Títol:	*Creación de un sitio web para una emisora de radio online usando un gestor de contenidos*
Volum:	*I*
Alumne:	*José Rodríguez Jiménez*
Director/Ponent:	*Antonio Cañabate Carmona*
Departament:	*Organització d'Empreses*
Data:	*7 de Novembre de 2008*

Presentación y agradecimientos

Buscando entre los proyectos propuestos por los departamentos de la FIB como posibles PFCs para estudiantes de Informática, y después de leer una larga lista de proyectos sobre algoritmos, análisis de tráfico de paquetes de datos, ejecución de procesos concurrentes, generación de modelos conceptuales, y otros temas que había tocado, en mayor o menor medida, durante la carrera, me llamó la atención uno bajo el título *Construcción del site de una emisora de radio por Internet.*

Mirando la descripción pude observar que el proyecto trataba de remodelar una página web, en concreto un emisora de radio por Internet. En el anuncio se comentaba que era algo nuevo donde destacaban palabras como CMS, Drupal, auto aprendizaje. Investigando un poco descubrí que debería aprender sobre un gestor de contenidos llamado Drupal, premiado con varios premios y para mí eso era todo un reto, al final decidí concertar una entrevista con el director del proyecto para poder llevarlo a cabo.

Poco más tarde sabría que la emisora en cuestión era la de barcelonajazzradio.com, que ya había descubierto hacía algún tiempo, y conocería a Josep Mestres, la persona que estaba detrás de todo esto y que se había propuesto difundir el jazz que se hace aquí en Barcelona utilizando la potencia de la red de redes.

En el equipo del proyecto éramos tres personas, Javi, Iván y yo mismo, siempre coordinados por el director del proyecto, Toni Cañabate, y seguidos de cerca por Josep Mestres, que se ha mostrado siempre receptivo a nuevas ideas e ilusionado con los avances.

A todos ellos quiero agradecer el ambiente de trabajo creado. Ha habido momentos de tensión, pero también facilidades, ayuda y colaboración entre todos. De una cosa no hay duda, ha sido toda una experiencia.

Por si fuera poco, he podido aprender un nuevo estilo de música que sólo conocía de "lejos" y que al adentrarme en el mundo del jazz pude descubrir que es un mundo apasionante, nada monótono (como yo tenía la sensación antes de hacer este proyecto) y que siempre está en continua evolución.

Después de mucho tiempo escuchando barcelonajazzradio.com descubrí que había mucho talento en nuestra ciudad, gente como Albert Bover, Abel Boquera, Oscar Peñas y el mítico Tete Montoliu. Me siento orgulloso de haber podido conocer la escena de jazz barcelonés.

Desgraciadamente a veces la vida te da algún susto que otro y no he podido dedicar todo el tiempo que he me hubiera gustado (más tiempo del que puramente he dedicado a este proyecto) ya que por problemas de salud he estado fuera del proyecto durante un breve espacio de tiempo. Gracias a todos los componentes del equipo por el esfuerzo extra de coordinación a causa de este problema.

Esto no ha impedido que al final, a día de hoy 4 de Noviembre de 2008 esté finalizando ya mi memoria y de lo que estoy tremendamente orgulloso.

Finalmente, agradezco a mis padres Romualdo y Josefa, mis hermanas Susana y Marta, mi pareja Miriam y mis amigos (incluyendo en especial a todos los componentes de este proyecto) por todo el apoyo durante este largo proceso, que ha sido duro para todos pero que con coraje hemos superado. Gracias de todo corazón.

Tabla de contenidos

1. Introducción

Hasta hace relativamente poco tiempo la música era algo que uno interpretaba o improvisaba en un lugar concreto. Cualquier persona que quisiera disfrutar de esa música tenía que estar presente en ese lugar en el momento exacto en el que eso sucedía, porque no existía otra forma de escucharla. Esto ha sido así desde el primer grito prehistórico y hasta finales del s.XIX, época de grandes inventos.

Dos hechos tan simples hoy en día como son la transmisión del sonido a través de ondas electromagnéticas o la posibilidad de registrarlo en un soporte para poderlo reproducir en cualquier momento fueron logros sorprendentes que revolucionarían, entre otros, el mundo de la música. Con la aparición de las primeras emisoras de radio y las grabaciones comerciales sobre vinilo a principios del siglo XX dejó de ser imprescindible asistir a conciertos y se hizo mucho más fácil para los músicos promocionarse y ser escuchados. Surgió así la industria musical, cuyo modelo de negocio consistía en vender esas grabaciones a todo aquél que quisiera escucharlas.

Y así fue durante muchos años. Cambiaron los músicos, cambiaron los soportes, mejoró la calidad, pero básicamente la música se seguía escuchando en directo, por la radio o bien comprando una grabación. La televisión proporcionaría otra forma de acceder a la música: musicales, grabaciones de conciertos y videoclips.

Con las cintas magnéticas se le dio la posibilidad al hombre de a pie de copiar la música proveniente de la radio o de las propias cintas en otras cintas vacías, llamadas *vírgenes*, dando lugar a lo que rápidamente se conocería como *piratería*.

Con la llegada de los CDs, los ordenadores personales y los formatos digitales, la música se convirtió en una cadena de unos y ceros, y creció dramáticamente la facilidad de transmisión de un soporte a otro y la capacidad de manipulación de esa música.

Y cuando Internet hizo acto de presencia las posibilidades aumentaron de forma exponencial. De entrada, surgieron las tiendas online, en las que se pusieron a la venta todo tipo de artículos, entre ellos los CDs de música.

Pero sería absurdo tener música en formato binario y limitarse a enviarla por correo postal. Las implicaciones de la digitalización del sonido eran mucho más interesantes: una persona en Estados Unidos podía copiar de un CD a su ordenador la música que acababa de comprar y enviársela en un tiempo relativamente corto a otra persona situada en Europa, que luego podría grabarla en un CD virgen para escucharla en un reproductor.

Como los archivos de música ocupaban demasiado para las velocidades de conexión de ese momento y para el espacio de los discos duros y de los CDs, se buscaron fórmulas para *comprimir* esos archivos, dando lugar al formato estrella, el MP3. En pocos minutos el internauta europeo recibiría mucha más música que antes en unas horas.

Paralelamente a todo esto empezaron a proliferar unos programas que se basaban en redes conocidas como *P2P* (*Peer to Peer* o *Punto a Punto*). Con ellas, grandes grupos de

usuarios podían enviarse archivos unos a otros con una gran facilidad, compartiendo enormes cantidades de información de forma eficiente.

El uso principal que se dio a estos programas, ya desde sus inicios, fue el de compartir música. Si bien con las cintas magnéticas la piratería nunca supuso un auténtico peligro para la industria musical ni para la propiedad intelectual, la llegada de Internet y de las redes P2P dejó muy claro que no sólo los sistemas de protección anticopia existentes sino también la filosofía de la propiedad intelectual habían quedado obsoletos. Las tiendas de música online pronto renovarían su forma de vender para apuntarse a la creciente fiebre del MP3.

Pero el P2P y los MP3 no fueron las únicas formas de acceso a la música que aparecieron en ese momento. Para poder escuchar la música de un archivo MP3 que aún no poseía, el usuario tenía que descargar previamente ese archivo. Cuanto más grande fuera el archivo, más tiempo tardaba el usuario en poder escucharlo. Para solucionar ese problema apareció una técnica de envío de datos llamada *streaming*. El streaming consiste en enviar, de forma continuada y de principio a fin, un archivo de un servidor a un cliente. A medida que el cliente recibe partes nuevas de un archivo de audio, puede irlas escuchando sin necesidad de esperar a que termine de descargarse el archivo entero.

El parecido de esta técnica con el funcionamiento de una radio es obvio, por lo que no tardarían en aparecer empresas que ofrecían envío de música a los oyentes mediante streaming, dando lugar a las primeras radios online.

Web 2.0

Llegados a este punto, podría parecer que las cosas han cambiado mucho para los oyentes desde las primeras grabaciones sonoras. Es cierto que el acceso a la música es ahora mucho más fácil, pero el papel del oyente se ha mantenido siempre igual: reproducir la música y escucharla.

Figura 1.1: *Tag cloud* de Markus Angermeier sobre la Web 2.0

Y es aproximadamente en este momento cuando se empieza a forjar en Internet la idea de la *Web 2.0*. Con esta denominación algo pretenciosa se pretenden definir unas tendencias que han ido naciendo poco a poco de algunos sitios web innovadores y con ansias de crecer. Por un lado, hace referencia a la creación de sitios web mediante el uso de

estándares como XHTML y CSS y tecnologías como Ajax, así como a unos diseños característicos (Figura 1.1); pero, sobre todo, la expresión Web 2.0 alude al comportamiento comunitario que ha tomado Internet en los últimos tiempos: blogs, comunidades online, wikis, redes sociales, juegos virtuales... La Tabla 1.1 de Tim O'Reilly, expresa claramente el cambio.

Web 1.0	Web 2.0
DoubleClick	» Google AdSense
Ofoto	» Flickr
Akamai	» BitTorrent
mp3.com	» Napster
Britannica Online	» Wikipedia
Personal websites	» Blogging
evite	» Upcoming.org and EVDB
domain name speculation	» Search engine optimization
page views	» cost per click
Screen scraping	» web services
publishing	» Participation
content management systems	» Wikis
directories (taxonomy)	» tagging ("folksonomy")
stickiness	» syndication

Tabla 1.1: Web 1.0 vs. 2.0

Internet hasta ahora se había comportado de una forma muy unidireccional. Una persona escribía algo en una web y las otras lo leían. El hipertexto le ha dado siempre a Internet un carácter muy expansivo, y la aparición de lenguajes capaces de generar HTML lo convertía en algo bastante dinámico, pero sólo a la hora de mostrar contenidos. Con los blogs se ha conseguido que los visitantes a una página web interesados en un determinado contenido puedan comentar, votar, reenviar, conectar, *menear*, difundir, criticar, marcar como spam, opinar, poner en favoritos, citar, etiquetar, sindicar... Y además de muchas maneras diferentes. Los blogs se comunican unos con otros, se unen para llevar a cabo proyectos mayores y, en definitiva, generan contenidos y relaciones entre personas de una forma mucho más viva. Los que antes eran sólo consumidores ahora se convierten también en generadores.

Las redes sociales van incluso más lejos. Cada individuo crea su propio espacio personal, con su círculo de amigos con intereses comunes, los cuales tienen otros amigos con intereses similares que se acaban conociendo entre ellos. Se ejecutan aplicaciones, se reciben avisos, se conoce lo que están haciendo los demás en un determinado momento, se comparten enlaces, vídeos, música...

Hay redes sociales de tipos muy diversos, unas más abiertas, otras más centradas en algo específico: Friendster, Facebook, MySpace, LinkedIn o incluso la Wikipedia o Youtube. La red social MySpace tiene una subcategoría llamada MySpace Music. Casi cualquier músico tiene una cuenta en este espacio, desde el cual se convierte en oyente de otros músicos, se comunica con ellos, etc.

Pero volviendo al tema de las radios, su visión del 2.0 consiste en lo siguiente: webs en las que un usuario puede escuchar la música que quiere, organizándose sus propias listas de reproducción y creando su propia radio online. A partir de los temas seleccionados, el sistema compara los gustos del usuario con los de otros usuarios que escuchan música similar y le hace propuestas. De esta forma uno puede conocer músicos de los que no había oído hablar y cuyos trabajos le resultarán probablemente atractivos. Otras opciones son permitir a los demás usuarios ver las listas de reproducción propias y escuchar la música que el otro ha elegido para sí mismo, es decir, sería como poder escuchar las radios

de los demás. Webs que llevan a cabo este tipo de funcionalidades son Last.fm, Deezer, Maestro.fm...

En definitiva, nos encontramos justo en el centro de un panorama en plena ebullición, lleno de recursos, con infinitas posibilidades y en el que todo es gratuito. Musicalmente, se cuestiona el papel de los creadores, de los oyentes, de los intermediarios (programadores de radio, entidades de gestión de derechos, empresas de marketing...). Se intercambian los roles, dando la oportunidad a todos de hacer el papel de los demás. Existe más comunicación y más información que nunca. Y si hay un estilo de música en el que se aprecie de verdad este exceso de información, ése es el jazz.

Jazz

El jazz es la música del siglo XX. Se puede decir que es la evolución lógica a la música que se hacía anteriormente pero con un toque inesperado. Las características que lo diferencian de la música previa son un uso mucho más exigente de ritmo y armonía, una libertad total para la experimentación y, sobretodo, que el momento de la creación coincide con el de la interpretación, es decir, que se improvisa mientras se toca, lo que hace del jazz una experiencia cambiante y diferente en cada ocasión.

Eso no lo convierte necesariamente en una música elitista y no apta para todos los públicos, pero es un hecho que cuantos más conocimientos de música, y concretamente de jazz, tenga el oyente, mucho más comprenderá y disfrutará lo que escuche. Por esa razón, en líneas generales, la mayoría de aficionados al jazz son músicos o bien tienen un interés por la música diferente al que puedan tener los aficionados a otros estilos.

El aficionado al jazz es, pues, alguien que presta atención a lo que escucha, que rebobina frecuentemente, que transcribe y estudia los solos de sus músicos favoritos y que luego intenta tocarlos, buscando siempre aprender algo nuevo. Ese interés tan personalizado por la música de un artista da a entender la importancia de los individuos en una banda o un álbum de jazz. Cada músico tiene nombre y apellidos, toca un instrumento concreto, suena de una manera específica y tiene una forma de improvisar que lo diferencia de otros músicos. El sonido, el fraseo o la forma de articular las notas pueden servir para identificar a un músico sin necesidad de saber de antemano de quién se trata. De hecho, ése es un ejercicio que se realiza habitualmente para entrenar el oído.

Una característica que ayuda a un aficionado a decidirse cuando va a adquirir un álbum es la formación de músicos e instrumentos. Los músicos de jazz suelen agruparse entre ellos para tocar en directo o grabar discos, sin ser necesario que toquen juntos habitualmente. Hay incluso músicos que están muy especializados en acompañar a otros músicos extranjeros que están de gira por su país. La lista de músicos, así como la formación de instrumentos, puede ser un gran atractivo a la hora de comprar un álbum o de decidirse a ir a un concierto. Por ejemplo, no es lo mismo Keith Jarrett tocando a piano solo que tocando a trío con Gary Peacock y Jack DeJohnette, son dos experiencias absolutamente diferentes.

Un hecho curioso que sorprende la primera vez que uno lo oye mencionar es que un grupo de músicos de jazz que no se conozcan de nada se puede encontrar en un lugar extraño, desenfundar los instrumentos, ponerse a tocar y crear música de gran calidad. Estos encuentros se conocen con el nombre de *jam sessions*. Existen dos buenas razones que hacen posible el entendimiento inmediato entre músicos desconocidos. Primero, y como es lógico, la improvisación se hace en base a unas ciertas reglas (también existe la versión sin reglas, llamada *free jazz*). Esas normas consisten en improvisar sobre unas armonías determinadas, una secuencia de acordes que todos los músicos del grupo deben conocer. Cómo se improvise sobre esos acordes definirá una tendencia de jazz o el estilo propio del músico.

El otro motivo por el que los músicos pueden tocar sin conocerse es que saben qué es lo que están tocando, es decir, conocen el tema en cuestión. Existe un larguísimo repertorio de composiciones que suelen tocar los músicos de jazz. Son los llamados *standards*. Los

standards son composiciones que han ido surgiendo a lo largo de la historia del jazz y que, por una u otra razón se han convertido en imprescindibles para el colectivo de músicos. La mayoría de ellos proviene de uno de estos dos orígenes:

- Son compuestos por músicos conocidos como *compositores de standards*. Recibieron este nombre precisamente por aportar muchos temas a la colección, a menudo escritos para musicales de Broadway, películas, etc. Entre esos músicos están los hermanos George e Ira Gershwin, Cole Porter, Jerome Kern, Dorothy Fields, Oscar Hammerstein, Jimmy McHughs, Johnny Mercer, Hoagy Charmichael...
- Son compuestos por músicos de jazz. Además de improvisar en directo, muchos músicos de jazz tienen la faceta de compositores, y han dejado grandes temas para la posteridad. Son músicos como Duke Ellington y Billy Strayhorn, Miles Davis, Thelonious Monk, Bill Evans, Herbie Hancock, Horace Silver, Wayne Shorter, Chick Corea...

