

Diseño, desarrollo y puesta en servicio de una aplicación
web para la gestión de salas virtuales de videoconferencia
Adobe Connect[®]

21 de mayo de 2013

A mi familia

Índice general

1. Introducción	4
1.1. Objetivos	6
1.2. Organización	6
2. Tecnologías empleadas	8
2.1. <i>Adobe Connect®</i> Profesional	8
2.2. SYMFONY	9
2.3. PHP	16
2.4. Apache	16
2.5. Javascript	18
2.6. MySQL	20
2.7. HTML 5 y CSS 3	22
3. Decisiones de diseño	23
3.1. <i>Adobe Connect®</i>	23
3.2. Symfony	25
3.3. PHP	26
3.4. Apache	26
3.5. MySQL	27
4. Implementación del proyecto	29
4.1. Archivos de configuración	30
4.2. Concepto de sala	31
4.3. CRON	34
4.4. Diseño de la base de datos	35
4.5. Especificación de la organización	41
4.6. El interfaz de usuario	42
4.7. URLs de acceso a la aplicación	62
4.8. Interfaz de <i>Adobe Connect®</i>	73
4.9. Directorio +	73
4.10. Seguridad de la Aplicación	75
4.11. Comandos para una gestión de mayor comodidad	76
4.12. Servicios de la aplicación	78

4.13. Controladores	82
4.14. Estilo de la aplicación con CSS y HTML	89
4.15. Funciones Javascript	90
5. Conclusiones y líneas Futuras	93
5.1. Conclusiones	93
5.2. Líneas Futuras	94
Bibliografía	95

Capítulo 1

Introducción

La comunicación humana es uno de los mecanismos que otorgan al ser humano una ventaja fundamental sobre el resto de seres que poblan nuestro planeta y uno de los motivos de su rápida evolución en comparación con otras criaturas inteligentes. En la figura 1.0.1, se puede observar un sintético diagrama sobre los elementos que componen la comunicación y en él se ha resaltado el canal de comunicaciones por ser uno de los temas que fundamentan esta escuela y, en esencia, el argumento central y motivación de este proyecto. Sin lugar a dudas el canal de comunicación ha cambiado desde que Claude Elwood Shannon, considerado el padre de la teoría de la información, publicó en 1948 “Una Teoría Matemática de la Comunicación”. Unido a la invención del transistor, provocó que se multiplicase la utilización del espectro radioeléctrico para la comunicación y con el tiempo que la red de redes llegase a convertirse en el engendro que es ahora. Muchos otros han colaborado en la evolución de la comunicación, pero destacamos este mecanismo para corregir errores de transmisión que es sin duda un gran paso en el avance de la comunicación.

El ser humano prefiere por naturaleza comunicarse cara a cara, ya que muchos otros factores influyen en la comunicación, como gestos, expresividad del interlocutor y una serie de conductas innatas que dan, a la comunicación presencial, preeminencia sobre meras trasmisiones de voz. Debido a esto, a lo largo de los años se ha trabajado en la evolución de las comunicaciones para llegar al estado actual en el que la palabra videoconferencia resulta algo cotidiano. Sin embargo, los primeros intentos en la carrera espacial no resultaron asequibles para el resto de la población debido a la mala calidad de imagen y la falta de técnicas eficientes de compresión de vídeo. No sería hasta 1980 cuando las redes digitales de transmisión de telefonía, como RDSI, lo hicieron posible asegurando una velocidad mínima (por lo general 128 kbits/s) para vídeo comprimido y transmisión de audio.

Estos mecanismos de comunicación por videoconferencia comenzaron a ser una herramienta esencial para el correcto desarrollo empresarial, ya que el aprendizaje y comunicación entre sedes alejadas territorialmente, ya sea de la misma empresa o de empresas del mismo sector, son sin duda fundamentales para el desarrollo ágil y la búsqueda de soluciones.

El único inconveniente es el periodo de adaptación desde que se pone al alcance del ciudadano de a pie una nueva tecnología y el momento en el cual es utilizada de forma innata hasta llegar a originar una revolución. El avance de la tecnología siempre ha estado ligado

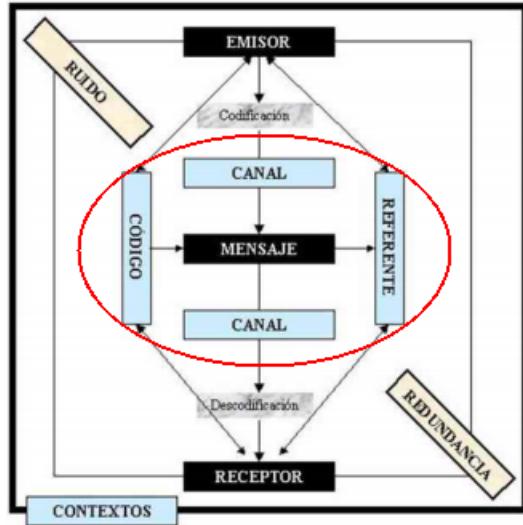


Figura 1.0.1: Diagrama de comunicación

al avance de los interfaces de usuario. Una herramienta útil, pero poco amigable y difícil de utilizar seguramente caerá en desuso. Por este motivo, adaptar las nuevas tecnologías al ser humano es una guerra en la que aún se lucha por ganar. Para este proyecto se ha jugado con la idea de acercar una sala de videoconferencia al propio escritorio de cada usuario. Se ha intentado mantener la idea de puerta de entrada y se ha multiplicado el significado de la palabra llave para dicha puerta.

Para llevar a cabo la puesta en servicio del proyecto se ha contando con un escenario inmejorable, el Campus de Excelencia “Campus do Mar” de Vigo. En este ámbito nos encontramos con colaboraciones entre universidades situadas en diferentes comunidades autónomas. Multitud de universidades españolas están trabajando en la actualidad para obtener una aplicación que agilice, gestione y cuantifique la reserva de salas virtuales, como es el caso de la aplicación “Mr. Bosy” en la Universidad Carlos III de Madrid. Le ofrecemos el siguiente enlace para acceder al contenido público disponible para esta aplicación: <https://mrbosy.uc3m.es/>.

El diseño visual de este proyecto toma prestada la estética de “Campus do Mar” para facilitar la comprensión al lector de las funciones realizadas, ya que ha sido el primer lugar donde se ha desplegado esta aplicación. Por el mismo motivo, se ha dotado a la aplicación con el nombre “eMeeting”¹, que resulta atractivo al usuario y provee a la aplicación de un arma más para ser utilizada.

¹Por comodidad nos referiremos a la aplicación con este término a lo largo de la memoria.

1.1. Objetivos

El presente proyecto surge de la necesidad de una plataforma de videoconferencia que se adapte a las necesidades y perfiles de usuarios de organizaciones concretas y facilite el uso de herramientas de videoconferencia a usuarios que no posean un perfil técnico.

Todo *software* de videoconferencia intenta brindar el mayor número de funcionalidades para copar el mercado y conseguir una gran cartera de clientes, por contrapartida, este hecho conlleva a una saturación del interfaz de usuario, lo cual dificulta su utilización en casos concretos. Alcanzar una alta proyección y ofrecer un buen servicio son algunos de los objetivos que se pretenden alcanzar con este proyecto. Aunque una de las más importantes metas ha sido diseñar una estructura de aplicación que permita adaptar o crear nuevas funcionalidades para los constantes cambios de los clientes potenciales para esta aplicación. Esta es una de las principales razones que nos llevan a desarrollar este proyecto.

Entre las funcionalidades más destacadas que implementaremos podemos resaltar la reserva de salas, múltiples posibilidades de disposición de los elementos de la sala, es decir, se podrá seleccionar el lugar dónde aparecerán los ponentes en la reunión y el contenido visual que aporten, también el acceso a gráficas estadísticas que muestran en tiempo real el uso de la aplicación o un servicio de alerta que advierte a los administradores de una posible saturación del sistema, ya sea por exceso de usuarios o por escasez de salas.

La búsqueda de una aplicación popular tan ansiada por todo desarrollador web nos obligará a desarrollar un interfaz intuitivo, de forma que alcancemos el objetivo de una rápida integración de los usuarios sin necesidad de conocimientos técnicos o lecturas extensas.

Este proyecto forma parte de un proyecto mayor en el que diferentes aplicaciones formarán parte de una macroaplicación que dará servicio al mencionado “Campus do Mar”. Por este motivo, cada aplicación que forme parte del proyecto debe incluir un servicio que sincronice las bases de datos de las diferentes aplicaciones que engloba el “Campus do Mar” como pueden ser un calendario o un gestor de tareas, de manera que todas las aplicaciones formen parte de la macroaplicación y se comuniquen entre sí. Este servicio consigue que nuestra aplicación sea más completa, incluyendo la opción de integración de una forma sencilla.

1.2. Organización

Después de esta breve introducción comenzaremos con un capítulo que sumergirá al lector en el mundo de las tecnologías que se han empleado para alcanzar los objetivos del proyecto. Se hablará de los motivos que nos han llevado a utilizar unas en detrimento de otras, siempre siguiendo un criterio de eficiencia temporal, desarrollo ágil, modularidad y apoyo por parte de la comunidad libre.

Continuaremos con una descripción del diseño de nuestra aplicación, construyendo el camino fielmente, desde el primer boceto del interfaz a los complejos entramados de la base de datos, incluyendo gráficas ilustrativas que aportarán luz sobre la enmarañada relación interna de las entidades del proyecto. Incluiremos capturas del interfaz de usuario con el

estilo diseñado para el despliegue en “Campus do Mar”. El departamento de tecnologías de la información y plataforma digital de “Campus do Mar” posee gran cantidad de recursos y por ello se han llevado a cabo importantes desarrollos tecnológicos en los que se encuentra integrado este proyecto. La importancia de este departamento es tal que posee nombre propio: “DigiMar” y que se utilizará como referencia a lo largo de este proyecto.

Se describirán las posibles líneas futuras desprendidas del desarrollo de la aplicación y de las experiencias de los usuarios que la han utilizado en la senda del desarrollo eficiente. Se comentarán las ideas surgidas durante el desarrollo y que han aportado grandes dosis de conocimiento e intuición para ampliar algunas de las funciones de la aplicación.

Se finalizará con la bibliografía empleada en el desarrollo del proyecto y de esta memoria para el deleite de usuarios que sientan predilección por el mundo del desarrollo web y sientan el deseo de aumentar sus conocimientos en este ámbito.

Capítulo 2

Tecnologías empleadas

Toda aplicación web y el trabajo de los desarrolladores tienen unas premisas comunes, tales como facilitar la interacción usuario-*software* y, sacando el mayor partido a las tecnologías disponibles, conseguir resultados adecuados en el menor tiempo posible. Para este fin utilizaremos como motor de videoconferencia el *software* propietario *Adobe Connect®* y su API, como motor de administración de bases de datos hemos optado por MySQL, para el desarrollo de la aplicación nos hemos declinado por el uso del *framework* Symfony específico para PHP, para las partes dinámicas que se ejecutan en el explorador hemos decidido utilizar Javascript y su API jQuery y jQuery UI, para la estructura que se visualiza en el explorador HTML 5 y para el estilo CSS 3. Todas estas tecnologías a excepción de *Adobe Connect®*, poseen licencia libre, lo cual ayuda al programador en el desarrollo y la búsqueda de soluciones.

2.1. *Adobe Connect®* Profesional

Ahora más que nunca, la gente necesita la capacidad de colaborar de forma eficiente con colegas, socios y clientes en todo el mundo, a través de dispositivos conectados a la red. Más y más organizaciones, incluyendo grandes empresas y agencias del gobierno, están utilizando *Adobe Connect®* para conducir extremo a extremo, flujos de trabajo críticos para sus negocios, reuniones web, *e-Learning* y seminarios. *Adobe Connect®* ofrece interacciones excepcionalmente ricas y permite a las organizaciones, desde el Departamento de Defensa de EE.UU. a las corporaciones líderes, incluyendo Toshiba America o Teltek Vídeo Research, S.L., mejorar sustancialmente su productividad.

El *software* propietario *Adobe Connect®* posee las características más avanzadas de videoconferencia, ofreciendo un interfaz perfecto para los diferentes tipos de comunicación, ya sea a nivel académico o profesional.

El sistema de comunicaciones web *Adobe Connect®* está compuesto por *Adobe Connect Enterprise Server®* (versión con licencia) o *Adobe Connect Enterprise Hosted®* (servicio alojado) y tres módulos, *Adobe Connect Training®*, *Adobe Connect Events®* y *Adobe Presenter*, que brindan apoyo a las funciones de *Adobe Acrobat Connect Professional* y las

amplían para proporcionar soluciones de comunicaciones web de formación, *marketing* y ventas. Este *software* propietario es sin duda uno de los más avanzados del mercado y su completa API nos proporciona la posibilidad de crear aplicaciones web de gran calidad aprovechando las características que más convengan a los usuarios finales.

Adobe Connect® permite a los invitados asistir a sus reuniones fácilmente desde el escritorio sin necesidad de un cliente de descarga, aunque en nuestro caso accederemos a través del explorador, y ofrece un completo interfaz entre dispositivos móviles para incrementar las capacidades de colaboración y hacer frente a las realidades de los actuales entornos empresariales, donde los empleados y los clientes están en continuo movimiento. Este complejo entramado en que se ha convertido la vida laboral e incluso privada de las personas, de la mano de las nuevas tecnologías de la comunicación y la información, hace indispensable la utilización de *software* de estas características, donde uno de los factores más importantes es el ahorro de tiempo y dinero. En un mundo sustancialmente capitalista como el actual, son dos pilares básicos para la sostenibilidad del sistema y como no podía ser de otro modo, en tiempos de crisis cualquier ventaja puede ser la diferencia entre el éxito y el fracaso.

La experiencia del usuario ha desvelado que el uso de *Adobe Connect Profesional®* es complejo para cualquier usuario que no posea un perfil técnico y carece de las necesidades específicas de los usuarios de “Campus do Mar”, este hecho provoca el nacimiento de esta aplicación.

2.2. SYMFONY

En las siguientes líneas abordaremos la solución a un problema mediante un *framework* específico para PHP y destinado al desarrollo de aplicaciones web. Una gran lista de aplicaciones se han desarrollado bajo el *framework* Symfony como Motilee (para crear foros de discusión), SteerCMS (gestor de contenidos), Hashbin (facilita la depuración colaborativa de las aplicaciones), Symfonians (punto de encuentro de usuarios y desarrolladores de Symfony), LiveChat (añadir a un sitio web chats en tiempo real), Lichess (servidor de juegos de ajedrez) [6].

Symfony es un *framework* PHP que facilita la organización y el desarrollo de las aplicaciones web, es decir, se encarga de todos los aspectos comunes y aburridos, dejando que el programador se dedique a aportar valor, desarrollando las características únicas de cada proyecto. En este proyecto se ha utilizado la segunda versión de este *framework* que posee una importante cantidad de mejoras tanto a nivel de desarrollador como internas, cabe resaltar el aumento de la facilidad de migración a versiones superiores mucho menos trabajado en el pasado. Es por ello que las referencias a Symfony pertenecen en todo el proyecto a su segunda versión, por tanto tendremos partes comunes y mejoras con respecto a versiones anteriores.

Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de *software* que modela las relaciones generales de las entidades del dominio, y provee una estructura y una especial metodología de trabajo,

la cual extiende a las aplicaciones del dominio. Las principales herramientas de Symfony imprescindibles para este proyecto se describen a continuación.

2.2.1. El controlador

En esta sección mostraremos cómo se organizan y qué son los controladores en la estructura de Symfony.

Se llama controlador al conjunto de directorios, en nuestro caso situados en la carpeta *src/Cmar/MeetingBundle*, que almacenan la lógica de los diferentes elementos de nuestra aplicación. Es, por así decirlo, el cerebro de cada una de las funciones que se realizan. El controlador recoge los datos de los formularios y de la base de datos y ofrece las respuestas que se visualizarán en las plantillas Twig, que describiremos en la sección 2.2.3.

Nos ofrecen la posibilidad de generar respuestas, como páginas HTML o ficheros XML para fuentes RSS o servicios Web, y JSON para peticiones AJAX, ajustando únicamente el valor de la variable *format* en la descripción de las funciones del controlador para las cuales necesitemos un tipo de respuesta u otra. Posee instrucciones específicas para acceder al objeto *request*¹, o de respuesta de un formulario, a través de una variable llamada *app.request[6][3]*.

Symfony posee un elegante objeto de sesión que representa el cliente (puede ser una persona real usando un navegador, un *bot* o un servicio web). Entre dos peticiones, Symfony almacena los atributos en una cookie usando sesiones nativas de PHP. De esta forma podemos generar funciones del tipo migas de pan de forma sencilla utilizando los atributos de esta *cookie* o generar acceso a un servidor evitando peticiones superfluas si aún no ha caducado la sesión.

2.2.2. Arquitectura

Uno de los principales beneficios de la utilización de un *framework* completo es normalizar los desarrollos. Gracias a la estructura predeterminada de archivos y directorios de Symfony, cualquier desarrollador con conocimientos básicos puede asumir el mantenimiento de un proyecto elaborado con este *framework*, premisa muy importante para el desarrollo colaborativo de todas las aplicaciones.

Adjuntamos una breve descripción de la funcionalidad de cada directorio de Symfony en la figura 2.2.2.

¹Se utiliza nomenclatura PHP para esta variable.

`app/`: La configuración de la aplicación;
`src/`: El código PHP del proyecto;
`vendor/`: Las dependencias a terceros;
`web/`: El directorio web raíz.

Figura 2.2.2: Funcionalidades de la estructura de directorios en Symfony

```
www/ <- your web root directory
  Symfony/ <- the unpacked archive
    app/
      cache/
      config/
      logs/
      Resources/
    bin/
    src/
      Acme/
        DemoBundle/
          Controller/
          Resources/
        ...
    vendor/
      symfony/
      doctrine/
    ...
  web/
    app.php
    ...
```

Figura 2.2.1: Estructura de directorios en Symfony

- `web/`: El directorio web raíz es el hogar de todos los archivos públicos y estáticos como imágenes, hojas de estilo y archivos JavaScript. También es donde reside cada controlador frontal. Un pequeño script PHP cuyo trabajo es arrancar la aplicación. El kernel requiere primero el archivo `bootstrap.php`, el cual inicia el *framework* y registra al *autoloader*. Como cualquier controlador frontal, `app.php` usa una clase *Kernel*, llamada `AppKernel`, para iniciar la aplicación.
- `app/`: La clase `AppKernel` es el punto de ingreso principal a la configuración de la aplicación y como tal, se ubica en este directorio.
 - La velocidad de Symfony se debe a su sistema de caché. La configuración de la aplicación sólo es analizada durante el primer *request* y luego compilada a código PHP que se guarda en el directorio `app/cache/`.
 - Durante el desarrollo de una aplicación web, las cosas pueden salir mal de muchas maneras. Los archivos de log en el directorio `app/logs/` de la aplicación nos ofrecen todo sobre los *requests* para arreglar el problema rápidamente.

- **vendor/** : Las librerías de terceros deberían estar ubicadas en este directorio que contiene inicialmente las bibliotecas de Symfony, SwiftMailer, el ORM Doctrine, el sistema de plantillas Twig y una selección de clases de *Zend Framework*, pero esto es sólo una convención siendo posible ubicarse en un directorio global del servidor o en uno local en cada proyecto.
- **src/** : Para describir este directorio es imprescindible mencionar el sistema de *bundles*, una de las más grandiosas y poderosas características de Symfony. Un *bundle* es similar a un *plugin* en otros lenguajes. Recibe el nombre de *bundle* y no de *plugin* porque *todo* es un *bundle* en Symfony. Esto le da la flexibilidad de usar características pre-construidas empaquetadas en *bundles* de terceros o distribuir sus propios *bundles*. Cada *bundle* puede ser adaptado por medio de archivos de configuración escritos en YAML, XML o PHP. Cada entrada llamada *framework* en el archivo de configuración define la composición para un *bundle*. Y a su vez cada entorno dentro del directorio *src/* puede sobreescribir la configuración por omisión si se provee de un archivo de configuración específico [15]. Además de ser una buena manera de organizar y configurar el código, un *bundle* puede extender otro, ya que los *bundles* soportan herencia. Esto le permite sobreescribir un *bundle* existente para adaptar sus controladores, plantillas y cualquier archivo que contenga. Es aquí donde los nombres lógicos resultan útiles ya que abstraen al programador del lugar en donde se encuentra realmente ubicado el recurso. Por este motivo como veremos en el apartado de diseño, los controladores tendrán nombres que hacen referencia a la función que desempeñan. A su vez Symfony crea una estructura de directorios por cada aplicación que definimos con el comando Symfony *generate:app*.

La estructura de directorios de nuestra aplicación parte de la base estructural de Symfony que acabamos de describir en la figura 2.2.1. A esta base estructural se le añade, en sus correspondientes directorios, los componentes que nuestra aplicación necesita, de forma que obtenemos una estructura de directorios como la que se muestra en la figura 2.2.3.

El contenido y la razón de la existencia de los directorios específicos para nuestra aplicación serán descritos con detalle en el capítulo 3.

2.2.3. La vista

- La plantilla TWIG

Twig es el motor de plantillas para PHP que utiliza Symfony. Sus características principales son:

- Rápido: Ofrece la posibilidad de optimizar las llamadas a los contenidos de las variables que PHP envía a las plantillas HTML. El tiempo de generación de las plantillas, en comparación con el habitual código PHP, se ve reducido a la mínima expresión.
- Seguro: Tiene un modo para evaluar código en la plantilla que no es de confianza. Esto permite utilizarlo como un lenguaje de plantillas para aplicaciones en las que

los usuarios pueden modificar el diseño de la plantilla. El objetivo de modularidad de nuestro proyecto agradece enormemente el uso de este motor de plantillas.

- Flexible: Posee un analizador léxico flexible. Esto permite al desarrollador definir sus propias etiquetas y filtros personalizados, y crear su propio lenguaje específico del dominio (DSL) [14].

Todas las plantillas utilizadas por la aplicación eMeeting se han realizado con este formato, optimizando la compilación y facilitando el acceso a los contenidos de las variables de la base de datos utilizadas en la lógica implementada con PHP.

Con estas plantillas también es muy sencillo incluir otras plantillas para evitar la redundancia de código o incluir URIs que hacen referencia a páginas de nuestro propio código. Las plantillas para cada sección de nuestra aplicación se alojan en la carpeta `src/Cmar/MeetingBundle/views`.

2.2.4. Doctrine

El Proyecto Doctrine consta de un conjunto de bibliotecas PHP seleccionadas principalmente para ofrecer servicios de persistencia y funcionalidades en las relaciones sobre las bases de datos que utilicemos. Entre sus principales características cabe destacar el asignador relacional de objetos (ORM) y la capa de abstracción de base de datos.

