Departamento de Informática

Ingeniería Técnica en Telecomunicaciones
especialidad en Sonido e Imagen

PROYECTO FIN DE CARRERA

Integración de contenidos Mediasite en el Portal Multimedia de la Universidad Carlos III de Madrid

Autor: Sergio Tejero López

Tutor: Dr. Javier García Guzmán

Leganés, julio de 2013

Título: **INTEGRACIÓN DE CONTENIDOS MEDIASITE EN EL PORTAL MULTIMEDIA DE LA UNIVERSIDAD CARLOS III DE MADRID**

Autor: Sergio Tejero López

Director: Dr. Javier García Guzmán

EL TRIBUNAL

Presidente: Carlos García Rubio

Vocal: Alberto Heredia García

Secretario: José Arturo Mora Soto

Realizado el acto de defensa del Proyecto Fin de Carrera el día 12 de Julio de 2013 en Leganés en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la calificación de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

La mejor noticia de estar escribiendo estas líneas es que por fin la memoria de mi Proyecto Fin de Carrera está acabada. En los últimos años he trabajado duro para conseguirlo, pero es cierto que ha habido muchos parones en los que algunas veces me sentía frustrado por no conseguir mis objetivos, pero gracias al apoyo de la gente que me rodea, he podido superar todos los obstáculos que se ponían en el camino.

En primer lugar, el agradecimiento a mis padres y mi familia. Cada uno a su manera siempre ha estado a mi lado, confiando plenamente en mí y realizando un esfuerzo económico para que yo recibiera la mejor educación posible. Agradecérselo sobre todo a mi madre, que ha convivido conmigo el día a día, aguantando mis cambios de humor, mis alegrías y mis fracasos, y que se merece buena parte del agradecimiento por sólo aguantarme. Sin la ayuda de todos ellos nunca hubiera cursado una titulación y me hubiera dado la oportunidad de conocer gente maravillosa en la Universidad.

Gente que ha estado conmigo en clase, en multitud de prácticas, en días interminables de estudio en la biblioteca, pero también jugando una partida de mus en la cafetería o tomando una cerveza al sol en el césped después de acabar los exámenes. Noé, Nacho, Jorge, Pablo, Sergio, Dani, Ire, Lore, JC y tantos otros que han pasado esa frontera de ser algo más que compañeros de clase y que espero seguir teniéndolos cerca ahora que mi vida universitaria se acaba.

Cualquiera que me conozca pensará que me he olvidado de una persona que durante los tres primeros años perteneció a ese grupo de gente importante, pero que un 12 de Mayo de 2007, se convirtió en la persona más importante de mi vida y lo mejor que me voy a llevar de mi periodo universitario. A ti Tamara te tengo que agradecer muchísimas cosas, tantas que no caben en este pequeño párrafo que te voy a dedicar. Primero agradecerte que me quieras y que me hayas dejado ser partícipe de tu vida, gracias a ti he podido ser mejor como persona. Gracias por tu paciencia, por no mandarme a la mierda durante estos años en los que el proyecto no salía adelante, sin tu apoyo moral y exigencia hubiera sido imposible acabar la aplicación y la memoria. Sólo espero que en este nuevo periodo de nuestra vida, que comenzará en unos meses, sigas a mi lado y seas mi compañera de viaje durante el resto de mi vida. Te quiero mucho.

Por ultimo agradecer a Javier García, David Santín y Paco Cruz, parte esencial en la realización del proyecto. A Javier por sus explicaciones y anotaciones que han conseguido guiar mi trabajo hasta conseguir el resultado final plasmado en esta memoria. A David y Paco por confiar en mí y darme la oportunidad de realizar un proyecto de estas características. Oportunidad que surgió por trabajar en el Área de Audiovisuales del Servicio de Informática, que gracias al buen ambiente que se respiraba he conocido a grandes personas como Patri, Jesús, Raul, Fernan, Paquito, la gente de la quiniela y tantos otros que han hecho que mi estancia en el departamento sea perfecta y que espero seguir teniendo relación con ellos cuando ponga punto final a mi trabajo en la Universidad.

Resumen

El uso de los sistemas de captura de clases es cada más frecuente en los centros docentes y de investigación. Los motivos que han provocado esta subida en la utilización de estos sistemas son el abaratamiento de los equipos necesarios, la alta velocidad de la red de internet o la necesidad de ofrecer al alumnado material adicional del impartido en la clase presencial.

La Universidad Carlos III de Madrid realizó una importante inversión para dotar a varias aulas con un sistema de captura de clases, concretamente con *Mediasite* del distribuidor *Sonic Foundry.*

Existe un desconocimiento por parte de la comunidad universitaria de este sistema, lo que provoca que su uso sea muy bajo. Uno de los motivos es que Mediasite es un sistema independiente y no está integrado en el Portal Multimedia de la Universidad Carlos III, ArcaMM.

Por tanto la motivación principal para realizar este proyecto es la de ayudar a la publicación de contenido generado por Mediasite en ArcaMM y así centralizar todo el material audiovisual en un mismo lugar.

El objetivo principal es la creación de una herramienta que formará parte de ArcaMM para catalogar los vídeos creados con Mediasite. Esta herramienta estará formada por un script de actualización, que extraerá todos los metadatos referentes a los vídeos, y una interfaz gráfica, que mostrará los videos que se pueden publicar y realizará la posterior publicación. Los objetivos secundarios son los de ofrecer al usuario otras funciones adicionales a través de la interfaz.

La metodología de trabajo se ha basado en un desarrollo iterativo e incremental. Se ha elegido esta opción porque se adapta bien a la planificación adaptativa que hemos seguido. Se ha elegido un entorno de desarrollo basado en software libre, y se han utilizado lenguajes para el desarrollo del código como PHP, SOAP, HTML o JAVASCRIPT y MYSQL para la gestión de la base de datos.

Como resultado final se ha obtenido una aplicación que formará parte de ArcaMM, que será accesible a través de web por medio de un nombre de usuario y una contraseña y que utilizará el personal del Área de Audiovisuales para publicar los contenidos generados con Mediasite en ArcaMM. Además el usuario podrá visualizar los vídeos programados para un futuro y realizar estadísticas de publicación. La herramienta está disponible desde Septiembre de 2012 en el área privada de ArcaMM.

Abstract

The use of lecture capture system is becoming more common in academic and research centers. The main causes for this rise are: the cheap equipment, a faster speed internet network or the need to provide different additional material to students.

The Carlos III University has made a relevant investment to equip various classrooms with lecture capture system, in particular with Mediasite (owner by the enterprise Sonic Foundry).

Nowadays, the use of these systems is very low in the university community, because there is a general ignorance of them. The main reason is that Mediasite is an independent system and is not integrated in the Multimedia Website of the Carlos III University (ArcaMM).

Therefore, the main motivation to make this project is to help the publication of Mediasite generated content in ArcaMM and so centralize all audiovisual content in one place.

The objective of this project is to provide a tool to ArcaMM in order to allow the users to publish the videos created with Mediasite. This tool is compound by an update script, which will extract the metadata from Mediasite videos, and a graphical user interface, which will show the videos that the user can publish and will provide additional features to the user.

The working methodology has been based in an iterative and incremental development, because it was well suited to the adaptative planning used in this project. The chosen development environment was free software, using languages such as PHP, HTML, SOAP o JAVASCRIPT and MYSQL for databases management.

As a result of this project, it has been developed an integrated application in ArcaMM, which will be accessible via web with personal authentication (username and password). This project provide to the personal a tool to publish Mediasite generated content in ArcaMM, see scheduled presentations and check or generate statistics. The tool is available from September 2012 in the ArcaMM restricted area.

Índice General

Lista de Figuras

Lista de Tablas

CAPÍTULO 1 INTRODUCCIÓN

En este primer capítulo se hará una breve introducción al proyecto, exponiendo en primer lugar cual es el problema que se ha detectado y que se pretende resolver, a continuación cuales son las motivaciones para resolver dicho problema, cuáles son sus objetivos, cual ha sido el método de resolución llevado a cabo, un listado de una serie de términos que nos ayudaran a seguir la lectura de esta memoria y por último, el contenido de dicha memoria.

1.1 PROBLEMA

El problema que se ha detectado y que se pretende resolver mediante este proyecto es que existe un recurso docente para la comunidad universitaria llamado *Mediasite* que no está integrado dentro del repositorio de contenido multimedia de la Universidad Carlos III de Madrid, la plataforma ArcaMM, donde se aglutina todo el contenido audiovisual de la universidad.

Para comprender mejor el problema, debemos conocer los principales agentes implicados: *Mediasite* y ArcaMM.

- ***Mediasite*** es una solución englobada en las herramientas conocidas como *Lecture Capture* que tiene como función principal la grabación de contenidos audiovisuales de forma automatizada en aulas o recintos habilitados para ello. Además realiza funciones adicionales de gestión de los contenidos generados y webcasting de estos ya sea en directo o bajo demanda. Su implantación en la Universidad Carlos III de Madrid se llevó a cabo de forma completa en el 2009 y se puede hacer uso de ella en al menos seis aulas repartidas por los tres campus de la universidad.

- **ArcaMM** es el nombre que recibe el portal de vídeos de la Universidad Carlos III. En él podemos encontrar todos los contenidos audiovisuales que han sido generados por el personal docente o administrativo para el uso de la comunidad universitaria. Estos contenidos pueden ser públicos o privados, dependiendo del uso que le quiera dar el autor del contenido y pueden encontrarse en distintos formatos (wmv, flv, mp4, mp3, pdf).

Este problema afecta tanto a personal docente y estudiantes, como al personal del Área de Audiovisuales del Servicio de Informática, ya que estos últimos son los encargados de catalogar e indexar todo el material audiovisual que se realiza en la universidad. Muchos profesores rehúsan utilizar *Mediasite* por el mero hecho de que a los vídeos de sus clases no se podrá acceder por la plataforma ArcaMM, y deciden utilizar las distintas aulas multimedia de los diferentes campus en las que un técnico de audiovisuales les realiza la grabación, edición (si fuera necesario) y posterior catalogación.

1.2 MOTIVACION DEL PROYECTO

El problema descrito en el apartado anterior y el afán de solucionar los problemas de los miembros de la comunidad universitaria, es lo que nos motiva a llevar a cabo una solución que permita que los vídeos grabados con *Mediasite*, sean accesibles desde la plataforma multimedia de la universidad, así facilitando su acceso y su difusión.

Otra motivación por la que se quiere dar solución a este problema, es la creciente demanda de crear contenido multimedia tanto de clases, congresos, tutoriales y demás actividades que se llevan a cabo en la universidad. No siempre existe la posibilidad de utilizar una sala multimedia donde poder hacer la grabación de una clase, ya sea porque están ocupadas por otro tipo de actos o porque no existe personal para llevar a cabo la grabación, con lo que a veces se declina el hacer uso de estos recursos. Por ello se acondicionaron aulas en los distintos campus en las que se podían hacer grabaciones con la herramienta *Mediasite,* donde simplemente colocándonos un micrófono de solapa y accionando un botón para empezar o terminar la grabación, nuestra clase quedaría grabada sin necesidad de tener ningún conocimiento audiovisual. A pesar de la existencia hace unos años de estas aulas, su aceptación no ha sido del todo la esperada, en parte por el problema que nos ha llevado a realizar este proyecto.

Por tanto, vamos a crear primero una herramienta que facilitará al personal del Área de Audiovisuales la catalogación de todos los vídeos creados con la herramienta *Mediasite* y segundo un fácil acceso a los usuarios de dichos vídeos a través de la plataforma ArcaMM, donde se pondrán integrar con otros tipos de materiales realizados con otro tipo de herramientas.

Por último, otra motivación por la que he decidido llevar a cabo este proyecto es el haber trabajado durante cuatro años en el Área de Audiovisuales, viendo en primera persona el problema. Al trabajar ahí, he podido disponer del material adecuado para llevar a cabo este proyecto, así como trabajar con la gente encargada del repositorio de contenido multimedia, tener acceso a una herramienta como *Mediasite* la cual se necesita una licencia para poder acceder a ella así como estar familiarizado con la plataforma ArcaMM y con todo lo referente a la catalogación de vídeos.

1.3 OBJETIVOS

El principal objetivo de este proyecto fin de carrera es el de **desarrollar una herramienta para ArcaMM, para catalogar los vídeos creados con *Mediasite,* integrando las dos plataformas, facilitando el acceso a dichos vídeos por parte de la comunidad universitaria**.

Figura 1.1: Esquema del objetivo a realizar

Además de este objetivo principal, este proyecto tiene otros objetivos que se enumeran a continuación:

- **Creación de una base de datos propia.** Lo primero que debemos hacer es crear una base de datos propia donde estén todos los enlaces a los vídeos con sus respectivas características y así poder realizar un nexo de unión entre *Mediasite* y ArcaMM. Para ello crearemos un *daemon* para automatizar las operaciones de añadir, modificar o eliminar vídeos.

- **Creación de la interfaz de muestra de contenidos.** El siguiente paso es crear una interfaz web que nos permita listar los vídeos tanto publicados como no publicados, así como realizar determinadas búsquedas dependiendo de unos criterios. Además de indicarnos las futuras presentaciones, generar estadísticas de publicación o configurar el acceso a la herramienta *Mediasite* por si esta cambiara de versión o lugar.

- **Publicar vídeos en el Portal ArcaMM.** El último paso a realizar es la publicación del vídeo en ArcaMM, donde seleccionando el vídeo deseado nos llevará a la pantalla *manager,* común a todos los vídeos que van a ser catalogados, y en donde por defecto se rellenarán los campos que el propio vídeo lleva intrínsecos, como el nombre, el autor, los datos del códec de vídeo, etc. No obstante, estos campos se pueden modificar si se desea, además de añadir otros campos tanto obligatorios como no obligatorios

1.4 MÉTODO DE RESOLUCIÓN

Una vez que tenemos identificado el problema y los objetivos que queremos llegar a conseguir debemos idear una estrategia para resolver el problema. El método que hemos llevado a cabo para resolverlo ha sido iterativo, ya que a pesar de que teníamos claros muchos requisitos que debía cumplir nuestra aplicación, había otros que debíamos definirlos a lo largo del proyecto, dependiendo de los avances que se pudieran dar tanto en la versión de *Mediasite* o en la plataforma ArcaMM.

Basándonos en que el método de desarrollo llevado a cabo es iterativo e incremental, decidimos llevar a cabo un desarrollo ágil de software en el que las reuniones del grupo de trabajo tienen mucha importancia y se enfatiza en las comunicaciones cara a cara en detrimento de la documentación. La frecuencia de estas reuniones se estableció al inicio del proyecto y se marcaron en dos semanas en la parte inicial y final, y en cuatro semanas en la parte de desarrollo.

Estas reuniones marcaban el principio de la iteración donde primero se evaluaba si se habían cumplido los requisitos de la iteración anterior (si existiera). Después se marcaban los nuevos objetivos de la siguiente iteración. Finalmente se proporcionaba al desarrollador la lista de tareas a realizar en el tiempo estipulado, dependiendo de la fase del proyecto en la que nos encontráramos.

Además, siempre se ha estado en contacto con otros grupos de trabajo, como el de la plataforma ArcaMM, que informaban cada 4 semanas de los cambios o mejoras que se hacían en la plataforma, con lo que a veces ha sido necesario modificar alguna iteración antes de que finalizara.

1.5 TERMINOLOGÍA

A continuación se muestra un glosario con terminología usada en la presente memoria que ayudará a comprender al lector la totalidad del escrito.

1.5.1 GLOSARIO DE TERMINOS

ArcaMM es la plataforma de contenido multimedia de la Universidad Carlos III de Madrid.

Cron es el programa que permite a los usuarios de Linux ejecutar automáticamente comandos o scripts (grupos de comandos) a una hora o fecha específica.

Daemon es un proceso en segundo plano y que a menudo se inicia como proceso.

JavaScript es un lenguaje de programación interpretado que se utiliza principalmente en el lado del cliente. Se utiliza principalmente para mejorar la interfaz y realizar páginas web dinámicas.

jQuery es una librería JavaScript que contiene las funcionalidades comunes de DOM, eventos, efectos y AJAX.

Mediasite es una herramienta corporativa que se utiliza para grabaciones de clases u otros eventos, en el que el vídeo y la presentación de diapositivas quedan integrados en una única interfaz facilitando la navegación a través del vídeo.

MySQL es un sistema de gestión de base de datos basado en código libre.

Recorder es cada uno de los dispositivos que se encargan de realizar las grabaciones en *Mediasite* y se pueden encontrar en las aulas habilitadas para tal efecto.

Streaming es la distribución de cualquier contenido multimedia a través de la red en la que el usuario puede disfrutar del contenido en su ordenador o smartphone mientras este se descarga.

Web Service es un conjunto de métodos o protocolos que nos permiten intercambiar información entre distintas aplicaciones

1.5.2 ABREVIATURAS

EDAS Del inglés *External Data Access Service*, permite realizar acciones sobre la base de datos de *Mediasite* desde un dispositivo externo.

FTP Del inglés *File Transfer Protocol*, es un protocolo de transferencia de archivos a través de la red.

HTTP Del inglés *Hypertext Transfer Protocol*, es el protocolo usado en cada transacción de la *World Wide Web*.

HTTPS Del inglés Hypertext Transfer Protocol Secure, es la versión segura del protocolo HTTP.

JSON Del inglés *JavaScript Object Notation*, es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

LDAP Del inglés *Lighweight Directory Access Protocol*, es un protocolo que permite acceder a un servicio de directorio ordenado y distribuido para buscar información en un entorno de red.

OFC Del inglés *Open Flash Chart*, es una herramienta para realizar graficas que es gratuita y de libre uso.

PHP Del inglés *Hypertext Pre-processor,* es un lenguaje de programación diseñado para crear páginas web dinámicas para la interpretación del lado del servidor.

SOAP Del inglés *Simple Object Access Protocol*, es un protocolo que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

UC3M Universidad Carlos III de Madrid.

URL Del inglés *Uniform Resource Locator*, es la cadena de caracteres con la cual se asigna una dirección única a cada uno de los recursos de información disponibles en la Internet.

WSDL Del inglés *Web Services Description Language* es un formato XML para describir un determinado Web Service que describe la interfaz pública a los servicios Web.

XML del inglés *eXtensible Markup Language* en un leguaje de marcas que permite definir la gramática de lenguajes específicos para estructurar documentos grandes. Permite intercambiar información de una manera sencilla y segura.

1.6 CONTENIDO DE L A MEMORÍA

En este último apartado introductorio explicamos brevemente el contenido de cada una de las secciones contenidas en esta memoria.

En el capítulo 1, *Introducción*, exponemos el problema que hemos detectado para motivarnos a la realización del proyecto, los objetivos que se desean cumplir, el método de resolución que se ha llevado a cabo, algunas aclaraciones sobre la terminología empleada en la memoria y el contenido de esta.

En el capítulo 2, *Estado del arte*, se analizan los principales sistemas de captura de clases docentes que se ofrecen en el mercado en la actualidad y la integración del sistema *Mediasite* en los centros docentes donde se utiliza dicha herramienta. Además se expone información del entorno de desarrollo elegido y las conclusiones a las que nos han llevado el análisis.

En el capítulo 3, *Especificación de Requisitos*, se exponen los requisitos que debe cumplir la aplicación web a realizar.

En el capítulo 4, *Diseño de la aplicación*, se detallan los pasos que se han seguido para la realización de la aplicación. Dividiremos el diseño en dos bloques: la extracción y almacenamiento de los metadatos referentes a los vídeos contenidos en *Mediasite* y la integración de esta información con el portal de vídeos de la UC3M, ArcaMM.

En el capítulo 5, *Pruebas de la aplicación*, se exponen los resultados obtenidos de las pruebas de sistema realizadas.

En el capítulo 6, *Conclusiones y líneas futuras*, se presentan las conclusiones a las que se han llegado durante la realización del proyecto y presentamos una serie de mejoras que se pueden llevar a cabo en la aplicación realizada.

CAPÍTULO 2 ESTADO DEL ARTE

El primer paso que se debe dar ante un proyecto como el que queremos llevar a cabo, es el análisis del contexto en el que se va a desarrollar la aplicación y los distintos sistemas que podemos encontrar en el panorama del e-learning cuyo fin es el mismo o similar, que el que se nos ofrece para el desarrollo de este proyecto, *Mediasite.*

En este capítulo expondremos el trabajo de investigación realizado en la búsqueda de los sistemas de grabación de clases existentes en el mercado, realizando una comparativa con el sistema que tenemos en la Universidad Carlos III. Además examinaremos las distintas formas en la que se ha integrado el sistema *Mediasite* en otras universidades, centros docentes o de investigación, intentando buscar similitudes con los objetivos que marcamos al principio de este proyecto. Por último explicaremos el entorno de desarrollo elegido y que herramientas hemos utilizado para la realización del proyecto.

El estudio se va a realizar mediante la búsqueda a través del buscador web Google y en la página de *Sonic Foundry* [12], creador de la herramienta *Mediasite*, en la que se puede extraer que entidades hacen uso de su sistema y de qué manera lo llevan a cabo.

2.1 ANÁLISIS DE SISTEMAS DE CAPTURA DE CLASES

En esta primera sección queremos presentar los sistemas más representativos en el panorama de la captura de clases existentes en el mercado, realizando un análisis comparativo de cada uno de ellos, observando sus puntos fuertes y débiles respecto al sistema utilizado en la realización de este proyecto.

Cabe reseñar, que este análisis no es realizado para evaluar si el sistema elegido por la UC3M, es el correcto, ya que esto se escaparía de los límites de este proyecto, sino para conocer que nos ofrecen este tipo de sistemas, en especial *Mediasite,* y posiblemente deducir los motivos que llevaron a los responsables del Área de Audiovisuales a elegir este sistema y desechar otros.

Antes de comenzar el análisis quisiéramos justificar de forma breve el porqué del incremento del uso de las tecnologías de captura de clases, conocidas como *Lecture Capture*.

El uso de estos sistemas en centros educativos, a pesar de que a primera vista podamos pensar que son usados desde hace poco tiempo, comenzaron a utilizarse en la década de los 70 en universidades como la de Ginebra o Carolina del Norte [16]. La tecnología usada en esos tiempos se basaba simplemente en la grabación mediante una cámara de vídeo y un micrófono que quedaba registrada en una cinta magnética, como podía ser un VHS o un BetaCam. En principio estas grabaciones quedaban disponibles para que la comunidad universitaria pudiera tener acceso a ellas siempre que quisiera, con ciertas limitaciones, como el número de copias existentes o la disponibilidad del contenido. Esto que a fecha de hoy nos puede parecer un método rudimentario, sentó las bases del e-learning y de los sistemas de captura de clases existentes en la actualidad.

Este sistema, que no pretendía remplazar las clases presenciales, ofrecía a la comunidad universitaria una alternativa para poder reforzar los contenidos adquiridos durante las clases o conferencias, permitiéndole visualizarlos las veces que se deseara. Este tipo de sistemas comenzó a tener buena aceptación dentro de los centros educativos dedicados a las ciencias de la salud, ya que era utilizado para grabar experimentos, cirugías o temarios muy abruptos debido al vocabulario técnico que contenían. A pesar de ello, no estaban respaldados por la mayoría, por las dificultades que planteaban su difusión, el alto coste de los equipos o la necesidad de tener personal cualificado para utilizar dichos equipos.

La globalización de internet acompañado de los avances tecnológicos en equipamiento audiovisual, tanto de captación como de almacenamiento, permitieron que la difusión de todos estos contenidos fuera mucho mejor y permitió que la aceptación de estas tecnologías fuera mucho mayor. Además empiezan a hacer su aparición sistemas que permiten integrar captura de vídeo, VGA y audio en un mismo contenido con ayuda de distintos dispositivos como *Picture in Picture*, codificadores, capturadoras a través de PC, etc.

Todos estos avances sumados a la alta velocidad de conexión a la red que puede tener acceso un usuario medio, nos llevan al panorama actual, donde existe un mercado específico en la captura de clases. Los distribuidores ofrecen un software que integra todas las señales en un mismo contenido con ayuda de un ordenador equipado con distintas capturadoras y para el que no son necesarios amplios conocimientos en la materia, ya que muchos de ellos se accionan mediante el pulsado de un botón. Además de la simplicidad del número de equipos que deben ser desplegados, ofrecen un servidor propio en el que el administrador puede asignar permisos para visualizar los contenidos, crear perfiles de grabación y gestionar de forma sencilla toda la infraestructura existente. En las posteriores secciones analizaremos algunos de estos sistemas en el que se encuentra la herramienta utilizada en este proyecto, *Mediasite*.

2.1.1 ACCORDENT

Uno de los sistemas existentes en el mercado es *Accordent*, adquirido hace unos años por la empresa líder en el mercado de las videoconferencias como es *Polycom*. La adquisición por parte de esta empresa es totalmente lógica, ya que el sistema está enfocado sobre todo a la transmisión en directo de conferencias.

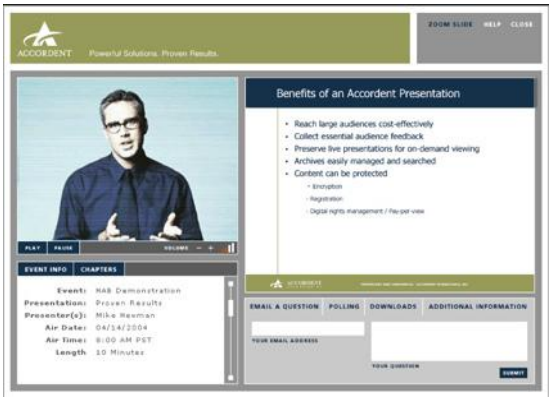


Figura 2.1: Accordent

El sistema que presenta esta empresa se trata de un dispositivo de sala totalmente automatizado al que se le pueden conectar señales de vídeo, audio y VGA, con los que crea un contenido en línea de forma inmediata para poder ser seguido en directo desde cualquier lugar [9]. El modelo por el cual el sistema es innovador, es el portátil, ya que puede ser trasladado a cualquier lugar donde exista una conexión a internet, una cámara y un micrófono, útil para grabaciones desde exteriores o desde varias salas distintas.

Figura 2.2: Accordent portátil

El problema de esta tecnología, es que no ofrece un servidor centralizado con un sistema de catalogación propio, con lo que los beneficiarios de este sistema necesitan amplios conocimientos para poder ofrecer los contenidos bajo demanda.

2.1.2 ECHO360

El sistema de captura de clases *Echo360,* junto al que tenemos en la UC3M, son los que ocupan la mayor parte del mercado en este tipo de tecnologías [17]. Una de las diferencias que ofrece respecto a otros sistemas, es el amplio abanico de posibilidades que ofrece a los usuarios, ya que dependiendo de las necesidades de este, como del presupuesto disponible, pueden adquirirse distintas alternativas que se ajusten a las preferencias del cliente.

En su versión más simple ofrece una herramienta basada en escritorio, la cual es recomendable hacer uso de ella a través de un portátil, ya que integra la captura de pantalla con la imagen tomada por la webcam. En esta versión, al usuario se le proporciona un servidor en el que se auto gestionaran sus contenidos, pero no ofrece ni programas de edición y la subida al servidor debe de hacerse de forma manual. La ventaja de esta versión es su bajo coste respecto a otras soluciones.

En su versión más potente, ofrece una amplia gama de características que otras versiones no poseen, tales como la captura automática, edición de contenidos directamente en el servidor, controles de sala, un diverso catálogo de manuales para la programación de captura de clases, interacción de usuarios durante las clases o la integración en sistemas CMS/LMS incluyendo Moodle, iTunes o Blackboard. Además pone a disposición del usuario API`s para que su flujo de información pueda ser integrado en aplicaciones propias, como es el caso de este proyecto [10].

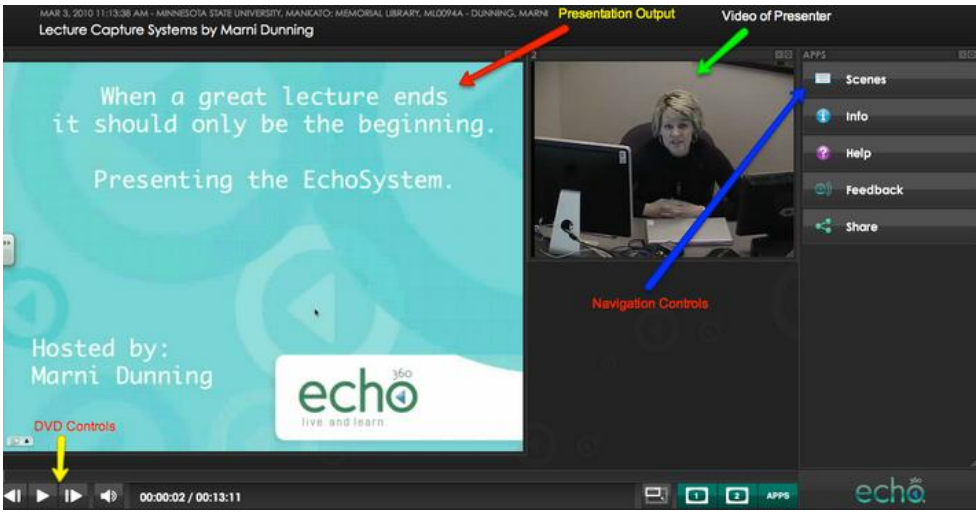


Figura 2.3: Echo360

2.1.3 OPENCAST MATTERHORN

El proyecto *Matterhorn* está desarrollado por instituciones educacionales de ámbito internacional que tienen como objetivo desarrollar un sistema automatizado de grabación, procesado y distribución de vídeos de carácter lectivo.

La principal cualidad que ofrece *Opencast Matterhorn* es que no hay que pagar por el propio sistema como pasa con el resto de herramientas que existen en el mercado, sino que únicamente hay que pagar por el equipamiento utilizado, lo cual reduce sustancialmente los costes, ya que parte del coste total pertenece al software ofrecido por los distribuidores.

Otras ventajas que ofrece la plataforma es la codificación de contenidos en varios formatos, lo que permite una mayor compatibilidad. Además añade un sistema de feeds que permite la integración con todo tipo de páginas webs. Este factor fue determinante para que el Área de Audiovisuales investigara acerca de este proyecto y a través del trabajo realizado en un PFC [7], integrara el sistema *Matterhorn* en la UC3M para su posterior difusión a través de ArcaMM.

Figura 2.4: Opencast Matterhorn

2.1.4 PANOPTO

Otro sistema que podemos encontrar en el mercado es Panopto, integrado en más de 400 universidades [11] y que ofrece un servicio similar a sus competidores pero más asequible económicamente.

Dentro de su solución enfocada a la educación, el sistema oferta contenidos de alta calidad de vídeo tanto en directo como bajo demanda, además de la posibilidad de interactuar por parte de los usuarios de los contenidos cuando estos se estén realizando.

En referencia a la interfaz que ofrecen al usuario es menos vistosa que otros sistemas, pero totalmente funcional, ya que integra el vídeo, la captura de ordenador y la ventana donde los alumnos pueden interactuar con el profesor. Además como la mayoría de estos sistemas permite la navegación a través del vídeo dependiendo de la transparencia en la que se quiera trabajar, lo que facilita mucho la búsqueda de un contenido concreto.

Figura 2.5: Panopto

Entre otros aspectos que ofrece *Panopto,* permite al usuario o administrador realizar una pequeña edición con una herramienta de edición de vídeo bastante completa. Además en sus últimas versiones están ofreciendo servicios a través de teléfonos móviles, lo que fomenta aún más la difusión de contenidos.

Por último permite una integración completa con moodle, Angel o Balckboard, lo que permite que podamos acceder al gestor de vídeos que nos ofrece la plataforma a través de cualquiera de estos sistemas. Además proporciona una API para desarrolladores para que puedan realizar aplicaciones adicionales.

2.1.5 MEDIASITE

Este último sistema que vamos a analizar es con el que vamos a trabajar a lo largo de este proyecto, ya que fue la apuesta por parte de la Universidad Carlos III para realizar grabaciones automáticas de clases.

Después de analizar otros sistemas, tanto *Panopto* como *Echo360*, ofrecen una solución muy similar y sin grandes diferencias importantes. A pesar de ello Mediaste lidera las ventas dentro del mercado de captura de clases, ya que copa el 17% del despliegue existente (2012) [8] [17].

Dentro de las características que ofrece podríamos destacar la compatibilidad de entradas tanto VGA y DVI, lo que permite la captura de pantalla en alta calidad, que unido a una interfaz que utiliza *Silverlight*, dan un resultado vistoso al contenido resultante.

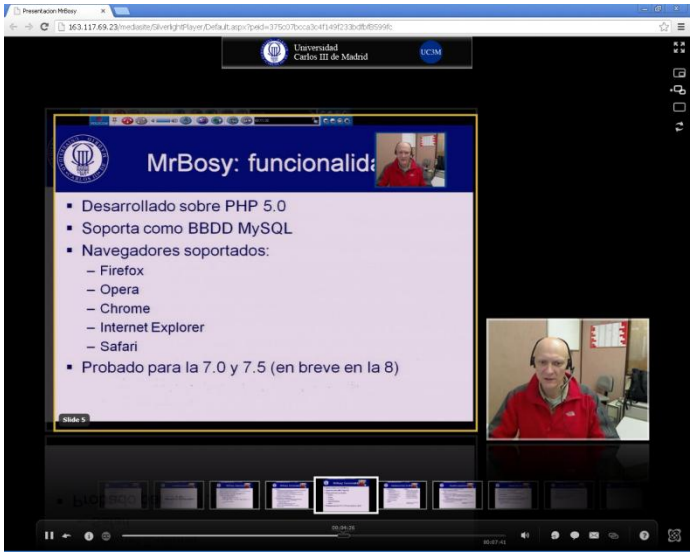


Figura 2.6: Mediasite

Finalmente podríamos destacar el planificador que posee el sistema que permite programar las grabaciones tanto para que se realicen de forma automática o mediante el pulsado del botón existente en el aula. Otros aspectos importantes a destacar es la posibilidad de añadir subtítulos a los contenidos, la integridad con plataformas educativas como Moodle o dar la posibilidad a desarrolladores mediante el EDAS de integrar la plataforma en otros sistemas, que es el objetivo principal de este proyecto.

2.2 ANÁLISIS DE INTEGRACIÓN DE MEDIASITE EN SISTEMAS DE CATALOGACIÓN

El siguiente paso que debemos realizar en nuestro trabajo de investigación, es encontrar aplicaciones similares al objetivo que queremos conseguir. Para ello nos ayudamos de la información que nos proporciona la empresa creadora de *Mediasite, Sonic Foundry*, que a través de su web nos muestran distintas aplicaciones realizadas para facilitar el uso de la herramienta [12].

2.2.1 SOLUCIONES SIN SISTEMA DE CATALOGACIÓN PROPIO

En nuestra búsqueda encontramos distintas formas de integrar el sistema Mediasite dentro del ámbito universitario, sobretodo el acceso que tiene el usuario al contenido.

Lo primero que debemos decir es que no hemos encontrado nada igual a lo que queríamos realizar en este proyecto, ya que después de buscar en 20 universidades que utilizan el sistema *Mediasite*, ninguna tiene un portal de vídeos similar al que se posee en la Universidad Carlos III. Cuando decimos esto, no queremos decir que no se ofrece al alumnado o profesorado contenidos multimedia, solo que no lo hacen a través de un portal que aglutine vídeos de todos tipo de contenidos o de todo tipo de formatos de grabación tal y como realiza ArcaMM.

El uso más extendido de integración de Mediasite en páginas web educativas, es el de enlazar a través de una url al alumnado a los vídeos o catálogos creados en *Mediasite*, utilizando la interfaz existente en el servidor de la herramienta. Desconocemos si para ello han utilizado el EDAS para llevarlo a cabo, pero en unos casos si nos sugiere que si lo han hecho por la cantidad de enlaces que proporcionan al usuario.

En un extremo podríamos encontrar la solución que adopta la Universidad de Drexel en *Philadelphia* que es añadir tanto los enlaces a catálogos como a las presentaciones a través de su página web en zonas, a nuestro parecer, no muy intuitivas, ya que no se encontraban en una zona donde hubiera más contenido multimedia, sino en una zona de soporte y ayuda [13]. A pesar de ello, desde ese lugar enlazan en el caso de los *Workshops*, a los catálogos generados en la plataforma o una presentación en concreto.

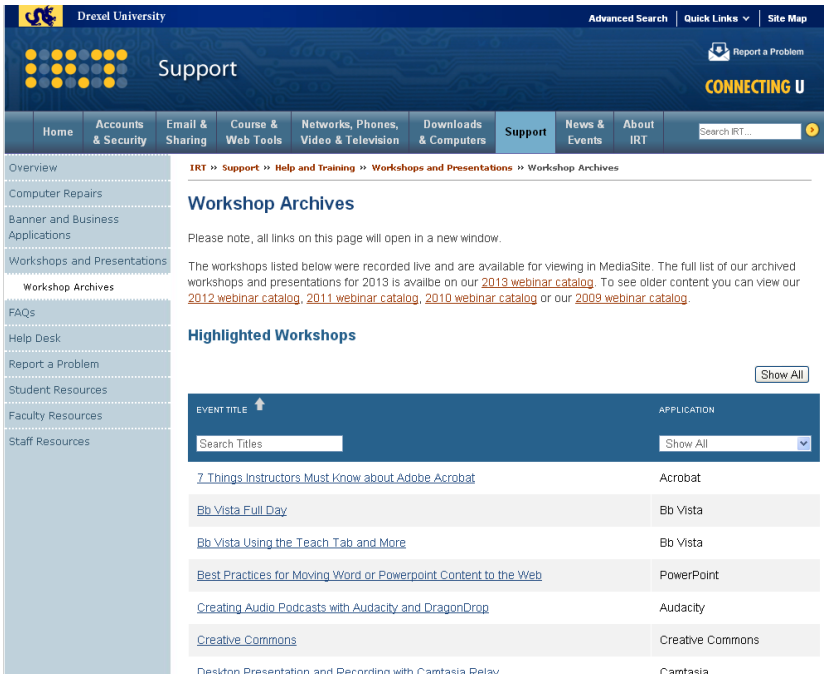


Figura 2.7: Universidad de Drexel

Una solución parecida, pero mejor implementada a nuestro parecer, es la que se realiza en la Universidad de Wisconsin *La Crosse*, ya que a través del departamento *Academic Technologic Services*, ponen a disposición de los usuarios todos los contenidos generados a través de Mediasite o de otras plataformas a través de una web dedicada para ello [14]. La solución es muy similar a la que se utiliza en la UC3M con ArcaMM, ya que centraliza todos los contenidos en un mismo portal, aunque difiere bastante en la interfaz y la manera de presentar los contenidos. En esta solución si creemos que la han llevado a cabo con el EDAS, ya que la cantidad de contenidos es bastante grande y nos resulta raro pensar que lo hagan de forma manual, aunque la extracción de datos realizada solo está basada en los links de accesos a contenidos, obviando el resto de datos referidos a las presentaciones. Una cosa que si nos podía haber sido de ayuda es que asocian una imagen a cada uno de los links, pero al comprobar si se trataba de un fotograma del vídeo, observamos que se trataba a una imagen que tenía relación con el contenido pero no estaba extraída del propio vídeo.

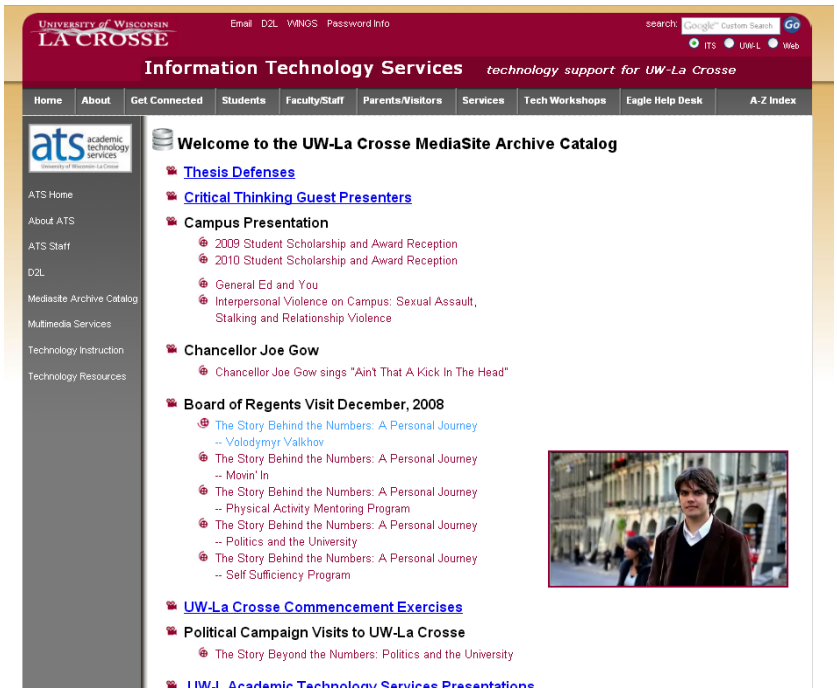


Figura 2.8: Universidad de Wisconsin La Crosse

En el otro extremo podemos encontrar la solución que aplican la mayoría de los centros educativos, que es aprovechar la interfaz que proporciona el propio sistema *Mediasite*, aunque realizan una serie de modificaciones para que cada uno de los entornos sea exclusivo. La interfaz que proporciona el proveedor permite al administrador del sistema crear un acceso web para que los usuarios puedan visualizar los contenidos grabados con *Mediasite*. El administrador puede decidir que contenidos pueden ser públicos y cuales necesitan ser accedidos mediante usuario y contraseña, donde tendrán un determinado rol y pertenecerá a un grupo de usuarios que tendrán acceso a los mismos contenidos.