Así como los primeros tienen un sonido más clásico, los otros suelen recordar más a la época en la que fueron escritos. La recopilación de estos temas se hace en unos libros que reciben el nombre de *Real Book* y, según las ediciones tienen más o menos composiciones, pero suelen contener de decenas a cientos, casi siempre incluyendo clásicos de Bossa Nova y de jazz latino.

No todos los músicos de jazz tienen capacidad ni interés para recordarlos todos, pero siempre recuerdan una buena parte de ellos. En cualquier caso, un subconjunto de esos temas es conocido por prácticamente todos los músicos, lo que les permite ponerse a tocar uno de ellos en cualquier momento. Y si no lo conocen, con unas pocas indicaciones y una lista de acordes en seguida puede estar improvisando.

Todo esto da una idea de las características del aficionado al jazz: tiene conocimientos musicales y está interesado en los músicos, los instrumentos y los temas que tocan. Eso hace que la información que se puede encontrar, por ejemplo, en las webs de venta de discos generalizadas sea normalmente insuficiente para lo que él busca. Muchas veces no está presente la lista de temas, aunque cada vez es menos habitual, pero lo que casi nunca se puede encontrar es la formación de músicos. Para saber quién toca en el álbum hay que leer comentarios de los compradores o, en la mayoría de los casos, ponerse a buscar reseñas del disco en webs de jazz especializadas.

Ocurre lo mismo con las radios online. Muestran el nombre del tema, el álbum y el artista o banda a cuyo nombre está el disco. Volviendo al ejemplo de antes, sí sólo se indica *Keith Jarrett* es imposible saber cómo sonará el disco. Esta carencia y descentralización de los contenidos dificulta mucho la compra y la escucha de jazz. Es un modelo suficiente para la web 1.0, pero en los tiempos que corren hace falta ir más lejos.

La primera mejora lógica sería añadir esa información que falta. Hacer que un oyente de una radio online pueda acceder a datos relevantes del álbum, del artista y del tema que están sonando. Que pueda saber quién toca en un álbum y qué otros temas lo componen. Que pueda leer una biografía de cada uno de esos artistas, ver sus álbumes, acceder a sus webs personales, blogs o perfiles de redes sociales, saber dónde tocará dentro de poco, etc. El sitio web de una radio de jazz por Internet enriquecida con esas características podría ubicarse dentro de la "Web 1.5", en caso de que existiese tal cosa. Esa playlist hiperenlazada y esa relación multimedia con los *metadatos* de la música que se escucha serían avances importantes en la comunicación entre músicos y aficionados, entre los que existe en la vida real un solapamiento muy grande.

Pero se puede llegar más lejos. La versión 2.0 de esa radio pasaría por convertirse ella misma en una red social, un punto de encuentro entre consumidores y creadores de jazz, que a menudo son las mismas personas. Un lugar en el que compartir grabaciones propias,

composiciones, transcripciones y experiencias. Donde sea la comunidad la que complemente la información de esa radio: anécdotas de la grabación de un disco, reseñas de conciertos, críticas... Un espacio en el que un músico pueda grabar su versión de un tema y enviársela a su propio compositor para conocer su opinión. En definitiva, una gran jam session en la que puede ocurrir de todo.

Tratándose de una música minoritaria, los aficionados al jazz de una región concreta son escasos comparados con los oyentes de otros estilos musicales, pero si se mira al mundo entero, la comunidad de aficionados puede ser muy grande, está muy dispersa y, dadas sus características, está deseando contar con esa interacción mayor.

1.1. Origen del proyecto

Así pues, las características que se han ido mencionando para el contexto en el que se enmarca este proyecto se concretan en la existencia de una comunidad de aficionados y/o músicos, profesionales o amateurs que cumple los siguientes puntos:

1. Está dispersa geográficamente por todo el mundo.
2. Es demandante de una experiencia intelectualmente más rica que la simple audición, que satisfaga su interés por un mayor conocimiento sobre la propia música, sus autores e intérpretes y sus otras obras. Por tanto, consumidora de información multimedia hipervinculada entre ella y con la propia experiencia auditiva.
3. Y, por último, con intereses comunes y proclive a establecer vínculos que les permitan compartir su pasión por el jazz, de forma en absoluto pasiva.

Estos tres factores definen una situación bastante adecuada para un proyecto de una radio en Internet que haga llegar la experiencia musical a los aficionados de todo el mundo (1); complementada con un sitio web que ofrece variados, selectos y frecuentemente actualizados contenidos, estrechamente vinculados con la programación de la radio (2) y ofreciendo servicios y aplicaciones a sus usuarios que les proporcionen diferente fórmulas para descubrirse, interaccionar y compartir experiencias, conocimientos, opiniones, gustos musicales, música e incluso algunos pasos del propio proceso creativo (3).

Esta situación, en conjunción con una dinámica escena artística en jazz contemporáneo en Barcelona, dibuja una oportunidad donde hay que buscar el origen de **barcelonajazzradio.com.**

En febrero de 2007 se dio a conocer una iniciativa en Barcelona destinada a los aficionados al Jazz. Josep Mestres acababa de poner en marcha una radio online, abierta a todo el mundo, totalmente gratuita, sin publicidad y 100% orientada al jazz contemporáneo, y especialmente atenta al jazz de nuestro país.

Había buenas razones para querer lanzar un proyecto como éste. El panorama actual del jazz en Barcelona es rico y de calidad. Hay bastantes músicos, muchos de ellos jóvenes, con un alto nivel musical y con una proyección internacional importante: estudian en Berklee, graban en Nueva York, tocan con músicos que vienen de otros países... Pero por alguna razón son injustamente ignorados en la radio y eso los priva de una expansión mayor, dejándolos encerrados en el círculo de Barcelona y siendo desconocidos para la gran comunidad de aficionados del mundo.

La gran ventaja de una radio online es que necesita muchos menos recursos de inversión que una radio tradicional. No hace falta pagar licencias al Estado ni pedir ningún tipo de permiso, lo único que hay que hacer es pagar derechos de autor. Eso permite centrarse en el verdadero problema: elegir música de buena calidad y ofrecer una buena programación.

Conjuntamente a la creación de la radio, y para promocionarla, se presentó un ciclo de conciertos con la colaboración de MASiMAS, la empresa que lleva buena parte de las salas

y clubs de jazz de Barcelona. Un ciclo de jazz contemporáneo en uno de los espacios más bien acondicionados para este tipo de música: la Cova del Drac.

1.1.1. La radio y el sitio web

La radio, que no ha cesado su programación desde entonces, emite principalmente grabaciones discográficas, conciertos grabados expresamente y autoproducciones no editadas (aportadas directamente por músicos). No hay espacios presentados programados, sino que es una radio con un formato radio-fórmula. Sin interrupciones y funcionando las 24 horas del día.

La intención era crear una herramienta sencilla para descubrir y estar al día sobre el jazz que se hace en nuestro país, pero sin ser exclusivamente de músicos de aquí y sin proponer un sonido o modelo de jazz localizado geográficamente. Sólo se quería asegurar una presencia considerable de los músicos que viven o trabajan en Barcelona, siempre que alcanzaran una calidad o madurez relevante.

De este modo, se había previsto que un 50% de los trabajos programados pertenecieran a músicos que trabajasen con cierta frecuencia en Barcelona, incluyendo a los músicos de fuera que tuvieran una estrecha relación con la ciudad. El resto de la programación sería principalmente jazz contemporáneo, aunque con libertad para añadir algo de jazz no necesariamente actual.

Sobre el estilo de música emitido, se querían evitar por todos los medios los modelos *easylistening*, *softjazz* y similares. Tampoco pretendía ser una propuesta elitista orientada sólo a una audiencia determinada, sino que habría una cierta variedad de estilos y géneros. Como sonido base se tomaría lo que podría llamarse *bebop contemporáneo*, que es el tipo de música que más se ha estado haciendo en Barcelona últimamente y que ha sido incomprensiblemente ignorado en la oferta radiofónica de nuestro país a pesar de su innegable madurez e incesante evolución.

Los sellos discográficos a los que se daría una cierta prioridad serían algunos especializados como los de Freshsound, Criss Cross, SJM, New Mood Jazz, Quadrant, Omnix...

De la emisión se encargaba Live365, una de las dos grandes compañías de radio por Internet junto con SHOUTcast. Live365 tiene unas 11.000 emisoras online y más de 4 millones de oyentes al mes, además de proporcionar varios tipos de cuenta, tanto de oyente como de emisor de radio.

Acerca de los derechos de autor y los aspectos legales, los servicios de Live365 incluyen los royalties de emisión desde los Estados Unidos de música registrada en ASCAP, BMI y SESAC, así como de las sociedades de propiedad intelectual afiliadas a éstas. Asimismo, se firmó una licencia de webcasting con SGAE, realizando los pagos de derechos correspondientes a su repertorio de trabajos registrados y formalizando los protocolos de escucha de música emitida que esa sociedad exige.

Por otro lado, en la producción y selección de la programación de la radio se utilizaban herramientas específicamente orientadas al cumplimiento de las normas DMCA. Eso significa, por ejemplo, que la secuencia de temas de la lista de reproducción no se publica nunca antes de ser emitida, que todos los archivos de audio contienen metadatos originales provenientes de discos digitalizados o que se guardan unas distancias en la emisión de trabajos de un mismo autor o una misma grabación.

Por último, y con ánimo de seguir las mejores prácticas a favor de los autores, productores y distribuidores, en la programación no se incluía ninguna producción discográfica completa. Esto quiere decir que ningún trabajo discográfico se digitalizó entero ni se emitía en su totalidad. En este mismo sentido, este criterio se mantenía de forma aún más restrictiva con los trabajos de reciente aparición en el mercado, de los cuales se emitía un número más reducido de temas.

La radio no estaba dirigida sólo al público local sino que pretendía difundir el jazz de aquí a nivel internacional, aplicando las nuevas oportunidades que brinda Internet como potente canal de transmisión y aportando interactividad a un medio, la radio, que siempre ha sido de un solo sentido. Eso unido a la buena acogida que tuvo la radio en sus emisiones de prueba, y conscientes de las condiciones óptimas para un proyecto más rico, llevó a la creación de una web dedicada a complementar y a dar nombre a la emisión de música, **www.barcelonajazzradio.com**, que también sería el punto de acceso principal a la radio.

La web era sencilla y tenía el formato de blog. Contaba con información básica sobre la programación, así como con los enlaces necesarios para acercarse al jazz de nuestro país. Con la ambición de promocionarse por todo el mundo, los contenidos estaban inicialmente en inglés. Además de centrarse en la programación, pretendía ser un directorio de los músicos que la componen, con el objetivo no tanto de incluir los contenidos sino de reunirlos para facilitar su acceso, así como ofrecer la posibilidad de comprar la música que se emitía. También estaban presentes enlaces a las webs de algunas discográficas del país.

A falta de recursos, se empezó a apuntar la posibilidad de añadir elementos 2.0, como algunos widgets de otras webs (mapas de los oyentes del mundo y de los amigos de la emisora) o un enlace al perfil en MySpace Music del usuario barcelonajazzradio.

La reacción no tardó en llegar a nivel mundial: los músicos empezaron a enviar sus trabajos para ser emitidos en la radio, las discográficas se interesaron por aparecer en la lista de enlaces, los oyentes quisieron entrar en el círculo de amigos de barcelonajazzradio...

Quedó claro que crear una radio y una web para difundir el jazz de Barcelona había sido una buena idea.

1.1.2. barcelonajazzradio.com en cifras

Si bien la radio nació oficialmente el 11 de febrero de 2007, hubo un período de pruebas desde octubre de 2006, con publicidad y sin *webplayer* personalizado (los usuarios debían visitar la página de la emisora Live365 y lanzar el *webplayer* estándar). Durante ese tiempo se contabilizaron alrededor de 18.000 horas de audiencia, repartidas a mitades iguales entre el estado español y el resto de países.

En la Figura 1.2 se puede observar un mapa de GeoVisitors generado el día 14 de febrero de 2007.

Inicialmente la programación giró sobre una base de más de 600 trabajos digitalizados, a partir de los cuales se programarían unas 400 horas de música, teniendo previsto incrementar considerablemente esa cantidad durante el resto del año, contando también con las emisiones en directo grabadas expresamente y la aportación de música propia por parte de los músicos.

En la actualidad, más de un año después de su lanzamiento oficial, los números han cambiado bastante.

Datos de audiencia

- Horas escuchadas desde su creación: 400.000
- Veces que se ha disparado el stream de audio en mayo de 2008: 24.156 (Figura 1.2)
- Horas escuchadas en mayo de 2008: 19.600 (Figura 1.3)
    - 30% España
    - 30% USA, Canadá i México
    - 20% UE
    - 20% Resto de países

- Tiempo medio de escucha por oyente: 50 min (Figura 1.4)

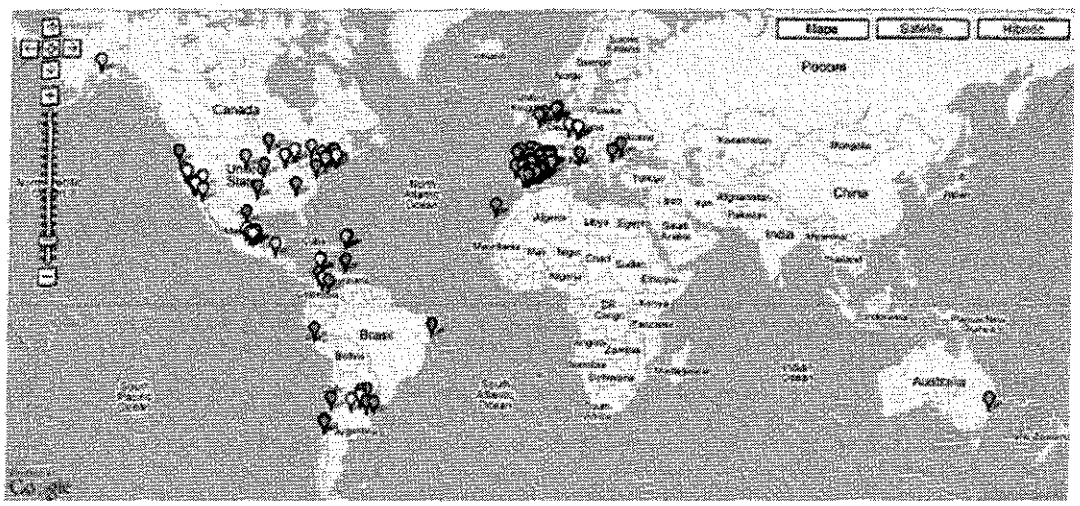


Figura 1.2: Visitantes del 14 de febrero de 2007

Country	Streams	TLH
Spain	6165	6141:14
United States	3691	2966:12
Argentina	2248	1721:19
Colombia	1222	1184:44
Mexico	1195	1016:42
United Kingdom	1112	931:59
Chile	1006	850:38
Germany	1344	481:05
Canada	445	479:23
Venezuela	281	286:12
France	235	283:00
Netherlands	429	240:12
Italy	367	217:47
Brazil	260	214:48
Peru	260	204:13
Greece	204	181:45
Czech Republic	266	173:31
Japan	211	153:49
Guatemala	321	140:27
Panama	64	127:14
Poland	127	116:20
Costa Rica	80	109:25
Israel	86	93:26
Turkey	169	90:06
Dominican Republic	80	89:45
Belgium	119	87:30
Finland	92	80:38
Australia	146	79:19
Ireland	113	76:10

Country	Streams	TLH
Switzerland	133	69:55
Austria	108	62:59
Uruguay	137	57:47
Portugal	103	56:11
Others	1337	554:54
Totals	**24156**	**19620:39**

Tabla 1.2: Horas de audiencia en mayo de 2008

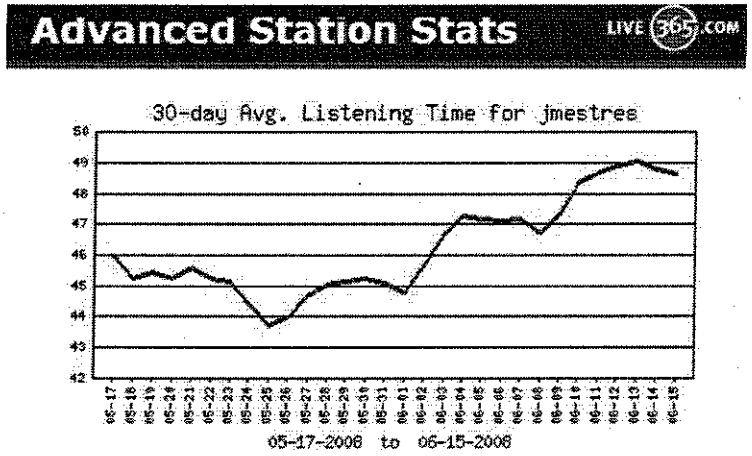


Figura 1.3: Horas de escucha mayo-junio de 2008

Posición en Live365

- Posición nº1 absoluta de la categoría Bop
- Posición nº8 de entre 500 emisoras en la categoría de Jazz (Figura 1.4)
- Posición nº138 alcanzada de entre 11.000 emisoras de Live365 (Figura 1.5)