Esta herramienta de Symfony nos permite actuar sobre la base de datos de una forma sencilla y en cualquier parte del código ya que permite abstraer al programador del motor de base de datos que utilicemos. Esta característica nos permite aumentar la modularidad de nuestro proyecto y poder desplegarlo sobre el motor de base de datos que posea el usuario final sin modificar apenas líneas de código. Sin duda, alguna de las consultas específicas y complejas que realice nuestra aplicación deberán ser reescritas para adaptarse al nuevo entorno, pero se intentará que sean ínfimas para cumplir con la modularidad propuesta en los objetivos del proyecto [6].

2.2.4.1. Asignador Relacional de Objetos

El asignador relacional de objetos (ORM) para PHP se encuentra en la parte superior de una poderosa abstracción en la capa de la base de datos (DBAL). Una de sus principales características es la opción de escribir las consultas de bases de datos en un dialecto propio orientado a objetos llamado Lenguaje SQL de consulta de Doctrine (DQL), inspirado en Hiberna HQL. Esto proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad sin necesidad de duplicar el código innecesariamente [6].

2.2.4.2. Capa de abstracción de la base de datos

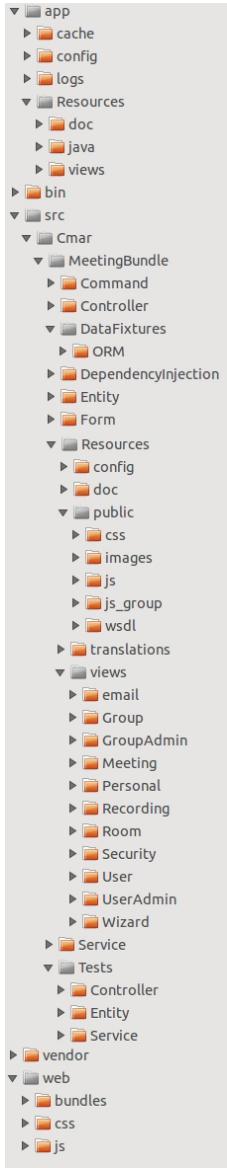


Figura 2.2.3: Estructura de directorios de eMeeting

para cada ruta de la aplicación. Como veremos en el capítulo 4, las rutas de Symfony cobran

Posee múltiples características para la introspección en el esquema de la base de datos, la administración de esquemas y la abstracción PDO.

La abstracción de base de datos de Doctrine y la capa de acceso (DBAL) ofrece una capa ligera alrededor de un API de PDO-like y multitud de características adicionales, como la introspección horizontal en el esquema de la base de datos y la manipulación a través de una API orientada a objetos.

El hecho de que la DBAL de Doctrine abstraiga del pasado API DOP mediante el uso de interfaces que se parecen mucho a la actual API de PDO, hace posible la implementación de los controladores personalizados que pueden utilizar las API's nativas existentes o las realizadas por el propio programador [6].

Para optimizar y aprovechar al máximo el uso de Doctrine se deben describir las tablas de la base de datos con el formato específico adoptado por Symfony en sus desarrollos para el ORM llamado *YAML*, que pasamos a describir a continuación.

2.2.5. El Formato YAML

De acuerdo con el sitio web oficial YAML “es una serialización de datos estándar muy amigable para todos los lenguajes de programación”. Dicho de otra manera, YAML es un lenguaje sencillo para describir los datos (*strings, integers, dates, arrays y hashes*).

En YAML, la estructura se muestra a través de la sangría, la secuencia de elementos se denotan por un guión, y los pares clave/valor están separados por dos puntos. YAML también tiene una sintaxis abreviada para describir la misma estructura con menos líneas, donde los arrays explícitamente se muestran con [] y los hashes o array asociativos con {} [8].

El framework Symfony lo utiliza ampliamente para sus archivos de configuración o para la creación de tablas y estructuras de Doctrine. Por ejemplo en los ficheros *security.yml* y *routing.yml* que mencionaremos en la sección 4.10. Estos ficheros siguen la estructura típica de un fichero YAML y la relación entre ambos ofrece la seguridad

especial importancia ya que están asociadas a funciones, en caso de que sea necesario, y la seguridad de alguna de ellas puede ser crítica.

Gracias al archivo de descripción de la base de datos *schema.yml*, podemos utilizar más ampliamente las simplificaciones que ofrece Doctrine al generar los comandos SQL necesarios para crear las tablas en nuestra base de datos.

Describimos, a modo de ejemplo, alguna de estas tareas:

- Para generar el SQL se debe construir el modelo a partir del contenido del archivo *schema.yml*.

```
php symfony doctrine:build-model
```

- Ahora que los modelos existen se puede generar e insertar el SQL.

```
php symfony doctrine:build-sql
```

- La tarea *doctrine:build-sql* genera comandos SQL en el directorio *data/sql/*, optimizado para el motor de base de datos que se ha configurado, en nuestro caso MySQL.

Gracias a esta característica, cualquier cambio en la base de datos, como añadir un nuevo campo a una entidad, lo que equivale a una nueva columna en la tabla correspondiente, con su valor es una acción simple y que Doctrine realizará de forma limpia y sin hacer peligrar el contenido de la base de datos. Ofrece funcionalidades como visualizar qué comandos específicos para la base de datos se realizarán con el cambio implementado, evitando así pérdidas de datos o acciones que traerán consecuencias irreversibles.

2.2.6. Las entidades

Estos elementos de Symfony guardan la estructura de las piezas del juego en nuestra aplicación. A partir de la estructura de las entidades, Doctrine construirá los elementos que llenarán las tablas de la base de datos y aportarán las características y relaciones que introduzcamos en las cabeceras de estos elementos.

Las entidades aportan a nuestro proyecto una gran modularidad ya que podremos insertar un nuevo elemento en nuestra base de datos o un nuevo campo en el elemento preciso para incrementar las funcionalidades o complementar las existentes si se queda en el tintero alguna que se descubra en el proceso de puesta en servicio. También nos ofrece facilidades a la hora de corregir errores en el proceso de diseño o cuando la aplicación está ya en producción.

2.3. PHP

PHP tal y como se conoce hoy en día es en realidad el sucesor de un producto llamado PHP/FI. Creado en 1994 por Rasmus Lerdorf.

La primera encarnación de PHP era un conjunto simple de ficheros binarios *Common Gateway Interface* (CGI) escritos en el lenguaje de programación C. Originalmente utilizado para rastrear visitas de su currículum online, llamó al conjunto de *scripts* “*Personal Home Page Tools*”, más frecuentemente referenciado como “*PHP Tools*”.

En junio de 1995, Rasmus publicó el código fuente de *PHP Tools*, lo que permitió a los desarrolladores usarlo como considerasen apropiado. Esto animó a los usuarios y permitió que proporcionasen soluciones a los errores del código [9].

Es un lenguaje interpretado orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos. El código fuente escrito en PHP es invisible al navegador web y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable. Aunque puede ser usado también desde un interfaz de línea de comandos o como aplicación de escritorio. Implementa capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL. Posee la capacidad de expandir su potencial utilizando módulos. Permite aplicar técnicas de programación orientada a objetos. No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución. PHP5 incorpora manejo de excepciones, una gran herramienta para el desarrollo de aplicaciones complejas orientadas a la interacción con usuarios, tales como aplicaciones web.

Si bien PHP no obliga a quien lo usa a seguir una determinada metodología a la hora de programar, el programador puede aplicar en su trabajo cualquier técnica de programación o de desarrollo que le permita escribir código ordenado, estructurado y manejable. Un ejemplo de esto son los desarrollos que en PHP se han hecho del patrón de diseño Modelo Vista Controlador (MVC), que permiten separar el tratamiento y acceso a los datos, la lógica de control y la interfaz de usuario en tres componentes independientes.

2.4. Apache

Para la puesta en servicio de la aplicación se ha optado por la opción de Apache 2. Utilizando PHP para el desarrollo es obvio el uso de Apache ya que está implementado como un módulo de forma que aprovecha la máxima integración con el servidor y velocidad posible.

El servidor HTTP Apache es un servidor web libre y de código abierto, el más popular en cuanto a uso, sirviendo de facto como plataforma de referencia para el diseño y evaluación de otros servidores web. Pero en la actualidad está incrementando su utilización para uso comercial.



Figura 2.3.1: Procesos que describen PHP



Figura 2.4.1: Arquitectura de directorios de Apache

Los directorios de Apache albergan funciones de muy diferentes tipos como configuración, almacenamiento de librerías, etc. Se dispone a continuación un breve recorrido por la estructura que podemos observar en la figura 2.4.1, que aclarará algunos conceptos sobre este servidor web.

En el directorio `bin/` se encuentra el ejecutable para arrancar la aplicación, diferentes *scripts* y el precompilador de JSP. En `common/` se albergan clases y JARs accesibles a todas las aplicaciones web y al propio Apache. Los archivos de configuración, como por ejemplo el `server.xml`, se encuentran en `conf/`. Los archivos de registro y los elementos de consulta de posibles errores en las aplicaciones se albergan en el directorio `logs/`. El directorio `server/` ha sido creado para albergar clases y aplicaciones sólo accesibles por Apache. Las clases y JARs accesibles por todas las aplicaciones web se almacenan en `shared/`. Como era de esperar los archivos temporales que genera Apache se almacenan en `temp/`. Las aplicaciones web de ejemplo, las creadas por usuarios y otras propias de Apache se almacenan en `webapps/`. En este directorio se encuentran los archivos WAR (*Web Application Archive* - Archivo de aplicación web), que son archivos JAR utilizados para distribuir una colección de *JavaServer*

Pages, servlets, clases Java, archivos XML, librerías de tags y páginas web estáticas (HTML y archivos relacionados) que juntos constituyen una aplicación web. Y por último, en el directorio `work/` se almacenan archivos temporales, como los JSP compilados, etc.

Algunas entornos de desarrollo como Eclipse o NetBeans instalan su propia instancia de Apache utilizando su propia configuración, lo que evita tener que instalar *plugins* para integrar Apache y generar archivos WAR para distribuir la aplicación web fácilmente u otros menesteres.

En el interior del directorio `webapps/` se encuentra la raíz de la aplicación web, aunque en nuestro caso hemos modificado el archivo `default` dentro del servidor creando un *virtualhost* que hace que cuando una URL determinada, que para nosotros puede ser `emeeting.campusdomar.es`, `emeetings.campusdomar.es`, `meeting.campusdomar.es`, `meetings.campusdomar.es`, intente acceder al servidor Apache, el propio archivo de configuración redirige la petición a la carpeta `web` de Symfony donde se encuentra el fichero `app.php` que arranca la aplicación.

También existen los directorios *WEB-INF* y *META-INF* (opcional), situados en una zona privada de la aplicación. *WEB-INF* contiene el descriptor de despliegue de la aplicación, las clases compiladas, las bibliotecas de clases y las etiquetas para usar en los documentos JSP. *META-INF* suele contener sólo el archivo *MANIFEST.MF*, que indica las bibliotecas de las que depende la aplicación. Se suele generar automáticamente.

2.5. Javascript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas.

Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

En esta sección se hablará de las librería javascript utilizadas para realizar las animaciones del interfaz de usuario, la implementación de peticiones AJAX y la lógica en la que se apoyan algunas de las funciones Javascript propias que se describirán en la subsección 4.15.1.

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java [10].

2.5.1. jQuery

jQuery es una biblioteca de JavaScript rápida y concisa que simplifica el código en documentos HTML, manejo de eventos, animación, y las interacciones Ajax para el desarrollo

web rápido.

El proyecto jQuery comenzó como una idea en 2005 y ha crecido hasta convertirse en el conjunto de proyectos que conocemos hoy en día. jQuery está diseñado para cambiar la forma en que el usuario escribe JavaScript. Sin duda utilizar la API de cualquier *software* es cómodo y sencillo. Es un camino rápido para conseguir las funcionalidades en una aplicación o página web, de una forma rápida obteniendo unos resultados verdaderamente satisfactorios [11].

- **jQuery 1.7.1**

En esta subsección se hablará de las mejoras que incorpora esta versión con respecto a sus predecesoras.

Una de las mejoras es la unificación de todos los caminos para lanzar eventos y detectarlos en un documento jQuery en las APIs *.on()* y *.off()*.

```
\$(elements).on( events {}, selector{} {} , data{} , handler );  
  
\$(elements).off( {} events {} {} , selector{} {} , handler{} ) ;
```

Esta mejora facilita la programación y evita las confusiones en casos especiales de objetos que no poseen determinados eventos o donde la forma de ejecutar un evento debido al tipo de elemento desde el que precisa lanzar un evento se realiza con la función *.delegate()* o *.bind()*.

Lanzar eventos se ha convertido en una práctica cada vez más importante con el crecimiento del tamaño y la complejidad de las páginas. Marcos de aplicaciones como Backbone, JavaScriptMVC y SproutCore hacen un uso intensivo del lanzamiento de eventos. Con esto en mente, jQuery 1.7 refactoriza el lanzamiento de eventos con miras a lanzar eventos mucho más rápido, especialmente para los casos más comunes.

Para optimizar el código de los casos más utilizados, se analizó una muestra representativa de código de Google CodeSearch. Casi dos tercios de los selectores utilizados en *.live()* y *.delegate()* llaman a métodos con la etiqueta *tag#id.class*.[12]

- **jQuery-ui-1.8.18**

Esta librería es sencilla y fácil de manejar utilizando la documentación detallada y sus tutoriales, además existe una comunidad muy activa a disposición de cualquier usuario. Compatible con IE 6.0 +, Firefox 3 +, Safari 3.1 +, Opera 9.6 + y Google Chrome. Ofrece al usuario una enorme potencia, que permite alcanzar un alto nivel en toda aplicación web.

jQuery UI proporciona abstracciones de bajo nivel de interacción y animación, efectos avanzados y de alto nivel y *widgets* configurables. Está construido en la parte superior de la biblioteca jQuery JavaScript, y se utiliza para crear aplicaciones web altamente interactivas [11][12].

- **jQuery-kamicgraphs**

Esta potente librería para generar gráficas se utilizada en este proyecto para mostrar las estadísticas al administrador. Con una petición AJAX selecciona tablas en la base de datos y posee un inmenso repertorio de gráficas para mostrar resultados a partir de los datos en formato JSON que le sirven las aplicaciones.

- **jQuery-tipTip**

Esta librería ofrece un amplio repertorio de animaciones para mostrar datos en ventanas emergentes o flotantes. *TipTip* detecta los bordes de la ventana del navegador y se asegurará de que la descripción incluida en las ventanas que genera se queda en el interior de la ventana actual. Como resultado, la información de una herramienta se ajusta para aparecer encima, debajo, a la izquierda o a la derecha del elemento, dependiendo de la necesidad para permanecer dentro de la ventana del navegador. *TipTip* es muy ligero y personalizable fácilmente. Todos los menús de información para los botones e iconos de ayuda implementa con *TipTip* la ventana emergente de información.

2.6. MySQL

MySQL es un Sistema de Gestión de Bases de Datos (SGBD) relacional, que por lo tanto utiliza SQL, multihilo y multiusuario del que se estiman más de un millón de instalaciones.

Lo que durante un tiempo se consideró como un sencillo juguete para su uso en lugares web, se ha convertido en la actualidad en una solución viable y de misión crítica para la administración de datos. Anteriormente, MySQL se consideraba como la opción ideal para webs principalmente por su gran velocidad. Sin embargo, en la actualidad incorpora muchas de las funciones necesarias para otros entornos conservando su rapidez característica. MySQL supera desde hace tiempo a muchas soluciones comerciales en velocidad y dispone de un sistema de permisos elegante y potente, además, la versión 4 incluye el motor de almacenamiento InnoDB compatible con ACID.

Los registros de las tablas InnoDB tienen las siguientes características:

- Cada registro de índice en InnoDB contiene un encabezado de seis bytes. El encabezado se emplea para enlazar entre sí registros consecutivos, y también en el bloqueo a nivel de fila.
- Los registros en el índice agrupado contienen campos para todas las columnas definidas por el usuario. Adicionalmente, hay un campo de seis bytes para el identificador de transacción y un campo de siete bytes para el *roll pointer*.
- Si no se definió una clave primaria para la tabla, cada registro de índice agrupado contiene un campo de identificación de fila de seis bytes.
- Cada registro de índice secundario contiene también todos los campos definidos para la clave del índice agrupado.

- Un registro contiene además un puntero a cada campo del mismo. Si la longitud total de los campos en un registro es menor de 128 bytes, el puntero medirá un byte, de lo contrario, tendrá dos bytes de longitud. La matriz de estos punteros se conoce como el directorio de registros. El área a donde señalan estos punteros se denomina la parte de datos del registro.
- Internamente, InnoDB almacena las columnas de caracteres de longitud fija en un formato de longitud fija. InnoDB trunca los espacios sobrantes de las columnas VARCHAR. Nótese que MySQL puede convertir internamente columnas CHAR a VARCHAR.
- Un valor NULL SQL reserva 1 ó 2 bytes en el directorio de registros. No reservará ningún byte en la parte de datos del registro si se almacena en una columna de longitud variable. En una columna de longitud fija, reservará en la parte de datos la longitud asignada a dicha columna. La razón por la que se reserva este espacio fijo a pesar de tratarse de un valor NULL, es que en el futuro se podrá insertar en su lugar un valor no-NUL sin provocar la fragmentación de la página de índice.

Los nombres de bases de datos, tablas, índices, columnas y alias son identificadores. Los identificadores de MySQL tienen una sintaxis concreta. La longitud máxima para bases de datos, tablas, índices y columnas es de 64 bytes y para alias de 255 bytes.

Adicionalmente a las restricciones que se acaban de detallar, ningún identificador puede contener un carácter ASCII 0 o un byte con un valor de 255. Los nombres de bases de datos, tablas y columnas no deberían terminar con caracteres de espacio. MySQL 5.0 permite el uso de comillas en identificadores, aunque es mejor evitarlos tanto como sea posible.

Los identificadores se almacenan empleando Unicode (UTF8). Esto se aplica a identificadores en las definiciones de tabla que se almacenan en ficheros .frm y a identificadores almacenados en las tablas de permisos en la base de datos mysql.

Es preciso ser cuidadoso al utilizar MD5 para producir nombres de tablas, porque puede producir nombres ilegales. En nuestro caso el MD5 es usado únicamente para generar unas URLs especiales para el acceso de usuarios y se almacenan en las tablas, pero nunca es utilizado para generar nombres de tablas.

MySQL posee procedimientos almacenados y funciones que son funcionalidades añadidas a partir de la versión de MySQL 5.0. Un procedimiento almacenado es un conjunto de comandos SQL que pueden almacenarse en el servidor. Una vez que se hace, los clientes no necesitan relanzar los comandos individuales pero pueden en su lugar referirse al procedimiento almacenado.

Los procedimientos almacenados pueden mejorar el rendimiento ya que se necesita enviar menos información entre el servidor y el cliente. Por contra se aumenta la carga de la base de datos del servidor ya que la mayoría del trabajo se realiza en la parte del servidor y no en el cliente.

Los procedimientos almacenados le permiten tener bibliotecas o funciones en el servidor que sirve de base de datos. Esta característica es compartida por los lenguajes de programación modernos que permiten este diseño interno, por ejemplo, usando clases.

MySQL sigue la sintaxis SQL:2008 para procedimientos almacenados.

Muchas de estas restricciones son corregidas por Doctrine y en caso de no tener la capacidad de corrección, la generación de las estructuras de la base de datos mediante las instrucciones de propias de Doctrine ofrecerá los errores correspondientes para una rápida corrección del código erróneo.

2.7. HTML 5 y CSS 3

La elección de este avanzado lenguaje es obvia. Dentro de no muchos años todos los exploradores sorportarán únicamente HTML 5 que incluye unas mejoras para la visualización de vídeos muy importantes e incluye características de estilo novedosas, con la ayuda de Twig es altamente eficiente la creación de plantillas HTML y la última versión revisada es la mejor opción en la actualidad.

CSS 3 incluye las últimas mejoras en el diseño de interfaces y en compatibilidad con exploradores. De esta forma sacaremos el mayor partido a nuestro diseño y facilitaremos la compatibilidad con todos los exploradores del mercado. El consorcio W3C valora cualquier cambio que se realice para conseguir un estándar actualizado y compatible con el mayor número de exploradores, por ello las novedades que incorpora CSS en su tercera versión son altamente recomendables.

Capítulo 3

Decisiones de diseño

En este capítulo se ofrecen los diferentes motivos de las decisiones que se han tomado para el posterior desarrollo del proyecto. Las premisas fundamentales han sido el desarrollo ágil y la obtención de un resultado óptimo en el menor tiempo posible.

3.1. *Adobe Connect*®

Durante más de 25 años, *Adobe* ha revolucionado la interacción de todo el mundo con ideas e información. Tanto en la web como en papel, en vídeo o en dispositivos móviles, las tecnologías y el *software* de *Adobe* han llegado a todos los rincones del mundo. *Adobe Flash*® Player es la plataforma de *software* más extendida, y se encuentra instalada en más del 98 % de los equipos de escritorio con conexión a Internet. La tecnología *Adobe Flash* es la más utilizada para producir vídeo en la Web, mientras que el *software* *Adobe Reader*® se ha convertido en el estándar mundial para compartir documentos e información. Las famosas marcas de *Adobe* como *Adobe Photoshop*®, *PDF*, *Acrobat Reader*® y *Flash*, fijan continuamente estándares para la producción y difusión de contenidos sofisticados que resulten atractivos para los usuarios.

Este sistema, que se complementa con *Adobe*® *Connect*™, es especial para el desarrollo de reuniones en línea al instante y clases virtuales, a través de una amplia gama de navegadores web y sin necesidad de descargas pesadas. Permite compartir contenido dinámico, incluido flujo de audio, vídeo y simulaciones de *software*, también hace posible las conferencias de vídeo entre varias personas. Además, las salas de reuniones personalizables de *Acrobat Connect Professional*® y todo su contenido se guardan automáticamente y siempre quedan disponibles, lo que se traduce en una reducción considerable del tiempo de preparación para los seminarios recurrentes, las reuniones de equipos y las presentaciones de ventas. Es la solución para conferencias web de tipo “todo en uno” más completa, y de lejos ofrece tres ventajas claras con respecto a los productos de la competencia:

1. Acceso inmediato

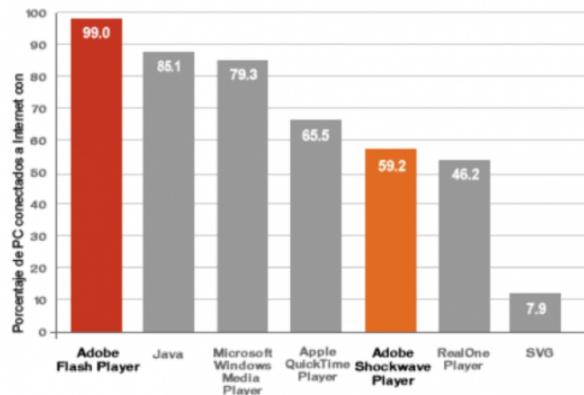


Figura 3.1.1: Utilización de Adobe Flash Player

Los asistentes pueden participar en las reuniones con sólo hacer clic en una URL, sin necesidad de perder tiempo con descargas o esperas molestas. *Acrobat Connect Professional®* permite que personas de todo el mundo puedan reunirse para colaborar e interactuar de forma inmediata. Lo único que se requiere es un navegador web y *Adobe Flash Player*, una aplicación que ya está instalada en más del 98 % de los equipos de escritorio conectados a Internet.