Sonic Foundry es la solución que nos recomienda, ya que lo único necesario es que exista un administrador con conocimientos del sistema encargado de gestionar el contenido, usuarios y grupos de usuarios. En principio esta es la forma implantada en la UC3M, hasta la realización de este proyecto, donde se quiere integrar todo el flujo de contenido de Mediasite en una misma plataforma, aprovechando el amplio conocimiento de la comunidad universitaria de la UC3M de la plataforma ArcaMM.

Como ejemplo podemos observar el uso de esta solución en *The Viticulture Enology Science and Technology Alliance (VESTA).* [13]

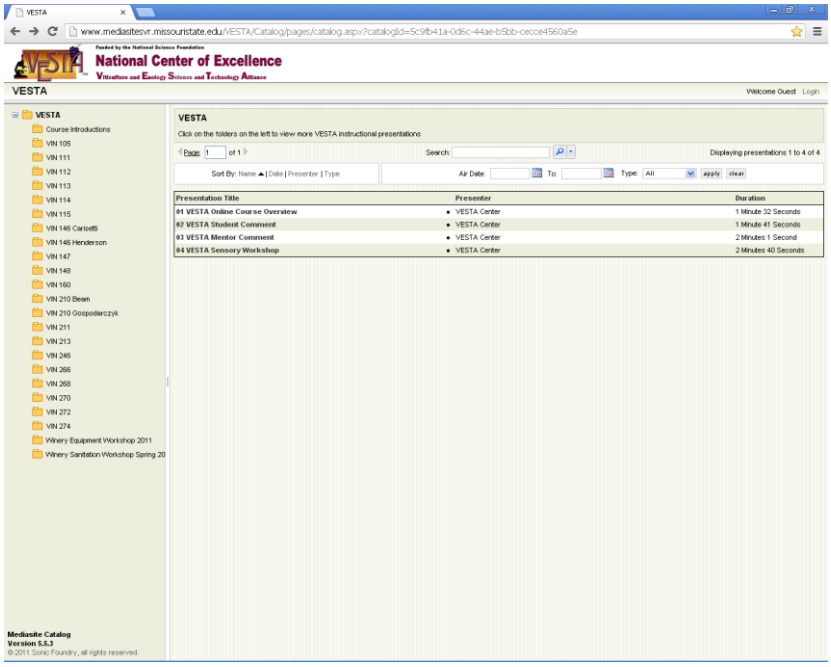


Figura 2.9: VESTA

2.2.2 SOLUCIONES CON SISTEMA DE CATALOGACIÓN PROPIO

En nuestro camino en busca de posibles soluciones que se asemejarán todo lo posible a los objetivos de este proyecto, no encontramos ninguna que nos satisfazca.

El problema de hacer uso de un sistema tan estandarizado como *Mediasite* residía en que cualquier tipo de implementación en un sistema tercero sería limitado y por ello la mayoría de compradores de *Mediasite* preferían tomar la solución sencilla que era hacer uso de su interfaz.

Por ello, nos fijamos en otro proyecto que se había realizado en la universidad, que fue la implantación dentro de la infraestructura de la UC3M del sistema *Opencast Matterhorn* [7]. Aunque el procedimientento y muchos aspectos son diferentes a los objetivos que buscamos en este proyecto, existe una parte que puede resultamos interesante que es la pasarela que se realiza para publicar contenidos generados con *Opencast* en ArcaMM.

Salvando las diferencias que conlleva tratar con un proyecto de código abierto a otro con un software comercial, existe un cierto paralelismo entre ambos objetivos finales. Cuando existe ya una base de datos de contenidos en *Opencast*, se realiza una aplicación web que es integrará dentro de la plataforma de contenidos multimedia de la UC3M. Esta aplicación permite un acceso al personal del Área de Audiovisuales a los contenidos que pueden ser publicados, eligiendo el vídeo publicar. Realmente esto es una de las cosas que se nos ha propuesto realizar, con lo que deberemos tener de referencia este proyecto, ya que nosotros perseguimos el mismo fin y además integrado en el mismo sistema en el que lo queremos hacer nosotros.

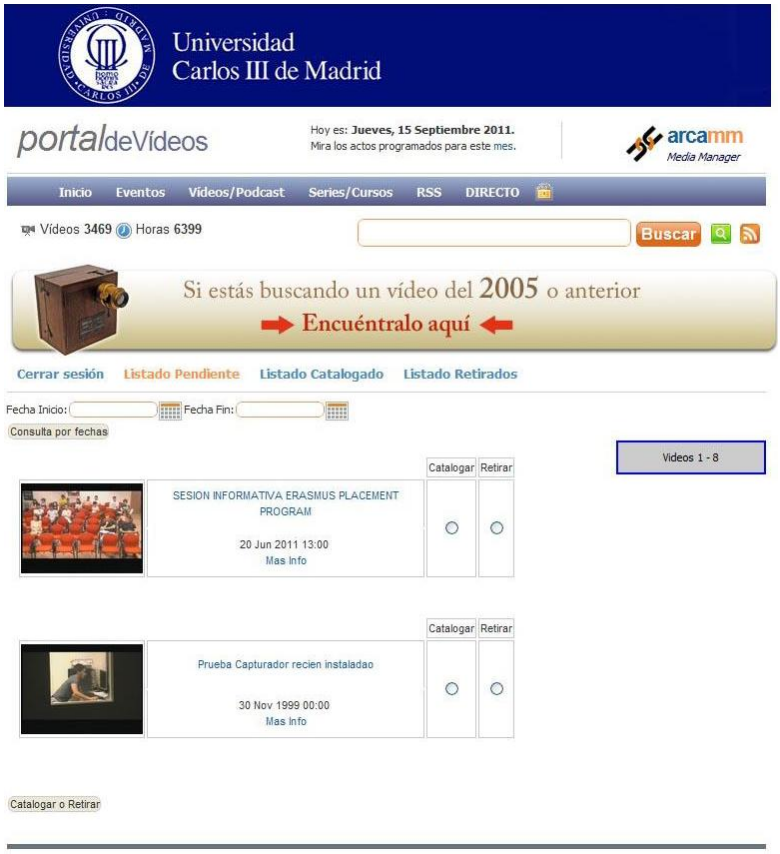


Figura 2.10: Opencast Matterhorn en la UC3M

2.3 ENTORNO DE DESARROLLO SELECCIONADO

En esta sección se analiza las herramientas necesarias para la realización de este proyecto. En definitiva vamos a detallar el entorno de desarrollo integrado (IDE) que hemos utilizado para realizar el trabajo de preproducción del proyecto antes de que fuera integrado en el servidor donde finalmente quedaría alojada nuestra aplicación.

Cabe destacar que la herramienta principal para la realización de este proyecto es poseer el software *Mediasite*, con su correspondiente licencia, ya que sin ello no se tendría acceso al servidor principal y por tanto el resto de herramientas no tendría ningún sentido tenerlas.

2.3.1 SISTEMA OPERATIVO

La primera elección a realizar por el equipo de trabajo es elegir el sistema operativo sobre el que montaremos nuestro entorno integrado. En un principio no se requiere un sistema operativo específico, con lo que la elección sería irrelevante, y dependería del gusto del programador.

Sin embargo, al analizar la estructura del servidor donde se alojará nuestra aplicación, observamos que se correspondía con un servidor Linux, con lo que se tomó la decisión de utilizar el sistema operativo Ubuntu. El motivo de la decisión fue trabajar en local en un entorno lo más parecido posible al del servidor. Esto sería positivo ya que en el momento que nuestro trabajo pasara a producción, la integración se realizará de forma sencilla, ya que no tendríamos que cambiar rutas de acceso a los archivos como pasaría si utilizáramos un sistema operativo basado en Windows o MacOS X.

2.3.2 MYSQL

En el momento que nos plantearon la realización de este proyecto, no teníamos todavía muy claro si era necesario crear una base de datos, ya que podríamos simplemente leer la información del servidor sin que interviniera ningún agente intermedio. A pesar de ello, sabíamos que si finalmente realizábamos una base de datos esta debía de ser gestionada por MYQSL, ya que debido a la cantidad de datos que queremos extraer del servidor de Mediasite, necesitamos un gestor que almacene todos estos datos de forma segura y organizada.

Desde el Área de Audiovisuales, no se nos impuso el uso de ningún tipo de herramienta, pero si se nos invitó a utilizar herramientas de código libre, como es el caso de MYSQL. Además de este motivo, el gestor encajaba perfectamente en los requerimientos que imponía el EDAS, ya que solo permitía desarrollar en C y PHP, con lo que MYSQL nos ofrecía API`s para ambos lenguajes [2].

Otros motivos que nos llevaron a elegir esta opción fueron: la base de datos de ArcaMM la gestionaba MYSQL, la gran portabilidad entre sistemas o la seguridad que se ofrece al tener que gestionar la base de datos mediante usuario y contraseña.

2.3.3 PHP

Cuando llegó por primera vez a nuestras manos el EDAS que proporcionaba *Mediasite* para desarrolladores, observamos que se nos permitía tener acceso a los datos del servidor mediante PHP o C, con lo que deberíamos elegir el lenguaje más apropiado a nuestro fin y a nuestros conocimientos.

Nuestra idea inicial era la de realizar un extractor de datos de los contenidos almacenados en el servidor de *Mediasite,* para su posterior publicación en la plataforma de contenidos multimedia ArcaMM. Teníamos claro que la aplicación web encargada de realizar la publicación la íbamos a realizar con código HTML, Java Script y PHP, ya que el Área de Audiovisuales nos impuso estos requisitos, ya que toda la plataforma ArcaMM está realizada con estas herramientas. El motivo era para no crear un conflicto con el resto de aplicaciones que ya estaban en funcionamiento y por ello una posible modificación en cualquiera de ellas.

Ya que PHP iba a ser utilizado en la segunda parte de este proyecto, creímos conveniente utilizar este lenguaje, ya que así estaríamos familiarizados con el lenguaje, otorgándonos una mayor fluidez en la realización de dicha parte. Además PHP nos ofrecía la posibilidad de realizar scripts en línea de comandos que no necesitaban interacción con el usuario. La importancia de estos scripts en nuestro proyecto es que al no necesitar interacción, podrían ser ejecutados de forma programada, muy útil para la tarea de actualización de datos, en el caso de que finalmente se llevara a cabo.

A pesar de que PHP estaba impuesto en la parte de realización de la interfaz, este lenguaje lo teníamos concebido como primera opción ya que al ser un lenguaje del lado del servidor, permite procesar formularios, mostrar información de una base de datos de forma dinámica o la facilidad de incorporarlos con etiquetas HTML para generar páginas web dinámicas [2]..

2.3.4 APACHE2

En la realización de este proyecto, es un requisito necesario el uso de un servidor web, tanto para la parte de extracción de información del servidor de *Mediasite* como para llevar a cabo la integración con ArcaMM.

El servidor elegido es Apache2, ya que permite el uso de lenguajes de programación como PHP, Perl, Phython o Ruby, y la comunicación con sistemas gestores de bases de datos como MYSQL. Este es el motivo principal del uso de este servidor ya que como hemos mencionado con anterioridad utilizaremos el lenguaje de programación PHP y el gestor de bases de datos MYSQL [2].

2.3.5 OTRAS HERRAMIENTAS

A continuación enumeraremos otras herramientas utilizadas en alguna fase de la realización de este proyecto:

- SOAP: Para comunicarnos con la base de datos de Mediasite debemos crear un cliente SOAP, ya que sin él no podemos extraer los datos. Para ello debemos de descargamos de la página de la distribución de PHP la extensión PHP-SOAP.

- PHPMYADMIN: Para administrar la base de datos creada con MYSQL utilizamos esta herramienta. Gracias a esta aplicación creamos las tablas necesarias, con sus respectivos campos, así como la integridad referencial entre ellas.

- JAVASCRIPT: Utilizamos este lenguaje del lado del cliente para dar vistosidad a la interfaz que realizaremos en la segunda parte de este proyecto con la ayuda de la librería JQuery. Además también nos fue muy útil para la verificación de formularios en dicha parte [3] [4].

- HTML: Utilizamos este lenguaje para realizar la segunda parte de este proyecto unido a PHP y JavaScript.

- FILEZILLA: A través de este cliente FTP realizamos todas las transferencias de archivos necesarias para poner nuestra aplicación en producción.

- BLUEFISH: Con este editor hemos realizado cada uno de los scripts pertenecientes a este proyecto, independientemente del lenguaje utilizado en cada uno de ellos.

2.4 CONCLUSIONES DEL ANÁLISIS

A través de este análisis hemos llegado a una serie de conclusiones que nos pueden servir de ayuda en la resolución del problema que se nos planteó.

Observando los sistemas que existen en el mercado de captura de clases, en cuanto a prestaciones *Panopto*, *Echo360* y *Mediasite* son muy similares pero creemos que la elección de la Universidad Carlos III para utilizar *Mediasite* en detrimento de los otros es principalmente por dos razones: liderazgo en el mercado y interfaz más vistosa.

En nuestra búsqueda de sistemas de catalogación similares al que queremos realizar, nos damos cuenta de que resulta más sencillo poner en práctica las funcionalidades que nos ofrece la propia plataforma *Mediasite,* que realizar otras nuevas como es el fin de este proyecto. A pesar de ello observamos que la Universidad de Wisconsin plantea una solución a medio camino de la que queremos llevar a cabo, utilizando además la documentación que les proporciona la plataforma. Al no encontrar una solución convincente con *Mediasite,* observamos como se ha realizado la implantación de *Opencast Matterhorn* en la UC3M, que guarda muchas similitudes con la aplicación que deseamos realizar. Además creemos que al ser una aplicación de captura de clases integrada en ArcaMM, nos puede resultar mucho más interesante que un sistema de catalogación con *Mediasite* en otro tipo de plataforma de otra Universidad o centro educativo.

Finalmente en la elección de las herramientas necesarias, hemos tenido muy poco margen, ya que la gran mayoría eran requeridas por el Área de Audiovisuales o bien de forma directa o indirecta. A pesar de ello hemos creído conveniente amoldar las decisiones tomadas a la infraestructura existente, ya que así evitábamos posibles problemas de compatibilidad con el resto de aplicaciones. No obstante si el equipo de trabajo hubiera encontrado incompatibilidades con los requerimientos impuestos por el cliente, se hubieran puesto en conocimiento de él para encontrar una posible solución. En el caso de las herramientas no existió ningún problema en este aspecto pero sí ocurrió en el caso de alguna funcionalidad que no se podría realizar (captura de un fotograma de un contenido).

CAPÍTULO 3 ESPECIFICACIÓN DE REQUISITOS

3.1 INTRODUCCIÓN

En este capítulo realizaremos una Especificación de Requisitos (ERS), en el que expondremos los requisitos necesarios para el buen funcionamiento del sistema a implantar. Este capítulo va dirigido a todas esas personas que solicitan la nueva aplicación y desean conocer los requisitos del sistema, como al equipo encargado de implantar dicho sistema.

Para el correcto desarrollo del escrito se utiliza como guía el estándar IEEE 830 [1] que permite definir los requisitos del sistema de una forma sencilla de comprender para el lector.

En primer lugar realizaremos una descripción general del sistema a desarrollar haciendo hincapié en aquellos factores que afectan al producto y a sus requisitos. En varias subsecciones abordaremos la perspectiva del producto. Ahí relacionaremos el futuro sistema con otros sistemas, las funciones del producto a grandes rasgos, las características generales de los usuarios finales del producto, como interaccionará nuestro producto con otros sistemas, las restricciones impuestas por el cliente o por sistemas secundarios, y las suposiciones y dependencias que se dan por hechas a la hora de plantear la realización del producto.

Finalmente, en la siguiente sección expondremos los requisitos específicos del producto a realizar, con un nivel de detalle suficiente para los diseñadores de este. Los requisitos específicos estarán divididos en: requisitos funcionales, interfaces externas, requisitos de rendimiento, restricciones de diseño y atributos del sistema.

3.2 DESCRIPCIÓN GENERAL

En esta sección se describen todos aquellos factores que afectan al producto y sus requisitos, haciendo hincapié en su contexto.

3.2.1 PERSPECTIVA DEL PRODUCTO

La necesidad del Área de Audiovisuales de crear una aplicación que consiga publicar el contenido generado con la herramienta *Mediasite* en la plataforma de contenidos multimedia de la UC3M, es la principal motivación que llevó a este departamento de llevar este proyecto a cabo.

El nombre que damos a la aplicación es “Publicador de contenido Mediasite”, que está compuesta por un script llamado “Actualizador de datos” y de la interfaz de publicación “Rol mediasite” con sus respectivas funciones. La nueva aplicación debe formar parte de ArcaMM y debe ser invisible para el usuario final que recibe el contenido. Al ser ArcaMM una plataforma conocida por la mayor parte de la comunidad universitaria de la UC3M, se desecha cualquier solución que no conciba que la nueva aplicación forme parte de ArcaMM. El motivo principal de esta decisión es que antes de de la realización de este proyecto, el usuario final solo podía acceder al contenido a través de la interfaz proporcionado por el distribuidor de *Mediasite.*

En la siguiente figura podemos observar como es la interacción con los sistemas implicados:

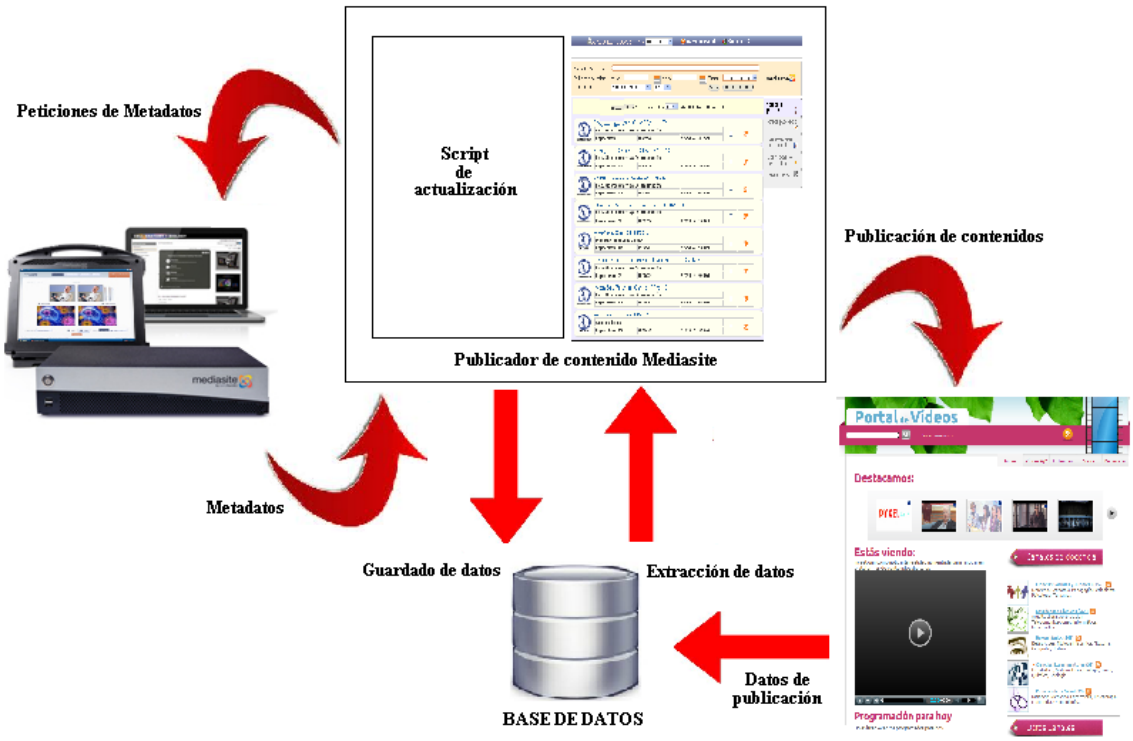


Figura 3.1: Interacción con otros sistemas

El producto a realizar, *Publicador de contenido Mediasite*, es una aplicación formada por un script de actualización, de una base de datos creada para dicho producto y una aplicación a la que se accederá vía web.

Al formar parte de un sistema mayor como es ArcaMM, debe cumplir una serie de requisitos implícitos en dicho sistema. Tanto el script de actualización como la interfaz gráfica deben formar parte de ArcaMM. El primero porque si se realiza una base de datos , esta debe de estar alojada en el mismo servidor que el resto de bases de datos que forman parte de ArcaMM y el segundo porque estará integrado en su interfaz con lo que tendrá que cumplir requisitos referentes al estilo o la comunicación entre aplicaciones alojadas en ArcaMM.

Debemos tener muy presente que nuestra aplicación debe de convivir con otras aplicaciones ya realizadas y que ya están en funcionamiento dentro de la plataforma. Estas aplicaciones únicamente pueden ser utilizadas por determinados usuarios que hayan sido dados de alta por parte del administrador del sistema. A continuación indicamos brevemente cuales son cada una de estas aplicaciones.

- **Admin** es el rol reservado a administradores y permite dar de alta a nuevos usuarios, asignar roles y configurar todo lo relativo al portal de vídeos.

- **Manager** permite publicar, editar y borrar cualquier contenido de ArcaMM.

- **Subtitulador** permite añadir subtítulos a cualquier contenido previamente publicado

- **Editor** es similar a manager pero sólo se pueden publicar determinados contenidos, así como editar o borrar solo contenido publicado por el propio usuario

- **AdminAgenda** permite añadir, editar o borrar los actos que se llevarán a cabo en las salas de audiovisuales de la UC3M.

- **ITunesU** permite publicar contenido en la plataforma iTunesU de la UC3M.

- **Encoder** permite codificar cualquier contenido publicado a mp4 para su posterior descarga a través de un ordenador o dispositivo móvil.

- **Entrans, Catrans, Eutrans y Gltrans** permiten gestionar y modificar los diccionarios utilizados en los posibles lenguajes en los que puede mostrarse el portal de vídeos: inglés, catalán, euskera y gallego.

- **Mediasite** permite recuperar todo el contenido generado con *Mediasite* para su publicación en ArcaMM. Entre otras funciones recupera las presentaciones programas, estadísticas de publicación y configuración de la herramienta de actualización de contenidos.

Esta última herramienta es la que vamos a llevar a cabo a lo largo de este proyecto, junto a un módulo que realizará estadísticas de publicación a petición del usuario. Además necesitamos realizar también un script que nos permita recuperar toda la información de la base de datos de *Mediasite* para poder ser tratada.

3.2.2 FUNCIONES DEL PRODUCTO

Las principales funciones que debe realizar la aplicación "Publicador de contenido Mediasite" son las siguientes:

- Recuperar toda la información referente a los contenidos generados con *Mediasite*
- Realizar tareas de actualización o borrado de contenidos y reflejarlo en la base de datos creada.
- Actualización automática.
- Acceso a través de usuario y contraseña
- Mostrar contenidos publicados, no publicados y programados
- Buscar presentaciones según unos criterios de búsqueda (nombre, autor, campus y fecha de realización)
- Ordenar resultados según un criterio
- Paginación de resultados
- Ventana emergente con información completa del vídeo
- Generar estadísticas de publicación según criterios elegibles
- Cambiar configuración de script de actualización

- Auto rellenar información del contenido antes de publicar
- Publicar contenidos en ArcaMM
- Comunicación con ArcaMM para actualización en la base de datos del estado de un contenido(Publicado o no publicado)

3.2.3 CARACTERÍSTICAS DE LOS USUARIOS

Podemos identificar dos tipos de usuarios del producto: usuarios directos y usuarios indirectos.

Los usuarios directos de la aplicación son los miembros del Área de Audiovisuales, ya que ellos y las personas que ellos decidan, ajenas a su departamento, tendrán acceso a la aplicación realizada. Al ser administradores del sistema decidirán que personas tienen acceso a esta aplicación y quién no.

Por otro lado los usuarios indirectos del producto son cualquier persona con acceso a un PC o dispositivo móvil con conexión a internet que acceda al Portal de vídeos de la UC3M, en particular los miembros de la comunidad universitaria de la UC3M. En el amplio abanico de estos usuarios, particularizando en los miembros del espacio UC3M, se encuentran alumnos, profesores, miembros de departamentos o miembros de empresas o entidades que en algún momento han tenido algún tipo de relación tanto profesional como académica y que estén interesados en recuperar un determinado contenido.

3.2.4 INTERACCIÓN CON OTROS SISTEMAS

La aplicación va a interactuar con varios sistemas externos cuyo mantenimiento o buen funcionamiento son ajenos a este proyecto y por tanto no dependen del desarrollador. Esto podría ocasionar problemas a los usuarios que intentarán tener acceso a la aplicación.

Lo sistemas con los que interactúa la aplicación son los siguientes:

- **Servidor *Mediasite1*.** A través de este servidor es con el que se comunica directamente el usuario para tener acceso a un contenido generado con *Mediasite* ya que es el que gestiona los vídeos.

- **Servidor Mediasite2.** En este servidor es donde se alojan todos los contenido generados con *Mediaste* y es el encargado de realizar el streaming de todas las presentaciones.

- **Servicio LDAP.** A través de este sistema de directorios el usuario accederá al área restringida de ArcaMM. Tanto ArcaMM como nuestra aplicación son ajenas a este acceso, ya que ArcaMM es un mero interlocutor entre el usuario que introduce el login de correo y su contraseña y estos datos son enviados al servicio de autentificación de correo que es el encargado de verificar si estos son correctos o no.

- **Portal ArcaMM.** Al estar nuestra aplicación integrada en el portal de vídeos depende totalmente del funcionamiento del servidor donde se encuentre alojado ArcaMM.

3.2.5 RESTRICCIONES

A continuación se muestran las restricciones con las que cuenta la aplicación, unas de ellas impuestas por el cliente y otras por el propio sistema *Mediasite:*

- Todas las herramientas utilizadas en la aplicación deben estar englobadas en el denominado software libre.

- La comunicación con el servidor de *Mediasite* se realiza a través de SOAP, es decir por intercambio de datos en XML.

- El script de actualización se deberá de ejecutar de forma automática sin intervención de ninguna persona física

- El script de actualización debe poder ser configurable desde una interfaz gráfica en previsión de cambios de versión, servidor , usuario y contraseña del sistema *Mediaste*

- La interfaz gráfica debe ser realizada con PHP, HTML y JavaScript

- La hojas de estilo creadas deben estar basadas en las que se utilizan en el área restringida de usuarios del Portal de vídeos de la UC3M.

- La criticidad del sistema reside en el tiempo de espera de las consultas a la base de datos.

- Todos los ficheros que formarán parte de la interfaz gráfica, deberán comprobar que usuario intenta acceder, si la autentificación se realizó de forma correcta y si el usuario tiene asignado el rol *Mediasite*.

3.2.6 SUPOSICIONES Y DEPEDENDENCIAS

La aplicación "Publicador de contenido Mediasite*"*, depende de tres servidores para su correcto y total funcionamiento.

En primer lugar hace uso del servicio de directorio proporcionado por el Servicio de Informática de la UC3M con el que se realiza la autentificación en el portal de vídeos con el login y contraseña de correo corporativo de la UC3M.

En segundo lugar se hace uso del servidor donde están alojados los vídeos capturados con *Mediasite*, ya que el enlace para poder visualizar tanto los contenidos publicados como los no publicados se realiza a través de dicho servidor.

La tercera y última dependencia se refiere al servidor donde quedará alojada nuestra aplicación, que será el mismo donde se aloja el Portal de vídeos de la UC3M y el resto de herramientas corporativas de acceso restringido.

En el caso de que cualquiera de estos servicios o servidores dejarán de funcionar, el "Publicador de contenido Mediasite*"* no funcionaría de forma correcta. Los casos más críticos serían el del servidor *Mediasite* y ArcaMM, ya que no permitirían ni la visualización ni la publicación de contenidos. En el caso de que fallara el servicio de directorio, para el usuario que únicamente visualiza los vídeos no existiría ningún problema de acceso a los contenidos publicados, pero el personal del Área de Audiovisuales no podría publicar nuevos vídeos en ArcaMM, como hacer uso del resto de funciones de la aplicación, ya que no podría acceder a ella.

3.3 REQUISITOS ESPECÍFICOS

En esta sección expondremos los requisitos con un nivel de detalle lo suficientemente grande que permita a los desarrolladores diseñar un sistema que satisfaga estos requisitos y que permita al grupo de pruebas planificar y realizar las pruebas que demuestren si el sistema satisface, o no, los requisitos. Todo requisito expuesto aquí describirá comportamientos externos del sistema perceptibles por cualquiera de los agentes involucrados en la aplicación

En la sección 3.3.1 se especificarán los requisitos funcionales que son los que tratan sobre la funcionalidad de la aplicación.

En la sección 3.3.2 se mostrarán las interfaces externas. Por interfaces externas se entienden tanto las vistas de la aplicación a realizar con las que el usuario puede interactuar como las interfaces de otros sistemas involucrados en cualquier función de la aplicación

En la sección 3.3.3 se especificarán los requisitos referidos al rendimiento que de la aplicación.

En la sección 3.3.4 se expondrán las restricciones de diseño que limitan ciertas funciones de la aplicación

Por último, en la sección 3.3.5 se definirán los atributos de los que está dotada la aplicación.

Para definir los requisitos se utilizará una tabla como la siguiente:

ID	Identificador	**Nombre**	Nombre de requisito	
Descripción	Descripción del requisito. Explicación que permita a cualquier programador implementarlo sin necesidad de ninguna otra información			
Dependencias	Otro Identificador	**Fecha**	dd/mm/aaaa	
Prioridad	(*) Alta		(*) Normal	(*) Baja
Necesidad	(*) Requerido		(*) Deseable	(*) Opcional

Tabla 3.1: Identificador – Requisito de ejemplo

Los identificadores elegidos para los requisitos constan de dos partes diferenciadas:

- La primera se trata de un código de dos letras que identifica el tipo de requisito.

  a) RF- Requisito funcional
  b) RI - Requisito de interfaz gráfica
  c) RR - Requisito de rendimiento
  d) RD - Restricción de diseño

- La segunda se corresponde con un número de tres cifras que comenzará en 001 y que sirve para ordenar los requisitos de un mismo tipo.

3.3.1 REQUISITOS FUNCIONALES

ID	RF-001	**Nombre**	El script recuperará todos los metadatos de *Mediasite*	
Descripción	El script de actualización debe ser capaz de recuperar cualquier tipo de información referente a un contenido generado con *Mediasite*, siempre y cuando exista una función en el EDAS que pueda realizar dicha función.			
Dependencias		**Fecha**	06/02/2012	
Prioridad	(*) Alta		( ) Normal	( ) Baja
Necesidad	(*) Requerido		( ) Deseable	() Opcional

Tabla 3.2: RF-001 – El script recuperará todos los metadatos de Mediasite

ID	RF-002	**Nombre**	El script guarda los metadatos en la base de datos
Descripción	Los metadatos referentes a los contenidos pueden ser guardados en una base de datos si el desarrollador lo cree conveniente.		
Dependencias	RF-001	**Fecha**	06/02/2012
Prioridad	() Alta	(*) Normal	( ) Baja
Necesidad	() Requerido	( ) Deseable	(*) Opcional

Tabla 3.3: RF-002 – El script guarda los metadatos en la base de datos

ID	RF-003	**Nombre**	Agregar, modificar y eliminar metadatos de la base de datos
Descripción	En el caso que se decida guardar los metadatos referentes a los contenidos, es necesario que se realicen tareas periódicas de agregado, modificado o borrado de estos metadatos, cuando esto ocurra en la base de datos de *Mediasite.*		
Dependencias	RF-002	**Fecha**	06/02/2012
Prioridad	() Alta	() Normal	(*) Baja
Necesidad	(*) Requerido	( ) Deseable	( ) Opcional

Tabla 3.4: RF-003 – Agregar, modificar y eliminar metadatos de la base de datos

ID	RF-004	**Nombre**	Actualización de la base de datos de forma automática
Descripción	En el caso que se decida guardar los metadatos referentes a los contenidos, es necesario que se realicen tareas periódicas de agregado, modificado o borrado de contenido de forma automática. Esto lo podemos realizar con el administrador regular de procesos *cron* disponible en el servidor de un sistema *Unix*		
Dependencias	RF-002, RF-003	**Fecha**	06/02/2012
Prioridad	() Alta	(*) Normal	( ) Baja
Necesidad	(*) Requerido	( ) Deseable	( ) Opcional

Tabla 3.5: RF-004 – Actualización de la base de datos de forma automática

ID	RF-005	**Nombre**	Acceso a la aplicación con login y contraseña de correo
Descripción	El acceso a la aplicación debe de realizarse a través del área restringida de http://arcamm.uc3m.es/arcamm, introduciendo el login y contraseña de correo corporativo. Para ello la aplicación se debe de conectar con el servidor donde se aloja el servicio de directorio de la UC3M y pasarle el login y la contraseña introducida por el usuario. El servicio de directorio se encarga de realizar la autentificación, devolviéndonos el resultado. En el caso de que el servicio nos devuelva que la autentificación se realizó de forma correcta dejaremos acceder al usuario, en caso contrario deberá introducir los datos de nuevo para acceder. Además el usuario debe tener asumido el rol *Mediasite* para acceder a la aplicación.		
Dependencias		**Fecha**	06/02/2012
Prioridad	(*) Alta	() Normal	( ) Baja
Necesidad	(*) Requerido	( ) Deseable	( ) Opcional

Tabla 3.6: RF-005 – Acceso a la aplicación con login y contraseña de correo

ID	RF-006	**Nombre**	Mostar los contenidos generados con *Mediasite*	
Descripción	Se deben mostrar al usuario todos los contenidos generados con *Mediasite* disponibles para publicación, así como los ya publicados en ArcaMM y los contenidos que están programados para ser realizados en un futuro			
Dependencias	RF-005, RF-001	**Fecha**	06/02/2012	
Prioridad	(*) Alta		() Normal	( ) Baja
Necesidad	(*) Requerido		( ) Deseable	( ) Opcional

Tabla 3.7: RF-006 – Mostar los contenidos generados con Mediasite

ID	RF-007	**Nombre**	Buscar contenidos por nombre y fecha	
Descripción	Se debe dar la opción al usuario de poder realizar un filtrado entre todos los contenidos existentes. Al menos se debe de poder filtrar por nombre de presentación y por un intervalo de fechas de realización del contenido.			
Dependencias	RF-005	**Fecha**	06/02/2012	
Prioridad	(*) Alta		() Normal	( ) Baja
Necesidad	(*) Requerido		( ) Deseable	( ) Opcional

Tabla 3.8: RF-007 – Buscar contenidos por nombre y fecha

ID	RF-008	**Nombre**	Buscar contenidos por autor y campus	
Descripción	El módulo de búsqueda ofrecerá otros criterios de búsqueda como el nombre del autor o el campus. Además permitirá la ordenación de resultados según un criterio tanto ascendente como descendente.			
Dependencias	RF-007	**Fecha**	15/03/2012	
Prioridad	( ) Alta		() Normal	(*) Baja
Necesidad	( ) Requerido		(*) Deseable	( ) Opcional

Tabla 3.9: RF-008 – Buscar contenidos por autor y campus

ID	RF-009	**Nombre**	Mostrar detalles de presentaciones	
Descripción	Mostrará todos los detalles de cada una de las presentaciones. Entre los detalles tiene que aparecer, nombre, autor, duración y fecha de grabación, número de diapositivas, carpeta en el directorio de *Mediasite*, visor de reproducción, y detalles de codificación de vídeo y audio utilizados en la toma del contenido.			
Dependencias	RF-006	**Fecha**	06/02/2012	
Prioridad	(*) Alta		() Normal	( ) Baja
Necesidad	(*) Requerido		( ) Deseable	( ) Opcional

Tabla 3.10: RF-009 – Mostrar detalles de presentaciones

ID	RF-010	**Nombre**	Paginación de contenidos mostrados	
Descripción	Debido a la gran cantidad de contenidos, estos se deberán mostrar en distintas páginas por las que el usuario podrá navegar. Aparecerán botones como "Primero", "Anterior", "Siguiente" y "Último", así como una lista desplegable con todas las páginas resultantes, a la que el usuario podrá navegar directamente. La paginación se debe de realizar tanto si hay filtrado de resultados como si no lo hay.			
Dependencias	RF-006	**Fecha**	15/03/2012	
Prioridad	( ) Alta		(*) Normal	( ) Baja
Necesidad	( ) Requerido		(*) Deseable	( ) Opcional

Tabla 3.11: RF-010 – Paginación de contenidos mostrados

ID	RF-011	**Nombre**	Generar estadísticas de contenidos en un intervalo temporal	
Descripción	Ofrecer la posibilidad de realizar estadísticas dinámicas dependiendo del valor que dé el usuario a unos determinados criterios. Al menos se pude poder elegir un intervalo de fechas así como entre vídeos publicados y no publicados.			
Dependencias	RF-005, RF-001	**Fecha**	02/04/2012	
Prioridad	( ) Alta		(*) Normal	( ) Baja
Necesidad	( ) Requerido		(*) Deseable	( ) Opcional

Tabla 3.12: RF-011 – Generar estadísticas de contenidos en un intervalo temporal

ID	RF-012	**Nombre**	Generar estadísticas de contenidos por campus o autor
Descripción	Ofrecer otros criterios para realizar estadísticas como el campus o el autor que realizó el contenido.		
Dependencias	RF-011	**Fecha**	16/04/2012
Prioridad	( ) Alta	( ) Normal	(*) Baja
Necesidad	( ) Requerido	( ) Deseable	(*) Opcional

Tabla 3.13: RF-012 – Generar estadísticas de contenidos por campus o autor

ID	RF-013	**Nombre**	Resultados de estadísticas en horas y número de grabaciones
Descripción	Los resultados que se deben de ofrecer son: el número de horas y el número de grabaciones realizadas en el intervalo de tiempo elegido. Además se debe mostrar un resumen numérico de la gráfica generada.		
Dependencias	RF-011	**Fecha**	02/04/2012
Prioridad	( ) Alta	(*) Normal	( ) Baja
Necesidad	( ) Requerido	(*) Deseable	() Opcional

Tabla 3.14: RF-013 – Resultados de estadísticas en horas y número de grabaciones

ID	RF-014	**Nombre**	Configuración del script por parte del usuario
Descripción	El sistema debe de permitir cambiar la configuración del script de actualización por parte del usuario en previsión de posibles cambios de servidor o autentificación del sistema *Mediasite.*		
Dependencias	RF-005	**Fecha**	02/04/2012
Prioridad	( ) Alta	(*) Normal	( ) Baja
Necesidad	(*) Requerido	() Deseable	( ) Opcional

Tabla 3.15: RF-014 – Configuración del script por parte del usuario

ID	RF-015	**Nombre**	Publicar contenido en ArcaMM	
Descripción	El usuario debe elegir del contenido mostrado, que vídeo desea publicar. El sistema debe de ser capaz de redireccionar al usuario a una nueva ventana donde podrá rellenar los datos de publicación.			
Dependencias	RF-006	**Fecha**	06/02/2012	
Prioridad	(*) Alta	() Normal	( ) Baja	
Necesidad	(*) Requerido	() Deseable	( ) Opcional	

Tabla 3.16: RF-015 – Publicar contenido en ArcaMM

ID	RF-016	**Nombre**	Rellenado automático de campos en la ventana de publicación	
Descripción	Se rellenarán de forma automática todos los campos posibles (nombre, descripción, fecha, autor, codecs de vídeo y audio, etc.) basándose en los datos con los que fueron generados los contenidos en *Mediasite*. A pesar de ello se da la opción al usuario que publica el contenido, de poder modificar cualquier campo si así lo considera.			
Dependencias	RF-014	**Fecha**	15/03/2012	
Prioridad	() Alta	(*) Normal	( ) Baja	
Necesidad	( ) Requerido	(*) Deseable	( ) Opcional	

Tabla 3.17: RF-016 – Rellenado automático de campos en la ventana de publicación

ID	RF-017	**Nombre**	Comunicación de la base de datos con ArcaMM
Descripción	Nuestro sistema deberá saber en qué momento se ha publicado un vídeo así como si alguno de los archivos publicados ha sido eliminado de ArcaMM. Para ello debemos tener una tabla en la base de datos de ArcaMM, o bien una propia que creemos, donde tengamos las referencias (ids) de cada uno de los contenidos. Asociado a cada referencia deberá existir un campo que indique si el vídeo esta publicado en ArcaMM. ArcaMM nos indicará si se agregó el nuevo contenido, así como si este es eliminado del portal. En el caso de que suceda esto, la aplicación deberá cambiar el estado del campo asociado a dicho contenido.		
Dependencias	RF-014	**Fecha**	06/02/2012
Prioridad	(*) Alta	() Normal	( ) Baja
Necesidad	(*) Requerido	() Deseable	( ) Opcional