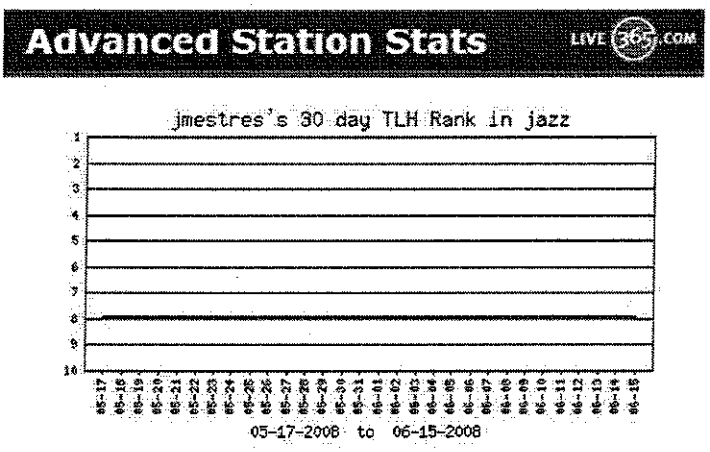


Figura 1.4: Posición 8 estable categoría Jazz

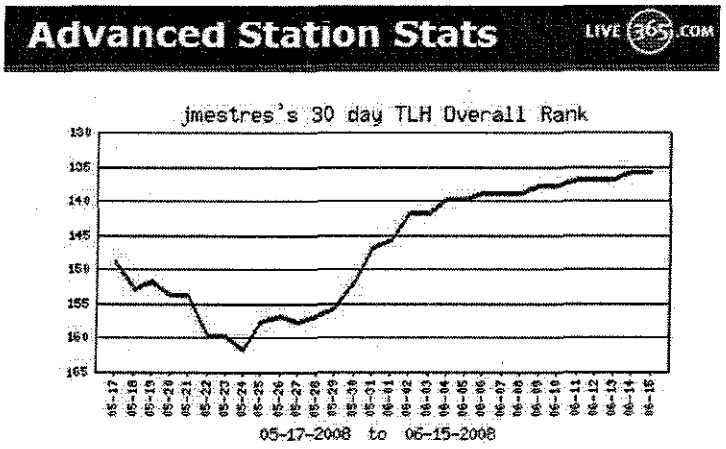


Figura 1.5: Posición 138 de Live365

Música programada

- Actualmente 6.500 temas (300 no editados)
- 95 % jazz contemporáneo hecho en los últimos 15 años
- 50% con alguna relación con Catalunya
- 25% músicos que trabajan actualmente en Catalunya
- 5 % música no editada enviada directamente por músicos

El número páginas vistas al mes de barcelonajazzradio.com, como puede verse en la Figura 1.6, es de unas 12.000 (unas 300 al día), que no es mucho, pero comparando el mapa de la Figura 1.2 (página 9) con el de la Figura 1.7 se observa un crecimiento claro, tanto en visitantes como en área geográfica abarcada, desde los inicios hasta el momento actual.

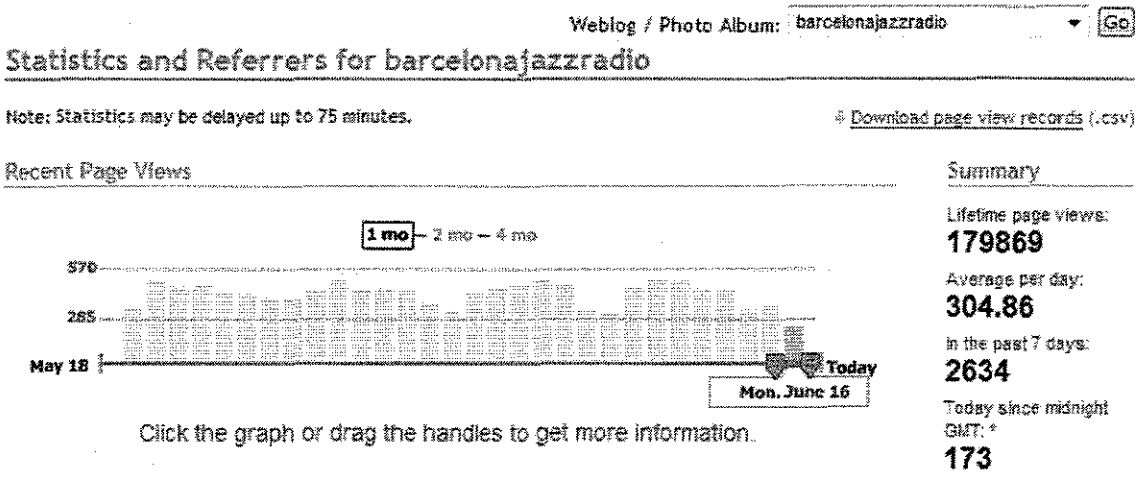


Figura 1.6: Páginas vistas al mes

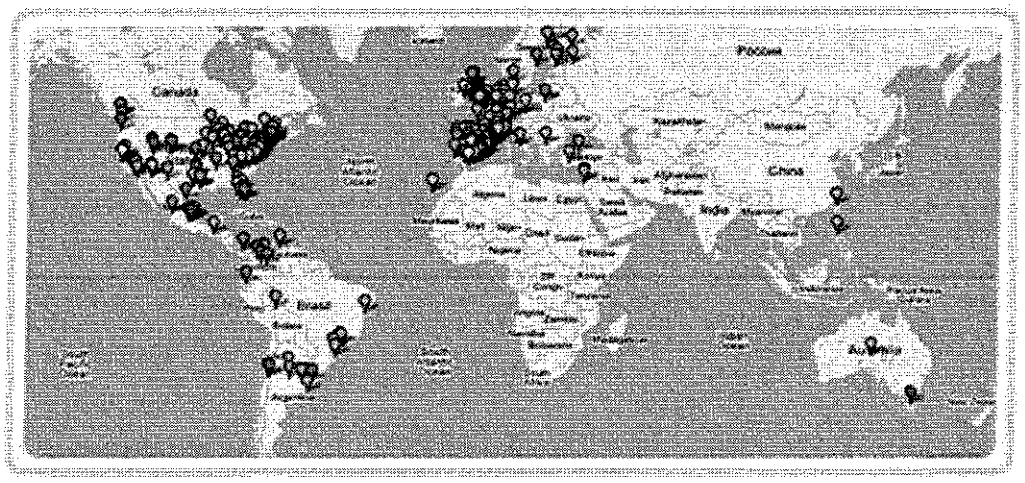


Figura 1.7: Visitantes en mayo de 2008

El usuario barcelonajazzradio de MySpace Music cuenta actualmente con un círculo de 1.165 amigos de todos los rincones del mundo.

1.2. Creación de un nuevo sitio web

1.2.1. Diagnóstico de la anterior barcelonajazzradio.com

Cuando se propuso la idea inicial que daría lugar a este y a otros PFCs, ya existía no sólo la radio por Internet, alojada en los servidores de Live365, sino también el sitio web www.barcelonajazzradio.com, realizado con la herramienta de creación de blogs Typepad y, como se ha dicho, redactada totalmente en inglés.

La radio tiene oyentes fieles, como se ha visto, que disfrutan con la música emitida y la página también cuenta con un cierto número de visitas (unas de 300 páginas vistas al día), pero falla a la hora de retener el tráfico que genera, ya que no ofrece muchos contenidos propios y sí muchos puntos de salida hacia otras páginas web. Prácticamente todo el contenido de la web se encuentra ubicado en la página principal, en la que hay, entre otros elementos (ver Figura 1.8):

- Enlaces para escuchar la radio con diferentes reproductores (Windows Media Player, RealPlayer, iTunes y un reproductor web).
- Fotografías de músicos que se van mostrando en un pase de diapositivas de www.slide.com.
- La lista de reproducción o *playlist* ofrecida por el propio servidor de la radio, www.live365.com.
- Un mapa proporcionado por www.frappr.com en el que puede verse la situación geográfica de los oyentes que quieran indicarla.
- Un banner que lleva a la red social www.myspace.com, concretamente al perfil del usuario barcelonajazzradio.
- Un banner anunciando el ciclo de conciertos que se organizó para promocionar barcelonajazzradio.com.
- Una lista con todos los músicos que suenan o han sonado en la radio online, *Featured Musicians.*
- Información de contacto animando a los artistas a enviar su música para ser emitida.

- Algunos elementos de publicidad relacionados con la música y el jazz.
- Muchos enlaces externos a páginas relacionadas con el jazz (festivales, salas, revistas, escuelas, sellos discográficos...), situados en ambos laterales de la web.

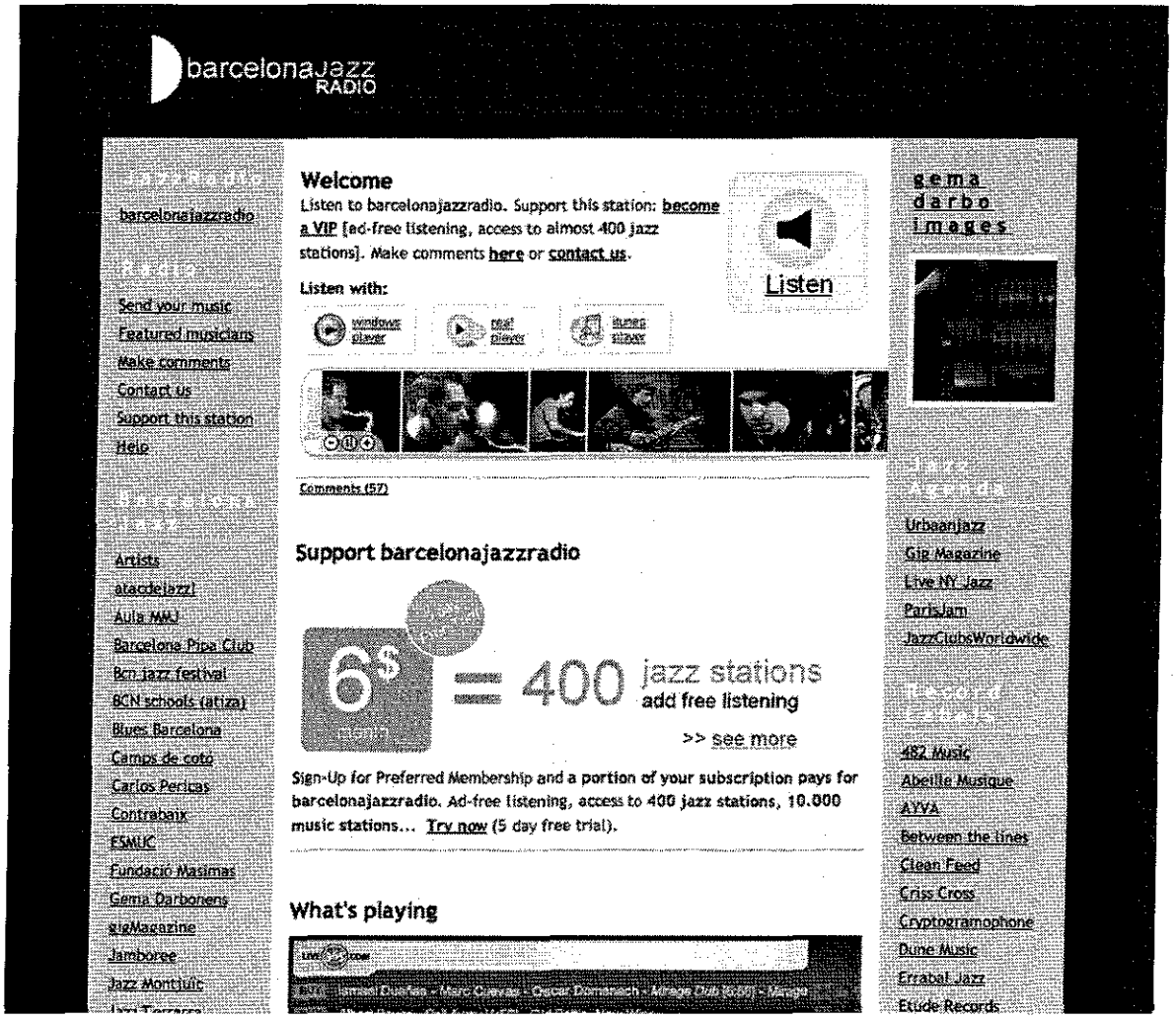


Figura 1.8: Web existente de www.barcelonajazzradio.com

Luego hay una página con una galería de fotos y otra en la que los visitantes pueden insertar comentarios. Las fotografías de calidad son tal vez el mayor atractivo de la web, ya que no pueden encontrarse en otras páginas, pero la forma de mostrarlas sin ningún orden en particular hace que los visitantes no puedan dirigirse hacia las que quieren ver, ni saber quiénes son las personas que aparecen en ellas, qué clase de música tocan o dónde pueden encontrar más información sobre ellas.

1.2.2. Motivación para un nuevo sitio web

La buena acogida de la web y de la radio impulsó la idea de hacer un remodelado general. Eso se hará en varias etapas.

1. El primer paso, del que forma parte este PFC, consiste en convertir esta web, hecha de un conglomerado de elementos externos y de enlaces a otras páginas, en una web más ambiciosa, con personalidad propia, que aporte los contenidos necesarios

para satisfacer a un tipo de oyente más sofisticado y exigente, y que a la vez sirva para proyectar la música que se hace en Barcelona hacia el resto del mundo.

2. El segundo, que se dejará para más adelante, tratará de añadir al sitio y a sus nuevos contenidos las funcionalidades 2.0 explicadas.

Para cubrir la primera fase de la remodelación se ha contado, por un lado, con el esquema general del funcionamiento lógico deseado, y por el otro, con unas especificaciones de la apariencia o *look&feel* de la web. Una de las características principales de la web, dados su intención expansiva y su carácter internacional, es que el idioma principal debe seguir siendo el inglés, tanto para los contenidos como para la interfaz.

Figura 1.9: Apartado *Featured musicians* original

Para hacerse una idea del tipo de actualización que se espera, los elementos citados arriba sufrirán algunos cambios importantes:

- La lista de más de 500 músicos existente (ver Figura 1.9), que actualmente hay que actualizar a mano y cuya intención, entre otras, es la de mejorar el posicionamiento en los buscadores, se convertirá en una lista de *enlaces* a las fichas de esos músicos, y por supuesto generada de forma automática.
- En dichas fichas podrá leerse una pequeña biografía de los artistas, ver noticias relacionadas, álbumes grabados, fotografías, etc., pasando a ser de una falsa ilusión

de contenido a una gran fuente de información, que es lo que realmente ayuda a indexar un sitio web.

