarios. Así, puede considerarse IIS un software seguro, mostrando como ejemplo que la página del tesoro lo utiliza.

### 3.7.2 HTTPS y SSL

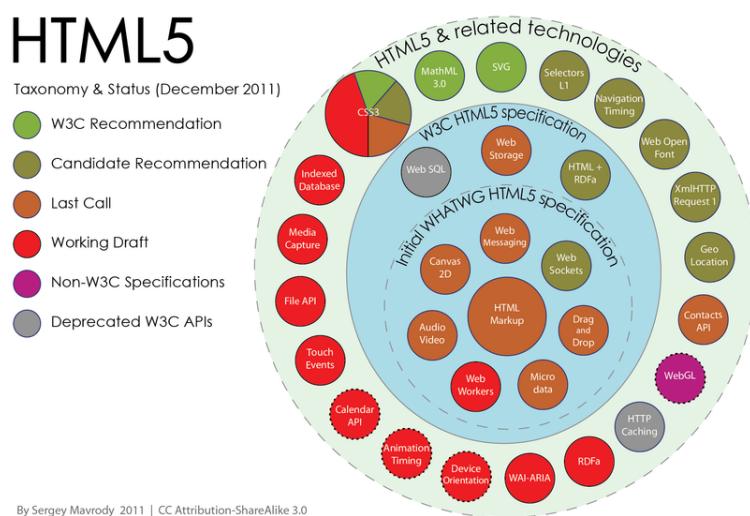
HTTP Seguro (HTTPS) es un protocolo de comunicación que se obtiene al colocar HTTP sobre SSL. La combinación de estos dos protocolos proporciona una conexión HTTP encriptado, de forma que la información intercambiada entre el servidor y el cliente no pueda ser leída por terceros.

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

### 3.7.3 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”[9] 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 aún un borrador, no todos los navegadores lo 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 sólo funcionan en algunos navegadores tal y como se detalla en el apartado de compatibilidad entre navegadores.



### 3.7.4 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 páginas web más efectivas e interactivas.”[10]

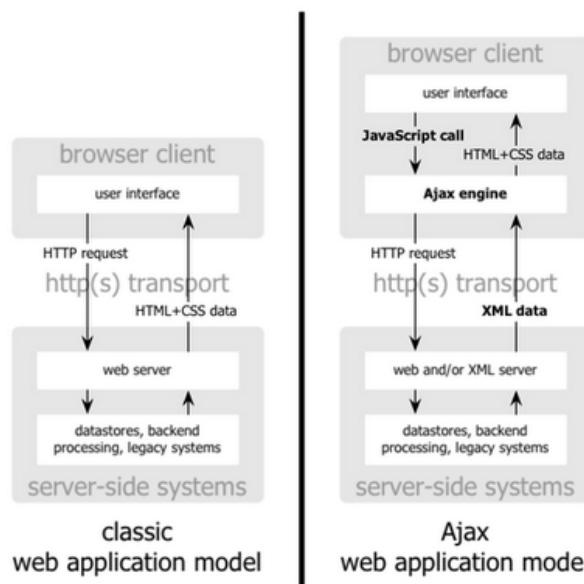
Además, para incrementar la compatibilidad entre los navegadores y realizar la codificación más agradable existen librerías como JQuery que permiten manejar JavaScript desde un nivel más alto.

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 página.

### 3.7.5 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”.[11] A pesar de que AJAX suele involucrar XML, éste 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.



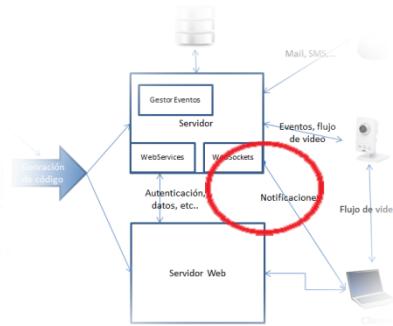
Con AJAX se añade una capa en el lado del cliente que está en comunicación con el servidor y actualiza la página dinámicamente sin necesidad de pedir al servidor que genere la página de nuevo. En nuestro caso utilizamos AJAX para comunicarnos con el servidor C# del sistema a través de los servicios web y permitir el login del usuario, obtención de los flujos de vídeo y demás operaciones que soporta éste. Sin esta tecnología no sería 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.

## 3.8 Comunicación Servidor-Cliente

### 3.8.1 Tecnologías Push

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

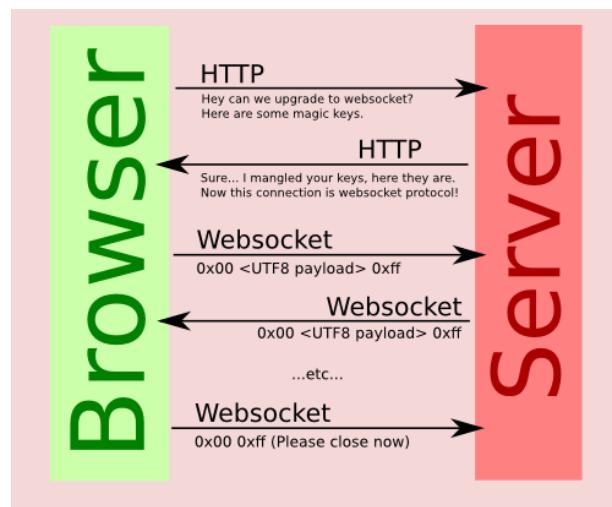


Este tipo de tecnología es fundamental para el desarrollo de aplicaciones web en tiempo real, debido a que la comunicación la inicia el servidor en el instante que ocurre un evento, en lugar de tener el cliente que consultar al servidor periódicamente produciendo un retraso inaceptable en ciertas situaciones.

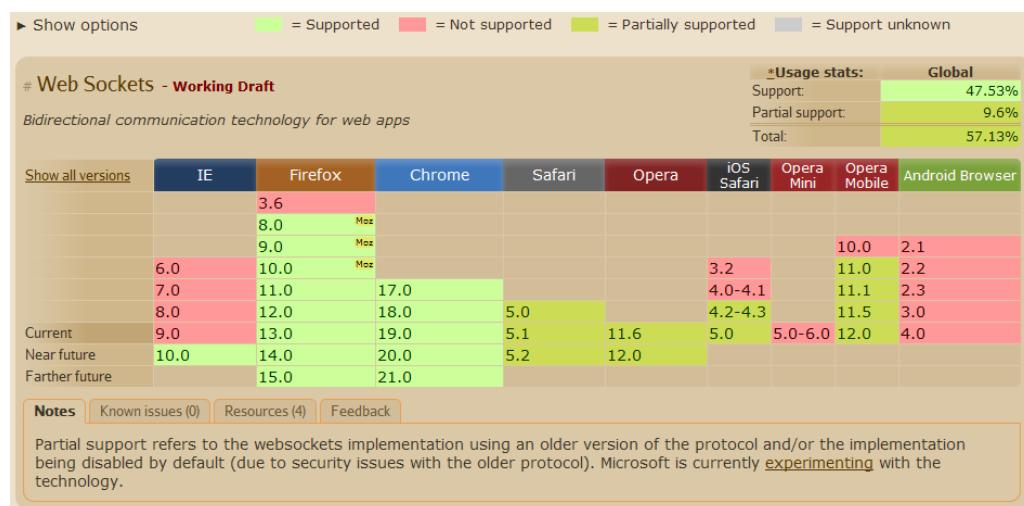
El principal problema de las tecnologías Push en la web es que la web en sí no fue diseñada para esto. En un principio, la web estaba concebida para realizar una única petición al servidor y mostrar todo el contenido, estableciendo una conexión única en la dirección cliente-servidor. Aun así, actualmente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

### 3.8.2 WebSockets

El protocolo de WebSockets posibilita una comunicación dúplex entre el servidor y el cliente[12]. Es aun una propuesta de estándar, pero todavía no está consolidado como tal. Una vez abierto un WebSocket tanto el cliente como el servidor pueden iniciar una request, lo que 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 (45% de la cuota de mercado de navegadores). De modo que si utilizáramos este protocolo, nuestra aplicación sería compatible sólo con la mitad de navegadores. Por el contrario si utilizáramos hybi-00 o hybi-10 nos encontraríamos con un soporte muy pobre (sólo Firefox y Chrome).



La implementación de esta tecnología se detalla en un anexo, aun así, merece la pena comentar las opciones que se tuvieron en cuenta a la hora de realizar la implementación en el lado servidor

- 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 [“\[http://nuget.codeplex.com/”\]\(http://nuget.codeplex.com/\)](http://nuget.codeplex.com/).

- SuperWebSocket: Aun siendo este proyecto muy interesante, basado en SuperSocket, está más 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/”](http://superwebsocket.codeplex.com/)
- Alchemy: API que utiliza la RFC6455 y que proporciona al programador todo el control sobre los usuarios conectados al WebSocket. Esta ha sido la opción utilizada. Se puede encontrar el código en “[http://alchemywebsockets.net/”](http://alchemywebsockets.net/).

Todas las APIs descritas nos permiten crear tanto servidores como clientes a través de código C#, aunque para nuestro caso en concreto sólo es necesario el lado servidor, puede ser interesante para el futuro.

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.

Se ha utilizado la librería llamada Alchemy ya que soporta diferentes protocolos (hybi00, hybi-10, hybi17 y RFC6455) y utiliza para cada cliente el que haya indicado en el handshake.

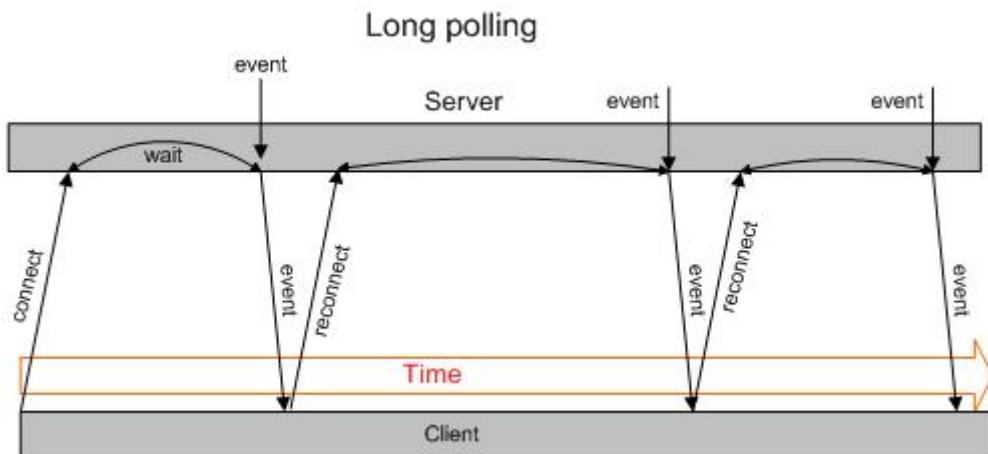
### **3.8.3 Long Polling / Comet**

Long polling, también conocido como Comet, es una emulación de la tecnología Push utilizando tecnología Pull. Long polling envía una request al servidor y éste, 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.[13]

Esto tiene grandes ventajas sobre realizar polling (encuesta cada X tiempo con respuesta inmediata)[14]:

- El mensaje desde el servidor se envía en el instante deseado.
- Ofrece información en tiempo real (debido al punto anterior).

- Produce menor carga de la red.



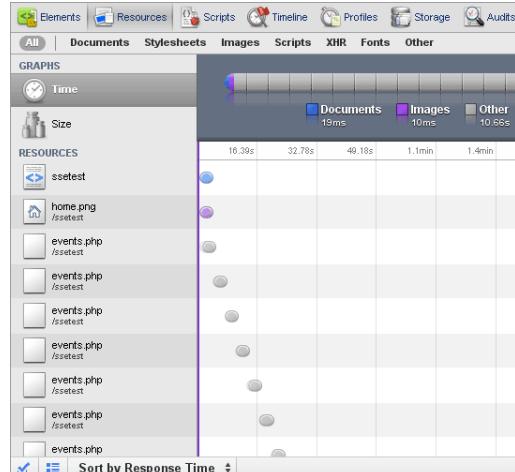
Aun así, debido a que esta tecnología no es realmente Push, sino una emulación de éste, y aunque tenga menos carga que polling sigue sin ser comparable a una tecnología como WebSockets, por ello finalmente no se implementó como principal mecanismo de comunicación desde el servidor al cliente.

### 3.8.4 Server SideEvents

Server SideEvents (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 está haciendo realmente es una especie de polling pero abstrayendo al programador de ésta. Es decir, todo el control se hace a través del API de EventSource siendo notificado el código JavaScript sólo cuando se produce un evento nuevo, pero la aplicación web está 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 ha incluido también por si se desea utilizar. Podemos encontrar un buen ejemplo de cómo implementar esta tecnología en: <http://dsheiko.com/weblog/html5-and-server-sent-events>. Finalmente no se utilizó esta tecnología debido a que no era realmente una tecnología tipo Push como tal, es una simulación de ésta a través de long-polling.



Actualmente está soportado por la mayoría de navegadores salvo Internet Explorer.



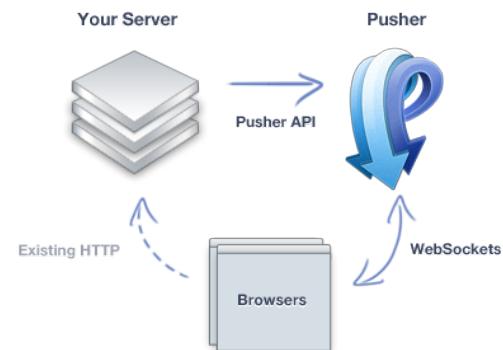
### 3.8.5 Pusher.com

Pusher.com define su producto como “Pusher es un API simple alojada 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 del API de éste 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, que proporciona cien mil mensajes diarios y hasta veinte conexiones simultáneas.

### Understanding Pusher



Sandbox	Bootstrap	Startup	Big Boy	Enterprise
FREE	\$19/month	\$49/month	\$199/month	Contact us
20 Max Connections	100 Max Connections	500 Max Connections	5,000 Max Connections	Millions Connections
100,000 Messages per day	200,000 Messages per day	1 million Messages per day	10 million Messages per day	Billions Messages per day
<del>Encryption</del> SSL Protection	<del>Encryption</del> SSL Protection	Encryption SSL Protection	Encryption SSL Protection	Premium support 24/7 Telephone

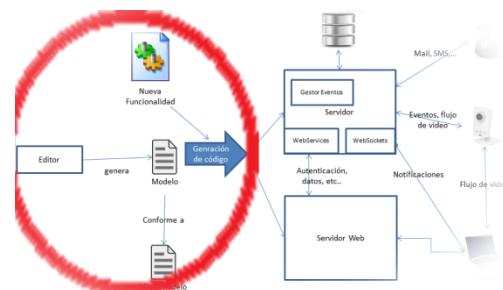
Tras ser probado, el servicio muestra una respuesta rápida y es independiente del estándar de WebSockets (puesto que de esto se encarga Pusher) siendo una solución más que aceptable para el envío de eventos desde el servidor a los clientes. Además, con vistas de futuro, Pusher permite también la comunicación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

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

En el apartado Docs de la web de Pusher podemos encontrar todo el código necesario para poner en funcionamiento Pusher.

### 3.9 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. El Desarrollo Dirigido por Modelos va a permitir que la aplicación sea más configurable y reutilizable, dado que el usuario define el sistema que se despliega en forma de modelo (logrando además una mayor comprensión de éste). Además, también afecta a la eficiencia, puesto que se conoce información sobre la instalación cuando se genera el código, generándolo así más acorde a las necesidades y menos genérico. Para ello se utilizan las tecnologías que se muestran a continuación.



### 3.9.1 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.[15]

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

### 3.9.2 GMF

GraphicalModeling Framework es un plugin de eclipse que permite la generación de editores gráficos para un meta modelo en particular. Éste 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.

### 3.9.3 OCL

ObjectConstraintLanguage es un lenguaje 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. OCL se ha convertido en un componente clave en cualquier técnica MDD como principal lenguaje para expresar querys o expresar restricciones[16].

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

Podemos encontrar información interesante sobre cómo 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>”.

### 3.9.4 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.[17]

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

### 3.9.5 JET

Java EmitterTemplates es una herramienta integrada en EMF inspirada en JSP, basado en tags y que permite realizar transformaciones modelo a texto[17]. 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 sólo permite transformaciones SIMO (Single Input Multiple Outputs), pero para nuestro objetivo, que tiene un modelo de entrada y uno de salida es más 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.

### 3.10 Clientes

Tras valorar detenidamente si era necesario un cliente nativo para PC, se decidió optar por no desarrollarlo en favor de un cliente web, que permite 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 conectarán al servidor, no descartando completamente la posibilidad de desarrollar aplicaciones nativas en el futuro para móviles o tablets.

### 3.11 Hardware Disponible

A continuación se detalla el hardware disponible para la realización y el despliegue del proyecto.

#### 3.11.1 Ordenador Servidor

Se dispone de un ordenador de sobremesa con Windows 7 que se utilizará como principal herramienta de desarrollo y como servidor del sistema desplegado.

#### 3.11.2 Cámaras Axis

Se dispone de las siguientes cámaras.

##### 3.11.2.1 Axis M5013 PTZ

Hay dos unidades de esta cámara. Dispone de movimiento horizontal y vertical. Vídeo en formato SVGA y H.264. Soporta detección de audio y movimiento.

Podemos encontrar la especificación detallada en:  
[“http://www.axis.com/files/manuals/um\\_m5013\\_42914\\_en\\_1104.pdf”.](http://www.axis.com/files/manuals/um_m5013_42914_en_1104.pdf)



##### 3.11.2.2 Axis P3344

Esta cámara es realmente interesante debido a que tiene una calidad muy alta (HDTV) y permite ver durante la noche. Una de las características más importantes es que en ella se puede instalar el

software de detección de cruce de línea (Cross Line) que puede ser utilizado en nuestro servidor para la implementación de un servicio de conteo de personas.

La especificación completa de la cámara se encuentra en  
[“http://www.axis.com/files/datasheet/ds\\_p33\\_43524\\_es\\_1111\\_lo.pdf”.](http://www.axis.com/files/datasheet/ds_p33_43524_es_1111_lo.pdf)



### [3.11.2.3 Axis M3113-VE](#)

Esta cámara está pensada para exteriores debido a su carcasa, tiene un diseño discreto y una excelente calidad de vídeo. Además, dispone de alarma de anti manipulación y es a prueba de agresiones. Dispone de detección de movimiento. La especificación completa se encuentra en “[http://www.axis.com/es/files/manuals/ig\\_m31ve\\_42270\\_es\\_1105.pdf](http://www.axis.com/es/files/manuals/ig_m31ve_42270_es_1105.pdf)”.



### [3.11.2.4 Axis 212 PTZ-V](#)

Esta cámara dispone de movimiento vertical y horizontal además de zoom instantáneo, no tiene piezas móviles.

La especificación completa se puede encontrar en  
[“http://www.axis.com/files/datasheet/ds\\_212ptz-v\\_34054\\_es\\_0812\\_lo.pdf”.](http://www.axis.com/files/datasheet/ds_212ptz-v_34054_es_0812_lo.pdf)



### [3.11.2.5 Axis 221](#)

Esta cámara es interesante debido a que ofrece función día/noche con filtro de infrarrojos. Dispone de detección de movimiento multi-ventana y de salidas/entrada binarias físicas. La especificación completa de esta cámara se encuentra en el siguiente enlace de la web de axis  
[“http://www.axis.com/es/files/datasheet/ds\\_221\\_35210\\_es\\_0904\\_lo.pdf”.](http://www.axis.com/es/files/datasheet/ds_221_35210_es_0904_lo.pdf)



### [3.11.2.6 Axis 211 W](#)

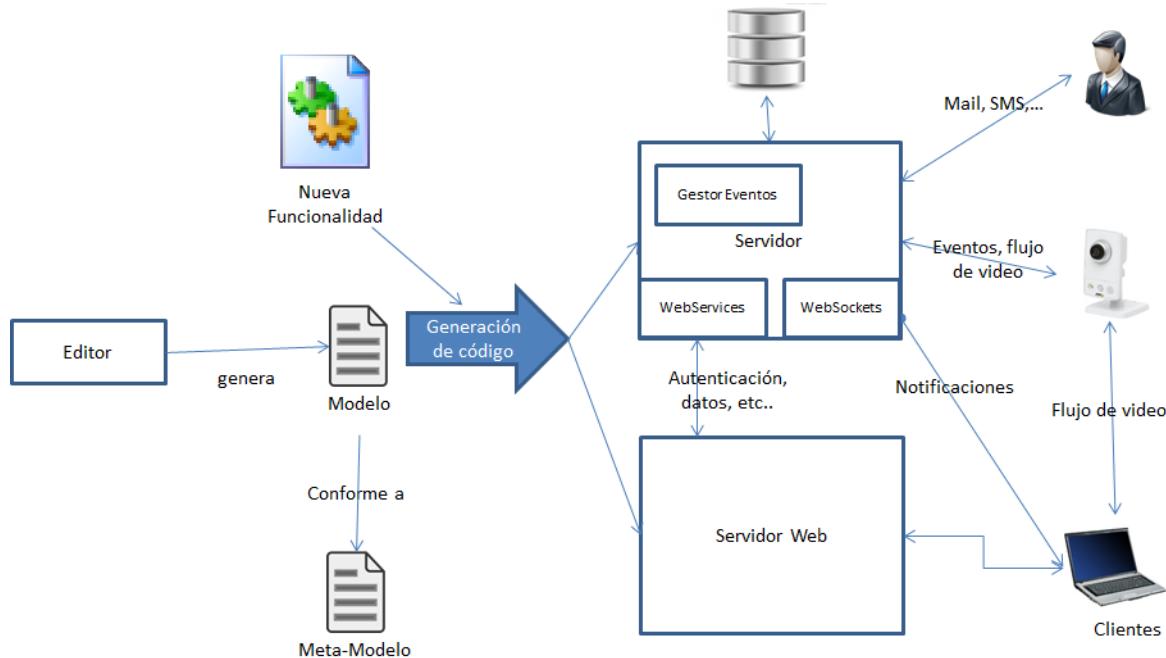
Este es un producto descontinuado, el de sustitución es la cámara M1011 W. Tiene una excelente calidad de imagen. Se puede adquirir un paquete adicional para instalar la cámara en exteriores. Dispone de detección de movimiento y una calidad de vídeo media-alta. La



especificación completa de la cámara se encuentra en  
[“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)

## 4. Análisis y Diseño

### 4.1 Arquitectura



#### 4.1.1 Aplicación Servidor

Ésta pretende ser una aplicación servidor que, por una parte, controle y procese los eventos recibidos según se haya indicado en el DSL que generará 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.

Ha sido implementada en la plataforma .Net, en concreto, en el lenguaje C#. 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 perfectamente los requisitos de la interfaz de usuario y de comunicación 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 RESTfullWebServices debido a que presentan una mayor facilidad de integración para los clientes al ser expuestos como URLs.

Se decidió utilizar una base de datos para almacenar los usuarios, la configuración de algunos parámetros y el log. Se estudió detalladamente qué parámetros deben establecerse en el DSL y cuáles 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, además de que esto haría que el DSL no fuera consistente con el sistema.

#### 4.1.2 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 sólo será necesario modificar la configuración del servidor, que será expuesta a través de los servicios web, siendo mostrada en el cliente web. Una vez el cliente recupere la información del servidor, éste podrá acceder directamente al flujo de vídeo de las cámaras.

Se ha decidido desarrollar el WebSite en HTML5 y JavaScript debido a ciertas opciones interesantes que ofrece HTML5 como Server SideEvents o la posibilidad de integrar vídeo que, aunque no se hayan implementado actualmente, son características importantes para el proyecto de cara al futuro. Otro punto de interés que decantó 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.

#### 4.1.3 Modelado y generación de código

A través de MDT se ha creado un conjunto de editores que permiten generar un modelo conforme a un meta modelo para definir la instalación y su configuración. Éste modelo se puede generar a través del editor en árbol, el editor gráfico o el editor textual. Una vez generado y verificado el modelo, puede realizarse la transformación JET modelo a texto que generará el WebSite y el servidor C#. Este paso de modelado y generación de código permite que el sistema se ajuste totalmente a las necesidades del usuario, permitiendo definir, conectar y añadir servicios como él deseé.

### 4.2 Análisis y diseño del gestor de eventos

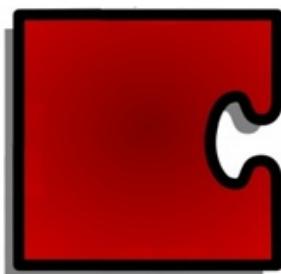
A continuación se expone el análisis y diseño del gestor de eventos, una parte del servidor C# que por su importancia se trata aparte.

#### 4.2.1 Visión general

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

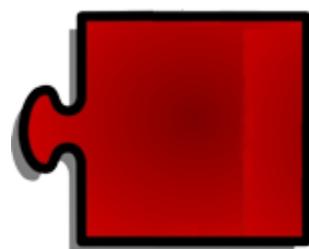
El gestor de eventos pretende ser una especie de cadena de bloques conectables, configurados al generar el código que responde a los eventos de una manera predeterminada. Además, se pretende definir una interfaz específica de forma que se puedan generar nuevas piezas y añadirlas a la cadena con el mínimo 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.

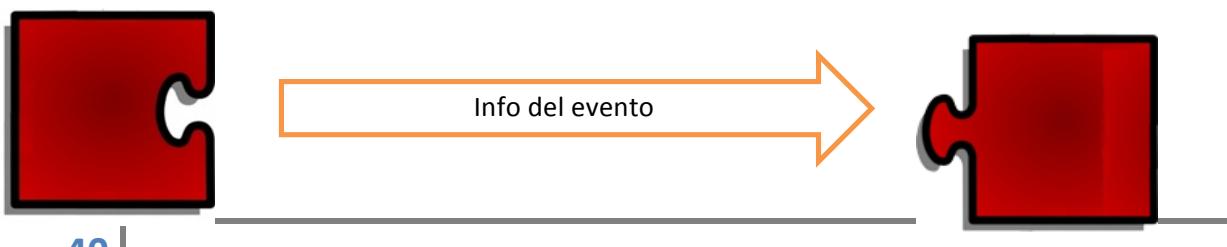


Los generadores de eventos estarían provistos de una interfaz a través de la cual, otras piezas, consumidores de eventos podrían conectarse. Se pueden conectar cuantas piezas se deseen, las que serán notificadas cuando se reciba un evento en ésta.

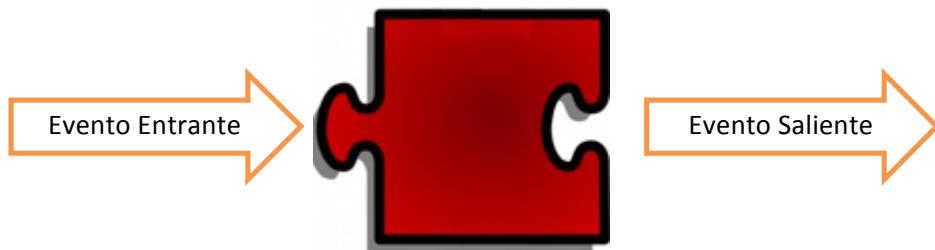
Las otras piezas también deben cumplir una interfaz específica, una para poder ser añadidas a las productoras de eventos. Estas piezas pueden ser conectadas a varios generadores de eventos, de forma que sean activadas en diferentes ocasiones.



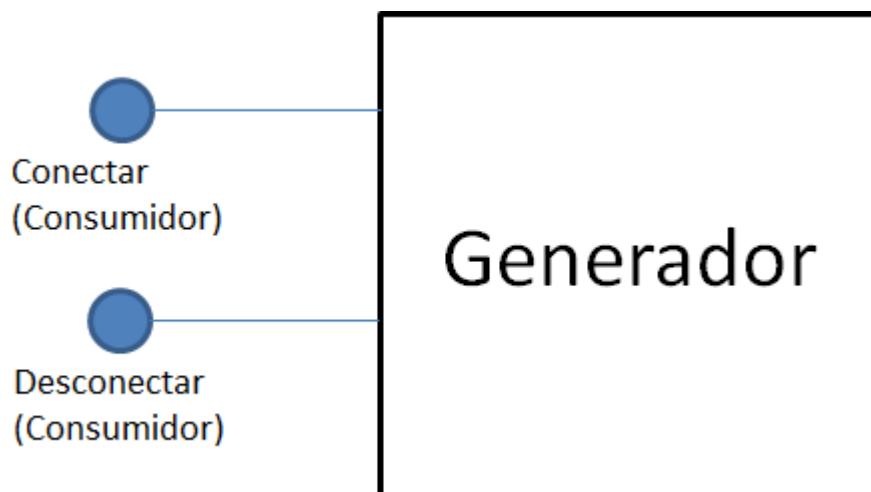
Cuando la primera pieza produce un evento (interno al sistema), envía a todas las piezas conectadas un documento con los datos del evento realizado.