Tabla 3.18: RF-017 – Comunicación de base de datos con ArcaMM

3.3.2 INTERFACES EXTERNAS

ID	RI-001	**Nombre**	Páginas HTML válidas
Descripción	Todas las páginas deben ser HTML válido. Deben superar el test del W3C (http://validator.w3.org)		
Dependencias		**Fecha**	06/02/2012
Prioridad	(*) Alta	( ) Normal	( ) Baja
Necesidad	(*) Requerido	( ) Deseable	( ) Opcional

Tabla 3.19: RI-001 – Páginas HTML válidas

ID	RI-002	**Nombre**	Hojas de estilo CSS válidas	
Descripción	Todas las hojas de estilo CSS deben ser válidas. Deben superar el test del W3C (http://jigsaw.w3.org/css-validator)			
Dependencias		**Fecha**	06/02/2012	
Prioridad	(*) Alta	( ) Normal	( ) Baja	
Necesidad	(*) Requerido	( ) Deseable	( ) Opcional	

Tabla 3.20: RI-002 – Hojas de estilo CSS válidas

ID	RI-003	**Nombre**	Compatibilidad del sistema con varios navegadores	
Descripción	El sistema debe ser compatible con Internet Explorer 8.0 o posterior, Mozilla Firefox 7.5 o posterior, Google Chrome 8.0 o posterior y Safari 5 o posterior			
Dependencias		**Fecha**	06/02/2012	
Prioridad	( ) Alta	(*) Normal	( ) Baja	
Necesidad	( ) Requerido	(*) Deseable	( ) Opcional	

Tabla 3.21: RI-003 – Compatibilidad del sistema con varios navegadores

ID	RI-004	**Nombre**	Comunicación a través de URL	
Descripción	La aplicación será capaz de recibir parámetros desde una URL con un formato concreto. Los parámetros son t*ab, name, dateI, campus, type* y *order*.			
Dependencias	RI-001, RI-002, RI-003	**Fecha**	15/03/2012	
Prioridad	( ) Alta	(*) Normal	( ) Baja	
Necesidad	( ) Requerido	(*) Deseable	( ) Opcional	

Tabla 3.22: RI-004 – Comunicación a través de la URL

ID	RI-005	**Nombre**	Uso de hojas de estilo de ArcaMM	
Descripción	El sistema debe de utilizar las hojas de estilo ya creadas para el portal ArcaMM.			
Dependencias	RI-002	**Fecha**	06/02/2012	
Prioridad	( ) Alta	(*) Normal	( ) Baja	
Necesidad	(*) Requerido	( ) Deseable	( ) Opcional	

Tabla 3.23: RI-005 – Uso de hojas de estilo de ArcaMM

ID	RI-006	**Nombre**	Módulo de búsqueda	
Descripción	El sistema mostrará un módulo de búsqueda con un campo de texto y dos más para introducir la fecha inicial y la fecha final. Tres listas desplegables: una con los tres campus, otra con los criterios en los que puede ser ordenará la búsqueda y una última para fijar el orden ascendente o descendente del resultado			
Dependencias	RI-004,RI-005	**Fecha**	15/03/2012	
Prioridad	( ) Alta	(*) Normal	( ) Baja	
Necesidad	(*) Requerido	( ) Deseable	( ) Opcional	

Tabla 3.24: RI-006 – Módulo de búsqueda

ID	RI-007	**Nombre**	Vista vídeos no publicados	
Descripción	El sistema mostrará una vista con todas las presentaciones no publicadas, los botones de navegación entre las distintas páginas de resultados y los detalles más significantes (campus, título, autor, duración, fecha de publicación y diapositivas). Además aparecerán dos botones: el primero para publicar el vídeo y el segundo para visualizar todos los detalles referentes al vídeo.			
Dependencias	RI-004,RI-005	**Fecha**	15/03/2012	
Prioridad	(*) Alta		( ) Normal	( ) Baja
Necesidad	(*) Requerido		( ) Deseable	( ) Opcional

Tabla 3.25: RI-007 – Vista vídeos no publicados

ID	RI-008	**Nombre**	Vista detalles de vídeo	
Descripción	Se mostrará en una ventana emergente todos los detalles referentes al vídeo. Además se añade un botón para cerrar la ventana y otro para publicar el contenido.			
Dependencias	RI-007	**Fecha**	22/05/2012	
Prioridad	( ) Alta		() Normal	(*) Baja
Necesidad	() Requerido		(*) Deseable	( ) Opcional

Tabla 3.26: RI-008 – Vista detalles de vídeo

ID	RI-009	**Nombre**	Vista vídeos publicados
Descripción	Se mostrará una vista con todas las presentaciones publicadas en ArcaMM con el campus, título, login de persona que publicó el contenido, fecha de publicación y duración. Además aparecerán los botones de navegación entre las distintas páginas si fuera necesario.		
Dependencias	RI-004,RI-005	**Fecha**	15/03/2012
Prioridad	(*) Alta	() Normal	( ) Baja
Necesidad	(*) Requerido	( ) Deseable	( ) Opcional

Tabla 3.27: RI-009 – Vista vídeos publicados

ID	RI-010	**Nombre**	Vista presentaciones programadas
Descripción	Se mostrará una vista con todas las presentaciones programadas en un futuro y que pueden ser seguidas en directo. Cada vídeo vendrá acompañado del campus, el título, el autor, la fecha y la hora en la que se realizará la grabación y el link donde se podrá seguir en directo. Además aparecerían los botones de navegación entre las distintas páginas si fuera necesario		
Dependencias	RI-004,RI-005	**Fecha**	22/05/2012
Prioridad	( ) Alta	() Normal	(*) Baja
Necesidad	( ) Requerido	( ) Deseable	(*) Opcional

Tabla 3.28: RI-010 – Vista presentaciones programadas

ID	RI-011	**Nombre**	Vista estadísticas: Formulario
Descripción	Dentro de la pestaña estadísticas, primero se debe mostrar un formulario donde poder elegir los criterios para realizar las gráficas. • Dos calendarios para indicar la fecha inicial y final. • Tipo de división temporal (Diario, mensual o anual) • Tipos de datos (Nº de horas o nº de grabaciones) • Campus • Tipo de presentación • Autor Se añade al final un botón para generar las gráficas.		
Dependencias	RI-004,RI-005	**Fecha**	02/04/2012
Prioridad	( ) Alta	(*) Normal	( ) Baja
Necesidad	( ) Requerido	(*) Deseable	( ) Opcional

Tabla 3.29: RI-011 – Vista estadísticas: Formulario

ID	RI-012	**Nombre**	Vista estadísticas: Gráficas
Descripción	Aquí se mostrarán los resultados en dos gráficas, una temporal y otra total. Además se mostrará un resumen numérico indicando el dato para cada uno de los intervalos temporales		
Dependencias	RI-011	**Fecha**	02/04/2012
Prioridad	( ) Alta	(*) Normal	( ) Baja
Necesidad	( ) Requerido	(*) Deseable	( ) Opcional

Tabla 3.30: RI-012 – Vista estadísticas: Gráficas

ID	RI-013	**Nombre**	Vista configuración: Configuración actual	
Descripción	Se mostrará la configuración actual del script de actualización. Aparecerán dos botones una para cargar el backup guardado en la base de datos y otro para modificar los datos			
Dependencias	RI-004,RI-005	**Fecha**	02/04/2012	
Prioridad	( ) Alta	(*) Normal	( ) Baja	
Necesidad	( ) Requerido	(*) Deseable	( ) Opcional	

Tabla 3.31: RI-013 – Vista configuración: Configuración actual

ID	RI-014	**Nombre**	Vista configuración: Cambios de parámetros	
Descripción	Se mostrará el valor actual y un área de texto donde se podrá introducir el nuevo valor. Hay cinco en total, uno para cada posible dato a modificar. Aparecerá un botón para aceptar los cambios que se mostrarán de nuevo para que se puedan revisar antes de confirmar.			
Dependencias	RI-013	**Fecha**	02/04/2012	
Prioridad	( ) Alta	(*) Normal	( ) Baja	
Necesidad	( ) Requerido	(*) Deseable	( ) Opcional	

Tabla 3.32: RI-014 – Pestaña configuración: Cambios de parámetros

ID	RI-015	**Nombre**	Vista de publicación	
Descripción	Se trata de la ventana de publicación perteneciente al rol manager de ArcaMM. Aparecen todos los campos que puede modificar el usuario .Esta ventana no ha sido creada para esta aplicación, sino que es común a todos los contenidos que van a ser publicados en ArcaMM.			
Dependencias	RI-004,RI-005	**Fecha**	07/09/2009	
Prioridad	(*) Alta		( ) Normal	( ) Baja
Necesidad	(*) Requerido		( ) Deseable	( ) Opcional

Tabla 3.33: RI-015 – Vista de publicación

3.3.3 REQUISITOS DE RENDIMIENTO

ID	RR-001	**Nombre**	Tiempo de actualización del script	
Descripción	El script de actualización deberá ejecutarse en un periodo de tiempo que optimice el rendimiento de la aplicación como el del servidor donde se encuentre alojada la aplicación			
Dependencias		**Fecha**	06/02/2012	
Prioridad	(*) Alta		( ) Normal	( ) Baja
Necesidad	(*) Requerido		( ) Deseable	( ) Opcional

Tabla 3.34: RR - 001 – Tiempo de actualización del script

3.3.4 RESTRICCIONES DE DISEÑO

ID	RD-001	**Nombre**	Lenguaje de programación
Descripción	La aplicación gráfica se desarrolla en HTML, PHP, MYSQL y Servidor Apache como servidor WWW		
Dependencias		**Fecha**	06/02/2012
Prioridad	(*) Alta	( ) Normal	( ) Baja
Necesidad	(*) Requerido	( ) Deseable	( ) Opcional

Tabla 3.35: RD - 001 – Lenguaje de programación

ID	RD-002	**Nombre**	Integración en ArcaMM
Descripción	La aplicación gráfica debe de englobarse dentro del Portal de vídeos de la UC3M.		
Dependencias		**Fecha**	06/02/2012
Prioridad	(*) Alta	( ) Normal	( ) Baja
Necesidad	(*) Requerido	( ) Deseable	( ) Opcional

Tabla 3.36: RD - 002 – Integración en ArcaMM

ID	RD-003	**Nombre**	Imágenes de vídeos
Descripción	Se crearán imágenes genéricas, ya que no se pueden capturar fotogramas de los contenidos.		
Dependencias		**Fecha**	22/05/2012
Prioridad	(*) Alta	( ) Normal	( ) Baja
Necesidad	(*) Requerido	( ) Deseable	( ) Opcional

Tabla 3.37: RD - 003 – Imágenes de vídeos

3.3.5 ATRIBUTOS DEL SISTEMA

La aplicación deberá ser fiable, es decir, debe estar programada para tener en cuenta e intentar resolver los posibles errores controlables que aparezcan en la ejecución de la aplicación. Por ejemplo en todas las entradas de texto de la interfaz gráfica se tiene en cuenta si el usuario introdujo palabras con tildes y controlamos que los resultados que ofrecemos al usuario engloban la misma palabra escrita con tilde o sin ella.

La aplicación deberá ser mantenida sin mucha dificultad por el equipo de desarrollo del Portal de vídeos. Para ello se proporciona un código debidamente comentado para que cualquier persona ajena a la realización del código de la aplicación y con nociones de programación, sepa cual es la función de cada trozo de código

Finalmente la aplicación debe estar preparada para responder a ataques. En principio nosotros sólo tenemos que tener en cuenta de que el acceso a la aplicación debe realizarse a través de login y contraseña de correo de la UC3M. Además debemos de comprobar en cada fichero php que el usuario autentificado tiene el permiso del administrador para poder trabajar en el rol *Mediasite.* Al no tratar más aspectos de seguridad en este proyecto, se presupone que el servidor donde quedará alojada nuestra aplicación, integra de forma correcta una capa de seguridad al igual que lo hace con el Portal de vídeos y resto de aplicaciones que lo conforman.

CAPÍTULO 4 DISEÑO DE LA APLICACIÓN

En este capítulo nos dedicaremos a explicar los pasos que hemos seguido para realizar el diseño de la aplicación de la que trata este Proyecto Fin de Carrera.

El diseño de la aplicación la dividiremos en dos bloques para facilitar su compresión. Una primera parte en la que abordaremos la extracción de datos de la base de datos de *Mediasite* a través de la documentación que nos proporciona el distribuidor de la herramienta y un segundo bloque en el que explicaremos la integración, de toda la información extraída, en el portal de vídeos de la Universidad Carlos III, ArcaMM, donde se presentarán los distintos vídeos que podemos publicar en dicho portal.

4.1 EXTRACCIÓN DE INFORMACIÓN DE MEDIASITE

Como hemos mencionado en la introducción de este capítulo, vamos a explicar cómo vamos a extraer toda información que nos sea posible de la base de datos propia de *Mediasite* para poder utilizarla posteriormente en la publicación de los vídeos en ArcaMM.

Cuando se realiza una grabación de un evento o clase con el sistema *Mediasite*, esta se sube automáticamente a la base de datos del sistema, siempre y cuando se haya creado, por uno de los técnicos del Área de Audiovisuales, un punto de publicación asociado a esa presentación o bien sea una presentación programada. En la siguiente figura podemos observar cual es la arquitectura actual en la Universidad Carlos III de Madrid.

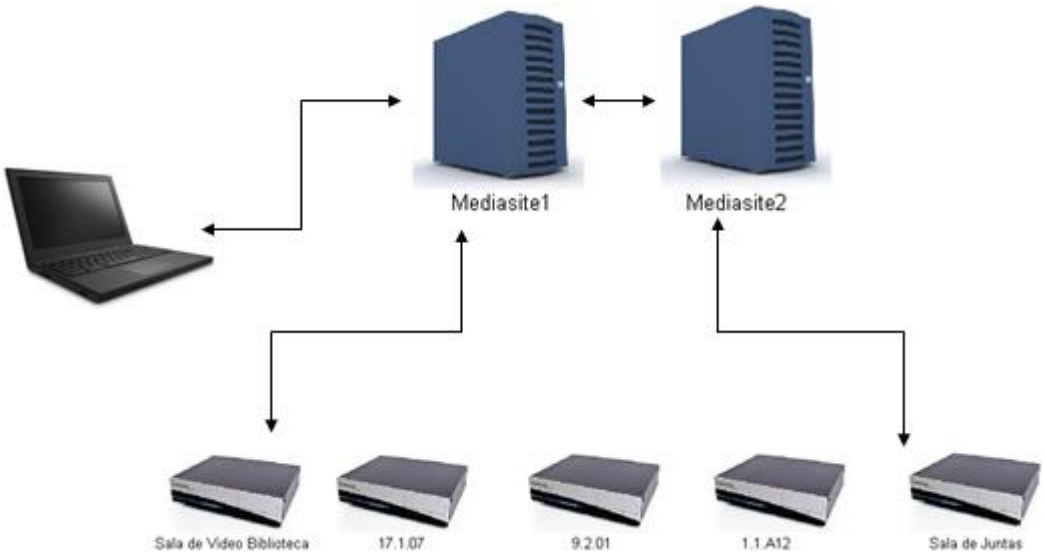


Figura 4.1: Arquitectura de Mediasite en la UC3M

Podemos diferenciar los cinco *recorders* que están distribuidos por los distintos campus de la universidad. Estos son los encargados de realizar las grabaciones para después asociar dicha presentación a una determinada publicación y publicarla en el servidor. La comunicación entre los grabadores y los servidores es bidireccional, ya que los equipos de grabación pueden acceder a los servidores como un cliente más. Esta comunicación se hace de manera diferente dependiendo del servidor con el que se comunique, ya que cada servidor tiene una función concreta.

El servidor denominado *Mediasite1* es el encargado de la gestión de *Mediasite*, donde se aloja el servidor web o interfaz gráfica del sistema y las capturas de pantalla de de las presentaciones asociadas a la grabación. Con estas imágenes el sistema va a poder crear puntos de navegación temporal coincidiendo con cada uno de los cambios que se producen en las transparencias. La comunicación entre el equipo de grabación y este servidor se realiza a través de FTP, por el puerto 2042 y el 2059. Además este servidor es el que se comunica con todos los clientes a través del protocolo HTTP.

El servidor denominado *Mesiasite2* es el encargado del almacenamiento de los vídeos y llevar a cabo el streaming de todas las presentaciones que son grabadas con los *recorders*. *Mediasite* da la posibilidad a los gestores del sitio de crear presentaciones programadas antes del día del evento, con lo que el sistema genera automáticamente una url en la que el acto se podrá seguir en directo. En la finalización de la clase esta url se reutilizará y será la misma para seguir la grabación bajo demanda. Este servidor se comunica con los recorders a través de HTTP, por el puerto 8080 y por FTP, por el puerto 1947, a la finalización del vídeo para comprobar que la transferencia se ha realizado perfectamente.

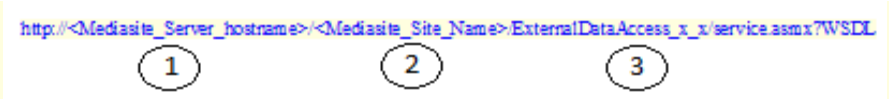
Ambos servidores están en continua comunicación, ya que como hemos indicado antes, los clientes se comunican con el servidor *Mediasite1* pero este a su vez recoge el vídeo del servidor *Mediasite2* para que el cliente pueda visualizarlo. Estas comunicaciones se realizan a través del protocolo HTTPS.

Una vez que conocemos la arquitectura y las funciones de cada uno de los agentes implicados, nos damos cuenta que el servidor que realmente nos interesa para el desarrollo de nuestra aplicación es el *Mediasite1* que es el responsable de la gestión del sitio y el que nos proporcionara todos los enlaces a las presentaciones con sus correspondiente metadatos. Para tener acceso a todos estos metadatos, como a los propios vídeos, el sistema nos proporciona un *External Data Access Service* (a partir de aquí EDAS), que es una documentación donde se facilitan los distintos métodos, la estructura de los *request* y *response* etc. Para el seguimiento correcto de este recomienda mirar el Web Service de dicho EADS que se proporciona en el Apéndice B.

Después de las primeras reuniones que tuvimos con el equipo del Área de Audiovisuales sobre los requisitos y diseño que debía tener la aplicación, decidimos que lo primero que necesitábamos era saber hasta qué punto nos permitía *Mediasite* la extracción de datos y de que relevancia eran estos. Sin esta información creímos conveniente no definir todo el plan de trabajo ya que lo único que podíamos conseguir era dar algún paso en falso que retrasaría el desarrollo de la aplicación

Con estas primeras premisas y después de ojear las posibilidades que nos da el EDAS, lo primero que necesitamos es la url donde se encuentra la WDSL, *Web Services Description Language*, de la correspondiente versión que vayamos a utilizar, ya que con ella crearemos el cliente *soap* con el que podremos llamar a las funciones que están listadas en el WDSL.

La forma que debe de tener la url es la siguiente:



1. Host donde está alojado el sistema Mediasite
2. Nombre del sitio
3. Versión del EADS

Figura 4.2: Formato de url de acceso a EDAS

Tanto esta url, como el nombre de usuario y la contraseña del sistema están almacenados en la base de datos de ArcaMM, accediendo a ella cada vez que se realiza una actualización de los vídeos y metadatos, con lo que conseguimos que si se cambia de versión del WDSL, el host o la contraseña, el sistema siga siendo estable. En la interfaz de usuario que crearemos posteriormente para la publicación de vídeos, existirá una pestaña donde estos parámetros serán modificables si esto fuera necesario y volviéndolos a almacenar de nuevo en la base de datos de nuestro sistema. Esto fue de gran ayuda en la realización de la aplicación, ya que cuando se estaba realizando la segunda parte de este proyecto, que es la interfaz e integración en ArcaMM, se instaló una nueva versión del EADS (la 5.0), con lo que modificando la url en la base de datos, todo lo realizado anteriormente funcionaba de forma correcta. Únicamente se añadieron nuevos métodos o se modificó algún otro que más adelante explicaremos con más detenimiento.

Una vez que tenemos creado el cliente *soap* con el que atacaremos a todos los métodos incluidos en el WDSL, necesitamos acceder a la base de datos de *Mediasite*, con el usuario y la contraseña del administrador del sistema que anteriormente hemos mencionado que tenemos almacenados en la base de datos de ArcaMM. Con estos tres datos ya podemos hacer uso del método *Login*, el cual si la autentificación se ha realizado de forma correcta, nos devuelve un ticket de sesión necesario para acceder a todos los métodos del Web Service.

```
$datos = array(
      "Username"=> $user,
      "Password"=> $pass,
      "ImpersonationUsername"=>$user);

$soapclient = new SoapClient($url);

$version = $soapclient->Login ($datos);
$ticket = $version->UserTicket;
```

Figura 4.3: Login

Si esta autentificación no se realiza de forma correcta, se envía un error ya que el ticket está vacío y no podrá acceder a los vídeos y metadatos, con lo que debemos de revisar si la contraseña de administrador o algún otro parámetro han sido modificados.

A partir de aquí se comenzaron a hacer pruebas con cada uno de los métodos que nos proporciona el EDAS, aunque ya teníamos presente que nuestra intención era solo utilizar los métodos de lectura ya que el propósito de esta aplicación no es el de crear, modificar o borrar presentaciones del sistema *Mediasite*, pero deja una puerta abierta a posibles mejoras o líneas futuras que posteriormente abordaremos.

Este proceso de pruebas, que duraría alrededor de una semana, abría un abanico de posibles soluciones que necesitaban ser contempladas por el Área de Audiovisuales, ya que ellos eran los usuarios finales que iban a hacer uso de la aplicación y de la cual se aprovecharía la comunidad universitaria. Por consiguiente, se llevó a cabo una segunda reunión con ellos en donde quedarán claros los pasos a seguir. Las soluciones que se pusieron sobre la mesa fueron las siguientes:

- La primera solución que se propuso fue la de leer directamente de la base de datos de *Mediasite* sin ningún tipo de intermediario con lo que si la información relativa a las presentaciones cambiaba esta lo hacía en el navegador cuando la página de visualización de vídeos fuera refrescada. A pesar de que esta solución posiblemente fuera bastante más rápida que cualquier otra, no nos permitía guardar toda la información de las presentaciones con lo que nos acarrearía problemas para poder realizar búsquedas entre las distintas presentaciones que quisiéramos publicar o generar estadísticas de los vídeos creados con *Mediasite* que han sido publicados en ArcaMM, con lo que decidimos declinar esta solución y abordar la segunda de las soluciones.

- La segunda y ultima de las soluciones fue la de extraer la información en intervalos de tiempo y almacenarlo en una base de datos propia que nos permitía resolver los problemas que nos generaba la primera solución. A cambio nos aparecieron otros problemas, pero que pensamos que tenían una resolución más sencilla que los que se nos planteaban en la primera solución. Uno de ellos era el de realizar actualizaciones en intervalos de tiempo, lo que convertía a nuestra aplicación en asíncrona, es decir si se actualizaba la base de datos de *Mediasite*, nuestra base de datos no se actualizaría hasta que no hubiera pasado el tiempo que finalmente decidiéramos. Este tiempo no debía de ser muy pequeño ya que saturaríamos el servidor donde de alojara la aplicación, que es el mismo donde se aloja ArcaMM, y tampoco demasiado grande para que el personal de Audiovisuales no tuviera que esperar para poder publicar el vídeo.

Finalmente y tras evaluar las dos soluciones decidimos quedarnos con la segunda porque consideramos que aunque sea algo más compleja es bastante más segura y los problemas que nos crea son más fáciles de solucionar que los que nos crea la primera solución. Para llevarla a cabo necesitaremos crear una base de datos en local que posteriormente añadiremos a la base de datos principal de ArcaMM.

4.1.1 DISEÑO DE LA BASE DE DATOS

La premisa principal para crear la base de datos es la de hacerla semejante a la que pudiera existir en el propio sistema *Mediasite.* Para ello nos ayudamos de todas las pruebas que hemos hecho durante la semana en las cuales podíamos hacernos una idea de la arquitectura de dicha base de datos. Observamos que los elementos importantes de la base de datos, presentaciones, presentadores, carpetas, visores y perfiles de grabación están identificados con un ID único, el cual nos ayudará para relacionar todos los elementos entre sí.

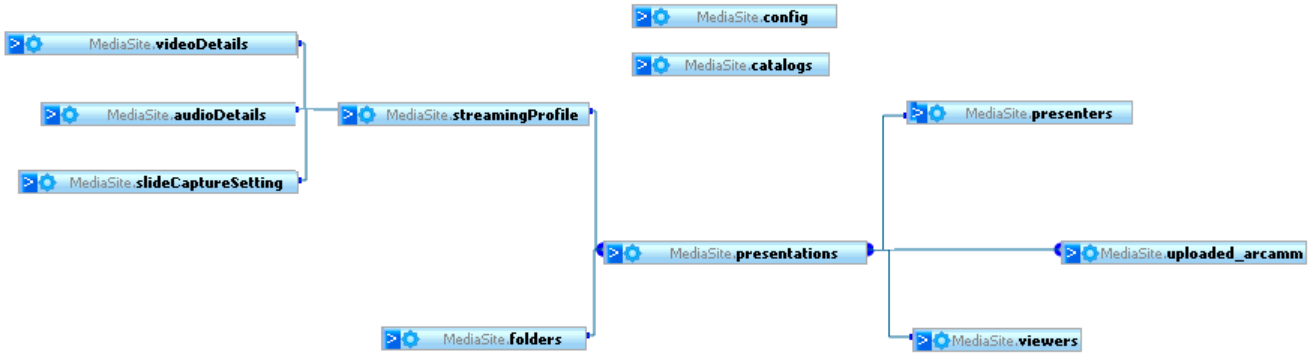


Figura 4.4: Diseño de la base de datos

La relación de las distintas tablas que vamos a crear deberemos hacerla a través PRIMARY KEYS asignadas al campo id de cada tabla, con lo que mantendrán una integridad referencial entre ellas facilitando las consultas, actualizaciones o borrado de la base de datos.

Partiendo de esta base, vamos a crear en primer lugar cinco tablas, una por cada uno de los elementos mencionados anteriormente. A continuación detallamos cada una de ellas con sus respectivos campos:

- Tabla *presentations*

En esta tabla se guarda todos los metadatos asociados a cada una de los vídeos que se encuentran en la base de datos de *Mediasite.*

Campo	Tipo	Descripción
id	varchar(36)	Id único asociado al vídeo
name	varchar(255)	Título
airDateTime	datetime	Fecha y hora de grabación
duration	time	Duración
url	varchar(255)	Url para visualizar el vídeo
slideCount	smallint(5)	Cantidad de cambios que hay en la presentación de diapositivas.
chapterCount	smallint(5)	Número de capitulo
status	varchar(32)	Estado del vídeo. (Scheduled, OpenForRecord, Recording, Recorded, Uploaded)
isLive	tinyint(1)	Si el vídeo se emitió en directo

isOnDemand	tinyint(1)	Si el vídeo está disponible bajo demanda
presenters	varchar(255)	Ids de los autores del vídeo
name_presenters	varchar(1024)	Nombre completo de cada uno de los autores
streamingProfile	varchar(36)	Id del perfil de grabación utilizado
viewer	varchar(36)	Id del visor para visualizar la presentación
urlPresentation	varchar(255)	Url de la presentación VGA
urlVideo	varchar(255)	Url de la parte del vídeo obtenida con la cámara
timeZone	varchar(4)	Zona horaria donde se realizó la grabación
supportingLinks	varchar(512)	Links asociados a la presentación
folder	varchar(36)	Id de la carpeta donde se almacena el vídeo
campus	varchar(64)	Campus donde se realizó la grabación
description	varchar(1024)	Descripción del contenido de la grabación
owner	varchar(128)	Persona de Audiovisuales que realizó la grabación
IsDB	tinyint(1)	Si el vídeo continua en la base de datos de Mediasite

Tabla 4.1: Presentations

- Tabla *presenters*

En esta tabla se guarda toda la información referida a cada uno de los autores que han realizado grabaciones con el sistema *Mediasite.*

Campo	Tipo	Descripción
id	varchar(36)	Id único asociado al autor de la grabación
prefix	varchar(255)	Prefijo del nombre completo(D,Dña, Sr,Sra,Dr)
firstName	varchar(255)	Nombre del autor
middleName	varchar(255)	Primer Apellido
lastName	varchar(255)	Segundo Apellido
nameComplete	varchar(256)	Nombre completo del autor
imageUrl	varchar(255)	Url de la imagen asociada al autor
email	varchar(255)	Email de contacto
bioUrl	varchar(128)	Url a una página asociada al autor
owner	varchar(255)	Persona de Audiovisuales que añadió el autor
additionalInfo	varchar(1024)	Información adicional del autor
isDB	tinyint(1)	Si el autor continua en la base de datos de Mediasite

Tabla 4.2: Presenters

- Tabla *folders*

En esta tabla almacenaremos toda la información asociada a todas las carpetas generadas en la base de datos de *Mediasite que* contienen en su interior alguna grabación. En principio esta información no es necesaria porque en la catalogación el árbol de carpetas no tendrá nada que ver con la manera en la que estaban almacenadas en *Mediasite* pero si nos puede ayudar a realizar dicha catalogación. Normalmente, el personal del Área de Audiovisuales guarda la presentación siguiendo un mismo patrón:

1. Selecciona el campus donde se realizo la grabación.
2. Selecciona le curso académico.
3. Finalmente guarda el vídeo en una carpeta representativa de la asignatura, congreso, jornada, etc. Puede que exista alguna carpeta intermedia para grados que son grabados en su totalidad, y que luego se desglosan en sus distintas asignaturas.

Aquí mostramos un ejemplo de cada campus de la ruta final que sería generada de este proceso.

CampusColmenarejo/curso 2010 2011/DBC. Curso1011

CampusLeganes/Curso 2010 2011/Grado en Ingeniería Informática/Curso 1

CampusGetafe/Curso 20082009/Grado en Relaciones Laborales y Empleo

Figura 4.5: Ejemplos de rutas de carpetas

Con esta forma de catalogar conseguimos catalogar las grabaciones de forma ordenada y ha sido de gran ayuda para poder clasificar las presentaciones por campus en la base de datos y por consiguiente en la posterior catalogación que se realizará en ArcaMM.

La tabla en la base de datos tendría los siguientes campos:

Campo	Tipo	Descripción
id	varchar(36)	Id único asociado a la carpeta que guarda el vídeo
name	varchar(128)	Nombre de la carpeta
description	varchar(255)	Descripción de la carpeta
parentId	varchar(36)	Id único de la carpeta padre
hasChildFolders	tinyint(1)	Si se tienen carpetas hijas
type	varchar(36)	Tipo de carpeta
folderPath	varchar(1024)	Ruta completa en el sistema *Mediasite*
campus	varchar(64)	Campus en el que se creó la carpeta
owner	varchar(36)	Persona de Audiovisuales que creó la carpeta
isDB	tinyint(1)	Si la carpeta continua en la base de datos de Mediasite

Tabla 4.3: Folders

- Tabla *viewers*

En esta tabla se almacenan los distintos visores que existen para poder visualizar las presentaciones. Cada presentación tendrá un visor predefinido en el momento de realizar la grabación.

Campo	Tipo	Descripción
id	varchar(36)	Id único asociado al visor del contenido
name	varchar(128)	Nombre del visor
isDB	tinyint(1)	Si el visor continua en la base de datos de Mediasite

Tabla 4.4: Viewers

- Tabla *streamingProfile*

En esta tabla almacenamos los metadatos asociados a los distintos perfiles de grabación con los que el personal técnico puede realizar las capturas y que por lo tanto están a disposición del usuario.

En un primer vistazo se pensó realizar una única tabla donde se aglutinaran todos los datos de la configuración del perfil de grabación, pero se optó por la opción de hacer una tabla general donde aparecía el id, el nombre, la descripción y la referencia a otra tabla donde se almacenaran los detalles de vídeo, de audio y de la captura de pantalla. El motivo de hacerlo así es que existen perfiles que tienen más de una entrada de vídeo y por tanto de audio, con lo que cada señal tiene su configuración independiente. Si se hubiera guardado todo en la misma tabla hubieran existido problemas en la configuración en la tabla ya que esta no sería homogénea para cada uno de los perfiles.

A continuación mostramos la forma que tiene la tabla general de perfiles de grabación.

Campo	Tipo	Descripción
id	varchar(36)	Id único asociado al perfil de grabación
name	varchar(128)	Nombre del perfil
description	varchar(255)	Descripción de las características del perfil
videoEncoding	varchar(32)	Referencia única a la tabla vídeoDetails
audioEncoding	varchar(32)	Referencia única a la tabla audioDetails
slideCaptureSettings	varchar(32)	Referencia única a la tabla slideCaptureSettings
owner	varchar(128)	Persona que creó el perfil
isDB	tinyint(1)	Si el visor continua en la base de datos de Mediasite

Tabla 4.5: StreamingProfile

Como podemos ver existen tres campos que guardan una referencia única a otra tabla donde se almacenan las características del perfil tanto de vídeo, audio como de la captura de pantalla. En las siguientes tablas presentamos cada una de estas tres tablas con sus correspondientes campos.

Campo	Tipo	Descripción
ref	smallint(5)	Referencia única de los detalles de vídeo
idProfile	varchar(36)	Id único del perfil al que pertenecen los detalles de vídeo
name	varchar(64)	Nombre del códec de vídeo utilizado
description	varchar(128)	Descripción de la configuración de audio
height	smallint(5)	Alto de la señal de vídeo
width	smallint(5)	Ancho de la señal de vídeo
effectiveBitRate	mediumint(8)	Tasa de bits del flujo de vídeo
frameRate	tinyint(3)	Tasa de fotogramas por segundo
buffersize	smallint(5)	Tamaño del buffer
imageQuality	tinyint(3)	Calidad de imagen
KeyFrame	smallint(5)	Distancia entre fotogramas clave

Tabla 4.6: VideoEncoding

Campo	Tipo	Descripción
ref	smallint(5)	Referencia única de los detalles de audio
idProfile	varchar(36)	Id único del perfil al que pertenecen los detalles de vídeo
name	varchar(128)	Nombre del códec de audio utilizado
description	varchar(255)	Descripción de la configuración de audio
bitRate	smallint(5)	Tasa de bits del flujo de audio
sampleRate	smallint(5)	Tasa de muestreo de audio
channels	tinyint(3)	Canales de audio. 1 Mono, 2 Stereo
encodingMode	varchar(128)	Modo de codificación

Tabla 4.7: AudioEncoding

Campo	Tipo	Descripción
ref	smallint(6)	Referencia única de los detalles de captura de pantalla
idProfile	varchar(36)	Id único del perfil al que pertenecen los detalles de captura de pantalla
description	varchar(1024)	Descripción de la configuración de captura de pantalla
changeSensitivy	smallint(6)	Sensibilidad a cambios
maxScanRate	smallint(6)	Tasa máxima de escaneo.
quality	smallint(6)	Calidad de la captura de pantalla
options	varchar(32)	Opciones

Tabla 4.8: SlideCaptureSettings

Con esto presentamos ocho de las once tablas que forman parte de la base de datos que hemos creado para guardar todos los metadatos e información relacionados con las presentaciones almacenadas en la base de datos de *Mediasite.*

De las tres tablas restantes solo existe una que tiene relación con las ocho mostradas anteriormente y por lo tanto tienen integridad referencial dentro de la base de datos, mientras que las otras dos no tienen ningún tipo de relación con el resto de tablas.

- Tabla *config*

Una de estas tablas es la tabla de configuración del sistema, donde almacenamos los datos referentes al sitio de Mediasite. Esta es la primera información que lee el script que actualiza los vídeos ya que de ella extrae el nombre de usuario, la contraseña, la ruta donde están almacenadas las presentaciones y su correspondiente información. Aunque entraremos en detalle cuando expliquemos la integración en ArcaMM, el motivo principal por el que realizamos esta tabla es que siempre existe la posibilidad de que el servidor de *Mediasite* pueda cambiar de ubicación, o bien el usuario y la contraseña pueden ser modificados por temas de seguridad. Con ayuda de la interfaz que crearemos para la integración con ArcaMM, podremos modificar cualquiera de estos parámetros y así hacer que nuestro sistema sea robusto en caso de cambios, ya que solo modificando cualquiera de estos campos el sistema funcionará y seguirá actualizando los vídeos y su información.

Campo	Tipo	Descripción
user	varchar(24)	Usuario de *Mediasite*
password	varchar(24)	Contraseña de acceso a *Mediasite*
uri	varchar(128)	Urn de la ruta donde encontramos el EDAS
url	varchar(128)	Ruta donde está cargado el EDAS de la versión utilizada por el sistema *Mediasite*
url_imagen	varchar(1024)	Url a la imagen predefinida para todos los vídeos catalogados en ArcaMM
userChange	varchar(128)	Usuario registrado en LDAP que realizó la última modificación
dateChange	datetime	Fecha del cambio de la configuración
backup	tinyint(4)	Indica que está disponible una configuración alternativa en el caso de que falle la configuración en uso
use	tinyint(4)	Indica si esta configuración está en uso

Tabla 4.9: Config

Cabe reseñar que en un principio esta tabla sólo tenía los cuatro primeros campos, pero según se fue desarrollando el proyecto, en especial la parte de integración en ArcaMM, se fueron añadiendo campos que eran necesarios para un correcto funcionamiento de la aplicación y que no se habían sopesado al inicio de la realización de esta y que justificaremos cada uno de ellos en la posterior fase del proyecto.

- Tabla *catalogs*

La otra tabla que no tiene relación con el resto de tablas, es la tabla de catálogos en la que almacenamos el nombre y url de los catálogos que se han creado desde la interfaz web del sistema *Mediasite*. Un catálogo es un conjunto de presentaciones que el personal de audiovisuales clasifica según un criterio específico.

Está función se agregó al sistema *Mediasite* mediante la actualización a la versión 5.0. Como esta actualización se llevó a cabo en plena realización de la interfaz de usuario y por tanto el script de actualización estaba acabado, el equipo de trabajo se reunió para valorar si merecía la pena y que coste temporal llevaba amoldar el script de actualización de vídeos. Se observó que el EDAS de la versión 5.0 respetaba los métodos de la versión 4.0 con lo que el trabajo realizado hasta ahora iba a funcionar correctamente con esta nueva versión. Las únicas modificaciones que podían mejorar nuestra aplicación, era la inclusión de métodos para poder manejar los catálogos, con lo que se decidió añadir el código necesario al script para poder obtener esta información, ya que la clasificación que existiera dentro de *Mediasite* la podríamos exportar a la catalogación de ArcaMM.

En un primer momento parecía que esta nueva función nos sería de gran ayuda para mejorar nuestro script, pero cuando comenzamos las primeras pruebas nos dimos cuenta que el sistema *Mediasite* solo nos proporcionaba la url al catalogo y obviaba cualquier tipo de relación del catálogo con las presentaciones que se encontraban en su interior, con lo que no podíamos proporcionarle al personal de Audiovisuales esa información a la hora de catalogar en ArcaMM.

Toda la información que se le proporciona al personal de audiovisuales es exactamente con la que se guardó el vídeo en el momento de su grabación, pero a la hora de catalogar en ArcaMM esta información es editable para evitar posibles errores que se produjeron al crear la presentación o modificaciones que se deseen realizar, con lo que si los catálogos en *Mediasite* hubieran funcionado como se esperaba, hubiera ayudado al catalogador a clasificar los vídeos en ArcaMM en una misma serie, análogo a los catálogos. A pesar de esto, como el trabajo estaba realizado, se decidió almacenarlo en la base de datos propia para alguna posible implementación futura, aunque para el desarrollo de este proyecto no fuera necesario.

Campo	Tipo	Descripción
id	varchar(36)	Id único del catalogo de Mediasite
name	varchar(128)	Nombre del catálogo
description	varchar(512)	Descripción del contenido del catalogo
url	varchar(1024)	Url para visualizar el catálogo
isDB	tinyint(1)	Indica si se encuentra en la base de datos

Tabla 4.10: Catalogs

- Tabla uploaded_arcamm

Por último nos queda por mostrar quizá la tabla más importante de la base de datos creada para la resolución del problema que nos ha llevado a realizar este proyecto. Esta tabla, que llamamos *uploaded_arcamm*, es la encargada de guardar la información de los presentaciones que han sido publicadas en portal de vídeos de la UC3M y por lo tanto relaciona las presentaciones de *Mediasite* con su publicación en ArcaMM.