- Habrá una cierta cantidad de fotografías, pero organizadas en galerías, por artista, fotógrafo, etc., y será fácil acceder a las fichas de los artistas a partir de estas galerías.
- Así como el enlace a *myspace* denotaba una clara necesidad de llevar el sitio web a un terreno más 2.0, añadiendo características de red social, se pretende dotar al sistema de la opción de dar de alta usuarios, aunque en la fase inicial será de forma controlada por el administrador y sólo para músicos cuyas interpretaciones se escuchen en la radio dejándolo preparado para dar el salto a web 2.0. Como rellenar todos los datos de las fichas de artistas puede llevar mucho tiempo, la idea es que algunos de esos músicos puedan contribuir con los suyos y llevar un cierto mantenimiento de su zona en la web.
- La web será más consciente de su función de soporte a una emisora de radio, por lo que la música que emite la radio y los contenidos de la web deben estar relacionados, a pesar de ser procesos generados por sistemas diferentes de forma independiente. Por lo tanto, sería ideal una cierta automatización en la inserción de contenidos, extrayendo la información ya existente en los metadatos de los archivos de audio, editándola y completándola, a la vez que conservando siempre la consistencia y el vínculo entre estos metadatos y la información que ofrece el sitio web. Para ello se ha de desarrollar un sistema que lea estos metadatos, permita editarlos y completarlos con información adicional y guarde la información resultante como contenidos del sitio web, así como en forma de nuevos valores de los metadatos de los archivos de audio que finalmente serán subidos al servidor de streaming. De esta forma, la música se emite en compañía de metadatos que permiten vincularla en tiempo real con los contenidos del sitio web.
- El punto de unión entre la emisión sonora y los contenidos de la web se hará a través de la lista de reproducción o *playlist*. En la versión existente, que se puede observar en la Figura 1.10, la playlist es ofrecida por Live365. Incluye el tema que está sonando actualmente y los dos temas que han sonado justo antes. Al hacer clic en alguno de los temas en este tipo de playlist, el usuario es redirigido a una url desde la que puede comprar el álbum al que pertenece el tema. El administrador puede definir esa url de compra para cada tema, pero de forma muy limitada. En la nueva versión del sitio web debe haber una playlist propia, ocupando el mismo lugar en todas las páginas, que no desentone con el *look&feel* de la web y cuyos enlaces no lleven a la url de compra del álbum sino a las fichas del artista y del álbum en barcelonajazzradio.com, de modo que cuando alguien quiera información sobre lo que está escuchando, lo único que tendrá que hacer es mirar en la playlist y hacer clic.

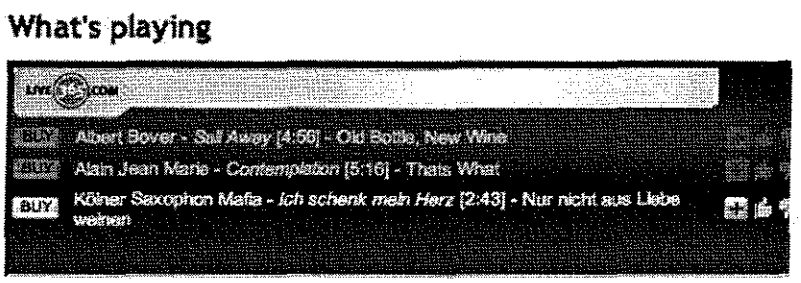


Figura 1.10: Playlist original de Live365

A parte de todo esto, los visitantes de la web tendrán acceso a galerías de imágenes, listas de álbumes, noticias relacionadas con el jazz en Barcelona, búsqueda de contenidos, etc., además de poder añadir comentarios a los distintos contenidos. Y los músicos, que son los

que salen realmente beneficiados, y a quienes va dirigido este proyecto, tendrán la oportunidad de darse a conocer y de enlazar su ficha con su web personal y con su perfil de myspace, en caso de que los tengan, así como conocer opiniones de otras personas sobre su música y poder responder a ellas, creando así un punto de encuentro entre músicos y aficionados.

La emisora no se ve modificada por el proyecto. Sin embargo, partes del proyecto tienen por objetivo facilitar la modificación de los metadatos de los archivos mp3, los temas que se programan, de forma que estos datos permitan hipervincular los temas que suenan con la información que sobre los mismos, sus autores e intérpretes y sobre los álbumes a los que pertenecen se ofrece en el sitio web.

2. Entorno del PFC

Este PFC forma parte de un proyecto de mayor envergadura que consiste en la actualización y puesta a punto de un sitio para una radio por Internet. Concretamente, la realización de este Proyecto Final de Carrera se ha hecho en conjunto con otros dos PFCs, y al terminarlo ya había otro alumno dispuesto a continuar con esta labor. Es posible que aún se puedan realizar dos PFCs más para completar toda la funcionalidad deseada. Para comprender bien los objetivos del PFC, pues, es necesario explicar el conjunto del proyecto, detallando sus distintas partes.

De entrada hay que diferenciar dos grandes fases en el proyecto completo: por un lado, la generación de contenidos de la web y su relación con la música que suena en la radio; por el otro, la dotación del sistema de funcionalidades de comunidad online y la conversión del sitio en una red social. Los tres PFCs mencionados se han destinado a la implementación de la primera fase, sentando también unas ciertas bases para que se pueda realizar la segunda fase con comodidad.

Como la segunda fase no se ha comenzado a realizar y habitualmente necesitaremos hacer referencia al trabajo de los otros dos PFCs que han acompañado a éste, desde este momento nos referiremos al trabajo conjunto de los tres PFCs como *la primera fase* de la remodelación, *el proyecto global* o, simplemente, *el proyecto*. Cuando necesitemos hablar del trabajo de este Proyecto Final de Carrera en concreto se dirá explícitamente *este PFC*. Y si en algún momento hace falta hablar de los cambios futuros se dejará muy claro que se habla de la segunda fase y de funcionalidades 2.0, y que lo comentado no entra en el ámbito del proyecto actual.

2.1. Descripción detallada del proyecto

El objetivo final del proyecto es el de crear una aplicación web que dé soporte a una emisora de radio por Internet. Tanto la emisora online como el sitio web ya existían antes de iniciar el proyecto, pero se pretende sustituir el sitio web por una aplicación web que, por una parte, facilite la incorporación y gestión de sus contenidos manteniéndolos vinculados con la música emitida y, por otra, ofrezca a los usuarios más funcionalidad para consulta de los mismos, empezando a mostrar un incipiente carácter de comunidad.

2.1.1. Conceptos previos

Las emisoras de radio online como la que nos ocupa funcionan de la siguiente manera: la música es comprimida en archivos, generalmente en formato mp3, y es almacenada en un servidor. Los usuarios de la radio se conectan al servidor a través de Internet y el servidor, siguiendo el orden de una *playlist* o lista de reproducción, les va enviando de forma paulatina fragmentos consecutivos de esos archivos, que son reproducidos inmediatamente en su ordenador. Este proceso de envío y reproducción constantes se conoce con el

nombre de *streaming*, y tiene la ventaja de que no requiere que el usuario descargue el archivo entero antes de poder empezar a escucharlo, sino que va escuchando música desde el principio a medida que la va recibiendo. El mismo sistema se utiliza para vídeos. Los archivos tienen un cierto nivel de compresión o *bitrate*. Cuanto más comprimido está un archivo (menor bitrate) más rápida es su transferencia y menores los requerimientos de ancho de banda, pero también es más baja su calidad. La conexión del ordenador del cliente al servidor se hace mediante algún tipo de software, que puede ser proporcionado la misma radio, ya sea en forma de programa que instala el usuario en su ordenador o bien en forma de reproductor web (en tal caso sólo es necesario un navegador web para poder recibir la música); también puede tratarse de un reproductor ya existente en el ordenador del cliente al que se indica cómo acceder al stream de audio.

La emisora que ocupa a este proyecto, www.barcelonajazzradio.com, emite jazz de forma continua a partir de una colección de archivos mp3 obtenidos de diversas fuentes (CDs, compras online, grabaciones de los propios músicos...). Tanto el alojamiento de esos archivos como su transmisión mediante streaming forman parte de los servicios que ofrece www.live365.com, una empresa dedicada a la creación y mantenimiento de emisoras de radio por Internet.

Respecto a las características de los archivos, provenga de donde provenga, todo archivo mp3 cuenta con la posibilidad de almacenar datos en los llamados *tags ID3* (o etiquetas ID3), que son fragmentos de metainformación que van incluidos en su cabecera. De esta forma, un mp3 no está formado únicamente por contenido musical comprimido sino que, si tiene correctamente rellenadas estas etiquetas, puede proporcionar, además, datos relevantes como el título de la obra, el intérprete, el nombre del disco, el número de pista dentro del disco, etc. Estos metadatos son utilizados de forma habitual por los reproductores de música para mostrar al usuario lo que está sonando en cada momento, y los reproductores de las radios online no son una excepción, si bien la corriente de audio y de metadatos se proporcionan por canales separados.

Los tags ID3 son útiles, pero la información que cabe en ellos es limitada y hace referencia exclusivamente al propio archivo. Es decir, un archivo mp3 *no sabe* qué otros archivos pertenecen a su mismo álbum o a su mismo autor. Aún así, leyendo la información ID3 de una colección suficientemente grande de mp3 se puede crear una base de datos interesante, que relacione los temas de un mismo disco o los discos de un mismo autor. Ésta es, en líneas generales, la primera de las dos grandes fases del proyecto, la obtención de los metadatos de la música que se emitirá en la radio; la segunda parte consiste en la ampliación de esta información, algo escasa, con contenido que sea útil tanto para el usuario de la radio como para los artistas cuya música se emite, de modo que se les proporcione una experiencia más interactiva y comunitaria.

2.1.2. Subsistemas que conforman la solución propuesta

Veamos con más detalle el funcionamiento del sistema sobre el que han trabajado los tres primeros PFCs, comenzando por definir sus dos componentes principales:

- *Music Library Editor (MLE):* Es un programa que se ejecuta de forma local y que sirve para visualizar y editar el contenido de las etiquetas incrustadas en los ficheros mp3 (tags ID3), como el artista, el álbum, el año de edición, etc. Así, extrae la información incluida en estas etiquetas ID3 del conjunto de ficheros de audio que conforma la colección de mp3 que se programa en la radio, la ofrece al usuario que administra los contenidos para que la edite y complete con datos adicionales y, finalmente, guarda los contenidos resultantes en la base de datos del sitio web, además de sobrescribir los valores de las etiquetas ID3 con los nuevos valores editados.
- *Radio Website (RW):* Es el sitio web que da soporte a la radio y su funcionamiento se describe con detalle en los próximos capítulos. A grandes rasgos, este espacio web pretende ser un lugar de conexión en la red entre oyentes, artistas, y

aficionados al jazz que proporcione la funcionalidad de una comunidad virtual a nivel mundial. Los usuarios podrán escuchar jazz, aprender más sobre los músicos leyendo sus fichas o posts que traten sobre ellos, compartir opiniones, hacer peticiones musicales o enviar su propia música para que sea emitida en la radio.

La conexión entre estos dos componentes hay que entenderla como sigue: el usuario accede al sitio web y decide escuchar la radio. En un momento dado, le interesa saber más sobre lo que se está emitiendo, de modo que se fija en la información que le dan el reproductor o la playlist del propio sitio web (proveniente de los tags ID3 del archivo) y, a partir de ella, accede a las páginas del sitio en las que se complementa esa información. Por ejemplo, puede leer sobre cada uno de los músicos que tocan o sobre el disco en cuestión. Si le apetece, puede hacer un comentario sobre lo que está escuchando como respuesta a un post del sitio web, al que podrán responder otros usuarios, ocasionalmente los mismos músicos a los que hace referencia el post.

Para que todo esto sea posible, la clave está en vincular de algún modo la información ID3 de los archivos obtenida por el MLE (Music Library Editor), que son simples campos de texto, con el resto de información del RW (Radio Website), que son páginas web con información textual, imágenes, etc., de tal forma que la introducción de datos en el RW pueda hacerse de forma manual o bien automáticamente a partir del MLE. Es decir, que el MLE y el RW tienen que trabajar sobre la misma información, y la comunicación tiene que existir en ambos sentidos:

1. El MLE tiene que poder generar datos que sean luego accesibles y modificables por el RW y que contengan la información mínima que ha extraído de los tags ID3.
2. El MLE tiene que poder acceder a la información ya almacenada en el RW para completar o relacionar los datos con los que trabaja, pudiendo modificar así los tags ID3 de los mp3.
3. Hay que evitar en todo momento la sobreescritura o eliminación no controlada de la información en cualquiera de los dos sentidos.

Es necesario, pues, algún tipo de sincronización o comunicación entre los dos componentes. Primeramente se pensó en dos bases de datos diferentes, una para el MLE y otra para el RW, y una sincronización explícita de la información, pero luego se optó por usar la misma base de datos y que las dos partes compartieran la misma información. Ambos métodos deben ser igualmente cuidadosos a la hora de actualizar la información, pero el segundo elimina muchos problemas potenciales a la hora de crear los protocolos de comunicación.

2.1.3. Esquema general del sistema

El funcionamiento de la radio se puede separar claramente en dos niveles y una capa intermedia de comunicación.

- Por un lado, el nivel *local*, donde se hace el tratamiento de los archivos de audio y se preparan los que serán utilizados en la radio
- Por el otro, una capa de *comunicación*, que permite el acceso a base de datos remota
- Finalmente, el nivel *remoto*, en el que se encuentra el RW y la inserción de contenidos extra, así como el servidor de Live365

En la Figura 2.1 se pueden observar estos niveles con los elementos que los constituyen, y a continuación se detalla el funcionamiento de éstos.

Nivel local: Colección de MP3, MLE y actualización de la programación

En el nivel inferior del esquema se encuentra la parte que tiene más que ver con los archivos mp3. Dichos archivos se obtienen de varias fuentes, como pueden ser CDs de música, tiendas online o envíos por parte de músicos. En cualquier caso, el resultado es una colección heterogénea de mp3, comprimidos a diversos bitrates y con información ID3 irregular. Hace falta un proceso de normalización para obtener un resultado uniforme. Tal proceso constaría de dos partes:

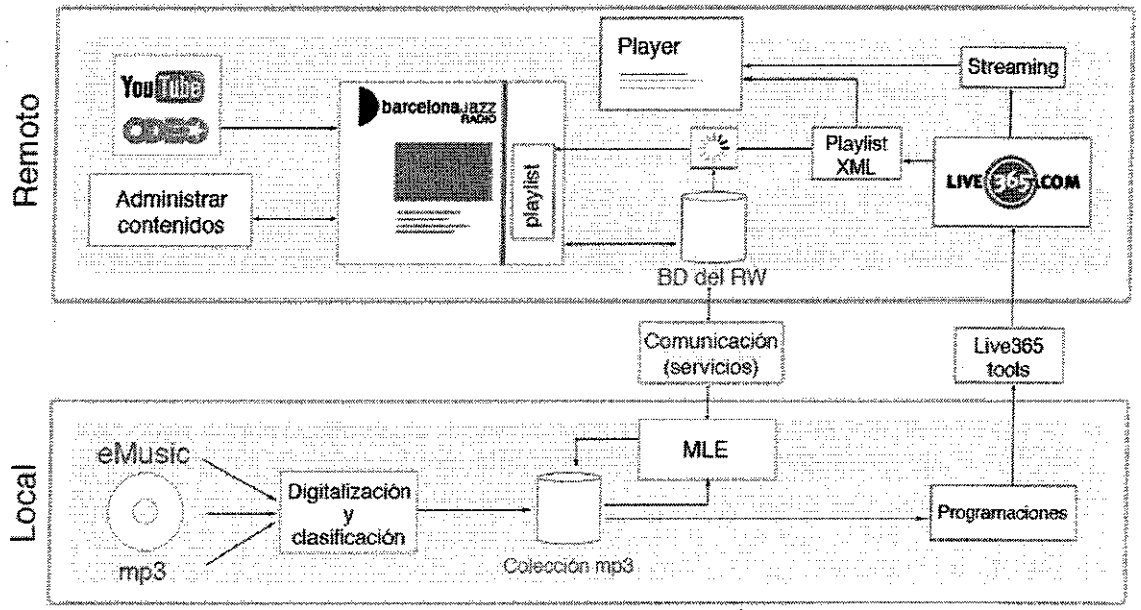


Figura 2.1: Esquema del sistema

1. Editar los metadatos contenidos en las etiquetas ID3 para uniformizarlos y añadir información adicional: dependiendo del sistema por el que se hayan obtenido los mp3, la información ID3 puede estar incompleta o incluso no existir. Si se adquieren en una tienda online, los mp3 probablemente tendrán bastantes datos y serán correctos; si se extraen de un CD, el mismo programa que los extrae en muchos casos se conecta a una base de datos externa como CDDB o FreeDB y, a partir de un identificador del CD, obtiene la información ID3 de cada una de sus pistas, que luego añade a los archivos resultantes. A pesar de todo esto, como el contenido de estas etiquetas se usará como base de partida para generar contenidos mucho más extensos y completos, pero que han de mantener su vinculación con los metadatos, es importante que la información sea tratada de una forma sistemática que asegure su consistencia. Y aquí es donde se requiere la funcionalidad del MLE.
2. Recomprimir todos los archivos al mismo *bitrate* para que puedan ser transmitidos por la radio y subirlos al servidor de streaming junto con las características del tipo de programación con que se desean emitir: aleatoria, definida mediante parrilla, etc. El proceso de recompresión debería dejar intacta la información ID3. Estas operaciones se realizan con unas herramientas que proporciona Live365.