A diferencia de otras soluciones líderes para conferencias web, que requieren que los participantes descarguen *plug-ins* o *software* propietario, lo cual puede significar grandes retrasos cuando los usuarios se enfrenten a problemas técnicos, e incluso puede impedir que algunos participantes asistan a la reunión.

2. Alto impacto

Adobe Flash es la tecnología utilizada en la mayoría de los archivos multimedia de vídeo e interactivos que se encuentran hoy en Internet. En las conferencias web, *Adobe Flash* permite disponer de vídeo, audio e interactividad lo cual brinda a los usuarios la sensación de estar asistiendo en persona. El elegante interfaz y las funciones de interactividad de *Acrobat Connect Professional®* mantienen al público interesado, con el fin de que trabajen de forma productiva y que retengan lo aprendido, al mismo tiempo que disfrutan mientras lo hacen.

3. En directo y bajo demanda

Acrobat Connect Professional® hace posible reunir a personas de distintas partes del mundo en una sesión de trabajo en tiempo real o en un aula virtual, y también ofrece la opción de que los usuarios accedan a contenidos importantes cuando y donde les resulte más conveniente.

Acrobat Connect Professional® se basa en una arquitectura abierta y ampliable que permite una integración rentable con la infraestructura existente y las inversiones futuras en tecnología. *Acrobat Connect Professional®* utiliza estándares como XML y Java™ para

el intercambio de datos, y es la única solución que admite servicios web que se pueden usar para la gestión de todos los elementos. *Adobe®* ofrece también un centro de recursos para desarrolladores, con cientos de interfaces de programación de aplicaciones (API) y kits de desarrollo de *software* (SDK) para ampliar funciones e integrar la solución con otras aplicaciones y servicios internos, hasta crear la solución personalizada que mejor se adapte a las necesidades exclusivas de la organización.

La API de *Adobe Connect®* es sin duda su principal baza para que cualquier desarrollador de aplicaciones web lo elija como motor de videoconferencia.

Adobe Connect® posee una de las API's más completas para sistemas de videoconferencia, lo que ofrece la posibilidad de configurar todos los parámetros de una sala de una forma muy precisa. Para este proyecto una de las funcionalidades más interesante es la posibilidad de configurar las plantillas de cada sala para adaptar nuestra aplicación a los diferentes escenarios, ya sean clases magistrales, reuniones de grupos de investigación con uno o varios ponentes, salas de trabajo en grupo, etc. También ofrece la opción de configurar el ancho de banda que ocupará el flujo de audio y vídeo de la aplicación en su viaje por la red.

3.2. Symfony

Es uno de los *frameworks* con mayor impacto en la actualidad. Symfony aumenta exponencialmente la productividad y ayuda a mejorar la calidad de las aplicaciones web aplicando todas las buenas prácticas y patrones de diseño que se han definido para la web [6]. Un gran abanico de aplicaciones web que nacen en la actualidad se realizan utilizando este *framework*, ya que ofrece una gran versatilidad y la posibilidad de testear el código con un eficiente sistema interno para la corrección de errores. Symfony también dota a las aplicaciones de una gran velocidad como se ha comentado en el apartado 2.2.2.

Su asignador relacional de objetos (ORM) y la capa de abstracción de base de datos de potente calado y gran capacidad de configuración, hacen que este *framework* ofrezca al desarrollador la posibilidad de realizar aplicaciones web basadas en PHP muy sofisticadas. Una de las virtudes más importantes para la elección de este *framework*, es la gran comunidad que lo respalda y que aporta corrección de errores e implantación de tecnologías punteras en sus últimas versiones. Las versiones actuales nos garantizan que la evolución de Symfony guarde una buena estructura de manera que actualizar el antiguo *software* implementado con Symfony no traerá grandes quebraderos de cabeza.

El proyecto Doctrine hace que la elección de Symfony sea obvia ante el desarrollo de cualquier aplicación digna de perdurar en el tiempo. Ofrece un gran abanico de opciones para ampliar nuestra base de datos y optimizar el tiempo empleado para conseguir las nuevas funcionalidades en un breve espacio de tiempo y sin un gran despliegue humano. Cualquier tecnología web puede ser incrustada en los ficheros de Symfony para mejorar o aumentar la compatibilidad entre aplicaciones y necesidades del usuario final.

3.3. PHP

Según crecía PHP, empezó a ser reconocido como una popular plataforma de desarrollo web. Una de las más interesantes formas de ver esta tendencia fue observando los libros de PHP que se han ido publicando a lo largo de los años. Este gran número de libros, escritos por diferentes autores, publicados por muchos editores, y su disponibilidad en tantas lenguas son un fuerte testimonio del éxito mundial de PHP [9].

Las cuatro grandes características para declinar la balanza hacia PHP han sido:

- Velocidad: No sólo la velocidad de ejecución, la cual es importante, sino además no crear demoras en la máquina. Por esta razón no debe requerir demasiados recursos de sistema. PHP se integra muy bien junto a otro *software*, especialmente bajo ambientes Unix, cuando se configura como módulo de Apache.
- Estabilidad: La velocidad no sirve de mucho si el sistema no es fiable. Ninguna aplicación es 100 % libre de errores, pero con una increíble comunidad de programadores y usuarios como respaldo es mucho mas difícil que sobrevivan. PHP utiliza su propio sistema de administración de recursos y dispone de un sofisticado método de manejo de variables, conformando un sistema robusto y estable.
- Seguridad: El sistema debe poseer protecciones contra ataques. PHP provee diferentes niveles de seguridad configurables.
- Simplicidad: Se les debe permitir a los programadores generar código productivamente en el menor tiempo posible. Usuarios con experiencia en C y C++ podrán utilizar PHP rápidamente. No cabe la menor duda de que la utilización de un *framework* simplifica todavía más su utilización.

Este conjunto de dogmas unidos a la gran comunidad que desarrolla PHP y ayuda a incluir las funcionalidades de última generación, hacen de PHP el *software* recomendado para el servidor Apache, que como describiremos a continuación hacen un fantástico equipo, altamente recomendado para la puesta en servicio de aplicaciones web.

3.4. Apache

La elección del servidor Apache se debe a las siguientes ventajas:

- Modular:
Podemos incluir multitud de módulos para adecuar el servidor a las necesidades de nuestra aplicación. De hecho, PHP posee un módulo específico para Apache, como ya se ha comentado en el capítulo 2, que aprovecha al máximo la velocidad de Apache ya que actúa sobre el núcleo del servidor de forma directa a través de este módulo.

- Código abierto:

Las plataformas de código abierto nos permiten desarrollar código de forma libre y a un menor coste que el *software* propietario. Los foros de discusión y resolución de errores son mucho mayores, de forma que un desarrollador puede encontrar solución a su problema de forma rápida y en caso de ser pionero en una casuística determinada, también obtendrá respuesta de forma rápida, ya que la comunidad libre es muy activa.

- Multi-plataforma:

Es imprescindible para cualquier aplicación poder desplegarse en todo sistema operativo existente en el mercado, y esta premisa la cumple Apache a la perfección.

- Extensible:

Consta de una amplia galería de módulos que hacen que podamos agregar multitud de funcionalidades, como el ya mencionado y utilizado en este proyecto módulo para PHP.

- Popular:

Gracias a su popularidad que crece día a día debido a su licencia, cualquier desarrollador cuenta con una amplia comunidad que lo respalda y aumenta la actividad tanto de desarrolladores como de expertos en este servidor.

3.5. MySQL

Son muchas las razones para escoger MySQL como solución de misión crítica para la administración de datos. Las más interesantes ligadas a este proyecto son:

- Coste: El coste de MySQL es gratuito en su mayor parte y su servicio de asistencia resulta económico en caso de ser necesario.
- Asistencia: MySQL AB, la compañía responsable del desarrollo de MySQL, ofrece contratos de asistencia a precios razonables y existe una nutrida y activa comunidad MySQL.
- Velocidad: MySQL es mucho mas rápido que la mayor parte de sus rivales, por tanto en este aspecto es obvia su elección.
- Funcionalidad: MySQL dispone de muchas de las funciones que exigen los desarrolladores profesionales, como compatibilidad completa con ACID, compatibilidad para la mayor parte de SQL ANSI, volcados online, duplicación, funciones SSL e interacción con la mayor parte de los entornos de programación. Se desarrolla y actualiza mas rápidamente que la mayoría de sus rivales, por lo que prácticamente todas las funciones estándar de MySQL no están en fase de desarrollo.

- Portabilidad: MySQL se ejecuta en la inmensa mayoría de sistemas operativos y, en la mayor parte de los casos, los datos se pueden transferir de un sistema a otro sin dificultad.
- Facilidad de uso: MySQL resulta fácil de utilizar y de administrar. Gran parte de las viejas bases de datos presentan problemas por utilizar sistemas obsoletos, lo que complica innecesariamente las tareas de administración. Las herramientas de MySQL son potentes y flexibles, sin sacrificar su capacidad.

MySQL 5.5 es rápido, dispone de funciones de volcado online e incorpora una gran cantidad de funciones nuevas. Son pocas las razones para desechar MySQL como solución de base de datos.

MySQL AB dispone de un sistema de asistencia eficiente y a un precio razonable, y, como ocurre con la mayor parte de las comunidades de código abierto, hay disponible una gran cantidad de ayuda en la web.

Junto con el visor de base de datos TOra que permite la edición de la base de datos de una forma sencilla, facilitando la búsqueda visual de datos para su depuración y mayor facilidad en el diseño, hacen de estas dos herramientas la mejor opción disponible en el mercado relación calidad-precio para el desarrollo de aplicaciones web y cada vez más en otros sectores.

Capítulo 4

Implementación del proyecto

La aplicación “eMeeting” está diseñada para crear salas de forma dinámica bajo demanda y acceder a las mismas de forma similar a una llamada telefónica, es decir, en el momento de acceso se consulta la disponibilidad de recursos y se deniega el acceso en el peor de los casos. En la creación de la sala se pueden seleccionar los usuarios destino de una lista o enviando la URL correspondiente por algún medio electrónico. El número máximo de conferencias simultáneas es una limitación para la aplicación, ya que el *software* propietario aumenta su coste con el incremento de salas. Sin embargo en nuestro diseño se ha supuesto que nunca se llegará a tal número y se han dimensionado las conexiones con un criterio que garantice un bajo porcentaje de bloqueo de petición de conexión, para lo cual se realizará un interfaz de seguimiento que mostrará estadísticas con el número máximo de conferencias simultáneas, número de usuarios en activo y otros parámetros de interés. De esta forma la lógica del número máximo de salas será transparente para el usuario.

En este capítulo describiremos las entidades, repositorios, controladores y servicios de la aplicación. Cada una de estas partes es fundamental y el propio *framework* nos ofrece cómodas herramientas para obtener un modelo vista controlador (MVC) funcional y de fácil implementación. Las entidades definen las tablas de la base de datos. En los repositorios se almacenan las consultas a la base de datos que no genera el *framework* por defecto. Los controladores contienen la lógica que procesará los datos de la aplicación y tienen asociadas las plantillas para mostrar o recoger los datos introducidos por el usuario. Los servicios contienen las funciones que usan la API *Adobe Connect®* y proporcionan a eMeeting las credenciales necesarias para conectarse como administrador al servidor *Adobe Connect®* y generar las salas, usuarios y todo lo necesario para el correcto funcionamiento de la aplicación.

Para llevar a cabo la aplicación se ha dispuesto de 5 elementos. Como se puede observar en la figura 4.0.1, contamos con un cliente que representa el conjunto de usuarios involucrados en la comunicación que disfrutan de la aplicación desde su explorador web, aunque también es posible operar desde un terminal móvil previa descarga de un *plugin*, y 4 servidores que pasamos a describir a continuación.

- El servidor eMeeting: almacena todos los ficheros necesarios para la configuración y el

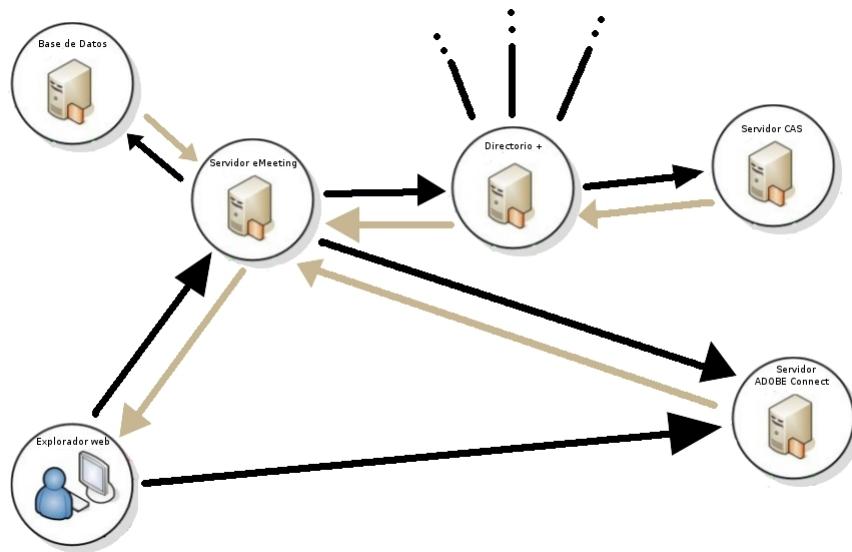


Figura 4.0.1: Diagrama estructural de la aplicación

funcionamiento de la aplicación web.

- La base de datos de respaldo de la aplicación: que en este caso se implementa utilizando el motor MySQL. El diseño de la aplicación nos permite modificar el motor de base de datos realizando cambios exiguos.
- Directorio +: este servidor ayudará a sincronizar la macroaplicación mencionada en la sección 1.1 y que se describirá con mayor detalle en este capítulo. Nuestra base de datos de respaldo sería redundante pero en la fase de despliegue y puesta en servicio es aconsejable para aportar una mayor fiabilidad y estabilidad al servicio.
- El servidor *Adobe Connect*[®]: donde se encuentra alojada la aplicación *Adobe Connect*[®] que usaremos como motor de videoconferencia.
- El servidor CAS: que aloja el servicio para autenticar a los usuarios que accedan a la aplicación y ayudará a “Directorio +” a detectar usuarios ya registrados en alguna aplicación para evitar registros superfluos en el resto de aplicaciones que componen “Campus do Mar”.

4.1. Archivos de configuración

Los archivos de configuración están almacenados en diferentes carpetas dentro de los directorios Symfony descritos en la subsección 2.2.

En el interior del directorio `app/config` se encuentran dos ficheros importantes: `routing.yml` y `security.yml`. Este par de ficheros, contienen la configuración que determina qué URLs de acceso a la aplicación serán públicas o privadas y cuál será la puerta de entrada para las URLs privadas.

En la carpeta `Resources/public/css` guardaremos los archivos para alojar el CSS, en `Resources/public/js` el Javascript, en `Resources/public/images` las imágenes y en `Resources/public/wsdl` la configuración de las funciones del *middleware* que describiremos en la subsección 4.9.1.

En la carpeta `Resources/views` guardaremos las plantillas que se mostrarán como página HTML para cada uno de los controladores de aplicación debidamente ordenados según corresponda. En el interior de este directorio se aloja una carpeta por cada controlador, con archivos que ofrecen información para completar los parámetros y elementos necesarios en cada plantilla HTML.

En la carpeta `Test` se almacenan archivos para diseñar pruebas específicas para cada controlador, entidad y servicio de la aplicación agrupadas en directorios con nombres relativos a los elementos sobre los cuales se ejecutarán las pruebas. En el interior de cada uno de estos tres directorios están disponibles los ficheros para realizar pruebas diseñadas por el programador que intentan detectar errores, fallos de diseño o puntos débiles de la aplicación. Esta funcionalidad de Symfony ha revelado una gran cantidad de errores en el código, *a priori* muy difíciles de detectar.

4.2. Concepto de sala

Se llama sala a la entidad fundamental de la aplicación. Cada sala contendrá una única reunión en la que todos los participantes colaboran según el rol que desempeñen. Cada usuario podrá crear las salas que precise oportunas y darle el nombre que deseé, aunque el número total de salas en uso no podrá superar diez, pero esta información será transparente al usuario y una lógica interna ha sido diseñada para alertar al administrador de la aplicación sobre la proximidad al número máximo de salas en uso, por si cree oportuno disponer de más salas y debe negociarlo con la compañía suministradora de recursos, en nuestro caso *Adobe Connect®*. La aplicación “eMeeting” podría utilizarse con otro motor de videoconferencia, pero sería de todas formas necesario informar al administrador de la proximidad del número máximo de salas en uso, ya que en cualquier motor el número de salas es un recurso limitado. Un inconveniente de la aplicación es no poder dar un nombre de sala que exista de entre las salas en activo, sería posible implementar una lógica para que cada usuario no se percate en primera instancia de este hecho, ya que *Adobe Connect®* no permite tener salas con el mismo nombre, pero esta solución cambiará el nombre de la sala en la pantalla de *Adobe Connect®*, por este motivo no se ha implementado esta funcionalidad.

El concepto de sala no se refleja en la lógica de la base de datos, pero se tiene muy presente a la hora de diseñar y mostrar la aplicación al usuario. Podríamos decir que una reunión se realiza en una sala, pero en realidad disponemos únicamente de diez salas, sin embargo el usuario puede crear todas las reuniones que deseé. En los primeros diseños, de forma transparente al usuario, se creaba una sala de videoconferencia para cada usuario



Figura 4.2.1: Ayuda de la aplicación

de “Campus do Mar”, y se mostraba al usuario los estados de sala (*room*) habilitada/deshabilitada, mágica/no mágica o secreta/pública. Definitivamente este diseño se abandonó y el usuario entra por primera vez a la aplicación y no posee ninguna sala, una enorme flecha le indica las instrucciones a seguir para crear su propia sala como se puede ver en la figura 4.2.2. También se dispone de un manual en red con indicaciones para realizar cualquier acción disponible dentro de la aplicación “eMeeting”, ver figura 4.2.1. En nuestra lógica de base de datos, las salas libres son en realidad reuniones (*meetings*) almacenadas en la tabla “meeting” y siempre están asociadas a un estado: *NOW*, *NEW*, *SCHEDULE*, *CANCELLED*, *FINICHED* o *LOCKED*. Estos 6 estados, que se explicarán con detalle en la sección 4.4, son fundamentales para gestionar las funcionalidades que ofrece la aplicación.

4.2.1. Sala mágica

Una sala mágica hace referencia a un tipo especial de sala que se pone a disposición de los usuarios de la aplicación para proporcionar un servicio exclusivo. Como se ha comentado en este capítulo el número de salas disponibles es limitado y cabe la posibilidad de denegación de servicio en momentos de máxima ocupación. Como describiremos con mayor profundidad en la subsección 4.3.3 el sistema alerta al administrador para que provoque con más salas a la aplicación si la ocupación se acerca al límite máximo disponible. Debido a esta implementación y ante saturación no deseada se ha implementado una lógica que recibe el nombre de sala mágica.

A partir de la lista total de licencias disponibles se configura el número máximo de salas



Figura 4.2.2: Pantalla de inicio por vez primera a la aplicación.

mágicas. El sistema permite configurar todas las licencias como mágicas aunque es una opción absurda ya que la finalidad del diseño de este sistema consiste en proporcionar muchas entidades *eMeeting* a cada usuario con una disposición de licencias limitada. Cualquier sala del sistema puede ser marcada como mágica por el administrador y por tanto pasará a estar siempre disponible, por contra una licencia estará siempre ocupada y no disponible para el resto de usuarios. Para entender mejor esta lógica pasamos a describir el caso concreto de “Campus do Mar”.

Existen diez licencias y tres salas mágicas como máximo. Cuando se marca una sala como mágica las licencias disponibles para el resto de usuarios pasa a ser nueve ya que la sala marcada como mágica ocupa la licencia en todo momento incluso cuando no hay una reunión en curso. De esta forma se garantiza la disponibilidad del recurso en detrimento del número de salas disponibles. Este tipo de salas permite al administrador del sistema asegurarse la disponibilidad de recursos incluso en períodos de máxima ocupación y ofrecer un servicio exclusivo a usuarios con un perfil determinado.

4.2.2. Acceso a las salas

Debido a los diferentes tipos de salas y usuarios es necesario implementar un control de acceso para las salas. Para esta lógica definimos los conceptos de *Viewsalt*, *Salt* y *SecretSalt*, el cual nos ayudará a construir la URL que dará acceso a las diferentes salas de la aplicación.

- *Viewsalt*: Se genera con el algoritmo MD5, a partir del nombre que recibe la reunión. Se hace referencia a ella en la sección 4.7. La URL con esta huella dará acceso a las reuniones públicas mediante registro a través del formulario correspondiente para actuar en la reunión como un usuario participante descrito en la subsección 4.5.1.
- *Salt*: Este campo que almacenamos asociado a cada una de las reuniones creadas, es generado con el login de usuario o el nombre del grupo, según corresponda. Construye

la URL de acceso para las salas públicas y usuarios presentadores descritos en la subsección 4.5.1.

- *SecretSalt*: Se genera con el algoritmo MD5 a partir del login de usuario o del nombre del grupo. Construye las URL para las salas privadas. Este acceso es realmente importante ya que una sala privada no posee un enlace común de acceso y sólo los usuarios invitados pueden acceder a ella. Este enlace permite el acceso a la sala a todo usuario que la posea a pesar de ser una sala privada, por así decirlo es como una puerta trasera a una sala que por definición no es accesible a usuarios no invitados.

Esta decisión se toma a partir del momento en que se decide crear todas las salas con el estado de públicas en el servidor *Adobe Connect®* y gestionar el control de acceso desde la propia aplicación *eMeeting*. De esta forma a partir de la URL generada el usuario puede decidir invitar a cualquier persona, ya sea miembro de la Plataforma “Campus del Mar” o no y otorgarle el rol que considere oportuno. Este planteamiento es altamente útil para dar una mayor flexibilidad a la hora de trabajar con personal ajeno a “Campus do Mar”, pero que en un momento dado puede llegar a colaborar en un proyecto de investigación.