Desde las primeras reuniones, sabíamos que la existencia de esta tabla era obligatoria por la razón mencionada anteriormente, aunque no se tenía claro cuál iba a ser su estructura final porque dependía mucho de la forma de resolver la segunda parte de este proyecto, la integración con ArcaMM. Por lo tanto de la estructura final, que

vamos a presentar a continuación, solo existían los campos *ref* , *idPresentation* y *Uploaded,* con lo que se creaba una tabla que no tenía mucha funcionalidad a priori ya que sólo relacionábamos esta tabla con la tabla *presentations* a través de la id de la presentación de *Mediasite.*

Aunque ahora mostramos la estructura final de la tabla no haremos mucho hincapié en ella porque explicaremos las razones que nos llevaron a agregar más campos en la segunda fase de la implementación.

Campo	Tipo	Descripción
ref	int(10)	Referencia de la presentación guardada
idPresentation	varchar(36)	Id único de la presentación en *Mediasite*
uploaded	tinyint(1)	Si la presentación está publicada en ArcaMM
url_arcamm	varchar(128)	Url a la catalogación en ArcaMM
date_uploaded	datetime	Fecha de publicación en ArcaMM
published_by	varchar(64)	Usuario LDAP que publica el vídeo

Tabla 4.11: Uploaded_arcamm

Una vez que tenemos claro cuál es el diseño final de nuestra base de datos, procedemos a crearla en el ordenador local donde llevaremos a cabo toda la preproducción del código y donde probaremos que cumplimos las funcionalidades y requisitos previstos que debe cumplir el proyecto.

Para crear la base de datos nos ayudamos de la herramienta *phpmyadmin* que a través de su interfaz gráfica nos permite manejar la administración de MYSQL de una manera sencilla sin escribir ninguna línea de código. A pesar de ello, tenemos muy presente el manual de referencia de MYSQL 5.0 que podemos encontrar en la página oficial de esta base de datos de código abierto, aunque nos será más útil en el momento que tengamos que hacer inserciones, consultas, modificaciones o borrados que es el siguiente paso en el proceso de realización del script de actualización

4.1.2 ALMACENAMIENTO DE INFORMACIÓN EN LA BASE DE DATOS

Una vez creada la base de datos ya tenemos el lugar donde guardar toda la información posible del sistema *Mediasite* con ayuda del *EDAS* que nos proporciona dicha distribución. Para mejorar la compresión y síntesis de esta parte de la memoria vamos a obviar muchas partes de código que pueden ser repetitivas y explicar el porqué de las decisiones tomadas.

El código de actualización está disgregado en varios ficheros *php* basándonos en uno de los primeros consejos que te dan cuando das los primeros pasos en la programación, "Divide y Vencerás". Esta filosofía, que también da un nombre a un algoritmo, nos invita a dividir nuestro problema en problemas más pequeños, que resolviéndolos, nos ayudará a resolver nuestro problema de una manera más sencilla. Esto aplicado concretamente a nuestro código nos ha ayudado a tener un código mucho más compresible y ordenado, que facilita la creación de este, como su posterior modificación en caso de errores o mantenimiento por personas ajenas a este proyecto que puedan formar parte de él en algún momento.

Aprovechando esta división vamos a proceder a explicar cada una de las partes del script por separado, que se corresponderán con cada uno de los ficheros php creados. En todo momento de la realización de este proceso tenemos como referencia el manual de MYSQL y de PHP, sobretodo el aparatado dedicado a las funciones dedicadas a hacer operaciones con la bese de datos.

- Configuración y conexión a la base de datos propia

Estos dos scripts son los encargados de conectarse a la base de datos propia y configurarla de forma correcta.

Primero *config.php* se conecta a la base de datos facilitando el host el usuario y la contraseña.

```
$link = mysql_connect("localhost", "root","*********");
```

Figura 4.6: Conexión con base de datos

Después *selectDB.php* elige la base de datos de entre las que existen en el sistema, que un primer momento solo existirá nuestra base de datos, pero en el momento que se añada a la de ArcaMM convivirá con el resto de bases de datos que forman parte de la plataforma de vídeos de la UC3M.

```
include ("config.php");
mysql_select_db("MediaSite",$link);
```

Figura 4.7: Elección de base de datos

- Obtención del ticket de conexión

Como para diseñar la base de datos necesitábamos conectarnos para saber qué información podíamos recabar de la base de datos de *Mediasite*, el código es básicamente el que mostrábamos en la figura 4.3, aunque como hemos creado una tabla de configuración, leemos los datos que necesitamos para crear el ticket que son `$user, $pass y $url`. La consulta que realizamos en la base de datos es la siguiente:

```
$result = mysql_query("SELECT * FROM config WHERE
          config.use=1",$link);
```

Figura 4.8: Obtención del ticket de conexión

Cuando ya tenemos el ticket, podemos obtener todos los datos referentes a las presentaciones. La forma de obtener la información tiene que ver con el diseño que hemos hecho de la base de datos, es decir como todo está relacionado con la tabla *presentations*, primero debemos obtener la información referente a los presentadores, carpetas, visores y perfil de grabación ya que dentro de la tabla *presentations* hay campos que hacen referencia a estos y no pueden ser nulos. Por el contrario el orden de obtener la información entre el resto de tablas no es importante, ya que no están relacionadas entre sí, pero es obligatoria que la última información que introduzcamos sea la de las presentaciones.

- Obtención de todos los presentadores

En primer lugar vamos a obtener todos los presentadores presentes en la base de datos. Para ello comprobamos si existen presentadores en ella con la siguiente instrucción:

```
$presenters=$soapclient->QueryPresenters($getMessage);
if(isset($presenters->Presenters->PresenterContext)){
```

Figura 4.9: Comprobación de existencia de presentadores

Si no existen presentadores el script se acaba y se busca la siguiente información.

Una vez que comprobamos que en la base de datos existen presentadores, para obtener los detalles de cada uno de ellos tenemos que hacer uso de otra función distinta a la que utilizamos anteriormente. A la nueva función, *QueryPresenterDetail*, hay que pasarle un array en el que vaya incluido el ticket, el usuario y un array de ids de los presentadores presentes en la base de datos. Esta array lo obtenemos recorriendo todos los presentadores y almacenando su id. El motivo de tener que hacer uso de dos funciones distintas, es que *QueryPresenters* sólo nos proporciona el nombre del presentador y su id cuando necesitamos todos los detalles y sólo podemos obtenerlos con *QueryPresenterDetail* proporcionando al método un array de ids que anteriormente hemos generado con la respuesta de *QueryPresenters.*

```
//crear lista de presentadores
    for($i=0;$i<count($presenters->Presenters->PresenterContext);$i ++){
            $presentersIdList[$i]=$presenters->Presenters->
              PresenterContext[$i]->Id;
    }

    $getPresenterD = array(
            "UserTicket"=>$ticket,
            "ImpersonationUsername"=>$user,
            "PresenterIdList"=>$presentersIdList);

    $presentersD=$soapclient->QueryPresenterDetails($getPresenterD);
```

Figura 4.10: Array de presentadores

Una vez que tenemos el array `$presentersD`, para obtener la información de cada uno de los presentadores tenemos que recorrer el array resultante. Antes de recorrer el array, debemos realizar una modificación en la tabla *presenters* y es la de poner el campo *isDB* a 0 de todos los autores presentes en la base de datos. Pare ello utilizamos esta *query* de MYSQL:

"UPDATE presentations set isDB='0';"

Después recorremos el array `$presentersD` pasando como parámetro el id del autor y en cada una de las iteraciones del bucle realizaremos el siguiente proceso:

1. Guardamos en variables cada uno de los datos que nos proporciona el sistema *Mediasite.* Para el nombre del autor, no nos parecía muy útil que nos lo ofreciera divido en prefijo, nombre y apellidos, con lo que finalmente uníamos todo en una variable que llamamos `$nameComplete`. Podemos ver como guardamos el valor de un determinado atributo en una variable cualquiera:

```
$presenter = $presentersDetails->Details->PresenterDetails[$i];
if(isset($presenter->Email)){
            $email = $presenter->Email;
        }else{
            $email = null;}
```

Figura 4.11: Guardado de valores

2. Realizamos una consulta a la base de datos propia para comprobar si existe ese presentador en la base de datos.

   *"SELECT * FROM presenters WHERE id ='".$idPresenter."';"*

   Si el autor ya está presente en la base de datos propia, comparamos esos valores con los que acabamos de guardar en variables de la base de datos de *Mediasite.* Además nos aseguramos que cualquier cambio que se produzca en la base de datos de *Mediasite* quede reflejado en nuestra aplicación. Con esto aumentamos el tiempo de computación pero a cambio tenemos un sistema más robusto, en este caso a posibles modificaciones. Además aunque no se haya modificado ningún campo, siempre modificaremos un campo, *isDB*, con lo que indicamos que el autor sigue presente en la base de datos de *Mediasite.*

```
$sql = "UPDATE presenters SET isDB='1'";
$u=0;
if($row["nameComplete"]!=$nameComplete){
      $sql=$sql. ", nameComplete='".$nameComplete."'";
      $u=1;}
$sql = $sql." WHERE id='".$idPresenter."'");
```

Figura 4.12: Comprobación de cambios

Si el autor no está presente en la base de datos insertamos el nuevo presentador con sus correspondientes datos.

"INSERT INTO presenters (id, prefix, firstName, middleName..,..,) VALUES('".$idPresenter."','".$prefix."','".$firstName."','".$middleName."',..)"

3. Cuando hemos comprobado que todos los autores de la base de datos de Mediasite están presentes en la base de datos propia, solo nos quedaría borrar los presentadores que ya no estuvieran presentes en el sistema Mediasite. Como cada vez que hemos hecho un *update* o un *insert* hemos dado al campo *IsDB* el valor de 1, los autores que debemos de borrar son los que tienen dicho campo con valor 0.

   "DELETE FROM presenters WHERE IsDB='0'"

- Obtención de todas las carpetas con presentaciones

Después de obtener los autores, el siguiente paso que se ejecuta en el script de actualización es el de guardar las carpetas que contienen presentaciones en la base de datos. El proceso es muy similar al del proceso de presentadores, con lo que omitiremos parte de código similar con en el de la fase anterior.

En primer lugar comprobamos que existen carpetas con presentaciones en la base de datos de *Mediasite.*

```
$foldersWithPresentations=$soapclient->QueryFolders
WithPresentations($getMessage);
if(isset($foldersWithPresentations->Folders->FolderDetails)){
```

Figura 4.13: Comprobación de existencia de carpetas

En el caso de que no existan carpetas, el script pasa a la siguiente fase.

A diferencia con el caso anterior, los presentadores, el método *QueryFoldersWithPresentatios* nos genera un array de carpetas que es el que recorreremos siguiendo el mismo proceso que en el caso de los presentadores. Por lo tanto, debemos de poner a 0 el campo IsDB de la tabla *folders* de todas las carpetas que tengamos añadidas en la base de datos propia. Posteriormente en cada una de las iteraciones del bucle realizamos el siguiente proceso:

1. Guardamos en variables cada uno de los datos suscritos a la carpeta consultada.

2. Realizamos una consulta a la base de datos propia para comprobar si ya está presente en ella. En el caso afirmativo comprobamos si alguno de los campos ha cambiado, añadiendo a la consulta de actualización los campos y valores que así lo hayan hecho. Además como mínimo habrá un campo que será modificado que es *IsDB,* que será actualizado al valor 1. En el caso negativo añadimos la nueva carpeta con el valor correspondiente de todos sus campos.

3. Cuando todas las carpetas han sido añadidas y modificadas borramos las entradas de la base de datos propia con el campo *IsDB* a 0, con lo que eliminamos carpetas que han sido borradas de la base de datos de *Mediasite*.

- Obtención de los visores

El proceso de actualización de los distintos visores para poder ver las grabaciones es exactamente igual al de las carpetas, con la diferencia que el método que se invoca es otro.

```
$players =$soapclient->QueryPlayers($getMessage);

if(isset($players->Players->PlayerContext)){
```

Figura 4.14: Comprobación de existencia de visores

El resto del proceso de actualización es exactamente igual que en el caso de las carpetas, con lo que lo omitiremos para hacer la lectura de la memoria menos repetitiva y más amena.

- Obtención de los perfiles de grabación

La ejecución del script continúa, y la siguiente fase es la actualización de los perfiles de grabación con los que se pueden realizar los vídeos. Cabe reseñar que esta es la última fase de actualización que hace usos de tablas que estarán relacionadas con la tabla principal, *presentations*, con lo que cuando acabáramos con esta parte de la ejecución podríamos añadir las presentaciones.

Como pasaba en el caso anterior, este proceso tiene una similitud muy grande con la obtención de presentadores, ya que directamente el método *QueryMediaEncodeProfiles* solo nos proporciona los *ids* y no los datos asociados a cada uno de los perfiles, que obtendremos con el método *QueryMediaEncodeProfileDetails*.

```
$Profiles=$soapclient->QueryMediaEncodeProfiles($getMessage);
if(isset($Profiles->MediaEncodeProfiles->
MediaEncodeProfileContext)){
    // crear lista de Profiles
for($i=0;$i<count($Profiles->MediaEncodeProfiles->
MediaEncodeProfileContext); $i ++){
$ProfilesIdList[$i] = $Profiles->MediaEncodeProfiles->
MediaEncodeProfileContext[$i]->Id;
    }
    $getProfilesDetails = array(
      "UserTicket"=>$ticket,
       "ImpersonationUsername"=>$user,
       "MediaEncodeProfileIdList"=>$ProfilesIdList);
$ProfilesDetails=$soapclient->
QueryMediaEncodeProfileDetails($getProfilesDetails);
```

Figura 4.15: Obtención de perfiles de grabación

A partir de aquí, la rutina de actualización es igual que en los casos anteriores, con la salvedad de que son tres tablas las involucradas en el proceso de actualización.

- Obtención de los catálogos

Como mencionamos anteriormente, esta información no es relevante para el fin de este proyecto, pero aprovechando que las pruebas ya habían sido realizadas se decidió seguir guardando los catálogos para futuras versiones tanto de la aplicación como del mismo sistema *Mediasite.* El proceso es exactamente igual que el de actualización de carpetas o visores, insertando nuevos catálogos, modificándolos o borrando según corresponda.

```
$catalogs =$soapclient->QueryCatalogShares($getMessage);
```

Figura 4.16: Obtención de catálogos

- Obtención de las presentaciones

Por último sólo nos queda añadir las presentaciones presentes en la base de datos del sistema *Mediasite*. El proceso es bastante similar a los anteriores, pero requiere el uso de filtros para filtrar las presentaciones a recuperar que lo hace diferente.

En primer lugar vamos a crear tres arrays necesarios para obtener las presentaciones.

1. El primero de ellos se corresponde con los distintos estados de visualización en los que se puede encontrar una presentación. Nuestra aplicación en principio obtiene todas las presentaciones en los distintos estados, ya que estos pueden cambiar y una presentación que en principio no nos es útil luego si lo puede ser porque su estado cambie o sea modificado. Los distintos estados de las presentaciones son los siguientes:

Scheduled	La presentación está programada en el servidor Mediasite y está lista para ser grabada.
OpenForRecord	La presentación está preparada para ser grabada en uno de los grabadores de Mediasite
Recording	La presentación ha sido grabada en uno de los grabadores
Recorded	La presentación está grabada y está lista para ser subida al servidor de Mediasite
Uploaded	La presentación grabada está subida en el servidor de Mediasite y está lista para ser visualizada bajo demanda.

Tabla 4.12: Tipos de presentación

De estos cinco estados, realmente nos interesan dos de ellos: *Scheduled*, con el que sabremos que presentaciones futuras están programadas y *Uploaded* que son las presentaciones visibles y que están disponibles ser publicadas en ArcaMM, donde podremos indicar si es privada o pública. Los otros tres estados normalmente se corresponden con grabaciones de prueba o vídeos desechados. A pesar de ello, las mantenemos en la base de datos porque su estado puede cambiar a causa de una modificación manual o automática como en el caso de las presentaciones con estado *Scheduled*, que cuando se ha realizado la grabación del evento pasan a estar *Uploaded*.

2. El Segundo array engloba un filtro en el que dando unos determinados valores a cada uno de los parámetros podemos conseguir una serie de presentaciones dependiendo del resultado que queramos obtener. En nuestro caso, como queremos obtener todas las presentaciones que se encuentran en la base de datos de *Mediasite*, la configuración de este filtro no será nada restrictiva. A continuación mostramos una tabla donde se facilitan todos los parámetros configurables, así como su descripción y el valor que se ha dado en la aplicación para conseguir el objetivo de conseguir todas las presentaciones.

Parámetro	Descripción	Valor
TitleRegEx	Titulo de la presentación	Null
StartDate	Fecha de comienzo de búsqueda	2000-01-01T00:00:00
EndDate	Fecha final de búsqueda	2099-12-31T23:59:59
PresenterIdentifier	Id del presentador	Null
Owner	Persona que publicó la presentación	Null
StatusFilterList	Estado de las presentaciones	$presentationDataStatus
PermissionMask	Permisos de las presentaciones	Read
FolderMask	Permisos de las carpetas	Normal
SearchText	Búsqueda de un texto	Null
FieldsToSearch	Búsqueda por algún campo	None
SearchType	Tipo de búsqueda	None
SortBy	Ordenar por un determinado campo	Date
SortDirection	Dirección de la ordenación	Ascending

Tabla 4.13: Filtro de presentaciones

3. El tercer y último array nos resultará familiar, ya que es muy parecido al que hemos usado en los demás procesos de actualización, y donde utilizaremos el ticket de sesión, el nombre de usuario y el filtro que hemos creado en la fase anterior.

A partir de este punto el proceso de actualización es muy similar al de las otras partes del script, aunque vamos a hacer hincapié en él debido a que existen otros filtros que no aparecen en la actualización del resto de metadatos.

Después de comprobar que existen presentaciones en la base de datos, debemos crear la lista de todos los identificadores correspondientes a cada una de las presentaciones. Como en otras ocasiones para obtener los detalles de una presentación debemos hacerlo pasándole como parámetro al método el id correspondiste, que previamente hemos guardado en la lista.

Seguidamente debemos de crear un nuevo filtro para poder recuperar todos los metadatos asociados a las presentaciones. En este nuevo filtro, lo realmente interesante para nuestro script es que recupere toda la información de los vídeos que están disponibles para lectura. El resto de parámetros no son de importancia pero deben de aparecer para conformar el filtro, con los que los daremos un valor de true para no ser restrictivos.

```
$filter= array(
                "PermissionMask"=>"Read",
                "IncludeMediaEncodeProfile"=>true,
                "IncludePresenterList"=>true,
                "IncludeSuportingLinks"=>true,
                "IncludeAttachments"=>true);
```

Figura 4.17: Formación del filtro

Después ya se pude conformar el array que será pasado como parámetro al método *QueryPresentationDetails.* Este array estará formado, como en otras ocasiones, por el ticket de sesión, el nombre de usuario que generó el ticket, la lista de presentaciones de las que se desea obtener la información, que en nuestro caso serán todas, y el anterior filtro descrito.

El resultado será un array con tantos ítems, como presentaciones presentes en la base de datos, que serán tratadas de igual manera que en las actualizaciones de datos anteriores, por lo que no volveremos a entrar en detalle en el proceso.

La estructura de nuestra base de datos nos permite ahorrarnos columnas a la hora de guardar las presentaciones, ya que el método nos devuelve toda la información referente a una presentación, pero también la información relativa al presentador o presentadores que realizaron la presentación, el visor, la carpeta o el perfil de grabación. Con nuestra estructura, solo almacenamos el id correspondiente, y dado que nuestra base de datos tiene integridad referencial entre las tablas, siempre podremos tener acceso a dicha información aunque se encuentre en otra tabla. Aún así, nos surgió un problema a la hora de guardar el campo *presenters* de la tabla *presentation*, ya que este campo podía tener más de un autor de la grabación.

Se propusieron dos soluciones para solventar el problema:

- Crear una nueva columna por cada presentador adicional que apareciera en la presentación. Con esto mantendríamos la integridad referencial intacta, pero nos surgiría otro problema que haría que nuestra tabla aumentara de tamaño en cualquier momento, afectando al resto de entradas que tuvieran menos presentadores que la presentación a introducir.

- Mantener una única columna, pero añadir los ids de los presentadores concatenados y separados por un carácter de escape conocido, que en nuestro caso sería "/". Con esto solucionamos la expansión de la base de datos, ya que la estructura no cambiará en ninguno de los casos pero a cambio perdemos la integridad referencial con la tabla *presenters*.

Se decide optar por la segunda de las opciones, ya que debido a la longitud fija de los ids, resulta muy fácil de recuperarlos y tratarlos de forma independiente cuando sea necesario. La perdida de la integridad referencial no lo vemos un inconveniente de peso para usar la otra opción que consideramos más costosa, ya que mantenemos los ids de la tabla *presenters* como *primary key,* con lo que si fuera necesario realizar alguna operación de borrado o actualización esta la haríamos de forma manual.

Con el final de este proceso daría por finalizada la actualización de los metadatos correspondientes a cada una de las presentaciones que se encuentren en la base de datos de *Mediasite.* Además siempre que una presentación nueva es añadida a la base de datos, aprovechamos esa inserción para añadir el id de dicha presentación a la tabla *uploaded_arcamm,* dejando ya preparada la presentación para su posible publicación. El resto de campos quedan vacios con excepción del campo *published* que tendrá un valor de 0, que significa que la presentación no está publicada en el portal de vídeos de la UC3M

Durante el periodo de pruebas y refinamiento del script de actualización observamos que el tiempo de ejecución es de unos 30 segundos aproximadamente y nos parece relativamente alto para el tamaño de la base de datos con la que trabajamos. Estudiamos las posibles causas por lo que esto pudiera suceder: realizamos demasiadas consultas a la base de datos propia o la transferencia de datos desde el servidor de *Mediasite* a nuestra máquina local era más lento de lo que esperamos.

Para comprobar a que se debe este retardo, ideamos una alternativa de diseño, en la que al inicio de la actualización, la base de datos propia es vaciada por completo y añadíamos toda la información correspondiente según la idea original y que hemos explicado anteriormente. Con esto eliminamos una operación que pensamos que es más costosa computacionalmente, la de comparación de nuestros datos guardados con los de la base de datos de *Mediasite*, ya que pueden hacerse modificaciones en los vídeos ya grabados, se añaden presentaciones nuevas o se borran. Un inconveniente que encontramos a esta alternativa de diseño es el tratamiento que debíamos de hacer a la tabla *uploaded_arcamm,* que al ser la encargada de unir nuestra base de datos con ArcaMM, no podíamos vaciarla ya que perderíamos la información correspondiente a la publicación. Esto haría que tuviéramos que hacer alguna modificación no programada en el plan de trabajo cuando llegáramos a ese punto del proyecto. Aún así, si esta alternativa reducía el tiempo de procesamiento, intentaríamos resolver el problema, pero cuando observamos que el tiempo se reducía en apenas unas decimas desechamos completamente esta alternativa de diseño.

Con esto concluimos que el retardo era producido en la comunicación de una máquina externa con el servidor *Mediasite*, encargado de guardar la información de las presentaciones, con lo que el tiempo nos sería imposible reducirlo.

4.1.3 AUTOMATIZACIÓN DEL PROCESO

Pasadas dos semanas en las que se testea el script y en los que se depuran errores de guardado de caracteres o modificación y borrado de presentaciones en la base de datos, observamos que ejecutando de forma manual el script a través del navegador este funciona de forma correcta. Esta parte del proyecto la daríamos por concluida quedándonos únicamente el automatizado del proceso.

La máquina donde está alojado el portal ArcaMM y donde se guardarán los metadatos y se ejecutará nuestra aplicación, usa el sistema operativo Linux, el mismo que hemos utilizado en nuestro entorno de desarrollo, con lo que esta automatización se puede hacer de forma sencilla utilizando la herramienta integrada *cron.*

Cron es el programa que permite a los usuarios ejecutar automáticamente comandos o scripts (grupos de comandos) a una hora o fecha específica. Esto es exactamente lo que buscamos. La versatilidad de la herramienta es bastante alta, ya que nos permite hacer rangos de tiempo bastante concretos excluyendo horas, días o meses. Este aspecto nos parece de bastante importancia porque no queremos saturar el servidor de forma innecesaria, ya que además de nuestro script, ejecuta muchos más procesos para el mantenimiento y buen funcionamiento del portal de vídeos de la UC3M.

El tiempo en el que nuestra script debe funcionar es durante el horario lectivo y laboral de la Universidad, que es de lunes a viernes de 8:00 a 22:00. Con esto excluimos 10 horas durante los días lectivos y dos días completos durante el fin de semana. Esto significa que nuestra aplicación funcionará 70 horas de las 168 que conforman el total de una semana, lo que supone un ahorro de recursos considerables al servidor. Además de crear estos tramos temporales nos queda decidir el intervalo en el que script se ejecutará dentro de esos tramos. Para decidirlo tenemos que tener presente en todo momento que debe existir un compromiso entre eficiencia del servidor y tiempo de espera para publicar el vídeo.

En este compromiso ponderamos los dos factores por igual con lo que no podemos elegir un tiempo muy pequeño (mínimo un minuto), ya que lo que hemos ganado creando los tramos temporales lo perdemos. Lo contrario pasa si elegimos un tiempo superior como una actualización al día, en las que el personal del Área de Audiovisuales y el autor del vídeo deberían esperar a que la base de datos se refrescara, pudiendo ser de un día esta espera. Para decidir el intervalo observamos la duración que tienen las presentaciones en la fecha actual (Junio 2012) donde 451 presentaciones tienen una duración total de 457 horas 53 minutos y 24 segundos, observando que una presentación suele tener de media una duración de poco más de una hora. Este dato nos parece bastante esclarecedor ya que un intervalo de tiempo menor a una hora habría muchísimas posibilidades de que no se hubiera creado ningún vídeo nuevo. Con esto no decimos que no pueda pasar, pero es menos probable estadísticamente. Consideramos que a partir de que el intervalo sea de una hora respetamos la eficiencia del servidor y es un tiempo prudencial de espera para el publicador, que como máximo esperará una hora para publicar el nuevo vídeo en el portal de la Universidad.

Además de estos tramos temporales que ya hemos mencionado, añadimos otros dos donde el script se actualizará únicamente una vez al día. Estos dos periodos son: la Navidad, desde el día 24 de Diciembre al 1 de Enero, y el verano, desde el 1 al 31 de Agosto. Podríamos añadir algún tramo adicional en los que la universidad permanece cerrada como festivos nacionales, pero al no ser fijos, preferimos no hacerlo y así evitar posibles problemas.

Finalmente solo nos quedaría alojar tanto la base de datos como los archivos necesarios en el servidor y añadir las rutinas antes expuestas en el *cron* de dicha máquina. Únicamente exportamos la estructura de la base de datos, ya que esta se rellenará en la primera ejecución del script de actualización.

Con esto todas las presentaciones estarán accesibles en nuestra base de datos y podrán ser publicadas en ArcaMM cuando se deseé. Esto nos lleva a la segunda parte de la implementación, que es la integración con ArcaMM, en la que crearemos una interfaz de usuario donde se mostrarán los vídeos disponibles para su publicación.

4.2 INTEGRACIÓN EN ARCAMM

En esta segunda y última parte del desarrollo de la herramienta de publicación, nuestro objetivo es crear una interfaz de usuario donde el personal de Audiovisuales pueda buscar y publicar un determinado vídeo en la plataforma. Al reunirnos con el beneficiario de la herramienta, el Área de Audiovisuales, nos propone una serie de requisitos mínimos que debe cumplir la interfaz para satisfacer las necesidades de su personal.

Los requisitos que se exigen son los siguientes:

- Mostrar todas las presentaciones disponibles para publicar.
- Mostrar las presentaciones publicadas.
- Permitir una búsqueda acotada a un determinado tag, autor, campus o entre dos determinadas fechas.
- Permitir al usuario crear estadísticas.
- La interfaz esté integrada con el resto de la plataforma ArcaMM. Nuestra herramienta debe ser un rol más dentro de ArcaMM y a la que sólo podremos acceder cuando el administrador del sistema asuma dicho rol a nuestra cuenta LDAP de la universidad. Esto implica que el estilo de la interfaz tiene que estar en armonía con el resto de herramientas presentes en la plataforma.

A partir de aquí nosotros como programadores tenemos libertad para resolver los requisitos de la manera que creamos oportuna, pero se nos recomienda que evaluemos la estructura actual de ArcaMM y que desarrollemos el código sobre esa base. Además se nos invita a crear cualquier función adicional que creamos oportuna para dar a la herramienta el máximo de funcionalidad.

Viendo las posibilidades que nos presentan los datos que extraemos creemos conveniente añadir las siguientes funciones:

- Facilitar al personal de Audiovisuales la fecha y el enlace de las presentaciones futuras programadas y que se pueden seguir en directo.

- Mostrar en una ventana aparte toda la información referente al vídeo. Esto nos permitiría hacer una interfaz mucho más visual ya que no recargaríamos la pantalla de una cantidad de datos que en ocasiones pueden ser innecesarios. El usuario decide de que vídeo quiere ver la totalidad de sus datos

- Paginación de los resultados que devuelve la búsqueda.

- Permitir al usuario cambiar la configuración del actualizador de vídeos en el caso de cambios de servidor, usuario, contraseña o imagen predefinida.

Para poder entender donde se alojará nuestra herramienta web primero debemos tener muy claro el funcionamiento del sitio ArcaMM.

Haremos una breve descripción del modo de acceso al sitio y las diferencias entre un cliente sin permisos y otro que si los tiene.

Lo primero que se encuentra el usuario al teclear http://arcamm.uc3m.es/arcamm/ es la página principal del sitio:

Figura 4.18: Portal de vídeos ArcaMM

La imagen no es exactamente real, ya que se ha dividido por la mitad para que el aspecto fuera similar a la impresión “de dos páginas en una cara”.

Como podemos observar cualquier usuario puede buscar un determinado vídeo (1), navegar libremente por las distintas secciones de la web (2) o mirar los eventos tanto futuros como pasados que se llevan a cabo en los distintos campus de la universidad (3). La diferencia entre un usuario con privilegios y otro que no los tiene, es el candado (4) que se encuentra en la parte inferior derecha de la pantalla. Pulsando sobre este icono nos redirigirá a una página de autentificación, donde debemos introducir nuestros credenciales para poder acceder a las distintas herramientas administrativas disponibles.

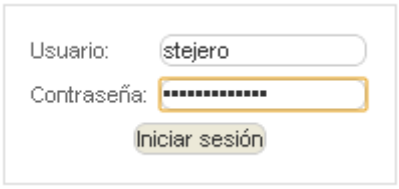


Figura 4.19: Login ArcaMM

Si tenemos permisos de acceso y nuestro autentificación se ha realizado de forma correcta, la siguiente pantalla que nos aparecerá será la de elección del rol que queremos asumir. Todos los usuarios con acceso no tienen el mismo número de roles asignados ya que eso depende de las funciones que cada usuario tiene dentro del sistema, siendo el administrador el único con acceso a todos los roles.

Figura 4.20: Selección de rol

Aquí se muestran los roles asignados a mi persona como miembro del personal del Área de Audiovisuales. Como vemos el acceso a la interfaz se realiza seleccionando el rol llamado *mediasite*. Al seleccionar dicho rol cargará el fichero *medisite.php*, que será la página principal de la herramienta.

La adecuación de ArcaMM para nuestra herramienta es un proceso realizado por el equipo de mantenimiento del sistema en el que nosotros no tuvimos que hacer nada. Ellos añadieron el rol a la base de datos y él único requisito que se nos exigió fue el de añadir una serie de comprobaciones de seguridad al inicio del fichero *mediasite.php* donde comprobáramos que la autentificación y el rol asumido eran correctos. Esta comprobación era similar a la que se hace en el resto de roles, ya que con esto evitábamos accesos a la herramienta a través de url, requiriendo siempre la autentificación.

Lo primero que se encuentra el usuario es la página principal de la herramienta es lo siguiente:

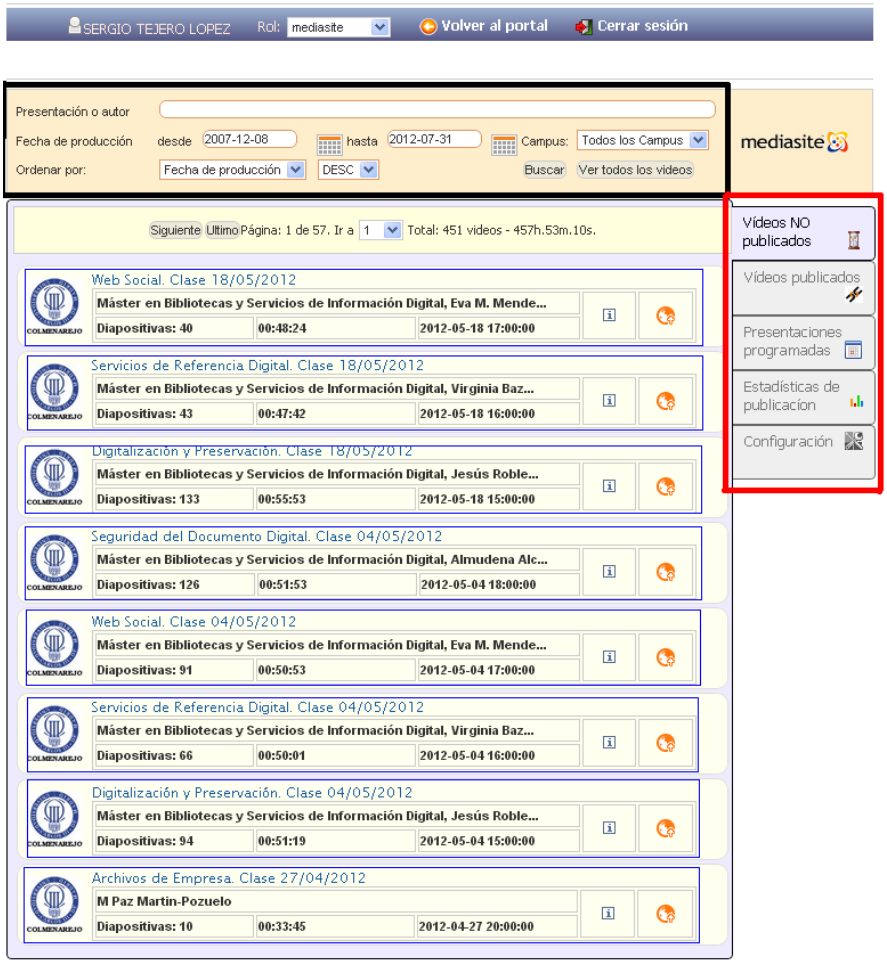


Figura 4.21: Herramienta de publicación

En la figura podemos distinguir tres partes diferenciadas:

- La que comprende el trazo grueso negro se corresponde con el módulo de búsqueda que se proporciona al usuario para que encuentre de forma intuitiva el vídeo que desea publicar.

- La que comprende el trazo grueso rojo se corresponde con las distintas secciones de las que goza la aplicación y que el usuario puede acceder pinchando sobre ellas

- El resto de la interfaz se corresponde con la muestra de resultados de cada una de las secciones. En el caso de la figura observamos que se muestran los vídeos disponibles para su publicación. Cada trazo fino azul se corresponde con cada uno de los ítems que se muestran. Como vemos en total existen 451 vídeos, de los que mostramos únicamente ocho para hacer más visual la muestra de resultados. El resto podemos verlos haciendo uso de la paginación.

A continuación explicaremos la función de cada una de las partes de la aplicación así como el porqué de las decisiones tomadas para la creación de dichas funciones.

4.2.1 MÓDULO DE BÚSQUEDA

Tal como se nos exigía en los requisitos que debía tener la aplicación, debíamos permitir al usuario realizar una búsqueda acotada al menos entre unas fechas determinadas. Posteriormente con la información que podíamos obtener de las presentaciones propusimos añadir otros criterios de búsqueda que facilitarían el trabajo al usuario.

Figura 4.22: Módulo de búsqueda

- **Presentación o autor:** Se trata en un campo de texto en el cual podemos introducir cualquier palabra que creamos que tiene algún tipo de correspondencia con el título de alguna presentación o el nombre de algún autor. Al hacer la consulta a la base de datos, después de haber mandado el formulario de búsqueda con este campo rellenado, nos encontramos con una serie de problemas: no podíamos introducir más de una palabra en el campo de texto ya que tomaba las palabra como una única y las palabras eran sensibles a tildes con lo que si el usuario introducía una palabra sin tilde y la manera correcta de escribirla era con tilde, la aplicación no devolvía ningún resultado. Igualmente ocurría en el caso contrario.

  Para resolver el primero de los problemas utilizamos la función definida en php *explode*, la cual nos permite separar una cadena de caracteres en cadenas más cortas usando un determinado carácter que deberá estar presente en dicha cadena. El carácter que utilizamos para separar la cadena es un espacio. El porqué de la elección de este carácter y no otro como puede ser la *coma (,)* es que la mayoría de los usuarios están acostumbrados a los buscadores web como Google en los que la separación de palabras se realiza con espacios y no con comas.

  Para resolver el segundo problema creamos una función a la que le pasaremos como parámetro cada una de las palabras introducidas por el usuario. Con esto haremos que las vocales que lleven tilde las convertiremos en vocales sin tilde. La razón de tomar esta decisión es que nos dimos cuenta que al realizar la consulta a la base de datos esta no era sensible a tildes, es decir si introducíamos la palabra *vídeo* nos devolvía resultados en los apareciera la palabra vídeo y *video.*

- **Fecha de producción:** En estos dos campos permitimos al usuario acotar la búsqueda entre una fecha inicial y otra final. En el caso de que alguno o ambos campos queden vacios, los criterios introducidos serán predeterminados. En el caso de que ambos estén vacios, no habrá ningún tipo de filtro. En el caso de que la fecha inicial esté vacía y la final rellenada, la fecha inicial predeterminada será el 01-01-2007, ya que no existen vídeos anteriores a dicha fecha. En el caso contrario, la fecha final será la fecha actual en la que se ha realizado la búsqueda, ya que no existen vídeos realizados en una fecha futura.

  Para ayudar al usuario a introducir la fecha en ambos campos, si pulsa sobre el icono adyacente, emergerá un calendario donde podrá elegir la fecha que deseé. Esto lo hacemos creando una función llamada *calendar.js*, que devuelve una cadena de texto que es introducida dentro del campo de fecha.

Figura 4.23: Calendarios en módulo de búsqueda

- **Campus:** Permitimos al usuario elegir entre cuatro opciones, Todos los campus, Getafe, Leganés o Colmenarejo. Esta es una función muy útil ya que el personal de Audiovisuales es diferente en cada uno de los campos y normalmente sólo publican vídeos realizados en su lugar de trabajo.

- **Ordenar por:** Con esta lista similar a la de Campus, damos la opción al usuario de ordenar los resultados por Fecha de Producción, Nombre, Autor o Duración. Además esta ordenación puede ser en orden ascendente o descendente que se elige con la lista desplegable adyacente.

Este módulo no está siempre visible, ya que cuando pulsamos sobre estadísticas o configuración, este se oculta de forma automática ya que no es necesario. El manejo para hacer estos cambios lo hacemos con la función *showParts*, ya que al hacer click sobre una de las pestañas se llama a esa función de *JavaScript*. Esta función además de mostrar la pestaña seleccionada y ocultar el resto, maneja el bloqueo o la ocultación del *div* donde se encuentra el módulo de búsqueda y da a la variable *tab* el valor de la pestaña activa. Esto es de suma importancia, ya que para conseguir que el módulo de búsqueda sea común a las tres pestañas, pero funcione de forma independiente, se necesita saber en qué pestaña está interactuando el usuario. Con esto se consigue que el mismo código sea común a tres páginas distintas pero que funcionara de forma independiente. Cuando el usuario pulsa el botón *Buscar* o *Ver todos los vídeos* el resultado será referente a los vídeos no publicados, publicados en ArcaMM *o* presentaciones programas dependiendo de la pestaña que tuviera activa el usuario.

4.2.2 PESTAÑAS

La segunda parte que podemos diferenciar en la interfaz de usuario es la zona donde se ubican las pestañas donde se permite navegar entre las distintas funcionalidades que ofrece la web.

Cada vez que nuestra página recibe un refresco, esta carga las cinco pestañas independientemente en la que esté trabajando el usuario. El manejo que hace la aplicación es mostrar después del refresco la misma página en la que se estaba trabajando y ocultar las otras cuatro. A pesar de esto el usuario puede elegir otra área de trabajo pulsando sobre cualquiera de las pestañas que se encuentran sombreadas, que se encontrarán en el mismo estado que estarían si se accediera a la aplicación por primera vez.

Al usuario se le ofrecen cinco secciones donde se le proporcionan funcionalidades distintas:

- **Vídeos no publicados:** Aquí el usuario podrá ver todos los vídeos que no están publicados en ArcaMM. Se le permite filtrar los resultados mediante el módulo de búsqueda. Este es el lugar desde donde se publicarán los vídeos, ya que cuando el usuario encuentre la presentación buscada, la aplicación le redireccionará a la página de ArcaMM donde se publican todo los vídeos que se visualizarán en la plataforma.