Además, de la combinación de estas dos piezas, aparece una nueva pieza cuya función es transmitir el evento procesándolo primero. Estas piezas, reciben un evento interno al sistema y producen otros. Por tanto permitirán la creación de una cadena de gestión de eventos que irá refinando y tratando cada situación hasta que sea gestionada debidamente

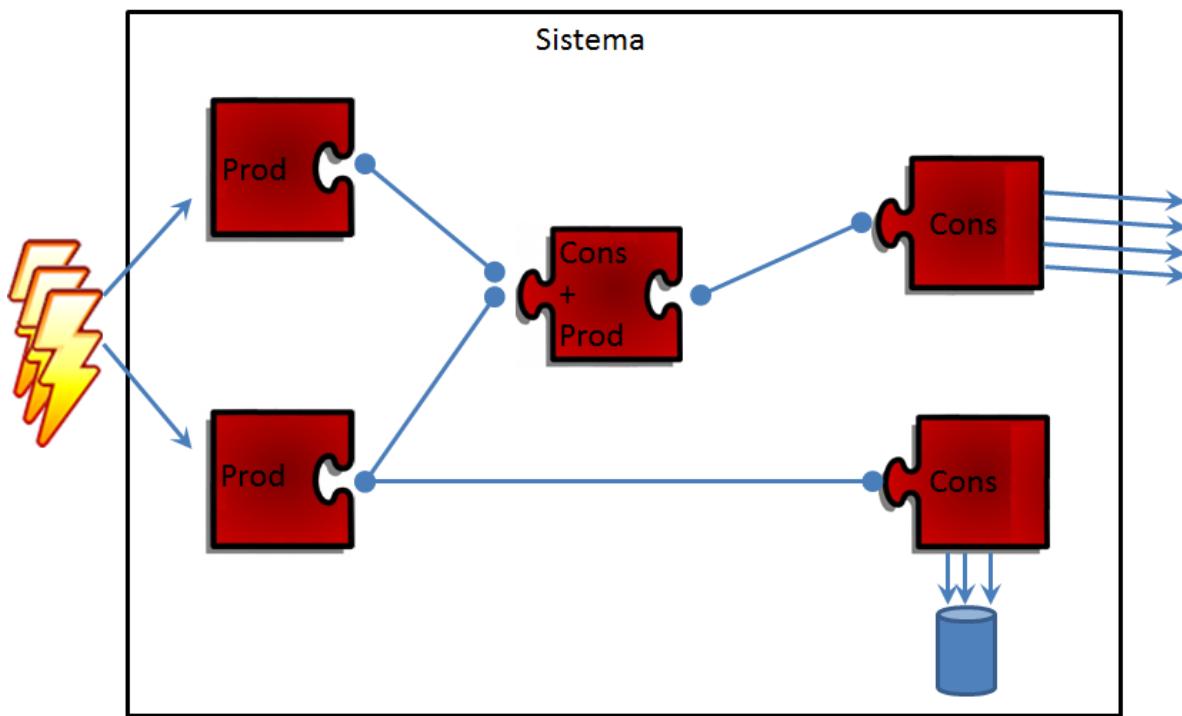


De esta forma, el interfaz de los productores queda definido de la siguiente forma:



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

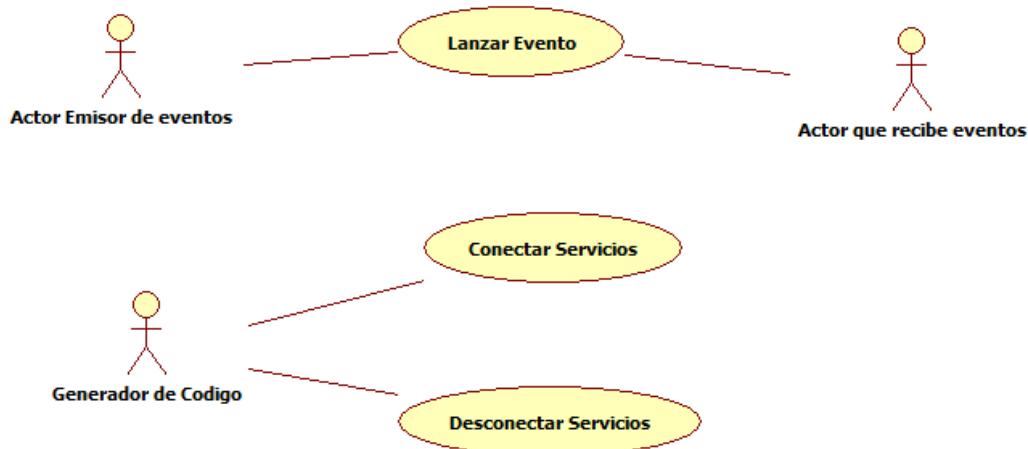
#### 4.2.2 Ejemplo



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 enviarán una notificación a los consumidores acoplados. Algunos de estos productores serán “piezas intermedias” para comprobar por ejemplo qué cámara ha lanzado el evento y continuar la cadena si es necesario, mientras que los consumidores finales normalmente generarán 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.

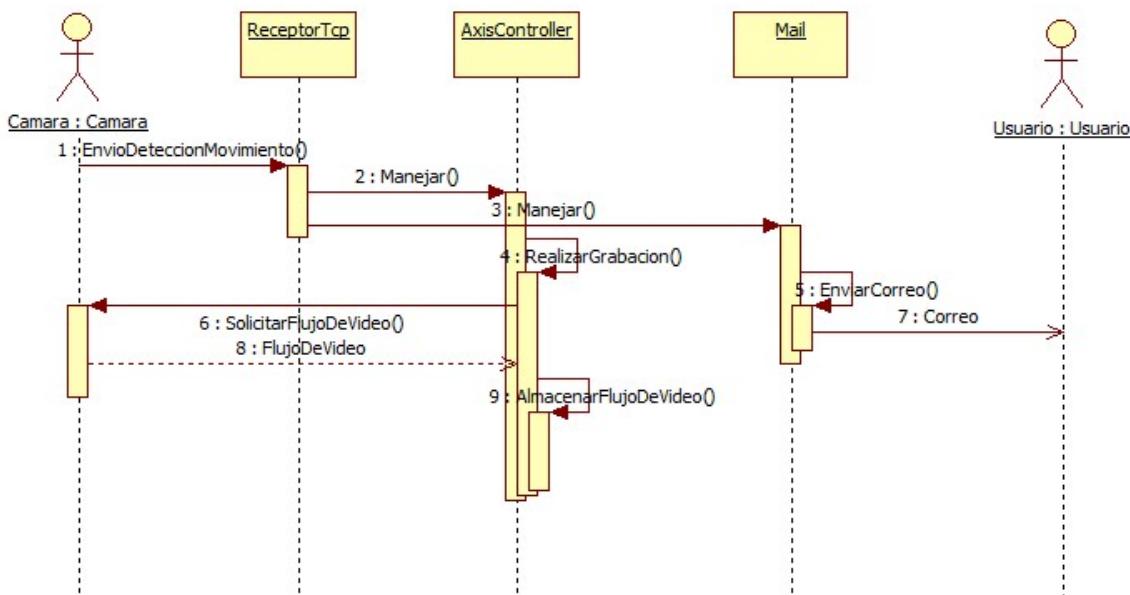
#### 4.2.3 Casos de uso



Esta imagen recoge las opciones que tiene el usuario en el gestor de eventos mostrando las operaciones que se pueden realizar con los servicios de forma general.

#### 4.2.4 Diagrama de Secuencia

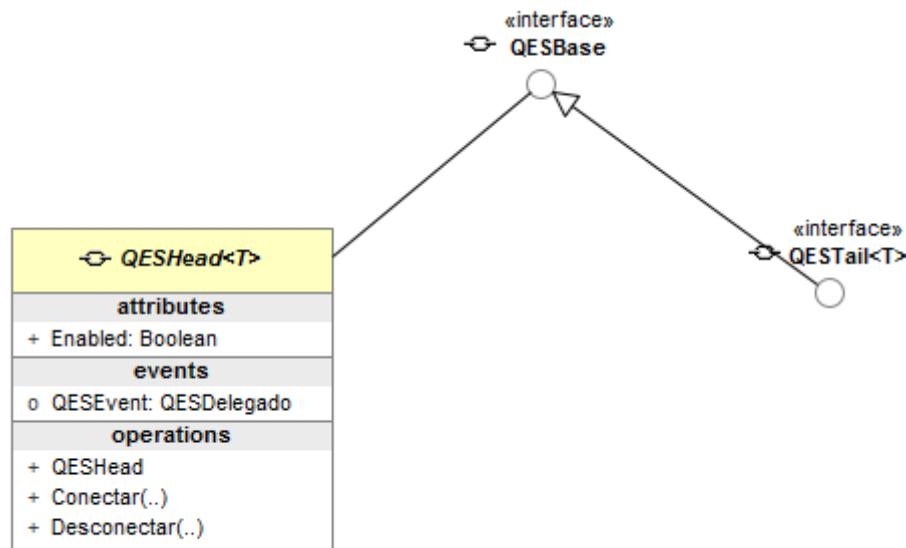
A continuación se expone un diagrama de secuencia, en éste los servicios se encuentran configurados para el control de eventos de detección de movimiento, respondiendo con el envío de un correo y la realización de una grabación temporizada.



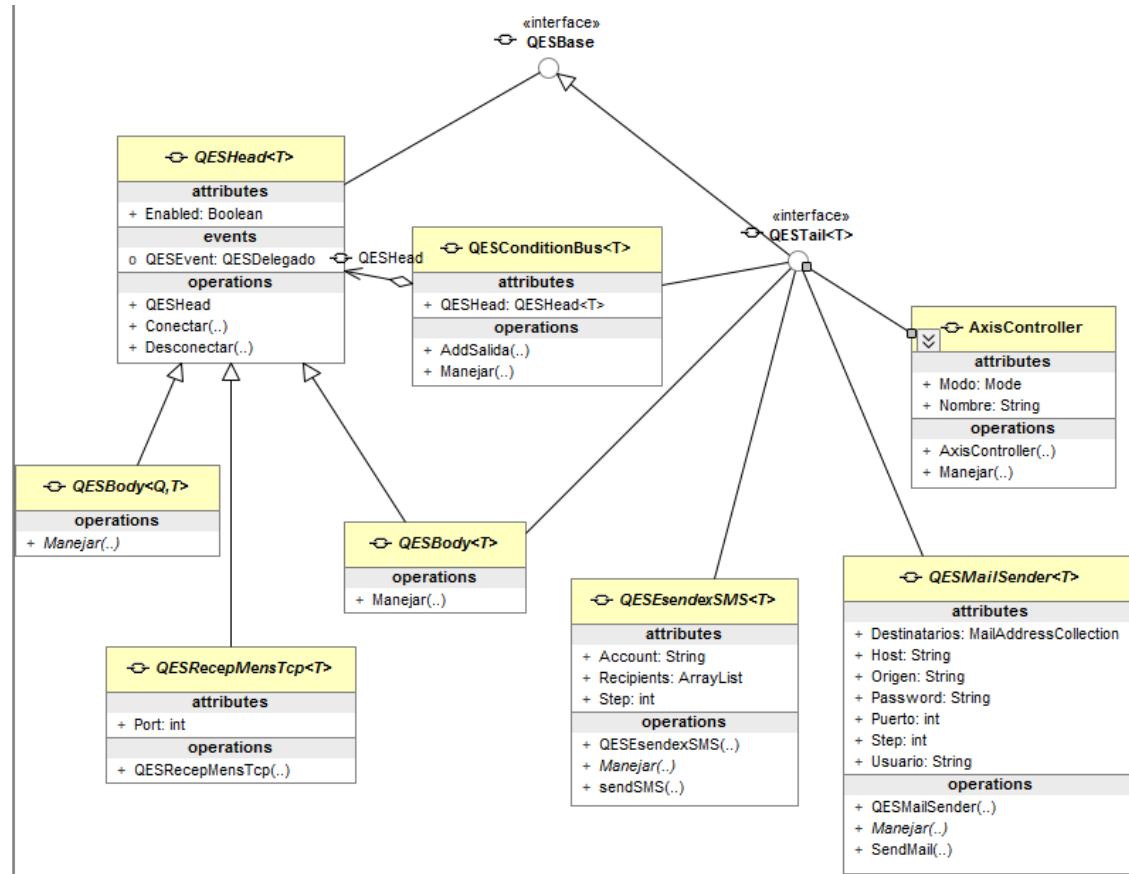
#### 4.2.5 Diagrama de clases

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 manejará el evento lanzado.

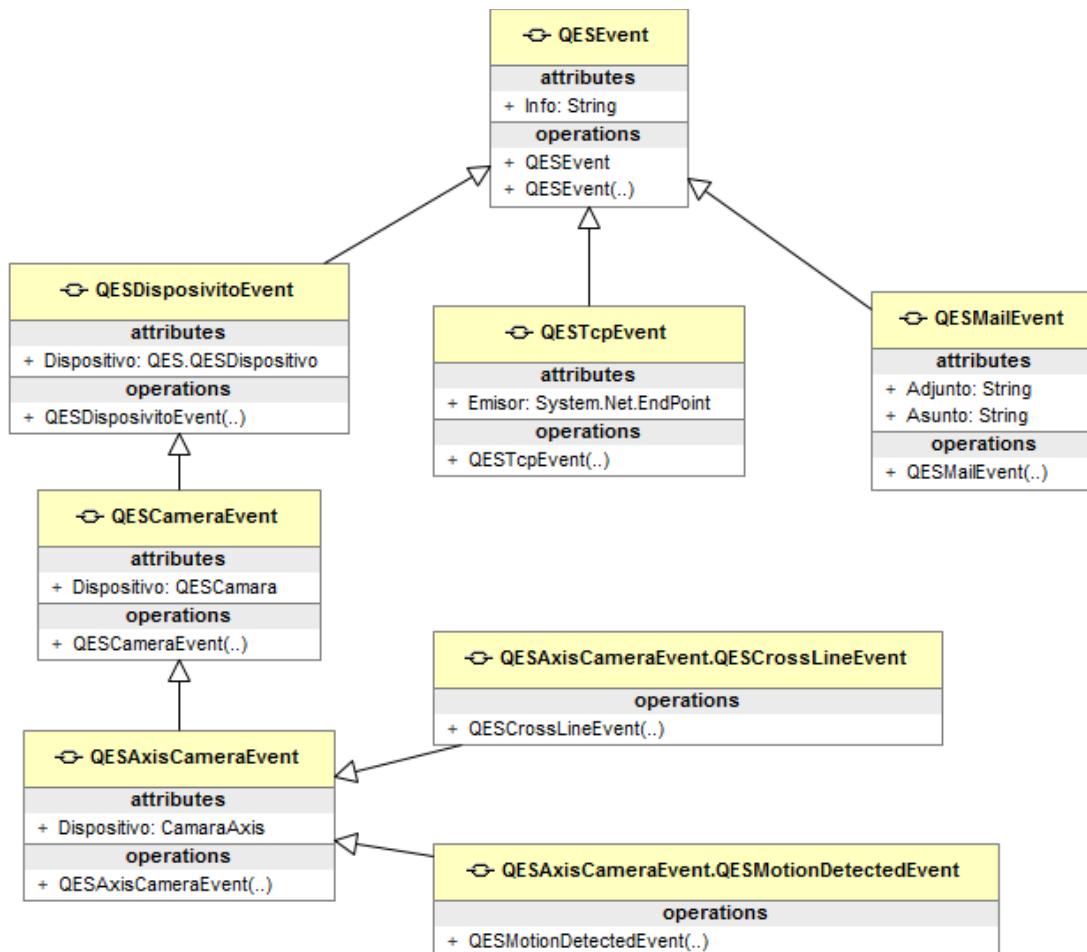


Con esta base, se ha implementado el siguiente modelo de servicios:



#### 4.2.6 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. En esta se encuentra la base QESEvent de la que heredan diferentes tipos para los diferentes dispositivos, un tipo para eventos TCP comunes y otro con información sobre correos.

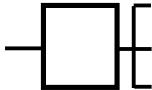
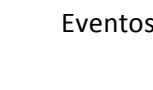


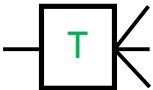
#### 4.2.7 Servicios Disponibles

En esta sección se explica brevemente cada uno de los servicios que se han diseñado. Estos servicios son las piezas que componen el gestor de eventos, el cual se encargará del tratamiento de los eventos en el servidor.

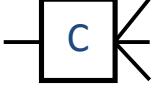
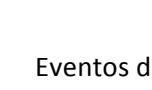
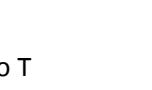
Se expone para cada servicio:

- Nombre
- Entradas
- Salidas
- Descripción

<i>QESBody&lt;T&gt;</i>	Abstracta
	<b>Entrada</b> Eventos del tipo T
	<b>Salida</b> Eventos del tipo T
	<b>Función</b> Pretende ser heredada para tratar/procesar eventos o distribuirlos.

<i>QUESTypeBus&lt;T&gt;</i>	
	<b>Entrada</b> Eventos del tipo T
	<b>Salida*</b> Eventos del tipo R(subtipos de T)
	<b>Función</b> Un evento entra y toma aquellas salidas que coincidan con su subtipo. De esta forma se puede redireccionar los eventos en función de su subtipo.

\*Hay tantas salidas como tipos se hayan añadido.

<i>QESConditionBus&lt;T&gt;</i>	
	<b>Entrada</b> Eventos del tipo T
	<b>Salida*</b> Eventos del tipo T
	<b>Función</b> 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.

<i>QESHead&lt;T&gt;</i>		Abstracta
	<b>Salida</b>	Eventos del tipo T
	<b>Función</b>	Clase a heredar para crear productores de eventos

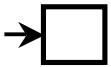
<i>QESRecepMensTcp&lt;T&gt;</i>		Abstracta
	<b>Salida</b>	Eventos del tipo T
	<b>Función</b>	Ser heredada para implementar el comportamiento al recibir un mensaje TCP.

<i>QESRecepMensTcp</i>		
	<b>Salida</b>	Eventos del tipo QESEvent (Subtipo QUESTcpEvent o QESDispositivoEvent Correspondiente)
	<b>Función</b>	Recoge mensajes Tcp y los transforma en eventos.

<i>QUESTimer&lt;T&gt;</i>		
	<b>Salida</b>	Eventos del tipo T
	<b>Función</b>	Se indica una fecha y una acción que produce un evento. Cuando se alcance la fecha, el evento resultante de la

acción se dispara

<i>QESAxisRecordingTimer</i>	
	<b>Salida</b> Eventos del tipo QESEvent(Subtipo QESAxisCameraEvent)
	<b>Función</b> Al cumplirse una fecha, se envía un evento de grabación, cámara + duración en campo info.

<i>QUESTail&lt;T&gt;</i>	Interfaz
	<b>Entrada</b> Eventos del tipo T
	<b>Función</b> Interfaz a implementar para definir un consumidor de eventos.

<i>AxisController</i>	
	<b>Entrada</b> Eventos del tipo QESEvent
	<b>Función</b> Sin importar el contenido, cuando recibe un evento, realiza una grabación con los parámetros predefinidos.

<i>GenericAxisController</i>	
	<b>Entrada</b> Eventos del tipo QESEvent (Subtipo QUESTcpEvent o QESAxisCameraEvent)

<b>Función</b>	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.
----------------	---

<i>QSEEsендexSMS&lt;T&gt;</i>	Abstracta
	<b>Entrada</b> Eventos del tipo T
	<b>Función</b> Clase a ser heredada para definir el como enviar un SMS desde la API de Esendex al recibir un evento.

<i>QSEEsендexSMS</i>	
	<b>Entrada</b> Eventos del tipo QESEvent
	<b>Función</b> Envía un SMS cuyo contenido es el atributo Info del evento recibido.

<i>QESMailSender&lt;T&gt;</i>	Abstracta
	<b>Entrada</b> Eventos del tipo T
	<b>Función</b> Clase a ser heredada para definir el como enviar un mail al recibir un evento.

<i>QESMailSender</i>	
<b>Entrada</b>	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.
→ 	<b>Función</b> 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.

<i>Pusher</i>	
<b>Entrada</b>	Eventos del tipo QESEvent (Subtipo QESTcpEvent o QESAxisCameraEvent)
→ 	<b>Función</b> Obteniendo la cámara del evento, envía una notificación a través de Pusher.com para los clientes web.

<i>QESWebSocket</i>	
<b>Entrada</b>	Eventos del tipo QESEvent (Subtipo QESTcpEvent o QESAxisCameraEvent)
→ 	<b>Función</b> Obteniendo la cámara del evento, envía una notificación a través de un WebSocket para los clientes web.

### 4.3 Análisis y diseño del 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:

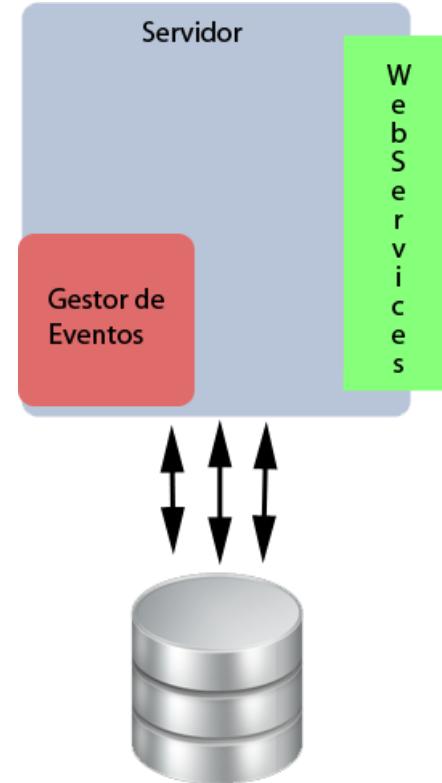
- Comunicación con la base de datos.
- Gestión de la configuración del sistema.
- Gestión general de cámaras, recintos y dispositivos.
- Control y tratamiento de eventos.
- Servicios para otras aplicaciones.

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 éste.

#### 4.3.1 Visión General

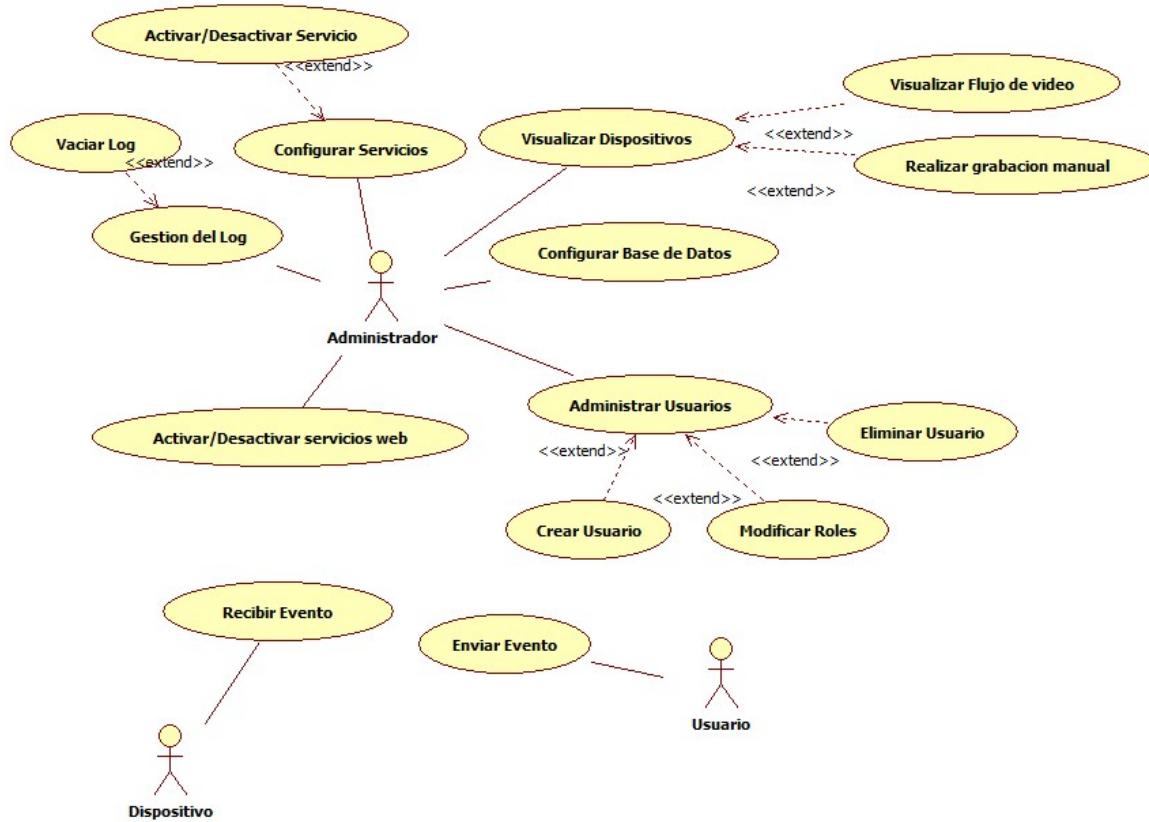
El servidor será la aplicación principal del sistema, comunicándose con la base de datos, conteniendo la gestión de los eventos y publicando la información necesaria a través de servicios web. La aplicación está orientada para ser utilizada por el administrador del sistema, ofreciendo diferentes opciones de configuración y visionado de los distintos servicios y dispositivos incluidos. Se presenta también el panel de usuarios para controlar el acceso que éstos 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 a la identificación 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 en el manual de usuario, para que éste funcione correctamente.

### 4.3.2 Diagramas de Casos de Uso

A continuación se expone el conjunto de acciones que recaen en los diferentes usuarios del sistema a través del diagrama de casos de uso.



### 4.3.3 Servicios Web

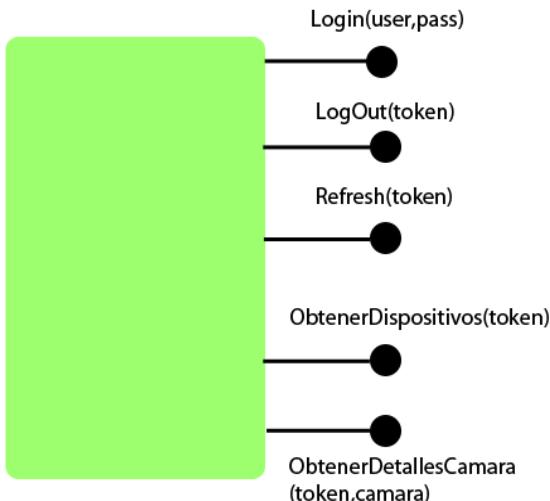
Se han implementado servicios web RESTful, con el objetivo de facilitar la interacción desde las aplicaciones al no tener que enviar el WSDL y poder mandar directamente JSON. Debido a que la información que se envía a través de los servicios web es de tipo sensible (usuarios, contraseñas, direcciones 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 Communication Foundation) en especial con WebServiceHost, debido a que la aplicación desarrollada no es una aplicación web, es una aplicación de escritorio, lo que 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. De esta forma se reduce drásticamente 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, será 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 desconecta del 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 qué roles tiene un usuario.
- `DesloguearA`: Termina la sesión de un usuario determinado.