Este control de acceso se implementa con la siguiente lógica:

- Pública:
 - Entrada con Salt: Pueden entrar los usuarios que conozcan la sala del usuario ya que la URL se construye con el login del usuario que la crea.
 - Entrada con SecretSalt: Pueden entrar todos los que conozcan la URL generada con el SecretSalt de la sala.
- Privada:
 - Entrada con Salt: Sólo puede entrar el usuario. Esta situación se genera como consecuencia de la lógica implementada, pero no tiene mucho sentido ya que un meeting para un sólo usuario es absurdo.
 - Entrada con SecretSalt: Pueden entrar todos los que conozcan la URL generada con el SecretSalt de la sala.

4.3. CRON

El cron es la parte del servidor “eMeeting” que se encarga de ejecutar funciones periódicas que ayudan a la sincronización, depuración y mantenimiento de la aplicación. Cada una de estas funciones realiza tareas críticas para el correcto funcionamiento de la aplicación y alerta a los administradores de posibles saturaciones en el servicio.

A continuación pasamos a describir las funciones ejecutadas por el cron.

4.3.1. Sincronización de grabaciones

Para poder visualizar las grabaciones en tiempo real y antes de eliminar la reunión, se ejecuta el comando CMAR:SYNC:REC diseñado para guardar en la base de datos la URL de acceso a las grabaciones únicamente cuando la grabación ha finalizado.

Esta acción es ejecutada cada 5 minutos por el cron y dispone en un elemento de la reunión la lista de grabaciones finalizadas para esa reunión.

La acción de borrado de la reunión también finaliza ejecutando esta acción para almacenar todas las grabaciones en una carpeta creada para tal efecto dentro del servidor de *Adobe Connect®*. Las grabaciones nunca se eliminan del servidor y siempre son accesibles por el usuario desde el panel de control que describiremos más adelante.

4.3.2. Sincronización de usuarios de *Adobe Connect®*

Cuando un usuario accede a una sala y no tiene una cuenta de acceso a la aplicación, se crea un usuario temporal en *Adobe Connect®*. Esta medida nos otorga control total sobre los privilegios de los participantes en la reunión. Se ha diseñado esta solución porque *Adobe Connect®* no ofrece ninguna solución en su API para abordar esta casuística, aunque se ha comprobado que internamente sí están implementados diferentes roles para usuarios anónimos. Dicho comando se llama CMAR:SYNC:USERADO.

El usuario creado para dar solución a este problema es borrado dos días después de su creación. Para ello en el campo descripción que ofrece *Adobe Connect®* se incluye el texto: *User created as guest* y en el campo reservado para la dirección de correo electrónico incluimos la fecha de creación para que la función compruebe que el usuario se ha creado hace dos días.

4.3.3. Recopilación de datos para la elaboración de estadísticas de la aplicación

Este script se ejecuta cada treinta minutos y recoge información sobre el número de salas activas y el número de usuarios en cada sala para elaborar estadísticas de uso del sistema. La función más importante consiste en comparar el número de salas activas con el número máximo contratado y alertar al administrador de la proximidad entre ambos ya que *Adobe Connect®* ofrece licencias de pago que se han de contratar previamente. Si se da el caso, el administrador debe ampliar el número de licencias para evitar que usuarios no puedan disfrutar de su multivideoconferencia. Este comando recibe el nombre de CMAR:SYNC:STORESTAT.

4.4. Diseño de la base de datos

Uno de los primeros desafíos al que nos enfrentamos es el diseño de la base de datos. Como se indica en el capítulo 3 con el *framework* Symfony es fácilmente implementable e

1	def	emeeting	meeting	id
2	def	emeeting	meeting	owner_id
3	def	emeeting	meeting	group_id
4	def	emeeting	meeting	state
5	def	emeeting	meeting	salt
6	def	emeeting	meeting	secretsalt
7	def	emeeting	meeting	viewsalt
8	def	emeeting	meeting	title
9	def	emeeting	meeting	description
10	def	emeeting	meeting	url
11	def	emeeting	meeting	date
12	def	emeeting	meeting	public
13	def	emeeting	meeting	magic
14	def	emeeting	meeting	concurrent
15	def	emeeting	meeting	created
16	def	emeeting	meeting	updated

Figura 4.4.1: Tabla para almacenar datos de reuniones

incluso puede modificarse durante la fase de prueba, en caso de encontrar alguna carencia en el diseño inicial. La utilización de MySQL, gracias a su velocidad, nos augura un desarrollo exitoso. A pesar de ello, con la aplicación en producción, modificar la base de datos puede ocasionar enormes quebraderos de cabeza. Como consecuencia de este hecho, implementar una base de datos sólida con la información más relevante para nuestra aplicación es una labor tediosa que hemos elaborado con gran detenimiento.

En la figura 4.4.1 podemos observar los diferentes campos que incluye la entidad “Meeting”. A continuación, describiremos el cometido de las diferentes columnas de la tabla.

- El campo “id” se genera de forma automática al introducir un elemento en la tabla e identifica cada entidad “Meeting” dentro de nuestra base de datos de forma única, requisito impuesto por el motor MySQL para identificar de forma única los contenidos de cada tabla.
- En la columna “owner_id” se almacena el identificador correspondiente al propietario y creador de cada reunión.
- “Group_id” se corresponde con una funcionalidad *Out of Scope*¹, por lo tanto no se ha utilizado en la actual versión del proyecto, pero está reservada para una funcionalidad en la que una reunión puede pertenecer a un grupo de usuarios además de al usuario creador.
- La columna “state” almacena el estado de cada reunión que puede ser *NOW*, *NEW*, *SCHEDULED*, *CANCELLED*, *FINISHED* o *LOCKED*.
 - El estado *NOW* nos indica que la reunión está siendo creada y nos ayuda a sincronizar la base de datos ante tiempos de demora. También se llega a este

¹Fuera de los objetivos de la versión final.

estado cuando una reunión con duración y fecha se está realizando en el momento actual.

- El estado *NEW* se corresponde con aquel en el cual una reunión está disponible pero no se le ha dado una fecha ni una duración. Para la versión en la que no utilizamos la funcionalidad de programar una reunión para el futuro, la mayoría de las reuniones se encuentran en el estado *NEW*.
 - El estado *SCHEDULED* nos informa de que la reunión tendrá lugar en el futuro.
 - La reunión *CANCELLED* es aquella que ha sido creada y luego borrada. La aplicación almacena datos para posteriores estadísticas.
 - El estado *FINISHED* nos indica que la reunión ha finalizado, funcionalidad desarrollada para la versión en la que podemos programar grabaciones y que se describirá con detenimiento en la sección 5.2.
 - Una reunión *LOCKED* es aquella que el usuario bloquea para que otros usuarios que han sido invitados no tengan acceso a la sala (como una llave que cierra la puerta a la entrada de la sala de videoconferencia).
- Las columnas “salt”, “secretsalt” y “viewsalt” almacenan una variable especial cada una, referente a la reunión a la que pertenecen, creada para dar un acceso con diferentes privilegios a usuarios tanto internos como externos a la aplicación.
 - Si la URL que recibe un usuario invitado a una videoconferencia contiene el “viewsalt” el usuario tendrá privilegios de presentador.
 - Si por el contrario contiene el “secretsalt” el usuario tendrá unos privilegios similares al creador de la reunión.
 - La variable “salt” guarda una URL simple creada con el título de la reunión que nos simplificará el trabajo a la hora de crear las URLs para asistir a la reunión en el caso de usuarios que tienen acceso a la aplicación “eMeeting”.
 - La variable “title” guarda el título dado a la reunión.
 - En “description” guardamos, si lo consideramos oportuno, una breve descripción de la reunión que aparecerá en la página principal acompañando a cada reunión.
 - La columna “URL” contiene la URL de acceso a cada reunión y de esta forma tenemos un acceso rápido para ofrecerla en cualquier parte de nuestro código.
 - “Date” indica la fecha de creación de la reunión.
 - “Public” es una columna con valor *booleano* que indica si la reunión es pública (en caso de ser cierta) o privada (en caso de ser falsa).
 - “Magic” es otra columna *booleana* que indica si la reunión es mágica o no para poder otorgarle los privilegios oportunos como se ha visto en la subsección 4.2.1.

#	Table Catalog	Table Schema	Table Name	Column Name
1	def	emeeting	user	id
2	def	emeeting	user	login
3	def	emeeting	user	password
4	def	emeeting	user	name
5	def	emeeting	user	surname
6	def	emeeting	user	email

Figura 4.4.2: Tabla para Usuarios

#	Table Catalog	Table Schema	Table Name	Column Name
1	def	emeeting	recording	id
2	def	emeeting	recording	meeting_id
3	def	emeeting	recording	state
4	def	emeeting	recording	title
5	def	emeeting	recording	url
6	def	emeeting	recording	dateCreated

Figura 4.4.3: Tabla para grabaciones

- La columna “created” almacena la fecha de creación de la reunión. La columna *updated* indica si hemos modificado alguna variable de la reunión y nos ayuda a tomar decisiones a la hora de eliminar o dar acceso a las reuniones.

Como podemos observar en la figura 4.4.2, en la tabla *user* almacenaremos datos referentes al usuario. Por simple inspección podemos intuir qué tipo de parámetros del usuario se guardan en cada columna de esta tabla. De nuevo se tiene una variable “id” imprescindible para el motor MySQL y poder así identificar cada entidad dentro de la tabla de forma única.

En la figura 4.4.3, observamos los datos que se almacenan de cada grabación. Son destacables las columnas “meeting_id” que almacena el identificador de la reunión a la que pertenece la grabación o “state” que almacena el estado de cada grabación. Este estado puede ser *REACHABLE* cuando tenemos acceso a la grabación o *LOCKED* cuando la grabación está bloqueada y no se tiene acceso a ella.

En las figuras 4.4.4 y 4.4.5, se muestra un esquema de las tablas que almacenan los datos estadísticos a partir de las funciones que se ejecutan en el cron como se ha detallado en la sección 4.3.

Table Catalog	Table Schema	Table Name	Column Name
def	emeeting	Log	id
def	emeeting	Log	datetime
def	emeeting	Log	sco_id
def	emeeting	Log	participants
def	emeeting	Log	length_minutes

Figura 4.4.4: Tabla para almacenar datos estadísticos parciales

Table Catalog	Table Schema	Table Name	Column Name
def	emeeting	Log_Total	id
def	emeeting	Log_Total	datetime
def	emeeting	Log_Total	participants
def	emeeting	Log_Total	active_rooms

Figura 4.4.5: Tabla para almacenar datos estadísticos totales

#	Table Catalog	Table Schema	Table Name	Column Name
1	def	emeeting	meeting_user	meeting_id
2	def	emeeting	meeting_user	user_id

Figura 4.4.6: Tabla de cruce de las tablas meeting y user

En la figura 4.4.4 podemos observar variables propias de una reunión concreta.

- La columna “datetime” guarda la fecha y hora de inicio de una reunión según el formato *datetime* de PHP.
- La columna “sco_id” guarda un “id” que identifica de forma única la reunión dentro del servidor *Adobe Connect®*.
- Las columnas “participants” y “length_minutes” guardan el número máximo de participantes de esa reunión y la duración en minutos de la misma.

Por su parte en la figura 4.4.5 se almacenan estadísticas totales del sistema y no referentes a una única reunión.

- La variable “datetime” almacena la fecha y hora del momento en el cual se tomó la muestra, es decir, el momento en el que el cron ejecutó el comando correspondiente que rellena esta tabla.
- En la variable “participants” se guarda el número total de participantes activos en el servidor *Adobe Connect®*, es decir, el número de usuarios que están participando en una videoconferencia.
- En la variable “active_rooms” se guarda el número de salas de videoconferencia que están siendo utilizadas.

Estas dos últimas variables serán procesadas por la función que hace referencia a ellas en el cron de forma que si se llega a una zona próxima a la saturación se enviará un correo electrónico al administrador del sistema para advertirle de tal situación.

La figura 4.4.6, es una tabla creada por Symfony a petición del programador y que almacena de forma automática un cruce de datos entre tablas, en este caso relaciona los identificadores de reuniones con sus respectivos usuarios. De esta forma se tendrá acceso a las tablas de reuniones y de usuarios en cualquier parte de nuestro código únicamente accediendo a uno de estos dos identificadores. Se podrá acceder a los datos de todas las

#	Table Catalog	Table Schema	Table Name	Column Name
1	def	emeeting	staff	id
2	def	emeeting	staff	key_
3	def	emeeting	staff	name
4	def	emeeting	staff	description
5	def	emeeting	staff	type

Figura 4.4.7: Tabla para grupos

#	Table Catalog	Table Schema	Table Name	Column Name
1	def	emeeting	users_staffs	user_id
2	def	emeeting	users_staffs	group_id

Figura 4.4.8: Tabla para el cruce de datos entre grupos y usuarios

reuniones de un usuario a partir de su identificador, a la vez que se podrán recorrer los datos de todos los usuarios de una reunión a partir del identificador de la reunión. Ofrece el acceso al nombre de un grupo a partir de un usuario que pertenezca a ese grupo sin más que realizar las instrucciones:

```
\$Group\_name=\$user->getGroup()->getName();
```

Muchas de estas asociaciones son indispensables para legitimar con facilidad los permisos que se da a cada rol de usuario sobre las reuniones o para acceder a información dentro de los controladores de aplicación o servicios, incluso para interactuar con el servidor *Adobe Connect®*, ya que en funciones determinadas se obtiene información sobre los permisos de acceso de un usuario a partir de la *cookie* generada por el servidor. Estas y otras útiles funcionalidades se encuentran disponibles gracias a esta tabla.

La tabla 4.4.7 está diseñada para guardar los datos de grupos de usuario. Como se puede observar, las columnas almacenan el nombre del grupo, una breve descripción y el tipo de grupo que compone la tabla. Esta tabla no es utilizada debido a la no implementación en la versión en producción de la lógica para grupos diseñada en primera instancia.

Como complemento a la tabla de grupos, mediante la configuración de la entidad *group* y de la entidad *user* de Symfony cruzaremos en nuestra base de datos los identificadores de ambas entidades mediante una relación “n→n”, ya que un usuario puede pertenecer a “n” grupos y un grupo puede contener “n” usuarios.

Después de diseñar nuestra base de datos debemos especificar a Symfony qué base de datos usará, para lo cual se ejecuta la siguiente instrucción:

```
php symfony configure:database --name=doctrine --class=sfDoctrineDatabase
' mysql:host=localhost; dbname=emeeting' root mySecret
```

Tipo de Usuario - Role	Owner	Presenter	Participant	Administrator
DigiMar Partner	X	X	X	X
External Partner		X	X	

Cuadro 4.1: Tipos de usuarios frente a tipos de registro en la aplicación

4.5. Especificación de la organización

En toda organización existen diferentes tipos de usuarios y cada usuario puede acceder a la aplicación y en este caso a las reuniones con diferentes roles como los que se pueden observar en el cuadro 4.1.

4.5.1. Definición de roles de acceso a la aplicación eMeeting

En esta subsección explicaremos los diferentes tipos de acceso que concede la aplicación eMeeting sobre las salas creadas en *Adobe Connect®*.

- Owner

Este tipo de usuario sólo es posible si formamos parte de los usuarios de DigiMar. Es el creador de la sala. En el servidor *Adobe Connect®* recibe el nombre de *mini-host*. Es el único con el privilegio de eliminar la sala y poder grabar las reuniones.

- Presenter

Este tipo de usuario es posible tanto si el usuario posee privilegios para acceder a DigiMar como si algún miembro de DigiMar ofrece alguna forma de acceso para usuarios externos. Se describirán estas formas de acceso más adelante, en las cuales el usuario puede interactuar en la sala pero no puede grabar reuniones.

- Participant

Es el usuario con menos privilegios, no le está permitido grabar ni interactuar en la sala, únicamente puede ver a los diferentes usuarios que participen en el eMeeting.

4.5.2. Definición de roles internos de eMeeting para “Campus do Mar”

La aplicación permite dos tipos de roles internos, de los cuales el administrador posee privilegios que hacen única la gestión de la aplicación.

- Administrador

Este usuario tiene acceso a la base de datos de la aplicación y puede manipular los contenidos a su voluntad. La siguiente lista hace referencia a las funcionalidades particulares de los administradores y nos ofrece una visión de las posibilidades de la aplicación.

- El administrador tiene acceso a las estadísticas de la aplicación. Mediante gráficos realizados con el API de Javascript JQUERY, podemos conocer el número máximo de salas concurrentes para tener una visión de la utilización de las salas, opción altamente útil, ya que las licencias de *Adobe Connect®* son un bien limitado. Del mismo modo, la aplicación genera correos electrónicos para los administradores cuando se llega a puntos críticos de utilización del servicio.
- El administrador puede dar privilegios a ciertas salas para que siempre se encuentren disponibles. Este es el concepto de sala mágica que se ha descrito en la subsección 4.2.1.

■ Usuario estándar

La mayoría de los usuarios poseerán este tipo de acceso que permite crear y borrar eMeetings dentro de la aplicación y la gestión de los mismos. Este tipo de usuario podrá invitar a sus reuniones a usuarios externos a la base de datos de “Campus do Mar”. Los usuarios externos serán monitorizados desde el acceso anónimo que pasaremos a describir a continuación.

■ Acceso anónimo

Cuando un usuario es invitado desde una URL generada para tal efecto, se le redirige a una pantalla de acceso, en la cual, se da la opción de ingresar en la aplicación como usuario de DigiMar o entrar como usuario anónimo.

La URL generada para este efecto tiene dos roles, presentador o participante, estos roles serán otorgados por el propietario (“owner”) de la reunión. El enlace para entrar como anónimo se construye a través de la URL que se le envía, apareciendo el parámetro *guest=true* en la URL, lo cual hace que por defecto el panel de acceso anónimo marque la opción de entrar como huésped. En cualquier otro caso la opción marcada por defecto será un formulario para entrar como usuario de DigiMar.

4.6. El interfaz de usuario

En esta sección describiremos las modificaciones y las decisiones de diseño que se han tomado en el interfaz de usuario desde el inicio de la implementación. Esta sección ayudará al lector en la interpretación de las diferentes funciones que se describirán a lo largo del capítulo y la relación de éstas con el interfaz de usuario.

Para comprender mejor las funcionalidades que ofrece el interfaz de la aplicación se describirá a continuación las entidades y los repositorios implementados para generar a base de datos y las funciones para acceder a todos los datos con las herramientas que ofrece Symfony.

4.6.1. Entidades y sus respectivos repositorios

En las entidades definiremos los campos que se guardarán en la base de datos, y una serie de sentencias sencillas para obtener valores de las tablas de la base de datos, tales como el nombre de una reunión, su identificador y el usuario creador.

Asociado a cada entidad tendremos su respectivo repositorio, donde estarán las sentencias a la base de datos que no implementa el *framework* por defecto. Como explicamos en el capítulo 2, Symfony nos ofrece la posibilidad de crear las tablas de la base de datos y una serie de consultas por defecto para acceder a los datos. Como es lógico hay una serie de consultas complejas necesarias para nuestra aplicación que no implementa el *framework*, por ello en los repositorios de las entidades se implementan estas consultas complejas.

- Meeting

A continuación y a modo de ejemplo se muestra el archivo de la entidad Meeting, sin duda la más importante de la aplicación, y que nos dará una idea de la estructura de la base de datos de la entidad central. A partir de la tabla que genera este código se dispondrá de una visión más clara de las funciones que describiremos en el interfaz de usuario. Las variables de esta tabla almacenan los diferentes estados en los que puede encontrarse una reunión, los identificadores de reunión o datos de creación.

```
/{*}{*}

{*} @var integer \$id

{*}

{*} @ORM\textbackslash{}Column(name='id', type='integer')

{*} @ORM\textbackslash{}Id {*}
{*} @ORM\textbackslash{}GeneratedValue(strategy='←
 AUTO')

{/}/

private \$id;

/*{/}{*}

{*} @var integer \$state

{*}
```

```
/* @ORM\textbackslash{}Column(name="state", type="integer")  
  
/*/  
  
private \$state = self::STATE\_NEW;  
  
/*{*/  
  
{*} @var string \$salt  
  
{*}  
  
{*} @Assert\textbackslash{}NotBlank()  
  
{*} @ORM\textbackslash{}Column(name="salt", type="string",  
length=255, nullable=false)  
  
/*/  
  
private \$salt;  
  
/*{*/  
  
{*} Link para entrar como usuario mini-host en ADO  
  
{*} @var string \$secretsalt  
  
{*}  
  
{*} @Assert\textbackslash{}NotBlank()  
  
{*} @ORM\textbackslash{}Column(name="secretsalt", type="string",  
length=255, nullable=false)  
  
/*/  
  
private \$secretsalt;  
  
/*{*/
```

```
 {*} Link para entrar como usuario view en ADO

{*} @var string \$viewsalt

{*}

{*} @Assert\textbackslash{}NotBlank()

{*} @ORM\textbackslash{}Column(name='viewsalt', type='string',
length=255, nullable=false)

{/}

private \$viewsalt;

/*{*}{*}

{*} @var string \$title

{*}

{*} @Assert\textbackslash{}NotBlank()

{*} @ORM\textbackslash{}Column(name='title', type='string',
length=255, nullable=false)

{/}

private \$title;

/*{*}{*}

{*} @var text \$description

{*}

{*} @ORM\textbackslash{}Column(name='description', type='text',
nullable=true)
```

```
/* */

private \$description;

/*}{*}

{*} @var string \$URL

{*}

{*} @ORM\textbackslash{}Column(name='URL', type='string',
length=255, nullable=true)

/*/

private \$URL;

/*}{*}

{*} @var string \$title

{*}

{*} @Assert\textbackslash{}NotBlank()

{*} @ORM\textbackslash{}Column(name='title', type='string',
length=255, nullable=false)

/*/

private \$title;

/*}{*}

{*} @var text \$description

{*}

{*} @ORM\textbackslash{}Column(name='description', type='text',
nullable=true)
```

```
{*/}/

private \$description;

/*}{*}

{*} @var string \$URL

{*}

{*} @ORM\textbackslash{}Column(name='URL', type='string',
length=255, nullable=true)

{*/}/

private \$URL;

/*}{*}

{*} @var datetime \$date

{*}

{*} @ORM\textbackslash{}Column(name='date', type='datetime')

{*/}/

private \$date;

/*}{*}

{*} @var boolean \$public

{*}

{*} @ORM\textbackslash{}Column(name='public', type='boolean',
nullable=true)
```

```
/* */

private \$public;

/*}{*}

{*} @var boolean \$public

{*}

{*} @ORM\textbackslash{}Column(name='magic', type='boolean',
nullable=false)

/*/

private \$magic;

/*}{*}

{*} @var User \$owner

{*}

{*} @ORM\textbackslash{}ManyToOne(targetEntity='User', inversedBy='id')

{*} @ORM\textbackslash{}JoinColumn(name='owner\_id', referencedColumnName<=
='id')

/*/

private \$owner;

/*}{*}

{*} @var Group \$group

{*}

{*} @ORM\textbackslash{}ManyToOne(targetEntity='Group', inversedBy='id',
cascade=\{'remove'\})
```

```
 {@*} @ORM\textbackslash{}JoinColumn(name='group\_id', referencedColumnName<--  
 = 'id')
```

```
{*}/
```

```
private \$group = null;
```

```
/{*}{*}
```

```
{*} @var ArrayCollection \$users
```

```
{*}
```

```
{*} @ORM\textbackslash{}ManyToMany(targetEntity='User', cascade=\{'remove'<--  
 , '\'})
```

```
{*} @ORM\textbackslash{}JoinTable(name='meeting\_user', joinColumns=\{@ORM\textbackslash{}--  
 textbackslash{}JoinColumn(name='meeting\_id',  
 referencedColumnName='id')\},
```

```
{*} inverseJoinColumns=\{@ORM\textbackslash{}JoinColumn(name='user\_id',  
 referencedColumnName='id')\} )
```

```
{*}/
```

```
private \$users;
```

```
/{*}{*}
```

```
{*} @var ArrayCollection \$recordings
```

```
{*}
```

```
{*} @ORM\textbackslash{}OneToMany(targetEntity='Recording', mappedBy='<--  
 meeting',  
 cascade=\{'remove'\})
```

```
{*}/
```

```
private \$recordings;
```

```
/{*}{*}

{*} @var ArrayCollection \$recordings

{*}

{*} @ORM\textbackslash{}OneToMany(targetEntity='NickName', mappedBy='←
    meeting',
    cascade=\{'remove'\}\})

{/}/

private \$nicknames;

/*{/}{*}

{*} @var integer \$concurrent

{*}

{*} @ORM\textbackslash{}Column(name='concurrent', type='integer')

{/}/

private \$concurrent = 0;

/*{/}{*}

{*} @var datetime \$created

{*}

{*} @ORM\textbackslash{}Column(name='created', type='datetime')

{/}/

private \$created;
```

```
/*{*}
 {*}<?var datetime \$updated
 {*}
 {*} @ORM\textbackslash{}Column(name='updated', type='datetime')
{/*}
private \$updated;
```

- Función que introduce datos al crear una reunión nueva:

```
public function __construct(User \$owner = null) \{
    \$this->owner = \$owner; \$this->created = \$this->updated = new \textbackslash{}DateTime('now');

    \$this->date = new \textbackslash{}DateTime('tomorrow');

    \$this->users = new \textbackslash{}Doctrine\textbackslash{}Common\textbackslash{}Collections\textbackslash{}ArrayCollection();

    \$this->recordings = new \textbackslash{}Doctrine\textbackslash{}Common\textbackslash{}Collections\textbackslash{}ArrayCollection();
}
```

- Función que devuelve el título de la reunión en forma de cadena de texto cuando solicitamos el título de esa variable de la forma `\$variable->getTitle()`:

```
public function __toString() \{
    return \$this->title;
}
```

- Función que devuelve el estado de la reunión.