- **Vídeos publicados:** Se muestran todos los vídeos creados con *Mediasite* que ya han sido publicados en ArcaMM. Desde aquí se puede acceder a la publicación para realizar cambios o simplemente visualizar el vídeo.

- **Presentaciones programadas:** Se muestran las presentaciones que en un futuro serán grabadas y que ya están programadas.

- **Estadísticas:** En esta sección permitimos al personal de Audiovisuales crear estadísticas sobre la publicación de los vídeos y uso de la aplicación.

- **Configuración:** Permitimos al usuario poder cambiar los parámetros de la herramienta de actualización de vídeos.

4.2.2.1 Vídeos no publicados

En esta primera sección permitimos al usuario publicar el vídeo que desee en la plataforma de vídeos de la Universidad Carlos III. Esta página es lo primero que se encuentra el usuario al acceder a la aplicación, ya que consideramos que es la sección más importante de la aplicación y en la que gira este proyecto.

Figura 4.24: Vídeos no publicados

Presentamos los distintos vídeos como ítems independientes proporcionando al usuario información relevante acerca de cada uno de los vídeos: nombre de la presentación, autor, número de diapositivas o transiciones en la presentación PPT, duración y fecha de realización de la grabación.

Como vemos únicamente se muestran 8 vídeos en la pantalla de los 535 que existen sin publicar. Esto es debido a que paginamos los resultados de la búsqueda en páginas de 8 ítems cada una, lo que permite una interfaz más intuitiva y mucho menos cargada de información.

Para rellenar la página de la información requerida, la página principal *mediasite.php* hace un *include* de los cinco ficheros que conforman cada una de las secciones, entre ellos el de *notPublished.php* que es el que nos interesa en este apartado.

El fichero lo primero que hace es comprobar si está recibiendo la variable *tab=notPublished*, lo que querrá decir que el usuario ha realizado una búsqueda y que además la estaba haciendo en los vídeos no publicados. En el caso de que no exista dicha variable o no tenga el valor *notPublished* el fichero cargará todas las presentaciones no publicadas sin filtrar y dejará la pestaña visible u oculta según corresponda.

Pongamonos en el caso de que el usuario este inmerso en una búsqueda con lo que hay que hacer una determinada consulta a la base de datos. La *query* debe formarse a partir de los criterios de búsqueda que ha introducido el usuario. Estos han sido enviados a través de un formulario y son recibidos por la nueva página a través del navegador. En la siguiente figura mostramos como se crea dicha *query:*

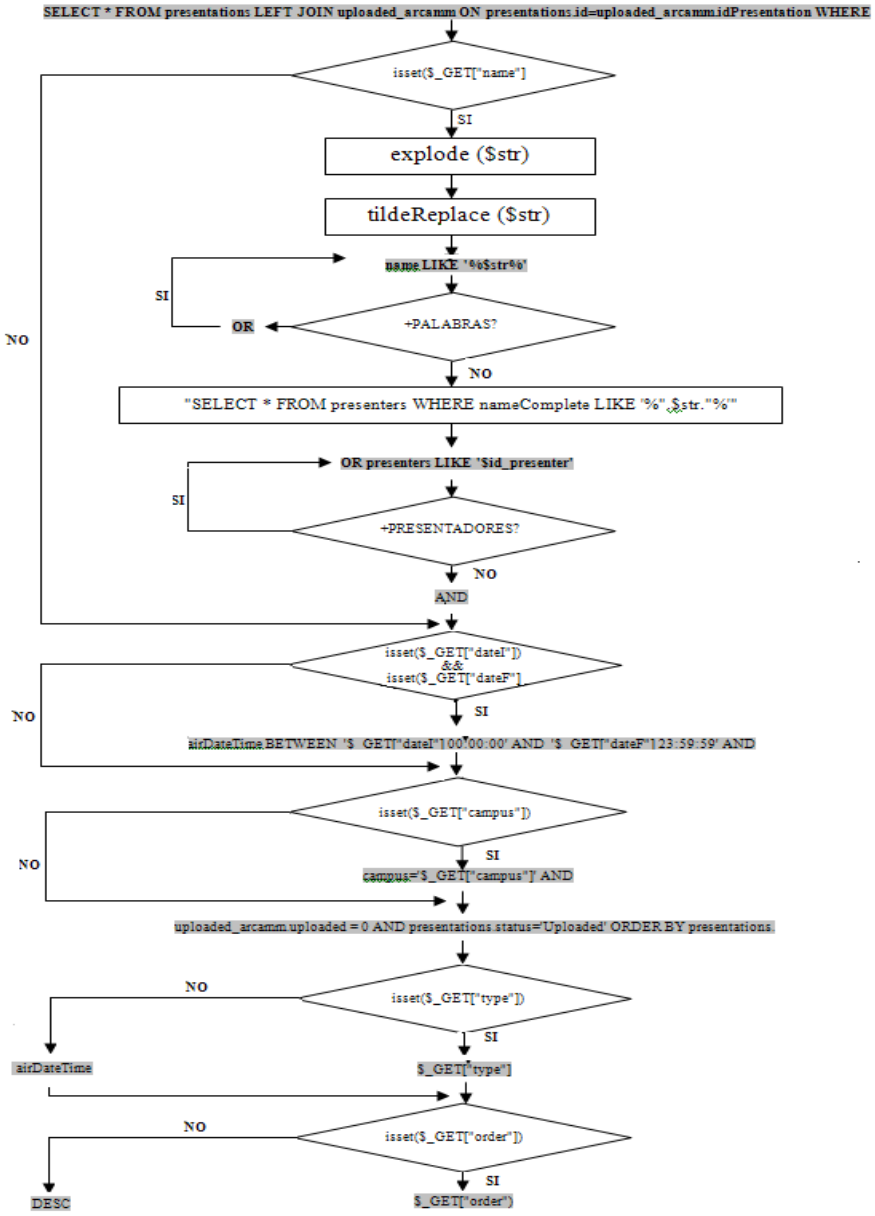


Figura 4.25: Consulta enviada a la base de datos

En el caso de que la búsqueda sea satisfactoria se mostrarán las presentaciones resultantes según los criterios de búsqueda. En el caso de la que búsqueda no encuentre ningún resultado, se mostrará el siguiente mensaje:

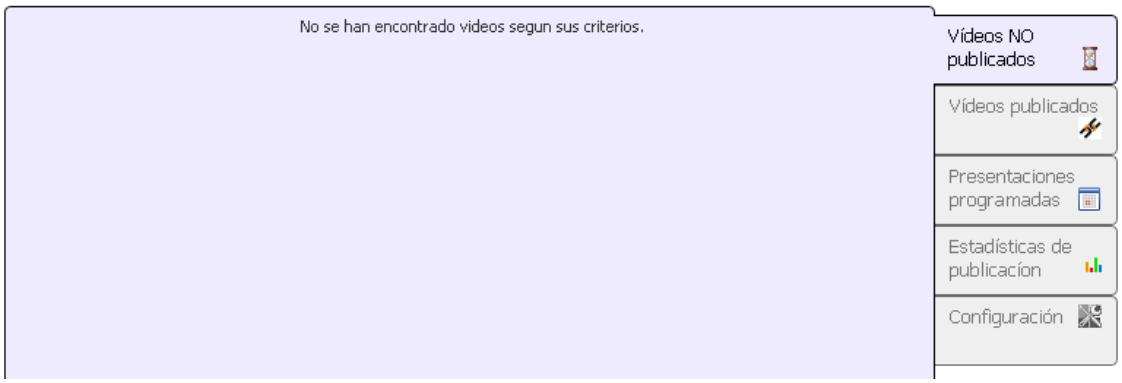


Figura 4.26: Búsqueda fallida

Cuando esto ocurra, el usuario puede volver a realizar otra búsqueda cambiando cualquiera de los criterios, ya que estos persisten en el módulo de búsqueda. Esto lo conseguimos comprobando en cada refresco si recibimos variables GET a través del navegador y si estás se corresponden con cualquiera de las variables de los campos de búsqueda, rellenándolos correspondientemente en caso afirmativo. En el caso de que queramos mostrar todas las presentaciones de nuevo siempre podemos recurrir al botón "Ver todos los vídeos".

En relación a los ítems o presentaciones que aparecen como resultados, decidimos mostrarlos de una forma resumida para no recargar los resultados de información, que en muchos casos sería innecesaria para el publicador. Para ello paginamos los resultados en grupos de ocho ítems, permitiendo al usuario navegar a través de distintos botones y de una lista desplegable para ir a una página concreta. Cuando se pulsa cualquiera de los botones de navegación o se elige una página a través de la lista desplegable, la aplicación refresca el navegador mandando las variables GET correspondientes a la búsqueda, la pestaña en la que se está trabajando y una nueva variable llamada *currentpage* que indica la página actual en la que se encuentra la paginación y que utilizaremos para añadir el valor de *LIMIT* en la *query* ya formada. Gracias a esta variable, como pasa con otras como *tab*, la aplicación tiene el control de las acciones que realiza el usuario dando la robustez suficiente para que la aplicación funcione correctamente.

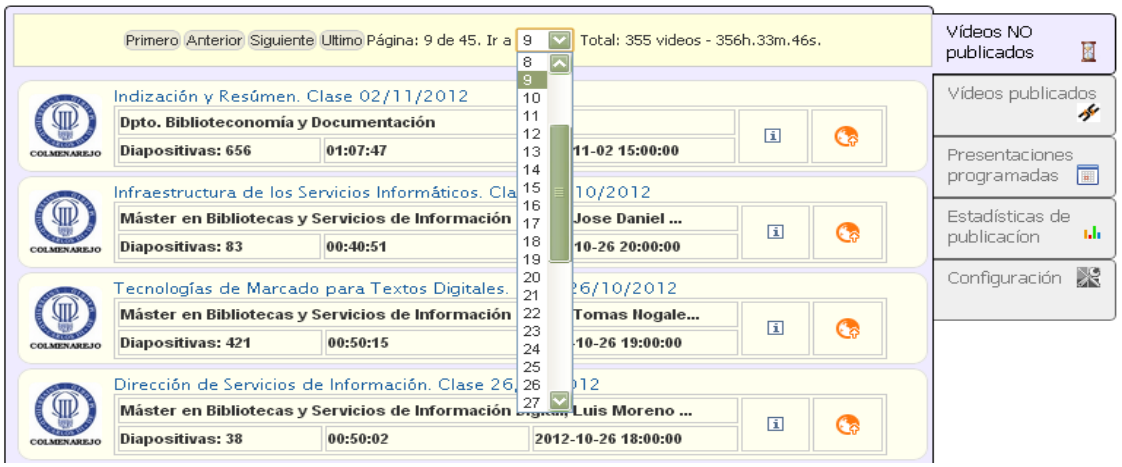


Figura 4.27: Paginación de resultados

La otra forma con la que conseguimos no recargar de información al usuario es mostrando sólo la información relativa a un vídeo que consideramos más relevante y que en la mayoría de las ocasiones será necesaria para encontrar una presentación en concreto.

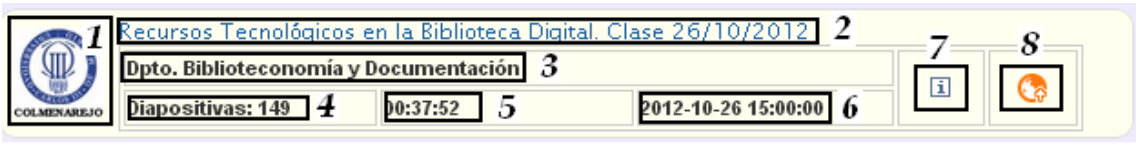


Figura 4.28: Secciones del ítem vídeo no publicado

Las partes diferenciadas que podemos observar son las siguientes:

1. **Imagen de la presentación**. Al no poder extraer fotogramas de la presentaciones, decidimos crear una foto genérica en el que estuviera presente el logo de la universidad y el campus donde se realizó la grabación. En el sistema existen tres posibles imágenes correspondientes a cada una de los tres campus de la universidad. La aplicación elige la imagen dependiendo del valor que tenga el campo "campus" en la base de datos para dicha presentación.

2. **Título de la presentación**. Además de proporcionar un título, es un enlace a la presentación para poder visualizarla y comprobar que es la que estamos buscando.

3. **Autor o autores de la presentación**. En el caso de que la longitud del autor o los autores sea superior a 75 caracteres, la aplicación solo mostrará esa longitud de la cadena y añadirá unos puntos suspensivos que indicarán que la cadena está incompleta. Hacemos esto para evitar que el autor aparezca en dos líneas y el tamaño del ítem sea diferente al resto y por lo tanto no sea una interfaz homogénea.

4. **Diapositivas**. En realidad no son diapositivas, son las veces que la señal VGA se modifica. Esto puede pasar: al pasar de una diapositiva a otra o puede ocurrir en diapositivas donde sus elementos aparecen de forma gradual cuando el profesor realiza un clic con el ratón o cuando realiza alguna anotación sobre la misma transparencia.

5. **Duración de la presentación**

6. **Fecha de producción del vídeo**.

7. **Información adicional de la presentación**

8. **Publicación del vídeo en ArcaMM.**

Como observamos en la figura anterior permitimos al publicador poder revisar la información adicional de cualquier presentación mediante el botón "Información Adicional". Al presionar sobre este botón nos emergerá una nueva ventana en la que se nos mostrará toda la información que tenemos almacenada en la base de datos relativa a la presentación.

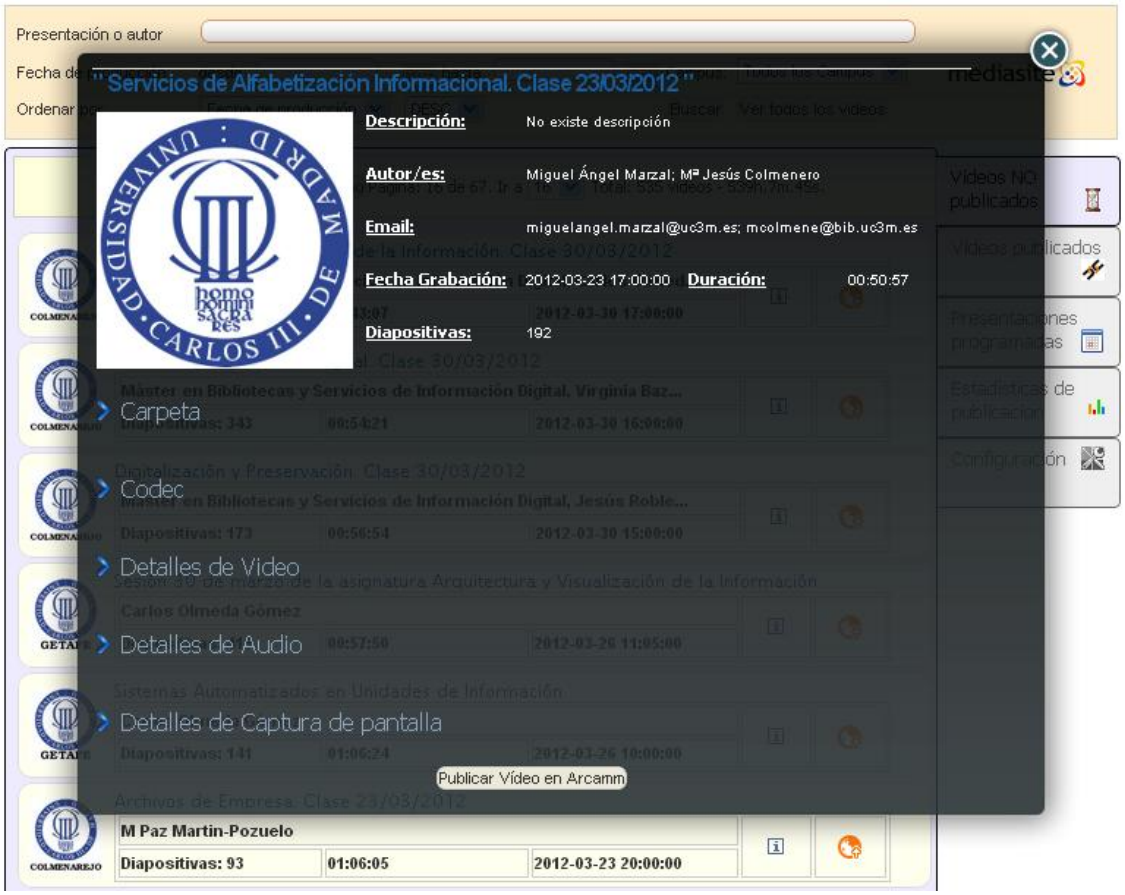


Figura 4.29: Información adicional

Para realizar este efecto hacemos uso de la librería de *JavaScript jQuery*, que de una forma sencilla podemos hacer emerger la ventana sin que la página sufra ningún refresco aparente. La manera en la que la cargamos es utilizando el componente *overlay* de *jQuery,* que nos proporciona que la ventana emerja gradualmente a su tamaño final desde el mismo botón. Esto lo conseguimos asignando el valor *apple* al parámetro configurable *effect.* Para cargar la información correctamente en la ventana debemos de manejar el evento *OnBeforeLoad,* en el cual primero guardaremos en una variable la locación del *div,* donde queremos que se cargue la información para posteriormente sobre esa variable cargar el atributo *href* de la etiqueta <a> en la que estamos trabajando

```
$(function() {
  $("a[rel]='overlay'").overlay({
      effect: 'apple',
      onBeforeLoad: function() {
         var wrap = this.getOverlay().find(".contentWrap");
         wrap.load(this.getTrigger().attr("href"));}});});
```

Figura 4.30: Cargar ventana en la página

Una vez hecho esto, en la ventana se cargará el valor de *href* al que apunta el botón. A través de este valor pasamos como parámetro el id del vídeo del que deseamos recuperar toda la información. Además se cargará el fichero *information.php*, aunque esto es totalmente invisible para el publicador. Este fichero que recibe como parámetro el id de la presentación, realiza una búsqueda en la base de datos propia y recupera toda la información relativa al vídeo. En principio tampoco mostramos toda la información al usuario, pero le permitimos poder interactuar con la ventana, ya que pulsando sobre las flechas, se mostrará o se ocultará la información relativa a cada una de las categorías. Para realizar este efecto volvemos a echar mano de la librería *jQuery*, en concreto de la función *toggle()*.

Figura 4.31: Mostrar/Ocultar detalles

Al final el usuario tiene la decisión de publicar el vídeo o bien cerrar la ventana y seguir buscando el vídeo que crea conveniente. Para ello se proporciona un botón que lleva al usuario a la pantalla de publicación y un botón para cerrar la ventana, devolviendo al usuario al mismo sitio donde lo dejó antes de visualizar la información adicional de la presentación.

Para finalizar este apartado solo nos quedaría hablar del botón “Publicar”, que como su propio nombre indica, es el que direcciona al usuario a la pantalla de publicación. Al ser el paso previo a la publicación del vídeo, los detalles los abordaremos en el apartado 4.2.3, así como los cambios necesarios que realizamos en la herramienta de publicación de ArcaMM, para convivir con el resto de aplicaciones corporativas de la Universidad Carlos III.

4.2.2.2 Vídeos publicados

En esta sección mostramos al usuario todos los vídeos que han sido publicados en ArcaMM y que han sido producidos con *Mediasite*. El diseño es muy similar al de la pestaña de vídeos no publicados, manteniendo así la homogeneidad de la página.

La función de esta sección es meramente informativa, aunque permite al usuario acceder directamente a la publicación y así poder realizar modificaciones o el borrado de la publicación en ArcaMM.

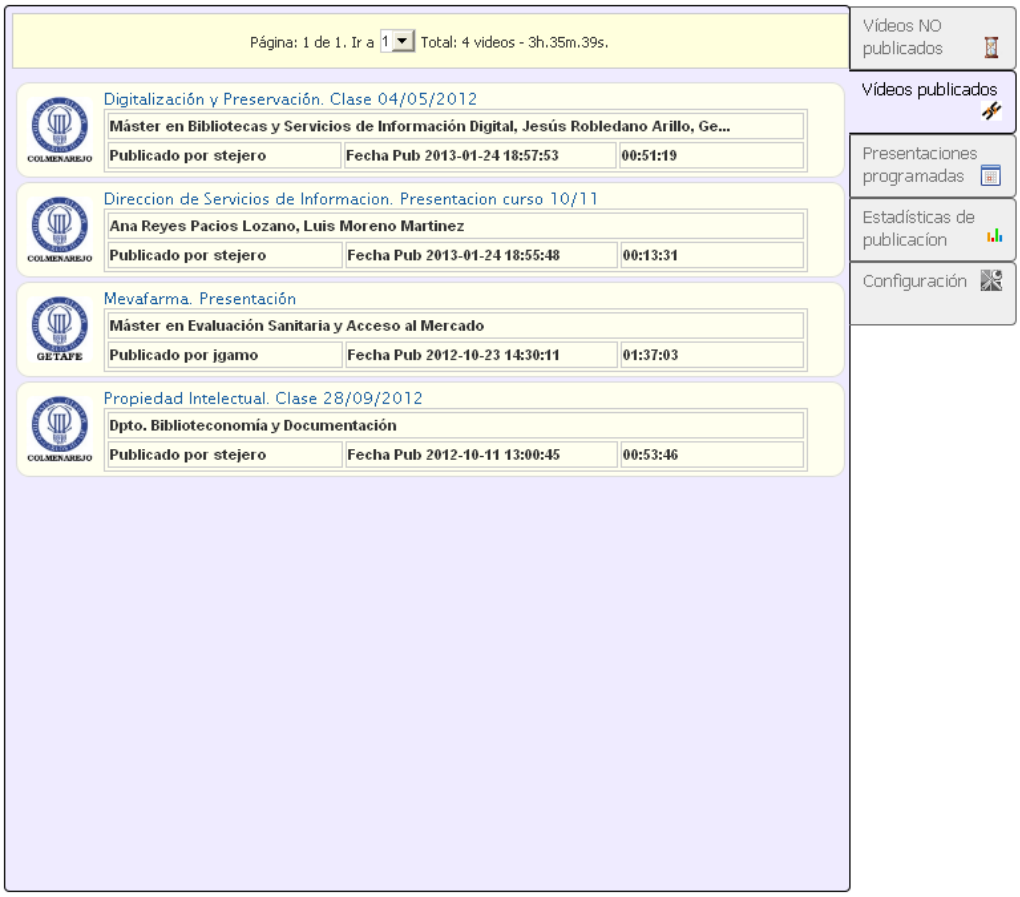


Figura 4.32: Vídeos publicados

La forma de realizar esta sección no supuso mucho tiempo al equipo de trabajo, ya que se utilizó como plantilla la sección "Vídeos no publicados", aunque como podemos observar en la imagen tuvimos que hacer algunos cambios, tanto visibles como invisibles al usuario.

En primer lugar, los ítems se presentan con datos diferentes al de los vídeos publicados, haciendo hincapié en los datos referentes a la publicación como son: la fecha de publicación o la persona que la realizó. Además mantenemos la imagen indicando el campus donde se realizó la grabación, así como el título, los autores o la duración de la grabación.

El resto de datos, como los que aparecían en la ventana "información adicional", decidimos omitirlos de esta vista ya que no es una información fundamental para la persona que esté haciendo uso de la aplicación y se encuentre trabajando en esta sección. Otra razón por la que omitimos mostrar más información es que esta es accesible desde la publicación en ArcaMM, por ello el título de la presentación es un enlace directo a la publicación, en la que podemos ver toda la información referente al vídeo, así como editarlo o borrarlo si se tienen los permisos de administración pertinentes. Por lo tanto el título aunque parezca a simple vista que tiene la misma funcionalidad que en la sección anterior, difiere en la dirección de destino, ya que en el caso de los vídeos no publicados somos dirigidos al enlace del vídeo dentro del mismo servidor de *Mediasite* mientras que en este caso nos lleva al Portal de vídeos de la UC3M.

En segundo y último lugar, tuvimos que hacer una modificación en la consulta que debemos hacer en la base de datos para que nos muestre los vídeos correctamente. Independiente de que la página se mostrará después de una búsqueda o de forma predeterminada, el cambio más sustancial en la consulta era que debíamos buscar únicamente en los vídeos que tuvieran el campo *uploaded* de la tabla *uploaded_arcamm* igual a 1. Además de esto, como en esta sección la fecha que nos interesa es la de publicación y no la de producción, cuando se filtra por fecha en la búsqueda, esta se realiza sobre la fecha de publicación. Aquí podemos ver a modo de ejemplo como sería la consulta por defecto:

```
$sql ="SELECT * FROM presentations LEFT JOIN uploaded_arcamm ON
      presentations.id=uploaded_arcamm.idPresentation WHERE
      uploaded_arcamm.uploaded=1 ORDER BY uploaded_arcamm.date_uploaded DESC";
```

Figura 4.33: Consulta para obtener vídeos publicados

4.2.2.3 Presentaciones programadas

En esta tercera pestaña presentamos al usuario todas las presentaciones programadas que van a ser realizadas en un futuro y que pueden seguirse en directo.

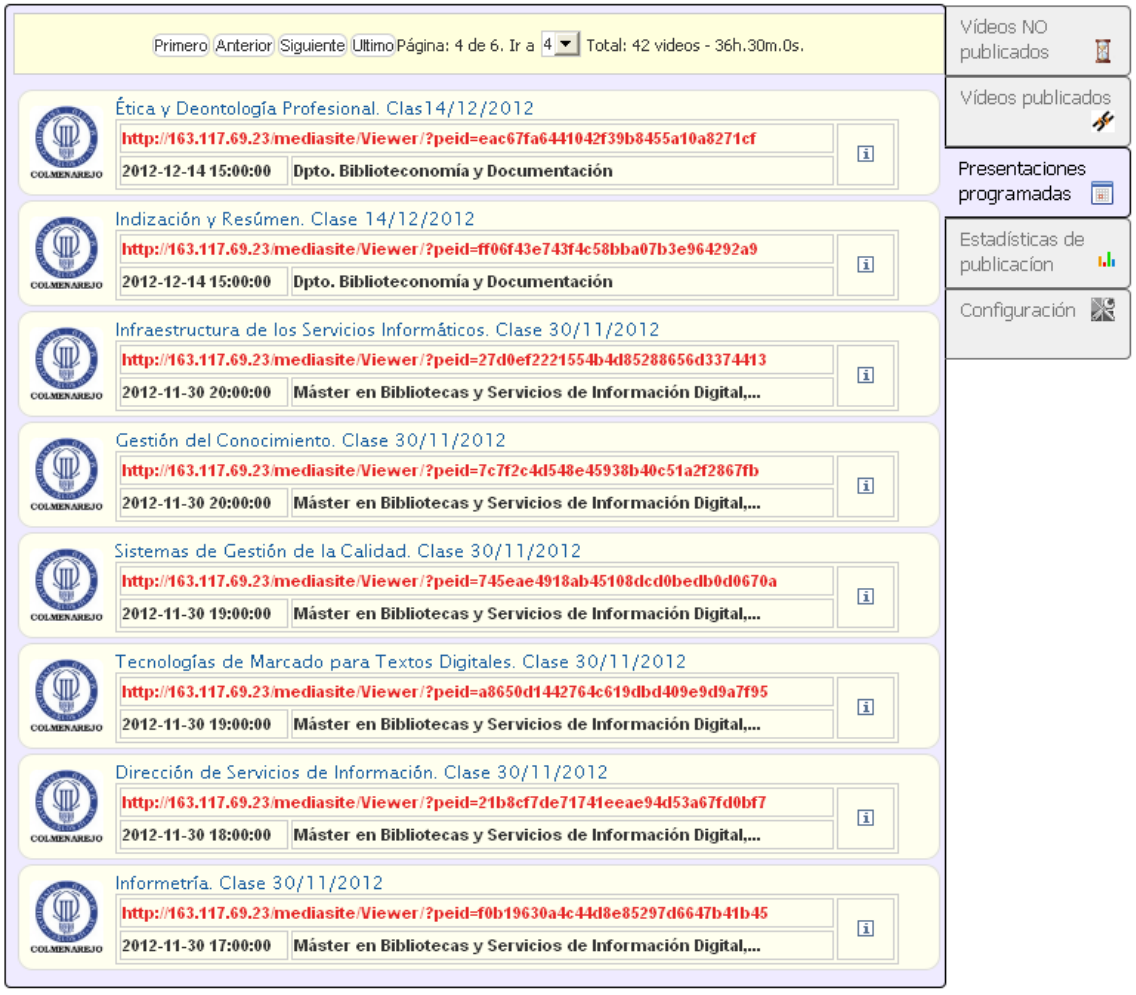


Figura 4.34: Presentaciones programadas

Como podemos observar el diseño es semejante al de las secciones anteriores, aunque difiere en algunos aspectos.

En primer lugar existen unos cambios perceptibles por el usuario como son el cambio de información que se le proporciona. Observamos que mantenemos la imagen con el nombre del campus donde se realizará la grabación, así como el título, la fecha de realización de la grabación o los autores. También observamos que se mantiene el botón de "información adicional" ya que al crear la presentación programada, la persona encargada de ello, rellena toda la información referente al vídeo. Su funcionamiento es exactamente igual al que se hacía en la pestaña "Vídeos no publicados", con la diferencia de que si se accede desde una presentación programada, no parece el botón de publicación.

Otra diferencia que puede observar el usuario es que en esta vista no se muestra ni la duración ni las diapositivas que tiene la presentación. El motivo es bien sencillo, ya que al tratarse de presentaciones futuras, los valores de estos dos campos son cero, hasta que no se realice la grabación y por tanto el estado de la presentación cambie a *Uploaded* en la base de datos de *Mediasite* y en la base de datos que hemos creado.

La diferencia más sustancial que presenta esta sección, es que se proporciona al usuario el link donde el vídeo se podrá ver en directo.

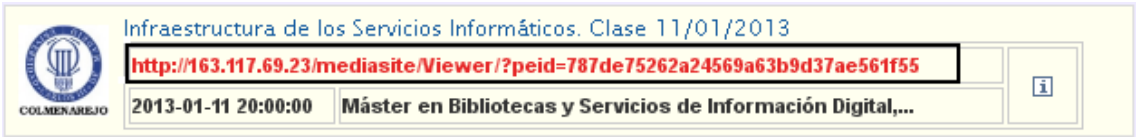


Figura 4.35: Secciones del ítem presentación programada

En un primer momento, cuando se observó de que existía la posibilidad de poder ver una clase en directo, se pensó en añadir esta información a la agenda de actos del Área de Audiovisuales de forma automática, con lo que facilitaríamos el acceso a esta información tanto a los usuarios externos como al personal del Área. Después de examinar los cambios que se debían realizar tanto en nuestro trabajo, como en el que había sido realizado en la confección de la nueva agenda, se desechó realizar este trabajo, al menos por el momento. El motivo de esta decisión fue que el trabajo a realizar era bastante mayor al previsto y al no ser uno de los requisitos iniciales se pensó que lo principal era centrarse en el objetivo inicial de este trabajo que era publicar vídeos realizados con *Mediasite* en el portal de vídeos de la Universidad Carlos III. A pesar de esto, no se quiso omitir esta información y decidió añadir esta sección en la que el usuario podría copiar el link y así poder añadirlo a correos electrónicos con profesores, alumnos o a la agenda de actos.

Por último debíamos de modificar la consulta que debíamos de realizar a la base de datos para mostrar las grabaciones futuras. En un primer momento pensamos que solamente era necesario cambiar el valor del campo *status* a *Scheduled*, pero observamos que aparecían vídeos pasados a la fecha actual, que por el motivo que fuera, no habían sido grabados, con lo que decidimos acotar la búsqueda por defecto a un valor siempre mayor a la fecha actual, aunque permitíamos a través del módulo de búsqueda poner una fecha pasada para tareas de limpieza o cualquier otro motivo.

```
$sql ="SELECT * FROM presentations LEFT JOIN uploaded_arcamm ON
          presentations.id=uploaded_arcamm.idPresentation
         WHERE uploaded_arcamm.uploaded=0 AND
          presentations.status='Scheduled' AND airDateTime BETWEEN '".date('Y-
          m-d')."' AND '2999-12-31' ORDER BY presentations.airDateTime DESC";
```

Figura 4.36: Consulta para obtener presentaciones programadas

4.2.2.4 Estadísticas de publicación

Uno de los requisitos principales que se nos exigía era la posibilidad de ofrecer al usuario estadísticas de publicación y que este las pudiera generar a su gusto dependiendo de la información que quisiera obtener.

Teniendo en cuenta que toda la fase de realización de la aplicación la hemos realizado con código libre, debíamos de buscar una herramienta que cumpliera con estos requisitos. Además debía de poseer una librería que nos permitiera integrarla en PHP. Teniendo presentes estas dos premisas, encontramos *Open Flash Chart* que encajaba perfectamente en nuestros requisitos. Además nos proporcionaba una interfaz bastante vistosa para el usuario y en principio tenía un amplio abanico de ejemplos que nos facilitarían mucho el trabajo [18].

En la documentación de la herramienta observamos ejemplos de cómo conformar las gráficas una vez que tenemos una colección de datos a nuestra disposición, pero en nuestro caso debemos obtener esos datos de forma dinámica y siempre cumpliendo los requerimientos del usuario, con lo que los datos a mostrar no siempre serán los mismos. Por lo tanto, se opta por ofrecer al usuario un formulario que deberá rellenar siempre que quiera generar una estadística.

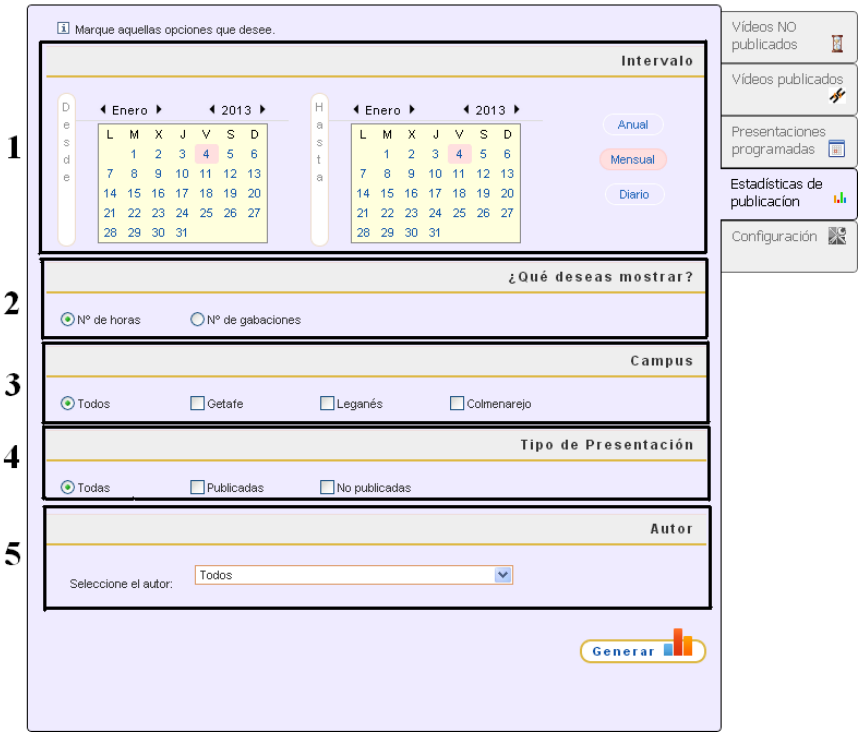


Figura 4.37: Formulario de estadísticas

Como podemos observar, tenemos cinco tipos de campos en lo que el usuario puede interactuar. A continuación vamos a identificar y explicar su función en la realización de la gráfica.

1. **Intervalo.** En esta sección el usuario debe de elegir el intervalo temporal de los datos que desea mostrar. Se le proporcionan dos calendarios, uno para la fecha inicial y otro para la fecha final. Para realizarlos hemos utilizado los mismos que utilizamos en el módulo de búsqueda, con la diferencia de que no emergen al presionar un icono, sino que siempre son visibles. Por defecto aparecen con la fecha actual en ambos, aunque el usuario puede elegir el rango que deseé, con la única condición de que la fecha inicial sea igual o menor a la fecha final. En el caso de que esto no se cumpla emergerá una pantalla de alerta que recuerda dicha condición y no permite realizar la consulta.

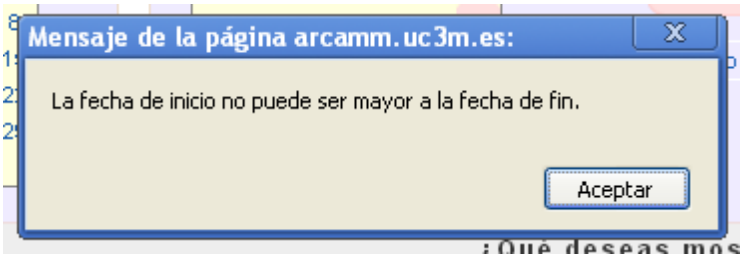


Figura 4.38: Ventana de alerta por error en fechas

Tanto la ventana emergente ,como la navegación a través de los meses y los años están controladas con eventos *JavaScript*, mientras que la inteligencia para pintar el calendario de forma correcta lo gestiona el fichero *calendar.php.*

Otra opción referente a la vista en la que se presentarán los datos, es la de seleccionar el rango en el que dividiremos el eje x de la gráfica, ya que esta división puede ser por días, meses o por años. Creemos que esta opción es de suma utilidad, ya que permite al usuario poder comparar el uso de *Mediasite* en distintos tramos de tiempo, observar picos de uso o rangos temporales en los que la aplicación es menos utilizada. Esto además ayudó al equipo de trabajo en la optimización del tiempo de ejecución del script de actualización.

2. **¿Qué deseas mostrar?** En esta sección el usuario puede elegir entre mostrar la cantidad de horas grabadas o bien mostrar la cantidad de grabaciones realizadas en un determinado periodo de tiempo. El motivo de ofrecer estas dos opciones, excluyentes entre ellas, es por decisión del Área de Audiovisuales como usuarios finales de la aplicación, ya que son los datos que les interesa recabar del uso de *Mediasite.* La cantidad de horas o grabaciones, determinará la altura de la gráfica en un determinado punto temporal.

3. **Campus**. En la siguiente sección se permite al usuario elegir la cantidad de gráficas que desea mostrar de forma simultánea, correspondiéndose cada una de ellas con cada uno de los campus de la universidad. Como mínimo debe de aparecer una gráfica que se puede corresponder con todos los campus o bien con uno en concreto, siendo el máximo de tres, aunque esto no es así realmente tal y como explicaremos en la siguiente sección.

   Como ocurría en otras secciones de la interfaz, para mantener el control de manera propia por parte de la aplicación, se hace uso de *JavaScript*. Para ello creamos dos funciones: una que controle cuando es pulsado uno de los *checkbox*, poniendo el atributo *checked* del *radio button* con valor falso y otra que haga exactamente lo contrario, que cuando el *radio button* sea pulsado ponga el atributo *checked* de todos los *checkbox* con valor falso. Con esto evitamos que el usuario pueda elegir todos los campus y además un campus en concreto, mientras que si le permitimos elegir todos los campus, pero de forma independiente, es decir cada campus representado con una gráfica distinta.

4. **Tipo de presentación.** Esta sección guarda muchas similitudes con la anterior, con la diferencia que en este caso pude elegir entre todas las presentaciones sin distinción o bien diferenciando las que han sido publicadas de las que no lo han sido. Como ocurría en el caso anterior el número de graficas debe ser como mínimo de una y como máximo de dos. En realidad la aplicación puede mostrar hasta un total de seis gráficas distintas, ya que si son pulsados los tres campus y los dos tipos de presentación mostrará dos gráficas por campus una para los vídeos publicados y otra para los no publicados. El número de gráficas que se van a mostrar se puede obtener de esta sencilla formula:

   (Nº de campus) X (Tipos de presentación)

   En el caso de que en cualquiera de las dos secciones se seleccionaran "Todos" o "Todas" en la fórmula contarían como uno. Para mantener el control de la selección del usuario, se ha hecho uso de dos funciones exactamente iguales que las utilizadas en la sección anterior, pero con el nombre distinto para que no fueran confundidas con las anteriores.

5. **Autor.** En esta última sección permitimos al usuario a través de una lista desplegable elegir entre los distintos autores de grabaciones que existen en la base de datos. Esta lista se crea de forma dinámica cada vez que la página es refrescada, ya que para crear las distintas opciones, en este caso autores, necesita realizar una consulta a la base de datos, donde obtendrá todos los autores de la tabla *presenters*. En este caso solamente permitimos mostrar los datos de un determinado autor o de todos como conjunto. El motivo de no permitir la comparación simultánea de varios autores es que si por cada autor se muestran seis gráficas como máximo, si elegimos más de un autor, el número de gráficas a mostrar sería muy elevado y en muchos casos serían ilegibles para el propio usuario.