El conjunto de metadatos de toda la colección de mp3 que se programa en la radio constituye una base de datos de estructura peculiar y con serias limitaciones en cuanto al mantenimiento de la consistencia de la información. MLE se ocupa también de asegurar esta consistencia ya permite acceder de forma sencilla a los metadatos de los mp3 y modificarlos de diversas formas. Por ejemplo, el nombre de un artista puede estar escrito de forma diferente en varios mp3 aunque se refieran al mismo (variantes tipográfica, etc.);

MLE ofrece la posibilidad de guardar el valor del nombre de artista de uno de los archivos en la base de datos y luego, cada vez que se editen otros mp3 del artista, tomar el valor que se ha almacenado como válido y escribirlo en los nuevos archivos. De esta forma, se traspasa la consistencia de la base de datos a la colección de mp3.

Además, permite añadir en la base de datos información que no está presente en los tags ID3, no sólo porque algunos tags pueden estar vacíos sino porque, como se ha dicho, estas etiquetas son limitadas y en ocasiones no contienen una información tan exhaustiva como se podría desear. En el caso de este proyecto, al tratarse de una base de datos de artistas y álbumes de jazz, algunos datos adquieren mayor relevancia, como pueden ser la lista completa de los músicos que tocan en un determinado tema o los instrumentos que toca cada músico en un disco. Los contenidos resultantes de este proceso de corregir y completar los metadatos son lo que se denominará de ahora en adelante el *Catálogo musical* de la radio. **El catálogo se almacena en la base de datos del RW y, de forma parcial y limitada también se almacena en las etiquetas ID3, tanto como lo permiten los tipos de metadatos de este estándar.**

En el nivel local se encuentra otro elemento importante, consistente en la creación y actualización frecuente de la programación de la radio, mediante la sustitución del subconjunto de ficheros mp3 que están alojados en el servidor de streaming de Live365 por otro subconjunto diferente, para mantener una programación fresca y no repetitiva. Live365 proporciona herramientas para hacer desde una lista específica hasta una ordenación aleatoria. En general, en el servidor de Live365 no se alojan todos los mp3 de la colección de la emisora, sino el subconjunto que se desea emitir durante las próximas 30 horas aproximadamente, ya que la capacidad de almacenamiento contratada es limitada. Por lo tanto, habrá que irlos subiendo y reemplazando regularmente, a medida que se quieran emitir.

Hay que destacar un hecho que se observa en el esquema de la Figura 2.1 relativo a la comunicación entre el nivel local y el nivel remoto. Los resultados obtenidos de la labor del nivel local son, por un lado, los datos generados a partir de la edición de los tags ID3 (Catálogo musical) y, por el otro, el conjunto formado por los archivos mp3 y la playlist. Como se puede ver, desde este momento cada uno de estos dos resultados sigue su propio camino: los contenidos del Catálogo viajan a través de unos servicios web hasta quedar almacenados en la base de datos relacionada con el sitio web de la radio mientras que los archivos, los metadatos y la programación van al servidor de Live365. Esto obliga a ingeniar un sistema que permita mantener vinculadas ambas versiones. El hecho de trabajar con un servicio de emisión ajeno al propio sitio web hace que la información se tenga que separar para luego volver a unirse, con la dificultad que esto supone de cara a mantener los vínculos entre las dos vías por las que viaja la información hasta el usuario: sitio web y stream de la radio.

Nivel remoto. RW y Live365

En el nivel remoto se cuenta con los datos recibidos desde el nivel local (mp3 y playlist) y con los que llegan a través de la capa de comunicación (Catálogo musical proveniente del MLE), y se trabaja con ellos para generar información útil.

Los dos elementos principales de este nivel son el sitio web de la radio o Radio Website (RW) y la emisora propiamente dicha, alojada en Live365.

El RW consiste en un sitio web con información sobre los artistas y los mp3 que suenan en la radio, junto con todo lo que les rodea: álbumes, fotografías, partituras, etc. Toda esta información debe ser creada y mantenida por un administrador de contenidos. Para hacer fácil esta tarea, se ha optado por el uso de un *CMS* (*Content Management System* o *Sistema Gestor de Contenidos*). Los detalles relativos a la implementación se irán viendo más adelante aunque, sin ánimos de desvelar el final, adelantaremos que el CMS elegido es Drupal. Por ahora bastará con decir que con un sistema gestor de contenidos se permite la inserción, modificación y eliminación de información de manera sencilla, de tal forma que esos contenidos son almacenados en la base de datos por el propio CMS, y las páginas web

que ve el usuario se generan a partir de esos datos y sólo en el momento en que se hace la petición a través de su navegador. Por supuesto, el usuario administrador puede añadir mucha más información al Catálogo con el RW que con el MLE: textos, enlaces, imágenes, vídeos u otros elementos almacenados en sitios web externos, etc.

Por otro lado, en este nivel remoto se puede observar la emisora de Live365, que ha recibido del nivel inferior los archivos mp3 que serán escuchados en la radio y la lista de reproducción que contiene la relación de los mp3 que se desea emitir. Los mp3 son almacenados en el servidor y se accede a ellos cuando toca emitirlos, con lo que se produce una primera salida del servidor: los datos a emitir mediante streaming. Antes se ha comentado que el acceso a una radio online se podía hacer mediante el reproductor que proporciona la misma emisora (y cuya apariencia puede ser personalizada al gusto del administrador del sitio) o bien usando reproductores que el usuario tiene instalados en su propio ordenador (como Windows Media Player, iTunes o RealPlayer), siempre que se indique a estos programas cómo acceder al contenido de audio. Pues bien, lo que hay que indicarle para que empiece a reproducir es la URL de este stream de audio.

Live365 proporciona, por otro canal, acceso a los metadatos del tema que se está emitiendo y los 9 últimos, en formato XML. A esta información se le suele llamar *playlist* y ofrece los siguientes datos: el título del tema, el artista, el álbum, el nombre del archivo mp3, una descripción, el tipo de pista, algunas url asociadas y la duración. Es decir, son precisamente los metadatos del tema actual y los 9 anteriormente emitidos que el servidor de streaming extrae de los archivos de audio para construir la playlist en formato XML, actualizándola cada vez que cambia el tema 'en el aire'. Estos metadatos sirven para ser mostrados en el reproductor en el momento de la emisión. El reproductor no sólo reproducirá el audio sino que además mostrará lo que está sonando. Para ello necesita conocer también la URL de ese archivo XML.

Es importante resaltar que el audio y los metadatos se proporcionan por dos canales diferentes. El reproductor accede tanto a la música, que se reproduce de forma continua, como a esa playlist en XML de forma puntual —la playlist incluye también el tiempo que resta aun al tema que suena, de forma que el reproductor sabe cuando ha de volver a solicitar de nuevo la playlist actualizada porque se emite un nuevo tema—. Cuando llega ese momento, el reproductor pide la nueva lista de reproducción y, si no ha habido retrasos en la emisión, en ese mismo instante el usuario verá que termina el tema actual y empieza el siguiente. Pero es importante remarcar que son procesos separados.

El usuario, pues, recibe dos tipos de contenido de este nivel: la música reproducida por el método que él elija y una página web donde complementar la información que le ofrece el reproductor.

En la web puede acceder a páginas de músicos y bandas, de discos, de fotos, etc. Estas páginas pueden contener partituras, vídeos o archivos de música alojados en servicios externos. El usuario puede ver la formación de músicos de un determinado disco o encontrar enlaces relevantes sobre estos músicos en Internet. Puede participar en foros, hacer peticiones de música para la radio y, si es músico, enviar su propia música como propuesta para ser emitida.

La conexión entre la música que suena y el catálogo del sitio web se produce de dos formas distintas:

- Si se está escuchando la radio a través del reproductor web, que se abre en una pequeña ventana del navegador, o a través de cualquier otro reproductor, se puede saber el tema que se está emitiendo y consultarlo en la web. En este proyecto no está contemplado, pero en posibles PFCs futuros se podría modificar el código HTML del player de Live365 para que muestre enlaces al sitio web.
- El sitio web incluye una playlist con los últimos temas que han sonado que se va actualizando automáticamente. Esta lista de temas la obtiene de la playlist en XML, que está disponible en una ubicación específica. Como se sabe el tiempo que durará un tema gracias al XML, se puede calcular el tiempo que falta para actualizar esa

pequeña playlist y, transcurrido ese tiempo, volver a consultar a Live365. Para obtener la lista nueva. En caso de errores de comunicación o de que la lista obtenida sea idéntica a la que ya tenía, se seguirá consultando cada cierto tiempo hasta conseguir una lista diferente.

Y aquí es donde se vuelven a unir los dos flujos de datos que se habían separado en el nivel local: el de Live365 y el del RW. La lista de los últimos temas reproducidos se genera con información recibida de Live365, pero el sistema tiene que ser capaz de identificar, a partir de esa información, la parte del Catálogo a la que se hace referencia, para poder transformar, por ejemplo, los nombres de álbumes y artistas en enlaces a las páginas de la propia web. Lo ideal sería poder pasar en el XML de Live365 un identificador único de cada tema, de modo que existiese una relación unívoca entre ambas partes. A causa de lo limitado de este XML, al que no se le pueden añadir campos, esto se conseguirá usando el nombre del artista, el nombre del álbum y el nombre del tema como delimitadores, ya que es un conjunto de datos que siempre identificará un único tema. Existen algunos casos concretos en los que se podrían crear pequeños conflictos (ver sección 2.3.4), pero son de fácil solución.

En la Figura 2.1 se puede observar la creación de la playlist como la unión de los datos del XML con los de la base de datos en un proceso que se ejecuta automáticamente en el momento en que cambia el tema que está sonando.

Capa de comunicación. Servicios

Como se ha dicho, el MLE recoge los metadatos de los mp3 y los almacena en una base de datos. Esa base de datos es la misma que utiliza el CMS para guardar la información de la página web que introduce el administrador manualmente.

Por lo tanto, el proceso de almacenamiento de los metadatos en la base de datos del sitio web tiene más relevancia de lo que pueda parecer, por dos razones:

1. Al ser una base de datos de un CMS, el MLE debe adaptarse a sus características y emular algunos comportamientos. Se presenta la dificultad de conseguir que el MLE, creado especialmente para el proyecto, almacene los datos en la BD de forma idéntica a como lo hace el gestor de contenidos existente, para que cuando éste necesite recuperarlos para mostrarlos al usuario tengan siempre las mismas características. Eso requiere un estudio en profundidad del CMS, y una cierta ingeniería inversa que permita conocer cómo están estructurados los contenidos en la base de datos, esta parte es una pieza clave de este PFC.
2. La base de datos no será local como se pensaba en un principio, sino que habrá que darle acceso de forma remota. La mayoría de hostings no ofrecen la posibilidad de acceder remotamente a sus bases de datos, ya que eso supondría un gran riesgo de seguridad. Cualquier consulta o modificación debe ser hecha desde el propio servidor del hosting.

Eso hace que se requiera la creación de unos servicios alojados en el servidor que permitan al MLE acceder a la base de datos de forma remota y manipularla de acuerdo con la forma de trabajar del CMS. Debido a su grado de complejidad y su necesidad para entender el funcionamiento del proyecto, la conexión con la base de datos se explicará en detalle en el capítulo de implementación, ya es una parte extremadamente importante y delicada implementada en este PFC.

2.2. Usuarios del sistema

Un sitio web como éste está pensado para que gente de todo el mundo pueda acceder a él en busca de información y de música para escuchar. Por otro lado, el contenido del sitio debe ser introducido y mantenido por alguien. Por lo tanto, es necesario crear varios *roles* o tipos de usuario con unos determinados permisos cada uno. Se ha determinado la necesidad de los siguientes roles:

- *Administrador (ADM):* Se encarga de todo el mantenimiento del sitio, lo que significa que debe ser capaz de crear contenido como páginas de artistas y de álbumes, galerías de imágenes, información sobre los autores de esas imágenes, etc. También mantendrá actualizado el contenido dinámico de la página principal, que tendrá una estructura similar a un blog, cuyos posts podrán estar relacionados con otras páginas del sitio.
- *Músico (MUS):* Si un artista de los que suenan en la radio cuenta con una ficha de artista en el sitio web y le interesa implicarse más en la difusión de su música, podrá ponerse en contacto con el administrador para solicitarle una cuenta de músico. Con esta cuenta tendrá la posibilidad de acceder a su ficha (y tal vez a las de las bandas de las que sea líder) y modificarla, añadiendo texto, fotografías, enlaces, etc.
- *Usuario no registrado o invitado (INV):* Estos usuarios son los visitantes del sitio web y no disponen de cuenta. Pueden ver las páginas del sitio y contactar con el administrador para hacer peticiones de música para la radio. También pueden participar en los foros y escribir comentarios a los posts del sitio. Naturalmente, cualquier cosa que pueda hacer un invitado la puede hacer cualquier otro usuario.
- *Sistema (SIS):* El usuario sistema es el que se encarga de tareas rutinarias que no necesitan la atención del administrador, como pueden ser hacer copias de seguridad, actualizar ciertos contenidos, enviar sitemaps a buscadores, etc.

El hecho de que sólo exista un tipo de usuario registrado además del administrador, que es el músico, puede hacer pensar que no es necesario crear un perfil específico que ofrezca funcionalidades comunes, ya que todos los usuarios registrados tendrán las mismas. Si se crea un perfil es pensando en futuras ampliaciones de la web. Tal vez se quiera añadir más adelante un perfil para usuarios fotógrafos, o cuentas *Premium* que ofrezcan más privilegios a los músicos.

Todo usuario registrado podrá determinar en su ficha de usuario si quiere que los demás visitantes vean si está conectado o no.

Hay que destacar que no existe un proceso de registro por parte de los propios usuarios, sino que es el administrador el que decide si se proporciona una cuenta a alguien o no, esto es así debido por el propio deseo del administrador de la radio.

La participación en los foros no necesitará de un registro previo, sino que requerirá que el usuario escriba un nombre, una dirección de correo y opcionalmente una web. Si el usuario está registrado (porque es un músico o el administrador), estos campos habrán sido rellenados previamente para que no necesite escribirlos.

Hay otro hecho a matizar, y es que existen tareas que son hechas por alguien que tiene claramente privilegios de administrador pero que no están relacionadas directamente con el sitio web y su contenido. Por ejemplo, alguien debe ejecutar el programa MLE para modificar y añadir datos al Catálogo musical, y del mismo modo alguien tiene que subir los archivos a Live365 y crear las listas de reproducción. Todo esto se hace al margen del RW, pero debe ser tenido en cuenta.

Aún existe otro tipo de usuario en el RW que no se ha mencionado, y es el administrador del CMS. Todo sistema gestor de contenidos tiene muchas otras tareas de administración además de la creación y el mantenimiento de contenidos y usuarios: ver registros de acceso y de errores del sitio, modificar la forma en la que se muestran los elementos, configurar parámetros diversos... No se contemplarán las tareas de este administrador en la lista de requisitos.

2.3. El Catálogo o librería musical

La base de datos del sitio guarda todo lo referente a los músicos relacionados con la escena jazzística de Barcelona, en especial los que suenen en la radio de barcelonajazzradio.com, y a su entorno. Se trata de una base de datos de jazz, lo que implica que tiene algunas necesidades importantes que no tienen otros tipos de catálogo musical. Por ejemplo, todo álbum debe contener información precisa sobre los músicos que intervienen en cada tema y los instrumentos que tocan estos músicos en el álbum.

El MLE recopila parte de esta información a partir de las etiqueta ID3 de los mp3 que complementa con datos adicionales que introduce el usuario. Lo que falta para completar la base de datos debe ser introducido mediante el RW.

A continuación se explican los elementos que intervienen en el Catálogo, listando los datos que se almacenan de cada uno —se profundizará en ello en el capítulo 5— y la funcionalidad que se desea para manipularlos desde el sitio web.

2.3.1. Artistas

En el vocabulario usado en el proyecto se llama artista tanto a un *músico* como a una *banda de músicos*. Esto es así porque los álbumes pueden estar a nombre de un músico individual o de un grupo. En jazz es habitual que unos músicos toquen con otros de forma esporádica, de modo que también hay álbumes a nombre de varios músicos en vez de ser el nombre de una banda (por ejemplo, un álbum puede tener como un único artista a *Roy Hargrove, Christian McBride, Stephen Scott*). En estos casos se considerará al conjunto completo de los músicos como artista, y no a cada uno de ellos. En este sentido, un artista se podría definir, en general, como cualquier persona o grupo de personas que tiene algún disco a su nombre en barcelonajazzradio.com. Normalmente será así, pero también puede ser que exista algún músico o agrupación de músicos que toque habitualmente en Barcelona pero que no haya grabado todavía ningún disco y aún así el administrador de contenidos del sitio haya decidido crear una página con su información.