Para cada una de las variables definidas en la entidad, tenemos una función get y set para recuperar o guardar la variable en el momento que precisemos y que será transparente para diferentes bases de datos, ya que el *framework* se encargará de construir la petición según la base de datos que introduzcamos en el fichero de configuración.

■ Usuario

Esta tabla posee todos los datos necesarios para identificar a un usuario que pertenece al sistema. Fecha de inicio en la aplicación, correo electrónico, nombre y contraseña. La lógica de esta entidad genera la tabla mostrada en la figura 4.4.2.

El repositorio de esta entidad posee las siguientes funciones:

- **findOneUserByEmail**: esta función encuentra un único usuario en la base de datos identificado por su correo electrónico. El correo electrónico es único para cada usuario y una herramienta potente para identificar usuarios de forma única. Las funciones que comienzan por *findOne* poseen una característica especial proporcionada por *Symfony*, esta característica consiste en proporcionar un error específico en caso de que se encuentre en la base de datos más de un usuario, es por ello, que se utiliza para realizar peticiones en las cuales se necesita un error en caso de existir más de una respuesta.
- **findUsersByTerm**: es la petición utilizada para la función autocompletar que se explicará en la sección 4.7. Esta petición recibe tres letras como mínimo y devuelve la entidad o entidades de usuario que contengan esas letras en el mismo orden en cualquiera de las columnas que forman la tabla de usuario.

■ Nickname

Esta entidad posee un identificador, el sobrenombre (*nickname*) que el usuario tendrá en cada reunión en la que participe, una breve descripción en caso de que fuese necesaria, la reunión y el usuario que donde se mostrará este sobrenombre y se utilizará esta tabla para almacenar el identificador de las reuniones que cada usuario haya minimizado en su interfaz como se puede ver en la figura 4.6.5.

Esta tabla posee las siguientes peticiones en su repositorio:

- **findOneNickNameByUserAndMeeting**: esta función devuelve la entidad *nickname* a partir de una entidad usuario y una entidad reunión.
- **findNickNameByUserOrderByRank**: devuelve la entidad *nickname* a partir de una entidad de usuario ordenada por orden de importancia.
- **findOneMinimizedByMeeting**: hace una petición a la base de datos para devolver una entidad *nickname* en caso de que la reunión se visualice como minimizada.

■ Grabaciones

Esta entidad posee un identificador propio, un identificador de reunión para conocer a qué reunión pertenece cada grabación, el estado de la grabación como se explicará

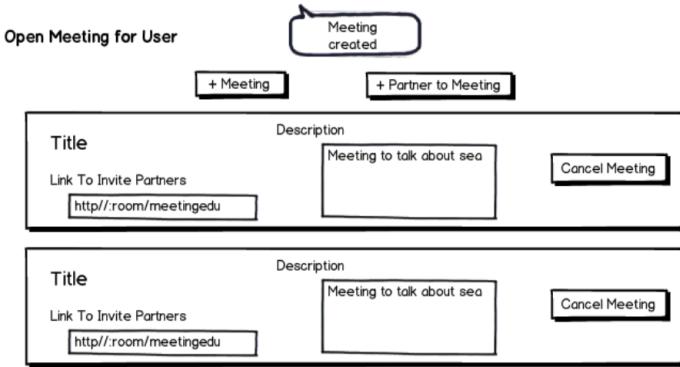


Figura 4.6.1: Maqueta de la primera versión

en la sección 4.4, el título de la grabación, la URL de acceso a la misma y la fecha de creación.

El repositorio de esta entidad posee la siguiente función:

- `findOneRecordingByTitle`: devuelve una entidad *recording* a partir del título de una grabación. Este hecho es posible debido a que el servidor *Adobe Connect®* no permite grabaciones con el mismo nombre.
- Estadísticas parciales y totales (Log y Log Total)

Ambas entidades son muy similares. Ambas poseen una columna para guardar el momento en el que se almacenan los datos. La primera posee el identificador de la reunión para relacionar los datos almacenados, el número de participantes de la reunión y la duración. La segunda posee el número de participantes totales en el servidor en el momento de la consulta del cron descrito en la sección 4.3 y el número de salas activas.

Estas entidades no poseen un repositorio ya que se actúa directamente sobre sus columnas para acceder a los datos. Por tanto las peticiones simples proporcionadas a la entidad por *Symfony* son suficientes.

Después de esta breve introducción a las entidades que forman la aplicación se describirá el interfaz de usuario.

Como se puede ver en la figura 4.6.1, así nació el primer interfaz de usuario para esta aplicación en el cual se tenía clara la información que se debe visualizar, pero no tanto la experiencia de usuario ante la visión de la información. Para el boceto de diseño de las primeras versiones de interfaz se ha utilizado la herramienta *Mockups* de *Balsamiq*[16].

En la figura 4.6.1 se observa lo esencial de una funcionalidad para crear una nueva reunión (botón “+ Meeting”) y un botón para agregar usuarios a la reunión (“+ Partner to Meeting”). Aparecen los datos que mostraremos para cada reunión, tales como el título, un enlace para invitar usuarios ajenos a la aplicación, una pequeña descripción y un botón

para eliminar la reunión (“Cancel Meeting”). En la parte superior central se vislumbra la idea de un mensaje superpuesto sobre la pantalla que alerta al usuario que acaba de crear una reunión. Este mensaje se desvanece progresivamente gracias a una funcionalidad de *jQuery UI* [12].

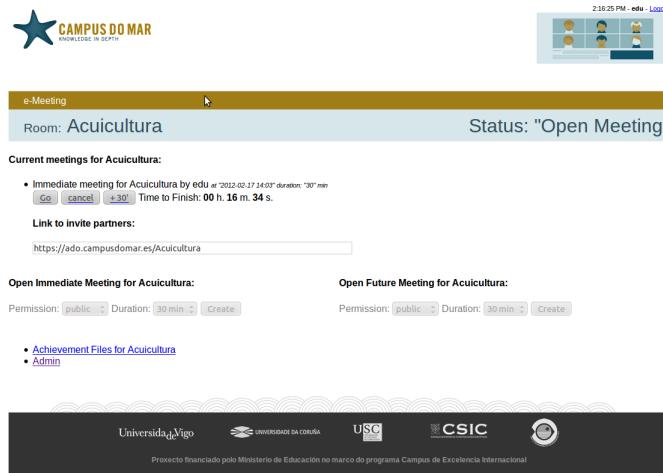


Figura 4.6.2: Primer interfaz de usuario visualizado en el explorador

A partir de la maqueta que observamos en la figura 4.6.1 se ha implementado un interfaz elemental como el de la figura 4.6.2, para facilitar un avance más diligente. Se puede concluir que no es muy atractivo, pero las funcionalidades más importantes, como el enlace para invitar a otros usuarios o las URLs para usuarios ajenos se han implementado y agilizarán la evolución del interfaz y de las funcionalidades finales. La necesidad de un interfaz más intuitivo en términos generales era obvia. El enriquecimiento del estilo también debía ocupar parte de nuestro tiempo para realizar un interfaz elegante y afín al usuario final. Decisiones y tiempo después se realizó un interfaz más atractivo y se empezó a evaluar la posibilidad de un panel de control en la parte superior del interfaz disponible en cualquier página de la aplicación que facilitase su navegación, como se puede observar en la figura 4.6.3. La presencia de *jQuery UI* se hacía indispensable para abordar problemas complejos de diseño y rematar la aplicación con un diseño sugerente. Todas las ventanas emergentes y los efectos visuales que aparecen al presionar los botones de la aplicación han sido diseñados con *jQuery UI* [12].



Figura 4.6.3: Versión final del interfaz de usuario

4.6.2. Botonera para cada reunión

Pasamos a describir las funciones de los botones que podemos contemplar en la figura 4.6.4. En esta figura podemos observar la botonera para administradores, a la izquierda, que posee el botón para cambiar una reunión al estado mágica mientras que la botonera del resto de usuarios no posee este botón.



(a) Botonera para no administradores (b) Botonera para administrador

Figura 4.6.4: Botonera para cada reunión

- Join

Permite el acceso a la sala para disfrutar de la videoconferencia. Cuando el usuario pulse este botón se abrirá un *iframe* en nuestro explorador que contendrá la ventana de *Adobe Connect®* con la URL que se haya configurado para esta sala. La URL de la sala cambia en función del estado de la misma, lógica que explicaremos con detenimiento en la sección 4.2.

- **Disabled**

Deshabilita la sala temporalmente. Este botón sólo está disponible para el dueño de la sala. Cuando se pincha en este botón se llama a la función asociada y mediante el método POST se pasa una variable verdadera o falsa en función de si la sala se encuentra habilitada o deshabilitada para alternar su estado entre uno u otro.

- **Delete**

Elimina la sala de la lista de reuniones disponibles en el interfaz de usuario. Sin embargo, se ha decidido por diseño que una reunión nunca se elimine por completo del sistema, es decir, se borra del servidor *Adobe Connect®*, pero nunca de la base de datos de la aplicación de forma que en el botón *Achievement Files* siempre estarán disponibles las reuniones eliminadas con su correspondiente lista de grabaciones.

- **Icono de flecha**

Este icono situado a la derecha del botón Delete, nos da la posibilidad de modificar la vista de la reunión. De esta forma el usuario puede observar más reuniones en la parte superior de la página, como se puede observar en la figura 4.6.4.

- **Recordings**

Proporciona el acceso a las grabaciones de la reunión a la que pertenece el botón. Las grabaciones que aparecen en la figura ya han finalizado, y aparecen después de ejecutarse la correspondiente función en el cron que se mencionó en la sección 4.3.

- **Change NickName**

Ofrece la posibilidad de cambiar el nombre que aparecerá en cada reunión de videoconferencia.

- También se muestra un párrafo con información sobre la sala:

1. Fecha de creación de la sala.
2. Tipo de sala, ya sea pública o privada.
3. Número de usuarios de “Campus do Mar” invitados a la reunión.

- **Sobre **

Al pulsar sobre este icono se despliega una ventana emergente como la de la figura. Nos muestra un texto destinado a usuarios ajenos a la aplicación de forma que si lo reciben puedan disfrutar de la sala. El texto contiene las instrucciones necesarias para entrar a la sala y la ventana emergente las necesarias para enviarlo.

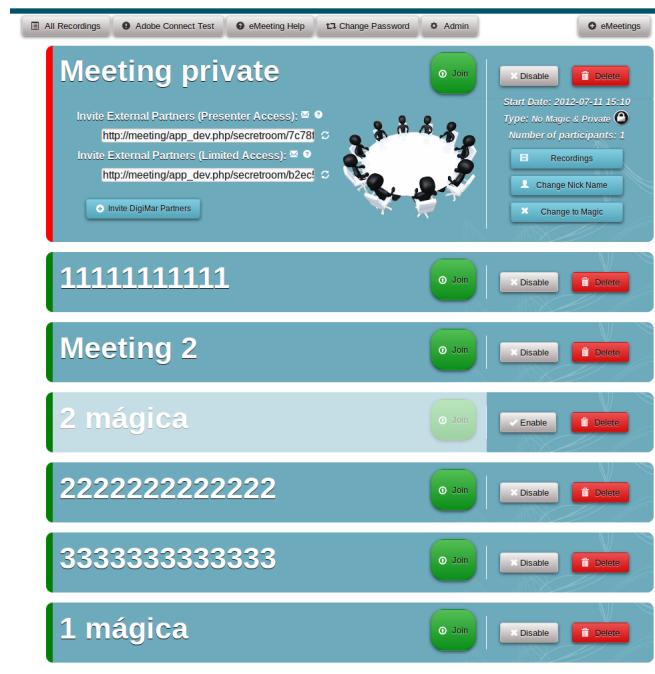


Figura 4.6.5: Lista de reuniones reducida

■ Interrogante

Este ícono aparece a lo largo de la aplicación y con la ayuda de jQuery emerge un mensaje cuando pasamos el puntero por encima que explica la funcionalidad del elemento al que acompaña.

■ Botón de recarga

Este ícono ofrece la posibilidad de modificar los enlaces para invitar a usuarios externos que se muestran en la figura 4.6.6. De esta forma podemos invitar a una lista de usuarios a una reunión con estos enlaces y en el futuro invalidar esos enlaces para invitar a otro grupo de usuarios de forma que los primeros no tendrán acceso a la reunión.

■ Url de enlace para otros usuarios:

• Presenter Access

Este enlace proporciona a su poseedor privilegios de presentador en la reunión. Un presentador podrá introducir contenido en la sala, como presentaciones o compartir su escritorio. Sin duda podrá compartir la imagen de su cámara web y el audio.

• Limited Access



Figura 4.6.6: Modos de invitación de usuarios a una reunión

Por el contrario, este enlace otorga la posibilidad de disfrutar de la reunión, pero no se podrá participar de forma activa en ella. Tampoco tendrá la opción de compartir su cámara web o su audio.

- **Invite DigiMar Users**

Se podrá invitar a una videoconferencia a usuarios ya registrados en la aplicación, a partir de una lista generada con la técnica de autocompletar. Para implementar esta ventana se ha empleado la librería jQuery y la tecnología AJAX.

4.6.3. Botonera superior de menú

Esta botonera situada en la parte superior del cuerpo central de la web, como se puede ver en la figura 4.6.7, nos proporciona el acceso a las funciones globales de la aplicación que pasamos a describir a continuación.



Figura 4.6.7: Menú superior

- *All Recordings*



Figura 4.6.8: Ventana emergente para *All recordings*

En este botón podemos acceder a una lista de reuniones como la mostrada en la figura 4.6.8 que han finalizado. Nos ofrece información sobre su fecha y hora de creación y la lista de grabaciones en caso de existir.

■ *Adobe Connect® Test*



Figura 4.6.9: Prueba de conexión de *Adobe Connect®*

Se accede a una prueba de conexión facilitada por *Adobe Connect®* para ofrecer al usuario la seguridad de que existe conexión con el servidor. Es un archivo Flash alojado en

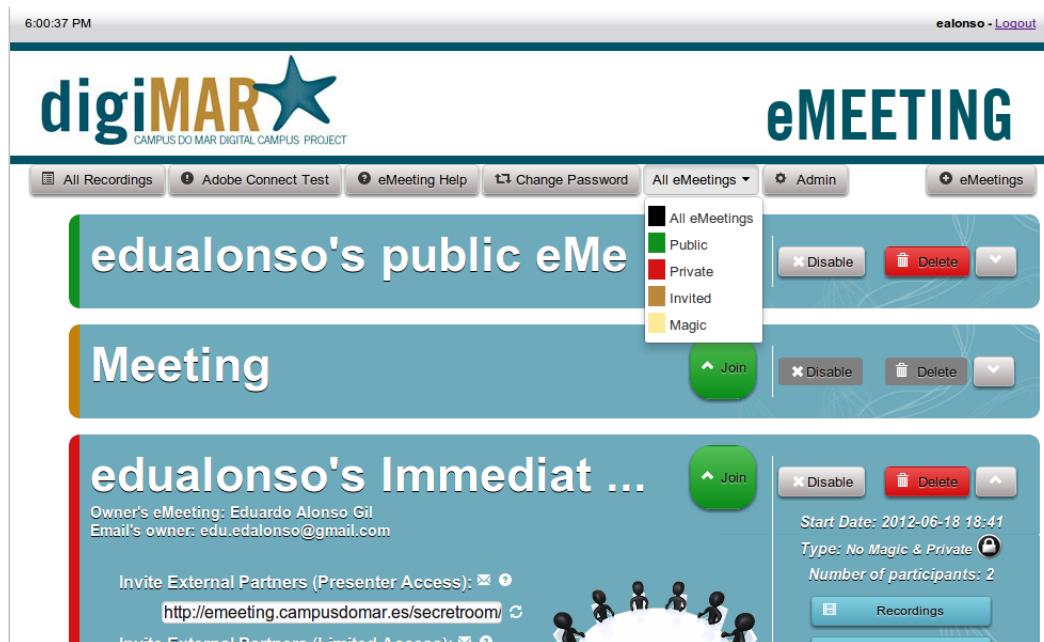


Figura 4.6.10: Desplegable para el filtro de reuniones

el servidor *Adobe Connect®*, que descarga un *applet* en nuestro navegador para realizar las pruebas de conexión similar al de la figura 4.6.9.

- *eMeeting Help*

Al pinchar sobre este botón se accede a una ayuda que indica las diferentes funcionalidades proporcionadas por *eMeeting* y explica como se utilizan. Los usuarios podrán resolver las diferentes dudas que puedan surgir sin más que ojear esta página. Ver figura 4.2.1

- *All eMeetings*

Este botón permite filtrar las reuniones de un tipo determinado, ya sean públicas, privadas, mágicas en caso del usuario administrador, las reuniones a las que ha sido invitado el usuario y la opción marcada por defecto, de ahí el nombre que vemos en la figura, que muestra todas las reuniones del usuario. Este desplegable puede verse en la figura 4.6.10, donde también puede observarse el código de colores relacionado con cada tipo de reunión.

- *Admin*

Sólo es accesible por los administradores de la aplicación y nos lleva a funcionalidades para actuar directamente sobre la base de datos o visualizar las estadísticas de la aplicación en tiempo real. Una de las funcionalidades a destacar es la posibilidad de poder modificar el estado de las reuniones a pesar de no ser el dueño. Ver figura 4.13.7.



Figura 4.6.11: Menú e creación de reuniones

- + eMeetings

En la ventana modal que se desplegará podremos seleccionar el título de nuestra reunión, un sobrenombre de usuario con el que nos conectaremos a la sala, invitar usuarios de “Campus do Mar” a nuestra nueva reunión y determinar el carácter de la sala (pública o privada). El campo descripción es opcional. Ver figura 4.6.11.

- Change Password

Se accede a una ventana modal que nos permite cambiar la contraseña de acceso a la aplicación.

4.6.4. Funciones PHP asociadas a la botonera

En esta subsección describiremos las funciones relacionadas con los botones que hemos descrito en las secciones anteriores. De esta forma podremos tocar en detalle las comprobaciones y la lógica que se realiza en cada botón y que permanece transparente al usuario.

- Join

Este botón realiza una serie de comprobaciones antes de acceder a la reunión correspondiente en *Adobe Connect®*. Comprueba el número de eMeetings simultáneos ya que el número está limitado por el número de licencias compradas por el administrador del servidor de conexión a *Adobe Connect®*. Comprueba que la sala todavía existe y que se tienen los privilegios necesarios para disfrutar el eMeeting, ya que el usuario creador ha podido deshabilitar la sala o incluso eliminarla antes de que la página se recargue y por tanto todavía aparezca como activa en nuestro interfaz.

- Enabled/Disabled

Con este botón el creador del eMeeting puede habilitar y deshabilitar la sala a su voluntad impidiendo la entrada a todos los usuarios, incluso aunque tengan acceso desde la puerta trasera creada en las salas privadas, acceso descrito en la sección 4.2.

- Delete

Con este botón eliminamos la sala de nuestro interfaz de usuario y también de todos los usuarios invitados al eMeeting. Únicamente el creador del eMeeting tiene el privilegio de eliminar su sala para evitar futuros conflictos y potenciar que los usuarios creen sus propias salas.

Se comprueba si el eMeeting posee grabaciones y se almacenan en una carpeta interna de *Adobe Connect®*.

En el botón antes descrito como “Achievement Files” tendremos acceso a las grabaciones de los eMeetings eliminados.