Cuando el usuario ha rellenado el formulario como desea, al pulsar sobre el botón "generar", los datos del formulario se envían mediante el método POST al mismo *php, stats.php,* con el valor de la variable *tab=stats,* para que la aplicación se posicione en esa sección después del refresco.

Para que no vuelva a aparecer el formulario, se comprueba si se reciben variables a través de POST. Cuando esto ocurre se realiza un *include* del fichero *graph.php*. En caso contrario, que es por ejemplo cuando se accede por primera vez a la sección, no se carga este fichero y si el código correspondiente al formulario.

La primera acción que se realiza en el fichero *graph.php,* es guardar en distintas variables los valores que nos llegan del formulario anterior. El primer objetivo es conformar la consulta que se hará a la base de datos dependiendo del valor que tenga la variable *subInterval,* que es la que marcará el intervalo temporal en el que pintaremos el eje x. Dependiendo de este valor el código que se ejecutará será uno u otro.

La diferencia principal entre estos tres códigos es la manera de generar el array, que por un lado será cada una de las fechas o intervalo de fechas en la que se buscará en la base de datos y por otro la leyenda del eje x de la gráfica a mostrar.

- En el caso de que el intervalo elegido sean días, se hace uso de la función *array_dias,* que recibe como parámetros la fecha de inicio y la de fin en la que se desea generar la gráfica. En esta función creamos todos los días que están incluidos en ese intervalo. Para ello pasamos tanto la fecha inicial como la final al formato UNIX (el número de segundos desde el 1 de Enero del 1970 00:00:00 UTC) con la función *strtotime*. Utilizamos esta función ya que así nos olvidamos de estar pendientes del número de días que tiene un mes o si el año es bisiesto. Realizamos un bucle en el que en cada interacción el valor de la fecha se incremente en 8640 (número de segundos que tiene un día). Dentro de cada interacción pasamos el valor de la fecha del formato Unix, al formato "Y-m-d" utilizando la función *date*. Cuando todas las iteraciones han acabado, la función devuelve un array con todas las fechas en el formato adecuado.

- En el caso de que el intervalo elegido sean meses, se hace uso de la función *array_meses*, que también recibe como parámetros la fecha inicial y la final. En primer lugar extraemos de las dos fechas el número de mes y el número de año para realizar una serie de operaciones para calcular el número de meses, y por lo tanto el número de divisiones que vamos a hacer del eje x. Cuando hemos calculado el número de meses que hay entre las dos fechas introducidas por el usuario guardamos en un array el número de mes seguido del año para diferenciar meses iguales de distintos años.

- Por último, si el intervalo seleccionado es por año, se hace uso de la función *array_anos,* que recibe como parámetros, como en los casos anteriores, la fecha inicial y la final. En esta función lo único que hacemos es extraer el año de cada una de las fechas con la ayuda de un bucle generamos todos los años comprendidos entre las dos fechas, que mínimo siempre será de uno.

A partir de este punto el proceso es muy similar independientemente del intervalo elegido.

Cuando tenemos conformado el intervalo temporal, debemos de realizar una consulta por cada una de las divisiones en las que esté dividido el eje x. Para ello nos ayudamos de tres bucles anidados, en el que el más interno se encarga de generar la consulta para cada intervalo temporal de cada una de las gráficas, el intermedio itera tantas veces como tipos de presentación hay y el más externo itera dependiendo de los campus seleccionados por el usuario.

A lo largo de este proceso se va creando un array bidimensional, en el que la primera dimensión se corresponde con el número de grafica y la segunda con el dato correspondiente con cada intervalo temporal, con lo que al final tendremos guardada la altura de cada uno de los puntos de cada una de las gráficas. Además durante ente proceso creamos otro array al que llamaremos "leyenda", que será el nombre de cada una de las gráficas generadas. Siempre tendrán la siguiente forma:

Leyenda = (Campus)_(Tipo de Presentación)

Además como vamos a realizar una gráfica con forma de queso, guardamos los totales de cada uno de las gráficas generadas en un array al que llamaremos *total*. En esta gráfica presentaremos el porcentaje de cada uno de los resultados de cada gráfica respecto del total.

Para finalizar toda la extracción de datos, guardaremos en otro array, al que llamaremos "eje_x", todos los intervalos temporales en la que está dividida la gráfica. En el caso de los días y de los años, el array es el mismo que nos devuelven las funciones *array_dias* y *array_anos* respectivamente, pero en el caso de los meses, realizamos una pequeña modificación donde cambiamos el número de mes por una abreviatura de tres letras del nombre de dicho mes. Para ello creamos una función que se llama *calcula_mes* y donde se contemplan los doce posibles casos.

Una vez tenemos todos los datos almacenados, el siguiente paso del proceso es generar las graficas correspondientes. Para ello creamos dos ficheros: *graph_chart.php* para crear la gráfica por intervalos de tiempo y *graph_chart_totales.php* para la gráfica de los datos totales.

- **Graph_chart.** Con este fichero crearemos las gráficas de la evolución temporal de un determinado dato, bien horas grabadas o número de grabaciones en un determinado periodo de tiempo.

  Para ello debemos de hacer un *include* del fichero del *open-flash-chart.php* que nos proporciona la propia herramienta *Open Flash Chart.* Con este archivo vamos a crear el *JSON* correspondiente a la temporal. A través de este fichero, podemos crear un objeto de la clase *open flash chart,* sobre el realizaremos todas las operaciones y configuración de la gráfica. En el caso del eje x debemos crear un objeto de la clase *x_axis* y otro objeto de la clase *x_axis_labels.* Sobre este último, con *set_labels* introducimos la leyenda del eje x por completo. Cuando esto ocurre añadimos esto al objeto x_axis y este a su vez al objeto open *flash chart.*

Para crear cada una de las líneas, debemos de recorrer la primera dimensión del array que se correspondía con el "eje y" y añadir estos datos a un objeto *line* que será añadido al objeto OFC.

Finalmente dependiendo de los valores de las gráficas, se conforma la altura que tendrá el "eje y" y su correspondiente leyenda. Cuando todos estos datos están incluidos dentro del objeto OFC, se pasa toda esa información a una cadena de texto, que es el que se va a escribir en el fichero JSON encargado de pintar la gráfica.

- ***Graph_chart_totales.*** Con este fichero crearemos la gráfica de queso en la que podremos observar el porcentaje de distintas partes sobre un total. Estas gráficas son muy visuales por que podemos observar cuanto porcentaje del total están publicadas o cuantas han sido grabadas en un determinado campus.

  Como en el caso anterior debemos hacer el *include* del fichero *open-flash-chart.php*, donde se encuentran las clases necesarias para poder crear el objeto que posteriormente creara el archivo JSON. En este caso debemos crear un objeto de la clase *pie* y otro de la clase *pie_values.* Sobre este último, a través de la función *set_values* introduciremos el valor del array *total* que creamos anteriormente y su correspondiente leyenda. Se crearan tantos objetos *pie_values* como gráficas haya que realizar, con un máximo de seis.

  Finalmente, debemos añadir el objeto *pie* al objeto de la clase OFC, que será el que pasaremos a una cadena de texto para su posterior escritura en el archivo JSON que creará la gráfica.

Una vez tenemos estos dos ficheros solo nos quedaría añadir la película flash a nuestra página utilizando este código para cada una de los gráficos que vamos a presentar:

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#v
ersion=8,0,0,0" width="570" height="315" id="grafica1" align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="mediasite/estadisticas/chart/open-flash-chart.swf" />
<param name="quality" value="high" />
<param value="transparent" name="wmode"/>
<param name="FlashVars" value="data-file=mediasite/estadisticas/parse.php" />
<embed src="mediasite/estadisticas/chart/open-flash-chart.swf" FlashVars="data-
file=mediasite/estadisticas/parse.php" wmode="transparent" quality="high"
bgcolor="#FFFFFF" width="570" height="315" name="chart" align="middle"
allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
pluginspage="http://www.macromedia.com/go/getflashplayer" />
</object>
```

Figura 4.39: Código para embeber gráfica temporal

En el caso del gráfico de queso, solo debemos cambiar el valor de *file* a la ruta donde se encuentra el archivo JSON donde se encuentra correspondiente a ese gráfico.

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#v
ersion=8,0,0,0" width="570" height="315" id="grafica1" align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="mediasite/estadisticas/chart/open-flash-chart.swf" />
<param name="quality" value="high" />
<param value="transparent" name="wmode"/>
<param name="FlashVars" value="data-
file=mediasite/estadisticas/parse_totales.php" />
<embed src="mediasite/estadisticas/chart/open-flash-chart.swf" FlashVars="data-
file=mediasite/estadisticas/parse_totales.php" wmode="transparent" quality="high"
bgcolor="#FFFFFF" width="570" height="315" name="chart" align="middle"
allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
pluginspage="http://www.macromedia.com/go/getflashplayer" />
</object>
```

Figura 4.40: Código para embeber gráfica de queso

El aspecto que tendría cada una de las gráficas sería la siguiente:

- Gráfica temporal

  Como podemos observar aparecen en esta gráfica la evolución de todas las grabaciones realizadas en los distintos campus de la universidad. En este caso como hemos seleccionado todas las grabaciones no se nos ha desglosado en publicadas y no publicadas. Como dijimos con anterioridad estas gráficas nos ayudaron a ver los picos de uso de la herramienta *Mediasite*, así como el uso que se daba de ella en cada uno de los campus de la Universidad Carlos III.

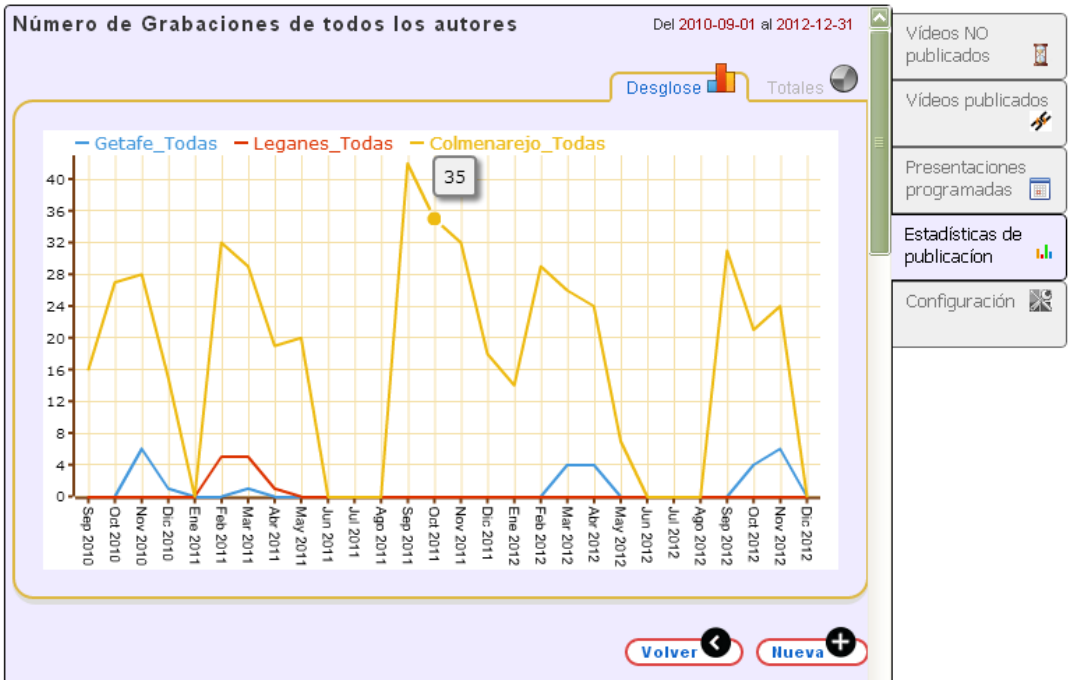


Figura 4.41: Gráfica temporal

- Gráfica de totales

  Si pulsamos sobre las pestaña *Totales*, el usuario podrá ver el porcentaje de uso en cada uno de los campus en el periodo de tiempo que él haya seleccionado.

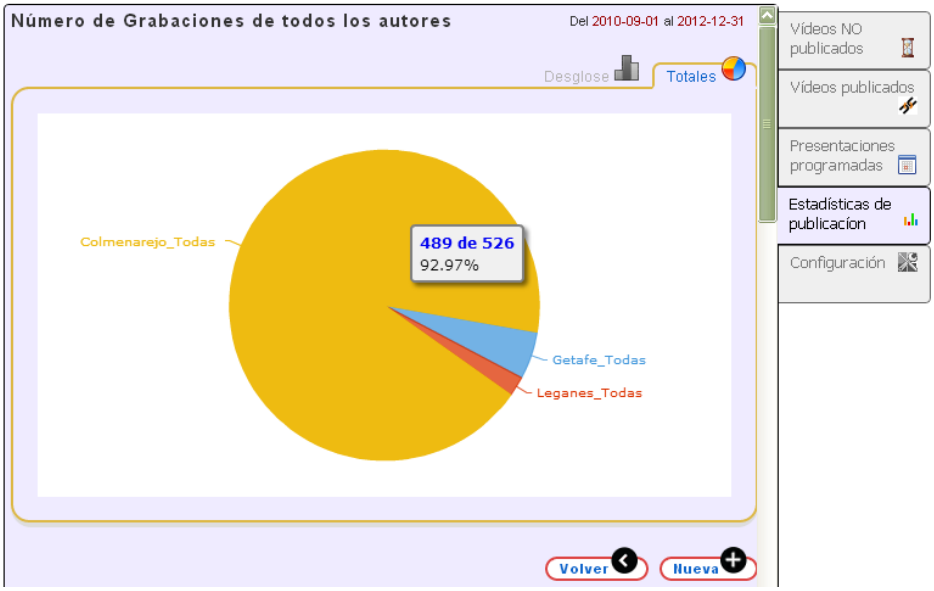
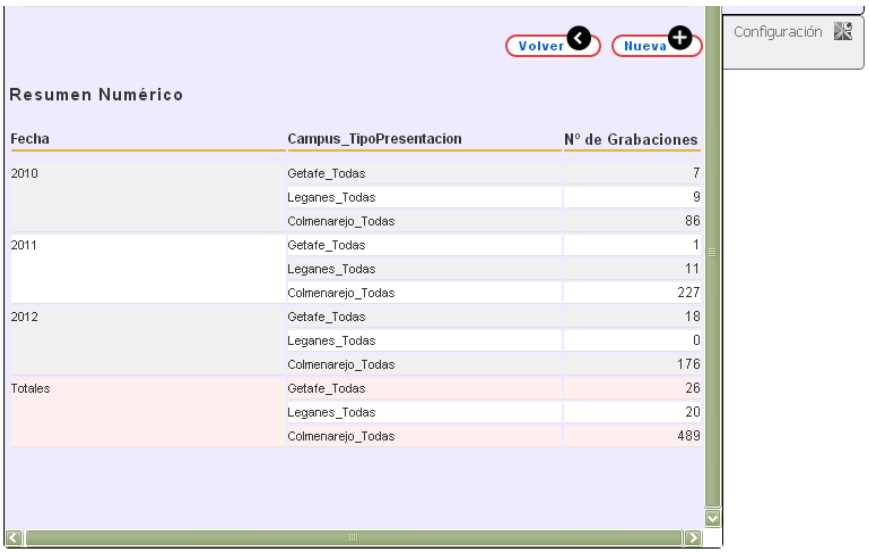


Figura 4.42: Gráfica de totales

Además de mostrar las gráficas, permitimos al usuario realizar una nueva búsqueda, o bien manteniendo los datos de la búsqueda anterior o bien realizando una gráfica totalmente independiente.

Finalmente se muestra un cuadro resumen con el desglose del total de las grabaciones según los criterios introducidos por el usuario. Únicamente añadimos una tabla que se generaría de forma dinámica y que su tamaño dependería de los datos obtenidos.



Fecha	Campus_TipoPresentacion	Nº de Grabaciones
2010	Getafe_Todas	7
	Leganes_Todas	9
	Colmenarejo_Todas	86
2011	Getafe_Todas	1
	Leganes_Todas	11
	Colmenarejo_Todas	227
2012	Getafe_Todas	18
	Leganes_Todas	0
	Colmenarejo_Todas	176
Totales	Getafe_Todas	26
	Leganes_Todas	20
	Colmenarejo_Todas	489

Figura 4.43: Resumen numérico

4.2.2.5 Configuración

En esta última sección decidimos crear una interfaz donde el usuario pudiera cambiar los parámetros de conexión con la base de datos de *Mediasite* en previsión de posibles cambios de servidor, contraseña o usuario.

En principio esto no era un requisito exigido por el cliente, pero nos parecía que podía hacer más robusta nuestra aplicación, y al proponerlo al Área de audiovisuales les pareció perfecto si nuestro equipo de trabajo estaba dispuesto. Estudiamos el tiempo que nos podía suponer, y llegamos a la conclusión de que no resultaría muy costoso.

Nuestro diseño consistiría en un formulario en el que se presentarían los campos que se podrían modificar además del valor actual de dicho campo. Aquí nos surgía una duda en referencia a la seguridad ya que la contraseña podría ser cambiada por cualquier persona que fuera ajena a la aplicación, pero el personal encargado del portal de vídeos, nos aseguró, que nuestra aplicación al estar incluida dentro del Portal, no podía ser accesible por ninguna persona que no tuviera permisos por parte del administrador mediante login y contraseña, por lo que quien pudiera acceder al rol *mediasite* previamente debía de haber sido dado de alta por el administrador de sistema.

En el primer acceso a esta sección el usuario se encuentra con los valores actuales de los respectivos campos que se pueden modificar:

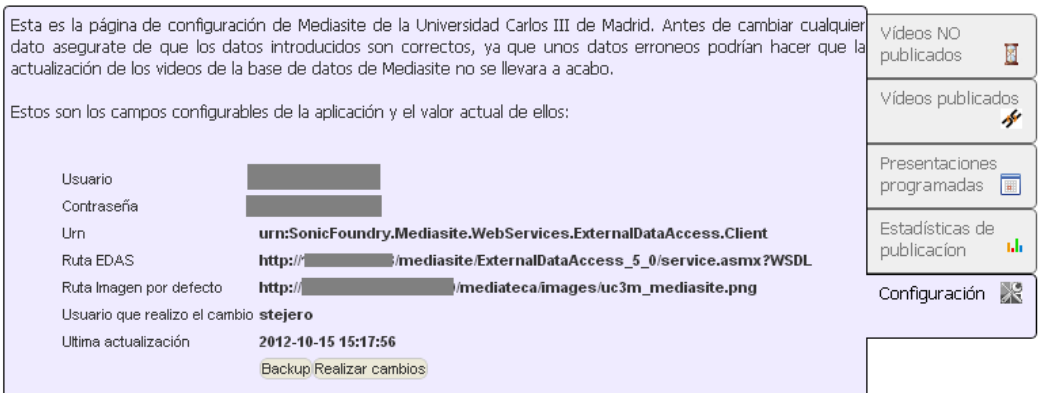


Figura 4.44: Configuración

El usuario puede realizar dos acciones: realizar los cambios que considere oportunos o bien cargar la configuración que se guardó en el backup que es la configuración anterior a la actual.

Si se decide cargar la configuración guardada en el backup, al pulsar sobre el botón aparecerían los valores antiguos de esos campos y un nuevo botón para verificar los cambios. La técnica que utilizamos para ocultar o mostrar el backup es muy similar a la que realizamos para mostrar y ocultar las distintas secciones de la aplicación, que es modificando el atributo *display* del estilo. En el caso de que se cargará la configuración antigua esta pasaba a ser la principal siendo sus datos los que utilizaría el script de actualización para acceder a la base de datos de *Mediasite*, mientras que la configuración anterior pasaría a ser la que se alojara en backup. Esta opción la consideramos muy útil para el caso de introducir una configuración errónea y poder volver a la anterior que sabemos que funcionaba correctamente.

En el caso de que se decida modificar cualquiera de los campos aparecería una pantalla como la siguiente:

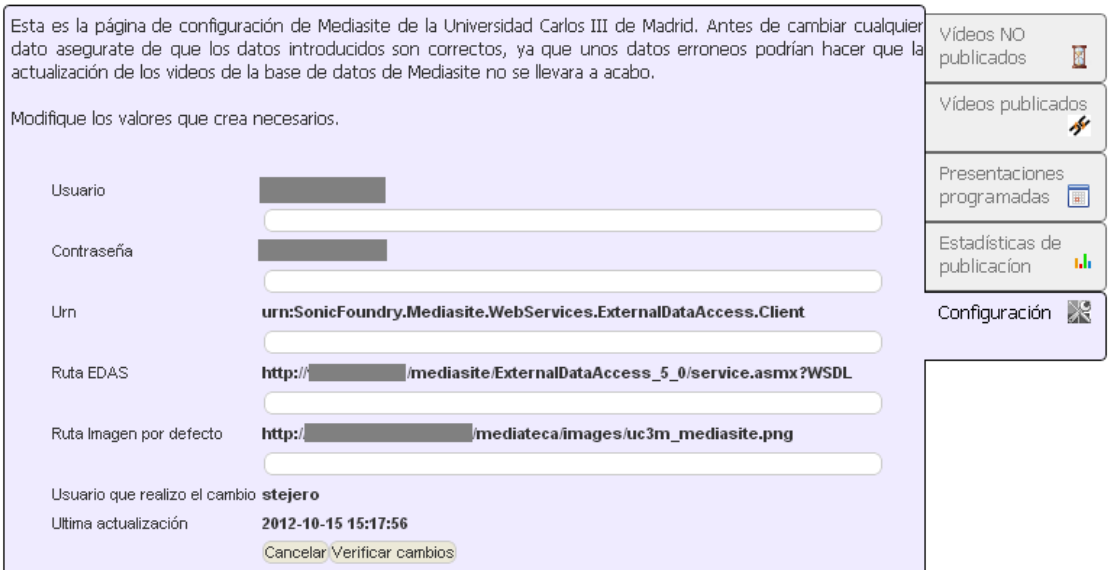


Figura 4.45: Modificar configuración

Como vemos aparecen una serie de cuadros de texto en el que se permite cambiar un mínimo de uno de ellos no siendo necesarios cambiarlos todos. En el caso de que no se modifique ninguno de ellos, al intentar verificar los cambios, aparece un cuadro de dialogo en el que se puede leer el mensaje "No has modificado ningún campo" no realizándose verificación alguna. Para evitar que el usuario tuviera que modificar todos los campos para que la modificación fuera posible, en el caso de que algún cuadro de texto estuviera vacio, tal y como parecen en la imagen, nuestro sistema reconocería que ese campo no debe de ser modificado y por tanto mantendría el valor actual de dicho campo.

Finalmente, después de modificar el campo que se considere oportuno, aparece una pantalla de verificación con los datos que se van a cargar como principales. En el caso de confirmar los cambios, estos datos irán acompañados con la fecha de dicha modificación y con el nombre de usuario de la persona que lo realizó. Así el administrador del sistema puede saber quien ha sido la última persona que ha realizado dichos cambios y pedir explicaciones por cambios de configuración que no estuvieran previstos.

4.2.3 PUBLICACIÓN EN ARCAMM

En este último apartado explicaremos las modificaciones que fueron necesarias en el portal ArcaMM para que nuestros vídeos pudieran ser publicados en él, sin que esto no fuera un problema para el funcionamiento del resto de herramientas que ya se utilizaban para publicar otro tipo de vídeos generados de distinta forma.

Como ya explicamos en el apartado 4.2.2.1, es esta sección es donde permitimos al usuario poder publicar en el repositorio de vídeos de la UC3M. El usuario después de elegir la presentación deseada deberá pulsar sobre el botón "publicar" que lo dirigirá al fichero *manager.php* encargado de la publicación.

Para que esto pueda ocurrir, el usuario tiene que tener asumido el rol *manager*, asignado por el administrador del sistema. Si esto no fuera así la misma aplicación expulsaría al usuario devolviéndole a la página de autentificación.

En primer lugar nos tuvimos que reunir con el equipo de desarrollo del portal ArcaMM para que nos indicará que ficheros debíamos modificar y nos proporcionara una máquina virtual para realizar pruebas sin poner en peligro el buen funcionamiento del portal de vídeos. En principio solo debíamos modificar el archivo *manager.php* y se nos recomendó realizar una nueva condición para poder tratar nuestros vídeos de forma independiente y así no arriesgarnos a modificar cualquier otra parte y que esta dejara de funcionar.

Cuando nos enfrentamos por primera vez al código del fichero, nos dimos cuenta que el fichero que debíamos modificar era *additem.php,* ya que en el fichero manager solo existían *includes* de los módulos que conformaban la estructura de toda la página. Al ponernos a trabajar con este último fichero nos fijamos que existían distintos casos dependiendo si el vídeo a tratar era uno que ya se encontraba publicado ArcaMM, se publicaba desde un vídeo guardado en *Metadata (*nube del personal de Audiovisuales donde almacena los vídeos que realizan en las distintas salas*)* o era una presentación realizada con la herramienta *Opencast* . En definitiva, debíamos realizar otro *elseif* dentro de ese fichero donde trataríamos nuestros vídeos.

Una de las funciones que queríamos ofrecer al usuario es que toda la información posible referente a las presentaciones fuera añadida de forma automática, aunque siempre podría ser modificada antes de publicar el vídeo si algún dato no era correcto o simplemente se prefería otro diferente.

Esta idea fue ofrecida al personal encargado del portal de vídeos y nos comentaron que esa funcionalidad ya existía y por lo tanto estaba concebida en el código, ya que cuando un vídeo ya publicado iba a ser modificado, se obtenía toda la información de la publicación y se mostraba una pantalla similar a la de publicación, pero con todos los datos referentes rellanados. La manera de hacer esto es con código *JavaScript* que comprueba si el valor de unas determinadas variables son nulas o no. En el caso de que sean nulas no rellena dicho campo, mientas que si este tiene algún valor, rellena el campo correspondiente.

Cuando se nos presentó esta posibilidad, fue un alivio para el grupo de trabajo, ya que facilitaba mucho nuestro fin. Lo único que necesitábamos era dar un valor a todas las variables posibles y el resto lo haría la propia tecnología utilizada para la modificación de publicaciones. Por lo tanto recabamos del código *JavaScript* que valores eran comprobados y que nombre tenían para nosotros pasárselos con el mismo nombre.

Para que esto no interfiriera con el resto de funcionalidades existentes, creamos una nueva condición, en la que si existía la variable *idMediasite*, el vídeo que llegaba para ser publicado se trataba de uno generado con *Mediasite* y por tanto iba a ser tratado de manera distinta al resto. Al tratarse de un estructura *if*, si el vídeo cumplía una determinada condición, se trataba de la forma correspondiente impuesta por el programador y no podía haber errores, dado el carácter excluyente de la estructura.

Para ello nosotros siempre debíamos añadir a la url la variable *idMediasite* con su valor correspondiente. Además de esto, se nos exigía por motivos de seguridad, que la variable mod tuviera el valor de *manager* y la variable amod el de *additem*. Esto era necesario ya que al publicar un vídeo, nuestro rol debía de cambiar de *mediasite* a *manager*. En el caso de que no tuviéramos permisos para usar el rol *manager*, la misma aplicación nos expulsaría y nos dirigiría a la página de autentificación. Toda esta información la añadiríamos a la hora de conformar el link al que apuntaría el botón "publicar" en la parte reconocida anteriormente como "Vídeos no publicados".

Una vez que estamos inmersos en el código que hemos generado, el valor que toman las distintas variables dependerá del valor guardado en nuestra base de datos, ya que con el valor de la variable *idMediasite* realizamos una consulta a la base de datos propia y obtenemos toda la información referente al vídeo. De toda la información que el portal nos obliga a introducir, solo dejamos vacío la categoría en la que se englobaría el vídeo, la serie en la que queremos introducirlo y la privacidad y difusión que nuestro vídeo tendrá. Como vemos estos son datos que a priori no se pueden saber, y que dependen única y exclusivamente de la publicación, con lo que nos sería imposible proporcionarlos rellenados.

En cambio todo lo que se refiere a información general (titulo, descripción, fecha de publicación, imagen, autor/es) y información técnica del vídeo (mime, duración, resolución, calidad) lo rellenamos de forma automática y ahorramos al usuario tiempo en añadir esa información de forma manual.

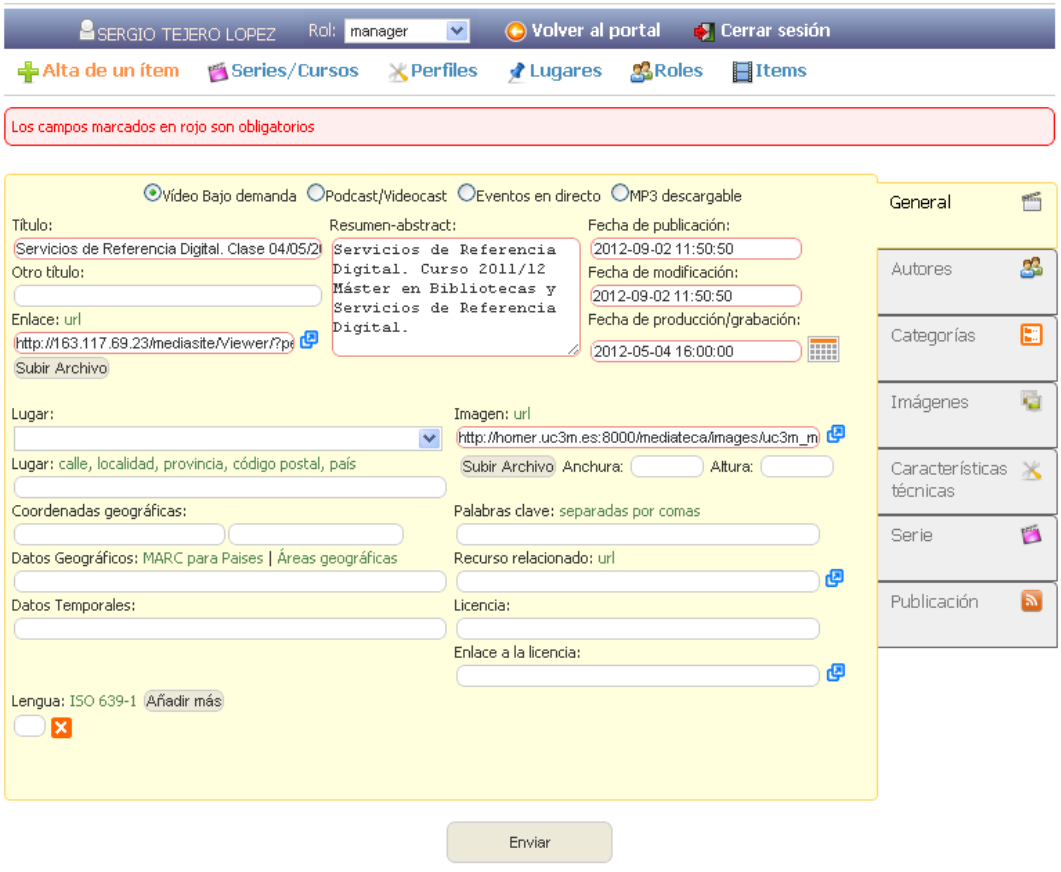


Figura 4.46: Rellenar campos de publicación

Una vez que el usuario crea que todos los campos están rellenados de la forma correcta podrá publicar el vídeo en el portal pulsando sobre el botón Enviar. En el caso que algún campo obligatorio no esté rellenado, la misma página informará al usuario que ese vídeo no será publicado hasta que esos campos no dejen de estar vacios.

A partir del momento en que se pulsa el botón enviar, el vídeo deja de estar al alcance de nuestra aplicación, ya que toda la gestión la lleva ahora el portal de vídeos y toda la información referente al vídeo ha sido almacenada en su base de datos. Como sabíamos que esto debía ser así, ya que así era el funcionamiento de la arquitectura del portal de vídeos, no queríamos perder todas las referencias acerca de nuestros vídeos publicados, ya que como recordaremos de apartados anteriores, nosotros queríamos llevar un control de cuáles de nuestros vídeos habían sido publicados y cuáles no. Para ello necesitábamos que ArcaMM nos informara de cuando un vídeo publicado con ayuda de nuestra aplicación, era añadido al portal de vídeos y de cuando era borrado para así nosotros controlar en todo momento cual era el uso de nuestras presentaciones.

Para el caso de añadido de contenido dentro del portal, en el momento que el usuario pulsa el botón enviar, la herramienta se traslada al fichero *ProcessingForm.php*, donde se comprueba que los campos obligatorios no están vacios. Si esto ocurre el nuevo *ítem* es introducido dentro de la base de datos del portal de vídeos. En el caso de que la inserción se realice de forma correcta, queremos que en ese preciso instante se nos informe para poder agregar a nuestra base de datos que esa presentación ha sido publicada. Para ello añadimos un código en ese mismo fichero en el que comprueba que si la inserción se hizo satisfactoriamente y el mime es igual a *video/mediasite*, realice una actualización de la base de datos que hemos creado en nuestra aplicación, en concreto en la tabla *uploaded_arcamm*, donde a través de esta sentencia indicaremos que la presentación ha sido publicada, en qué fecha ha ocurrido y quien han sido la persona que lo ha realizado.

```
$sqlmedia="UPDATE uploaded_arcamm SET uploaded ='1',
          url_arcamm='".$url_arcamm."',date_uploaded='".$date_uploaded."',
         published_by='".$owner."' WHEREidPresentation='".$id_mediasite."'";
```

Figura 4.47: Consulta para añadir vídeo publicado

Cuando esto ocurre, en nuestra aplicación, se mostrará el vídeo publicado en la pestaña “Vídeos publicados”.

Para el caso de borrado de contenido, seguiríamos el mismo procedimiento, cuando un ítem es borrado del portal de vídeos necesitamos cambiar el campo de la tabla de *uploaded_arcamm* de 1 a 0, ya que con esto indicamos que el vídeo pasa de nuevo a ser un vídeo no publicado y volverá a aparecer en la sección “Vídeos no publicados”. Para ello debemos de modificar el fichero *delete.php* donde comprobamos que se si ha realizado de forma correcta el borrado del ítem del portal de vídeos, realizamos la siguiente actualización en nuestra base de datos.

```
$sql="UPDATE uploaded_arcamm INNER JOIN presentations ON
          uploaded_arcamm.idPresentation=presentations.id SET
          uploaded_arcamm.uploaded='0', uploaded_arcamm.url_arcamm='',
          uploaded_arcamm.date_uploaded=NULL,
          uploaded_arcamm.published_by=NULL WHERE
          presentations.url='".$row['link']."';";
```

Figura 4.48: Consulta para borrar un vídeo

Con esta consulta cambiamos el campo que indica si el vídeo esta publicado o no y además ponemos el resto de campos vacios. Para ello nos ayudamos del link de acceso al vídeo en ArcaMM, ya que lo tenemos almacenado en el momento que se realizó la inserción en el portal. Con estos cambios, nuestra aplicación queda incluida en el portal de vídeos de la universidad y puede convivir con el resto de herramientas sin causar ningún tipo de problema.

Por último debíamos de realizar una modificación en el visor de vídeos, ya que el portal, realiza un embebido del reproductor y con los vídeos creados con Mediasite no nos era posible realizar esta acción. El motivo es que el mismo sistema tenía su propia interfaz en la que embebía la señal de vídeo con la señal de ordenador permitiendo al usuario navegar a través de las distintas diapositivas creadas durante la grabación.

Figura 4.49: Visor de ArcaMM

Para solucionar este aspecto se decidió crear una imagen fija que se encuadraría en el marco donde en principio se alojan los vídeos del portal y que al pulsar sobre ella mostraría en una nueva ventana la presentación.

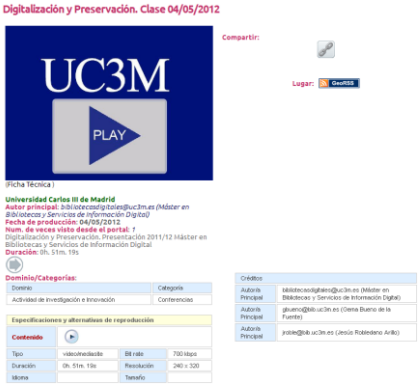


Figura 4.50: Visor de ArcaMM para vídeo Mediasite

Para agregar al marco del vídeo la imagen o el vídeo embebido dependiendo de cómo la presentación ha sido creada, nos basamos en el valor que tenga el mime, usando la imagen en el caso de que sea igual a *vídeo/mediasite*.

Al pulsar sobre la imagen, accederemos al sistema *Mediasite*, donde podremos navegar a través de la interfaz o cambiar el formato de presentación de forma independiente al portal de vídeos.

Aquí podemos ver un ejemplo de cómo puede ser esa interfaz, que dependerá del visor con el que haya sido creada dicha presentación.

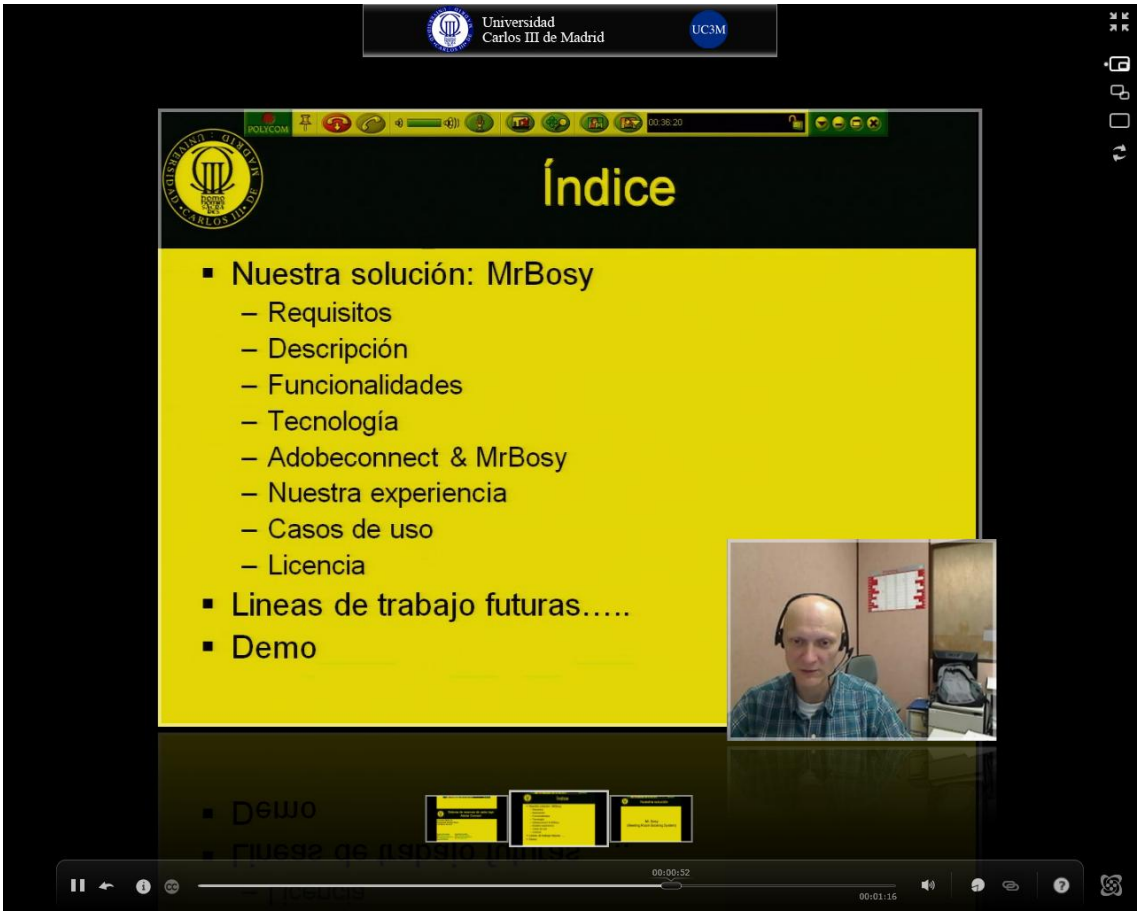


Figura 4.51: Ejemplo visor de Mediasite

Pues con esta última modificación daríamos por finalizado nuestro trabajo en el diseño de la aplicación, quedando resueltos todos los requerimientos que nos exigía el cliente.

CAPÍTULO 5 PRUEBAS DE LA APLICACIÓN

En este capítulo vamos a definir los casos de pruebas de sistema que permiten verificar el correcto funcionamiento de la aplicación.

Las pruebas de software son una serie de procesos de evaluación, cuyo objetivo es el de proporcionar información objetiva e independiente sobre la calidad del producto que se le ofrece al cliente final de la aplicación.

Dependiendo del proyecto que se haya realizado, las metodologías utilizadas pueden cambiar de forma sustancial. En el presente proyecto se realizarán únicamente pruebas de sistema ya que el cliente va a tomar parte activa en la creación de pruebas y no se cree conveniente hacer otros tipos de pruebas como unitarias, de aceptación o de integración.