Sea como sea, hay datos que son comunes a todo artista, mientras que hay otros que son específicos dependiendo de si el artista es un músico o una banda.

Entre los comunes estarían el nombre del músico o banda, una pequeña descripción o biografía, un enlace a su página web personal y un id de myspace (en caso de que los tengan), y una lista de enlaces a páginas web que hagan referencia al artista, así como una relación de los álbumes que ha grabado y los posts del sitio web que estén relacionados con él. También se incluye una fotografía que será la imagen de su ficha. Los datos que diferencian a un músico de una banda son los siguientes:

- ***Músico***: El instrumento o los instrumentos que toca habitualmente. Este campo podrá ser rellenado de forma automática a partir de la información de los discos en los que toca, pero será un campo modificable por el administrador.
- ***Banda***: La formación, es decir, los componentes del grupo. Constará del nombre de los músicos que la forman y de los instrumentos que toca cada uno. En caso de tener ficha, el nombre de cada uno de estos músicos será un enlace a esa ficha. La determinación de la formación de una banda conlleva algunas dificultades. En términos generales, una banda es una formación de músicos que ha grabado algún disco, pero es habitual que los músicos cambien de actividades y que esas formaciones sufran cambios con el tiempo. Si, por ejemplo, una banda tiene normalmente unos ciertos componentes pero en un disco concreto uno de ellos ha sido sustituido por otro, eso no debería afectar a la formación habitual. Y los músicos invitados de un disco tampoco deberían aparecer como fijos. Por eso se podrá determinar la formación de una banda de forma automática a partir de los discos pero también será posible editar manualmente estos campos. Aún así, habrá casos difíciles de clasificar, como por ejemplo bandas que han tenido dos o más formaciones importantes a lo largo de su historia, todas con el mismo nombre, y

que no sea admisible obviar ninguna de ellas. En este caso existen varias opciones: crearlas como bandas distintas en el Catálogo —por ejemplo, añadiendo una numeración como sufijo: *Bill Evans Trio (II)* —; añadir varias formaciones en la ficha de la banda, indicando los años entre los que estuvo activa; o bien usar sólo una de ellas y explicar textualmente, en la descripción de la banda, las distintas etapas por las que pasó y los músicos que pertenecieron a ella en cada etapa. Se ha optado por permitir en la ficha de artista una única formación, de modo que se deja al criterio del administrador de contenido la decisión sobre los músicos que aparecerán formando parte de una banda. Otro hecho a tener en cuenta es que habrá que poder especificar cuál de los componentes es el líder de la banda, si es que lo hay.

La ficha del artista podrá ser visible o no según lo decida el administrador. Si, por ejemplo, un artista no tiene más información que el nombre, porque aparecía como invitado en un disco, el administrador puede decidir que no se muestre todavía su ficha y esperar a tener más datos.

En el sitio web habrá una página que contendrá la lista de todos los artistas (*featured artists*), con enlaces a sus fichas, en caso de que estén activas, lo que incluirá tanto músicos como bandas.

2.3.2. Álbumes

Un álbum es la representación de un disco en el Catálogo. Tiene un título, un año, una discográfica y una ciudad de edición, una lista de temas y una formación de músicos.

Hay que destacar la diferencia entre la formación de músicos de una banda como se ha explicado en el apartado anterior y la formación de músicos de un disco. En una banda se entiende por formación una configuración estable de músicos, que puede ser la habitual, la última, la más famosa... En un álbum la formación es la relación exacta de los músicos que han participado en él, lo que puede incluir artistas invitados, músicos que no tocan en ciertos temas, músicos que tocan instrumentos que no son sus habituales, etc.

Es decir, que la formación de un álbum depende de los músicos que tocan en él, de los instrumentos y de los temas. Para simplificar, se ha decidido que no consten los instrumentos concretos que ha usado cada músico en un tema, sino que se incluye sólo la lista de los que toca en todo el disco. En cambio, sí se guardará constancia de los temas en los que participa cada músico. Todo esto se gestionará ya en el MLE, con lo que aparecerán varios campos que no estaban en los tags ID3. Esta información se podrá modificar posteriormente desde el RW, añadiendo o quitando músicos de la formación, cambiando sus instrumentos o los temas en los que participan.

Otra información que se podrá incluye el álbum y que puede ser añadida tanto desde el MLE como desde el RW es su portada. Los álbumes se mostrarán dentro del sitio web con una página propia, pero también aparecerán en forma de resumen como parte de la ficha del músico o banda.

Hay un hecho importante en referencia a los álbumes, y es que, como se ha mencionado, los usuarios del sitio web pueden enviar al administrador de la radio música en mp3. Esa música la mayoría de las veces pertenecerá a discos concretos, pero ocasionalmente podrán ser temas que los músicos han grabado en su casa o en conciertos privados, o cuya procedencia no es clara, etc. En ese caso se introducirán como pertenecientes a un disco de título preestablecido, como por ejemplo *Not published* o *Not released*. Será el administrador de la radio el que tome esa decisión.

2.3.3. Instrumentos y formaciones

Generalmente, cada músico del Catálogo tocará, por lo menos, un instrumento. Los instrumentos que toque un artista se verán reflejados en su ficha. Se ha dicho que hay dos tipos de formaciones: la de la banda y la del álbum. Cualquiera de las dos formaciones consiste en una lista de parejas músico – instrumento, indicando si el músico es el líder de

la formación y, en el caso de formaciones de álbum, incluyendo la lista de los temas en los que toca el músico.

En relación con las formaciones, es habitual poner primero los instrumentos de viento (trompeta, saxo alto, saxo tenor, flauta...) y luego los instrumentos armónicos y rítmicos (guitarra, piano, contrabajo y batería). Esto no es una norma, sino un simple convenio, pero puede ser obviado en cualquier momento. Por ejemplo, cuando el líder de la banda es el batería, se suele poner el primero. Por eso las formaciones deberán poder ordenarse como se estime oportuno, tanto en una banda como en un álbum. Lo mismo ocurrirá con los instrumentos que toque un mismo músico.

2.3.4. Temas

Un tema es una obra musical que se interpreta en un disco. Dado que en el jazz es habitual que los músicos toquen y graben temas que han tocado muchos otros músicos (temas que reciben el nombre de *standards*) y que todos conocen, se planteó la posibilidad de introducir esa información como una entidad propia, con sus autores, año de composición y demás, y que cada vez que un músico o banda de barcelonajazzradio.com tocara ese tema se hiciera referencia a esa entidad. De esa forma se podría ver, por ejemplo, qué músicos tienen su propia interpretación del tema, en cuántos álbumes aparece, etc. Pero se desechó la idea debido a que hay temas que son conocidos con diversos nombres, o que están escritos de forma diferente en discos distintos (por ejemplo, *Body and Soul – Body & Soul*, o *Round Midnight – Round About Midnight – 'Round Midnight*, o el tema *Miles* de Miles Davis, que en discos posteriores a su primera grabación aparece como *Milestones*), y se prefiere guardar la información tal y como aparece en el álbum.

En muchos álbumes de jazz, además, se puede encontrar un mismo tema interpretado dos o más veces, ya que las improvisaciones de todas las grabaciones que se hicieron parecieron suficientemente buenas como para no desecharlas. Cuando ocurre eso se suele añadir entre paréntesis un indicador de *toma alternativa* (*Alt. take*), incluso numerados si hay más de dos. Es decir, que son el mismo tema pero también tienen nombres diferentes.

Como todas estas cuestiones dependen de cada caso concreto (del compositor, de la discográfica, de los otros músicos que han tocado el tema...), un Catálogo musical como éste lo único que puede hacer es adaptarse a esta realidad de la mejor forma posible, en lugar de intentar encontrar una solución que no existe.

Por lo tanto, un tema en el contexto de este proyecto será una "interpretación de un tema real en un disco concreto", es decir, que cada tema tendrá una relación directa con un único mp3. Si se desea saber qué artistas han tocado un mismo tema, se podrá usar el buscador el sitio web, utilizando como consulta palabras del título.

Existe aún otra razón más importante para no hacer un tratamiento completo de los temas y mostrarlos en las fichas de los álbumes, y es que no necesariamente todos los temas que pertenezcan a un álbum serán emitidos en la radio. Algunos podrán ser emitidos en el futuro, pero otros tal vez no se emitirán nunca y ni siquiera formarán parte del Catálogo. Si eso ocurriera, como resulta que la inserción de temas se hará directa y automáticamente desde el MLE a partir de los mp3 físicos, se darían casos en que la ficha de un álbum podría mostrar una lista incompleta de los temas que lo forman, lo cual sería más engañoso de cara al usuario que no mostrar ninguno. Tal vez en futuros PFCs se decida solventar este problema de algún otro modo, pero en el ámbito de este proyecto los temas sólo tienen sentido para el MLE, el RW los recoge en su base de datos a efectos de relacionarlos con su álbum y artista en la playlist.

2.3.5. Fotógrafos

Los fotógrafos son entidades especiales del sitio web. Es necesario introducirlos en el RW, ya que deben contar con varios datos, como el nombre, ciudad, año de nacimiento, una fotografía, una pequeña biografía, etc. Pero como se prevé que habrá muy pocos

fotógrafos, y su papel no es el mismo que el de los músicos en la web, y además es posible que no todos deseen que se muestren esos datos, se ha tomado la decisión de no crear una estructura compleja para ellos con datos obligatorios, sino que la información que se almacene tendrá un formato libre, que permita introducir todo lo necesario en cada caso.

2.3.6. Imágenes

En la ficha de un artista, así como en la de un álbum, habrá una fotografía relacionada con cada uno de ellos. Pero una de las funcionalidades que permitirá el sitio web será la de subir más imágenes relacionadas con los artistas. Estas imágenes serán hechas por un fotógrafo y tal vez estén relacionadas con otros tipos de elementos: conciertos, ciudades, etc. El sitio web permitirá de algún modo poder visualizar estas imágenes siguiendo diversos criterios en forma de galerías: todas las imágenes de un mismo artista, todas las de un mismo fotógrafo, etc.

También se incluirán imágenes de las portadas de los álbumes.

2.3.7. Sellos discográficos

Cada álbum estará editado por un sello discográfico, por lo tanto esta información se tendrá que reflejar en el conjunto del sitio web, almacenando los nombres de estos sellos.

2.4. Ámbito de este PFC

Ya se ha explicado que este PFC, junto con otros dos, forma parte de un bloque de acciones destinadas a la remodelación de barcelonajazzradio.com. Ese bloque es, a su vez, la primera parte de un conjunto mayor de modificaciones.

En una primera fase, se confeccionó una lista de requisitos. Como ya se ha mencionado antes, estos requisitos cubrían la funcionalidad de dos grandes etapas. La primera orientada a construir un sitio web rico en contenido vinculado a la programación de la radio y las herramientas necesarias para facilitar la gestión de los mismos, dejando por otra parte preparado el sistema para añadir de una forma ágil funcionalidad de comunidad.

La segunda etapa orientada a incluir en el sitio web funcionalidades de comunidad o web 2.0. En concreto mi PFC cubre el 25-35% de todos los requisitos. Junto con los otros dos PFC's se cubre el 65-75% del total de requisitos del proyecto, suficientes para que la primera etapa o fase del proyecto pueda funcionar correctamente.

De todo lo explicado hasta ahora, este PFC se centrará en la parte del RW (Radio Website), es decir, la implantación del gestor de contenidos y las modificaciones oportunas para adaptarlo a las necesidades del proyecto, lo cual incluye desde un estudio de la base de datos a la creación de uno o más módulos que permitan almacenar el Catálogo musical (artistas, álbumes, temas, imágenes...).

El otro elemento muy importante del PFC es la creación de una arquitectura que permita conectar RW y MLE, esta infraestructura también forma parte de este proyecto.

De los otros dos PFCs, uno se dedicará casi exclusivamente al MLE (Francisco Javier Alonso) y el otro al RW y la Playlist (Iván Marcos).

Los esquemas de las Figura 2.2, Figura 2.3 y Figura 2.4, que hacen referencia a la Figura 2.1, muestran las partes a las que se han dedicado los tres proyectos. Los elementos que quedan fuera de los tres esquemas son aquéllos de los que se encargará el administrador de contenidos o bien son procesos externos.

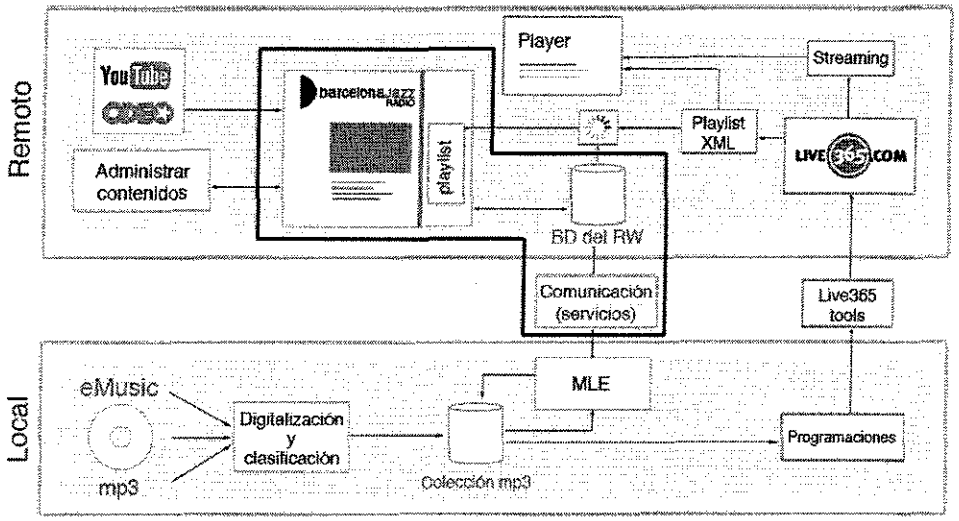


Figura 2.2: PFC José Rodríguez (éste)

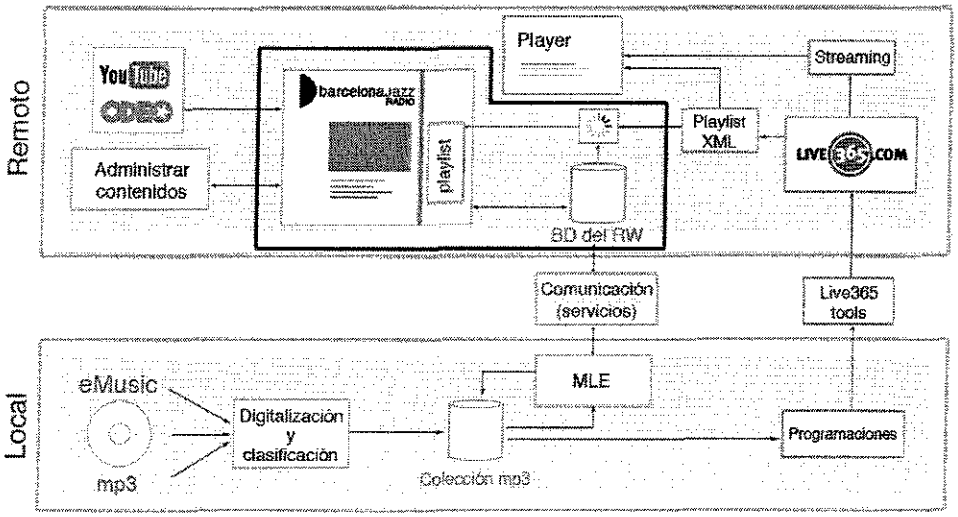


Figura 2.3: PFC Ivan Marcos

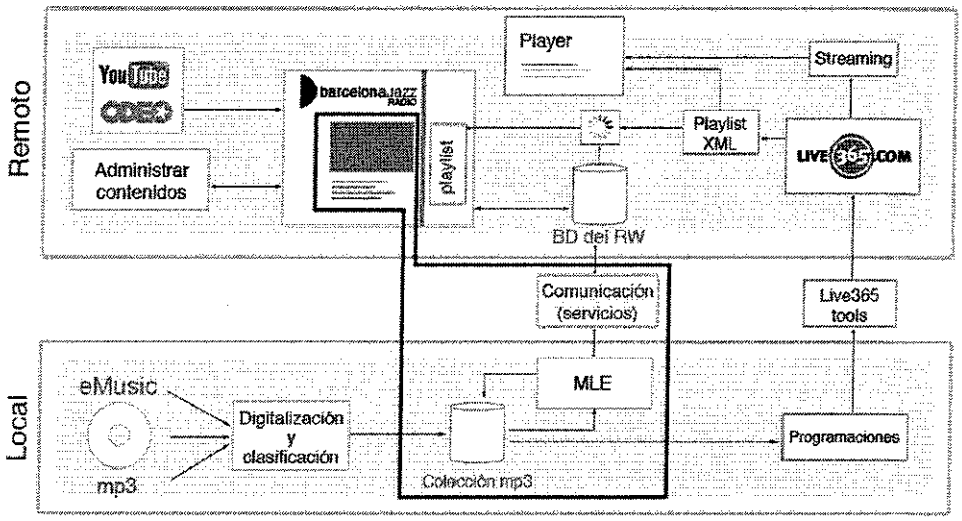


Figura 2.4: PFC Francisco Javier Alonso

Si bien los requisitos de este PFC no tienen nada que ver con la Playlist ni con su acceso a servidores externos, es necesario en todo momento tener una visión general de los requisitos globales del proyecto y una comunicación constante con los otros dos alumnos.