Algunos servicios devuelven un tipo de dato específico, para entender estos tipos, a continuación se describen en código C#.

```
enumCODE {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	<ul style="list-style-type: none"> <li>user: Usuario</li> <li>password: Contraseña</li> </ul>	LoginReturn	Autentica a un usuario contra el sistema.	Se permiten varias sesiones por usuario
LogOut	POST	<ul style="list-style-type: none"> <li>token: Token de autenticación</li> </ul>		Cierra la sesión de un usuario.	
Refresh	POST	<ul style="list-style-type: none"> <li>token: Token de autenticación</li> </ul>		Renueva la sesión de un usuario.	
Get Dispositivos	GET	<ul style="list-style-type: none"> <li>token: Token de autenticación</li> </ul>	Dispositivos Return	Obtiene un listado de los dispositivos.	Sólo se obtienen los que usuario tiene privilegios para ver.
Get Detalles Camara	GET	<ul style="list-style-type: none"> <li>token: Token de autenticación.</li> <li>Cámara: Cámara de la que se solicitan los datos</li> </ul>	CámaraDetalleReturn	Obtiene los detalles para poder visualizar el stream de una cámara	

GetUsuarios	GET	<ul style="list-style-type: none"> <li>token: Token de autenticación.</li> </ul>	Usuarios Return	Obtiene un listado de los usuarios junto con su estado.	[ADMIN]
Crear Usuario	POST	<ul style="list-style-type: none"> <li>token: Token de autenticación</li> <li>name: Nombre del nuevo usuario</li> <li>password: Contraseña del usuario</li> </ul>	Código de confirmación (CODE)	Registra a un nuevo usuario en el sistema.	Dos usuarios no pueden tener el mismo nombre. [ADMIN]
UpdatePassword	POST	<ul style="list-style-type: none"> <li>token: Token de autenticación.</li> <li>Old: Contraseña actual</li> <li>Password: Nueva contraseña</li> </ul>	Código de confirmación (CODE)	Cambia la contraseña del usuario.	El usuario se obtiene a través del token de autenticación.
Get Servicios	GET	<ul style="list-style-type: none"> <li>Token: Token de autenticación.</li> </ul>	Servicios Return	Obtiene un listado de los servicios en el sistema y su estado actual	[ADMIN]
Set Servicio	POST	<ul style="list-style-type: none"> <li>Token: Token de autenticación.</li> <li>Servicio: Nombre del servicio.</li> <li>Status: True = activo False = inactivo</li> </ul>	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]
GetPrivilegios	GET	<ul style="list-style-type: none"> <li>Token: Token de autenticación.</li> <li>User: Usuario a</li> </ul>	Privilegios Return	Devuelve una lista con todos los roles y un	[ADMIN]

		consultar		campo que indica si el usuario lo posee.	
Set Privilegios	POST	<ul style="list-style-type: none"> <li>• Token: Token de autenticación.</li> <li>• User: Usuario</li> <li>• Priv: Rol</li> <li>• Value: Asignado/ No</li> </ul>	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	<ul style="list-style-type: none"> <li>• Token: Token de autenticación.</li> <li>• Usuario: Usuario a expulsar</li> </ul>	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.

#### 4.3.4 Base de datos

El sistema necesita acceso a una base de datos cuya configuración se detalla en el siguiente punto.

La base de datos es de extrema simplicidad, utilizándose únicamente para hacer persistentes datos como usuarios, log del sistema y los parámetros configurables de los servicios. Tal y como se ha dicho con anterioridad, no es necesario almacenar más información puesto que ésta está presente en el DSL que define el sistema y por lo tanto está almacenada a nivel de código. Además, esta información a nivel de código no debe ser modificada por el usuario (y por ello no se permite) ya que haría al DSL inconsistente con el sistema.

Las tablas de la base de datos son creadas y administradas por el sistema siguiendo el siguiente diagrama:

Users	Params	Log
<pre>id serial name varchar(20) superrole varchar(100) pass varchar(50)</pre>	<pre>id serial componente varchar(20) parametro varchar(20) valor varchar(100)</pre>	<pre>id: serial fecha: date hora: time content: varchar(200)</pre>

#### 4.3.4.1 Log

La tabla log es una tabla cuyo propósito consiste en almacenar cada una de las entradas que se han producido en el log del sistema. Junto con el contenido se guarda la fecha, la hora y un identificador de la entrada.

Atributo	Tipo	Descripción
id	Autoincrement	Clave primaria
fecha	date	Día de la entrada
hora	time	Hora de la entrada
content	Varchar(200)	Contenido de la entrada

#### 4.3.4.2 Params

Params es una tabla que guarda cada uno de los parámetros configurables de los servicios que tiene el sistema.

Atributo	Tipo	Descripción
id	Autoincrement	Clave primaria
Componente	Varchar(20)	Servicio que contiene el parámetro
parametro	Varchar(20)	Parámetro que se configura
content	Varchar(200)	Valor que debe tener el parámetro.

#### 4.3.4.3 Users

En esta tabla se almacenan todos los usuarios con la contraseña y los roles de los que hereda.

Atributo	Tipo	Descripción

id	Autoincrement	Clave primaria
name	Varchar(20)	Nombre del usuario
superrole	Varchar(100)	Lista de roles que contiene
pass	Varchar(50)	Contraseña en MD5

#### 4.3.5 Conexión a la base de datos

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 más eficiente que la segunda, pero sólo 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.

##### 4.3.5.1 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.

#### 4.3.6 Gestión de contraseñas

Como sistema de video vigilancia el cuidado de la seguridad ha pasado por un riguroso análisis, finalizando con el siguiente modelo de gestión de contraseñas.

Las contraseñas se almacenan en MD5 (tanto en la base de datos como durante la ejecución en memoria) para incrementar la seguridad del sistema. Esto hace que incluso si la base de datos fuera atacada con éxito y un tercero obtuviera acceso a la información almacenada en ésta le sería igualmente imposible entrar al servidor. A pesar de que se hayan encontrado colisiones en MD5 y esto debido a la paradoja



del cumpleaños pueda suponer un riesgo sobre la seguridad de este método de cifrado[18], la comunidad responde que “MD5 no se ha vuelto de repente inseguro”[19]

A pesar de que con MD5 las contraseñas son prácticamente seguras a ataques de fuerza bruta hay un agujero de seguridad importante, las contraseñas “fáciles”. Se trata de las contraseñas que se repiten frecuentemente y que, por tanto, se encuentran en diccionarios que hacen fácil el recuperado de estas, y es que el 40% de las contraseñas aparecen en el top 100 y el 71% en el top 500[20]. Para solucionar este problema se utiliza un mecanismo denominado SALT. Lo que se hace es que a la contraseña introducida por el usuario se añaden una serie de caracteres que las hacen invulnerables a ataques de diccionario[21].

#### 4.3.7 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 él se almacena:

- Configuración de la conexión de la base de datos.
- Tiempo tras el que un usuario será desconectado en el sistema.

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

#### 4.3.8 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 los controle. Así, se definen una serie de roles

básicos, un administrador, un rol para cada dispositivo y un rol para todos los dispositivos. Con esta base, el administrador puede crear nuevos roles de la conjunción de los existentes.

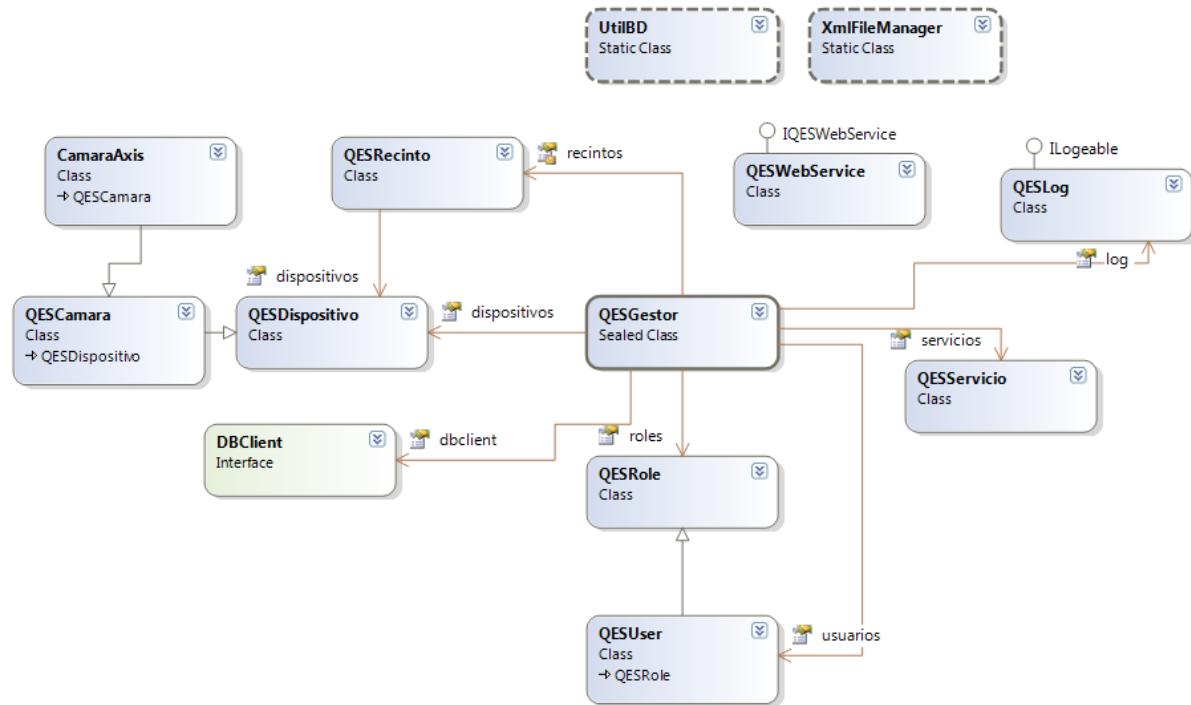


Además, un rol puede tener un conjunto de roles de los que hereda todas sus propiedades (como administrador que hereda todos o “Viewer” que hereda el de todos los dispositivos). De forma que si un usuario posee el rol Administrador también poseerá el rol de una cámara específica.

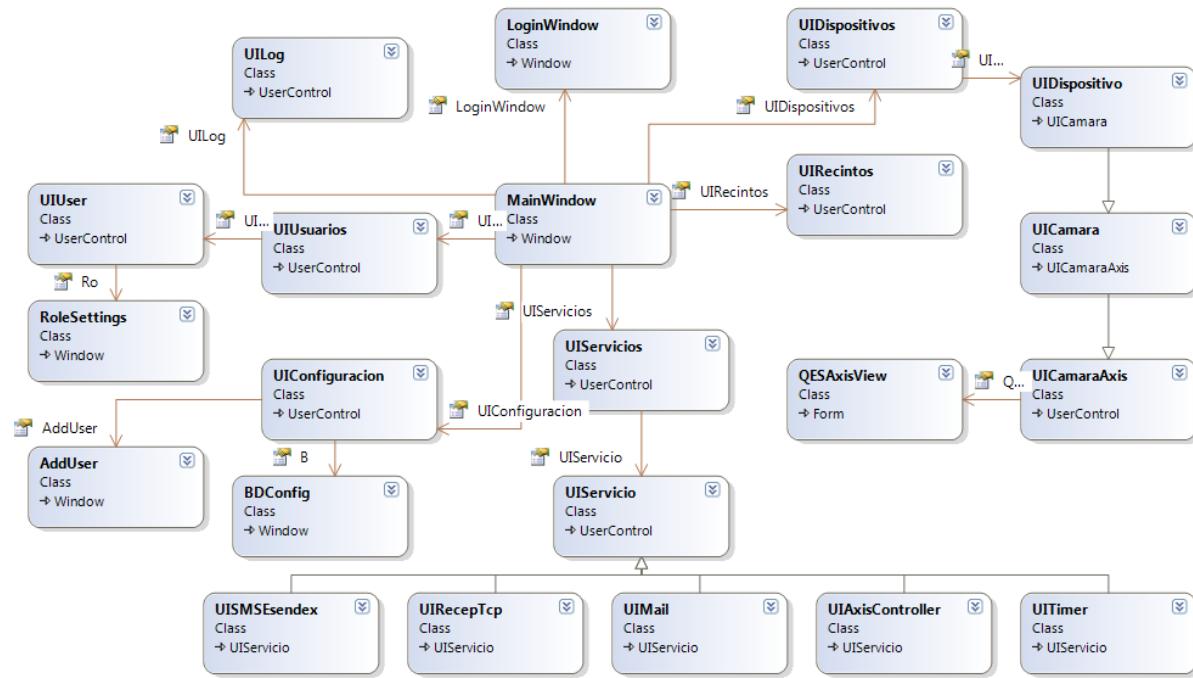
Junto a los roles tenemos los usuarios, que representan a las personas que acceden al sistema. Estos usuarios tendrán una serie de roles que representarán sus capacidades dentro del sistema. Para añadir más flexibilidad, los usuarios se han definido de forma que un usuario puede tener como rol a otro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

#### 4.3.9 Diagramas de clases Conceptuales

##### 4.3.9.1 Lógica del sistema:

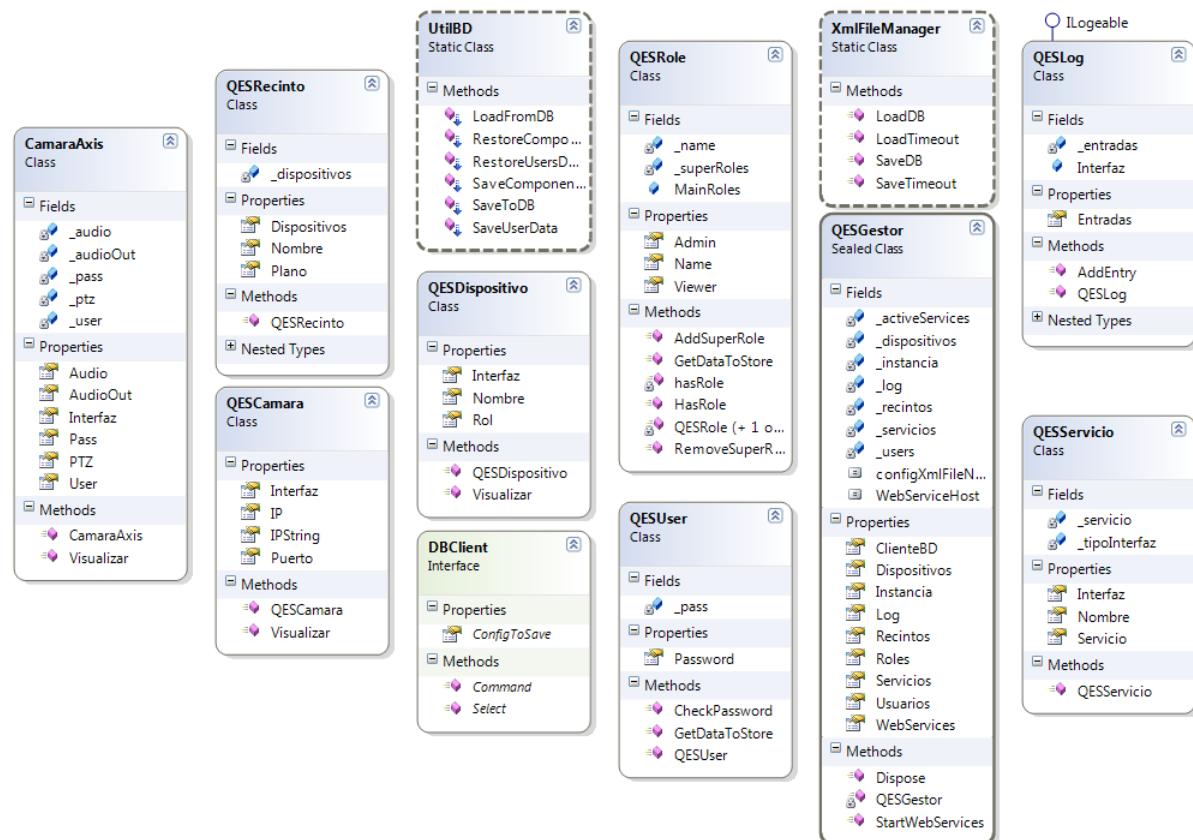


#### 4.3.9.2 Interfaz:

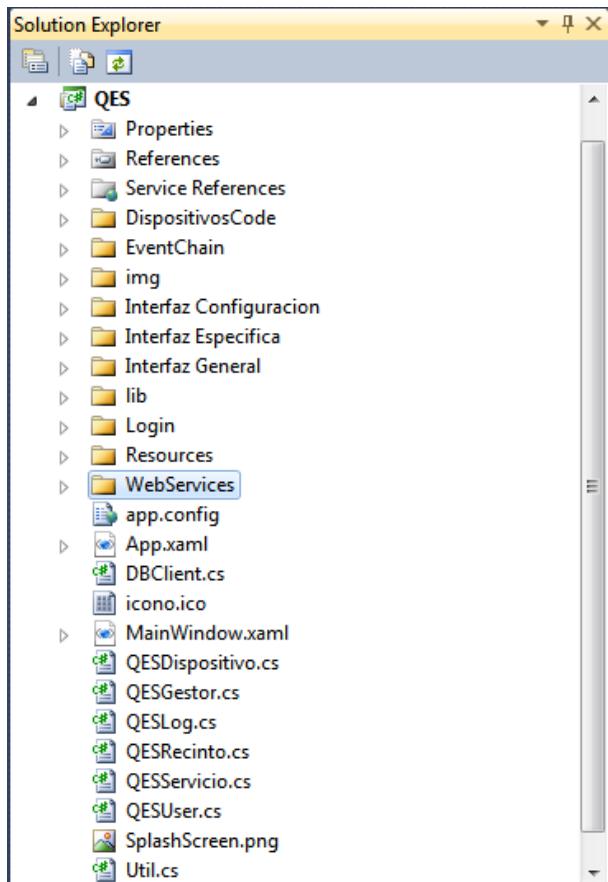


#### 4.3.10 Diagrama de Clases

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



#### 4.3.11 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 específicas de un dispositivo/servicio.
- **Interfaz general:** Interfaces generales

del sistema (usuarios, dispositivitos, etc...)

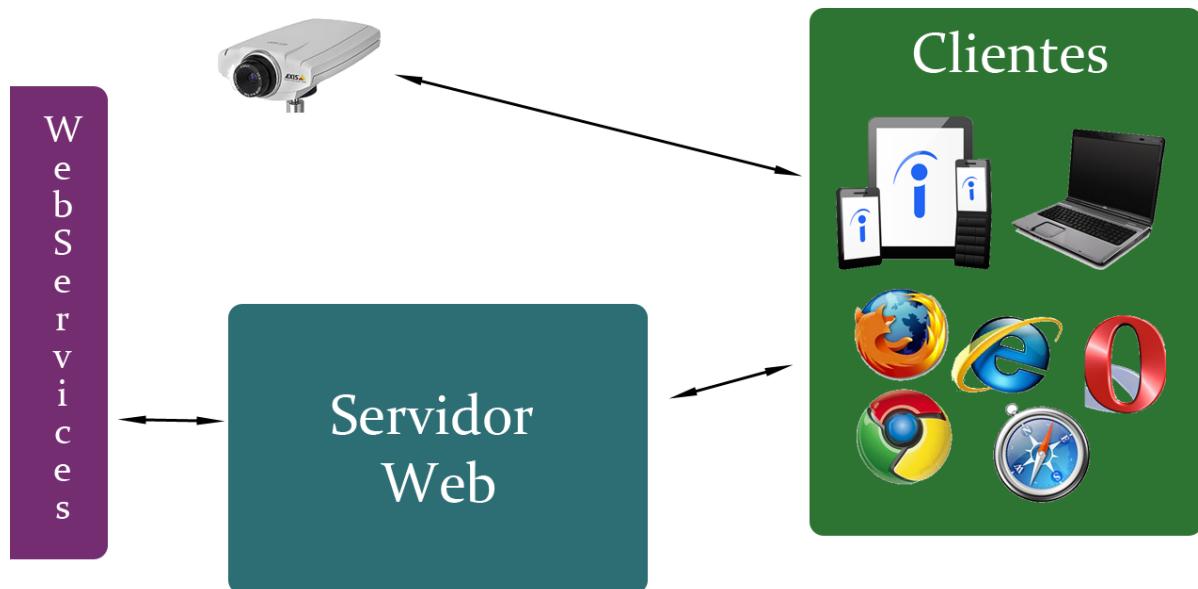
- **Login:** Clases para el control del login.
- **WebServices:** Servicios web implementados en el sistema.

Además, hay una serie de archivos:

- **App.xaml:** startingpoint de la aplicación
- **DBClient:** Conexiones a la base de datos.
- **MainWindow.xaml:** Pantalla principal de la aplicación.
- **QESDispositivo:** 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:** “Extensionmethods” para ampliar la funcionalidad de otras clases y clase para el tratado del fichero XML.

## 4.4 Análisis y diseño del cliente Web

### 4.4.1 Visión General

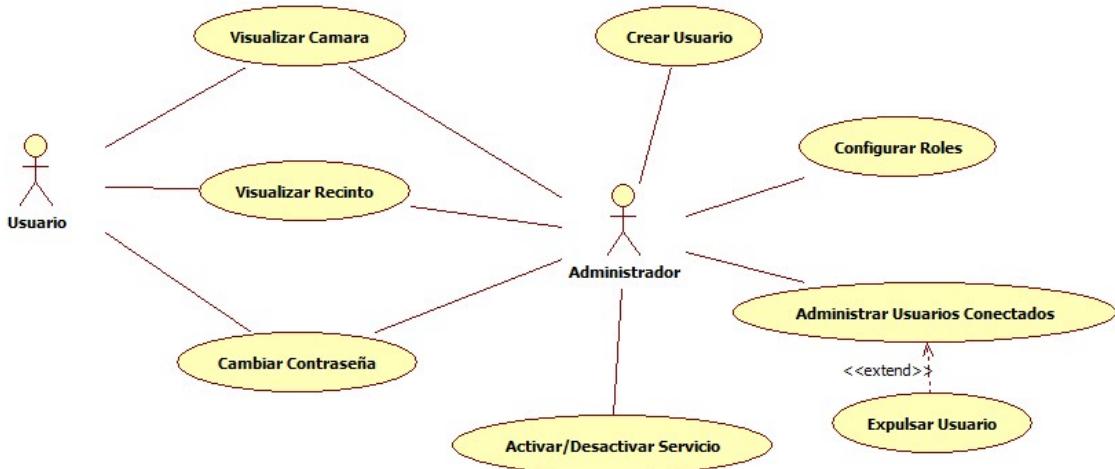


El servidor web, implementado en HTML5 y JavaScript y levantado sobre IIS responde a las peticiones de los clientes, los navegadores web. Este servidor web no contiene datos, sólo una capa de presentación y la lógica de ejecución, de esta forma, todos los datos (Usuarios, cámaras, dispositivos, recintos,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

Así, cuando un usuario desea acceder al flujo de vídeo 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 vídeo. Se optó por esta vía en lugar de pasar el flujo por el servidor para reducir la latencia y la carga en el servidor.

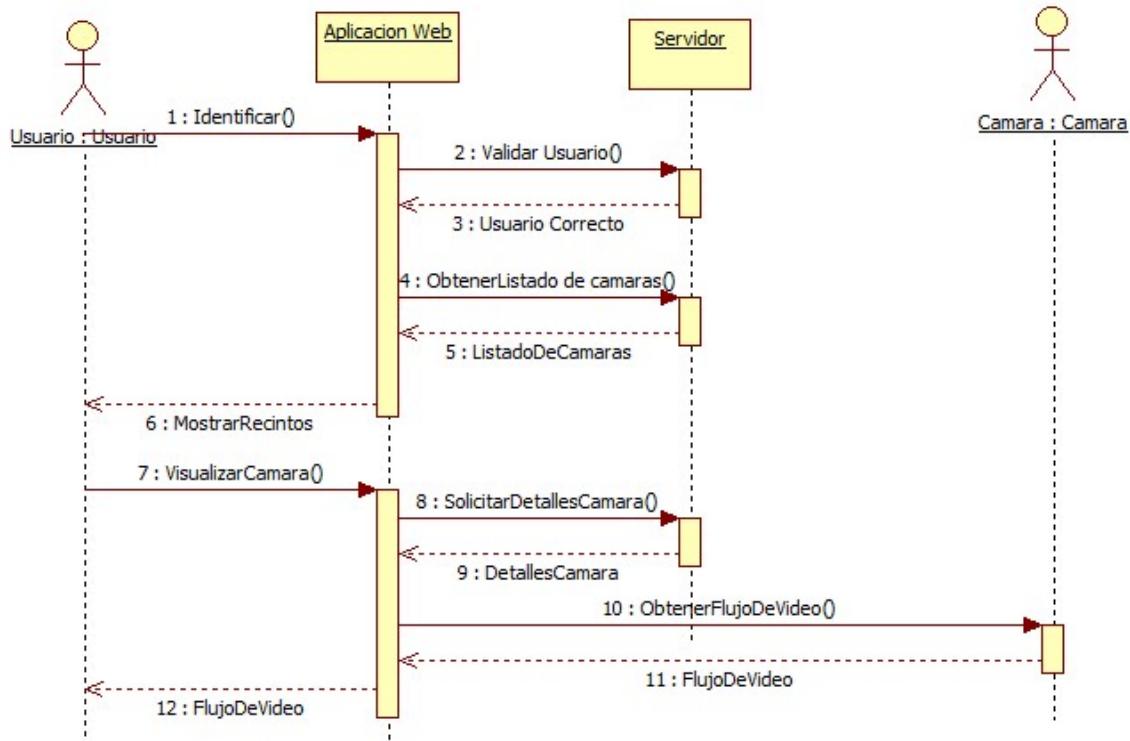
### 4.4.2 Diagramas de Casos de Uso

En esta sección se muestran el diagrama de casos de usos que expone la funcionalidad a la que puede acceder el usuario.



#### 4.4.3 Diagrama de secuencia

A continuación se expone el diagrama de secuencia para la obtención del flujo de vídeo en el cliente.

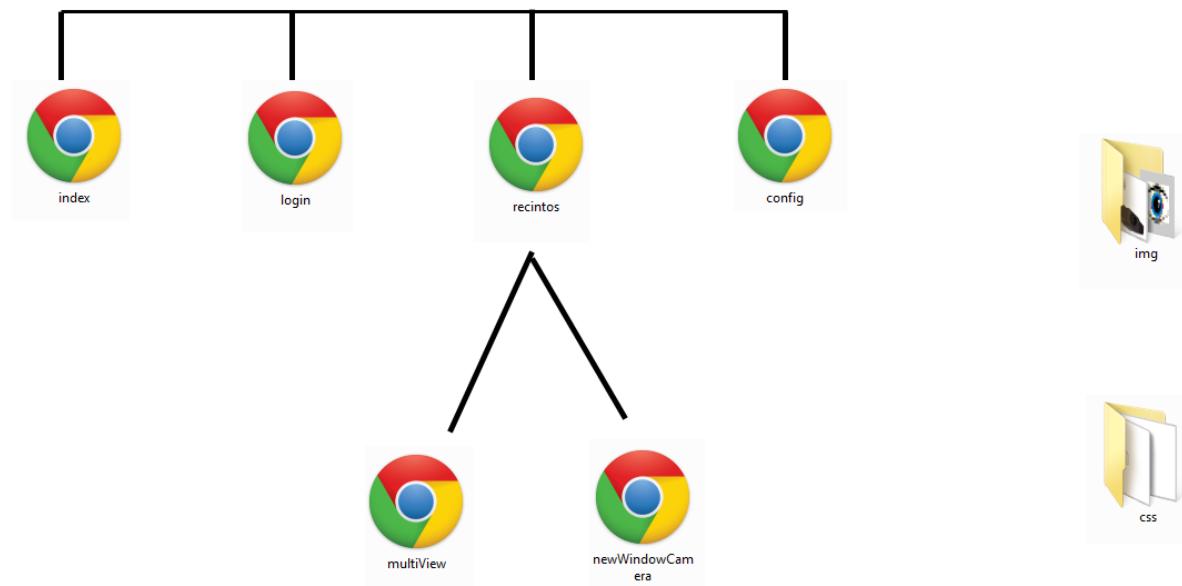


#### 4.4.4 Seguridad

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.

#### 4.4.5 Árbol de navegación



Aún con este árbol de navegación es importante tener en cuenta que sólo las páginas index y login pueden ser accedidas si el usuario no está identificado, 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.

#### 4.4.6 Script

El WebSite utiliza un script, "qesscript.js", que 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/";", define la ubicación de los servicios web a los que se accederá para comunicarse con el servidor.

#### 4.4.7 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, la vista de éste aumentará de tamaño automáticamente, su borde

incrementara su grosor y cambiará de color a la vez que se reproducirá un aviso sonoro para llamar la atención.



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

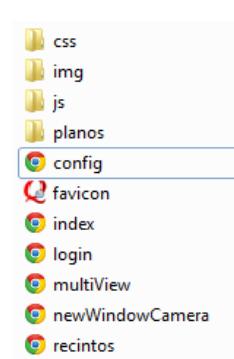
#### 4.4.8 Conexión a las cámaras

La captación del flujo de vídeo 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 ésta al servidor, y con éstos, 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 vídeo en tiempo real, puesto que pasar el vídeo a través del servidor podría provocar una sobrecarga de éste y podría producir un retraso considerable en el vídeo.

#### 4.4.9 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.

#### 4.4.10 Soporte

El servidor web pretende ser tan portable como sea posible, por ello no utiliza plugins dependientes de ninguna arquitectura específica. Está 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.

##### 4.4.10.1 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 vídeo, 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, ya que no es aceptada por éste puesto que carga los WebSockets en otro dominio (el de Pusher).

##### 4.4.10.2 Firefox y Safari



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

##### 4.4.10.3 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 así, 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 ha sido duramente criticado en la comunidad open source debido a que parece inadecuado desactivar por completo una herramienta estándar[22] que es actualmente usada por gran multitud de sistemas, sin 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)

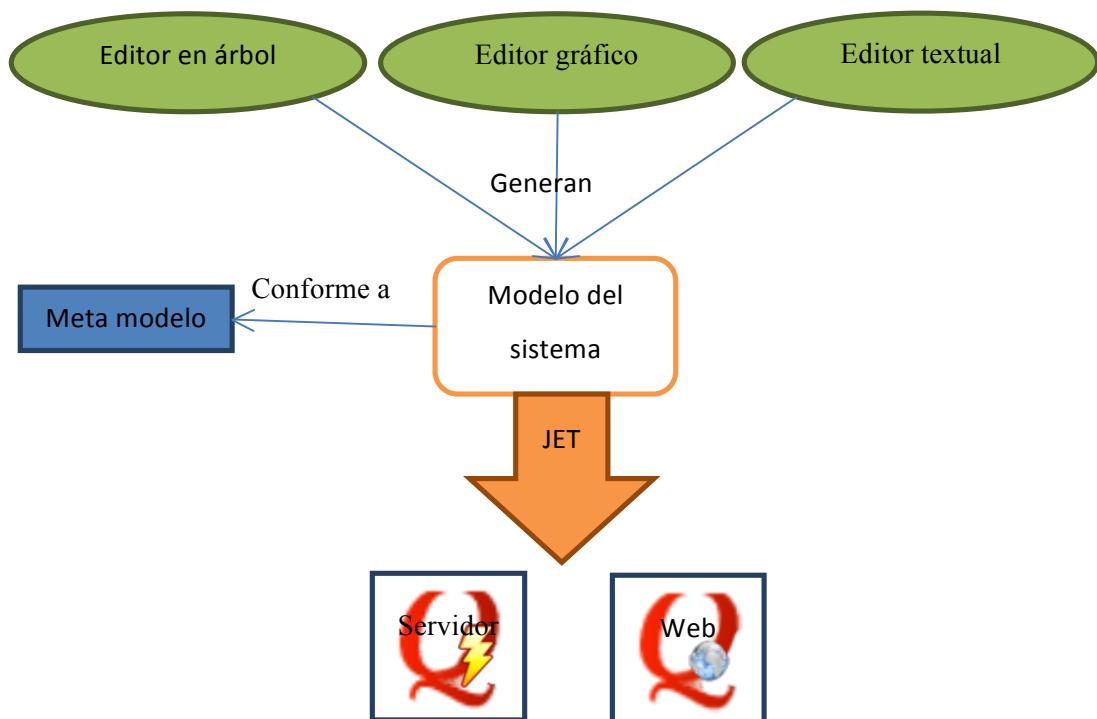
## 4.5 DSL

Por encima de las dos aplicaciones detalladas anteriormente, el servidor C# y la aplicación web, se encuentra un desarrollo dirigido por para desarrollar un DSL y proporcionar a la aplicación de una flexibilidad que no se puede obtener con el desarrollo tradicional de software. A través de esta parte del proyecto se permite al usuario (que configurará la aplicación) definir un modelo que represente el sistema para que éste se adapte a sus necesidades.

Este modelo debe ser conforme al meta modelo diseñado y se proporcionan tres editores para crearlo.

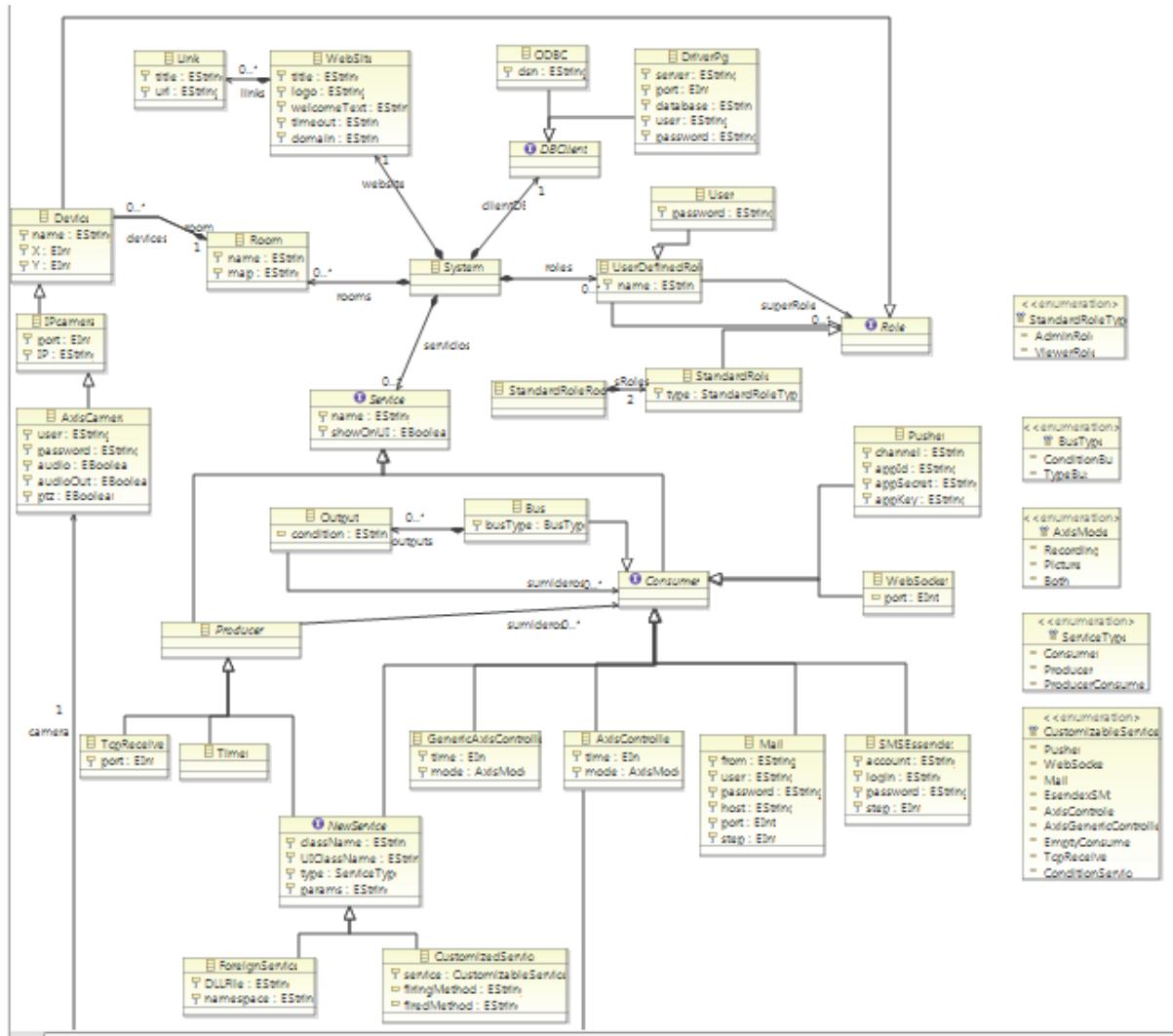
- Editor en árbol
- Editor gráfico
- Editor textual

A través de cualquiera de estos tres editores, se generará el modelo que será utilizado para la configuración y adaptación del sistema a las características que el usuario desea. Esto hace que el sistema sea además de sólido, escalable y altamente flexible.



#### 4.5.1 Meta modelo

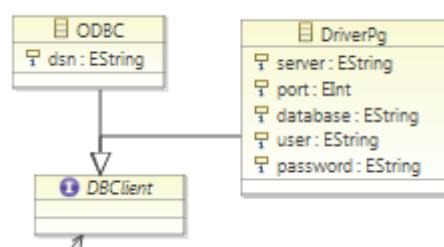
El meta modelo ha sido diseñado buscando que permita la mayor flexibilidad posible y la mayor facilidad de construcción de modelos para el usuario. Así el meta modelo es el siguiente (versión con mayor calidad disponible en la copia digital):



Observando cada parte en detalle, tenemos:

##### 4.5.1.1 Base de datos

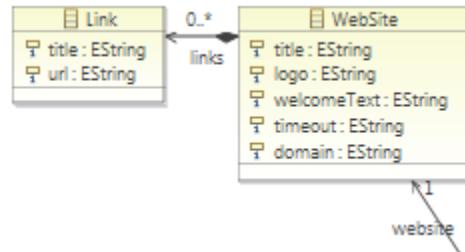
La configuración de la base de datos se realiza mediante una de las dos clases DriverPg u ODBC. La primera significará realizar una conexión directamente desde el servidor a la base de datos PostgreSQL (se obliga a usar este SGBD) con los parámetros de configuración que ésta tiene como atributos mientras que la segunda utilizará los orígenes de datos



de Windows para realizar la conexión, siendo necesario únicamente indicar el DSN a utilizar (la configuración del ODBC se realizará aparte). Sólo puede incluirse una única entidad de uno de estos dos tipos.

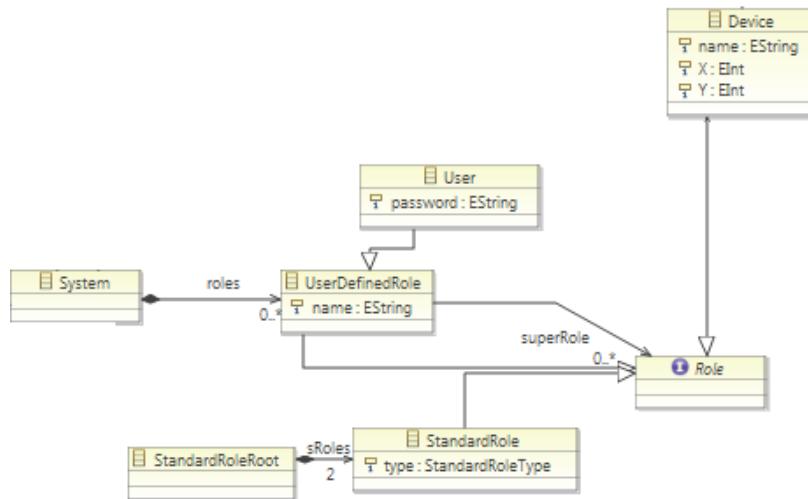
#### 4.5.1.2 WebSite

El meta modelo incluye también un apartado para la definición del WebSite. Éste tiene un atributo domain que es el dominio tras el que debe encontrarse todo el sistema y otros parámetros para configurar el layout de la página como el texto de bienvenida, enlaces de la pagina de inicio, titulo o el icono utilizado como logo.



#### 4.5.1.3 Usuarios y roles

La gestión de usuarios y roles ha sido diseñada estudiando cada detalle, resultando finalmente en una separación de Roles estándares (Admin y Viewer), privilegios asociados a las cámaras y roles definidos por el usuario (dentro de los cuales están los usuarios, que son un rol con contraseña). Los roles definidos por el usuario pueden contener super roles asociados (de cualquier tipo). Esto significa que los roles estándares no podrán ser creados por el usuario, sólo aplicados a roles definidos por el usuario como super roles.



Es importante tener en cuenta que cuando se cree un modelo será necesario importar un recurso que sea un modelo que contenga los dos superroles (Admin y Viewer). Para la generación de éste modelo se ha creado un elemento Root para éstos, StandardRoleRoot.

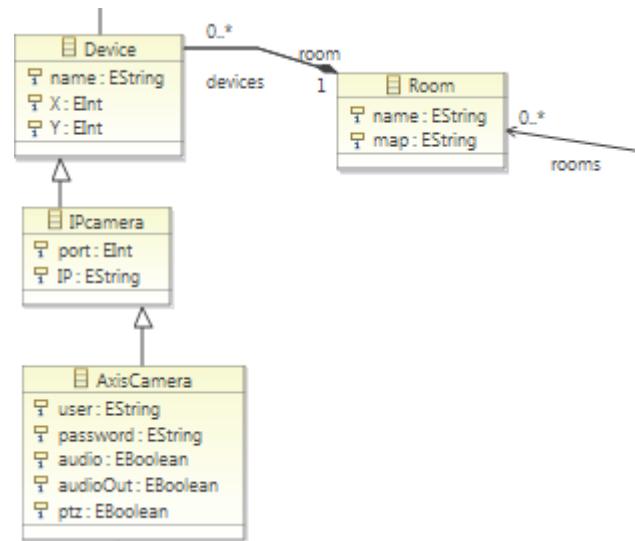
#### 4.5.1.4 Dispositivos

Los dispositivos se encuentran agrupados en recintos, los cuales tienen un nombre y una imagen asociada como mapa. Cada dispositivo tiene diferentes atributos, pero todos tienen en común las coordenadas en las que deben situarse en el mapa.

Actualmente hay tres tipos de dispositivos que pueden definirse en el sistema:

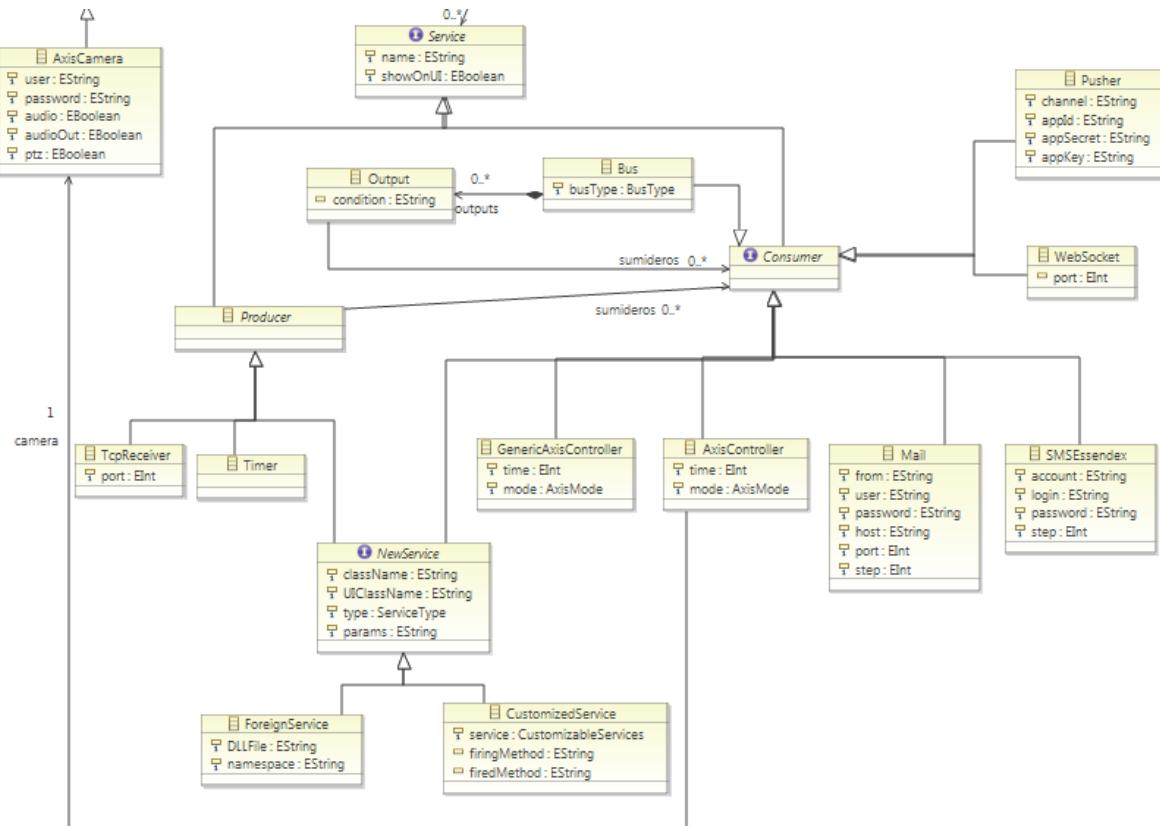
- Cámaras Axis
- Cámaras IP
- Dispositivos

Se espera que en un futuro se amplíe el conjunto de dispositivos, especialmente para sensores ya que actualmente se agrupan dentro de dispositivo y podría obtenerse una mayor funcionalidad.



#### 4.5.1.5 Servicios

La especificación de los servicios ha sido la más cuidada, representando el principal objetivo del proyecto. A través de ésta se podrá configurar el comportamiento del gestor de eventos del sistema definiendo aquí sus conexiones, así como incluir nuevos servicios añadidos al estilo de plugins, haciendo al sistema realmente flexible.



Los servicios se encuentran principalmente divididos en dos grupos, Productores y Consumidores de eventos, pudiendo estar los consumidores conectados a los productores, tal y como se explicó en el capítulo dedicado a la cadena de eventos.

Los servicios que pueden crearse y que ya han sido explicados anteriormente son:

- **TcpReceiver**: Receptor de mensajes TCP para gestionar los eventos emitidos desde la cámara como detección de movimiento o cross line.
- **Timer**: Temporizador, permite lanzar eventos en un momento determinado.
- **GenericAxisController**: Servicio para realizar grabaciones/captura de imágenes utilizando el ActiveX de Axis en cualquier cámara.
- **AxisController**: Servicio para realizar grabaciones/captura de imágenes en una cámara determinada.
- **Mail**: Servicio para enviar correos a través de un servidor SMTP.
- **SMSessendex**: Servicio para el envío de mensajes de texto.
- **Pusher**: Servicio de notificación a los clientes web.
- **WebSocket**: Servicio de notificación a los clientes web.
- **Bus**: Permite distribuir los eventos en base a condiciones a través de varias salidas.

Además, se incluye la posibilidad de añadir nuevos servicios de dos formas:

- Personalización de servicios existentes.
- Creación de servicios nuevos.

La primera consiste en tomar un servicio existente y redefinir el método que realiza el envío/manejo del evento. Esto permite personalizar el comportamiento de un servicio por dentro sin necesidad de tocar más que un método, pudiéndose hacer desde el propio editor de eclipse.

La segunda opción, mucho más interesante, consiste en, utilizar un DLL que proporciona el proyecto nombrado “BaseDLL.dll” para realizar la implementación de un servicio nosotros mismos. En el DLL mencionado se encuentran las principales interfaces necesarias para la implementación de un servicio tal y como se detalló en el capítulo que habla sobre la cadena de eventos. Con esto es posible crear un proyecto tipo biblioteca de clases en Visual Studio, importar el DLL y generar nuestro propio servicio, así como un UserControl que podrá ser utilizado como interfaz. Merece la pena mencionar que la implementación del servicio no tiene por qué ser en C#, pudiendo ser en cualquiera de los lenguajes que soporta .Net, ya que será traducido cuando se genere el DLL.

Gracias a estas dos características, como se puede ver, el sistema proporciona una flexibilidad abismal permitiendo añadir servicios como plugins junto con su interfaz y conectándolos como desee sin necesidad de tocar el proyecto original. Esta es sin duda, la pieza clave del proyecto.

#### 4.5.2 Restricciones OCL

Para asegurar que el modelo es correcto ha sido necesario añadir restricciones que no se podían definir a través del meta modelo en sí, por lo que se ha utilizado OCL. A continuación se detallan las restricciones.

##### 4.5.2.1 System