5.1 PRUEBAS DE SISTEMA

Como hemos adelantado en la introducción del capítulo, vamos a detallar las pruebas de sistema realizadas a nuestra aplicación para verificar su correcto funcionamiento. Dado que se han utilizado técnicas agiles de desarrollo para el proyecto, en las pruebas también se seguirá dicha metodología, siendo el cliente quien valore si el proyecto avanza como desea y cumple sus expectativas. Para ello se debe tener en cuenta una serie de premisas para que el resultado de las pruebas sea el óptimo:

- El cliente debe de formar parte activa tanto en la creación de pruebas como en la verificación de estas.
- Existen pruebas que no se podrán realizar hasta que el software no sea entregado al cliente y será este el que verifique si realmente cumple sus necesidades iniciales.
- Únicamente se deben realizar pruebas de los requisitos que inicialmente nos ha impuesto el cliente.

Cada una de las pruebas se presentará en una tabla similar a la utilizada en el capítulo 3, en la que se especificarán una serie de campos comunes a cada una de las pruebas.

- **ID**: es el identificador de la prueba. Estará formado por dos valores separados por un punto, en el que el primer valor indica la parte de la aplicación en la que se realiza la prueba, que en nuestro caso son dos: el extractor de datos de Mediasite y la interfaz de publicación de contenidos. El segundo valor indica el número de prueba en esa parte de la aplicación.
- **Nombre**: es el nombre que se le atribuye a la prueba.
- **Resultados esperados**: objetivos que tiene la prueba para ser pasada.

- **Pasos**: son la serie de pasos que se ha de hacer en la aplicación para poder realizar la prueba.

- **Errores posibles**: son los posibles errores que se han encontrado o pueden llegar a encontrarse al realizar los pasos para realizar la prueba. En el caso de que estos existan la prueba de considerará como fallida.

- **Requisitos:** son los requisitos expuestos en el capítulo 3 con los que la prueba está relacionada.

- **Fecha:** fecha en la que la prueba se obtuvo de forma satisfactoria o en la que se obtuvieron mejores resultados.

- **Estado:** es el resultado final de la prueba. Existirán tres tipos de resultados, Pasada, Pasada con errores y No pasada. En el primer caso la prueba cumple todos los requisitos exigidos por el cliente. En el segundo caso la prueba lanza unos resultados satisfactorios, pero existen errores que no son significativos para el cliente y que se depurarán en versiones futuras de la aplicación. El tercer y último caso la prueba realizada muestra unos resultados negativos ya que uno o varios requisitos no se cumplen y por lo tanto no satisfacen al cliente.

ID		**Nombre**	
Resultados esperados			
Pasos			
Errores posibles			
Requisitos			
Fecha		**Estado**	

Tabla 5.1: Plantilla de prueba

Para todas las pruebas damos por supuesto que el código se encuentra en el servidor ArcaMM, la base de datos está creada, las instrucciones en el *cron* están dadas y el servidor está activo.

La primera de las pruebas que se realiza, especificada en la tabla 5.2 prueba que la primera vez que se arranca la aplicación, con la base de datos propia vacía, esta se rellene con todos los datos referentes a los vídeos que se encuentran en la base de datos de *Mediasite*.

ID	P1.1	**Nombre**	Recuperar metadatos de la base de datos de *Mediasite*
Resultados esperados	Todos los metadatos referidos a los vídeos se almacenan en la base de datos creada para la aplicación		
Pasos	1. Agregar los parámetros de configuración a través de la pestaña "Configuración" de la interfaz gráfica		
Errores posibles	• Los datos de configuración son erróneos • El servidor de *Mediasite* está apagado • La conexión con el servidor no es posible • El servidor MYSQL no está arrancado.		
Requisitos	RF-001, RF-002, RF-014, RI-013, RI-014		
Fecha	22/05/2012	**Estado**	PASADA

Tabla 5.2: P1.1 – Recuperar metadatos de la base de datos de Mediasite

En la siguiente prueba se observa si la actualización de la base de datos propia se hace de forma correcta en el caso de agregado, modificado o borrado de contenido de *Mediasite*.

ID	P1.2	**Nombre**	Actualización automática del contenido de la base de datos
Resultados esperados	La base de datos es actualizada de forma automática en intervalos de tiempo. En esta quedarán reflejados los cambios que se puedan producir en la base de datos de *Mediasite*, como agregado, modificación o borrado de contenido.		
Pasos	1. Agregar, modificar o eliminar contenido de Mediasite 2. Verificar una hora después si se actualiza el contenido en la aplicación		
Errores posibles	• Los datos de configuración son erróneos • El servidor de Mediasite está apagado • La conexión con el servidor no es posible • La instrucción en el *cron* es incorrecta • El servidor MYSQL no está arrancado.		
Requisitos	RF-002, RF-003, RF-004, RI-007, RI-009, RI-009, RF-014, RI-013, RI-014,RR-001		
Fecha	25/05/2012	**Estado**	PASADA

Tabla 5.3: P1.2 – Actualización automática del contenido de la base de datos

Estas fueron las pruebas realizadas en la primera parte del proyecto, el extractor de datos de contenidos *Mediasite*, que demuestran que el script de actualización funciona de forma correcta.

Las siguientes pruebas se corresponden con la interfaz gráfica realizada y la publicación en ArcaMM. En las dos primeras pruebas de esta parte se verifica el código HTML y CSS respectivamente.

ID	P2.1	**Nombre**	Páginas HTML válidas
Resultados esperados	Todas las páginas HTML que forman la aplicación no deben de tener errores		
Pasos	1. Realizar el test **http://validator.w3.org** a cada uno de los archivos HTML que forman la aplicación.		
Errores posibles	• Etiquetas mal cerradas. • Atributos incorrectos.		
Requisitos	RI-001		
Fecha	10/09/2012	**Estado**	PASADA CON ERRORES

Tabla 5.4: P2.1 – Páginas HTML válidas

ID	P2.2	**Nombre**	Hojas de estilo CSS válidas
Resultados esperados	Todas las hojas de estilo CSS que forman la aplicación no deben de tener errores		
Pasos	1. Todas las hojas de estilo CSS deben ser válidas. Deben superar el test del W3C (http://jigsaw.w3.org/css-validator.		
Errores posibles	• Selectores incorrectos • Propiedades erróneas para un determinado selector		
Requisitos	RI-002		
Fecha	10/09/2012	**Estado**	PASADA

Tabla 5.5: P2.2 – Hojas de estilo CSS válidas

En ambos casos al realizar las pruebas se encuentra que ambos test reportaban errores, que en el caso de las hojas de estilo pudieron ser solucionados, pero que en el caso de las páginas HTML no pudieron ser resueltos todos.

Al formar parte la aplicación del portal de vídeos de la UC3M muchos de estos errores, que en la gran mayoría de los casos eran *warnings*, se encuentran en ficheros *php* o *html* que no han sido creados en la realización de este proyecto pero que son referenciados por los ficheros creados.

Por ejemplo todas las páginas que forman ArcaMM tienen en común el encabezado y pie de página, con lo que se debían incluir en nuestros ficheros de forma obligatoria, y al pasar los ficheros por el test el 95% de los errores eran del fichero de encabezado y pie de página.

Al no ser errores críticos, la aplicación no se veía afectada por ellos, pero se puso en conocimiento del personal de desarrollo y mantenimiento del Portal de vídeos dichos errores.

A partir de este punto todas las pruebas realizadas tienen que ver con la funcionalidad de la aplicación de usuario realizada. Estas pruebas se realizan de forma conjunta por el equipo de trabajo y por el cliente, que forma también parte del equipo de trabajo en temas de asesoramiento.

En la siguiente prueba se comprueba el acceso a la aplicación realizada.

ID	P2.3	**Nombre**	Acceso a la aplicación
Resultados esperados	El usuario debe autentificarse en el Portal de Vídeos ArcaMM y acceder a la aplicación		
Pasos	1. El usuario debe de teclear en un navegador la url del Portal de vídeos ArcaMM http://arcamm.uc3m.es 2. Pulsará sobre el candado que se encuentra en la parte inferior de la página principal para acceder al área privada 3. El navegador le pedirá la autentificación y el usuario deberá escribir su login y contraseña de correo. 4. Si la autentificación se realiza de forma correcta, aparecerá una lista desplegable en la que elegirá la aplicación "Mediasite".		
Errores posibles	• No existe conexión a internet • El login o contraseña de correo son incorrectas • El usuario no tiene permitido el acceso • El usuario no tiene asignado el rol "Mediasite"		
Requisitos	RF-005, RI-005, RD-002		
Fecha	04/06/2012	**Estado**	PASADA

Tabla 5.6: P2.3 – Acceso a la aplicación

En la siguiente prueba se comprueba si la interfaz es cargada de forma correcta y cada una de las pestañas ser muestra la información correspondiente a cada una de ellas en el primer acceso.

ID	P2.4	**Nombre**	Visualización de la interfaz
Resultados esperados	Se visualizará el módulo de búsqueda y la carga inicial en cada una de las pestañas: • Vídeos que aún no fueron publicados • Vídeos publicados en ArcaMM • Grabaciones futuras programadas • Formulario para generar una estadística • Configuración actual de script		
Pasos	1. Se elegirá la pestaña de trabajo que deseé el usuario.		
Errores posibles	• No existe conexión a internet • No hay comunicación con la base de datos		
Requisitos	RF-006, RI-006, RI-007, RI-009, RI-010, RI-011, RI-013		
Fecha	06/06/2012	**Estado**	PASADA

Tabla 5.7: P2.4 – Visualización de la interfaz

En la siguiente prueba se comprueba que se realiza la búsqueda de forma correcta. Esto quiere decir que se muestran todos los resultados posibles de dicha búsqueda y esta se realiza sobre el grupo de presentaciones en las que esté el usuario trabajando.

ID	P2.5	**Nombre**	Búsqueda de contenidos haciendo uso del módulo de búsqueda
Resultados esperados	La búsqueda deberá devolver todos los resultados de forma correcta y en el refresco de la página situar la aplicación en el área de trabajo en la que el usuario realizó la búsqueda.		
Pasos	1. El usuario se posicionará en la pestaña en la que deseé realizar la búsqueda. 2. Rellenará los campos que crea conveniente del módulo de búsqueda		
Errores posibles	• No existe conexión a internet • No hay comunicación con la base de datos.		
Requisitos	RF-007, RF-008, RI-006, RI-007, RI-009, RI-010		
Fecha	14/06/2012	**Estado**	PASADA

Tabla 5.8: P2.5 – Búsqueda de contenidos haciendo uso del módulo de búsqueda

En la siguiente prueba se comprueba que la paginación de los ítems que se muestran se realiza de forma correcta. Como cada una de las pestañas es independiente del resto, cada una posee su propia interfaz de paginación, con lo que si se avanza o se retrocede de página deberá realizarse sobre la pestaña en la que se esté trabajando, manteniendo la misma página de resultados en el resto.

ID	P2.6	**Nombre**	Paginación de resultados de una búsqueda
Resultados esperados	Bien sea en el estado inicial o después de una búsqueda, en las pestañas "Vídeos no publicados", "Vídeos publicados " y "Presentaciones programadas" los ítems resultantes deben de ser paginados siempre que estos sean superiores a ocho en un mismo área de trabajo		
Pasos	1. El usuario entrará por primera vez a la aplicación o realizará una búsqueda si ya ha realizado alguna operación anterior 2. En el caso de que sea la primera vez que entra en la aplicación, seleccionará la pestaña de trabajo. Si bien ha realizado una búsqueda, la misma aplicación posicionará al usuario en el área de trabajo donde realizó la búsqueda. 3. Entre los ítems resultantes, el usuario podrá buscar con el que desea trabajar. Para ello se proporciona una interfaz de navegación en los que aparecen botones tales como "Primero", "Anterior", "Siguiente", "Ultimo" y una lista desplegable para seleccionar una página en concreto.		
Errores posibles	• No existe conexión a internet • No hay comunicación con la base de datos.		
Requisitos	RF-010, RI-006, RI-007, RI-009, RI-010		
Fecha	18/06/2012	**Estado**	PASADA

Tabla 5.9: P2.6 – Paginación de resultados de una búsqueda

En la siguiente prueba se debe comprobar si la ventana donde se muestran todos los detalles referentes a un vídeo se carga de forma correcta y se muestran todos los detalles menos relevantes ocultos, dando al usuario el poder de visualizar el dato que él deseé.

ID	P2.7	**Nombre**	Mostrar detalles de vídeos no publicados
Resultados esperados	Deberá mostrar todos los detalles referentes a una presentación no publicada. Los detalles referentes a la carpeta en la que se aloja en *Mediasite* o el códec de vídeo y audio, aparecerán ocultos en primera instancia. No obstante, el usuario puede mostrarlos u ocultarlos de forma independiente a su antojo pulsando un botón.		
Pasos	1. El usuario entrará por primera vez a la aplicación o realizará una búsqueda si ya ha realizado alguna operación anterior. 2. Si fuera necesario, entre los resultados obtenidos buscará con la interfaz de paginación el ítem en el que desea trabajar. 3. Una vez encontrado, pulsará sobre el botón de información del ítem y emergerá una ventana nueva con los detalles de la presentación 4. Mostrará u ocultará la información que considere el usuario a través de los botones con forma de flecha que aparecen en la ventana		
Errores posibles	• No existe conexión a internet • No hay comunicación con la base de datos. • El usuario no tiene activado Java Script en su navegador		
Requisitos	RF-009, RF-010, RI-006, RI-007, RI-008		
Fecha	02/07/2012	**Estado**	PASADA

Tabla 5.10: P2.7 – Mostrar detalles de vídeos no publicados

En la siguiente prueba se comprueba si la comunicación a través de url entre las distintas funciones de la aplicación se realiza de forma correcta. Todas las funciones de la aplicación (búsqueda, paginación, mostrar detalles, pestaña activa, etc), como las que se realizan en la publicación del vídeo, se realizan pasando una serie de parámetros a través de la url.

ID	P2.8	**Nombre**	Comunicación a través de url
Resultados esperados	Cualquiera de las funcionalidades que recibe parámetros a través de url debe funcionar de forma correcta		
Pasos	1. Realizar cualquiera de estas acciones: • Cambiar de pestaña • Búsqueda • Cambiar de página • Mostrar detalles de un vídeo • Mandar formulario para crear estadísticas • Cambiar algún parámetro de una determinada estadística ya realizada • Cambiar configuración del script • Publicar un contenido		
Errores posibles	• No existe conexión a internet • No hay comunicación con la base de datos. • El usuario no tiene activado *JavaScript* en su navegador		
Requisitos	RI-004, RI-006, RI-007, RI-008, RI-009, RI-010, RI-011, RI-012, RI-013, RI-014, RI-015.		
Fecha	30/07/2012	**Estado**	PASADA

Tabla 5.11: P2.8 – Comunicación a través de url

En la siguiente prueba se comprueba que un determinado contenido seleccionado en la pestaña "Vídeos no publicados" se publica en ArcaMM. Esta es la prueba más relevante, ya que se demostrará si cumple el objetivo principal del PFC.

ID	P2.9	**Nombre**	Publicar vídeos en ArcaMM
Resultados esperados	El vídeo seleccionado debe estar publicado en ArcaMM y ser visualizado desde este portal		
Pasos	1. Buscar el vídeo que se desea publicar 2. Utilizar el módulo de búsqueda o la interfaz de paginación 3. Visualizar los detalles del vídeo (opcional) 4. Pulsar en el botón "Publicar en ArcaMM" o bien desde el botón para tal efecto en la pestaña "Vídeos no publicados" o desde la ventana emergente de detalles del vídeo. 5. En el rol "manager" rellenar los campos obligatorios que la aplicación no puede rellenar como la categoría de la publicación, la serie o la privacidad de la publicación. 6. Modificar si se considera necesario alguno de los campos rellenados automáticamente como el título, la descripción, la imagen predefinida o los autores. 7. Pulsar el botón enviar situado en la parte inferior de la pantalla, cuando todos los datos del vídeo estén rellenados. En caso de que falte algún campo obligatorio la aplicación avisará al usuario del campo/s que faltan por rellenar. 8. Visualizar el vídeo en ArcaMM.		
Errores posibles	• No existe conexión a internet • No hay comunicación con la base de datos. • El usuario no tiene activado Java Script en su navegador • El rol "manager" no funciona correctamente		
Requisitos	RF-015, RF-016, RF-017, RI-006, RI-007, RI-008, RI-015		
Fecha	03/09/2012	**Estado**	PASADA

Tabla 5.12: P2.9 – Publicar vídeos en ArcaMM

En la siguiente prueba se comprueba que la comunicación entre ArcaMM y nuestra aplicación es la correcta. En los casos en los que se publique o se borre un contenido en ArcaMM, nuestra aplicación deberá ser avisada para saber que presentaciones están publicadas y cuáles no.

ID	P2.10	**Nombre**	Recibir parámetros desde ArcaMM
Resultados esperados	En el caso de que se publique un vídeo en ArcaMM, el portal debe de avisar de que la publicación se realizó de la forma correcta. En el caso de que se borre un contenido anteriormente publicado con la aplicación o con el *mime* "vídeo/mediasite", ArcaMM debe comunicar a la aplicación que cambie el estado de la presentación en la base de datos, de publicado a no publicado.		
Pasos	1. Después de publicar un vídeo, se debe comprobar que este aparece el primero en la pestaña de "Vídeos publicados" y no aparece en "Vídeos no publicados". 2. En el caso de borrado de contenido se debe comprobar que el vídeo no aparece en la pestaña "Vídeos publicados " y puede volver a ser publicado en la pestaña "Vídeos no publicados "		
Errores posibles	• No existe conexión a internet • No hay comunicación con la base de datos. • El rol "manager" no funciona correctamente • No hay comunicación con la base de datos de ArcaMM • El parámetro no se ha enviado correctamente.		
Requisitos	RF-002, RF-017, RI-006, RI-007, RI-008, RI-015		
Fecha	12/09/2012	**Estado**	PASADA

Tabla 5.13: P2.10 – Recibir parámetros de ArcaMM

En la siguiente prueba se comprueba si se realizan las gráficas de forma correcta.

ID	P2.11	**Nombre**	Realizar una estadística
Resultados esperados	Mostrará la estadística con los términos que el usuario crea conveniente. Se facilitarán los siguientes resultados: • Gráfica de evolución temporal • Gráfica de resultados totales • Tabla resumen disgregada en los espacios temporales elegidos y los totales.		
Pasos	1. Acceder a la aplicación 2. Elegir la pestaña "Estadísticas de publicación" 3. Rellenar cada uno de los campos del formulario 4. Una vez generada la estadística, se puede ver una gráfica y ocultar la otra a través de dos pestañas. La tabla resumen siempre será visible. 5. Poner el ratón sobre cualquier punto de la gráfica y emergerá un mensaje con la cifra en ese espacio temporal. 6. Pulsar "volver" si se desea cambiar algún parámetro. 7. Pulsar "nueva" si se desea realizar una nueva estadística.		
Errores posibles	• No existe conexión a internet • No hay comunicación con la base de datos. • El usuario no tiene activado *JavaScript* en su navegador • *Adobe Flash Player* no funciona correctamente o la versión está obsoleta.		
Requisitos	RF-011, RF-012, RF-013, RI-011, RI-012		
Fecha	08/07/2012	**Estado**	PASADA

Tabla 5.14: P2.11 – Realizar una estadística

En la siguiente prueba se comprueba que tanto la modificación del script, como la carga de la configuración de respaldo se realizan de forma correcta.

ID	P2.12	**Nombre**	Modificar o cargar una nueva configuración del script
Resultados esperados	Los datos del script de actualización se deben de modificar a través de la modificación realizada a través de la pestaña "Configuración".		
Pasos	1. Acceder a la aplicación 2. Elegir la pestaña "Configuración" 3. Elegir entre el botón "Backup" o "Realizar Cambios" 4. En el caso de elegir "Backup", aparecerá la versión de respaldo y si se desea se puede poner como principal al pulsar sobre "Aplicar y guardar estos datos". 5. En el caso de elegir "Realizar Cambios", aparecerá un formulario con cada uno de los datos modificables. En cada uno de los campos aparecerá el valor actual y debajo un cuadro de texto donde se introducirá el nuevo valor. 6. Pulsamos sobre "Verificar cambios" para comprobar que son correctos. En caso de no estar conforme se puede volver atrás. En caso de que sean correctos pulsar "Enviar" para guardar la nueva configuración. 7. Comprobar que la nueva configuración es correcta en el transcurso de una hora, que es el tiempo de actualización del script.		
Errores posibles	• No existe conexión a internet • No hay comunicación con la base de datos.		
Requisitos	RF-014, RI-013, RI-014, RR-001		
Fecha	14/07/2012	**Estado**	PASADA

Tabla 5.15: P2.12 – Modificar o cargar una nueva configuración del script

Por último la última prueba que se realiza es si la aplicación es accesible desde los cuatro navegadores más utilizados en España en la actualizad: Internet Explorer, Mozilla Firefox, Google Chrome y Safari.

ID	P2.13	**Nombre**	Compatibilidad con navegadores web convencionales
Resultados esperados	La aplicación debe ser funcional y visible independientemente del navegador que se esté utilizando. Las prueba ha sido realizada con: • Internet Explorer 8 • Mozilla Firefox 20.0 • Google Chrome 26.0.1410.43 m • Safari 5		
Pasos	1. Acceder a la aplicación con Internet Explorer 8 2. Acceder a la aplicación con Mozilla Firefox 20.0 3. Acceder a la aplicación con Google Chrome 26.0.1410.43m 4. Acceder a la aplicación con Safari 5		
Errores posibles	• No existe conexión a internet • No hay comunicación con la base de datos. • Cambios en el tratamiento de hojas de estilo CSS • Extensiones deshabilitadas como *JavaScript* o *Adobe Flash Player*		
Requisitos	RI-003		
Fecha	12/09/2012	**Estado**	PASADA

Tabla 5.16: P2.13 – Compatibilidad con navegadores web convencionales

CAPÍTULO 6 CONCLUSIONES Y LÍNEAS FUTURAS

En este sexto y último capítulo se incluyen las conclusiones a las que se ha llegado con la finalización del proyecto, entre las que se puede encontrar una evaluación personal de los resultados obtenidos, experiencias y aprendizaje adquiridos. Además se incluye, en un segundo apartado, las líneas futuras donde se sugieren ideas que pueden hacer la aplicación mejor y con más funcionalidades

6.1 CONCLUSIONES

Se ha realizado como Proyecto Fin de Carrera una aplicación web que tiene como fin publicar el contenido generado con una herramienta corporativa llamada Mediasite, en el Portal de Vídeos de la Universidad Carlos III de Madrid. Es una aplicación a la que solo podrán tener acceso determinados usuarios elegidos por el administrador del sistema, pero del que se beneficiarán indirectamente toda la comunidad universitaria. Creemos que la aplicación realizada es de gran utilidad ya que conseguimos integrar los vídeos generados con *Mediasite* en una plataforma totalmente independiente y en la que podrá convivir con otros contenidos audiovisuales tales como grabaciones de vídeo en distintos formatos, grabaciones de audio o archivos descargables.

El resultado final obtenido es muy satisfactorio tanto para el cliente como para el equipo de desarrollo, ya que se han cumplido todos los requisitos exigidos por el cliente y el equipo de desarrollo siente que su trabajo fue útil. Uno de los principales motivos para esta doble satisfacción es el método de desarrollo utilizado para la realización de este proyecto, basado en un método iterativo e incremental. Como vimos en el punto 1.4 de esta memoria, lo que principalmente nos dicta este método, es dividir el desarrollo de la aplicación en iteraciones temporales en el que la dificultad y los objetivos que se cumplen, aumentan con el paso de las iteraciones, haciendo hincapié en las reuniones que se llevan a cabo con el cliente entra cada una de las iteraciones. Uno de los motivos principales para utilizar este método de resolución es la cercanía y buena relación que el equipo de trabajo tiene con el cliente ya que comparten la misma área laboral, el Servicio de Informática de la UC3M. Otro de los motivos de la elección de esta metodología es la cantidad de beneficios que podría reportar a un proyecto de las características de nuestra aplicación y que se pusieron de manifiesto en el transcurso de la realización de este. Entre estos beneficios podemos encontrar:

- El cliente podría hacer cambios en los objetivos o requisitos propuestos inicialmente. Al finalizar cada una de las iteraciones y después de presentar por parte del equipo de trabajo los resultados obtenidos, el cliente podría agregar, modificar o eliminar requisitos en función de estos.

- Iteraciones temporalmente cortas en las que se presentan los resultados mediante reuniones, lo que favorece la interacción cliente/equipo de trabajo y la toma de decisiones.

- En caso de desechar unos determinados resultados, solo perderemos los recursos de esa iteración y no los de todo el proyecto.

- Al tener que mostrar los resultados de los requisitos obtenidos al final de cada una de las iteraciones, se consigue minimizar los errores y aumentar la calidad del producto final. Además estas reuniones permiten gestionar la complejidad de la solución a realizar y mitigar riesgos.

Durante el desarrollo del proyecto se han puesto en práctica bastantes conocimientos adquiridos en diferentes asignaturas cursadas en la carrera, principalmente en asignaturas relacionadas con la programación, la gestión de bases de datos o el diseño web. Además se han tenido que adquirir nociones sobre el desarrollo de software, ya que vagamente habían sido conocidos en una asignatura y para la realización de este proyecto era una parte importante. Otro aspecto importante era conocer el funcionamiento interno de *Mediasite* (como se comunicaban los distintos servidores, codificación de audio y vídeo, creación de presentaciones, etc.). Para comprenderlo de una forma rápida fue de gran ayuda los conocimientos adquiridos en asignaturas relacionadas con arquitectura de redes, codificación y tratamiento de audio y vídeo, que formaban el grueso de las asignaturas de la I.T.T. especialidad en Sonido e Imagen. Por último, y relacionado con los conocimientos audiovisuales, estos fueron puestos en práctica en el Área de Audiovisuales de la UC3M desde que entré como becario en el año 2007, después cuando en 2009 me contrataron como responsable técnico, hasta la actualidad, dándome la oportunidad de afrontar un proyecto de las características del realizado. Para mi son de gran importancia mis años en el Área de Audiovisuales, ya que a pesar de que no forma parte de mi carrera, personalmente me ha ayudado a comprender muchos aspectos adquiridos de forma teórica en la carrera.

En el apartado personal, me gustaría destacar la relación positiva adquirida con el equipo de trabajo, ya que he aprendido a trabajar en equipo y a escuchar las exigencias de un cliente en un proyecto real. Por otro lado la satisfacción personal de dar a conocer a la comunidad universitaria una herramienta como *Mediasite,* que puede facilitar la docencia teórica e incluso práctica por parte de profesores y alumnos, ahorrando tiempo que puede ser utilizado en otras actividades. Además se han adquirido bastantes conocimientos no aprendidos durante el transcurso de la carrera, sobre todo en el desarrollo de software, que espero que sean de gran utilidad en mi vida laboral. Por tanto en este apartado puedo concluir que personalmente el resultado final de la aplicación ha sido muy gratificante.

A lo largo del trabajo en la realización del proyecto, se encontraron algunos problemas que o bien ralentizaron el desarrollo del proyecto o obligaron a desechar una idea inicial y pensar una solución alternativa con los recursos disponibles. Durante los primeros días, incluso semanas, el desconocimiento interno de *Mediasite* por parte del equipo de trabajo ralentizó de manera sustancial el inicio del proyecto, ya que se necesitaba evaluar la arquitectura del sistema, así como las posibilidades que ofrecía la documentación proporcionada por el fabricante. En esta fase inicial apareció otro problema, al revisar la documentación encontramos que muchos datos que se proporcionaban en la interfaz gráfica del propio sistema *Mediasite* no eran accesibles a través del EDAS, lo cual desechaba ciertas exigencias por parte del cliente como por ejemplo el uso de fotogramas de la grabación o el uso de las estadísticas que proporcionaba el propio sistema. A pesar de estas restricciones por parte del fabricante, el equipo de trabajo ideó alternativas para cumplir dichos objetivos como la creación de tres imágenes con el logo institucional de la Universidad y el nombre del campus de realización de la grabación, y la posibilidad de crear estadísticas de publicación por

parte del usuario que no se podían generar con el sistema *Mediasite*. Otro problema al que se tuvo que hacer frente, es que había que familiarizarse con el uso de *JavaScript* y j*Query*, ya que hasta el momento de plantearnos la realización del proyecto, estos lenguajes no habían sido utilizados y esos conocimientos debían ser adquiridos. El aprendizaje de dichos lenguajes supuso una pausa en el proyecto, quizá más larga de la deseada, pero permitieron en la reanudación realizar la interfaz gráfica de la aplicación de una forma más rápida y eficaz.

Por otro lado, en ningún momento se exigió por parte del cliente un límite máximo de duración de desarrollo del proyecto, aunque se invitó a realizarlo lo antes posible. El motivo principal de esta decisión fue el desconocimiento tanto por el cliente como por el equipo de trabajo de las posibilidades que ofrecía la documentación del sistema. A pesar de que este hecho pueda parecer una falta de exigencia al equipo de trabajo, no se consideró como tal, ya que al utilizar una metodología iterativa e incremental, siempre estaba marcado en el calendario la fecha de la siguiente reunión, en la que se debían presentar al cliente las posibilidades que ofrecía la documentación, el cumplimiento de los objetivos, soluciones a los problemas surgidos o las posibles mejoras, con lo que generaba que el equipo de trabajo no bajara los brazos en ningún momento.

Para finalizar este apartado de conclusiones, se evaluarán la consecución de los objetivos que se propusieron en un principio al inicio de este proyecto y que fueron enumerados en el punto 1.3 de esta memoria.

- **Desarrollar una herramienta para ArcaMM, para catalogar los vídeos creados con *Mediasite*, integrando las dos plataformas, facilitando el acceso a dichos vídeos por parte de la comunidad universitaria**: Este constituye el objetivo principal y se ha visto cumplido gracias al cumplimiento de los objetivos concretos explicados en los siguientes puntos.

- **Creación de una base de datos propia:** Este objetivo era el menos prioritario, ya que dependiendo de las posibilidades que podía ofrecer el acceso a los datos de los vídeos de *Mediasite*, se decidiría si había que cumplir dicho objetivo. Se optó por realizarlo, cumpliendo las expectativas mediante una base de datos propia de la herramienta y la automatización de la actualización de esta.

- **Creación de la interfaz de muestra de contenidos:** Este objetivo era necesario que se cumpliera de la mejor forma posible, ya que era lo que el usuario iba a poder ver con sus ojos y el cual iba a facilitar la publicación de los contenidos en ArcaMM. A pesar de las restricciones de diseño impuestas por el cliente o las que generaba la no extracción de determinados datos y el aprendizaje de los lenguajes necesarios para realizarla, el objetivo ha sido conseguido ampliamente ya que se ha realizado una interfaz sencilla y en la que el usuario se sentirá familiarizado rápidamente debido al parecido con el resto de herramientas de ArcaMM.

- **Publicar vídeos en el Portal ArcaMM:** Este objetivo fue por el cual se pensó en la realización de este proyecto, ya que principalmente se buscaba integrar los contenidos generados con *Mediasite* en el Portal de Vídeos de la Universidad Carlos III. A pesar de que es una parte menos vistosa para el usuario final de la aplicación, para el cliente este objetivo era el más importante. Los objetivos se han cumplido ampliamente y así nos lo ha confirmado el cliente. A pesar de ello han quedado una serie de mejoras, que se explicarán en el siguiente apartado y que se ha optado que sean líneas futuras de trabajo para mejorar la aplicación realizada. Los problemas a la hora de embeber la interfaz de vídeo de *Mediasite* en el portal, han sido los causantes de que este objetivo no se complete en su totalidad.

6.2 LÍNEAS FUTURAS

Una vez expuestas las conclusiones obtenidas en la finalización del proyecto, se puede pensar en una serie de líneas futuras que en algunos casos son parte de algún objetivo que no se ha podido cumplir o bien que el equipo de trabajo considera interesantes que se desarrollasen para mejorar la aplicación realizada.

6.2.1 ADECUACIÓN DEL SCRIPT A LA VERSIÓN 6 DE MEDIASITE

Durante las últimas semanas de la realización de este proyecto fin de carrera, se nos comunicó que la versión de *Mediasite* iba a ser actualizada a la versión 6. Una de las mejoras más sustanciales que ofrecía esta nueva versión era la de la total compatibilidad con dispositivos móviles, con sistemas operativos como iOS y Android. Para ello el servicio técnico del fabricante debía añadir un nuevo servidor a la arquitectura existente para llevar a cabo la actualización.

Antes de que esto se llevara a cabo, se proporcionó el EDAS de la versión 6, como ya pasó en su momento con el de la versión 5, para que el equipo de trabajo evaluara si era posible adaptar el script de actualización a esta nueva versión. En seguida se demostró, que el cambio era bastante notable y se puso en conocimiento del cliente y se concluyó dejar ese trabajo como una futura mejora. En principio los vídeos generados antes de la actualización estarían en el servidor antiguo con lo que seguirían siendo accesibles a través de nuestra aplicación y no se predería la publicación ya realizada en ArcaMM. Además como era necesario realizar una codificación de todos los vídeos antiguos para que fueran compatibles para dispositivos móviles y ser traspasados al nuevo servidor, se concluyó que tanto como lo anterior como la adecuación del script a la nueva versión se hiciera en un futuro proyecto, en el que posiblemente se reutilizaría la interfaz realizada, así como la base de datos. Por desgracia, el cambio ha sido tan grande que las funciones del EDAS han cambiado de nombre, atributos, etc, haciendo imposible reciclar cualquiera de las partes del script, aunque esperemos que tanto el código y los comentarios realizados en él sean de gran ayuda para el futuro programador.

6.2.2 AÑADIR PRESENTACIONES FUTURAS A LA AGENDA

En la actualidad existe una agenda de actos, en la que tanto los trabajadores del Área de Audiovisuales como el usuario de las salas multimedia, puede visualizar la disponibilidad de las distintas salas de audiovisuales repartidas entre los distintos campus.

Aquí podemos ver en la imagen la forma que tiene esta agenda:

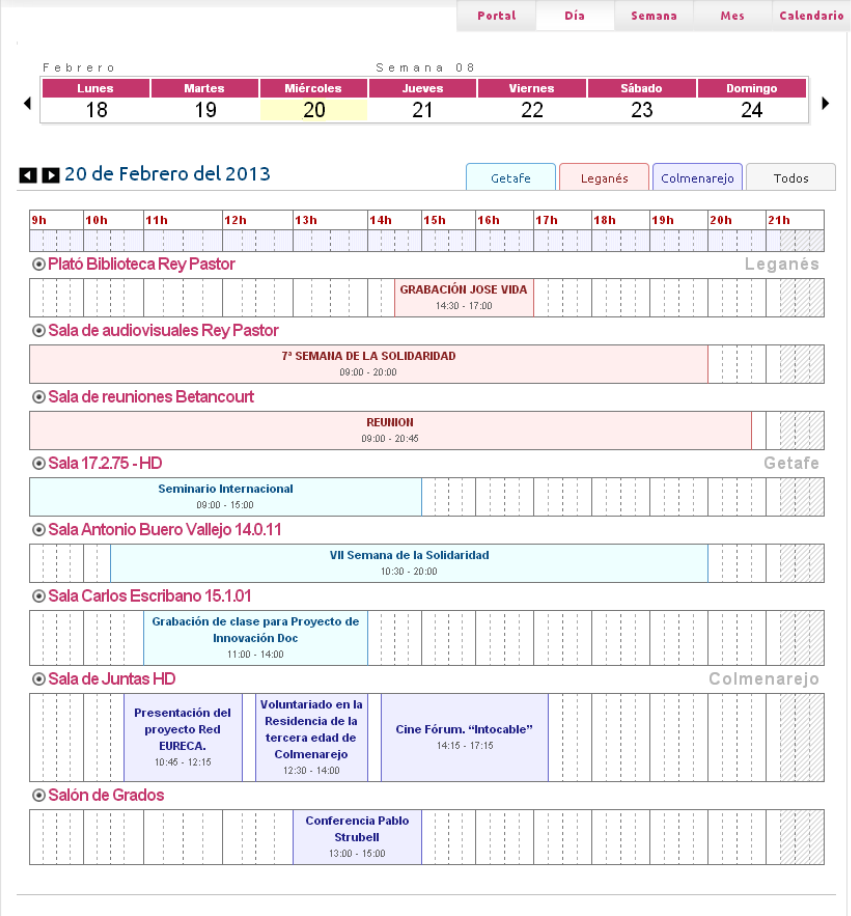


Figura 6.1: Agenda de actos del Área de Audiovisuales

La mejora que se propone es la de añadir de forma automática las grabaciones programadas en *Mediasite* en esta agenda, proporcionado al usuario el enlace donde se podría seguir el evento en directo. Como se ha visto a lo largo de la memoria, esta mejora se intentó implementar en la aplicación realizada, pero la dificultad que conllevaba, así como la necesidad de colaboración del equipo de trabajo de la agenda cuando estaban inmersos en plena actualización de su aplicación, desecho la mejora, siendo propuesta para una posible línea futura.

A pesar de este problema, se decidió realizar una solución intermedia, que como vimos en el apartado 4.2.2.3, que consistía en presentar al usuario de la aplicación las distintas presentaciones programadas en el sistema *Mediasite*, proporcionando el enlace donde se podía seguir el evento en directo. Este enlace se podía utilizar para ser pegado en correos, foros de la asignatura o para realizar una entrada manual en la agenda de actos.

6.2.3 CREAR PRESENTACIONES DESDE LA INTERFAZ CREADA

Una de las posibilidades aportada por el EDAS del fabricante y que no ha sido utilizada en este proyecto, es la de creación de presentaciones, presentador, plantillas o perfiles de grabación a través de las funciones que ofrecía la documentación.

Para ello se debería crear una pestaña adicional en la interfaz gráfica, en la que se ofreciera al usuario esta posibilidad. Para ello se deberían mostrar una serie de parámetros modificables para poder elegir el tipo de ítem que se va a crear y una serie de cuadros de texto o listas desplegables en los que el usuario introduciría los datos correspondientes al ítem que desea crear. En el caso de una nueva presentación, el usuario introduciría el título, la fecha y hora del evento, el presentador, el perfil de grabación y el tipo de reproductor que se usará para visualizar el contenido. Finalmente, se podrá enviar la información al servidor de *Mediasite*, tras las comprobaciones pertinentes.

6.2.4 EMBEBER VÍDEO EN LA PÁGINA DE ARCAMM

Para finalizar el apartado de líneas futuras, se hará hincapié en la integración total del reproductor de *Mediaste* en el Portal de Vídeos de la UC3M. Al ser una cosa que no depende del programador, ya que se trata de una restricción del fabricante, se deberá esperar a que este agregue dicha mejora a una futura versión.

Al parecer esa mejora ya se ha llevado a cabo en la versión 6 de *Mediasite,* dando la posibilidad de embeber cualquiera de los reproductores en cualquier página web que tenga compatibilidad con HTML5. En un principio, esta mejora no se puede realizar en ArcaMM, ya que en estos momentos se está trabajando en la adecuación del portal a esta nueva tecnología que elevará las posibilidades que ArcaMM ofrecerá al usuario.

En el momento que esto ocurra, se podrá eliminar la imagen (Figura 4.47) del cuadro de vídeo y en el momento que se cargue la página, aparecerá la reproducción del vídeo con todos los botones que ofrece la interfaz de *Mediasite*, con el que el usuario tendrá el control sobre el vídeo si así lo desea, sin la necesidad de que se abra la reproducción en una ventana nueva.

BIBLIOGRAFÍA

[1] IEEE. 830-1993 - IEEE Recommended Practice for Software Requirements Specifications. The Institute of Electronic and electrical Enginners , Inc. Diciembre 1993.

[2] MELONIE, Julie C. PHP, MySQL y Apache. Madrid: Anaya Multimedia, D.L. 2008. 656 p. ISBN: 9788441525412.

[3] FLANAGAN, David. JavaScript : la guía definitiva. Madrid: Anaya Multimedia 2007. ISBN: 9788441522022.

[4] CHAFFER, Jonathan; SWEDBERG, Karl. Learning jQuery Third Edition. Packt Publishing 2011. 428p. ISBN-13:978-1-84951-654-9

[5] RIISMANDEL, Paul. Capture Lecture, Skip Class?. Streaming Media Magazine. United States, Information Today, Inc. Oct-Dec 2011. ISSN: 15598039.

[6] BRITTAN-POWELL, C; EL HAGGAN, A; BRAHA, H; GREGORY S.". Lecture Capture Systems- Are they worth it?". En: International conference on Education and New Learning Technology (Barcelona, 5-7 de julio de 2010)

[7] COTILLAS PALOMO, Christian. Opencast-Matterhorn: Una solución abierta para la captura de clases y su implantación dentro de la infraestructura de la UC3M. Leganés 2012.