2.5. Objetivos

De todo lo que se ha venido exponiendo más arriba se derivan los objetivos de este PFC que se concretan en:

- Crear un sitio web (RW) que proporcione la posibilidad de mantener un Catálogo de jazz. Es decir, un modelo de datos con artistas, álbumes, temas, imágenes, formaciones, instrumentos, etc.
- Usar un CMS o Sistema Gestor de Contenidos de código abierto para albergar este modelo de datos. Eso requiere ampliar y modificar el CMS básico con módulos hechos por la comunidad de desarrolladores y con otros creados especialmente para el proyecto y que podrán ser devueltos a la comunidad para que sirvan a otros programadores.
- Teniendo en cuenta que el PFC no funciona de manera aislada, trabajar en común con los otros dos PFCs y tomar las decisiones de diseño de forma que faciliten siempre la comunicación entre el MLE y el RW, ya que utilizarán la misma BD.
- Vincular la información del catálogo con el stream de audio de la radio a través de una playlist con enlaces. La playlist será específica para el sitio web y será capaz de mostrar al usuario el tema que está sonando y los anteriores; también debe poder actualizarse automáticamente y contar con algunos parámetros configurables.
- Ofrecer información a los usuarios de la radio cumpliendo dos requisitos:
    - La usabilidad y el diseño gráfico seguirán unas pautas concretas especificadas en un documento sobre navegación y apariencia que se nos facilitó en su momento.
    - El sitio web debe contar con ciertas características que propicien una buena indexación en los buscadores web.

2.6. Metodología usada

2.6.1. Fases del proyecto

Los tres PFCs que se han encargado de realizar el proyecto son independientes, pero a la vez están muy relacionados, por lo que han compartido las distintas fases. Se tratará de explicar aquí estas fases, así como la metodología usada y mi papel en cada una de ellas.

- **Toma de contacto con el proyecto:** Mediante unas reuniones y emails posteriores a éstas quedaron claras las directrices a seguir respecto a las ideas básicas del proyecto.
- **Aprendizaje:** Como ya estaban claros los lenguajes y tecnologías que se utilizarían, comencé por una parte de formación en todas estas herramientas. Se utilizaron desde libros impresos a libros digitalizados, y por supuesto mucho material de Internet: documentación, tutoriales, vídeos, etc.
- **Toma de requisitos:** Los requisitos de este PFC tenían mucho que ver con los de los otros dos, pero eran independientes. Aún así, la decisión de lo que sería tanto mi PFC como la de mis compañeros se hizo en varias reuniones con todos los componentes del equipo de desarrollo. Al final del proceso, redacté un documento de los requerimientos que se deben implementar, luego otro compañero hizo una segunda versión del documento con los cambios que surgieron desde su incorporación. Este documento contiene a parte de los requerimientos a implementar como se ha comentado antes, una explicación detallada del funcionamiento general del sistema.
- **Especificación:** Al haber definidas muchas partes del proyecto, la especificación también tuvo que hacerse de acuerdo con los demás proyectistas, especialmente con Iván Marcos, quien se encargaba de la parte qué más tenía en común con mi proyecto. El modelo conceptual debía partir del mío y se debería ir complementando. Los casos de uso también debían tener en cuenta los de Iván.
- **Diseño:** El diseño también tenía una clara dependencia de los otros proyectos, ya que las distintas capas estaban ya medio elaboradas según el resultado de las reuniones iniciales. Se realizó un diagrama de clases de diseño y para la capa de presentación contábamos con un documento en el que se especificaban muy claramente los aspectos navegacionales y visuales de la web.
- **Implementación:** El código ya se implementó por separado, y cada uno trabajaba en su parte. Sin embargo, en todo momento estuvimos en contacto unos con otros para tomar las decisiones correspondientes de forma común. Al principio, dado que la implementación se estaba compaginando con un aprendizaje sobretodo del funcionamiento de Drupal, trabajamos según la técnica de *peer programming*, que tiene por objetivo mejorar la calidad del código y que es particularmente indicada para acelerar el aprendizaje de las herramientas y lenguajes utilizados. Una vez que se le cogió el método de programar con Drupal y al uso que se hace de PHP, lenguaje en el que está escrito, la programación fue más fluida e independiente.
- **Pruebas:** Las pruebas unitarias se han ido haciendo sobretodo con cada parte finalizada, corrigiendo errores y retocando funcionamientos incorrectos. Las pruebas de integración se hacían pasando las nuevas partes implementadas del servidor de desarrollo al de pruebas y, en él, se realizaban pruebas más generales, en las que también participaba el futuro administrador de los contenidos. Al final, se realizó un documento con las pruebas que se realizaron tanto a RW como a MLE.
- **Documentación:** Incluso en la redacción de la memoria ha habido partes comunes. Algunas de ellas ya estaban bastante encaminadas desde la toma de requerimientos, pero otras las he redactado una vez terminado el proyecto. De

hecho, el grueso de la memoria se ha escrito al final de todo, aprovechando todos los documentos realizados a lo largo de este PFC.

2.6.2. Colaboración y coordinación

Si hay algo que no ha faltado en todo el tiempo que ha durado la realización de este proyecto ha sido comunicación. Y para ello se han usado todo tipo de recursos disponibles.

Ya desde la primera fase del proyecto, que consistía en una etapa de formación en las herramientas a emplear durante el desarrollo (PHP, MySQL, Drupal...), una de las fuentes de recursos que se han utilizado han sido los Grupos de Google. Allí el director del proyecto, Antonio Cañabate, había definido un plan de aprendizaje de varias semanas destinado a formar a alumnos que fuesen a utilizar esas herramientas. Además era un punto de encuentro entre los alumnos donde podíamos preguntar dudas y responder a otras, así como ampliar la documentación o la bibliografía relacionada con el proyecto.

Durante la realización del PFC, además, han sido necesarios no sólo un seguimiento por parte del director del proyecto sino también una coordinación con los demás componentes del equipo de desarrollo.

Para contribuir a ambos objetivos se creó una hoja de cálculo de planificación con una relación detallada de las tareas que debía cumplir cada uno de los que trabajábamos en el proyecto, así como las fechas de entrega estimadas de esas tareas. A medida que se iban terminando se marcaban como finalizadas y se pasaba a la siguiente. Cuando una tarea estaba a medias se ponía el porcentaje de trabajo completado (de forma aproximada). La hoja de cálculo incluía tareas y subtareas y estaba suficientemente elaborada como para poder saber, de una forma eficiente, lo que tenía pendiente cada uno, aquello en lo que estaba trabajando y las dependencias que podía crear en los demás.

Como enviar cada vez el archivo de la hoja de cálculo por email supondría una forma bastante tediosa de trabajar, se utilizó una herramienta, FolderShare, que permite sincronizar de forma transparente los ficheros del disco entre varias personas. Cada vez que alguien modifica un archivo se lanza un proceso de sincronización y el archivo es enviado a todos los demás miembros del grupo de trabajo. Así, no sólo se compartía la hoja de planificación sino cualquier archivo que fuese relevante tener a mano en un momento dado, desde documentos de requisitos a manuales o fragmentos de código. Es una forma sencilla y ágil de estar todos informados y contar con las últimas versiones de documentos importantes. Los problemas de conflictos de escritura los soluciona el mismo programa guardando una copia del archivo con otro nombre en el mismo directorio.

Además del seguimiento general, cada semana el director del proyecto debía recibir por email un informe con explicaciones mínimamente detalladas sobre los avances hechos durante la semana, con una justificación de las decisiones tomadas, los retrasos, los cambios, etc.

Los encuentros en persona también han sido habituales, tanto de forma individual con el director, Antonio Cañabate, como con el equipo al completo. Regularmente también se unía al grupo Josep Mestres, el fundador, DJ y administrador de contenidos de la radio y del sitio web a remodelar, para darnos indicaciones, resolver dudas y orientarnos.

Cuando no era posible reunirnos físicamente, ya sea por falta de tiempo o porque la reunión se preveía corta y no justificaba el desplazamiento, se han realizado encuentros a través del servicio de VoIP Skype, que también ha estado presente desde el principio como servicio de mensajería instantánea (IM). Y aún otra herramienta que ha sido de utilidad a la hora de dar explicaciones concretas a distancia es TeamViewer, un programa que permite compartir el escritorio y controlar la máquina de forma remota.

2.6.3. Método de desarrollo en equipo y gestión de versiones

La web www.barcelonajazzradio.com debía seguir operativa mientras durara la elaboración del proyecto. Por lo tanto, desde un primer momento se contrató un servicio de hosting en el que poder desarrollar el proyecto y al que todos podíamos acceder por FTP. Se utilizaron varias versiones, tanto para desarrollo como para pruebas, y en algunas ocasiones se trabajó en un servidor local y se subieron los cambios posteriormente.

Al tratarse de un proyecto de varias personas nos planteamos la posibilidad de utilizar CVS (Concurrent Versions System) como herramienta de trabajo. CVS es un sistema de control de versiones muy utilizado en software libre. No sólo permite a un grupo de personas trabajar sobre los mismos ficheros remotamente evitando modificaciones accidentales sino que además mantiene un registro de todos los cambios hechos a un archivo. Pronto nos dimos cuenta de que no era necesario montar una infraestructura como esa cuando en realidad sólo dos personas iban a trabajar en lo mismo (RW) y, a partir de un momento dado ni siquiera iban a tocar los mismos ficheros. Al estar continuamente comunicados vía Skype, era mucho más sencillo avisarnos cuando fuésemos a hacer algún cambio potencialmente peligroso.

Cada día se hacía una copia de seguridad de todo el contenido del servidor, así como de las bases de datos, y esa copia era enviada por ftp a uno de nuestros ordenadores. De esa forma, si alguien cometía un error grave o se perdían datos importantes, era cuestión de minutos volver a tenerlo todo funcionando con un contenido bastante actualizado.

En cuanto al idioma de trabajo, hay que indicar que en el código del RW se ha usado predominantemente el inglés, por dos razones: primero, el sitio web tiene como idioma principal el inglés, así que era más cómodo usar palabras como *artist* o *band* si luego eran ésas las palabras que se leerían en el contenido y en la interfaz; la otra razón es que se trabajó pensando en la posible incorporación de los módulos creados al repositorio de la comunidad del CMS, comunidad que agradecería enormemente que estuviesen redactados en ese idioma.

2.7. Organización de la memoria

Como se acaba de comentar, el idioma de la página a remodelar es el inglés, lo cual nos ha obligado a trabajar usando ese idioma como base para la nomenclatura: playlist, artists, bands, catalogue, featured artists... Para mantener la coherencia con el proyecto se utilizará frecuentemente este vocabulario a lo largo de la memoria. Aunque se explicará cada concepto y la mayoría de palabras son de fácil comprensión, los términos que presenten dificultades estarán definidos en el glosario, en la página 213.

La memoria consta de los siguientes capítulos:

1. Una **introducción**, donde se ha dado una visión muy general del proyecto, justificando las causas de una remodelación y actualización de un sitio web preexistente para convertirlo en algo totalmente diferente.
2. La descripción del **entorno del PFC**. En ella se han explicado detalladamente las partes del sistema completo y se ha delimitado el ámbito de este proyecto en concreto. También se han marcado los objetivos y se ha explicado la metodología seguida durante el desarrollo. Finalmente se está hablando de la organización de este documento, que no detallaré aquí por un problema obvio de recursividad que provocaría un desbordamiento de pila y una sobrecarga de la *memoria.*
3. El **análisis de requisitos** es la definición precisa de lo que debe cumplir la aplicación para satisfacer sus objetivos. Se han separado los objetivos en más prioritarios y menos prioritarios para poder ajustar mejor los tiempos en caso de una planificación no del todo acertada.

4. En el capítulo de **planificación y costes** se describe la planificación inicial obtenida del análisis de requisitos y se hace una estimación de los costes del mismo. También se hace un cálculo aproximado de los costes del proyecto.
5. **Especificación**. En esta etapa se define *qué* se hará independientemente del *cómo* se acabe haciendo.
6. El capítulo de **diseño** habla sobre las distintas arquitecturas usadas en Ingeniería del Software y elige una, para luego desglosar el sistema en sus partes.
7. **Sistemas Gestores de Contenidos**. Qué son, cuáles hay y por qué se ha elegido uno de ellos. Se da una explicación general del funcionamiento del gestor de contenidos y de lo que hay que hacer para añadir modificaciones o código propio.
8. El capítulo de **implementación** empieza explicando cómo se trabaja con Drupal y luego hace un repaso de los diferentes bloques de código que se han desarrollado.
9. **Juegos de prueba** desglosados por bloques de funcionalidad.
10. Los **manuales** incluyen el de instalación de Drupal, el de instalación de los módulos necesarios y el manual de usuario.
11. Finalmente, se extraen unas **conclusiones** del conjunto del proyecto.

Además, se cuenta con:

- **Anexo I: Bibliografía**
- **Anexo II: Glosario**
- **Anexo III: Diagrama de clases completo**

3.Análisis de requisitos

En este capítulo se hablará por encima de lo que es el Análisis de Requisitos de un sistema software y se explicará el funcionamiento general del sistema que se trata en este proyecto, así como los tipos de usuario que participarán en él. Una vez aclarados los conceptos básicos se procederá a listar la relación de requisitos que debe cumplir junto con los usuarios a los que se asignará cada uno.

3.1. Introducción

El Análisis de Requerimientos, aplicado a un sistema de software, es una disciplina que pretende identificar y clasificar las condiciones que debe cumplir dicho sistema, así como estudiar las relaciones que existen entre ellas. Estas condiciones son los requisitos.

De las muchas definiciones que recibe la palabra *requisito*, a continuación se presenta la que aparece en el glosario de la IEEE:

1. Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo.
2. Una condición o capacidad que debe estar presente en un sistema o en un componente de un sistema para satisfacer un contrato estándar, especificación u otro documento formal.
3. Una representación documentada de una condición o capacidad como en *1* o en *2.*

En siguientes subapartados se estudiarán esas necesidades en relación con este proyecto para acabar representándolas de forma documentada.

El Análisis de Requisitos se puede dividir en cuatro grandes fases:

- Obtener información acerca de lo que los usuarios desean: el sistema debe, por encima de todo, satisfacer las necesidades del usuario o cliente. Para determinar estas necesidades se hacen entrevistas y se recopila información. No siempre es fácil llegar a identificar lo que el usuario quiere, y sus necesidades suelen cambiar con el tiempo, por lo que es aconsejable conservar documentos de esas reuniones.
- Clasificar los deseos del cliente: para poder estructurar los requisitos hay que separar las necesidades del cliente en varios bloques, ya que no todos los requisitos tendrán la misma importancia ni deberán ser tratados de la misma forma.
- Establecer una jerarquía: se analiza el sistema para separar sus elementos en niveles más genéricos y más específicos. En el conjunto de estos niveles hay que

ubicar los distintos requisitos. Cuanto más arriba esté un requisito dentro de la jerarquía, más costosa será su implementación.

- Especificar los requisitos de acuerdo a la audiencia a la que van dirigidos: hay varias formas de plasmar los requisitos, algunas más formales y otras en un lenguaje más natural. Con las primeras se corre el riesgo de que aquéllos a los que van dirigidos los requisitos tengan problemas para comprenderlos; las segundas tienen el peligro de perder precisión y de aumentar la ambigüedad. El lenguaje UML incorpora los *casos de uso*, que son representaciones abstractas acompañadas de una descripción textual.