```
invariantOneAdminRequired('Al menos debe haber un administrador definido'):
User.allInstances()->exists(u : User | u.superRole->exists(sr : StandardRole | sr.type =
StandardRoleType::AdminRole));
Comprueba que al menos existe un administrador en el sistema.
```

##### 4.5.2.2 AxisCamera

```
InvariantUserAndPassword('Debe indicar usuario Y contraseña o dejar ambos en blanco para indicar que no tiene.'):
not (user = nullxorpassword = null);
Si una cámara tiene establecido usuario o contraseña debe tener establecido los dos.
```

#### 4.5.2.3 Pusher

```
invariant Consistencia('Si indica uno de los tres atributos debe introducirlos todos(appSecret,appKey y appID)'):

not (appId = nullxorappSecret = nullxorappKey = null);
```

Si se establece una de las propiedades de configuración del servicio pusher deben establecerse todas (salvo channel) ya que están relacionadas.

#### 4.5.2.4 TcpReceiver

```
invariantDifferentPort('No puede haber dos receptores con el mismo puerto.'):
not TcpReceiver.allInstances()->exists(wb : TcpReceiver | (wb<>selfandwb.port = self.port));
```

No puede haber dos receptores TCP con el mismo puerto de escucha.

#### 4.5.2.5 NewService

```
InvariantServiceTypeRestrictions('El servicio especificado tiene un tipo indicado que no permite las conexiones realizadas.'):
type = ServiceType::ProducerConsumerortype =
    ServiceType::ProducerandnotProducer.allInstances()->includes(self) ortype =
        ServiceType::Consumerandself.sumideros->size() = 0;
```

Si un servicio está marcado como consumidor no puede tener conexiones salientes y si está marcado como productor no puede tener conexiones entrantes.

#### 4.5.2.6 CustomizedService

```
invariantTypeNotConsistent('El tipo de servicio no es consistente con el servicio a personalizar.'):
service = CustomizableServices::Pusherandtype = ServiceType::Consumerorservice =
    CustomizableServices::WebSocketandtype = ServiceType::Consumerorservice =
        CustomizableServices::Mailandtype = ServiceType::Consumerorservice =
            CustomizableServices::EsendexSMSandtype = ServiceType::Consumerorservice =
                AxisControllerandtype = ServiceType::Consumerorservice =
                    CustomizableServices::AxisGenericControllerandtype =
                        ServiceType::Consumerorservice = CustomizableServices::EmptyConsumerandtype =
                            ServiceType::Consumerorservice = CustomizableServices::TcpReceiverandtype =
                                ServiceType::Producerorservice = CustomizableServices::ConditionServiceandtype =
                                    = ServiceType::ProducerConsumer;
```

El tipo de servicio es acorde con el servicio seleccionado a personalizar.

#### 4.5.2.7 WebSocket

```
invariantDifferentPort('No puede haber dos WebSockets con el mismo puerto.'):
not WebSocket.allInstances()->exists(wb : WebSocket | (wb<>selfandwb.port = self.port));
```

No puede utilizarse un puerto de salida para dos WebSockets.

#### 4.5.2.8 UserDefinedRole

```
invariantInappropriateName('Un rol definido por el usuario no puede llamarse Admin o Viewer'):
```

```
name<>'Admin' and name<>'Viewer';
```

Los roles definidos por el usuario no pueden tener como nombre "Admin" o "Viewer".

#### StandardRoleRoot

```
invariantSingleAdmin('Debe haber un \u00fAnico rol Admin en el sistema.');
```

```
StandardRole.allInstances()->select(r : StandardRole | r.type = StandardRoleType::AdminRole)->size() = 1;
```

```
invariantSingleViewer('Debe haber un unico rol Viewer en el sistema.');
```

```
StandardRole.allInstances()->select(r : StandardRole | r.type = StandardRoleType::ViewerRole)->size() = 1;
```

Debe definirse un rol Admin y un Viewer únicamente.

### 4.5.3 Decisiones generales del modelo

#### 4.5.3.1 Configuración y coherencia

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.

#### 4.5.3.2 Servicios

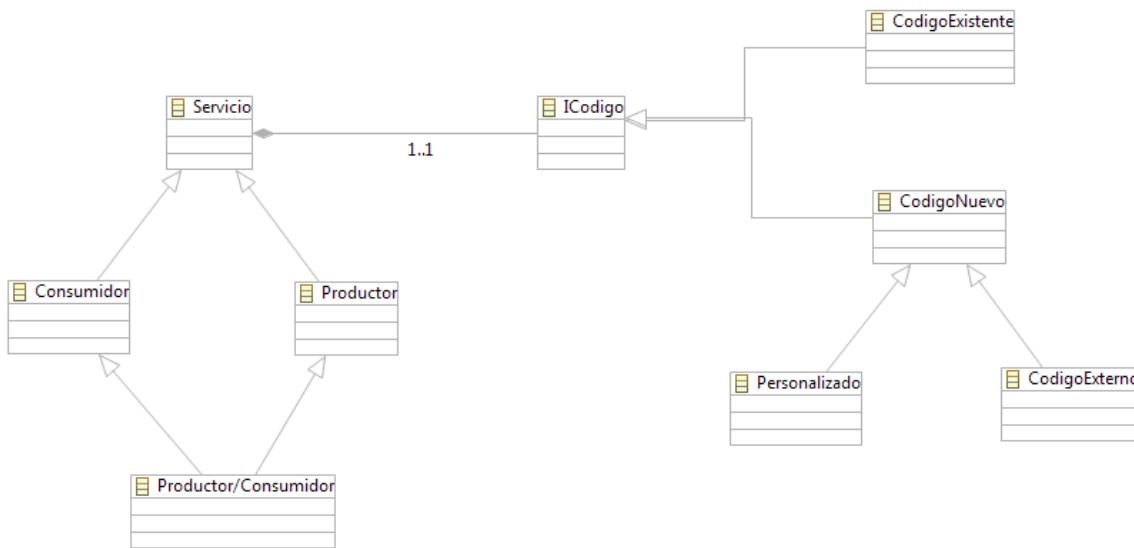
La definición del meta-modelo para la inclusión de servicios definidos por el usuario fue uno de los puntos más complicados, ya que hay dos formas de categorizar los servicios:

- Consumidores.
- Productores.
- Productores/Consumidores.

Y

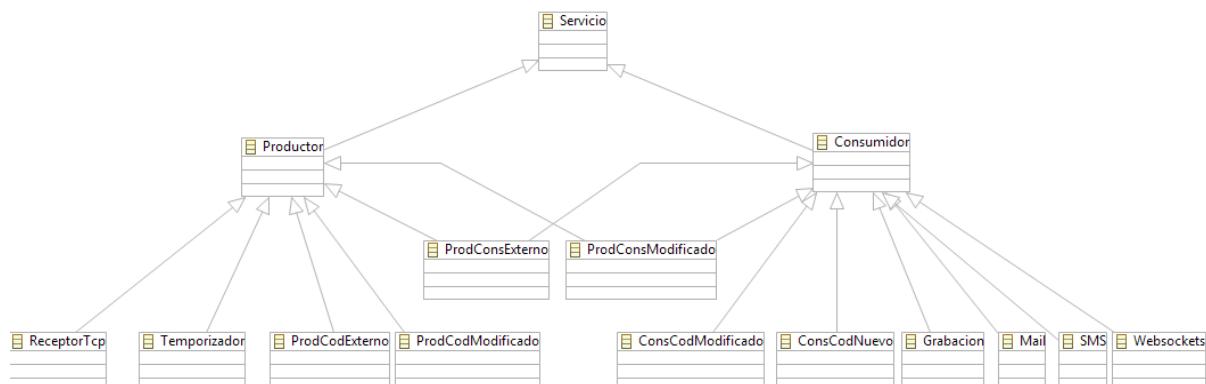
- Servicios ya existentes.
- Servicios nuevos a partir de una personalización de un servicio existente.
- Servicios totalmente nuevos.

Como solución se propusieron los siguientes meta-modelos:



En este modelo se definen en un principio todos los servicios por igual, indicando sólo 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 una diferenciación correcta en MDT para poner diferentes iconos o que el usuario no sabría qué parámetros son obligatorios. Aun así, estos inconvenientes son menores en comparación con la simplicidad que ofrece.

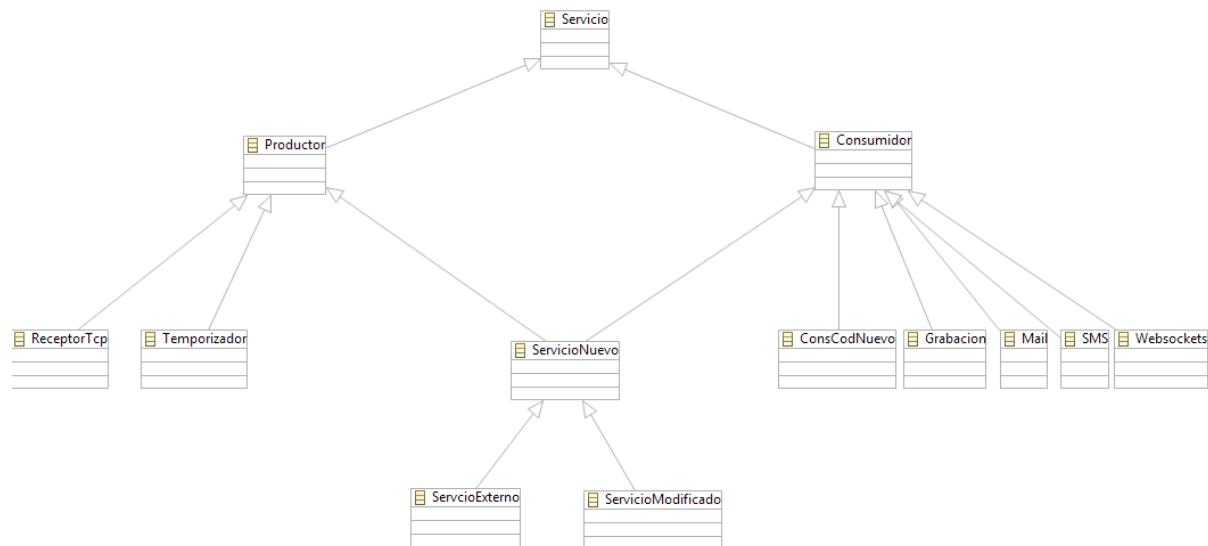


Este modelo diferencia en primer lugar los servicios en Consumidores y Productores y después en los diferentes tipos de servicio que el sistema ya posee. Además, añade dos subtipos a consumidor, dos subtipos a productor y dos subtipos comunes a ambos para añadir nuevos servicios.

Este modelo es más complicado 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 sólo 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 utilizar una alternativa intermedia, 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 conteniendo un atributo para indicar cual de los dos implementa. De esta clase heredan dos clases para definir un servicio que extiende otro o para definir un servicio totalmente nuevo.

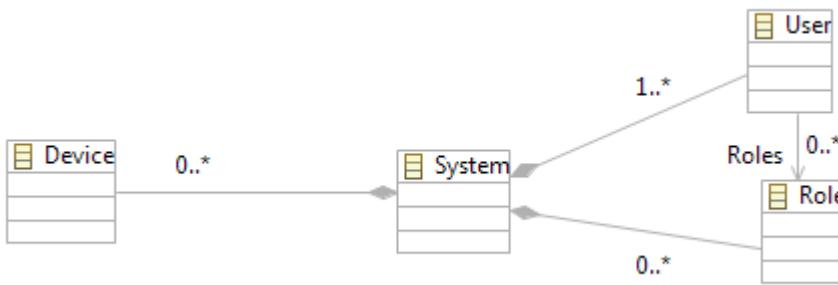


#### 4.5.3.3 Roles

La definición de los roles y los usuarios ha sido complicada en la definición del meta modelo.

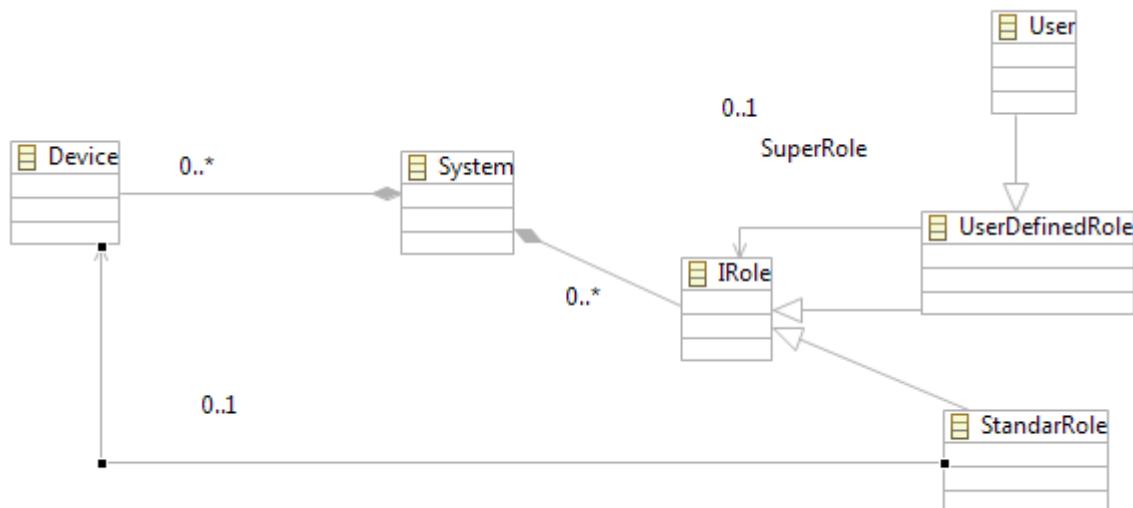
Esto se ha debido a diferentes problemas:

- 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.



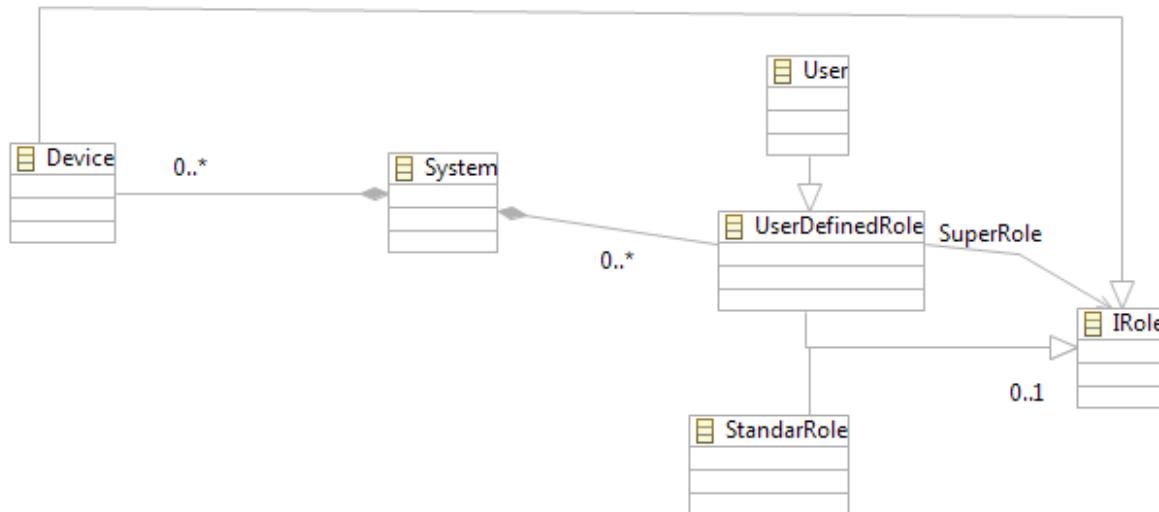
En un primer lugar se optó por un diseño “convencional” en el que el sistema tenía roles y usuarios (que tenían roles asociados como privilegios). En esta opción debían crearse una serie de roles de forma obligatoria (a mano por el usuario): Admin, Viewer y uno por cada dispositivo en el sistema. Al 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 tenía 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 sí 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 éste 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.

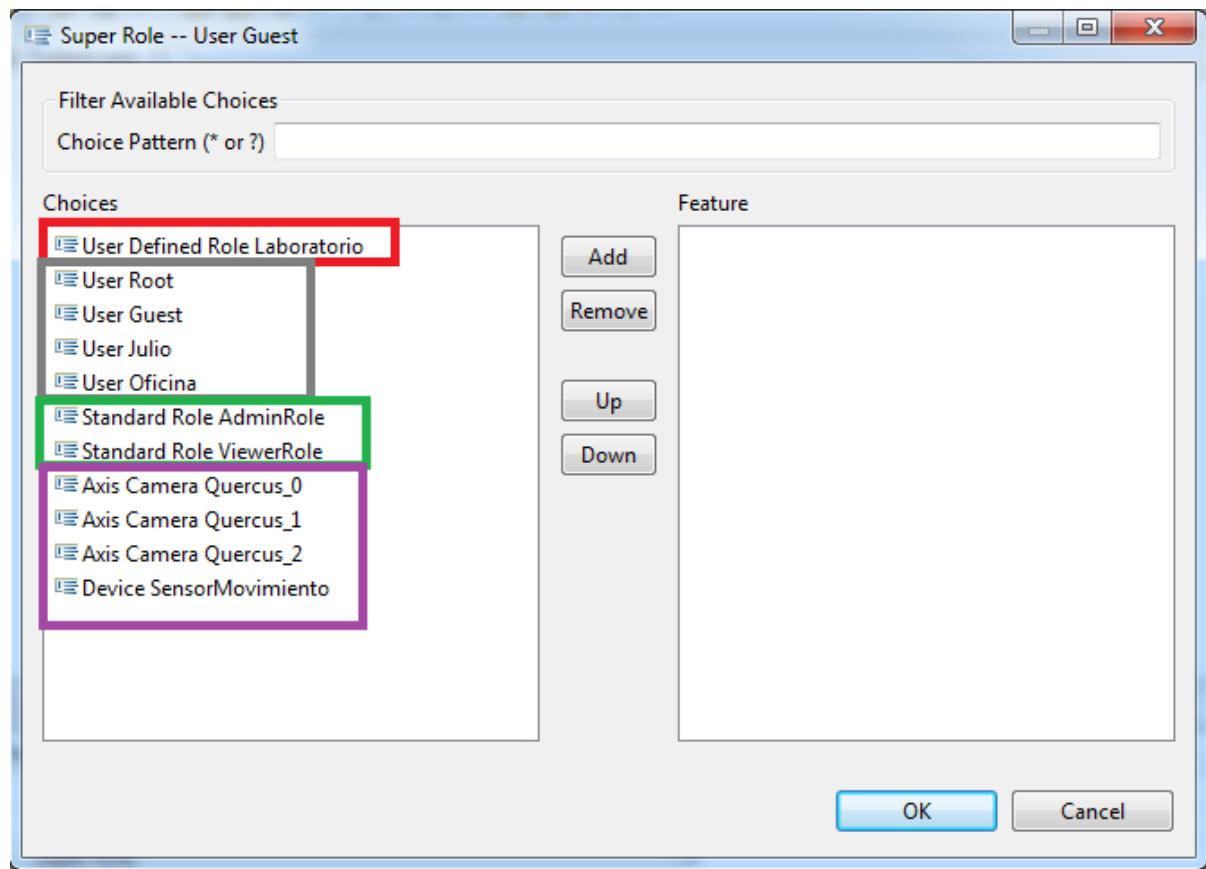


Pero esta solución aun tenía un problema, se obligaba al usuario a crear los roles del sistema, lo que carece de sentido, puesto que son roles que siempre van a existir, en todos los modelos que hagamos. De esta forma, se transforma el meta modelo primero haciendo que el sistema contenga únicamente los roles definidos por el usuario, eliminando así la capacidad de crear roles estándar al usuario, y, en segundo 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. 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 al 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.

#### *4.5.3.4 Servicios de notificación en el modelo*

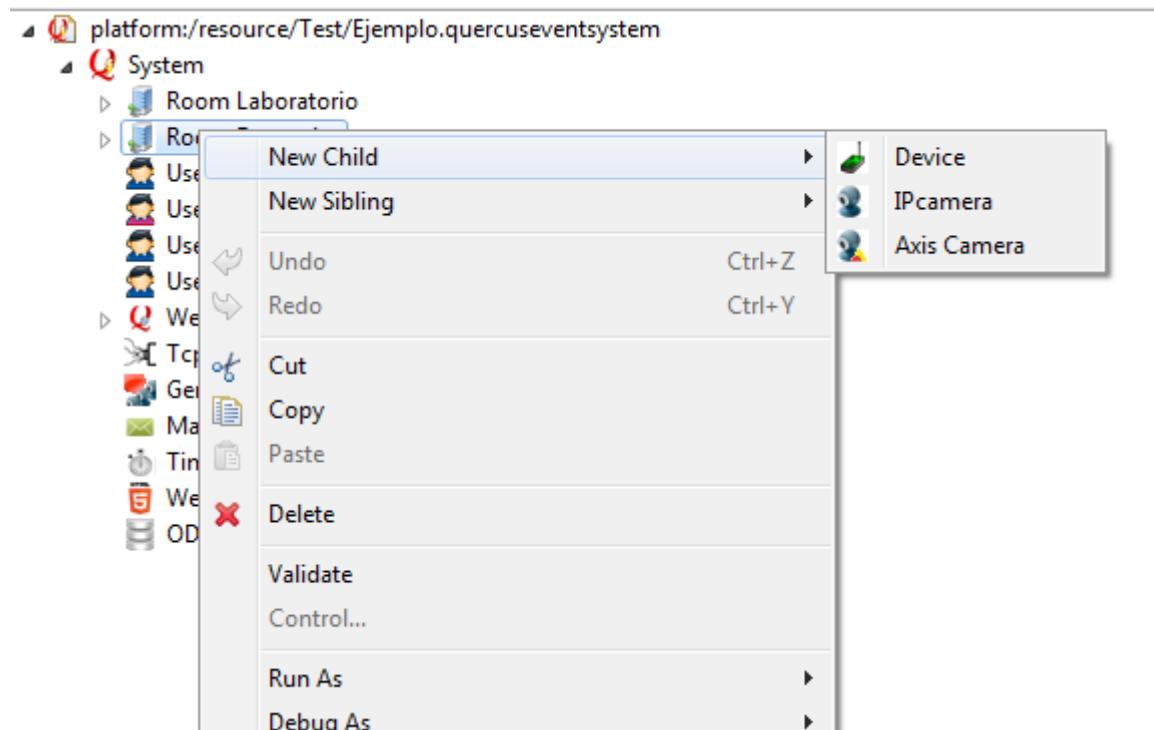
Otro punto interesante del modelo fue el modo de incluir los servicios de WebSockets y Pusher, ya que no tiene sentido tener ambos y puede ser considerado una característica del sistema en lugar de un servicio. Pero debido a que en un futuro puede ser utilizado como un servicio (para enviar mensajes al navegador por ejemplo), que se desea dar al usuario la posibilidad de elegir uno y que es necesaria la definición de ciertos parámetros para ambos, se decidió incluirlos como servicios.

#### **4.5.4 Editores de modelo**

Con el objetivo de dotar al usuario final de diferentes formas de crear el modelo del sistema, se han creado tres editores, pudiendo el usuario determinar cual le resulta mas cómodo de usar.

#### 4.5.4.1 Editor en Árbol

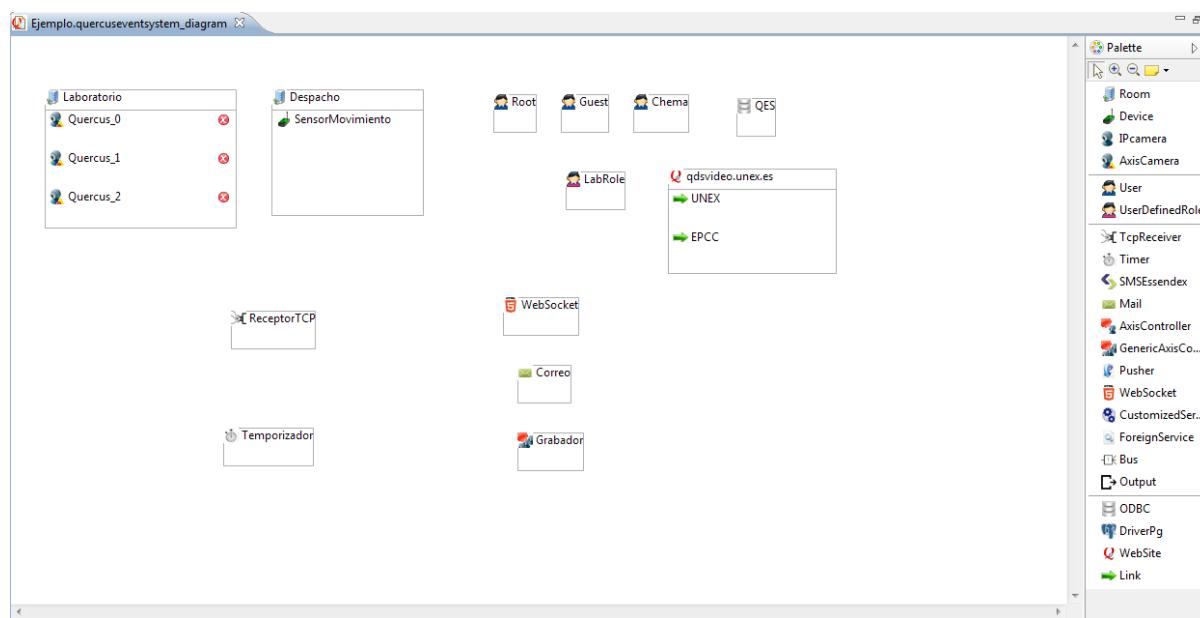
Para la creación de modelos del sistema se ha creado un editor en árbol conforme al meta modelo. Este editor es el mismo editor que se utiliza para la creación de meta modelos en eclipse y se genera a través de EMF.



#### 4.5.4.2 Editor Gráfico

Para la creación de modelos de una forma mas amena se ha generado un editor grafico que permite diseñar el sistema de arrastrando componentes desde una paleta.

El editor gráfico ha sido generado sobre GMF que se apoya en EMF, por lo que realmente estamos creando dos ficheros al crear un modelo de esta forma, un fichero del modelo tal y como generamos con el editor en árbol (de hecho se puede abrir con éste) y un fichero que contiene la configuración gráfica del modelo.



#### 4.5.4.3 Editor Textual

Se dispone de un editor textual, generado a través de Xtext. El editor, conforme al meta modelo descrito anteriormente, permite definir un modelo del sistema de una forma que puede resultar más agradable a usuarios expertos para instalaciones grandes. Además, este editor puede ser enriquecido con sintaxis propia que ayude a la definición de servicios nuevos desde el propio DSL o a la configuración del sistema cuando ciertas propiedades se repiten a través de bucles y otras herramientas.

El editor generado dispone de las siguientes características:

- Auto Completado. Pulsando Ctrl + Espacio el editor ayuda al usuario mostrando las opciones posibles o completando una palabra.
- Detección de errores sintácticos y gramaticales mientras se escribe. El editor informa de los posibles fallos sintácticos y gramáticos mientras se escribe.
- Validación semántica. Una vez terminado el modelo, se puede hacer segundo click y validarla contra el meta modelo, comprobando también todas las restricciones.
- Coloreado de sintaxis.
- Ayuda en la corrección de errores. Cuando se detecta un error, el editor además intenta ofrecer soluciones alternativas.