- Recordings

Con este botón podremos visualizar las grabaciones realizadas en el eMeeting hasta el momento. Se accede por AJAX al *EndPoint* correspondiente de Symfony para conseguir los valores actualizados de nuestra lista de grabaciones. En la sección 4.3 se ha hablado de la función que se ejecuta en el servidor cada minuto para almacenar las nuevas grabaciones en nuestra base de datos al finalizar la misma.

- Change Nick Name

En la ventana modal que despliega este botón se nos da la opción de cambiar nuestro “NickName” en cada reunión. Este código comprueba que no insertamos un “Nickname” vacío, aunque sería posible en el *Adobe Connect®* la posibilidad de no tener “Nickname”.

4.7. URLs de acceso a la aplicación

En esta sección, y para acompañar a las funcionalidades descritas en la sección 4.6, explicaremos las diferentes rutas que se han implementado para esta aplicación y que dan acceso a las funciones de los controladores. Symfony nos ofrece esta potente funcionalidad, de forma que podemos indicar la función que realizará la aplicación en función de la URL de acceso a la aplicación. A raíz de la explicación de estas rutas de acceso en la sección 4.2 el lector ya posee una visión más completa de la estructura de la aplicación y también del gran trabajo simplificado mediante la utilización de Symfony.

- index_personal ANY /personal/

Esta función que se encuentra en el archivo *Controller/PersonalController.php*, devuelve el usuario registrado, la lista de grupos a los que pertenece el usuario, la lista de reuniones con estado NOW y la lista de reuniones asociada a cada usuario con estado NOW que no pertenecen al grupo.

- group ANY /groupadmin/
Situado en *Controller/GroupAdminController.php*, devuelve un array con la lista de todos los grupos que posee la aplicación.
- group_show ANY /groupadmin/{id}²/show
Situado en *Controller/GroupAdminController.php*, recibe como parámetro el identificador del grupo que se visualizará en el formulario de edición para el menú de administrador.
- group_new ANY /groupadmin/new
Situado en *Controller/GroupAdminController.php*, crea un formulario para añadir un nuevo grupo en el menú de administrador.
- group_create POST /groupadmin/create
Situado en *Controller/GroupAdminController.php*, valida el formulario creado en group_new y actualiza la base de datos.
- group_edit ANY /groupadmin/{id}/edit
Situado en *Controller/GroupAdminController.php*, recibe como parámetro el identificador del grupo que se solicita editar en el menú de administración.
- group_update POST /groupadmin/{id}/update
Situado en *Controller/GroupAdminController.php*, valida el formulario de edición y actualiza la base de datos.
- group_delete POST /groupadmin/{id}/delete
Situado en *Controller/GroupAdminController.php*, borra un grupo desde el menú de administrador actualizando la base de datos.
- meeting ANY /meeting/
Situado en *Controller/MeetingController.php*, realiza una consulta a la base de datos para mostrar al administrador una lista con todas las reuniones de la aplicación.

²Los términos que aparecen entre llaves hacen referencia a la variable que se pasa por el método POST a cada una de las funciones.

- Las funciones meeting_show ANY /meeting/{id}/show, meeting_new ANY /meeting/new, meeting_create POST /meeting/create, meeting_edit ANY /meeting/{id}/edit, meeting_update POST /meeting/{id}/update, meeting_delete POST /meeting/{id}/delete situadas en *Controller/MeetingController.php*, son análogas a las mencionadas en los párrafos anteriores para los grupos y ofrecen al administrador las funcionalidades descritas y la validación de los datos, para actuar sobre la base de datos, de una forma casi instantánea para el programador.

- index_welcome ANY /welcome
Situada en *Controller/SecurityController.php*, recibe la dirección de correo electrónico que introduce el usuario al entrar en la aplicación, la valida y si existe nos redirige a la pantalla de entrada. También incluye el código que enviará un correo electrónico de alerta a un usuario si se intenta acceder a su cuenta y se cumple el máximo número de intentos de introducción de contraseña.

- mail_sent ANY /mail/sent
Situada en *Controller/SecurityController.php*, función que envía el correo electrónico en caso de que un intruso intente acceder a una cuenta de forma errónea.

- recover ANY /recover/{email}/{hash}
Situada en *Controller/SecurityController.php*, si recibe como parámetro POST una dirección de correo electrónico y una huella (*hash*) asociada al usuario dueño del correo electrónico, se cambiará la contraseña del usuario en la base de datos.

- recover_error ANY /recover/error
Situada en *Controller/SecurityController.php*, si la huella que pasamos en la función anterior es correcta o la dirección de correo electrónico no existe se supone un intento de ataque, para defender a la aplicación de ello se redirige a esta función que tomará las medidas oportunas.

- recover_update ANY /recover/update
Situada en *Controller/SecurityController.php*, si la función recover ha tenido éxito, se redirige a esta función que mostrará un mensaje informando de que la contraseña se ha modificado correctamente.

- Las funciones wizard ANY /wizard/, wizard_metadata ANY /wizard/metadata, wizard_metadata_submit POST /wizard/metadata_submit, wizard_date ANY /wizard/date, wizard_date_submit POST /wizard/date_submit, wizard_users ANY /wizard/users, wizard_users_submit POST /wizard/users_submit, wizard_persist

ANY /wizard/persist, wizard_error ANY /wizard/error, wizard_end ANY /wizard/end

Situadas en *Controller/WizardController.php*, realizan las operaciones necesarias para controlar las reuniones que tendrán lugar en el futuro. Operaciones tales como agregar una reunión futura, agregar un usuario a una reunión futura, cambiar la fecha a una reunión futura, controlar que los datos de la reunión sean coherentes, etc.

- recording ANY /recording/

Situada en *Controller/RecordingController.php*, devuelve todas las grabaciones de la base de datos.

- Las funciones recording_show ANY /recording/{id}/show, recording_new ANY /recording/new, recording_create POST /recording/create, recording_edit ANY /recording/{id}/edit, recording_update POST /recording/{id}/update, recording_delete POST /recording/{id}/delete

Situadas en *Controller/RecordingController.php*, de nuevo ofrecen las mismas funcionalidades descritas para las reuniones y para los grupos, en las cuales el administrador puede actuar directamente sobre la base de datos si fuese necesario.

- Para terminar con estas funciones específicas para el administrador de la aplicación user ANY /useradmin/, user_show ANY /useradmin/{id}/show, user_new ANY /useradmin/new, user_create POST /useradmin/create, user_edit ANY /useradmin/{id}/edit, user_update POST /useradmin/{id}/update, user_delete POST /useradmin/{id}/delete

Situadas en *Controller/RecordingController.php*, también realizan el tipo de operaciones relacionadas con la edición directa de la base de datos por parte del administrador.

- cmar_meeting_middleware_index ANY /middleware

Ésta es la ruta de acceso a las funciones del *middleware*. La llamada de “Directorio +” para recopilar o enviar datos a la aplicación, invocará esta ruta acompañada de la función o funciones que se soliciten.

- index ANY /

Función situada en *Controller/UserController.php*, envía la información necesaria a la plantilla que visualiza el usuario al entrar en la aplicación. Esta función hace una petición a la base de datos para obtener todos los usuarios de la base de datos, la entidad con los datos del usuario que realiza la petición, las reuniones que el usuario posee activas y los nombres que el propio usuario posee para cada reunión (Nicknames). Esta función también contiene los datos de las funciones futuras en caso de implementarse esta funcionalidad. Incluso se ha implementado la lógica para mostrar las reuniones activas de otros usuarios por si fuese de interés.

- **index_recording ANY /recording/{id}**

Función situada en *Controler/UserController.php*. Cuando se invoca acompaña del identificador de una grabación comprueba si la grabación está bloqueada devolviendo un mensaje con la información o en caso contrario, redirigiendo al usuario al servidor *Adobe Connect®* para visualizar la grabación. Sólo un usuario registrado en la aplicación podrá visualizar la grabación.

- **index_public_recording ANY /recording_public/{id}**

Función situada en *Controler/UserController.php*. Cuando se invoca realiza comprobaciones similares a la función anterior pero el usuario no precisa estar registrado en la aplicación. Característica muy útil si se trabaja con usuarios ajenos a la aplicación.

- **index_immediate ANY /immediate**

Función situada en *Controler/UserController.php*, para crear una nueva reunión en la base de datos. Se realizan las siguientes comprobaciones antes de crear la reunión:

1. En caso de elegir la opción de sala mágica se comprueba si se ha llegado al número máximo para este tipo de salas, en caso afirmativo se emite un error.
2. Se comprueba si el estado de la sala está entre los posibles, para evitar posibles ataques a la aplicación y obtener un error sobre la base de datos.
3. Los títulos para las salas en *Adobe Connect®* no pueden exceder de 60 caracteres, por tanto se realiza esta comprobación y si es necesario se da el error correspondiente.
4. Se comprueba si el título elegido existe en la aplicación ya que el servidor *Adobe Connect®* no permite títulos duplicados.
5. Se procede a crear la sala en la base de datos y en el servidor *Adobe Connect®*, para ello se invoca a la función *createMeeting*. Esta función se encuentra en el módulo de servicio para las reuniones en *Service/MeetingService.php*.
6. En caso de que todo sea correcto se muestra el mensaje de sala creada correctamente y en caso contrario el mensaje: *Error in ADO, you may need to change the title*, ya que es posible que el título elegido contenga algún carácter no permitido.

- **changeNickname ANY /changenick**

Función situada en *Controler/UserController.php*, que cambia el nombre que un usuario mostrará en una reunión determinada. Esta función comprueba que se introduce algún carácter y a continuación si el nombre que se introduce no es el mismo que ya existe en la base de datos para evitar modificaciones superfluas. Cabe mencionar que por defecto este nombre de usuario para cada reunión, en caso de que el usuario no especifique ninguno, será el nombre y apellidos.

- update_rank ANY /updaterank

Función situada en *Controler/UserController.php*, que ordena la lista de reuniones que visualiza el usuario en su cuenta. En el momento en el cual se accede a la aplicación y el usuario realiza la consulta para obtener la lista de reuniones, se pide a la base de datos que ordene esta lista según la variable *rank*, perteneciente a cada reunión. Esta variable *rank* es la que actualiza esta función. Esta actualización se realiza mediante una petición AJAX para actualizar la base de datos. La animación de movimiento de una reunión al pinchar sobre ella con el ratón se ha implementado con la librería jQuery UI.

- addusers_meeting ANY /addusers

Función situada en *Controler/UserController.php*. Para añadir usuarios a una reunión se pincha en el botón *Invite DigiMar Partners* y se despliega una ventana emergente en la cual podemos visualizar los usuarios que ya han sido invitados a la reunión. Esta ventana invoca a la función que se está describiendo, la cual realiza la petición a la base de datos con la lista de usuarios para dicha reunión. Esta ventana, mediante la técnica de autocompletado, nos ofrece una lista de usuarios a partir de tres letras que han de ser tecleadas en el correspondiente cajetín de entrada. La funcionalidad de autocompletado realiza una petición AJAX para obtener los usuarios que coinciden con las letras introducidas en dicho cajetín de entrada. Si pinchamos sobre un usuario, éste se añade a la caja de usuarios invitados a la reunión. Si por el contrario pinchamos en el aspa a la derecha de algún usuario invitado, éste será eliminado de la reunión. Las operaciones realizadas sobre esta ventana emergente se confirmarán pinchando en el botón *Save*.



Figura 4.7.1: Ventana emergente al pinchar en el botón *Invite DigiMar Partners*

- updateviewsalt_meeting ANY /updateviewsalt/{id}, updatesecretsalt_meeting ANY /updatesecretsalt/{id}

Situadas en *Controller/UserController.php*, ambas rutas comparten la misma función y en su interior se evalúa si el parámetro indicado es *viewsalt* o el *secretssalt*. Esta función ofrece entrada a las reuniones con diferentes privilegios: si se envía *viewsalt* los privilegios son limitados, acceso sólo para los que visualizarán la reunión pero no podrán participar en ella; si se envía el *secretssalt* se accederá a la reunión con privilegios de presentador.

- **index_cancel ANY /cancel/{id}**

Situada en Controller/UserController.php, es la función que invoca el botón Delete de cada reunión. En esta función se comprueba primeramente si la reunión ha finalizado ya, por si ocurriese algún problema de sincronización, en caso contrario pasamos a cambiar el estado de la reunión en la base de datos, lo cual eliminará las función de la lista de reuniones en el índice de la aplicación para el usuario. Para ello se invoca a la función *stop* en el servicio de reuniones.

- **index_edit ANY /edit/{id}**

Situada en Controller/UserController.php, despliega un formulario para modificar una reunión existente, comprobando de inicio si el usuario es el dueño de la reunión.

- **index_update POST /update/{id}**

Situada en Controller/UserController.php, actualiza la base de datos a partir de la información del formulario creado en *index_edit*.

- **index_search_meeting ANY /recordings/{meeting_id}**

Situada en Controller/UserController.php, crea una lista de grabaciones de una reunión para mostrarla en cuando se pincha en el botón Achievement Files.

- **index_historical ANY /historical**

Situada en Controller/UserController.php, genera la lista de reuniones pasadas o eliminadas cuando pinchamos en el botón Achievement Files.

- **index_month ANY /historical/{string_month}**

Situada en Controller/UserController.php, ofrece la funcionalidad de mostrar la lista de reuniones ordenadas por año y mes. Como se puede ver en la ventana emergente al pinchar en el botón *Achievement Files*, existe un menú para seleccionar año y mes, este menú realiza una llamada a esta función para servir la lista de reuniones correspondientes a esta fecha. También se puede ver que se muestra el número de grabaciones para esa fecha, consulta realizada por esta función para mostrarse en la ventana emergente de la figura 4.7.2.



Figura 4.7.2: Lista de grabaciones de un usuario

- `change_password ANY /change_password/{email}`

Situada en `Controller/UserController.php`, funcionalidad asociada al botón *Change Password* situado en el menú superior de la aplicación, ver figura 4.6.4. Esta función busca el usuario en la base de datos a partir de su dirección de correo electrónico, para luego actualizar la base de datos con la contraseña introducida en la ventana emergente. Si el método utilizado no es POST se reporta un error ya que se ha detectado un intento de ataque sobre esta función.

- `recording_list ANY /r_list/{id}`

Situada en `Controller/UserController.php`, está asociada a la petición AJAX que realiza el botón *Recordings* de cada reunión. Devuelve el usuario dueño de la reunión y la entidad que representa la reunión.

User edit

The screenshot shows a user editing form with the following fields and values:

- Login: ealonso
- Password: (empty)
- Name: Eduardo
- Surname: Alonso Gil
- Email: edu.edalonso@gmail.com

Below the form is an "Edit" button and a list of links:

- [Back to the list](#)
- [Delete](#)

Figura 4.7.3: Plantilla para la edición de usuarios por el administrador

- **locked_recording ANY /locked_record/{locked}/{id}**

Situada en Controller/UserController.php, bloquea las grabaciones cuando pulsamos en el botón *pausa*, ver figura 4.13.1, asociado a cada grabación o la desbloquea en caso de que pulsemos en dicho botón y la grabación esté bloqueada. Este botón realiza una petición AJAX sobre esta función para que actualice la base de datos.

- **recording_public_list ANY /r_public_list/{secretsalt}**

Situada en Controller/UserController.php, ofrece una lista de grabaciones cuando enviamos a usuarios ajenos a la aplicación la URL que aparece en la parte superior de la ventana emergente que se despliega al pulsar en el botón de grabaciones.

- **locked_meeting ANY /locked/{locked}/{id}**

Situada en Controller/UserController.php, bloquea las reuniones cuando pulsamos en el botón *Enabled/Disabled* asociado a cada reunión, ver figura 4.6.4. Con tecnología AJAX este botón cambia al estado deshabilitado (*enabled*) cuando la reunión está deshabilitada, de forma que si se pulsa la reunión se desbloquea. Este botón realiza una petición AJAX sobre esta función para que actualice la base de datos.

- **user_form ANY /u_form/{id}**

Situada en Controller/UserController.php, devuelve los datos del usuario indicado por el identificador que se le pasa mediante POST a la función que genera el formulario para el menú de usuarios del administrador. La plantilla generada puede verse en la figura 4.7.3.

- **user_list ANY /u_list**

Situada en Controller/UserController.php, devuelve la lista de usuarios coincidentes con las letras introducidas en el cajetín de búsqueda de usuarios por AJAX, cuando se quiere añadir nuevos usuarios a una reunión.

- admin ANY /admin

Situada en Controller/UserController.php, consulta las tablas Log y Log_Total de la base de datos y envía los datos a la plantilla que genera las tablas estadísticas, con el número de participantes activos para cada reunión y totales, el tiempo de duración de las reuniones activas, la fecha de inicio de la reunión, número de reuniones activas, la fecha y hora de la consulta de la base de datos para completar la tabla Log_Total y el identificador único para la reunión dentro del servidor *Adobe Connect®*.

- indexgroup ANY /group/{key}

Situada en Controller/GroupController.php, comprueba que existe el grupo solicitado, a continuación que el usuario que solicita la información pertenece al grupo y finalmente devuelve a la plantilla las reuniones actuales, las reuniones programadas y la entidad que describe las características del grupo.

- indexgroup_recording ANY /group/recording/{id}

Situada en Controller/GroupController.php, redirige a la grabación del grupo solicitada.

- indexgroup_immediate ANY /group/immediate/{key}

Situada en Controller/GroupController.php, construye una reunión para un grupo determinado de manera que todos los componentes del grupo serán invitados a la reunión creada.

- indexgroup_cancel ANY /group/cancel/{id}

Situada en Controller/GroupController.php, actualiza el estado de una reunión a *CANCELLED* con lo cual ya no aparece en la lista de reuniones para los usuarios pertenecientes al grupo y pasa a formar parte de la lista de *Achievement Files*.

- indexgroup_delete ANY /group/delete/{id}

Situada en Controller/GroupController.php, elimina la reunión de la base de datos. Esta función no se utiliza en la versión final ya que se ha preferido mantener la información de todas las reuniones en la base de datos y únicamente actualizar su estado para poder acceder a información si fuese necesario en caso de que el usuario elimine su reunión.

- indexgroup_edit ANY /group/edit/{id}

Situada en Controller/GroupController.php, crea el formulario necesario para editar información a cerca de una reunión de grupo ya creada.

- indexgroup_update POST /group/update/{id}

Situada en Controller/GroupController.php, la información de la reunión descrita en el formulario creado en la función anterior se valida en esta función y se actualiza en la base de datos.

- indexgroup_historical ANY /group/historical/{key}

Situada en Controller/GroupController.php, crea una lista con las reuniones de un grupo que ya han finalizado y poseen el estado *CANCELLED*.

- indexgroup_month ANY /group/historical/{key}/{string_month}

Situada en Controller/GroupController.php, devuelve la lista de reuniones de grupos con sus respectivas grabaciones. La petición se hace para un mes y año concretos, de forma similar al *Achievement Files* para las reuniones de usuarios individuales. A la vez que se visualiza el número de reuniones para un mes en la lista desplegable, también se puede saber el número de reuniones para ese mes.

- index_room ANY /room/{salt}, index_secretroom ANY /secretroom/{salt}, index_noanonymousroom ANY /noanonymousroom/{salt}, index_noanonymoussecretroom ANY /noanonymoussecretroom/{salt}

Situadas en Controller/RoomController.php, todas estas URLs llaman a la misma función. Implementa un control de acceso a las reuniones a partir de una URL secreta generada con el algoritmo MD5 a partir del nombre de la reunión y otros datos, a parte de una huella aleatoria. El usuario que posea la URL podrá acceder a la reunión con unos privilegios determinados en función de si la URL que posee contiene el parámetro *salt*, *secret salt* o *viewsalt* de la reunión. Para dar acceso a una reunión se comprueba ese parámetro de la URL y el estado de la reunión, de forma que el usuario que accede a la misma podrá ser presentador o asistente. El acceso también podrá ser anónimo, es decir, un usuario que no tenga una cuenta en la aplicación también tendrá acceso a través de las URLs que no contienen *noanonymousroom* o *noanonymoussecretroom*. Estas URLs no redirigen a sus correspondientes plantillas para entrada de acceso anónimo o para que accedamos a un menú de registro en la aplicación. De esta forma la aplicación siempre tendrá el control sobre los usuarios que acceden, pertenezcan o no a la lista de usuarios registrados en la aplicación. Para obtener control total de los privilegios de acceso de usuarios anónimos, no registrados en la aplicación, se crea un usuario temporal en el servidor *Adobe Connect®*, que luego será eliminado desde la función del cron correspondiente como se ha explicado en la sección 4.3.

- login ANY /login

Esta ruta redirige a la plantilla de entrada a la aplicación, si se accede a una ruta configurada como privada en el archivo *security.yml*. Es una ruta por defecto que ofrece

el *framework* Symfony para simplificar el intercambio entre mecanismos de registro en aplicaciones y para simplificar el acceso de usuario anónimos para la aplicación. Esta ruta, al igual que las dos que describiremos a continuación, se encuentra en el fichero *routing.yml*. Nos ofrece un formulario para acceder a través del correo electrónico del usuario y su contraseña asociada, además de un enlace a un formulario de recuperación de contraseña. Para mayor seguridad el formulario de recuperación de contraseña envía un correo electrónico a la cuenta indicada y se genera un enlace único enviado a ese correo electrónico que nos redirige a un nuevo formulario para cambiar la contraseña directamente en la base de datos.

- `login_check ANY /`

Esta ruta que pertenece al fichero *routing.yml* controlará el envío de formularios de inicio de sesión. Es decir todo usuario que intente acceder a la aplicación será reenviado a la función `loginAction` del archivo `Controller/SecurityController.php`. En este archivo se evaluará el resto de la URL de acceso para comprobar el tipo de acceso que hemos definido en el archivo *security.yml*.

- `logout ANY /logout`

Esta función perteneciente al fichero *routing.yml* realiza la salida de la aplicación. Es una función interna a la lógica de Symfony, al igual que las dos anteriores y ofrece potentes mecanismos de seguridad.

4.8. Interfaz de *Adobe Connect®*

En esta sección describiremos el interfaz de *Adobe Connect Professional®* al que se accede al pinchar en el botón *Join* de la aplicación. Como ya se ha comentado en la sección 3.1 y se puede observar en la figura 4.8.1 el interfaz de videoconferencia lo ofrece *Adobe Connect Professional®*, de todas formas este interfaz posee varios modos que pueden ser configurados desde la aplicación “eMeeting”. En cada uno de estos modos la disposición de los elementos de la pantalla cambia para ofrecer interfaces apropiados a cada uso, como puede ser una videoconferencia múltiple, una ponencia con presentaciones de diapositivas o reuniones colaborativas donde las herramientas de intervención son más accesibles. En la figura podemos observar las funcionalidades que nos ofrece la aplicación tales como un chat, la lista de usuarios con sus roles, el menú para actuar sobre la reunión para poder grabarla o finalizarla, los diferentes diseños para la plantilla o la posibilidad de grabar el audio, escuchar otros audios o incluso compartirlos en la videoconferencia.