[8] FROST & SULLIVAN. The Lecture Capture Solutions Market - A Frost & Sullivan Insight [En línea]. <http://www.frost.com/sublib/display-market-insight-top.do?id=191225269> [Consulta: 3 Febrero 2013]

[9] POLYCOM. Polycom Accordent PresenterPRO. [En línea]. <http://www.polycom.es/products/uc_infrastructure/realpresence_platform/video_content_management_solutions/enterprise_video_capture/realpresence_broadcast_producer.html>[Consulta: 15 Febrero 2013]

[10] ECHO 360. Integration. [En línea] <http://echo360.com/integration> [Consulta: 15 Febrero 2013]

[11] PANOPTO. Panopto for education. [En línea] <http://www.panopto.com/lecture-capture>[Consulta:16 Febrero 2013]

[12] MEDIASITE. Centro del conocimiento- Case Stuidies. [En línea]. <http://www.sonicfoundry.com/knowledge-center/case-studies/list> [Consulta 17 de Febrero 2013]

Bibliografía

[13] DREXEL UNIVERSITY. Workshop Archives. [En línea]
<http://www.drexel.edu/irt/help/workshops/archive/>
[Consulta: 18 Febrero 2013]

[14] UNIVERSITY OF WISCONSIN LA CROSSE. UW-La Crosse Mediasite Archive Catalog.[En línea]. <http://www.uwlax.edu/its/ats/mediasite/index.htm>
[Consulta 18 Febrero 2013]

[15] VESTA. VESTA Mediasite Catalog. [En línea]
<http://www.mediasitesvr.missouristate.edu/VESTA/Catalog/pages/catalog.aspx?catalogId=5c9fb41a-0d6c-44ae-b5bb-cecce4560a5e>
[Consulta 18 Febrero 2013]

[16] EDUCAUSE. 7 things you should know about Lecture Capture. [En línea]
<http://net.educause.edu/ir/library/pdf/ELI7044.pdf> [Consulta: 3 Febrero 2013]

[17] CAMPUS TECNOLOGY. Capturing the market. [En línea]
<http://campustechnology.com/Articles/2009/06/01/Lecture-Capture.aspx>
[Consulta: 6 Febrero 2013]

[18] OFC. Open Flash Chart. [En línea]
< http://teethgrinder.co.uk/open-flash-chart/> [Consulta: 27 Mayo 2012]

APÉNDICES

APÉNDICE A PRESUSUESTO DEL PROYECTO

Una vez que se ha conocido todo lo necesario para la realización del proyecto, sólo faltaría detallar cual sería el coste mínimo para la realización de este. En primer lugar se ha de computar los gastos asociados al personal que se ha encargado de la realización de la aplicación.

Como personal se ha incluido a Sergio Tejero López como principal desarrollador de la herramienta. Además se incluye a Javier García Guzmán como tutor del proyecto y a David Santín Cristóbal como responsable de desarrollo del Área de Audiovisuales, ya que también se deben de computar las horas de reuniones mantenidas con ambos que han servido de mucha ayuda para la realización del proyecto.

El coste total asociado al personal es de 8.178 €:

Personal	Categoría	Coste persona/hora	Dedicación (horas)	Coste (euros)
Tejero López, Sergio	Ingeniero Junior	18	389	7.002
García Guzmán, Javier	Ingeniero Sénior	42	16	672
Santín Cristóbal, David	Ingeniero Sénior	42	12	504
Total persona hora		102	**Total**	**8.178**

Tabla 7.1: Coste del personal

También se deben de tener en cuenta en el coste total del proyecto las amortizaciones de los ordenadores o servidores que se han utilizado en cualquiera de las fases de la realización del proyecto.

El coste imputable se ha calculado utilizando los siguientes datos:

- U = Número de meses desde la fecha de facturación en que el equipo se ha utilizado
- PD = Periodo de deprecación
- C = Coste del equipo o servicio utilizado sin IVA
- p = % del uso que se dedica a la realización del proyecto

Teniendo en cuenta los anteriores datos, calcularíamos el coste imputable de cada uno de los servicios o equipos, aplicando la siguiente fórmula.

$$U/PD * C*p$$

El coste total asociado a las amortizaciones es de 399.41 €

Descripción	Coste (Euro)	Dedicación (Meses)	Uso dedicado al proyecto (%)	Periodo de depreciación (meses)	Coste Imputable
Ordenador ASUS CG8270 i7-3770 12G 2T+64G W8	1.345	4	100	48	112.08
Sistema(*) Mediasite	41.100	2	20	60	274
Servidor ArcaMM	4000	2	10	60	13.33
				Total	**399.41**

Tabla 7.2: Coste de las amortizaciones

En la siguiente tabla se desglosa el coste total del sistema *Mediasite*

Sistema Mediasite(*)			
Descripción	**Unidades**	**Coste Unitario**	**Coste Total**
Licencia de Software	1	20.000	20.000
Servidores HP	2	3.800	7.600
Recorders	5	1.900	9.500
Mantenimiento	1	4.000	4.000
		Total	41.100

Tabla 7.3: Desglose del coste del sistema Mediasite

En esta última tabla se muestra el coste final de la aplicación, sumando todos los costes obtenidos anteriormente. El coste total asciende a 8577.41 euros (ocho mil quinientos setenta y siete con cuarenta y un céntimos).

Presupuesto	Coste(Euro)
Coste del personal	8.178
Coste de amortizaciones	399.41
Coste total	**8577.41**

Tabla 7.4: Coste total

APÉNDICE B WEB SERVICE

Web Service

List of all public methods available in the External Data Access Service is discussed in this section.

Namespace
Sonic Foundry.Mediasite.WebServices.ExternalDataAccess.Client

Web Proxy Methods

These methods are available on the Proxy object for the External Data Access Web Service.

Signature	**TestResponse Test(TestRequest)**
Description	Test the Connection to the External Data Access Service.

GetVersion

Signature	**GetVersionResponse GetVersion(GetVersionRequest)**
Description	Gets the version of the External Data Access Service.

Login:

Signature	**LoginResponse Login(LoginRequest)**
Description	Log into the Mediasite Server using a username and password. The Mediasite Server must be configured to use security.

CreatePresentation:

Signature	**CreatePresentationResponse CreatePresentation(CreatePresentationRequest)**
Description	Create a scheduled presentation in the Mediasite Server.

QueryPresenters:

Signature	**QueryPresentersResponse QueryPresenters(QueryPresentersRequest)**
Description	Get a list of presenters from the Mediasite Server.

QueryPresenterDetails:

Signature	**QueryPresenterDetailsResponse QueryPresenterDetails(QueryPresenterDetailsRequest)**
Description	Get presenter details for a presenter from the Mediasite Server.

QueryFolderDetails:

Signature	**QueryFolderDetailsResponse QueryFolderDetails(QueryFolderDetailsRequest)**
Description	Get folder details for a folder from the Mediasite Server.

QueryViewerSkins:

Signature	**QueryViewerSkinsResponse QueryViewerSkins(QueryViewerSkinsRequest)**
Description	Get a list of viewer skins from the Mediasite Server.

QueryServerGroups:

Signature	**QueryServerGroupsResponse QueryServerGroups(QueryServerGroupsRequest)**
Description	Get a list of server groups from the Mediasite Server.

QueryStreamingProfiles:

Signature	**QueryStreamingProfilesResponse QueryStreamingProfiles(QueryStreamingProfilesRequest)**
Description	Get a list of streaming profiles from the Mediasite Server.

QueryStreamingProfileDetails:

Signature	**QueryStreamingProfileDetailsResponse QueryStreamingProfileDetails(QueryStreamingProfileDetailsRequest)**
Description	Get streaming profile details for a streaming profile from the Mediasite Server.

QueryTimeZones:

Signature	**QueryTimeZonesResponse QueryTimeZones(QueryTimeZonesRequest)**
Description	Get a list of time zones from the Mediasite Server. The list returned includes only time zones that are available for use on the Mediasite Server.

QueryTimeZoneDetails:

Signature	**QueryTimeZoneDetailsResponse QueryTimeZoneDetails(QueryTimeZoneDetailsRequest)**
Description	Get time zone details for a time zone from the Mediasite Server.

QueryPresentations:

Signature	**QueryPresentationsResponse QueryPresentations(QueryPresentationsRequest)**
Description	Get a list of presentations in a folder from the Mediasite Server.

QueryAllPresentations:

Signature	**QueryAllPresentationsResponse QueryAllPresentations(QueryAllPresentationsRequest)**
Description	Get a list of presentations from the Mediasite Server.

QueryPresentationDetails:

Signature	**QueryPresentationDetailsResponse QueryPresentationDetails(QueryPresentationDetailsRequest)**
Description	Get presentation details (metadata) for a presentation from the Mediasite Server.

QuerySlides:

Signature	QuerySlidesResponse QuerySlides(QuerySlidesRequest)
Description	Get the slide text, number and timing for slides in a presentation. If a presentation has a large number of slides, it is recommended that multiple calls are made to retrieve slide information. Each call will retrieve information for a set of slides. This method takes in a string identifier (unique Id) that identifies a presentation in the Mediasite database, a start index and a count. The start index and count determine how many slides must be returned from what position.

QueryChapterPoints:

Signature	QueryChapterPointsResponse QueryChapterPoints(QueryChapterPointsRequest)
Description	Get the chapter name, number and timing for chapters in a presentation. If a presentation has a large number of chapters, it is recommended that multiple calls are made to retrieve chapter information. Each call will retrieve information for a set of chapters. This method takes in a string identifier (unique Id) that identifies a presentation in the Mediasite database, a start index and a count. The start index and count determine how many chapters must be returned from what position.

QueryContentServerDetails:

Signature	QueryContentServerDetailsResponse QueryContentServerDetails(QueryContentServerDetailsRequest)
Description	Get the content servers of a given type used the Mediasite Server. Content Servers retrieved could be for a presentation or all content servers of a given type. The content server returned for a presentation contains the base paths for storage and distribution for a given type. To get the actual storage location for the presentation data (slides), append the presentation ID (a unique Id) to the base path. The actual storage location of the video is the base path itself.

QueryPresentationTemplates:

Signature	QueryPresentationTemplatesResponse QueryPresentationTemplates(QueryPresentationTemplatesRequest)
Description	Get the presentation templates from the Mediasite Server. The presentation template can be used to create a presentation.

QueryPresentationTemplateDetails:

Signature	QueryPresentationTemplateDetailsResponse QueryPresentationTemplateDetails(QueryPresentationTemplateDetailsRequest)
Description	Get the presentation template details for the presentation template from the Mediasite Server.

QuerySiteProperties:

Signature	QuerySitePropertiesResponse QuerySiteProperties (QuerySitePropertiesRequest)
Description	Get the site properties from the Mediasite Server.

QueryResourcePermissions:

Signature	QueryResourcePermissionsResponse QueryResourcePermissions (QueryResourcesPermissionsRequest)
Description	Get the resource permissions for a given Mediasite resource from the Mediasite Server.

QueryFoldersWithPresentations:

Signature	QueryFoldersWithPresentationsResponse QueryFoldersWithPresentations (QueryFoldersWithPresentationsRequest)
Description	Get a list of folders that contain presentations from the Mediasite Server.

QuerySubFolderDetails:

Signature	QuerySubFolderDetailsResponse QuerySubFoldersDetails (QuerySubFoldersDetailRequest)
Description	Get a list of sub-folders from a given folder from the Mediasite Server. If the IncludeAllSubFolders flag in the QuerySubFoldersDetailRequest is set then all sub-folders from the given folder are returned as a sub-tree. If this flag is set to false, then all child folders of the given folder are returned.

CreatePresentationFromTemplate:

Signature	CreatePresentationFromTemplateResponse CreatePresentationFromTemplate (CreatePresentationFromTemplateRequest)
Description	Create a scheduled presentation in the Mediasite Server using a presentation template.

Logout:

Signature	LogoutResponse Logout(LogoutRequest)
Description	Log out of the Mediasite Server.

Message Types for Proxy

The External Data Access Service uses a standard Request / Response model. Message types defined in the proxy are discussed in this section.

TestRequest:
Note: Has no properties.

TestResponse:

Properties	Data Type	Description
Value	boolean	If true indicates that the test passed else failed

GetVersionRequest:
Note: Has no properties.

GetVersionResponse:

Properties	Data Type	Description
Version	string	The version of the Mediasite Server

LoginRequest:

Properties	Data Type	Description
Username	string	User Id of the user attempting to log in
Password	string	Password of the user attempting to log in
Impersonate	boolean	Indicates whether a user logging in is impersonating another user.
ImpersonationUsername	string	The User Id of the user being impersonated
ImpersonationRoles	string	The list of roles in the Mediasite System that the impersonated user belongs to

LoginResponse:

Properties	Data Type	Description
UserTicket	string	A ticket issued as a result of a successful login.
ShortName	string	The short name for the user logging in. Normally the display name.
LongName	string	The long name of the user logging in. Normally corresponds to the full DN (distinguishedName) entry in a directory that the user belongs to.
Roles	string	A list of roles that the user belongs to in the Mediasite database.

LogoutRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued as a result of a successful login.

LogoutResponse:
Note: Has no properties.

CreatePresentationRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued in response to a login operation.
CreationDetails	PresentationCreationDetails	An object containing information about the presentation to be created on the Mediasite Server

CreatePresentationResponse:

Properties	Data Type	Description
Presentation	PresentationContext	An object containing information about the presentation created on the Mediasite Server

CreatePresentationFromTemplateRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued in response to a login operation.
CreationDetails	PresentationTemplateCreationDetails	An object containing information about the presentation to be created, based on a presentation template on the Mediasite Server

CreatePresentationFromTemplateResponse:

Properties	Data Type	Description
Presentation	PresentationContext	An object containing information about the presentation created on the Mediasite Server

QueryPresentersRequest:

Properties	Data Type	Description
UserTicket	String	A ticket issued in response to a login operation.

QueryPresentersResponse:

Properties	Data Type	Description
Presenters	PresenterContext[]	An object containing list of presenters available on the Mediasite Server

QueryPresenterDetailsRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued in response to a login operation.
PresenterIdList	String[]	A list of Presenter Id's. A Presenter Id uniquely identifies a presenter residing in the Mediasite database.

QueryPresenterDetailsResponse:

Properties	Data Type	Description
PresenterDetails	PresenterDetails[]	An object containing list of presenter details available on the Mediasite Server. The list of presenter details returned includes all presenters that the user has access to.

QuerySubFolderDetailsRequest:

Properties	Data Type	Description
UserTicket	String	A ticket issued in response to a login operation.
ParentFolderIdList	String[]	A list of Folder Id's. Each folder Id is assumed to be a parent. A Folder Id uniquely identifies a folder residing in the Mediasite database. If any of the folderId's in the string array is set to an empty string, the folder details for the root folder are returned.
IncludeAllSubFolders	String	A boolean flag when set, performs a sub-tree search and returns all the sub-folders for a given folder in the list. When not set, performs a one-level search and returns the immediate sub-folders for a given folder.

QuerySubFolderDetailsResponse:

Properties	Data Type	Description
Folders	FolderDetails[]	An object containing list of folder details available on the Mediasite Server. The list of folder details returned includes all folders that the user has access to.

QueryFolderDetailsRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued in response to a login operation.
FolderIdList	String[]	A list of Folder Id's. A Folder Id uniquely identifies a folder residing in the Mediasite database. If any of the folderId's in the string array is set to an empty string, the folder details for the root folder are returned.

QueryFolderDetailsResponse:

Properties	Data Type	Description
FolderDetails	FolderDetails[]	An object containing list of folder details available on the Mediasite Server. The list of folder details returned includes all folders that the user has access to.

QueryViewerSkinsRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued in response to a login operation.

QueryViewerSkinsResponse:

Properties	Data Type	Description
ViewerSkins	ViewerSkinContext[]	An object containing list of viewer skins available on the Mediasite Server

QueryServerGroupsRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued in response to a login operation.

QueryServerGroupsResponse:

Properties	Data Type	Description
ServerGroups	ServerGroupContext[]	An object containing list of server groups available on the Mediasite Server

QueryStreamingProfilesRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued in response to a login operation.

QueryStreamingProfilesResponse:

Properties	Data Type	Description
StreamingProfiles	StreamingProfileContext[]	An object containing list of streaming profiles available on the Mediasite Server

QueryStreamingProfileDetailsRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued in response to a login operation.
StreamingProfileIdList	String[]	A list of Streaming Profile Id's. A Streaming Profile Id uniquely identifies a streaming profile residing in the Mediasite database.

QueryStreamingProfileDetailsResponse:

Properties	Data Type	Description
StreamingProfileDetails	StreamingProfileDetails[]	An object containing list of streaming profile details available on the Mediasite Server. The list of streaming profile details returned includes all streaming profiles that the user has access to.

QueryTimeZonesRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued in response to a login operation.

QueryTimeZonesResponse:

Properties	Data Type	Description
TimeZones	TimeZoneContext[]	An object containing list of time zones available for use on the Mediasite Server

QueryTimeZoneDetailsRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued in response to a login operation.
TimeZoneIdList	String[]	A list of time zone Id's. A TimeZone Id uniquely identifies a time zone residing in the Mediasite database.

QueryTimeZoneDetailsResponse:

Properties	Data Type	Description
TimeZoneDetails	TimeZoneDetails[]	An object containing list of time zone details available on the Mediasite Server.

QueryPresentationsRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued in response to a login operation.
FolderId	string	The Folder Id uniquely identifies a folder in the Mediasite database.
Filter	PresentationFilter	A filter that allows only presentations matching the filter criteria to be returned.

QueryPresentationsResponse:

Properties	Data Type	Description
Presentations	PresentationContext[]	An object containing list of presentations available in a given folder on the Mediasite Server

QueryAllPresentationsRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued in response to a login operation.
Username	string	The user in the Mediasite database for which the presentations must be retrieved.
RoleList	string[]	The list of roles that the user belongs to, for whom the presentations must be retrieved. If the Username is set and the RoleList is empty, then only those presentations for which the user is the owner are returned. If both Username and RoleList are empty then only presentations with anonymous access are returned.
Filter	PresentationFilter	A filter that allows only presentations matching the filter criteria to be returned.

QueryAllPresentationsResponse:

Properties	Data Type	Description
Presentations	PresentationContext[]	An object containing list of presentations available for a given user in the Mediasite Server

QueryPresentationDetailsRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued in response to a login operation.
ExperienceIdList	String[]	A list of Experience Id's. An Experience Id uniquely identifies a presentation residing in the Mediasite database.

QueryPresentationDetailsResponse:

Properties	Data Type	Description
Details	PresentationDetails[]	An object containing list of presentation details available on the Mediasite Server. The list of presentation details returned includes all presentations that the user has access to.

QuerySlidesRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued in response to a login operation.
ExperienceId	string	An Experience Id uniquely identifies a presentation residing in the Mediasite database.
StartIndex	Int	The start index determines the point from which the slides must be returned.
Count	Int	Count determines the number of slides to be returned.

QuerySlidesResponse:

Properties	Data Type	Description
Slides	Slide[]	An object containing list of slide objects a presentation on the Mediasite Server.

QueryChapterPointsRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued in response to a login operation.
ExperienceId	string	An Experience Id uniquely identifies a presentation residing in the Mediasite database.
StartIndex	Int	The start index determines the point from which the chapters must be returned.
Count	Int	Count determines the number of chapters to be returned.

QueryChapterPointsResponse:

Properties	Data Type	Description
ChapterPoints	ChapterPoint[]	An object containing list of ChapterPoint objects for a presentation on the Mediasite Server.

QueryContentServerDetailsRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued in response to a login operation.
QueryBy	ContentServerQueryBy	Content servers can be queried for a presentation (experience) or a server type
ExperienceId	string	An Experience Id uniquely identifies a presentation residing in the Mediasite database. The ExperienceId is needed only if the content servers are queried by experience.
ServerType	ContentServerType	The type of the content server for which information is requested. The ServerType is needed only if the content servers are queried by server type.
IncludeStorageSettings	boolean	A flag that returns storage URL and storage credentials for a content server.

QueryContentServerDetailsResponse:

Properties	Data Type	Description
Servers	ContentServerDetails[]	An object containing list of storage server objects on the Mediasite Server. The storage server objects could be for a presentation or all content servers of a type.

QuerySitePropertiesRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued in response to a login operation.

QuerySitePropertiesResponse:

Properties	Data Type	Description
Properties	SiteProperties[]	An object containing list of site properties for the Mediasite installation.

QueryPresentationTemplatesRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued in response to a login operation.

QueryPresentationTemplateResponse:

Properties	Data Type	Description
Templates	PresentationTemplateContext[]	An object containing list of presentation templates available in the Mediasite Server

QueryPresentationTemplateDetailsRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued in response to a login operation.
PresentationTemplateIdList	string[]	A list of presentation template Id's. A PresentationTemplate Id uniquely identifies a presentation template residing in the Mediasite database.

QueryPresentationTemplateDetailsResponse:

Properties	Data Type	Description
Details	PresentationTemplateDetails[]	An object containing list of presentation template details available on the Mediasite Server.

QueryResourcePermisisonsRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued in response to a login operation.
Username	string	A user id of the person requesting the permissions on a resource.
RoleList	String[]	A list of roles for whom the permissions need to be returned for a resource
ResourceList	ResourceObjectRequest[]	A list of resources for which the permissions need to be returned.

QueryResourcePermissionsResponse:

Properties	Data Type	Description
ResourceList	ResourceObjectResponse[]	A list of resources with a permission mask.

QueryFoldersWithPresentationsRequest:

Properties	Data Type	Description
UserTicket	string	A ticket issued in response to a login operation.
Username	string	The user in the Mediasite database for which the presentations must be retrieved.
RoleList	String[]	The list of roles that the user belongs to, for whom the presentations must be retrieved. If the Username is set and the RoleList is empty, then only those presentations for which the user is the owner are returned. If both Username and RoleList are empty then only presentations with anonymous access are returned.

QueryFoldersWithPresentationsResponse:

Properties	Data Type	Description
Folders	FolderDetails[]	A list of folders which contain presentations accessible.

Data Types in Proxy

The list of objects available is discussed in this section. Each object is a data type and defined as a class.

MediasiteContext:

Description	This is the base class from which all Context objects are derived.	
Parent Object		
Properties	**Data Type**	**Description**
Id	String	Identifies an object in the Mediasite database.
Name	String	Specifies the friendly name of the object in the Mediasite database. The name is recommended to be less than 255 characters.

PresenterContext:

Description	Information about a Presenter.
Parent Object	**MediasiteContext**

PresenterDetails:

Description	Details about a Presenter.	
Parent Object		
Properties	**Data Type**	**Description**
Id	String	Determines a unique identity of a presenter in the Mediasite database.
Name	String	Specifies the name of the presenter.
ImageUrl	String	Specifies the location of the presenter's image.
Email	String	Contains the email address of the presenter.
BioUrl	String	Specifies the location of the URL for the presenter's biodata.

ViewerSkinContext:

Description	Information about a Viewer Skin.
Parent Object	**MediasiteContext**

ServerGroupContext:

Description	Information about a Server Group.
Parent Object	**MediasiteContext**

StreamingProfileContext:

Description	Information about a Streaming Profile.
Parent Object	**MediasiteContext**

StreamingProfileDetails:

Description	Details about a Streaming Profile.	
Parent Object		
Properties	**Data Type**	**Description**
Id	String	Determines a unique identity of a streaming profile in the Mediasite database.
Name	String	Specifies the name of the streaming profile.
Description	String	Represents the textual description for the streaming profile.
VideoEncodings	VideoEncodingDetails[]	Contains information about audio and video streams along with the encoding settings for each stream in the media file.
AudioEncodings	AudioEncodingDetails[]	Contains information about an audio stream along with the encoding settings for an audio stream in the media file. This is available for streaming profiles that are set up as audio only.

VideoEncodingDetails:

Description	Details about an Audio/Video stream.	
Parent Object		
Properties	**Data Type**	**Description**
Name	String	Specifies the name of the codec.
Description	String	Represents the textual description for the video stream.
Height	Int	Represents the height of the encoded video stream.
Width	Int	Represents the width of the encoded video stream.
EffectiveBitRate	Int	Represents the bitrate of the encoded video stream.
FrameRate	Int	Represents the frame rate of the encoded video stream.
BufferSize	Int	Represents the buffer size to be set up by the windows media player when the encoded video stream is streamed from a media server.
ImageQuality	Int	Represents on a scale of 0 to 100, the quality of the captured frames in the encoded video stream. The value 0 yields the smoothest image while the value 100 yields the clearest image.
KeyFrame	Int	Represents the key frame interval to optimize seeking in the encoded video stream.
AudioEncloding	AudioEncodingDetails	Represents the audio encoding parameters for the associated audio stream.

AudioEncodingDetails:

Description	Details about an Audio stream.	
Parent Object		
Properties	**Data Type**	**Description**
Name	String	Specifies the name of the codec.
Description	String	Represents the textual description for the audio stream.
BitRate	Int	Represents the bit rate of the encoded audio stream.
SampleRate	Int	Represents the sampling rate of the encoded audio stream.
Channels	Int	Represents the number of channels in the encoded audio stream.
EncodingBits	Int	Represents the number of encoding bits for the encoded audio stream.
EncodingMode	AudioEncodingMode	Determines whether the encoding mode of the encoded audio stream.
EncodingType	AudioEncodingType	Determines if the audio stream is associated with a video stream or not.

TimeZoneContext:

Description	Information about a Time Zone.
Parent Object	**MediasiteContext**

TimeZoneDetails:

Description	Details about a Time zone.	
Parent Object		
Properties	**Data Type**	**Description**
Id	String	Determines a unique identity of a time zone in the Mediasite database.
Name	String	Specifies the name of the time zone.
Abbreviation	String	Represents an abbreviation for the time zone and is used for display purposes
Description	String	Represents the textual description for the time zone.
RegistryKey	String	Associates the Mediasite time zone with the corresponding time zone value in the registry.

ContentServerDetails:

Description	Information about content servers. A content server contains locations and/or credentials needed to access the location of the media and/or images.	
Parent Object		
Properties	**Data Type**	**Description**
Id	string	Determines a unique identity of a content server in the Mediasite database
Name	string	Identifies the name of the content server.
ServerType	StorageServerType	Determines the type of the storage server.
ServerConnections	ContentServerConnectionSettings[]	Contains a list of content server connections that contain the URLs and/or credentials for a server type.

ContentServerConnectionSetting:

Description	Information about a content server connection that contains the URLs and/or credentials for a server type.	
Parent Object		
Properties	**Data Type**	**Description**
Protocol	ContentServerProtocol	Determines the protocol used in the ContentServerConnection object
URL	string	Points to the location specified in the ContentServerConnection object.
Username	string	Contains the user id that can log in to the location pointed to by the URL property. This property is set when the protocol is set to FTP or SFTP
Password	string	Contains the password of the user that can log in to the location pointed to by the URL property. This property is set when the protocol is set to FTP or SFTP.

SupportingLink:

Description	Information about a Supporting Link. Supporting links are metadata for a presentation stored in the Mediasite database.	
Parent Object		
Properties	**Data Type**	**Description**
Description	String	Contains the text description of the supporting link that appears when the link is displayed.
Url	String	The actual URL to the supporting material
OrderNumber	int	Identifies the order of the supporting link in a list of supporting links.

AudioLanguage:

Description	Information describing the audio languages available in the presentation. Audio streams must be inserted into the media file during post production and specified using the Mediasite Management Portal.	
Parent Object		
Properties	**Data Type**	**Description**
LanguageId	String	Uniquely identifies the audio language in the Mediasite database.
Name	String	Identifies the name of the audio language.
OrderNumber	int	Identifies the order of the audio language in a list of audio languages.
PresentationDataId	String	Determines the unique identity of the presentation data for which the audio languages are specified.

ChapterPoint:

Description	Information about Chapters. Chapters are metadata for a presentation stored in the Mediasite database that identify seek points into the presentation.	
Parent Object		
Properties	**Data Type**	**Description**
ChapterNumber	int	Determines a unique identity for a chapter in the Mediasite database.
PresentationDataId	String	Determines the unique identity of the presentation data for which the audio languages are specified.
Title	String	Determines the chapter title.
OrderNumber	int	Identifies the order of the chapter in a list of chapters.

PresentationContext:

Description	Information about a Presentation. **Note:** If both properties IsLive and IsOnDemand are set, then the presentation is available on-demand viewing as well as a live broadcast	
Parent Object	**MediasiteContext**	
Properties	**Data Type**	**Description**
Status	PresentationDataStatus	Determines the status of the presentation. The Status is defined as PresentationDataStatus which is an enumeration defined in the Enumerations section.
IsOnDemand	boolean	When true indicates that the presentation is available for on demand viewing only
IsLive	boolean	When true indicates that the presentation is available as a live broadcast only.

PresentationDetails:

Description	Details about a Presentation. **Note**: If both properties IsLive and IsOnDemand are set, then the presentation is available on-demand viewing as well as a live broadcast	
Parent Object		
Properties	**Data Type**	**Description**
Id	String	Uniquely identifies a presentation in the Mediasite database.
Name	String	Determines the name of the presentation.
Description	String	Describes the presentation.
AirDateTime	DateTime	Indicates when the presentation was aired / recorded.
Duration	Int	Indicates the length of the presentation. The duration is specified in milliseconds.
ViewerUrl	String	Specifies the URL to play back a presentation.
PresentationDataId	String	Uniquely identifies data associated with a presentation in the Mediasite database.
SlideCount	Int	Returns the number of slides for a presentation.
ChapterPointCount	Int	Returns the number of chapters for a presentation.
Status	PresentationDataStatus	Determines the status of the presentation. The Status is defined as PresentationDataStatus which is an enumeration defined in the Enumerations section.
IsOnDemand	boolean	When true indicates that the presentation is available for on demand viewing only.
IsLive	boolean	When true indicates that the presentation is available as a live broadcast only.
Presenters	PresenterContext[]	Indicates the list of presenters associated with the Presentation. The first item in the list is the primary presenter. The presenters associated with a presentation are returned only if the user has read permissions.
PrimaryPresenterId	String	This field is now obsolete. It is available only for EDAS version 1.0. Indicates the Id of the primary presenter in the list of presenters returned.
PresenterIdList	String[]	Indicates the list of presenter identifiers associated with the Presentation. The first item in the list is the primary presenter.
StreamingProfileId	String	Identifies the streaming profile in the Mediasite database used to record the presentation.
StreamingProfile	StreamingProfileDetails	The streaming profile used by a presentation is returned if the user has read permissions. If the user does not have permissions the value will be null.
ViewerId	String	Identifies the Viewer in the Mediasite database used to playback the presentation.
PresentationDataUrl	String	Contains the HTTP location of data files for a presentation. Captioning information and Alternate text for slides for a presentation is stored in data files. In version 4.0 chapters in a presentations as well as external links for a presentation a are also stored in a data file.
VideoUrl	String	Contains the distribution URL for the video. The distribution URL is also the video playback URL and typically, the scheme for this URL is mms.
TimeZoneId	String	Identifies a time zone in the Mediasite database. The time zone along with the AirDateTime property uniquely identify the time at which the presentation was recorded.
TimeZoneAbbreviation	String	Represents the abbreviation for a time zone in the Mediasite database.
SupportingLinks	Supporting Link	Contain reference links specified in the Mediasite database for a presentation.
AudioLanguages	AudioLanguage	Contain a list of languages specified in the Mediasite database for a presentation. Each AudioLanguage for a presentation maps to an audio stream in the media file. Additional audio streams added as a post production step.
TargetId	String	Identifies the target The target experience is created when the presentation is first created.
ViewerLayoutFileUrl	String	Identifies the location of the viewer layout file used by a presentation for playback.
FolderId	String	Identifies the folder that the presentation resides in.

Slide:

Description	Information about a Slide.	
Parent Object		
Properties	**Properties**	**Description**
Number	Int	Determines the slide number in a presentation.
Time	Int	Determines the time at which the slide change occurred in the presentation.
Url	String	Specifies the location of the image for the slide.
Title	String	Determines the title of the slide, which is a part of the slide text.
Description	String	Determines the description of the slide, which is a part of the slide text.

FolderDetails:

Description	Information about a Folder.	
Parent Object		
Properties	**Properties**	**Description**
Id	String	Uniquely identifies a folder in the Mediasite database.
Name	String	Determines the name of the folder.
Description	String	Determines the description for a folder.
ParentId	string	Determines the parent of the folder. The Parented of the root folder is set to an empty string
HasChildFolders	bool	Determines if the folder has any children.
Type	FolderType	Determines the type of the folder.

PresentationCreationDetails:

Description	Information needed to create a presentation.	
Parent Object		
Properties	**Data Type**	**Description**
Title	string	Determines the title of the presentation.
Description	string	A brief description of the presentation.
RecordDateTime	DateTime	Indicates when the presentation will be recorded.
Duration	int	Indicates the approximate length of the presentation when a presentation is created. Once a presentation is recorded, this property is automatically updated to the correct value using the length of the presentation.
LiveBroadcast	boolean	Indicates if a presentation will be aired live.
AutomaticUpload	boolean	Indicates if a presentation will be automatically uploaded to the Mediasite Server from the Recorder after it is recorded.
AutomaticViewableStatus	boolean	Indicates if a presentation will be available for viewing from the Mediasite Server immediately after it is uploaded from the Mediasite Recorder.
AllowPollSubmissions	boolean	Indicates if polls can be added to a presentation.
AllowViewingPollResults	boolean	Indicates if poll results can be viewed by the audience in the Mediasite Viewer.
UseQAForum	boolean	Indicates if the audience can ask questions to the presenter using the Mediasite Viewer.
PresenterId	string	Uniquely identifies a presenter in the Mediasite database. This identifier is obtained from a PresenterContext object returned from the call QueryPresenters().
FolderId	string	Uniquely identifies a folder in the Mediasite database. This identifier is obtained from a FolderContext object returned from the call QueryFolders().
ViewerSkinId	string	Uniquely identifies a viewer skin in the Mediasite database. This identifier is obtained from a ViewerContext object returned from the call QueryViewerSkins().
ServerGroupId	string	Uniquely identifies a server group in the Mediasite database. This identifier is obtained from a ServerGroupContext object returned from the call QueryServerGroups().
CDNPublishingPoint	string	The name of the publishing point for a presentation that uses a live media server configured for CDN distribution. The CDNPublishingPoint specified must be manually created on the live media server.
StreamingProfileId	string	Uniquely identifies a streaming profile in the Mediasite database. This identifier is obtained from a StreamingProfileContext object returned from the call QueryStreamingProfiles().
TimeZoneId	string	Uniquely identifies a time zone in the Mediasite database. This identifier is obtained from a TimeZoneContext object returned from the call QueryTimeZones().

PresentationCreationTemplateDetails:

Description	Information needed to create a presentation using a presentation template.	
Parent Object		
Properties	**Data Type**	**Description**
PresentationTemplateId	string	Uniquely identifies a presentation template in the Mediasite database.
Title	string	The title of a presentation to be created.
Description	string	Describes the presentation to be created
RecordDateTime	DateTime	Indicates when the presentation will be recorded.
Duration	int	Indicates the approximate length of the presentation when a presentation is created. Once a presentation is recorded, this property is automatically updated to the correct value using the length of the presentation.
FolderId	string	Determines the folder in which the presentation will be created.
CDNPublishingPoint	string	Determines the name of the publishing point that the presentation to be created will use, if the live server specified in the server group that the presentation template uses is set up for CDN distribution.

PresentationTemplateDetails:

Description	Details about a Presentation Template.	
Parent Object		
Properties	**Data Type**	**Description**
Id	String	Uniquely identifies a presentation template in the Mediasite database.
Name	String	Determines the name of the presentation template.
AllowPollSubmissions	boolean	Indicates whether poll submission is allowed for presentations that are created using the presentation template.
AllowViewingPollResults	boolean	Indicates whether poll results can be viewed by the audience playing back a presentation that is created using the presentation template.
UseQAForum	boolean	Indicates whether question submission by the audience playing back the presentation is allowed for a presentation that is created using the presentation template.
LiveBroadcast	boolean	Indicates whether the presentation that is created using the presentation template will be broadcast live.
AutomaticUpload	boolean	Indicates whether the presentation that is created using the presentation template will have its content automatically transferred to the Mediasite Server's data store after recording.
AutomaticViewableStatus	boolean	Indicates whether the presentation that is created using the presentation template will be automatically published for playback on the Mediasite Server after recording.
AudioOnly	boolean	Indicates whether the presentation that is created using the presentation template will be an audio only presentation.
TimeZoneId	String	Indicates a unique time zone in the Mediasite database that is set for the presentation created using the presentation template.
ServerGroupId	String	Indicates a unique server group in the Mediasite database that is set for the presentation created using the presentation template.
StreamingProfileId	String	Indicates a unique streaming profile in the Mediasite database that is set for the presentation created using the presentation template.
ViewerSkinId	String	Indicates a unique viewer skin in the Mediasite database that is set for the presentation created using the presentation template.
Presenters	PresenterContext[]	Indicates a list of presenters specified for the presentation created using the presentation template.

SiteProperties:

Description	Details about Site Properties.	
Parent Object		
Properties	**Data Type**	**Description**
SiteId	String	Uniquely identifies a Mediasite.
Name	String	Determines the name of the Mediasite.
Description	string	Determines the description of the Mediasite
Edition	string	Determines the edition of the Mediasite EX Server
Version	string	Determines the version of the Mediasite EX Server
BuildNumber	string	Determines the build number of the Mediasite EX Server
Owner	string	Determines the name of the Mediasite System Owner
OwnerContact	string	Determines the contact information of the owner
OwnerEmail	string	Determines the email address for the owner

PresentationFilter:

Description	Allows setting a filter for methods attempting to retrieve presentations.	
Parent Object		
Properties	**Data Type**	**Description**
StatusFilterList	PresentationDataStatus[]	Return presentations with the presentation status(es) provided
TitleRegEx	string	Return presentations whose title match the regular expression provided
PermissionMask	Int	Return presentations with the corresponding permission mask. The permission mask is defined as follows: None=0, Read=0x01, Write=0x02, Execute=0x04
FolderMask	int	Return presentations with the corresponding folder mask. The folder mask is defined as follows: Normal=0x01, RecycleBin=0x02

ImpersonationCredentials:

Description	Contains impersonated user's credentials needed to access data in the Mediasite System.
Parent Object	

Properties	**Data Type**	**Description**
Username	string	Username of the impersonated user
RoleList	string[]	List of roles that the impersonated user belongs to.

Enumerations in Proxy
The List of enumerations (enum) available is discussed in this section.

PresentationDataStatus:

Value	**Description**
Scheduled	The presentation is scheduled on the Mediasite Server and is ready to be recorded
OpenForRecord	A scheduled presentation is opened for recording on a Mediasite Recorder
Recording	The presentation is being recorded by a Mediasite Recorder
Recorded	The presentation is recorded by a Mediasite Recorder and is ready to be uploaded
Uploaded	The recorded presentation on the Mediasite Recorder is uploaded to the Mediasite Server and available for on-demand playback

ContentServerType:

Value	Description
ReplayVideo	The content server specified is for On-demand media streams
LiveVideo	The content server specified is for live media streams
PresentationData	The content server specified is for slide data, presentation links and any other information that is secure. There is no direct HTTP URL to this location.
Publishing	The content server specified is for the publishing location that stores graphics and publishing information for the Viewer and Catalog.
Presenters	The content server specified for presenter data
Presentations	The content server specified contains information about presentations that can be accessed using a direct HTTP URL. Multicast files are stored in this location.

ContentServerProtocol:

Value	Description
Unknown	Protocol scheme not known.
Ftp	Protocol scheme is Ftp
Http	Protocol scheme is Http or Https
File	Protocol scheme is File
Mms	Protocol scheme is Mms
Sftp	Protocol scheme is Sftp (secure FTP)

ContentServerQueryBy:

Value	Description
Experience	All content servers for a given experience are returned.
ServerType	All content servers of a given type are returned

AudioEncodingType:

Value	Description
Audio	Encoding type is audio only. No video stream associated.
AudioVideo	Encoding type is audio-video. A video stream uses the audio stream.

AudioEncodingMode:

Value	Description
ConstantBitRate	Audio encoding mode uses a constant bit rate.
VariableBitRate	Video encoding mode uses a constant bit rate.

FolderType:

Value	Description
Folder	Folder returned has a parent. Does not include the recycle bin.
Root	The root node in the folder hierarchy.
RecycleBin	The recycle bin folder

ResourceType:

Value	Description
PresentationData	Permissions requested are for a resource type of presentation data. Presentation data maps to the presentation content and metadata.
PresentationExperience	Permissions requested are for a resource type of presentation experience. Presentation experience maps to the presentation that can be played back.
StreamingProfile	Permissions requested are for a resource type of Streaming Profile.
ServerGroup	Permissions requested are for a resource type of Server Group.
Presenter	Permissions requested are for a resource type of Presenter.
Viewer	Permissions requested are for a resource type of Viewer.
PresentationTemplate	Permissions requested are for a resource type of PresentationTemplate.
SystemOperation	Permissions requested are for a resource type of ServerOperation.
PortalResource	Permissions requested are for a resource type of PortalResource.
Folder	Permissions requested are for a resource type of Folder.