```

    TcpReceiver Receptor_TCP {
        showOnUI true
        port 11111
        sumideros ( ^WebSocjket,      ^Mail )
    },
    WebSocket ^WebSo
        showOnUI tru
    },
    Mail ^Mail {
        showOnUI tru
        from "quercu
        user "quercu
        password quercuseventsyste
        host "smtp.gmail.com"
        port 587
        step 5
    }
}

```

Couldnt resolve reference to Consumer 'WebSocjket'.  
 3 quick fixes available:  
 ↗ Change to 'Grabacion'  
 ↗ Change to 'Mail'  
 ↗ Change to 'WebSocket'

Press 'F2' for focus

## Gramática

La gramática se ha realizado totalmente acorde al meta modelo siguiendo las siguientes convenciones: la definición de un nuevo elemento se hace siguiendo el patrón de “Tipo” “Nombre” {atributos}; los atributos se separan por espacios; el nombre de atributo y el valor se separan por espacios; los valores booleanos no obligatorios se consideran false si no están presentes y true en caso contrario; las cadenas no tienen que llevar doble comilla necesariamente(a no ser que haya espacios o caracteres extraños en la cadena); para referenciar a otro elemento basta con indicar su nombre; en composición o referencia de varios elementos, éstos se separan por comas y se pueden incluir comentarios a través de “//” para toda una línea o con “/\*” y “\*/”.

La definición de la gramática entera se puede encontrar en el proyecto del DSL en eclipse en el fichero MyDSL.xtext.

### 4.5.5 Transformaciones M2T (JET)

La última parte del modelado es un proyecto JET que transforma un modelo generado por EMF (o cualquiera de los editores basados en éste) en el código del sistema a desplegar totalmente configurado tal y como indica el modelo.

El proyecto genera las siguientes salidas:

- Aplicación web: Se genera código HTML5 y JavaScript que configuran la aplicación web.
- Aplicación Servidor: Se genera código C# y archivos de Microsoft para la compilación que realiza toda la configuración del servidor y queda los ficheros de proyecto listo para compilar el proyecto con las librerías necesarias.

- Batch de despliegue: Un archivo .bat es generado que realiza la copia de los archivos necesarios (DLL, imágenes, mapas, etc...), realiza la compilación y al ser ejecutado dispone de un menú para la instalación del software necesario para el sistema o el inicio del sistema.

Lista de ficheros generados:

- Index.html (Código HTML5)
- QES.bat (Batch de despliegue y configuración)
- QES.csproj (Proyecto C# incluyendo nuevos servicios)
- QESGestor.cs (Fichero C# principal con la configuración del sistema, servicio, usuarios, etc...)
- Qesscript.js (Fichero JavaScript con la configuración para la aplicación web)

```
C:\Windows\system32\cmd.exe
BUILDING...
  1 file(s) copied.
  1 file(s) copied.
  1 file(s) copied.
  1 file(s) copied.
  1 file(s) copied.
QES -> D:\Mis Documentos\workspace\EclipseMDA\quercuseventsyste...jet\output\QuercusEventSystem\QuercusEventSystem\bin\Release\QuercusEventSystem.exe

PRESS 1, 2, 3 OR 4 to select your task, or 5 to EXIT.
-----
1 - Start Installer
2 - Execute Quercus Server
3 - Open the output folder
4 - Open C# Solution at VS <(VS Required>
5 - EXIT

Type 1, 2, 3, 4 or 5 then press ENTER:_

```

El batch generado debe ser ejecutado para realizar la copia de archivos y la compilación. Esta herramienta ofrece la posibilidad de abrir el menú de instalación (PostgreSQL, herramienta ODBC, el cliente, servidor, etc...), ejecutar la aplicación directamente, abrir la carpeta de salida o abrir la solución en Visual Studio si éste está instalado.

Para la correcta generación de la solución, deberemos incluir en la carpeta input los siguientes ficheros:

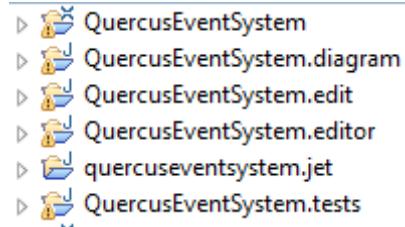
- Imágenes correspondientes a los mapas de los recintos.
- Archivos DLL en los que se encuentran definidos los servicios externos.
- Imagen del logo de la web.
- Modelo de entrada.

Para generar los DLL, se dispone de un archivo baseDLL.dll con el que hay que generar en Visual Studio una solución de tipo Biblioteca de Clases (importando baseDLL.dll) cuya generación crea el fichero necesario.

#### 4.5.6 Workspace

En el workspace de eclipse que se utilizo para le modelado se pueden encontrar los siguientes proyectos:

- QuercusEventSystem: Proyecto principal, contiene el meta modelo, el generador del editor en árbol y los ficheros básicos para la generación del editor grafico. Contiene también un ejemplo de sistema y un modelo de los roles estándares. En la carpeta src se encuentra la implementación java del meta modelo.
- QuercusEventSystem.diagram: Editor grafico.
- QuercusEventSystem.edit y QuercusEventSystem.editor: Implementación del editor en árbol.
- QuercusEventSystem.qesdsl y QuercusEventSystem.qesdsl.ui: Implementación del editor textual.
- QuercusEventSystem.jet: Implementación de las transformaciones modelo a texto a través de JET.



## 5. Manual de usuario

Debido a que el sistema es usado por diferentes tipos de usuarios se han creado tres manuales diferentes, uno para cada aplicación.

### 5.1 Manual de la aplicación servidor

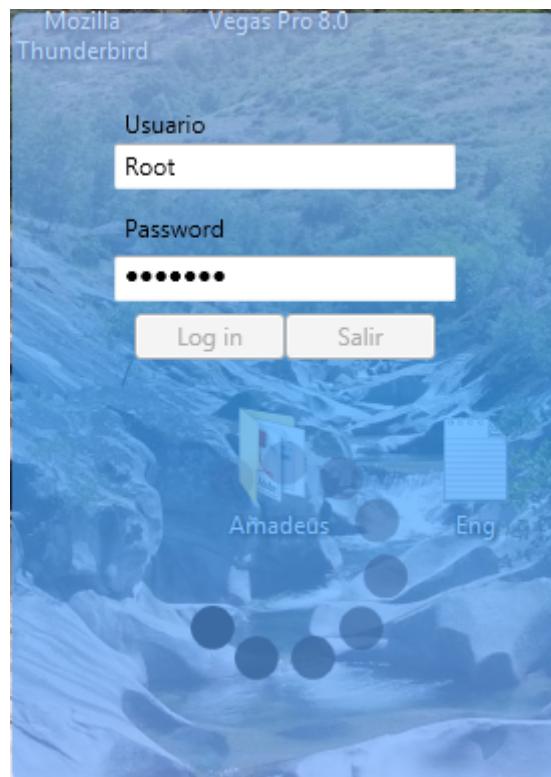
La aplicación servidor debe ser utilizada únicamente por los administradores del sistema.

#### 5.1.1 Interfaz de usuario

Cuando iniciemos la aplicación ésta necesitará unos segundos para iniciarse, debido a que se realiza la conexión a la base de datos, el inicio de la cadena gestora de eventos y la activación de los servicios web al inicio de la aplicación. Mientras estas tareas se realizan, se muestra la pantalla de espera mostrada a continuación.

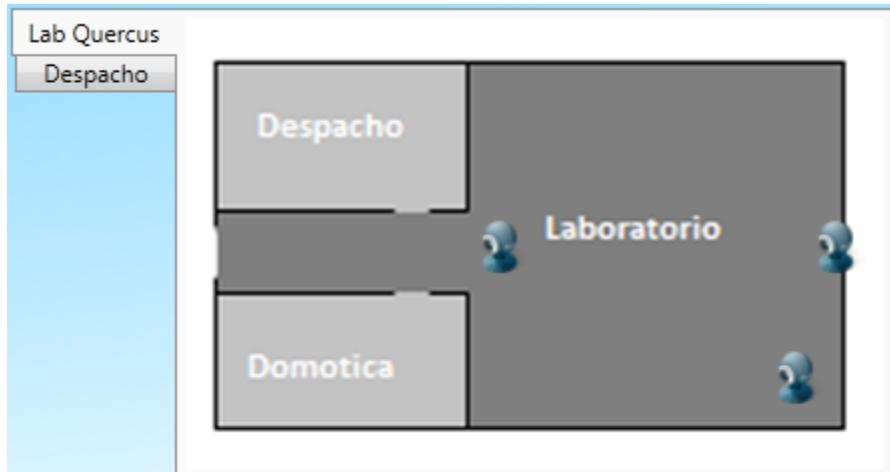


Cuando la aplicación haya iniciado lo primero que debemos hacer es identificarnos en el sistema a través de la interfaz que se muestra a continuación.



Una vez identificados, el interfaz de usuario del servidor(construido con WPF en VS) se encuentra dividido en siete pestañas.

#### 5.1.1.1 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 visualizará su contenido (streaming de vídeo para las cámaras)

### 5.1.1.2 Servicios

The screenshot shows a configuration interface for a service named "GenericAxisController". On the left, there is a sidebar with options: Timer, Mail, EssendexSMS, Receptor TCP, Quercus 0 Recorder, and Grabador Generico. The "Grabador Generico" option is selected and highlighted in blue. The main panel displays the following details:

- Nombre:** Grabador Generico
- Modo:** Recording
- Camaras:**
  - Quercus 0
  - Quercus 1
  - Quercus 2

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. La interfaz individual de cada servicio se encuentra explicada en el manual de la aplicación de modelado ya que es en esta aplicación en la que se decide qué servicios se incluirán en el servidor.

### 5.1.1.3 Dispositivos

The screenshot shows a configuration interface for a device named "CAMARA Axis". On the left, there is a sidebar with options: Quercus 0, Quercus 1, Quercus 2, and Sensor Puerta. The "Quercus 0" option is selected and highlighted in blue. The main panel displays the following details:

- Nombre:** Quercus 0
- Direccion ip:** 158.49.245.162
- Puerto:** 900
- Audio:** True
- Emision de Sonido:** False
- PTZ:** False

At the bottom right is a blue button labeled "Abrir".

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

### 5.1.1.4 Usuarios

The screenshot shows a configuration interface for a user account named "USUARIO". On the left, there is a sidebar with options: Root, Guest, Julio, and Chema. The "Root" option is selected and highlighted in blue. The main panel displays the following details:

- Nombre:** Root

At the bottom right are two buttons: "Eliminar" (Grayed Out) and "Administrar" (highlighted in blue).

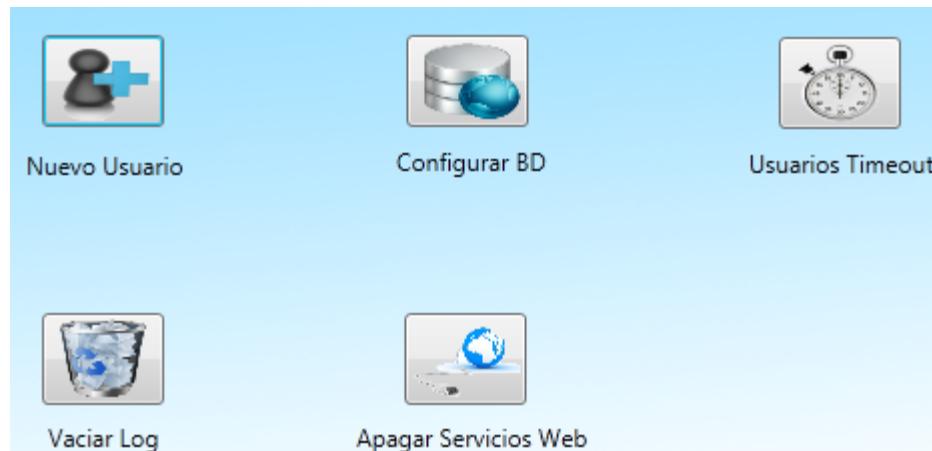
Esta pestaña permite visualizar y configurar los usuarios que tiene el sistema así como crear nuevos. Al hacer click en Administrar un usuario podremos cambiar los roles que tiene asociados, teniendo en cuenta que al activar un rol que tiene algún subrol, los subroles se asignarán también.

#### 5.1.1.5 Log

26/05/2012	23:41:00 -> Grabacion realizada en la camara [Quercus 0]
23/05/2012	23:40:11 -> Configuracion de la base de datos cargada desde config.xml.
22/05/2012	23:40:11 -> Inicio de la aplicacion.
21/05/2012	23:39:53 -> Cierre de la aplicacion
20/05/2012	23:37:00 -> Grabacion realizada en la camara [Quercus 0]
19/05/2012	23:35:00 -> Captura de imagen en la camara [Quercus 0]
18/05/2012	23:33:43 -> Configuracion de la base de datos cargada desde config.xml.
	23:33:43 -> Inicio de la aplicacion.

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

#### 5.1.1.6 Configuración



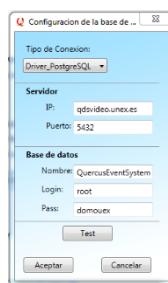
En este panel podemos acceder a la configuración general del sistema. Nos permite crear usuarios, cambiar el intervalo de tiempo tras el que la sesión de un usuario expira, vaciar el log, apagar/encender los servicios web y configurar la conexión a la base de datos. Éste último dispone de un pequeño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

#### 5.1.2 Configuración de la base de datos

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

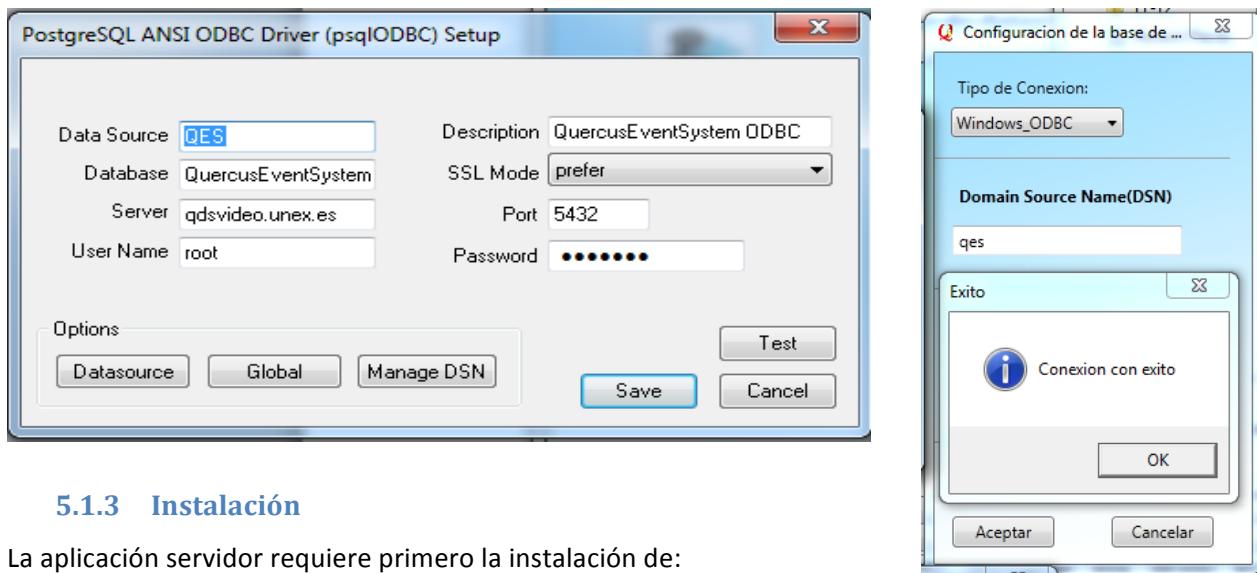
- Driver PostgreSQL
- Windows ODBC

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.



### 5.1.3 Instalación

La aplicación servidor requiere primero la instalación de:

- .Net Framework 4.0
- Sistema Gestor de Base de Datos
- Internet InformationServices
- Axis Media Control

#### 5.1.3.1 .Net Framework 4.0

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

#### 5.1.3.2 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/>

### *5.1.3.3 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”.

### *5.1.3.4 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](http://www.axis.com/techsup/cam_servers/dev/activex.htm)

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

## **5.1.4 Configuración de las cámaras**

### *5.1.4.1 Configuración*

La configuración de la cámara se realiza a través de la pestaña **| Setup |** una vez se ha entrado en la ip de la cámara a través del navegador.

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.
- Vídeo&Image: Opciones de vídeo como resolución, tipo de vídeo, etc...
- EventConfig: Configuración para detección de eventos y control de estos.

### *5.1.4.2 Acceso a las cámaras*

El acceso a la cámara desde la aplicación servidor se realiza a través del ActiveX Axis Media Control proporcionando acceso al stream de vídeo y audio a la vez que métodos para realizar grabaciones, captura de imágenes, etc...

### *5.1.4.3 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 que implica que el servidor tiene

un cliente 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 detección de movimiento o cross line, se recomienda el formato: ":" Tipo de evento ":" Dispositivo ":" Información 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.

## 5.2 Manual del servidor web

Este es el manual para todos los usuarios que deseen acceder al sistema a través del WebSite que éste ofrece.

### 5.2.1 Descripción de las páginas

A continuación se detallan las páginas que contiene el sistema describiendo su funcionalidad y modo de uso.

#### 5.2.1.1 Index.html

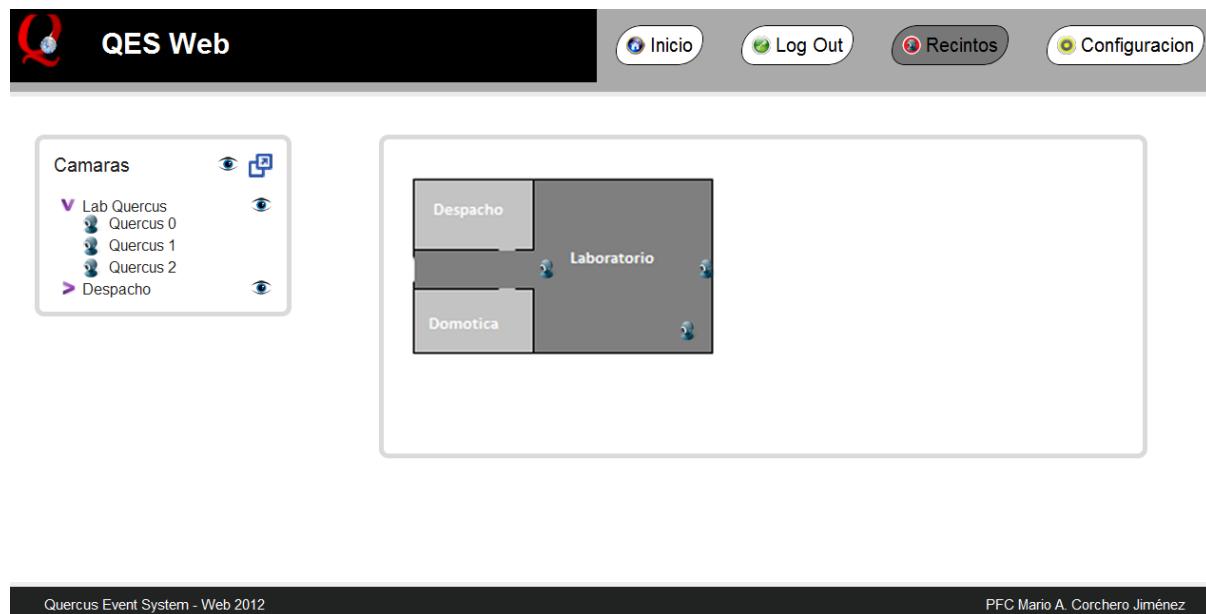
The screenshot shows the homepage of the QES Web application. At the top, there's a header bar with a logo on the left and four buttons: 'Inicio', 'Log Out', 'Recintos', and 'Configuracion'. Below the header, on the left, is a sidebar with a 'Enlaces' section containing three items: 'Unex', 'Epc', and 'Quercus'. The main content area features a large red stylized 'Q' logo on the left and a 'Bienvenido' (Welcome) message on the right. The message reads: 'Bienvenido a la aplicacion web del sistema gestor de eventos y gestor de vigilancia. Esta aplicacion esta orientada a la visualizacion de camaras y dispositivos instalados y configurados previamente en el sistema. Para acceder a la visualizacion de estos es necesario disponer de un usuario y una contraseña creados en el servidor, si no dispone de estos datos, contacte con el administrador.' At the bottom of the page, there's a footer bar with the text 'Quercus Event System - Web 2012' on the left and 'PFC Mario A. Corchero Jiménez' on the right.

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

### 5.2.1.2 Login.html

Esta página permite al usuario autenticarse en el sistema. Cuando se introduce el usuario y la contraseña, éstas 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 esté conectado. Es totalmente necesario que el servidor esté levantado para proceder a la autenticación, en caso contrario se obtendrá un mensaje: "Error al comunicar con el servidor".

### 5.2.1.3 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 cómo se muestran. Si hacemos click en el ícono abriremos todos los dispositivos dentro de un recinto o del sistema entero. Arriba a la derecha, tenemos un ícono que nos permite cambiar el modo de vista:

- **Vista en Ventana:** Al hacer click en un recinto se mostrará el mapa de éste, 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 mostrará 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 ésta se despliegan las cámaras sobre las que hagamos click. Las cámaras pueden cambiar de posición arrastrándolas, redimensionarlas como cualquier ventana (para mantener proporción pulsaremos shift) y cerrarse haciendo doble click sobre ellas.

### 5.2.1.4 Config.html

Quercus Event System - Web 2012 PFC Mario A. Corchero Jiménez

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 está 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 salir 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 sólo si el usuario está actualmente dentro del sistema). Además, hay un botón al final del área de la derecha para crear un

usuario.

**Nuevo Usuario**

Rellene todos los campos.

Nombre del usuario	<input type="text" value=""/>
Contraseña	<input type="password"/>
Repetir Contraseña	<input type="password"/>
<input type="button" value="Aceptar"/> <input type="button" value="Cancelar"/>	

Este diálogo comprobará que los datos introducidos son correctos y creará 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.



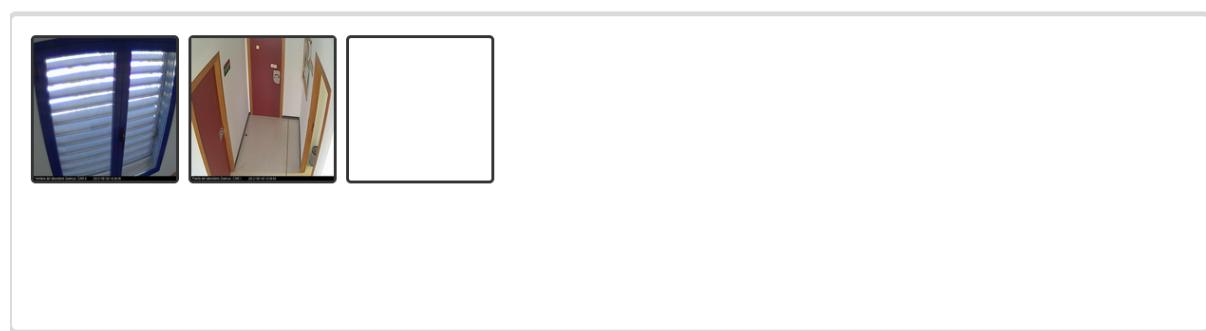
### Servicios

- Timer
- Mail
- EssendexSMS
- Receptor TCP
- Quercus 0 Recorder
- Grabador Generico

- Dispositivos: En este menú podemos visualizar todos los dispositivos así como sus detalles.

#### 5.2.1.5 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.

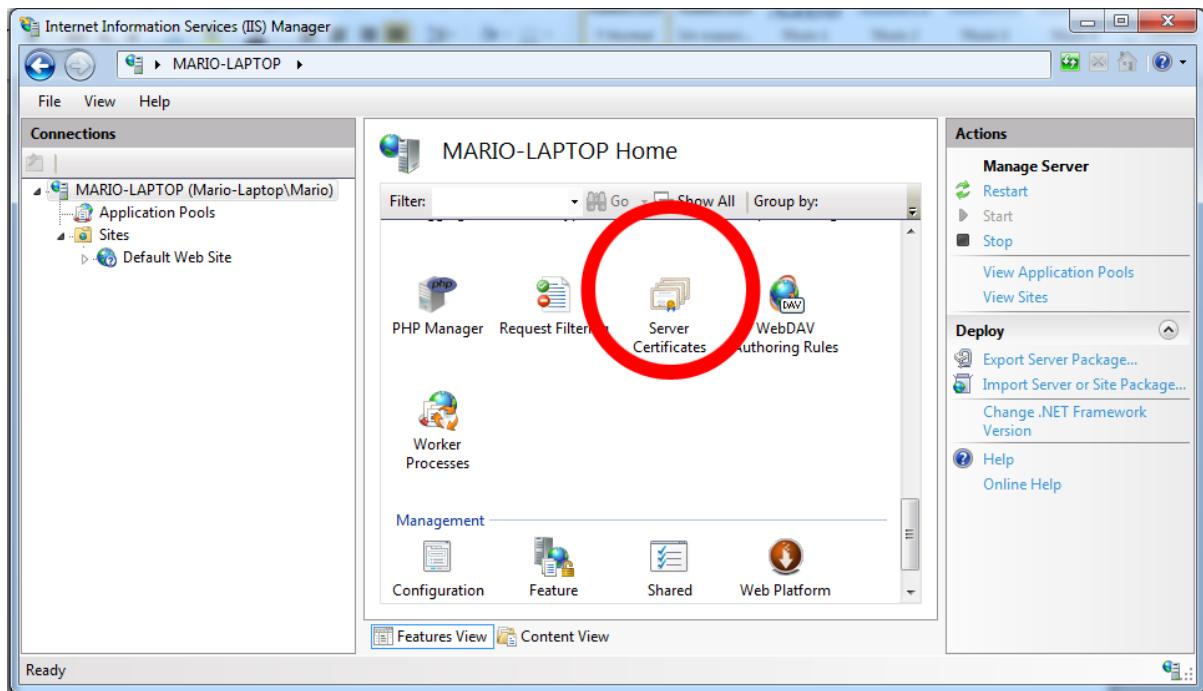
#### 5.2.2 Instalación

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

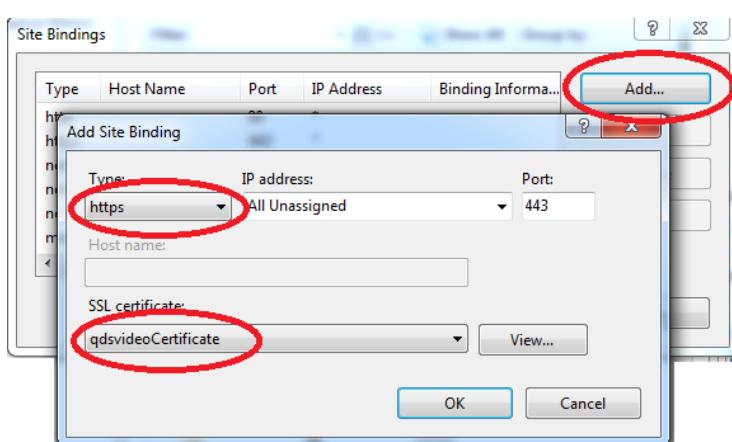
Para instalar IIS debemos ir al panel de control de Windows y a la opción “Activar/desactivar características de Windows”, y una vez en esta ventana deberemos activar el checkbox de Internet InformationServices (Servicios de Información de Internet).

#### 5.2.2.1 Certificados

Tras tener IIS instalado podremos acceder al administrador de éste, donde deberemos 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).



Haciendo click derecho en “Default 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>” significará que IIS se ha configurado correctamente.

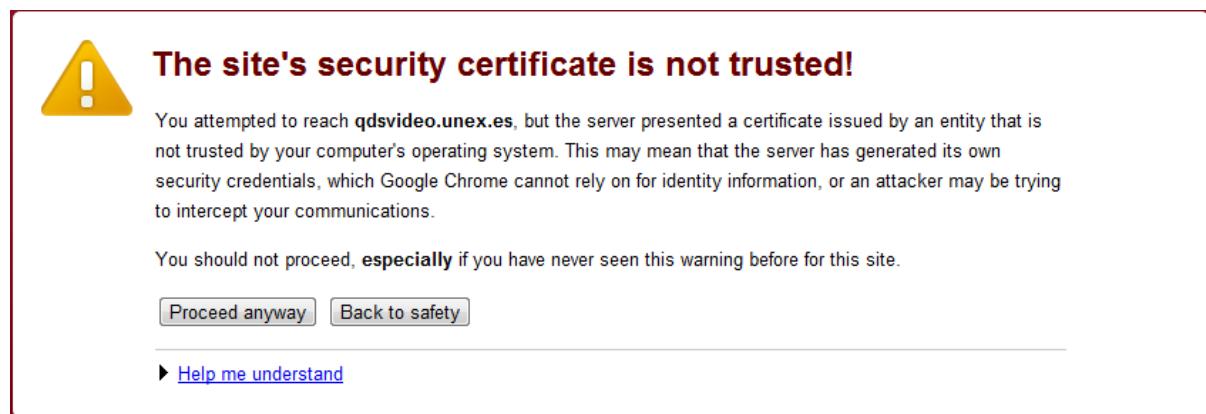
Una vez instalado y configurado IIS debemos indicar la ruta de nuestra web o mover todos los archivos a la ruta por defecto de IIS “C:\inetpub\wwwroot”.

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 servicios web 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 debemos crear un WebSite y asociarle en enlaces el puerto 80 (el sitio con HTTPS debe tenerlo desactivado). Una vez creado el sitio 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 posible que si el certificado que importamos a nuestro servidor es “self-signed” obtengamos mensajes de aviso al entrar en la web como:





There is a problem with this website's security certificate.

The security certificate presented by this website was not issued by a trusted certificate authority.  
The security certificate presented by this website was issued for a different website's address.

Security certificate problems may indicate an attempt to fool you or intercept any data you send to the server.

**We recommend that you close this webpage and do not continue to this website.**

- [Click here to close this webpage.](#)
- [Continue to this website \(not recommended\).](#)
- [More information](#)



### Esta conexión no está verificada

Ha pedido a Firefox que se conecte de forma segura a **localhost**, pero no se puede confirmar que la conexión sea segura.

Normalmente, cuando se intente conectar de forma segura, los sitios presentan información verificada para asegurar que está en el sitio correcto. Sin embargo, la identidad de este sitio no puede ser verificada.

#### ¿Qué debería hacer?

Si normalmente accede a este sitio sin problemas, este error puede estar ocurriendo porque alguien está intentando suplantar al sitio, y no debería continuar.

[¡Sácame de aquí!](#)

► [Detalles técnicos](#)

► [Entiendo los riesgos](#)



En navegadores como Firefox o Safari podemos añadir directamente el certificado a nuestra store para que este mensaje no aparezca más. Pero si utilizamos Chrome o Internet Explorer deberemos buscar el certificado y añadirlo manualmente al navegador desde las opciones.

#### 5.2.2.2 Configuración

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:

```
varpusherKey = '9a3fe2f64106d87e0291';
varpusherChannel = 'QESTest';
varsoundFile = "beep.wav";
```

Con los datos obtenidos al crear una cuenta en “<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[23]:

	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 incluir varios ficheros con diferentes formatos, pero por comodidad para el usuario, sólo se requiere uno que sea wav ya que es actualmente el más compatible.

## 5.3 Manual del DSL

Este es el manual que el usuario encargado de la configuración y despliegue del sistema debe conocer.

Para la creación del sistema es necesario seguir los siguientes pasos:

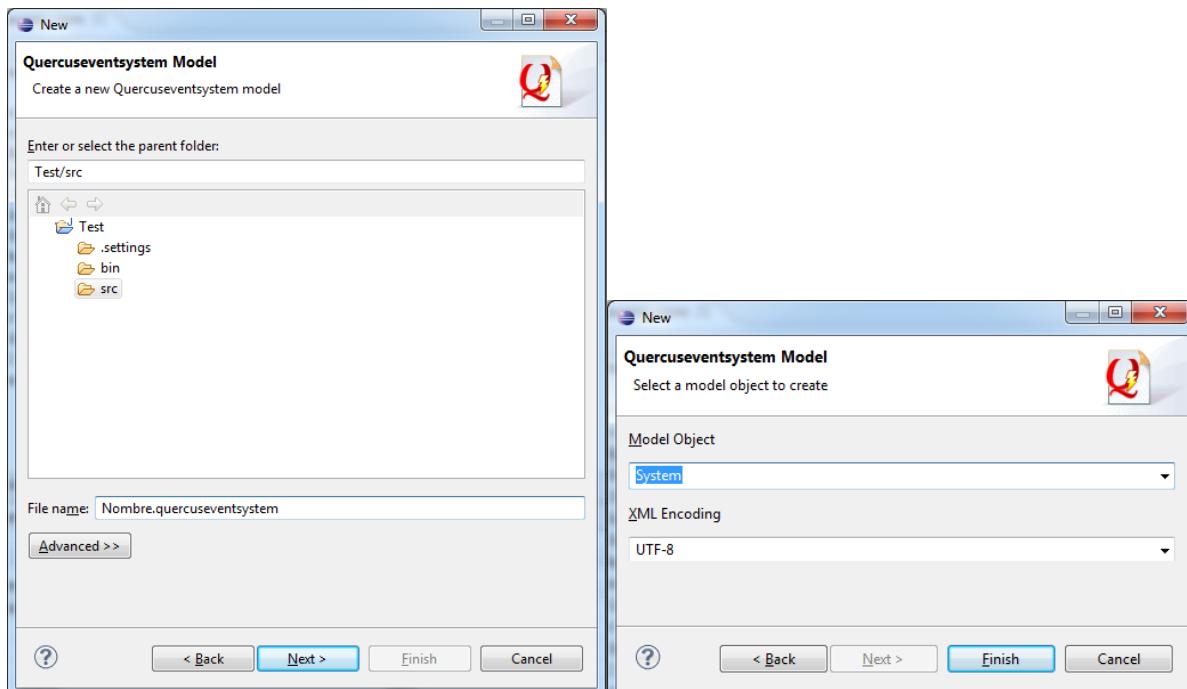
- (1) Creación de un modelo.
- (2) Verificación del modelo.
- (3) Generación del modelo.
- (4) Despliegue del sistema.

A continuación se detallan las herramientas disponibles para realizar estos pasos.

### 5.3.1 Editor en árbol

Para la creación de un modelo a través de este editor, sólo debemos, teniendo un proyecto abierto y seleccionado, crear un nuevo elemento (Ctrl + N), pinchar en Quercuseventsyste Model QuercusEventSystem Model y el asistente nos guiará en el proceso de creación. Si no encontramos la opción, podemos utilizar el cuadro de búsqueda situado en la parte superior etiquetado como "Wizards".

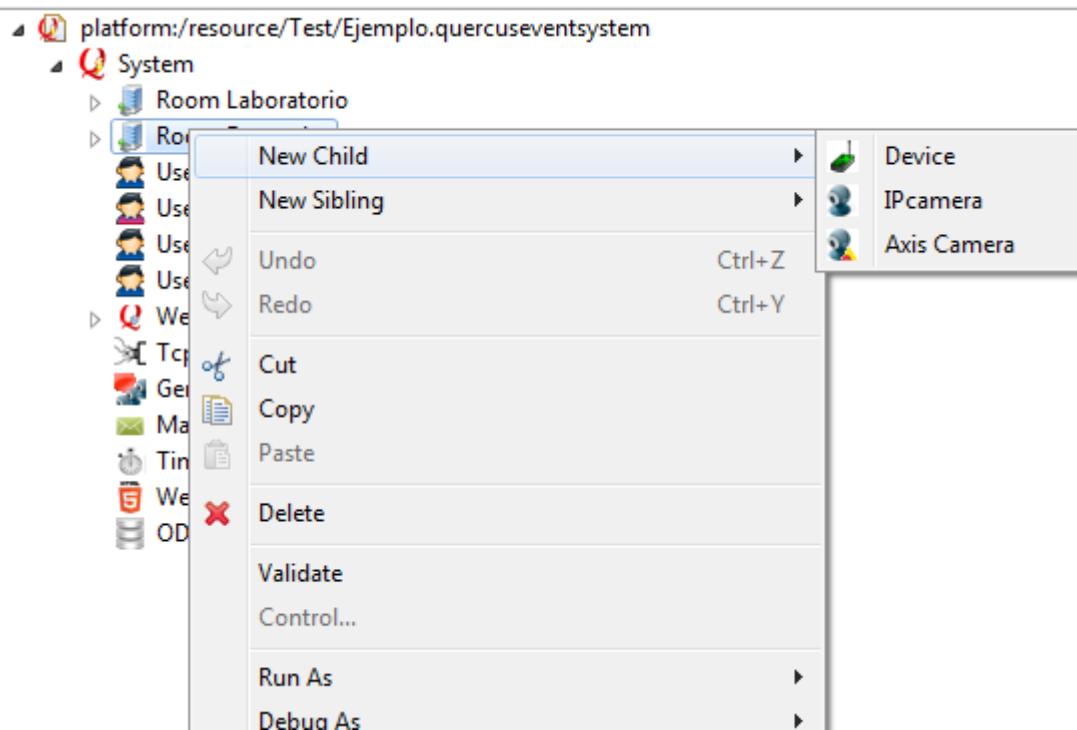
En el asistente deberemos elegir la ruta y el nombre del modelo junto con el Objeto Root de éste, en nuestro caso “system” (si se desea crear el modelo para la carga de los roles estándares elegir StandardRoleRoot).



Tras esto tendremos nuestro fichero de modelo creado, el cual podremos abrir con doble click o click derecho, open with, QuercusEventSystem Model Editor.

El editor permite crear elementos uno a uno a través de click derecho sobre un nodo y pinchando en “new Child” para crear un elemento que depende del actual o “new Sibling” para crear un elemento a la misma altura que el actual.

Por lo que si hiciéramos click derecho sobre un Room en el modelo veríamos la siguiente imagen:



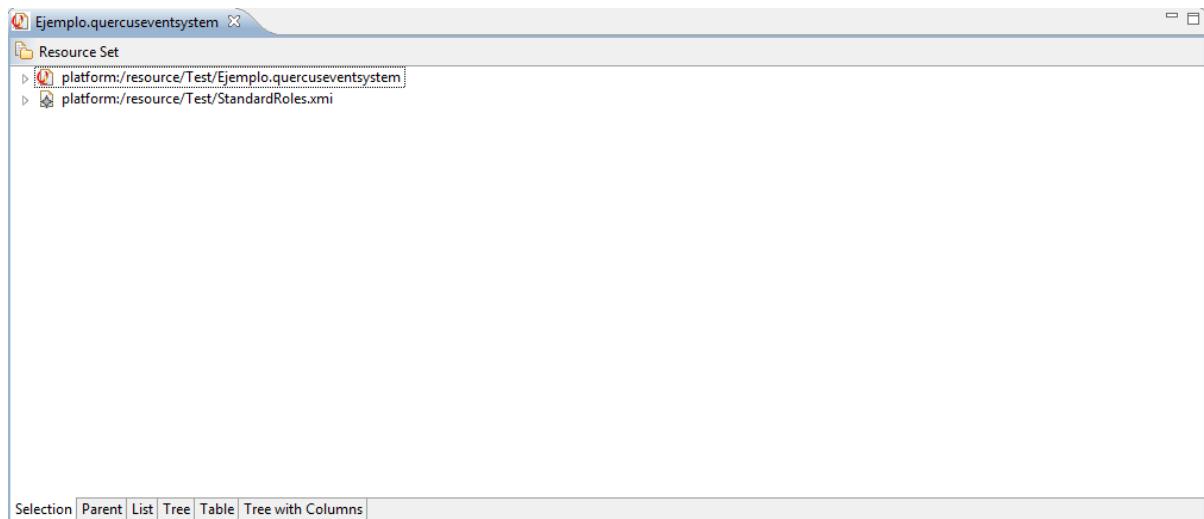
Para cada elemento que creemos podremos definir su propiedades en la ventana de propiedades cuando el elemento este seleccionado. En caso de que esta ventana no aparezca, hacemos click derecho en el elemento y pinchamos en “show propertiesview”. Esta ventana muestra los atributos del elemento seleccionado y permite cambiarlos.

Properties	
Property	Value
Name	Root
Password	domouex
Super Role	Standard Role AdminRole

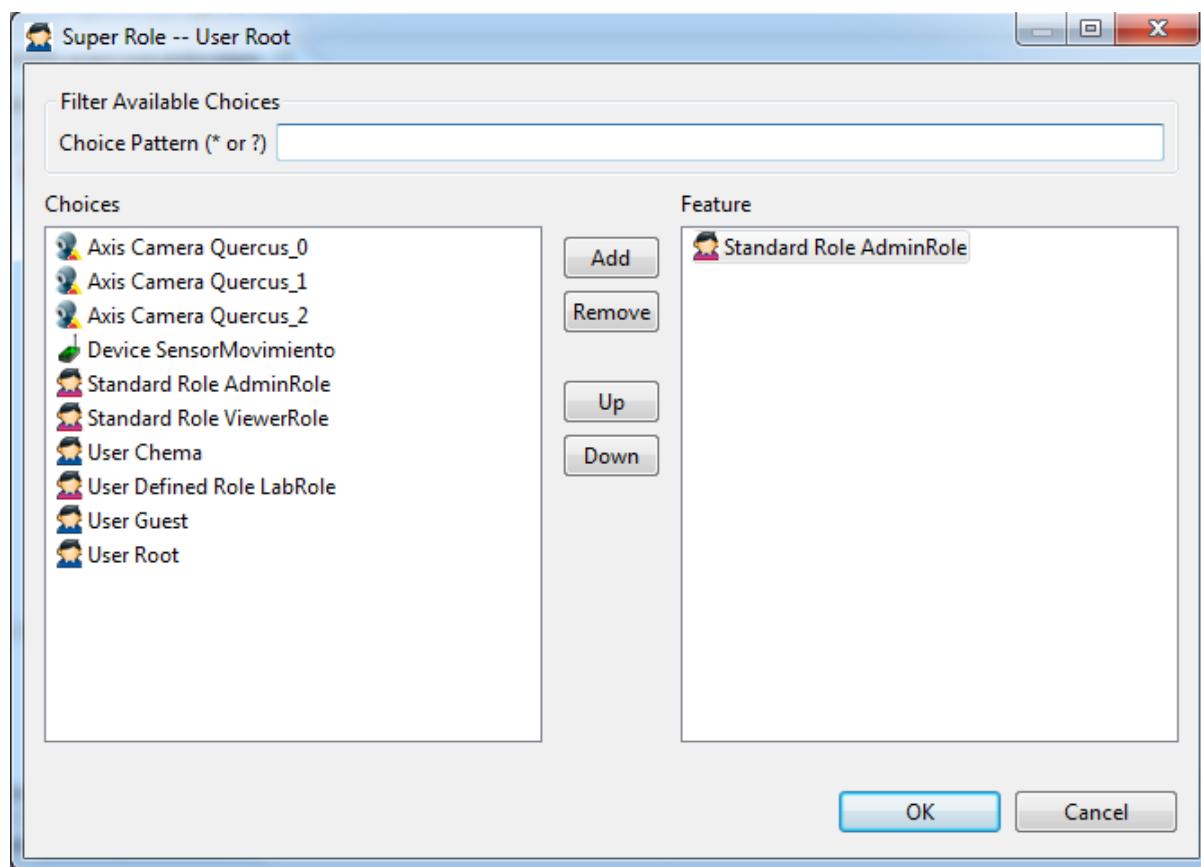
Para nuestro sistema, necesitaremos importar un archivo de recursos con los roles estándares Admin y Viewer. Para ello, deberemos hacer click derecho en cualquier parte en blanco del modelo, y pinchar en “Load Resource”. Nos aparecerá una ventana en la cual tendremos que buscar el

modelo con los roles que se proporciona con los archivos en este proyecto, o crear nosotros mismos uno siguiendo los pasos que se indicaron anteriormente pero seleccionando como RootStandardRoleRoot.

Si se ha cargado con éxito, en el editor del modelo deben aparecernos dos recursos. El modelo que estamos diseñando y el modelo de los roles cargados tal y como se muestra en la siguiente imagen.

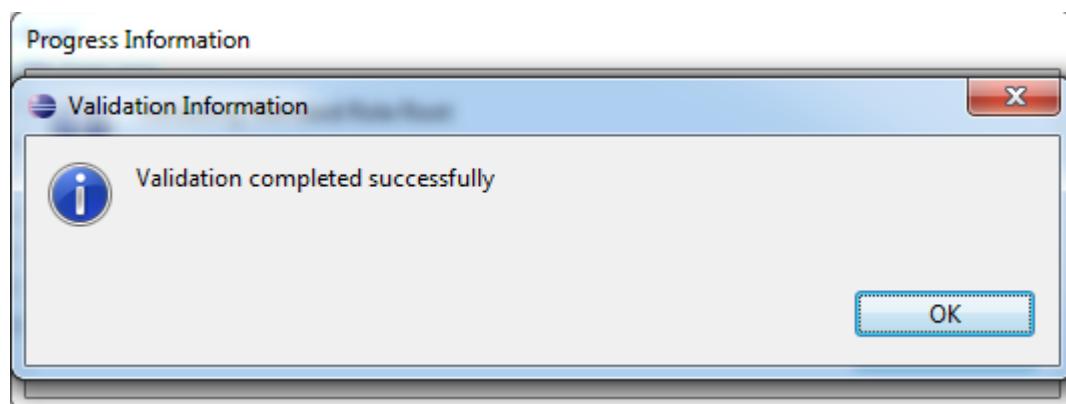


Además, tras incluir el modelo de los roles estándares éstos aparecerán como opciones cuando pinchemos en superroles en la ventana de propiedades de un usuario o un rol definido por usuario.



Como podemos observar, el aspecto grafico del editor a sido personalizado, intentando que sea lo más cómodo e intuitivo posible a través de iconos y otras facilidades.

Es importante que validemos el modelo según vayamos añadiendo elementos y estableciendo sus propiedades, puesto que esto nos confirmará que el modelo tiene los elementos que requiere y que los campos han sido rellenados de forma correcta. Para ello, debemos seleccionar el nodo más alto, system y hacer click derecho pinchando después en Validate. Si todo está correcto aparecerá un mensaje como el que se muestra a continuación:



En caso contrario, recibiremos un mensaje de error, en el cual, pinchando en details, obtendremos una lista de los errores que deben solucionarse para que el modelo sea válido.

Este archivo realmente es un archivo XML y si lo deseamos, podemos abrirlo con un editor de texto normal para ver su contenido:

```

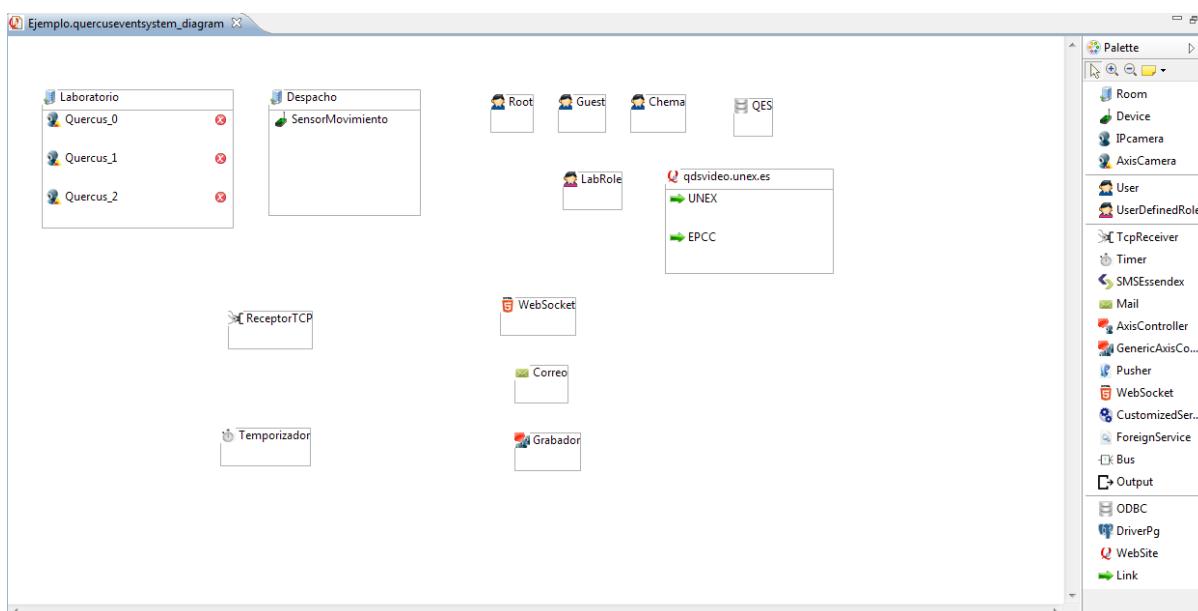
<?xml version="1.0" encoding="UTF-8"?>
<quercuseventsyste...System xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:quercuseventsyste...="http://quercuseventsyste.../1.0">
  <rooms map="pinvestigacion.png" name="Laboratorio">
    <devices xsi:type="quercuseventsyste...AxisCamera" name="Quercus_0"/>
    <devices xsi:type="quercuseventsyste...AxisCamera" name="Quercus_1"/>
    <devices xsi:type="quercuseventsyste...AxisCamera" name="Quercus_2"/>
  </rooms>
  <rooms map="aselcom.png" name="Despacho">
    <devices name="SensorMovimiento"/>
  </rooms>
  <roles xsi:type="quercuseventsyste...User" name="Root" superRole="quercuseventsyste...StandardRole StandardRoles.xmi#@sRoles.0" password="1234"/>
  <roles name="LabRole" superRole="#Quercus_0 #Quercus_1 #Quercus_2"/>
  <roles xsi:type="quercuseventsyste...User" name="Chema" superRole="quercuseventsyste...StandardRole StandardRoles.xmi#@sRoles.1" password="1234"/>
  <roles xsi:type="quercuseventsyste...User" name="Guest" superRole="#LabRole" password="1234"/>
  <website domain="qdsvideo.unex.es">
    <links title="UNEX" url="http://unex.es"/>
    <links title="EPCC" url="http://epcc.unex.es"/>
  </website>
  <servicios xsi:type="quercuseventsyste...TcpReceiver" name="ReceptorTCP" sumideros="#Correo #WebSocket"/>
  <servicios xsi:type="quercuseventsyste...GenericAxisController" name="Grabador" time="15"/>
  <servicios xsi:type="quercuseventsyste...Mail" name="Correo"/>
  <servicios xsi:type="quercuseventsyste...Timer" name="Temporizador" sumideros="#Grabador"/>
  <servicios xsi:type="quercuseventsyste...WebSocket" name="WebSocket"/>
  <clientDB xsi:type="quercuseventsyste...ODBC" dsn="QES"/>
</quercuseventsyste...System>

```

Como podemos ver, se puede entender bastante bien el fichero e incluso ser editado para pequeñas correcciones de este modo.

### 5.3.2 Editor Grafico

El editor tiene el siguiente aspecto:

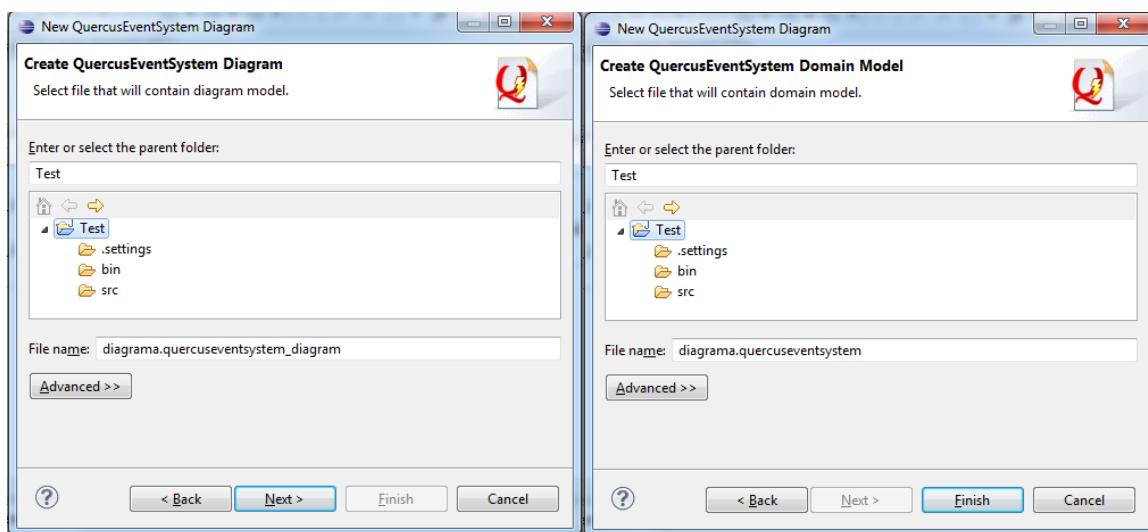


Hay dos formas de editar gráficamente un modelo. La primera haciendo click derecho sobre un modelo ya existente y pinchar en "Initializequercuseventsyste...\_diagramDiagram File" lo cual nos

creará un fichero de diagrama y además aparecerá toda la información que tengamos ya añadida en el modelo.

La otra opción consiste en crear los dos ficheros a la vez, teniendo el proyecto seleccionado, creamos un nuevo elemento (Ctrl + N), y buscamos QuercusEventSystem Diagram.

 **QuercusEventSystem Diagram**. Después un asistente nos guiará en el proceso de creación:



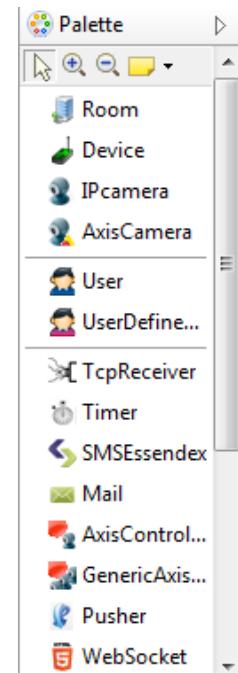
Creando así dos ficheros, uno de diagrama y otro con el modelo.

Dentro del editor gráfico, para crear elementos, deberemos pinchar en el elemento en la paleta de la derecha y luego situarlo en el contenedor principal.

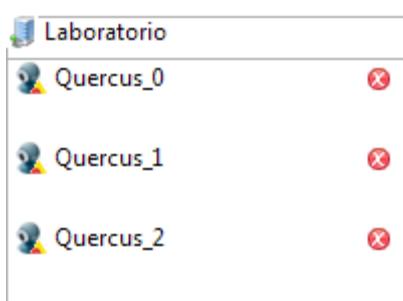
Es importante tener en cuenta que hay ciertos elementos con limitaciones a la hora de ser utilizados. Por ejemplo sólo puede utilizarse un conector de base de datos y algunos elementos como Output, Dispositivos o Enlaces deben ir dentro de Bus, Recinto y WebSite respectivamente.

En la paleta encontramos herramientas también para añadir anotaciones, hacer zoom y mover elementos de sitio.

Para acceder a la vista de propiedades de cada elemento, deberemos hacer click derecho sobre él y pinchar en "Show Properties View".



Es importante recordar que para que podamos ver los roles estándares, deberemos cargar un recurso con éstos a través de click derecho, load resource e indicar su ruta.



Por último, una vez completado el modelo, deberemos validararlo. Esta acción puede realizarse directamente desde el editor gráfico en el menú Edit, Validate, al lado de File.

En caso de que el diagrama contenga algún error, nos aparecerá un mensaje y se nos mostrará una X roja en el elemento que dé problemas.

### 5.3.3 Editor Textual

#### 5.3.3.1 Crear un fichero con el DSL

Para crear un fichero con las opciones que ofrece el editor gráfico, sólo necesitamos hacer click en el segundo botón sobre el proyecto y añadir un “file” normal pero nombrándolo con la extensión “.quesdsl” para que al abrirlo (haciendo dobleclick) se lance el editor textual.

#### 5.3.3.2 Conversión a Modelo

Para pasar de fichero de texto DSL a un fichero XMI conforme al meta modelo diseñado y viceversa, se han creado dos clases en un paquete llamado serialization. Estas dos clases, XMI2QESDSL y QESDSL2XMI pueden ser ejecutadas configurando la ruta de los archivos para realizar esta transformación.

#### 5.3.3.3 Herramientas del editor

- Auto Completado. Pulsando Ctrl + Espacio el editor ayuda al usuario mostrando las opciones posible o completando una palabra.
- Detección de errores sintácticos y gramaticales mientras se escribe. El editor informa de los posibles fallos sintácticos y gramáticos mientras se escribe.
- Validación semántica. Una vez terminado el modelo, se puede hacer segundo click y validarla contra el meta modelo, comprobando también todas las restricciones.
- Coloreado de sintaxis.
- Ayuda en la corrección de errores. Cuando se detecta un error, el editor además intenta ofrecer soluciones alternativas.

### 5.3.4 Generación de código

Para la generación de código se dispone de un proyecto JET que dados unos ficheros de entrada genera todo el sistema y un archivo .bat de despliegue.

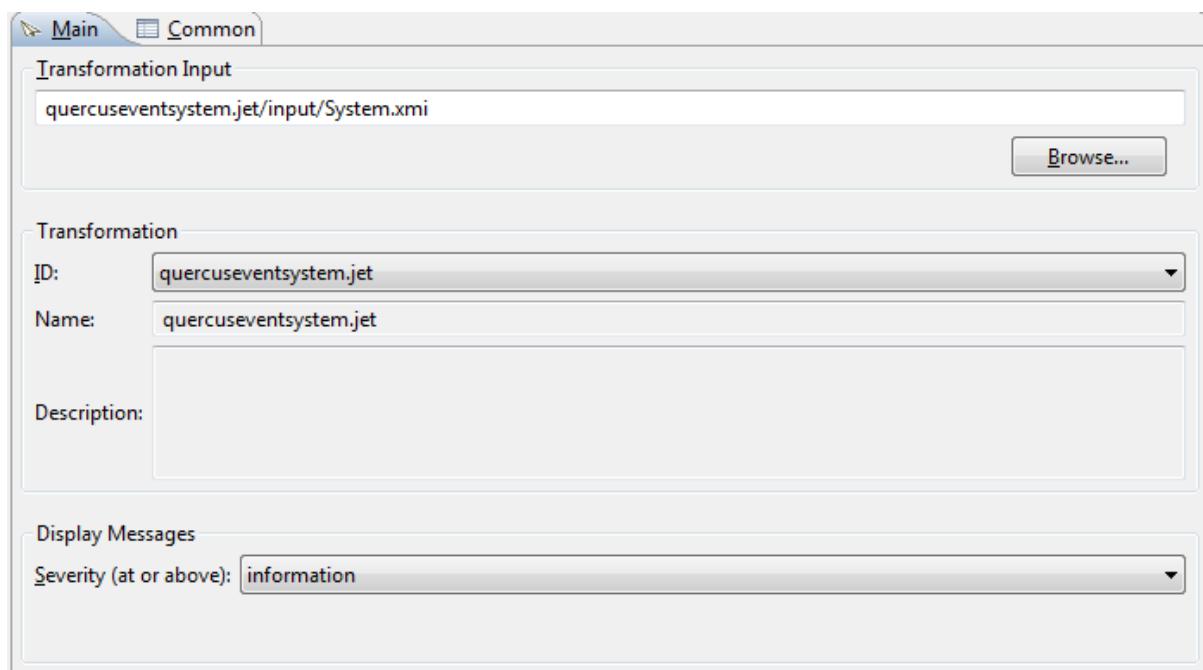
#### 5.3.4.1 Ficheros de entrada

Para la correcta generación de la solución, debemos incluir en la carpeta input los siguientes ficheros:

- Imágenes correspondientes a los mapas de los recintos.
- Archivos DLL en los que se encuentran definidos los servicios externos.
- Imagen del logo de la web.
- Modelo de entrada.

#### 5.3.4.2 Ejecución de la transformación

Para ejecutar la transformación debemos ejecutar el proyecto como “JET Transformation” en eclipse, indicando en la configuración el modelo de entrada incluido en input.



Tras ejecutarlo, la carpeta output será debidamente generada conforme al modelo introducido

#### 5.3.4.3 Generación de nuevos Servicios

Para generar los DLL, se dispone de un archivo baseDLL.dll con el que hay que generar en Visual Studio una solución de tipo Biblioteca de Clases(importando baseDLL.dll) cuya generación crea el fichero necesario.

### 5.3.5 Despliegue del sistema

Una vez generado el sistema, puede encontrarse en la carpeta output un archivo nombrado QES.bat. Este archivo debe ser ejecutado (ser recomienda como administrador) para iniciar el despliegue del sistema.

#### 5.3.5.1 Compilación

Este archivo bat comenzará con la copia de los archivos necesarios para el sistema tras lo que realizará la compilación del proyecto. Si algún error ocurre en la compilación (al añadir un nuevo servicio) será mostrado en rojo y no se permitirá el acceso al menú de instalación.

```
BUILDING...
  1 file(s) copied.
  1 file(s) copied.
  1 file(s) copied.
  1 file(s) copied.
  1 file(s) copied.
QES -> D:\Mis Documentos\workspace\EclipseMDA\quercuseventsyste...jet\output\QuercusEventSystem\QuercusEventSystem\bin\Release\QuercusEventSystem.exe
```

#### 5.3.5.2 Menú principal

En este menú se muestran las principales opciones de despliegue.

```
PRESS 1, 2, 3 OR 4 to select your task, or 5 to EXIT.
-----
1 - Start Installer
2 - Execute Quercus Server
3 - Open the output folder
4 - Open C# Solution at VS (VS Required)
5 - EXIT
Type 1, 2, 3, 4 or 5 then press ENTER:
```

#### 5.3.5.3 Menú Instalación

Al elegir la primera opción en el menú principal accederemos al menú de instalación, el cual contiene las principales aplicaciones necesarias para el sistema (salvo IIS que debe ser instalado manualmente).

```
PRESS 1, 2, 3, 4, 5 OR 6 to select the program to install, or 7 to EXIT.
-----
1 - QuercusEventSystem
2 - QES Web
3 - Axis Media Control
4 - .Net Framework 4.0
5 - Open ODBC Configuration
6 - PostgreSQL
7 - Exit
Type 1, 2, 3, 4, 5, 6 or 7 then press ENTER:_
```

### 5.3.6 Diseño de los servicios

A continuación se detalla el diseño de cada uno de los servicios que puede definir el usuario del DSL para estar disponibles en el sistema. Es importante que el usuario tenga conocimiento de cómo están definidos estos servicios para utilizarlos correctamente y por si es necesaria alguna pequeña modificación.

#### 5.3.6.1 Bus

Se disponen de dos tipos de este servicio, bus, el cual tiene una entrada (QUESTail) y múltiples salidas (QESHead). La idea es crear un servicio que distribuya los eventos en base a una condición.

Los dos tipos de bus son:

##### TypeBus

Este tipo de bus reparte los eventos que le entran por sus salidas en función del tipo del evento. Es importante tener en cuenta que el tipo debe coincidir exactamente, es decir, el evento se propagará por la salida que encaje con el tipo del evento en su totalidad, no propagándose por sus subtipos o supertipos.

Para conectar una de las salidas del bus con un consumidor basta con llamar a su método publico “Conectar”:

```
void Conectar<T>(QUESTail<T> cola)
```

Siendo T el tipo de evento que define a la salida y cola el consumidor que recibirá los eventos.

##### ConditionBus

Este tipo de bus clasifica y distribuye los eventos a partir de condiciones que se le indican a la hora de realizar una conexión. La condición se pasa como un delegado (llamado DelegadoCondicion) que debe devolver un booleano al recibir un Evento. Así el método conectar se define de la forma:

```
public void Conectar(DelegadoCondicion cond, QUESTail<T> salida)
```

Gracias a la flexibilidad de C#, cond puede ser desde el nombre de un método a una expresión lambda como la siguiente:

```
X => X is QESDispositivoEvent &&
(X as QESDispositivoEvent).Dispositivo.Nombre = "Quercus 0"
```

Para obtener más información sobre funciones lambda en C# consulte:

<http://msdn.microsoft.com/en-us/library/bb397687.aspx>

### 5.3.6.2 Controladores Axis

Sirviéndonos del ActiveX que ofrece Axis Communications®, hay varias clases que permiten realizar grabaciones y captura de imágenes de cámaras Axis. Todas las clases son consumidores de eventos y pueden configurarse para capturar imágenes y realizar grabaciones temporizadas cuya franja de tiempo puede ser establecida también como parámetro.

#### AxisController

Esta clase sin importar qué le llegue como evento realizará una grabación/captura de imagen con los parámetros con los que está configurada.

El constructor de la clase permite definir todos estos parámetros, siendo su firma:

```
public AxisController(string ip, string login, string pass, int time = 15, Mode modo =
Mode.Recording, string port = "80", string name = null, bool showGrabacion =
true, ILogable log = null, string path = "")
```

Las propiedades de AxisController son:

- IP: dirección ip de la cámara
- Login: usuario de la cámara
- Pass: contraseña de la cámara
- Time: tiempo de grabación. 15 por defecto.
- Modo: Modo de respuesta.
  - Recording(Modo por defecto)
  - Photo
  - Both
- Port: puerto de escucha de la cámara. 80 por defecto
- Name: Nombre que aparecerá en la ventana de grabación.
- showGrabacion: indica si debe mostrarse la ventana de grabación al iniciarse una. True por defecto.
- Log: instancia del log para producir entradas. Null por defecto.
- Path: Carpeta donde se almacenaran las grabaciones.

- Enabled: Estado del servicio. Activo por defecto.

Hay una interfaz que se debe asociar a esta clase, UIAxisController que permite ver algunos detalles del servicio y configurar otros.



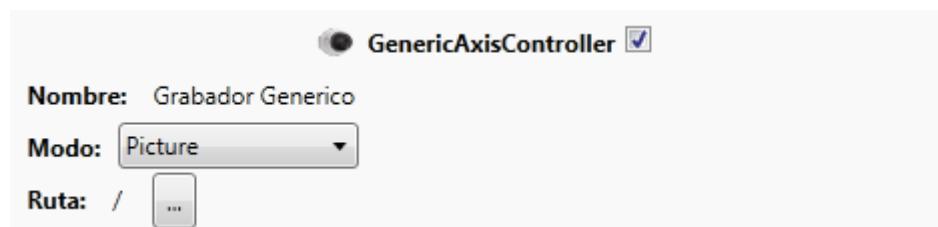
### AxisGenericController

Esta clase permite realizar grabaciones y captura de imágenes al igual que AxisController pero con una diferencia, los parámetros de grabación son transmitidos a través del evento recibido. Es por ello que el constructor de esta clase es más sencillo.

```
public GenericAxisController(int time, AxisController.Mode modo, bool showGrabacion =
false, ILogable log = null, string path = "")
```

Cuando esta clase recibe un evento de tipo QESAxisCameraEvent obtiene el dispositivo de éste y en caso de que el campo Info del evento contenga un entero se tomará como el intervalo de grabación elegido.

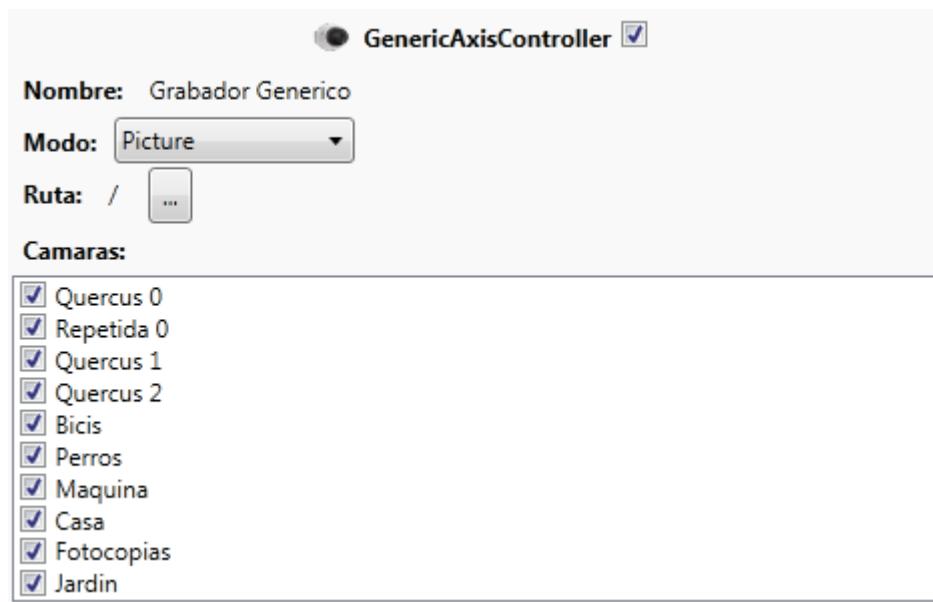
La interfaz del servicio, UIAxisController, es la siguiente:



### AxisConditionalController

Esta clase implementa un servicio muy similar a GenericAxisController pero con la diferencia de que contiene una lista de las cámaras activas/desactivadas para grabación. Es decir, se puede indicar que aunque reciba un evento para realizar una grabación en una cámara determinada, no se realice.

Esta propiedad se puede configurar en la interfaz tal y como se muestra a continuación:

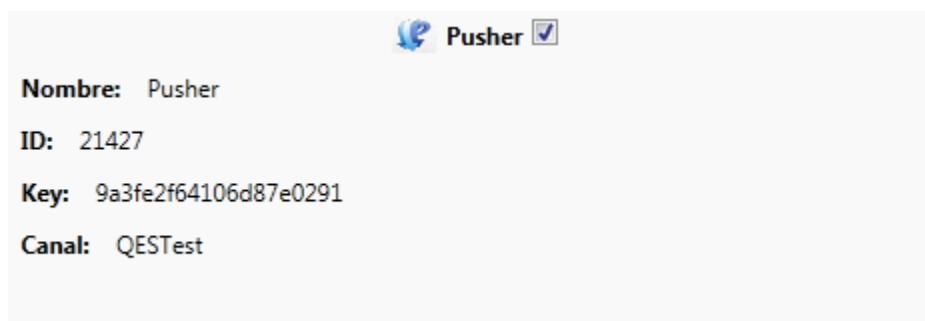


#### 5.3.6.3 Pusher

Pusher es un servicio que hace uso de la herramienta Pusher, una API que permite trabajar con comunicación Push servidor-cliente con una gran comodidad.

Dispone de un método, `EnviarCameraNotification` que envía un mensaje al canal determinado (en Pusher.com) en el constructor. Este método es lanzado cuando el servicio Pusher recibe un evento (dentro del sistema) de tipo `QESAxisCameraEvent`, del cual obtiene el nombre de la cámara y el contenido del mensaje.

Su interfaz, `UIPusher`, se limita a mostrar los datos de configuración:

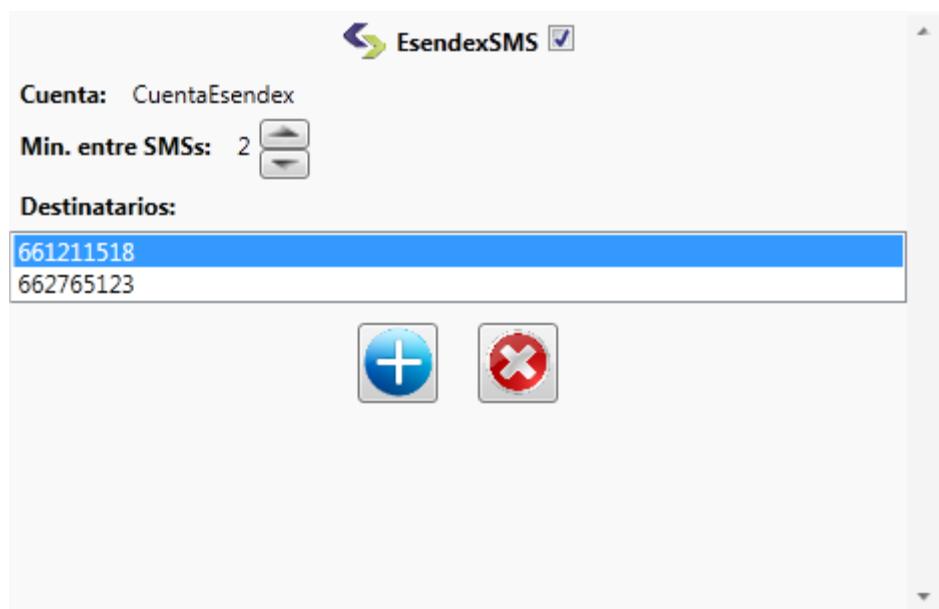


#### 5.3.6.4 QESEsendexSMS

Es un servicio que nos permite comunicarnos con la API de Esendex a través de una cuenta para el envío de mensajes a móviles. Al recibir el servicio un evento, éste se limita a mandar la información contenida en Info del evento a todos los números registrados en el servicio. El servicio permite también la definición de un parámetro para indicar el intervalo en minutos que debe pasar entre el

envío de un mensaje y otro, debido a que puede ser molesto recibir demasiados mensajes seguidos para eventos parecidos.

Tanto los números de teléfonos asociados como el intervalo de envío se pueden configurar cómodamente en la interfaz UISMSEsendex como se muestra a continuación:



#### 5.3.6.5 QESMailSender

Este servicio permite enviar correos utilizando un servidor SMTP externo.

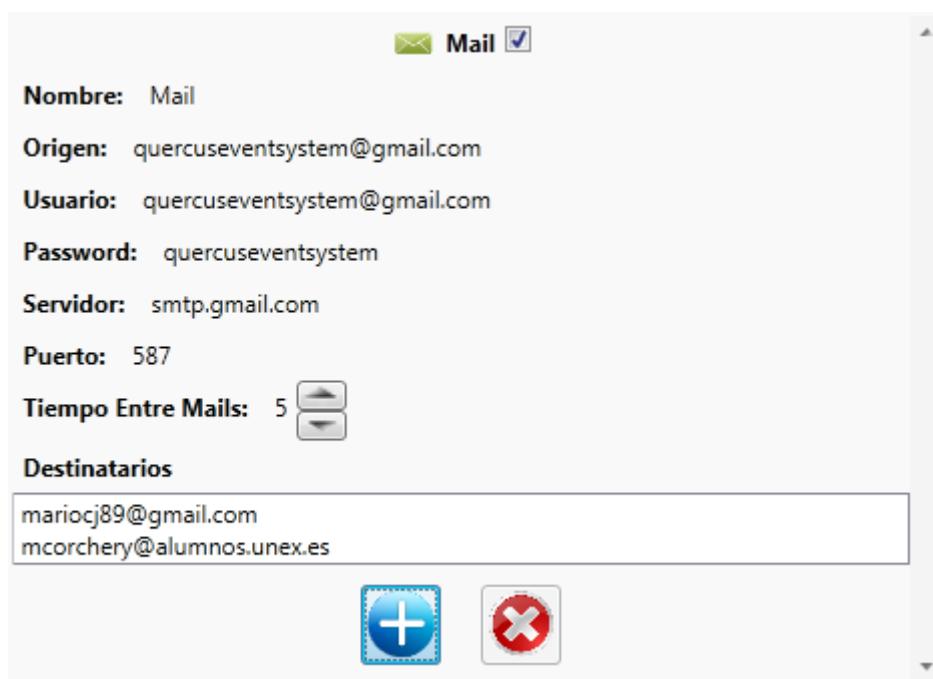
El servicio permite configurar una serie de parámetros en su constructor:

```
public QESMailSender(String origen, String usuario, String pass, String host =
"smtp.gmail.com", int puerto = 587, int step = 0, ILogable log = null)
```

Estos parámetros son:

- Origen: Correo que aparecerá como origen del mensaje.
- Usuario: Usuario en el servidor SMTP
- Pass: Contraseña en el servidor
- Host: Dirección del servidor SMTP. Gmail por defecto
- Puerto: Puerto de conexión con el servidor. 587(seguro) por defecto.
- Step: Tiempo entre correos en minutos. 0 por defecto.
- Log: instancia del log para añadir entradas. Null por defecto.

Así como en el interfaz UIMail:



### 5.3.6.6 QESSMS

Esta es una clase que utilizando un servicio web publico permite enviar mensajes a móviles de diferentes países de forma gratuita.

La especificación del servicio web se puede encontrar en:

[“<http://www.webservicex.net/ws/WSDetails.aspx?CATID=4&WSID=60>”](http://www.webservicex.net/ws/WSDetails.aspx?CATID=4&WSID=60)

La última vez que se testeo este servicio no funcionaba correctamente.

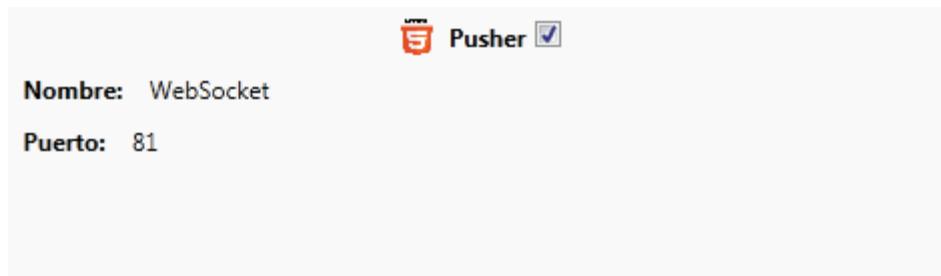
### 5.3.6.7 QESWebSocket

Este servicio, utilizando Alchemy, implementa la parte servidor de WebSockets para comunicarse con los clientes. El servicio permite configurar únicamente el puerto de salida del servidor, 81 por defecto.

El servidor utiliza la librería que permite definir los métodos con los que se tratarán los eventos que se produzcan en el lado del servidor (conexión de clientes, desconexión, etc...). La clase guarda la dirección de los clientes cuando se conectan y las borra cuando se desconectan, de esta forma, cuando se desea enviar un mensaje, se ha creado un método Broadcast que envía los mensajes uno a uno a todos los clientes.

Cuando el servicio recibe un evento QESAxisCameraEvent llama al método broadcast enviando como mensaje el nombre de la cámara indicado en el evento.

La interfaz, UIWebSocket, vale únicamente para mostrar el puerto:



#### *5.3.6.8 VAPIXController*

Esta clase implementa un servicio que permite realizar grabaciones y captura de imágenes a través de la API HTTP que ofrecen las cámaras Axis, la captura de imágenes funciona correctamente, pero las grabaciones debido a un error en las cabeceras HTTP que no fue posible solucionar no ha sido implementado correctamente, es por ello por lo que se recomienda el uso de AxisController en lugar de este servicio ya que ofrece la misma funcionalidad a través del ActiveX.

Se puede encontrar más información sobre la API HTTP en el siguiente documento:

[“http://www.axis.com/techsup/cam\\_servers/dev/cam\\_http\\_api\\_index.php”](http://www.axis.com/techsup/cam_servers/dev/cam_http_api_index.php)

#### *5.3.7 Implementar un nuevo servicio*

Antes de crear un nuevo servicio el primer paso sería intentar localizar uno existente y personalizarlo, de esta 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.

##### *5.3.7.1 Código*

A continuación se detalla el código que deberá incluir el usuario.

##### Tipo

Antes de implementar un servicio, hay que decidir qué 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>}
- `VoidRestoreFromData(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 sólo se podrá conectar con otros que tengan el mismo tipo que él, es decir, un productor de servicios QESEvent sólo 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 él cualquier clase heredada.

#### 5.3.7.2 Interfaz

Si deseamos que el servicio tenga una interfaz específica, ésta se puede crear como un “User Control” o “Control de usuario” en MS Visual Studio, MS ExpressionBlend o MS ExpressionDesign.

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

Si no se desea especificar una interfaz se asignará automáticamente la interfaz de servicio genérico, ésta sólo muestra el nombre del servicio.

## 6. FAQ del sistema

- La aplicación no envía nunca mails:
  - Comprueba que el usuario y la contraseña sean correctas en la interfaz del servicio en el servidor o en el DSL.
- No es posible acceder al servidor web, servidor no encontrado:
  - Al configurar IIS primero probar con la pantalla de bienvenida de IIS por defecto.
- No se puede acceder desde el exterior:
  - Comprobar firewall y añadir excepciones para los puertos necesarios(443)
- Acceso no permitido a IIS:
  - Quitar sólo lectura a la carpeta o copiar archivos a la carpeta de IIS por defecto.
  - Añadir IIS\_IUSRS en seguridad, permisos (haciendo click derecho sobre la carpeta).
- Puedo ver la página, tengo el servidor levantado y me dice error en el servidor al logearme.
  - Comprueba que accedes a la URL que configuraste. Por ejemplo: Si la aplicación está configurada para trabajar en qdsvídeo.unex.es no puedes acceder desde localhost.
  - Comprueba el protocolo, si sólo está configurado para HTTPS no podrás acceder por HTTP.
  - Comprueba que el servidor esté encendido, conectado a internet, con la aplicación ejecutándose y los servicios web activos.
- Cuando abro una cámara en modo pop-up no ocurre nada.
  - Asegúrate de que permites que el navegador abra pop ups para la aplicación actual.
- En Internet Explorer no me deja abrir en una ventana nueva.
  - Primero tienes que abrir uno en panel y aceptar que se use el ActiveX de axis.
- No puedo acceder a la base de datos PostgreSQL desde otro ordenador.
  - Comprueba el firewall y después añade una línea a pg\_hba para permitir al acceso.
- Añado una línea a pg\_hba pero cuando lo vuelvo a abrir se ha borrado.
  - Asegúrate de que has editado el documento en modo administrador.
- En safari no puedo ver las cámaras en ventana emergente
  - Configuración->Preferencias->Seguridad->(Uncheck)Bloquear ventanas de aparición automática
- El servidor está en una máquina diferente a la base de datos y al logear desde el servidor no entra. Uso PostgreSQL.

- ¿Tiene PostgreSQL una línea en pg\_hba.conf para permitir el acceso? En caso contrario no podrás conectarte a éste y por lo tanto comprobar que tu usuario y contraseña son correctas.
- Al configurar IIS la opción X no me aparece
  - Asegúrate de que tienes la opción instalada en "Activar/Desactivar características de Windows/IIS", hay muchas características de IIS y no todas se instalan por defecto
- No recuerdo la contraseña, intento recuperarla en la base de datos pero aparecen unos caracteres raros
  - No es posible recuperar las contraseñas, esto sería un agujero de seguridad, todas las contraseñas se almacenan en MD5 con Salt.
- Al crear un modelo en el editor gráfico, cuando intento abrirlo no me aparece nada, o falta información:
  - El editor grafico falla a veces al interpretar el nombre de los recintos, abre el modelo en modo texto y asegúrate de que no tienes espacios en éste y de que está escrito de la forma name="Nombre".
- Cuando creo un usuario en el editor gráfico no aparece.
  - A veces eclipse se queda parado y no responde correctamente al crear un usuario, crea un rol y aparecerán ambos, el rol y el usuario.
- No me aparecen los roles estándares al elegir el superrol de un usuario.
  - Esto se debe a que no has importado el modelo correctamente, haz click derecho, load resource y busca el modelo StandardRoles.xmi o créalo tú mismo.
- Se producen errores extraños cuando intento verificar el modelo y creo que todo está bien
  - Debido a la forma en la que EMF realiza referencias en el modelo, es recomendable que el nombre de los atributos identificadores de las clases no lleven espacios, comprueba que ninguno de los dispositivos, recintos o servicios tiene caracteres extraños o espacios.

## 7. Conclusiones

En esta sección se exponen las conclusiones resultantes del desarrollo del proyecto.

### 7.1 Requisitos alcanzados

A continuación se expone como se han alcanzado los requisitos especificados al inicio del proyecto.

#### 7.1.1 Integración del vídeo de las cámaras en el sistema

El vídeo de las cámaras ha sido integrado en la aplicación servidor a través del ActiveX que proporciona Axis, AxisMediaControl, embebiéndolo en el código y la interfaz del servidor a través de C#. En el servidor web, con el objetivo de crear una solución soportada por todos los navegadores, no ha sido posible utilizar el ActiveX de Axis (puesto que sólo pueden ser utilizados en Internet Explorer) y en su lugar se extrae directamente el flujo de vídeo de las cámaras y se presenta al usuario en el WebSite.

#### 7.1.2 Acceso al sistema a través de internet

Se ha desarrollado un WebSite en HTML5 que permite el acceso para la visualización y la configuración del sistema a través de internet. El servicio se expone a través de IIS y es accesible desde cualquier parte del mundo.

#### 7.1.3 Funcionalidad de respuesta de eventos configurable

Se ha desarrollado la gestión de eventos como una cadena de elementos independiente que consumen y/o transmiten eventos unas a otras. Esta cadena, de gran flexibilidad, es configurable a través del DSL, cambiando la forma con la que los eventos se procesan.

#### 7.1.4 Integración de cámaras Axis en el sistema

Gracias a el ActiveX proporcionado por Axis Communications®, ha sido posible incluir toda la funcionalidad que las cámaras Axis ofrecen. Se gestionan eventos, realizan grabaciones y capturas de imágenes y se visualiza el vídeo gracias a éste.

#### 7.1.5 Notificación de los eventos a los usuarios

A través de los servicios de envío de correo y envío de SMS a través de un servidor SMTP y del API de Esendex respectivamente se ha conseguido enviar notificaciones a los usuarios cuando estos no se encuentran delante de la pantalla, incrementando así la velocidad de respuesta ante una situación en el sistema.

### 7.1.6 Garantizar la seguridad del sistema

Dada la sensibilidad de la información que trata este sistema, garantizar la seguridad ha sido una tarea difícil, pero se dispone de las siguientes características que maximizan esta propiedad.

- Si la base de datos y la aplicación servidor están situadas en diferentes ubicaciones la transmisión de datos entre el servidor y la base de datos se puede marcar como segura.
- Los servicios web se exponen mediante HTTPS haciendo el sistema seguro a sniffers.
- La comunicación con las cámaras puede realizarse utilizando un puerto seguro.
- El sistema no dispone de la contraseña en ningún momento, éste sólo dispone de la clave en MD5, haciendo que sea inútil escanear la memoria o romper la seguridad de la base de datos puesto que no es posible obtener la contraseña original.
- Para protegerse contra contraseñas comunes, se añade Salt a todas las contraseñas antes de codificarlas, haciendo así todas las contraseñas más seguras.

### 7.1.7 Soporte para múltiples usuarios de forma simultánea

La forma en la que está estructurada la arquitectura hace que a través del WebSite se puedan conectar tantos usuarios como sea necesario. Los WebServices expuestos por el servidor pueden dar respuesta a multitud de usuarios sin verse por ello afectado el rendimiento, al igual que los WebSockets.

### 7.1.8 Posibilidad de añadir funcionalidad externa al sistema

La forma en la que se ha desarrollado la gestión de eventos permite que a través del DSL se añadan servicios creados por el usuario. Además, gracias a tecnologías como WPF puede crearse incluso una interfaz gráfica asociada a este. Este es uno de los puntos más importantes del sistema, ya lo hace realmente versátil y configurable.

### 7.1.9 Personalización del layout del sistema

A través del modelado es posible configurar algunos aspectos de presentación de la aplicación, aun así, este requisito no ha sido totalmente satisfecho, dado que son pocos los detalles que se pueden configurar (logo, texto inicial, planos, etc...).

### 7.1.10 Integración de otro tipo de dispositivos en el sistema

Actualmente es posible manejar eventos desde dispositivos como sensores y cámaras no axis, pero el sistema no está totalmente preparado para otro tipos de dispositivo (no se puede visualizar en la interfaz el estado actual de estos dispositivos). Aun así, el sistema se ha dejado preparado para la futura ampliación con estos dispositivos con un esfuerzo de codificación mínimo. Por ello este requisito se da como parcialmente satisfecho.

### 7.1.11 Acceso al sistema optimizado para dispositivos móviles.

Este requisito no ha sido alcanzado por limitaciones de tiempo. Actualmente se puede acceder a través de la web, pero no se ofrece toda la funcionalidad que se debería, ya que los dispositivos móviles tienen limitaciones que no tienen tablets u ordenadores portátiles/sobremesa.

## 7.2 Aprendizaje Adquirido

A lo largo de los tres años en los que se ha desarrollado el proyecto, he obtenido gran cantidad de conocimientos y experiencia dentro del grupo de ingeniería de software Quercus. A continuación se detallan las más importantes.

- Arquitecturas Distribuidas.
- Integración de componentes.
- Arquitecturas Orientadas a Servicios.
- Desarrollo de sistemas dirigidos por modelos.
- C#, Java, HTML5, CSS, JavaScript y PHP.
- Servicios de notificación.
- Gestión de bases de datos PostgreSQL.
- Métodos de autenticación y seguridad.

## 7.3 Opinión sobre sistema desarrollado

Creo que se ha desarrollado un sistema amplio pero bien estructurado, flexible y sobre todo muy escalable. Se ha cuidado mucho la codificación, haciendo el código fácil de leer y entender, así como de ampliar. Multitud de tecnologías han sido evaluadas en innumerables reuniones, analizando cada beneficio e inconveniente de cada una.

Se ha creado un proyecto con un concepto nuevo y de calidad que puede ser utilizado en el futuro y ser ampliado de múltiples formas, pudiendo llegar a ser un software comercializable e importante en el mundo de vídeo vigilancia al ofrecer características que ningún otro ofrece. La sensación final del proyecto es de alegría y satisfacción con el trabajo realizado, ya que el sistema ha superado con creces las expectativas.

## 8. Ampliaciones

El proyecto en sí puede ser considerado como una base para futuros desarrollos, siendo un sistema altamente configurable y con una arquitectura bien diseñada para acoger con facilidad nuevas ampliaciones. Además, debido al tamaño del proyecto hay detalles que no han sido pulidos con toda la atención que se merecía, por lo que hay partes del proyecto que son bastante mejorables. A continuación se detallan las principales líneas que se han identificado en las que el proyecto puede avanzar:

### 8.1 Integración de las grabaciones en la web

Dado que hemos utilizado HTML5 es posible integrar vídeo con una gran facilidad en la web. Esto hace interesante ampliar la aplicación web para añadir funcionalidades de gestión de vídeo para controlar las grabaciones realizadas desde la web y poder acceder a ellas en cualquier momento.

### 8.2 Generación de un eclipse propio

A lo largo del modelado se han visto diferentes herramientas muy útiles, pero la generación del modelo y el paso de modelo a código son dos procesos totalmente separados, provocando que el usuario tenga que copiar los archivos de un proyecto a otro además de hacer que el usuario tenga que bajar eclipse y utilizar el workspace creado.

Por ello, se propone la creación de un eclipse personalizado para el proyecto, con todos los Plugins instalados y el interfaz optimizado, resultado en un entorno de desarrollo totalmente independiente. Esto es fácil de alcanzar a través del tipo de tecnología que proporciona eclipse basada en Plugin[24].

### 8.3 Desarrollo para tablets

A pesar de que la web puede ser visualizada en estos dispositivos, se recomienda realizar un desarrollo para tablets, siendo estos dispositivos realmente interesantes para este sistema. La posibilidad de administrar totalmente el sistema desde una Tablet es una utilidad que puede ser bien acogida por el sector de la vídeo vigilancia al poder ser asignado el dispositivo a un empleado de seguridad.

Dada la estructura del sistema este proyecto sería de desarrollo relativamente rápido gracias a la amplia gama de comunicación que ofrece, pudiendo acceder la Tablet al sistema utilizando los servicios web que sirven a la web y desplegando alguno nuevo si fuera necesario. Además, se cuenta ya con los WebSockets implementados para enviar notificaciones desde el servidor a las tablets.

## 8.4 Integración del modelado en el servidor

Sera interesante mostrar en la configuración del servidor el modelo que define el sistema así como quizás poder configurar el modelo desde el propio servidor. Esto se podría hacer intentando integrar eclipse en la aplicación de alguna manera.

## 8.5 Configuración del sistema desde el móvil

Actualmente se puede acceder a la web desde el móvil, pero se puede llegar más lejos. Podría ser interesante crear un servicio que lancara eventos en el sistema al recibir un mensaje desde el móvil para iniciar una grabación, consultar el estado de un dispositivo o cambiar alguna configuración. Esto es fácilmente realizable debido a que se puede utilizar el servicio de Esendex para manejar la recepción de mensajes a través de un API y al integrarse en la cadena de eventos se podría obtener el resultado deseado.

## 8.6 Mayor soporte para otros tipos de dispositivo

Tal y como esta estructurado el sistema, añadir nuevos tipos de dispositivos se ve como algo simple y de rápido desarrollo, podría considerarse también la posibilidad de añadir tipos de dispositivos como plug-in (tal y como se añaden los servicios) a la hora del modelado para incrementar notablemente la funcionalidad del sistema.

## 8.7 Mejora del modelado

Las herramientas de modelado proporcionadas pueden ser claramente mejoradas, un editor gráfico más amigable o ayuda en tiempo de modelado para la creación de los servicios sería un punto muy importante para el futuro del proyecto.

También se podría añadir al editor textual un lenguaje propio para definir los servicios personalizados/nuevos de forma que no haya que escribir código C#. Puede ser muy interesante el avance en la funcionalidad de este editor puesto que permite a usuarios expertos una configuración más rápida y con más opciones si se prepara para ello.

## 8.8 Nuevos servicios por defecto

La inclusión de más servicios por defecto puede darle mayor versatilidad al sistema. Algunas propuestas son:

- Servicio de mensajes WhatsApp.
- Control de sensores.
- Control de las señales binarias de las cámaras.

- Recepción de correo/sms.
- Conteo de personas.
- Reconocimiento facial.
- Vídeo llamada desde el sistema a un teléfono mostrando el flujo de vídeo capturado.
- Sistema de complexeventprocesing, integrando inteligencia artificial que permita reconocer patrones y responder a ellos.

## 8.9 Incremento de la personalización del aspecto de las aplicaciones generadas

El desarrollo dirigido por modelos nos ofrece la posibilidad de personalizar la aplicación de una forma bastante sencilla tal y como se hace con el WebSite. Esta personalización se puede llevar mucho más lejos, añadiendo más características configurables para hacer las aplicaciones generadas más cercanas al usuario final y mas acordes con el estilo de la empresa/usuario que las utilice.

Un buen enfoque sería intentar que el editor gráfico permitiera diseñar el layout de las aplicaciones, de forma que el usuario vea lo que quiere que se muestre.

## 9. Agradecimientos

No es posible concluir el proyecto sin nombrar en agradecimiento a todas las personas que me dieron su apoyo, y aunque la lista sea larga, es totalmente necesaria.

A mi padre, que siempre creyó en mí y me dio todo el apoyo que pudo, hacerlo sentir orgulloso hacía que cada noche de estudio tuviera sentido. Sé que aunque no pueda venir a la presentación a él era a quien mas ilusión le hacía verme terminar la carrera, siempre recordare como disfrutaba contando cuando yo no estaba que su hijo había entrado en un departamento o que se iba al extranjero a estudiar. Muchísimas gracias papá, no sé cuanto he aprendido de ti, ojalá algún día tenga tu fuerza.

Como dijo Geroge Herbert, “Un buen padre vale mas que un centenar de maestros de escuela”.

A mi madre, por intentar que estudiara siempre en las mejores condiciones y por permitir todas las cosas que me ha permitido, sin su cariño esto no habría sido posible.

A mis hermanos por todo el apoyo, siempre he tenido una relación muy cercana con mi hermana Olga, y aunque la distancia nos separa, todo el mundo sabe lo orgulloso que estoy de tener una hermana como la que tengo en Madrid que hace ya unos años pasó también por esta misma facultad y fue la que me animó a hacer lo que ella había hecho. No puedo olvidarme de mi hermano que siempre está ahí al lado para darme un consejo cuando lo necesito.

Gracias a los cuatro bichos que tuvieron como hijos mis hermanas, los que me quitaron todo el protagonismo de la casa, les tengo un cariño que no se puede escribir, hacerlos sentir orgullosos ha sido también una de mis metas, ojalá algún día lo consiga.

A Melania, por la paciencia que ha tenido, quedándose a mi lado tantas noches mientras estudiaba, dándome fuerzas y ánimos. Me has endulzado muchísimo estos años, espero que nunca te canses. Sabes de sobra la fuerza que me daba tu sonrisa cuando ya no podía más.

A mis amigos por animarme y apoyarme desde que empecé la carrera, por darme tantas horas de diversión. Entre ellos se encuentran personas como mi compañero de viaje Manuel Cantonero, del que espero no separarme y que me ha ayudado tanto en este año tan difícil, nunca lo olvidaré; David Naranjo, Daniel García y “los hermanos Mateo” con los que he pasado tantas horas estudiando y jugando a los juegos de mesa que nos traía el tío de David, cada vez con una historia de una parte del mundo; así como todos los amigos hechos en la carrera y como en mi pueblo.

Y por último y no por ello menos importante a todos los profesores que me he encontrado en la universidad, muchos han hecho de la docencia algo apasionante. En especial quiero acordarme de

Juan Hernández, aun no entiendo como le dio tres oportunidades a ese niño despistado y desastroso que era yo en segundo, él junto con José María Conejero me han enseñado todo lo que aquí se muestra, han sido muchos años corrigiéndome fallos y enseñándome como plantear problemas y enfrentarme a nuevas tecnologías. Muchas gracias, me voy con muy buen sabor de boca, creo que estos cinco años han merecido realmente la pena.

## 10. Referencias

- [1] D. D. Seligmann, 'Computing Now Archive | April 2012 | Video-Based Detection Methods', Apr-2012. [Online]. Available: [http://www.computer.org/portal/web/computingnow/archive/april2012?mkt\\_tok=3RkMMJWWfF9wsRonua7KZXonjHpfX56+8pUaW2IMI/0ER3fOvrPUfGjI4ITtQhcOuuEwcWGog8wwNVCvWBeZI=](http://www.computer.org/portal/web/computingnow/archive/april2012?mkt_tok=3RkMMJWWfF9wsRonua7KZXonjHpfX56+8pUaW2IMI/0ER3fOvrPUfGjI4ITtQhcOuuEwcWGog8wwNVCvWBeZI=). [Accessed: 08-Jul-2012].
- [2] Cabinet Office, 'The cost of cyber crime | Cabinet Office'. [Online]. Available: <http://www.cabinetoffice.gov.uk/resource-library/cost-of-cyber-crime>. [Accessed: 08-Jul-2012].
- [3] F. Truyen, 'The basics of model driven architecture'. Jan-2006.
- [4] J. Liberty, *Programming C#*, 3rd ed. Sebastopol, CA, USA: O'Reilly & Associates, Inc., 2003.
- [5] A. Jones and A. Freeman, 'Windows Presentation Foundation', in *Visual C# 2010 Recipes*, Berkeley, CA: Apress, 2010, pp. 789–904.
- [6] 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.
- [7] M. Rosen, *Applied SOA: service-oriented architecture and design strategies*. Wiley, 2008.
- [8] 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].
- [9] M. MacDonald, *HTML5: The Missing Manual*. O'Reilly Media, Inc., 2011.
- [10] D. S. McFarland, *JavaScript & JQuery: The Missing Manual*. 2011.
- [11] Wikipedia contributors, 'Ajax (programming)', *Wikipedia, the free encyclopedia*. Wikimedia Foundation, Inc., 10-Jun-2012.
- [12] I. Fette and A. Melnikov, 'The WebSocket Protocol', Dec-2011. [Online]. Available: <http://grenache.tools.ietf.org/html/rfc6455>. [Accessed: 10-Jun-2012].
- [13] D. Schiemann, 'The Long-Polling Technique', Nov-2007. [Online]. Available: <http://cometdaily.com/2007/11/15/the-long-polling-technique/>. [Accessed: 10-Jun-2012].

- [14] 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].
- [15] D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro, *Emf: Eclipse Modeling Framework*. Pearson Education, 2008.
- [16] J. Cabot and M. Gogolla, 'Object Constraint Language (OCL): A Definitive Guide', 2012.
- [17] C. Vicente, D. Alonso, and J. F. Ingles, *Introducción Práctica al Desarrollo de Software Dirigido por Modelos*. 2011.
- [18] E. Thompson, 'MD5 collisions and the impact on computer forensics', *Digital Investigation*, vol. 2, no. 1, pp. 36–40, Feb. 2005.
- [19] B. Schneier and C. Ellison, 'Ten Risks of Pki', in *Public Key Infrastructure*, Auerbach Publications, 2004.
- [20] M. Burnett, '10,000 Top Passwords « Xato', 20-Jun-2011.[Online]. Available: <http://xato.net/passwords/more-top-worst-passwords/>. [Accessed: 28-Jun-2012].
- [21] Wikipedia contributors, 'Salt (cryptography)', *Wikipedia, the free encyclopedia*. Wikimedia Foundation, Inc., 27-Jun-2012.
- [22] 'Basic AuthStandar'. [Online]. Available: <http://www.ietf.org/rfc/rfc2617.txt>. [Accessed: 28-May-2012].
- [23] Wikipedia contributors, 'HTML5 Audio', *Wikipedia, the free encyclopedia*. Wikimedia Foundation, Inc., 06-Jun-2012.
- [24] J. F. Inglés Romero, 'HuRoME: entorno de modelado para el software de un robot humanoide', 27-Sep-2010. [Online]. Available: <http://repositorio.bib.upct.es:8080/jspui/handle/10317/1802>. [Accessed: 29-Jun-2012].

## Anexo I: Utilización Pusher

En este anexo se explica como utilizar la herramienta de notificación servidor-cliente Pusher.

### *Crear una cuenta en Pusher*

Para crear una cuenta en Pusher basta con acceder a <http://pusher.com> hacer click en

**Sign Up**, introducir los datos requeridos y pinchar en “Create your free API account”. A continuación 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 identificado. 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 ocurren eventos en Pusher (conexiones, desconexiones, etc...).
- DebugConsole: Consola de depuración.
- EventCreator: Herramienta para producir eventos.
- Collaborators: Página 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 “Debugconsole” nos permite depurar el sistema de Pusher ya que registra todas las comunicaciones que se producen. Conexiones de clientes, envío de mensajes, etc...

The screenshot shows the Pusher Debug console interface. At the top, there are tabs: Stats, API access, Settings, WebHooks, Debug console (which is highlighted in blue), Event creator, and Collaborators. Below the tabs, it says "QES" and "Debug Console". There are two buttons: "Clear" and "Pause". A table follows, with columns: TYPE, SOCKET, DETAILS, and TIME. The table contains the following data:

TYPE	SOCKET	DETAILS	TIME
Connection	7664.287402	Origin: https://qdsvideo.unex.es	19:25:44
Subscribed	7664.287402	Channel: QESTest	19:25:44
Occupied		Channel: QESTest	19:25:44
▼ API Message		Channel: QESTest, Event: alertaCamara	19:28:01
{ "camara": "Quercus 0", "mensaje": "MDI" }			

At the bottom, there is a footer with the text: © 2012 Pusher | [Support](#) | [Logout](#).

### *Enviar eventos desde Pusher*

Al pinchar en “Eventcreator” 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.

The screenshot shows the Pusher Event creator interface. At the top, there are tabs: Stats, API access, Settings, WebHooks, Debug console, Event creator (which is highlighted in blue), and Collaborators. Below the tabs, it says "QES" and "Event creator".

The main area has three input fields:

- Channel name: QESTest (example: "project\_42", "room\_8").
- Event name: alertaCamara (example: "user\_created", "new\_message").
- Event data: A code editor containing the following JSON:

```
{
  "camara": "Quercus 0",
  "mensaje": "MDI"
}
```

Below the input fields, there is a message: "No events sent yet" and a "Send event" button. At the bottom, there is a footer with the text: © 2012 Pusher | [Support](#) | [Logout](#).

### *Código de Pusher*

A continuación se expone el código necesario para producir un evento

Código para .Net:

```
var request = newObjectPusherRequest("test_channel",
"my_event",
new { hello = "world" });

var provider = newPusherProvider(appId, appKey, appSecret);

provider.Trigger(request);
```

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

```
varpusher = new Pusher(pusherKey);
varchannel = 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 nueva 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 el API de Pusher (tanto del lado servidor como desde la aplicación web) en la documentación disponible en la web.

## Anexo II: Despliegue de servicios web tipo REST sobre HTTPS en una aplicación no web.

Este ha sido un punto complicado en el desarrollo del sistema y por ello se realiza este pequeña guía para poder recrear el proceso.

Éste se divide en dos partes, la exposición de los servicios en el servidor y la consumición en el cliente.

### Servidor

A continuación se detalla el proceso para exponer servicios web tipo REST en una aplicación no web en C#, debido a que esta era la aplicación servidor del sistema.

En primer lugar, debe crearse una interfaz que defina el contrato para exponer estos servicios y marcarla como “[`ServiceContract`]”, en esta clase debemos definir todos los métodos que se van a publicar, marcándolos como “[`OperationContract`]” e indicando como debe accederse a ellos. Por ejemplo, para un servicio web que se acceda con GET y tenga un parámetro token utilizaremos:

```
[WebInvoke(ResponseFormat = WebMessageFormat.Json, UriTemplate =
"GetDispositivos?token={token}", Method = "GET")]
```

Otro ejemplo utilizando POST sería:

```
[WebInvoke(UriTemplate = "Logout?token={token}", Method = "POST")]
```

Es importante marcar qué tipo devuelven, siendo por defecto XML. En nuestro caso, dado que utilizamos JavaScript en el cliente, nos conviene más el uso de JSON por su simplicidad y facilidad de uso. Esto se hace a través del parámetro ResponseFormat. Para indicar los parámetros que debe tener la URL con la que se accede al servicio utilizaremos UriTemplate, proporcionando una cadena indicando entre corchetes el parámetro que deberá tener el mismo nombre en la firma del método.

Es importante tener en cuenta que cuando se devuelven tipos no primitivos, es necesario indicar cómo deben traducirse al formato que deseemos devolver. Para ello, lo más sencillo es definir el tipo que se devuelve con propiedades, tal y como se muestra en el ejemplo a continuación para el tipo de objeto que se devuelve al identificarse en el sistema. Así, cuando se requiera el paso a JSON, se utilizará como nombre de atributo el nombre de la propiedad, y como contenido el resultado de llamar al método get.

```

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

```

(Al utilizar propiedades automáticas se devuelve y se establece directamente el valor)

Un ejemplo de interfaz sería:

```

[ServiceContract]
public interface IQESWebService
{
    [OperationContract]
    [WebInvoke(ResponseFormat=WebMessageFormat.Json, UriTemplate =
    "Login?user={username}&password={password}", Method = "GET")]
    LoginReturn Login(string username, string password);

    [OperationContract]
    [WebInvoke(UriTemplate = "Logout?token={token}", Method = "POST")]
    void LogOut(string token);
}

```

Una vez definida la interfaz del contrato correctamente, debemos declarar una clase que la implemente, definiendo en ésta todos los métodos que describimos en la interfaz. Una vez hecho esto, tendremos el servicio web listo para desplegar.

Debido a que el tipo de aplicación no es web, es necesario utilizar una clase que realice el despliegue del servicio, esta clase se encuentra en WCF, una parte de .Net framework para el desarrollo rápido de arquitecturas orientadas a servicios. Esta clase es WebServiceHost, y se utiliza de la siguiente forma:

```

WebServiceHost serviceHost = new WebServiceHost(typeof(QESWebService), new Uri(uri));
serviceHost.Open();

```

Con esto, se creará un endpoint en nuestra aplicación que expondrá los métodos que se definan en la clase QESWebService. El parámetro URI es la dirección en la que se encontrará este endpoint, ej.: "<https://localhost/WS/>"; En caso de que indiquemos HTTPS, tenemos dos opciones, hacerlo desde código, atando al usuario a un certificado específico o realizarlo mediante IIS, ya que WebServiceHost expondrá los servicios a través de IIS, utilizando la configuración que éste tenga incluyendo SSL.

## Cliente

El consumo de los servicios web desde JavaScript es muy sencillo, aun más si lo que devuelven es JSON. Utilizando JQuery (una librería de JavaScript) podemos utilizar el método getJSON que hará todo el trabajo por nosotros.

```
jQuery.getJSON( url [, data] [, success(data, textStatus, jqXHR)] )
```

Este método, tiene un parámetro con la URL en la que se encuentra el servicio, un parámetro data con un JSON que contenga todos los parámetros que deben pasarse y una función callback que se ejecutara cuando se reciba una respuesta exitosa. Es importante tener en cuenta que el código seguirá ejecutándose, es decir, no se espera a tener una respuesta para ejecutar la siguiente línea de código.

Un ejemplo de consumo de un servicio web sería:

```
$.getJSON(server+'GetDetallesCamara',{token:readCookie('token'),camara:$('#this').html()});function(data){  
    if(data.Code == -1)  
        ...gestion del callack  
};});
```

Como podemos ver, resulta muy sencillo, y para acceder a los datos recibidos, al ser JSON, utilizamos el “.”.

JQuery dispone también de las siguientes funciones para el acceso a servicios web:

- \$.get():<http://api.jquery.com/jQuery.get/>
- \$.post():<http://api.jquery.com/jQuery.post/>
- \$.ajax():<http://api.jquery.com/jQuery.ajax/>

## Anexo III: Despliegue de WebSockets

Debido al estado actual de estandarización de los WebSockets por la W3C, la implementación de esta comunicación no ha sido sencilla. Como ejemplo de explicación, se utilizará la configuración que se ha realizado para el proyecto, con un punto ejerciendo como servidor e indefinidos clientes.

### Servidor

Para la implementación del servidor se recomienda la utilización de una librería que nos proporcione métodos para utilizar directamente las primitivas de WebSockets, en lugar de tener que implementarlas nosotros mismos. En caso de que se utilice .Net framework 4.5 o superior (no disponible actualmente) se incluye soporte directo en esta versión. En caso contrario, tal y como ocurre en el proyecto desarrollado, habrá que utilizar código externo.

Tras la revisión de varias librerías (consultar estudio documentación) se recomienda el uso de Alchemy, una implementación para .Net de WebSockets muy flexible. Ésta se puede encontrar en [“http://alchemywebsockets.net/”](http://alchemywebsockets.net/).

La utilización de este API es tan sencillo como instanciar la clase “`WebSocketServer`” e indicar las acciones a realizar en los diferentes eventos que ésta define, siendo las mas importantes “`OnConnect`” y “`OnDisconnect`”, en las que debemos almacenar uno de los parámetros “`UserContext`” que corresponde a la dirección del otro extremo WebSocket. Con esta, podremos enviar mensajes a los clientes llamando al método “`Send`”. A continuación se expone código de ejemplo.

```
Usuarios = newList<UserContext>();

wsServer = newWebSocketServer(81, System.Net.IPEndPoint.Any)
{
    OnConnected = OnConnect,
    OnDisconnect = OnDisconnect
};
wsServer.Start();
public void OnConnect(UserContext context)
{
    Usuarios.Add(context);
}
public void OnDisconnect(UserContext context)
{
    Usuarios.Remove(context);
}
public void Broadcast(String mensaje)
{
    foreach (var uc in Usuarios)
        uc.Send(mensaje);
}
```

Con este código tendríamos implementado un servidor a través de WebSocket. Para detener el servidor llamaremos a “wsServer.Stop();“.

Además, este API contiene también otros métodos que pueden consultarse en la web como “OnReceive” para implementar un cliente en .Net o realizar una conexión dúplex.

Es importante tener en cuenta una de las características mas importantes de este API y es que aunque por defecto utilice el protocolo estándar RFC6455, responde también a los tres protocolos hybi (00, 10 y 17).

## Cliente

En el cliente podemos utilizar el estándar de la forma:

```
Var ws = new WebSocket(direccionWS, '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 debemos 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 sólo lectura, representa el estado actual de la conexión y puede tener los siguientes valores:  0. CONNECTING: El valor 0 indica que la conexión aún no ha sido establecida. 1. OPEN: El valor 1 representa que la conexión ha sido correctamente establecida y que la comunicación con el otro extremo es posible. 2. CLOSING: El valor 2 se utiliza para indicar que la conexión está utilizando la primitiva de cierre de conexión. 3. CLOSED: Con el valor 3 se indica que la conexión ya ha sido cerrada o
-------------------	--

	no fue posible abrirla.
Socket.bufferedAmount	The readonly attribute <b>bufferedAmount</b> 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.