4.9. Directorio +

Es un servidor que almacena los datos de todas las aplicaciones y ofrece un servicio de sincronización e información. Comercialmente recibe el nombre de “Directorio +” y proveerá



Figura 4.8.1: Interfaz de videoconferencia de *Adobe Connect®*

a todas las aplicaciones de “Campus do Mar” de información para cargar sus bases de datos, tales como usuarios, sus datos personales, pertenencia a determinados grupos, contraseñas y otros datos de interés. Es la parte más compleja del sistema ya que en combinación con el servidor CAS, que se explicará en la sección 4.10 y ofrecerá un servicio de autenticación, permitirá registrarse a todos los usuarios en todas las aplicaciones registrándose únicamente en una de ellas. La información modificada en cualquier aplicación se sincronizará con el resto gracias al “Directorio +”. Este servidor realizará peticiones periódicas a todas las aplicaciones, incluida “eMeeting”, para actualizar datos. Este flujo de datos se realizará en ambos sentidos y con un formato determinado. La descripción de los datos solicitados se almacena en archivos XML y “Directorio +” ejecutará las funciones que necesite para actualizarse adecuadamente. La descripción de las funciones que ejecutará Directorio + referentes a eMeeting se almacenan en un fichero llamado `middleware.wsdl` y que ambas partes poseen. Evidentemente el archivo de “eMeeting” sólo contiene sus funciones. Las funciones asociadas a este archivo y la descripción de su contenido pasamos a describirlas a continuación.

4.9.1. El Middleware

Middleware, es un término utilizado para referirse al *software* que media para conectar dos entidades *software* en un sistema distribuido.

En el fichero `middleware.wsdl` indicaremos el formato de información que directorio plus

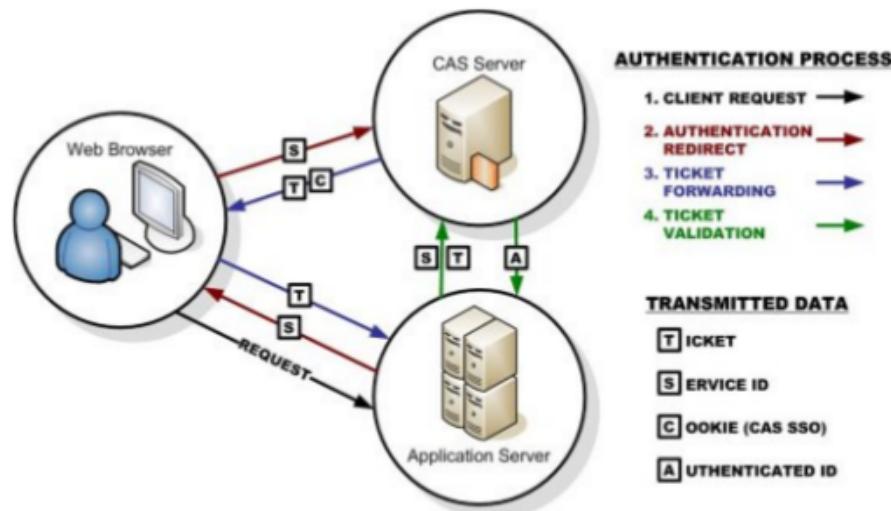
envía a nuestra aplicación en formato XML. Esta información se procesará en las distintas funciones implementadas en la aplicación “eMeeting” para llenar los campos necesarios en nuestra base de datos, para los usuarios y grupos pertenecientes a la plataforma “Campus do Mar”. Algunas de estas funciones son: *createUser*, *EnjoinGroup* o *addGroup*. Cada una de estas funciones, a partir de los flujos generados por “Directorio +” hacia nuestra aplicación, adaptan los datos enviados para almacenarlos en la base de datos, necesarios para la sincronización de la aplicación “eMeeting” de forma completamente transparente para el usuario. La primera función añade un nuevo usuario a nuestra aplicación cuando un nuevo usuario es añadido a “Directorio +”. Si un nuevo usuario es añadido a un grupo se invoca la segunda función y en caso de crear un nuevo grupo se invocará la tercera función.

4.10. Seguridad de la Aplicación

En el fichero `security.yml` de Symfony tenemos configuradas tres opciones de registro en nuestra aplicación. La ofrecida por html con un desplegable donde se puede introducir login y contraseña, el método de configuración de servidores de control de acceso (entidades certificadoras) o la que ofrece Symfony mediante la redirección a un formulario que se puede diseñar a nuestro gusto como puerta de acceso a nuestra aplicación.

De esta forma podemos alternar fácilmente entre tres tipos de registro sin más que comentar y descomentar unas cuantas líneas en el fichero `security.yml`.

Como aplicación independiente descrita en el capítulo 3 de diseño, es imprescindible que “eMeeting” ofrezca estos diferentes tipos de seguridad y registro de usuarios. Una de las más interesantes es la posibilidad de registrarse a través de un servidor CAS, que pasamos a describir.



Cuadro 4.2: Servidor CAS

CAS (*Central Authentication Service*) es una aplicación web que nos permite implementar el conocido SSO (*Single Sign On*) que es un procedimiento de autenticación que habilita a un usuario para acceder a distintas aplicaciones web (en distintos dominios y en distintos servidores) con hacer login una única vez. En la figura 4.2 se describe el intercambio de tickets entre las entidades que realizan el registro. En general, cuando un usuario se conecta a una de las aplicaciones involucradas en el sistema comprueba si está autenticado y si no lo está lo redirige a la pantalla del servidor de autenticación. Si la autenticación es correcta el sistema de autenticación, en este caso CAS, vuelve a redirigir al usuario a la página que se solicitaba en un primer momento. El desarrollador no tiene que preocuparse por mantener la seguridad y tener un formulario de login en cada una de las aplicaciones web que desarrolla, sino que simplemente se realiza la comprobación de si el usuario ya está registrado, que es tan sencillo como efectuar una petición request de esta forma: `request.getRemoteUser`. Si el método devuelve null es que no está registrado y será redireccionado a la página de login de CAS. Si ya está registrado por CAS, este método devolverá un valor con el que sabremos que es un usuario registrado. Algo que debe quedar muy claro es que CAS se encarga únicamente y exclusivamente de la autenticación, es decir, de comprobar contra una fuente de datos específica si el usuario y contraseña facilitados existen y son correctos. No se encarga de la autorización, que sería la gestión de lo que puede o no puede hacer ese usuario en función de sus roles.

Para entrar en la aplicación eMeeting debe realizarse un control de acceso mediante dicho servidor CAS, implementado para todas las aplicaciones de “Campus do Mar” introduciendo usuario y contraseña en el formulario de la figura 4.10.1a.

Para “eMeeting” también se ha implementado la posibilidad de acceder a través de un formulario creado con Symfony. Symfony compara la información introducida con la almacenada en la base de datos de la aplicación, para permitir el acceso a la aplicación.

Como se observa en la figura 4.10.1b, el acceso es sencillo y posee un enlace para recuperar la contraseña en caso de que el usuario la olvide.

Por último se ha implementado la opción de registro que ofrece HTML con un formulario sencillo que accede a la base de datos para comprobar que el usuario y contraseña introducidos son correctos como se muestra en la figura 4.10.1c.

El fichero `routing.yml` es el único que se debe modificar para alternar entre todas las posibilidades de acceso implementadas. Cabe resaltar que incluso se puede definir el tipo de acceso para cada una de las rutas descritas en la sección 4.7. La parte útil de este híbrido denominadas públicas sean accesibles por usuarios no registrados en la aplicación, como por ejemplo, las grabaciones de las reuniones. Para el acceso a las zonas públicas se ha implementado un control de acceso manejado por el usuario.

4.11. Comandos para una gestión de mayor comodidad

En esta sección describiremos una lista de comandos creados para gestionar diferentes parámetros de la aplicación. Uno de los usos más importantes de estos comandos es la



(a) Formulario de registro en el servidor CAS

The screenshot shows a login form for "CAMPUS DO MAR" with a starfish logo. It includes fields for "Email" and "Password", a "Login" button, and a "Forgotten it?" link.

(b) Formulario de acceso con diseño "Campus do Mar"

The screenshot shows a modal dialog box asking for permission to access the user's information. It contains fields for "Nombre de usuario" and "Contraseña", and buttons for "Cancelar" and "Aceptar".

(c) Formulario básico que ofrece HTML

Figura 4.10.1: Formularios de registro

posibilidad de ejecutarlos a través de un cron o *bot* en el servidor para realizar tareas automatizadas de mantenimiento o extraer estadísticas.

- Importación de usuarios definida en la base de datos interna de la aplicación

La aplicación posee una base de datos interna que carga una serie de usuarios por defecto definidos en el corazón de la aplicación que facilita el testeo de la aplicación, y sus respectivas pruebas en un nuevo servidor. En el fichero *DataFixtures/ORM/LoadUserData.php* creamos los usuarios que serán introducidos por defecto en la base de datos cuando ejecutemos el comando `cmar:import:user`.

- Estado de las reuniones

El comando `cmar:meeting:cron` nos permite conocer el estado de las reuniones y testear las últimas funcionalidades de manera ágil y rápida.

- Comprobación de la sincronización de usuarios

El comando `cmar:sync:user`, nos permite comprobar si los usuarios de la base de datos interna de la aplicación existen en servidor de *Adobe Connect®*. En caso contrario los usuarios no podrán disfrutar de videoconferencias dentro de la aplicación como usuarios internos.

- Comprobación de conectividad con el servidor de *Adobe Connect®*

El comando `cmar:test:meeting`, nos permite comprobar si el servidor donde alojaremos la aplicación posee conectividad con el servidor donde está alojada la aplicación *Adobe Connect®*. Una forma rápida de detectar errores de conexión con dicho servidor nos ayudará a descartar errores con mayor agilidad.

4.12. Servicios de la aplicación

Los servicios están descritos en el fichero de configuración *Resources/config/services.xml*. En este fichero debemos configurar el tipo de dato de entrada para cada servicio a usar en la aplicación. Este fichero en formato XML guarda en sus variables los datos de conexión al servidor *Adobe Connect®*. Estas variables son la URL donde está alojado el servidor *Adobe Connect®* con los datos de registro del usuario administrador para este servidor, ya que es necesario para utilizar todas las opciones de la API. También se encuentran aquí los datos que nos limitan el uso del servidor como son el número total de salas contratadas en el servidor. Definimos aquí también el número de salas que usaremos como mágicas y por tanto también el número de salas que serán de libre acceso para cualquier usuario, de esta forma si deseamos modificar este número sólo debemos cambiar este archivo. Como ya comentamos dentro del servidor existe una carpeta en la que almacenaremos todas las grabaciones de las reuniones que los usuarios borren. El identificador de esta carpeta que será permanente durante la vida de cada aplicación, se configurará también en este archivo. Pasamos ahora a describir los servicios de la aplicación y los parámetros relativos a cada servicio.

- Servicio de Meeting

Este servicio recibe como parámetros desde el fichero *services.xml* el identificador de Doctrine, usado para el acceso a la base de datos y el identificador del servicio AdoAdmin, debido a que se usarán gran cantidad de las funciones que se crean en este fichero para acceder al servidor *Adobe Connect®*. Se le envía el número total de salas disponibles, o lo que es lo mismo, el máximo número de licencias de *Adobe Connect®* disponibles, el número de salas simultáneas y el número de salas mágicas. Por último también se envía el identificador del servicio de registro ya que es necesario que estas funciones se encuentren disponibles para posibles accesos a las salas y al interfaz de usuario. El servicio de registro nos permite obtener los datos del usuario que accede a la aplicación para poder ofrecerle sus parámetros y los privilegios que posee dentro de la aplicación.

En este servicio proporcionamos a la aplicación las funciones para actuar sobre las reuniones de la aplicación y para llamar a funciones de la API de *Adobe Connect®* relacionadas con las reuniones de nuestra aplicación.

- Función para crear reuniones en la base de datos de la aplicación

En este servicio se aloja la función que crea las reuniones en la base de datos de la aplicación. Recoge los datos que le ofrece el formulario de creación de reuniones de la aplicación como el título, el usuario que la crea, el sobrenombre (*nickname*) que ese usuario deseé mostrar para esta reunión en concreto, por defecto su nombre y apellidos, un parámetro que indica si la sala es pública o privada, una pequeña descripción de la reunión si se desea, y para los usuarios administradores también aparece la opción de elegir si la sala será mágica o no como se ha descrito en la subsección 4.2.1.

- Función para crear reuniones en el servidor *Adobe Connect®*

En este servicio se aloja la función que crea las reuniones en el servidor *Adobe Connect®*. Cuando pulsamos el botón “Join” de la aplicación se ejecuta esta función que mediante el uso del servicio de administrador de *Adobe Connect®* crea la reunión en el servidor *Adobe Connect®* y nos introduce en la sala. También se cambia el estado de la sala y se almacena en la base de datos la URL que, para *Adobe Connect®*, identificará esta sala.

- Función para eliminar reuniones en el servidor *Adobe Connect®*

En esta función se hace una búsqueda de la reunión en la base de datos de *Adobe Connect®* mediante la URL que nos proporcionó en la función para crear las reuniones. Una vez identificada la reunión se procede a guardar sus grabaciones, si las hubiese, en una carpeta dentro del servidor *Adobe Connect®* destinada a tal fin. Acto seguido se cambia el estado de la reunión en la base de datos de la aplicación finalizada.

- Función para añadir usuarios del “Campus do Mar” a reuniones en el servidor *Adobe Connect®*

Esta función recibe como parámetros la entidad reunión y una lista con los usuarios que debe contener la reunión, por tanto, eliminamos todos los usuarios de

la reunión menos el creador y añadimos todos los usuarios que contiene la lista recibida. Debido a que es posible que un usuario quiera borrar únicamente al propietario de la reunión y no al resto se añade una comprobación a mayores en la cual si en la lista recibida no está el propietario de la reunión, se devuelve un error y no se modifica la lista de usuarios de la misma.

- Función para crear usuarios en el servidor *Adobe Connect®*

Esta función recibe todos los campos necesarios para crear un usuario en el servidor *Adobe Connect®*, login, nombre, apellidos, correo electrónico, descripción y contraseña. Estos datos se introducen en el servidor con una llamada **principal-update** de la API de *Adobe Connect®*.

- Control de acceso a las reuniones en el servidor *Adobe Connect®*

Esta función devuelve verdadero o falso tras realizar una serie de comprobaciones que indican si está permitido el acceso a una determinada reunión, comprobando si el usuario está en la base de datos o es un usuario anónimo, si la reunión está bloqueada, si el usuario pertenece a lista de invitados o es el creador, y si, en caso de ser un usuario anónimo, accede desde la URL correspondiente para tal fin.

- Función para cambiar el sobrenombre (*nickname*) de usuario

La aplicación permite tener un nombre corto para cada reunión. De esta forma en la actual función implementamos una lógica que nos permite modificar el nombre corto asociado a cada usuario en cada reunión.

- Función para cambiar el hash de la entrada secreta

En esta función se implementa un cambio aleatorio de una clave que nos permite entrar a una sala con ciertos privilegios otorgados por el creador de la reunión que aloja dicha sala. Podríamos decir que es una llave maestra que nos otorga el creador de la sala para poder entrar en ella, en ocasiones con los mismo privilegios que el creador. Esta función en concreto implementa la lógica que nos permite modificar dicha llave maestra para que los usuarios que tengan la llave maestra obsoleta ya no puedan de disfrutar de la sala y se les tenga que enviar la nueva llave otra vez.

- Función para cambiar el hash de la entrada común

Al igual que para la entrada secreta para la entrada común también se implementa un hash con el algoritmo MD5 que nos permite el acceso de forma común a la sala e identifica cada reunión.

- Función para cambiar la contraseña de usuario

En esta función implementamos una llamada a la API de *Adobe Connect®* para cambiar la contraseña de usuario en dicho servidor, ya que la contraseña del usuario deben estar sincronizadas tanto en el servidor *Adobe Connect®* como en el servidor de la aplicación.

- Servicio de aplicación de *Adobe Connect®*

En este servicio se implementan dos funciones que relacionan de forma directa la aplicación con el servidor *Adobe Connect®*. Una obtiene los datos para registrarse con privilegios de administrador y poder controlar la API y la otra crea usuarios nuevos en el servidor dando las credenciales oportunas.

■ Servidor

Esta función obtiene las credenciales del servidor donde se aloja la aplicación *Adobe Connect®*.

• Cookies

Se refiere a diferentes funciones que guardan y rescatan la cookie de sesión que el servidor *Adobe* ofrece cada vez que realizamos un login.

• Función de login

Esta función realiza el login en el servidor *Adobe* con las credenciales del usuario que realiza la petición desde la aplicación. Realiza una serie de peticiones utilizando la función curl [4] accediendo a la API de *Adobe* y procesar las respuestas del servidor para determinar si la operación se ha realizado con éxito. Se han diseñado una serie de excepciones para determinar posibles errores en el login y atajarlos ágilmente.

■ Servicio de administrador de *Adobe Connect® AdoAdmin*

Desde el archivo *services.xml* se envía la URL donde se aloja el servidor *Adobe Connect®*, el usuario y contraseña de acceso para el usuario administrador, la carpeta donde se alojarán las grabaciones cuando eliminamos una sala y el servicio de registro.

En este servicio albergamos los comandos de la API de *Adobe Connect®* para realizar las diferentes acciones sobre el servidor. Algunas de estas acciones son la creación de usuarios, la creación de reuniones, el cambio de contraseña para un usuario o la actualización de cualquiera de las entidades que alberga el servidor *Adobe*.

■ Factoría de *Adobe*

Desde el archivo *services.xml* se envía la URL donde se aloja el servidor *Adobe Connect®* y el servicio de registro.

Proporciona funciones para utilizar el API de *Adobe Connect®*. Desde este servicio obtenemos el nombre del servidor donde se aloja *Adobe Connect®* para poder comunicarnos con él. También se obtiene el usuario que está conectado en ese momento y nos da la capacidad de hacer una llamada a la administración de *Adobe* para crear un usuario en caso de que no estemos ya registrados en el servidor.

■ Interfaz de administración para *Adobe Connect®*

En el interfaz de administración debemos añadir las cabeceras de las funciones más importantes de la aplicación como son crear y borrar reuniones.

- Servicio de Mail

Este servicio nos permite utilizar las herramientas de Symfony para enviar correos tanto a los usuarios como a los administradores.

- Servicio de Middleware

Desde el archivo *services.xml*, se envía el identificador de Doctrine, el servicio AdoAdmin, el servicio de validación y el servicio de registro. El servicio de validación nos permitirá validar los campos de nuestra clase *Profile* (esto puede ser inyectado en cualquiera de nuestros servicios que necesitan validar un objeto).

Este servicio contiene las acciones que se ejecutan en la aplicación para las diferentes llamadas desde Directorio +, tales como actualizar datos de usuario o de reuniones.

- Servicio EntityListener

Este servicio ofrece la posibilidad de ejecutar tareas o mensajes cuando ocurre una determinada acción dentro de Doctrine.

4.13. Controladores

Los controladores contienen toda la lógica que procesa los datos enviados al servidor y realiza las acciones correspondientes según la ruta a la que se accede. Son los encargados de almacenar datos o recopilarlos para enviar a la plantilla de respuesta.

4.13.1. Controlador de meeting

Este controlador contiene múltiples funciones sobre las reuniones de la aplicación.

Una de las primeras funciones es crear un útil controlador de la tabla meeting de la base de datos, que nos permite actualizar la base de datos vía web. Esta herramienta es sumamente útil para facilitar las actualizaciones de la base de datos a los administradores de la aplicación, como se ha descrito en la sección 4.4. Una función muy importante para el administrador es poder convertir en salas mágicas u observar qué salas son mágicas en la aplicación de una forma rápida. Este controlador nos permite administrar las salas, incluido el campo *magic*, rápidamente. Podemos también convertir una sala pública en privada y viceversa, cambiar el nombre o su descripción.

4.13.2. Controlador personal

En una versión anterior este controlador nos permitía obtener la lista de todas las reuniones disponibles para cada usuario. En la actualidad esta funcionalidad está fuera del alcance de la versión liberada.

4.13.3. Controlador de grabaciones

Este controlador ofrece funcionalidades para controlar la tabla de grabaciones de la base de datos. Al igual que la tabla *meeting*, es posible manipular vía web la tabla de las grabaciones de la aplicación. Es un controlador menos usado que el de las reuniones, pero igualmente útil.

4.13.4. Controlador de sala

Este es uno de los controladores más importantes para el buen funcionamiento de la aplicación. Posee un control de acceso para cada tipo de enlace ofrecido por la aplicación para acceder a las salas. Con este control de acceso definimos mediante la URL proporcionada los privilegios que cada usuario poseerá sobre la reunión a la que es invitado o que ha creado.

En este controlador se comprueban si las huellas (*hashes*) son correctas para permitir el acceso de cada usuario a la reunión. También se diseñan los errores que la aplicación reportará en caso de no acceso a la reunión. La función que realiza este control de acceso crea la URL con la que el usuario accederá al servidor Adobe.

4.13.5. Controlador de seguridad

Este controlador contiene las funciones para la puerta de entrada a la aplicación para cualquier usuario que desee disfrutar de ella. En la primera función se comprueba que el usuario se encuentra en la base de datos y que su contraseña es correcta. Si se produce algún error en el acceso, Symfony ofrece una serie de errores detectados que se muestran en el formulario de acceso. Para esta aplicación hemos filtrado esos errores para que sólo se muestren una serie de mensajes diseñados por el programador, para que el usuario los entienda fácilmente. Estos mensajes se muestran en modo flash de forma que la aplicación se detiene por un instante para ejecutar el mensaje y mostrarlo sobre la propia pantalla donde nos encontramos, en este caso la pantalla de registro. Este tipo flash para mostrar errores se usará frecuentemente a lo largo de la aplicación.

Las funciones para recoger los datos de los formularios de olvido de contraseña también se encuentran en este controlador, que se encargará de llamar al servicio de correo electrónico para enviar un mensaje de “reseteo” de contraseña al usuario en cuestión.

4.13.6. Controlador de administrador

Este controlador contiene la lógica para el menú de los usuarios administradores que permite administrar las bases de datos de reuniones, grabaciones y usuarios. También incluye el código Javascript para mostrar las gráficas estadísticas para controlar el uso de la aplicación.

4.13.7. Controlador de usuario

Este controlador nos permite consultar en la base de datos las diferentes reuniones filtrando según los criterios que precisemos para mostrárselas al usuario en su página principal. Primero se hace una consulta al servidor para que devuelva el identificador del usuario que va a realizar las consultas: `$user = $this->get('security.context')->getToken()->getUser();`

Los diferentes criterios de filtrado son:

- `$meetings_scheduled`: esta variable almacena las reuniones futuras pertenecientes al usuario reservando así la sala para ese instante.

```
\$meetings_scheduled=\$repo->findByStateAndUser(Meeting::STATE_SCHEDULED,\$user);
```

- `$meetings_finished`: esta variable almacena todas las reuniones del usuario que ya han finalizado. Como se puede observar hay dos tipos de estado (cancelado y terminado); esto se debe a que es posible que el usuario cancele la reunión o que la fecha de fin de reunión sea en el pasado.

```
\$meeting_finished=\$repo->findByStatesAndUser(array(Meeting::STATE_CANCELLED,Meeting::STATE_FINISHED),\$user);
```

- `$other_meetings_now`: esta variable almacena las reuniones activas que no pertenecen al usuario y están en la plataforma.

```
\$other_meetings_now=\$repo->findByStateAndNotUser(Meeting::STATE_NOW,\$user);
```

- `$meetings_now`: permite mostrar al usuario todas las reuniones que tiene activas en el momento de la consulta.

```
\$meetings_now=\$repo->findByStatesAndUser(array(Meeting::STATE_NOW,Meeting::STATE_LOCKED),\$user);
```

- `$meeting_now_rank`: permite ordenar las reuniones que pertenecen al usuario según el ranking que ocupen en su página principal, ya que el usuario puede ordenar a su gusto las reuniones que posee para que aparezcan en primer lugar.

```
\$meetings_now_rank=\$repo->findByUserAndStatesOrderByRank(\$user,array(Meeting::STATE_NOW,Meeting::STATE_LOCKED));
```

En este controlador también se realiza el control de acceso a las grabaciones de reuniones de cada usuario. Para ello se comprueba si la grabación ha sido bloqueada por el usuario. Si es así se muestra el correspondiente mensaje de error; en caso contrario se produce la reproducción de la grabación.

Se encuentra también la lógica para la publicación de grabaciones y se genera un enlace a dichas grabaciones a las que podrán acceder todos los usuarios que obtengan dicho enlace que será entregado por el usuario.

Otra función que se encuentra en este controlador es la necesaria para crear una nueva reunión. Para la creación de una reunión debemos comprobar si existen salas disponibles, en caso de ser administrador y precisar crear una sala mágica, también se comprueba si hay salas mágicas disponibles. Estas variables `$numRooms` y `$numRoomsForNonMagic` se pueden modificar en el archivo de configuración según el número de licencias que tengamos disponibles o salas que pongamos a disposición de los usuarios.

Otra función que se encuentra en este controlador es la que nos permite cambiar nuestro sobrenombre en las reuniones del *Adobe Connect®*.

La función con la lógica que usa la consulta Javascript para actualizar el ranking que establece el usuario para las reuniones también se encuentra en este controlador.

- `addUsersAction` es la función que se invoca si precisamos añadir un usuario a una reunión que pertenece a la plataforma “Campus do Mar”. También está disponible un enlace para invitar a usuarios que no pertenecen a “Campus do Mar” a una reunión dada. Existe la posibilidad de cambiar este enlace, ya que es posible que reutilicemos nuestra sala para diferentes grupos de usuarios.
- `cancelAction` es función que se invoca cuando pulsamos el botón de cancelar una reunión. Esta función comprueba si hay grabaciones asociadas a dicha reunión y en caso afirmativo se almacenan en una carpeta especial en el servidor *Adobe Connect®* para que los usuarios tenga acceso a ellas.
- `recordingAction` es la función que mostrará las grabaciones en su correspondiente plantilla. Esta función es invocada por el usuario cuando pulsa el botón `All eMeetings`. Cuando nos encontramos en la ventana modal que abre este botón se ofrece la posibilidad de acceder a reuniones con sus respectivas grabaciones ordenadas por año y mes.
- `historicalAction` contiene la lógica para acceder a la lista de grabaciones.
- `recordingListAction` es la función que hace la petición a la base de datos para devolver la lista de grabaciones para cada reunión, ver figura 4.13.1.



Figura 4.13.1: Lista de grabaciones accesible desde la cuenta del usuario

- `lockedRecordingAction` modifica la base de datos para bloquear las grabaciones y que no puedan ser vistas por otros usuarios. Para alternar entre los diferentes estados de una grabación se pulsarán los botones que pueden verse en la figura 4.13.1.
- `recordingPublicListAction` realiza la petición a la base de datos para mostrar las grabaciones a usuarios que no pertenecen a “Campus do Mar”, ver figura 4.13.2. En esta figura puede verse el título de la reunión a la que pertenecen las grabaciones. El color verde del título de la grabación indica que puede verse, por el contrario, el color rojo indica que la grabación está bloqueada.



Figura 4.13.2: Lista de grabaciones mostrada a usuarios externos

- `lockedAction` contiene la lógica para bloquear las reuniones de forma temporal pulsando el botón *Disabled* para bloquear o *Enabled* para desbloquear, ver figura 4.13.3.



Figura 4.13.3: Plantilla con estado bloqueado y desbloqueado de sala

- `changePasswordAction` con tienen la lógica para cambiar la contraseña de un usuario que también se encuentra, como no podía ser de otro modo, en el controlador de usuario. Esta función sincroniza la contraseña de aplicación con la del usuario en el servidor *Adobe Connect®*.
- `editAction` es una función generada fácilmente con Symfony que reúne los datos necesarios para generar un formulario de edición como el de la figura 4.13.4.

Figura 4.13.4: Formulario de edición de datos de una reunión en modo administrador

- `updateAction` cuando el formulario generado por `editAction` se envía de nuevo al servidor esta función es la encargada de actuar sobre la base de datos para cambiar los datos. En esta función se comprueba que el usuario que solicita cambiar datos es el propietario de la reunión, en caso afirmativo se procede a guardar los datos.

- `userFormAction` contiene la lógica para crear una nueva reunión directamente en la base de datos para un usuario concreto. El *framework* Symfony nos ofrece la posibilidad de crear un formulario rápidamente a partir de la tabla de la base de datos definida para las reuniones. Esta función tiene un formulario asociado como el que se muestra en la figura 4.13.5.

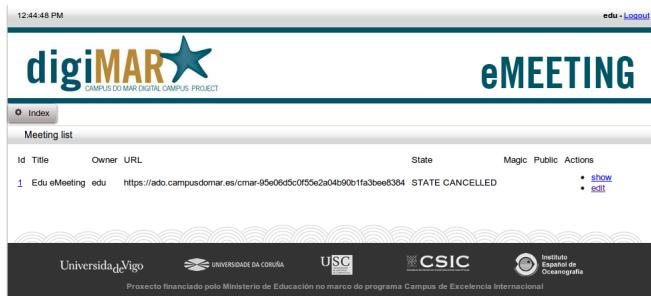


Figura 4.13.5: Formulario para crear nueva reunión modo administrador

- `listAction` es la función que contiene la lógica para la acción Javascript de autocompletado que se implementa en la ventana modal para agregar usuarios de “Campus do Mar” a las reuniones. De esta forma si introducimos en la caja de agregado tres letras, la función nos devolverá una lista de usuarios que contengan estas tres letras en el orden establecido, tanto en el nombre y apellidos como en el correo electrónico, ver figura 4.13.6.

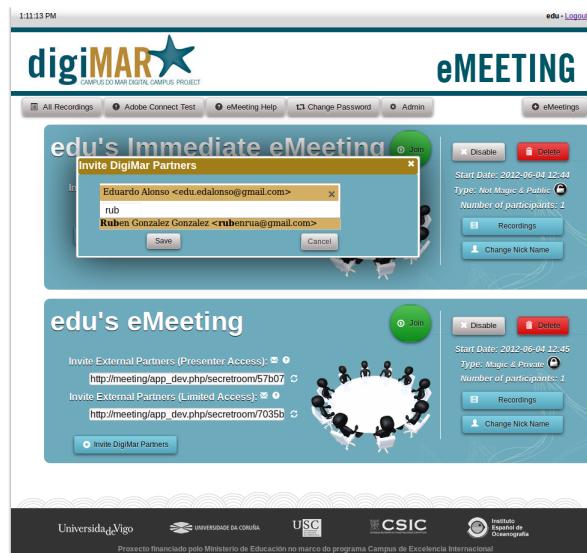


Figura 4.13.6: Función autocompletado para agregar usuarios

- **adminAction** ofrece un menú exclusivo para los usuarios administradores de la aplicación como puede verse en la figura 4.13.7. Este usuario tendrá acceso a los formularios de creación y edición de reuniones directamente sobre la base de datos, también para la creación de usuarios y su correspondiente edición. Además se han introducido dos gráficas estadísticas que nos ofrecen la ocupación máxima de salas cada día y el número total de usuarios simultáneos en la aplicación por cada reunión.

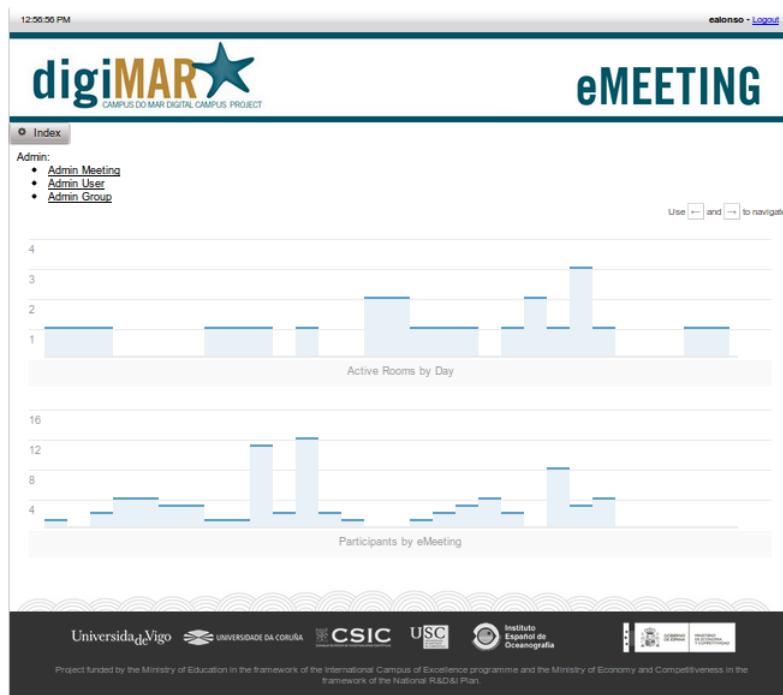


Figura 4.13.7: Menú de administración

4.14. Estilo de la aplicación con CSS y HTML

En el archivo *cmar.css* y *meeting.css* tenemos nuestras hojas de estilo para la aplicación “meeting”, como es obvio en el archivo *cmar.css* se encuentra el estilo común a toda la plataforma “Campus do Mar” y incluyendo todas las aplicaciones y webs que forman parte de la macroaplicación descrita en la sección 1.1 y en *meeting.css* tenemos el estilo específico y necesario para cada uno de los conceptos y plantillas relatados en el capítulo 4.

Para el diseño del estilo y de las plantillas de la aplicación se han utilizado algunos de los últimos avances de HTML y CSS para simplificar y sacar el mayor partido a las funciones desarrolladas. El uso de formularios y listas HTML son fundamentales en cualquier aplicación web a lo cual añadimos el estilo en cascada para proporcionar un interfaz agradable, intuitivo y elegante.

4.15. Funciones Javascript

Para las funciones javascript hemos usado la librería jQuery en su versión 1.7.1, también hemos usado la librería específica para interfaces jquery-ui-1.8.18, jquery-kamicgraphs, jquery.tokeninputs(esta es para la entrada anónima), bootstrap (la parte de los menús desplegables, librería muy popular debido a su vinculación con Twitter) y la librería tipTip. Pasamos a describir a continuación una breve descripción de cada una de estas librerías, cuya principal función han sido la simplificación de los desarrollos y nos han dado la posibilidad de implementar un interfaz avanzado de forma sencilla.

Todas las funciones que hemos implementado a partir de estas librerías y las propias librerías se encuentran en la carpeta *src/Cmar/MeetingBundle/Resources/Public/js*.

Estos ficheros se han renombrado con una técnica típica de numeración para que se carguen en el explorador en el orden adecuado, ya que unos dependen de otros y la carga incorrecta generará incongruencias de código y el mal funcionamiento de la aplicación.

4.15.1. Funciones propias

Las funciones propias reciben en su mayoría el nombre de la función que desempeñan para la aplicación. Describimos a continuación los archivos javascript y las funciones que contienen.

- EL fichero “js.js” contiene la función updateClock que como su nombre indica actualiza el reloj junto al nombre de usuario en la parte superior derecha de la aplicación. Esta función recoge la hora, minutos y segundos del sistema en el que se encuentre alojado y actualiza el reloj visualizado por el usuario con una precisión de segundos. Realiza un parseado para obtener siempre números enteros en cada valor que muestra el reloj y también obtiene la franja horaria para mostrar “AM” o “PM” o si lo deseamos hora en formato en 24 horas.
- El fichero “tabla.js” despliega los elementos de una lista utilizando el evento jQuery fadeIn, el cuál embellece la acción ya que la lista de elementos se despliegan de forma suave y constante.
- EL fichero “addbox.js” se utilizó en un primera versión del proyecto para agregar elementos de una lista que aparecía a la izquierda del usuario en su homóloga en la parte izquierda. De esta manera se invita al resto de usuarios a participar de la videoconferencia que ha creado. Con el uso de la aplicación se comprobó que la lista de usuarios era demasiado grande y debían introducirse otro tipo de técnicas para tal invitación. En esta acción también se incluye la posibilidad de invitar a todos los usuarios del sistema. Este tipo de mecanismos es más utilizado cuando el número de usuarios a invitar se aplica a una lista de contactos seleccionados por el usuario y no al total de usuarios de la aplicación como es el caso.
- A raíz de la experiencia descrita en el fichero anterior se desarrolla el fichero “autocomplete.js”. Este fichero realiza una petición AJAX sobre la base de datos a partir de la

introducción de tres caracteres en una caja de texto. De esta manera se despliega una lista bajo la caja de texto con la coincidencia de los caracteres introducidos. Dicha coincidencia puede darse tanto en el nombre completo de un usuario como en su correo electrónico. Gracias a jQuery y su función *autocomplete* la lógica de esta función sólo necesita que incluyamos los parámetros de interés tales como en qué lugar se encuentra nuestra consulta sobre la base de datos y de qué forma mostraremos al usuario la información retornada por dicha consulta. En nuestro código aparecen dos lugares en los cuales podremos añadir usuarios a nuestra videoconferencia, de ahí que la lógica aparezca duplicada para el identificador *users* y *users1*.

- EL fichero “modal_dialog.js” nos ofrece la lógica para desplegar las ventanas modales que se despliegan para realizar acciones sobre las videoconferencias de la aplicación y proporcionan funcionalidades a mayores implementadas para dar información al usuario, tales como el menú que ofrece el servidor *Adobe Connect®* para informarnos si tenemos conectividad a nivel de red con el servidor o si nuestra conexión es suficiente para mantener una videoconferencia de calidad. En esta función definimos los identificadores de los botones que accionarán los menús, el tipo de efecto que cerrará la ventana, el tamaño de la ventana y a mayores para algunas ventanas especiales hemos añadido el parámetro *data-ajax*. Cuando le pasamos este parámetro a la ventana modal irá acompañado del parámetro *data-url* que nos indica el lugar en el que se encuentra la lógica de que se ejecutará tras nuestra petición AJAX. Del mismo modo y gracias a las facilidades ofrecidas por Symfony en la ventana se desplegará el template asociado a la URL que indicamos en la llamada a la ventana modal con los datos de la función que se ha ejecutado. La posición de la ventana y la clase CSS asociada a la ventana también se describen en este fichero. Los identificadores y clases intentan describir a qué parte del código pertenecen las ventanas modales para facilitar la fácil lectura del código y que la incorporación de un nuevo miembro al proyecto se lleve a cabo en el menor tiempo posible.
- El fichero “desplegable.js” ofrece la lógica para menús desplegables, una funcionalidad muy similar a la del fichero “menu.js” pero con sutiles cambios en la estética. En este fichero también se incluye la lógica que reducirá el número de letras a mostrar del título para que un título muy largo no rompa la estética del interfaz, aunque el título se mostrará completamente si el ratón se sitúa sobre él. Otra función desarrollada en este fichero es el minimizado de la entidad de videoconferencia, ver figura 4.6.5, que muestra la aplicación, de forma que si pulsamos dos veces sobre el título la entidad se minimiza y viceversa.
- El fichero “filter_meeting.js” realiza la función de filtrado de salas de videoconferencia que ofrece el botón del menú que por defecto tiene el texto *All Meetings*. Esta función recibe el texto del desplegable que coincide con los diferentes estados que puede tener una sala de nuestra aplicación: Pública, privada, mágica (Sólo para usuarios administradores) o sala a la que he sido invitado. Esta función muestra u oculta las salas con el estado que corresponda.
- El fichero “anonymous.js” ofrece toda la lógica del menú de entrada anónima. Este

menú ofrece la posibilidad de acceder a las salas como anónimo o acceder con nuestro usuario. Se implementa aquí la lógica de error cuando no coinciden las dos contraseñas introducidas como confirmación liberando así al servidor de peticiones superfluas sobre la base de datos.

- En algunas de las plantillas del código se encuentra código Javascript incrustado referente a dicha plantilla para ejecutar acciones como minimizar el área efectiva de las reuniones en la lista de usuario como se mostró en la figura 4.6.5. También está incrustado el código que permite ordenar las reuniones para actualizar la variable *rank* de la tabla *meeting*. El *plugin jQuery UI* permite realizar la animación de movimiento de la reunión mediante el ratón y el código incrustado actualiza la variable para que se almacene la preferencia de orden en la base de datos. Las llamadas a la librería *Tip Tip* y el propio mensaje que muestra de ayuda en los iconos comentados en la subsección 4.6.4 se encuentran también en secciones Javascript en los templates correspondientes.

Capítulo 5

Conclusiones y líneas Futuras

5.1. Conclusiones

A raíz de los datos obtenidos sobre la utilización de la aplicación por parte de los usuarios de “Campus do Mar” podemos afirmar que ha sido un rotundo éxito de acogida. Por su fácil funcionamiento y por la rapidez con la que los usuarios pueden crear sus salas de videoconferencia y gestionarlas es sin duda un producto ventajoso frente a su más directa competencia. El uso de un framework aumenta altamente la productividad del diseñador y ofrece una organización muy completa y ágil, ideal para elaborar una excelente documentación.

Sorprendentemente para el programador se ha profundizado también en el complejo conocimiento de las necesidades y singularidades del ser humano, a raíz del trabajo realizado en el despliegue de la aplicación. En la experiencia adquirida después del uso de la aplicación cabe destacar, lo diferente de la psique del ser humano aún perteneciendo a perfiles afines y como las relaciones mejoran si se tiene un buena herramienta de comunicación.

Los datos recopilados en este último año de utilización se detallan a continuación. Existen más de ochenta usuarios registrados, contando entre ellos con personal de “Campus do Mar”, de la universidad de Vigo y de otros organismos públicos. Más de doscientas grabaciones han sido realizadas por los usuarios de la aplicación en las ciento treinta y dos reuniones creadas, contando entre ellas con reuniones de la plataforma de código abierto “*Opencast*” que ayuda al desarrollo de la plataforma de grabación automatizada de clases “*Mattehorn*” para “DigiMar”. Las gráficas de la sección de administrador como la mostrada en la figura 4.13.7, nos ofrecen una visión real de la utilización de la aplicación, en ellas podemos observar una media de tres reuniones diarias sobre noventa reuniones al mes.

Otro dato a tener en cuenta al observar las gráficas del menú de administración es que un día se ha llegado al número máximo de salas disponibles por lo que es posible que se amplíe el número de licencias para disminuir la posibilidad de denegación de servicio.

5.2. Líneas Futuras

A raíz de la utilización de la aplicación han surgido una serie de mejoras que se comentarán en esta sección.

Una de las líneas más importantes para este sistema desde el principio fue utilizar un *software* de soporte para videoconferencia no propietario como podría ser Asterix o FreeSswitch [13]. De esta manera el coste de cada sala sería sólo debido a la conexión y equipos necesarios para albergar la aplicación.

La siguiente línea derivada de la utilización de la aplicación y muy conveniente para el escenario que hemos descrito para “Campus do Mar” es la posibilidad de reservar salas para lo cual debe utilizarse la variable *concurrent* de la base de datos. Esta variable almacena el número de reuniones que coincidirán en el tiempo, para asegurarnos de que todas las reuniones tendrán disponibilidad de sala de videoconferencia. Otra modificación en la base de datos será añadir un campo que albergue la duración de la reunión.

Como ya se ha explicado *Symfony* nos ofrece las entidades para modificar nuestra base de datos y aportar una serie de consultas simples sobre la base de datos y sobre las nuevas columnas introducidas, lo cual es altamente efectivo a la hora de modificar la base de datos.

Para implementar un interfaz que permita al usuario reservar salas en el futuro es necesario un controlador de reuniones futuras que implemente un calendario y un formulario que recoja los datos para introducir en la base de datos. La lógica de este controlador deberá comprobar sobre la variable *concurrent* el número de salas simultáneas que coinciden con la nueva sala a crear para devolver al usuario un error en caso de no disponer de la sala deseada. Esta variable será actualizada cada vez que se cree o se borre una sala en el intervalo de tiempo que ocupe la reunión para conocer en todo momento el número de salas concurrentes en ese intervalo y mostrar los mensajes oportunos al usuario.

Bibliografía

- [1] Zaninotto F., Potencier F. Symfony 1.2, la guía definitiva
- [2] Potencier F. Jobeet
- [3] ICB Editores, Programación De Páginas Web Dinámicas Con Apache, Base De Datos Mysql Y Php 10
- [4] Domine Php Y Mysql. 2^a Edición, López Quijado, José 81
- [5] <http://www.symfony-project.org>
- [6] <http://www.symfony.com/> 9, 10, 13, 14, 25
- [7] Dubois P., La Biblia de MySQL
- [8] <http://www.yaml.org/> 14
- [9] <http://php.net> 16, 26
- [10] <http://www.sun.com/suntrademarks/> 18
- [11] <http://jquery.org/> 19
- [12] <http://jqueryui.com/> 19, 54
- [13] <http://www.freeswitch.org/> 94
- [14] <http://twig.sensiolabs.org/doc/intro.html> 13
- [15] Manual de Symfony 2 12
- [16] <http://www.balsamiq.com/products/mockups> 53