Los requisitos deben ser revisados y aprobados por el cliente y por el equipo del proyecto, y tanto los requisitos técnicos como los no técnicos deben ser tenidos en cuenta como un mismo conjunto. Si cambia un requisito técnico, como puede ser extender una cierta funcionalidad, los requisitos no técnicos como la calidad, el coste o los plazos de entrega también variarán inevitablemente. Toda modificación en algún requisito implica un nuevo ajuste en los planes ya aprobados con anterioridad.

Existen dos tipos de requisitos: funcionales y no funcionales.

- Los ***requisitos funcionales*** describen cada uno de los procesos, acciones, y cálculos que debe llevar a cabo el sistema, así como cada una de las salidas que deben poderse obtener.
- Los ***requisitos no funcionales*** no se refieren directamente a las funciones específicas que brinda el sistema, sino a sus propiedades emergentes: fiabilidad, tiempo de respuesta, capacidad de almacenamiento, etc. Definen las restricciones del sistema: capacidad de los dispositivos de entrada/salida, representación de datos que se utilicen en los interfaces... También definen las características que indican cómo el sistema debe realizar su trabajo, como pueden ser la eficiencia, el hardware o el software necesarios.

Un conjunto de requisitos en estado de madurez debe presentar una serie de características, tanto individualmente como en grupo. A continuación se presentan las más importantes:

- ***Necesario:*** Un requisito es necesario si su omisión provoca una deficiencia en el sistema a construir, así como si su capacidad, características físicas o factor de calidad no pueden ser reemplazados por otras capacidades del producto o del proceso.
- ***Conciso:*** Un requisito es conciso si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en un futuro.
- ***Completo:*** Un requisito está completo si no necesita ampliar detalles en su redacción, es decir, si se proporciona la información suficiente para su comprensión.
- ***Consistente:*** Un requisito es consistente si no es contradictorio con otro requisito.
- ***No ambiguo:*** Se dice que un requisito es no ambiguo cuando tiene una sola interpretación. El lenguaje usado en su definición no debe causar confusiones al lector.
- ***Verificable:*** Un requisito será verificable si puede ser cuantificado de manera que permita hacer uso de los siguientes métodos de verificación: inspección, análisis, demostración y pruebas.

Es importante mencionar el concepto de *Ingeniería de Requisitos* (IR), que es un enfoque sistemático para recolectar, organizar y documentar los requisitos del sistema; es también el proceso que establece y mantiene los acuerdos sobre los cambios de requisitos entre los clientes y el equipo del proyecto.

Los principales beneficios que se obtienen de la Ingeniería de Requisitos son:

- Permite gestionar las necesidades de un proyecto de forma estructurada: Cada actividad de la IR consiste en una serie de pasos organizados y bien definidos.
- Mejora la capacidad de predecir cronogramas de proyectos, así como sus resultados: La IR proporciona un punto de partida para controles subsecuentes y actividades de mantenimiento, tales como la estimación de costes, de tiempo y de recursos necesarios.
- Disminuye los costes y retrasos del proyecto: Muchos estudios han demostrado que reparar errores por un mal desarrollo no descubierto a tiempo es algo sumamente caro.
- Mejora la calidad del software: La calidad en el software tiene que ver con cumplir un conjunto de requisitos (funcionalidad, facilidad de uso, confiabilidad, desempeño, etc.)
- Mejora la comunicación entre equipos: La especificación de requisitos representa una forma de consenso entre clientes y desarrolladores. Si este consenso no ocurre, el proyecto no será exitoso.
- Evita rechazos de usuarios finales: La IR obliga al cliente a considerar sus requisitos cuidadosamente y revisarlos dentro del marco del problema, por lo que se le involucra durante todo el desarrollo del proyecto.

3.2. Requisitos funcionales

3.2.1. Lista de requisitos RW

Como se ha dicho, los requisitos del proyecto global abarcan mucho más que la responsabilidad de este PFC, y los tres PFCs que se han hecho a la vez tienen tareas tanto del MLE (Music Library Editor) como del RW (Radio Website).

Los requisitos de este PFC, de los que hablaremos aquí, hacen toda referencia exclusivamente al RW y a uno de MLE (muy importante por cierto). Éstos se han llevado a cabo por mí y por otro proyectista y, como ya se ha mencionado, al principio trabajamos en pareja. Por ello, se indica los que se implementaron en común y los que implementé en solitario.

Dentro de la lista de requisitos hay algunos que tienen más importancia que otros, ya sea porque su funcionalidad es más necesaria para el funcionamiento del sitio web o porque se ha decidido que era un reto interesante para el PFC. Dado que el tiempo para realizar el proyecto es limitado y que las estimaciones temporales y de cantidad de trabajo nunca son del todo precisas, se ha decidido fijar un umbral de importancia. Los requisitos que estén por encima del umbral serán de realización obligatoria, mientras que los otros servirán para ajustar el tiempo total del proyecto. La numeración será la siguiente:

1. **Requisitos prioritarios**
2. **Requisitos secundarios**

Para los de prioridad 2 se han dejado funcionalidades menos imprescindibles o con características más propias de la Web 2.0. Como se ha explicado al principio del capítulo 2, estas funcionalidades se implementarán en PFCs posteriores. Si se ha estimado más esfuerzo del que luego se acaba realizando, se podrán hacer más tareas de prioridad 2, y si la estimación ha sido a la baja, se harán menos.

La Tabla 3.1 muestra el conjunto de requisitos funcionales del sistema. La primera columna indica el código que se ha asociado a la tarea, de modo que por el nombre se pueda identificar el requisito general al que pertenece. Estos grandes bloques son los siguientes:

- ***CON***: Contenido del sitio web.

- ***USR***: Mantenimiento de usuarios.
- ***PRT***: Participación de los usuarios en el sitio web.
- ***CAT***: Catálogo musical.

La segunda columna especifica el nombre del requisito, y la tercera, su prioridad; el resto de columnas indica los perfiles de usuario que tienen capacidad para ejecutar el requisito.

Hay que aclarar que algunos de los requisitos se pueden descomponer en varias tareas y además, dada la forma en la que el CMS y sus módulos gestionan ciertos elementos, después de trabajar en la resolución de un requisito partes de otros podrían quedar automáticamente resueltas. Esto hace que exista un solapamiento difícil de especificar en una tabla, ya que algunos de estos requisitos pueden haber sido hechos entre los dos proyectistas que teníamos como objetivo principal el RW.

Los detalles sobre las partes realizadas por mí quedarán más claros en los capítulos de implementación, pero para reflejarlo de la mejor manera posible en la tabla, los requisitos que están en **negrita** son aquéllos en los que he participado de forma suficientemente activa como para tenerlo en cuenta y los que además están en ***cursiva*** son los requisitos realizados completamente por mí. Sin embargo, y hasta cierto punto, dada la estrecha colaboración con el otro proyectista, en mucha menor medida puedo decir que he participado en el diseño e implementación de prácticamente TODOS los requisitos que se han completado de RW.

Código	Requisito	PR	Usuarios			
			ADM	MUS	INV	SIS
CON-001	***Insertar y editar contenido (entradas de blog)***	1	X			
CON-002	***Categorización de contenidos***	1	X			
CON-003	***Búsqueda de contenidos***	1	X	X	X	
CON-004	***Archivo de contenidos***	1	X	X	X	X
USR-001	***Creación de cuenta de usuario***	1	X			
USR-002	Administración de usuarios	1	X			
USR-003	***Autentificación (login)***	1	X	X		
USR-004	Edición del perfil de usuario	1	X	X		
USR-005	Cambio de idioma de la IU	2	X	X	X	
USR-006	Idioma según servicios de GeoIP	2				X
USR-007	Diferentes idiomas para los contenidos	2	X			
PRT-001	Creación y administración de foros	2	X			
PRT-002	Participación en foros	2	X	X	X	
PRT-003	***Admin. listas de distribución para newsletters***	1	X			
PRT-004	***Creación, envío y mantenimiento de newsletters***	1	X			
PRT-005	Envío de música al administrador	2		X	X	
PRT-006	Envío de música al músico	2	X	X	X	
PRT-007	**Subscripción RSS a contenidos**	1	X	X	X	

Código	Requisito	PR	Usuarios ADM	MUS	INV	SIS
	dinámicos					
PRT-008	Mensaje resumen de alertas al administrador	2				X
CAT-001	***Creación y edición de artista***	1	X	X		
CAT-002	**Administrar enlaces útiles de un artista**	1	X	X		
CAT-003	***Subir o enlazar imágenes de un artista***	1	X	X		
CAT-004	Subir o enlazar partituras de un artista	2	X	X		
CAT-005	Subir o enlazar temas de un artista (podcast)	2	X	X		
CAT-006	**Visualización de artista**	1	X	X	X	
CAT-007	***Creación y edición de álbum***	1	X			
CAT-008	**Visualización de álbum**	1	X	X	X	
CAT-009	Featured Artists	1	X	X	X	X
PLT-001	Playlist	1	X	X	X	X

Tabla 3.1: Lista de requisitos funcionales de RW

3.2.2. Lista de requisitos MLE

De forma análoga al capítulo anterior, a continuación, se muestra una lista con los requerimientos de MLE, éstos estarán marcados de forma idéntica al anterior (**negrita**, he participado y ***cursiva*** los he implementado yo mismo).La lista de requisitos funcionales de MLE es más larga lo que para no extender este documento innecesariamente se ha creído conveniente solo exponer los que tienen que ver con este PFC.

Código	Requisito	PR	Usuarios ADM	MUS	INV	SIS
SYNC-001	Comunicación entre servicios y MLE	1				X
SYNC-002	***Comunicación entre los servicios y la BD Drupal***	1				X

Tabla 3.2: Lista de requisitos funcionales de MLE

3.2.3. Descripción de los requisitos

La descripción textual de cada uno de los requisitos mostrados en la Tabla 3.1 y Tabla 3.2, junto con sus entradas y salidas, se especifica a continuación.

CON-001: Insertar y editar contenido (entradas de blog)

Prioridad: 1

Usuarios: ADM

Descripción: El administrador tiene la capacidad de insertar, modificar y eliminar contenido del sitio web. Este contenido tiene forma de entrada de blog, en el sentido de que consta de un título, el cuerpo del mensaje, la fecha, etc. y puede recibir comentarios por parte de otros usuarios. Estos contenidos podrán ser, por ejemplo, *Artista del mes* o *Calendario de conciertos*, o bien una reseña sobre un disco, etc.

Los posts del blog podrán pertenecer a un artista, en el sentido de que su contenido se referirá a ellos, con lo que aparecerán en la página de dicho artista. Ocasionalmente, el administrador podrá decidir "promocionar" estos posts a la página principal y eliminarlos de ella cuando lo desee.

El administrador podrá ordenar estos posts según los criterios que estime oportuno.

Entrada: El texto o código HTML introducido por el administrador de contenidos a través de un formulario web. En este formulario también podrá especificar la posición del contenido en relación a otros elementos de la página mediante un *peso*, así como indicar en qué categorías o partes del sitio web se debe mostrar. El uso de código HTML le permitirá incluir enlaces e incrustar objetos con herramientas de sitios ajenos (vídeos, mapas, etc.).

Salida: El artículo del site, en forma de post de blog, en el orden especificado y en las partes de la web que haya indicado.

CON-002: Categorización de contenidos

Prioridad: 1

Usuarios: ADM

Descripción: Los contenidos del site se deben poder clasificar en las categorías o *taxonomías* que estime oportuno el administrador. El objetivo es facilitar al usuario la navegación por contenidos relacionados.

Entrada: Las taxonomías generales del site, creadas por el administrador, y la asignación manual de cada fragmento de contenido dentro las clasificaciones correspondientes.

Salida: Cada elemento del contenido clasificado en las categorías que requiera.

CON-003: Búsqueda de contenidos

Prioridad: 1

Usuarios: ADM, MUS, INV

Descripción: Además del uso de las taxonomías, los contenidos deberán ser accesibles a través de un formulario de búsqueda. Este formulario estará visible en todas las páginas, de modo que a partir de cualquier información que se lea en la web se pueda generar una búsqueda relacionada sin tener que acceder a una página específica de búsqueda.

Entrada: Una consulta escrita en un cuadro de búsqueda.

Salida: Una página con la lista de resultados de búsqueda relacionados con las palabras clave impuestas por el usuario en el formulario de búsqueda donde se podrán filtrar los resultados.

CON-004: Archivo de contenidos

Prioridad: 1

Usuarios: ADM, MUS, INV, SIS

Descripción: Los contenidos antiguos tienen que ser archivados de forma automática por parte del sistema, además de ser accesibles para los usuarios. El administrador determinará la mejor forma para el acceso a estos contenidos (acceso por fechas, búsquedas...).

Entrada: La configuración de acceso especificada por el administrador y el propio contenido.

Salida: El contenido del sitio web archivado y las páginas que lo muestran.

USR-001: Creación de cuenta de usuario

Prioridad: 1

Usuarios: ADM

Descripción: Sólo el administrador puede crear usuarios. A estos usuarios se les asignará un perfil de *músico* (en el futuro podrá haber más perfiles) y en el mismo momento de creación se podrá indicar la ficha del artista con la que está relacionado. No existirá un proceso de registro o de solicitud de cuenta (a no ser el envío de un email directamente al administrador) (Aunque en un futuro podrá haberlo).

Entrada: Los datos del usuario introducidos por el administrador en el back office del web.

Salida: El nuevo usuario queda registrado en la web con un perfil determinado y es informado de ello vía email, recibiendo un nombre de usuario y una contraseña que podrá modificar posteriormente.

USR-002: Administración de usuarios

Prioridad: 1

Usuarios: ADM

Descripción: El administrador puede dar de alta y de baja usuarios, así como modificar sus datos. Los propios usuarios tienen un cierto acceso limitado sobre algunos de esos datos, incluyendo su idioma por defecto y posiblemente la opción de mostrar si están conectados en un momento dado. El administrador puede también bloquear o desbloquear cuentas, cambiarles el perfil, modificar esos perfiles o crear nuevos, así como añadir y quitar permisos de acceso a zonas determinadas del sitio web.

Entrada: Modificaciones hechas por el administrador desde la sección de administración de usuarios en el back office de la web.

Salida: Los perfiles y/o las cuentas de usuario modificados según las instrucciones del administrador.

USR-003: Autentificación (login)

Prioridad: 1

Usuarios: ADM, MUS

Descripción: Todo usuario, ya sea un músico o el propio administrador debe poder iniciar sesión para acceder a la zona privada de la comunidad web, lo que incluirá más o menos privilegios en función de los permisos de su perfil.

Entrada: Introducción de los datos en el formulario de acceso por parte del usuario. La interfaz debe permitir iniciar sesión.

Salida: Una vez iniciada la sesión, el usuario queda dentro de la zona privada de la comunidad, de la que vuelve a salir al hacer logout.

USR-004: Edición del perfil de usuario

Prioridad: 1

Usuarios: ADM, MUS

Descripción: Todo usuario registrado (músico) podrá acceder a su perfil de usuario, donde podrá realizar diversas acciones. Entre ellas están: modificar sus datos de contacto, la elección de su idioma por defecto, cambio de contraseña, suscripción o baja del newsletter, etc. El administrador podrá modificar las fichas de artista a las que tiene acceso el usuario.

Entrada: Los datos introducidos por el usuario o el administrador en el perfil de usuario.

Salida: La ficha de usuario modificada según los cambios realizados por él mismo o por el administrador.

USR-005: Cambio de idioma de la IU

Prioridad: 2

Usuarios: ADM, MUS, INV

Descripción: Cualquier usuario de la web, aunque no esté registrado, puede cambiar el idioma de la interfaz, probablemente con enlaces en forma de banderas. Por defecto, el sistema tendrá una prioridad de lenguajes: inglés, catalán y castellano.

Entrada: El idioma del usuario mediante un control web.

Salida: El idioma de la interfaz del sitio web se verá modificado si el idioma está disponible. Los tres anteriores siempre estarán disponibles.

USR-006: Idioma según servicios de GeoIP

Prioridad: 2

Usuarios: SIS

Descripción: Si se dispone de un servicio GeoIP, se detectará automáticamente el idioma del usuario y se adaptará la interfaz de usuario a ese idioma.

Entrada: La IP del visitante.

Salida: El idioma de la interfaz se verá modificado si el idioma está disponible.

USR-007: Diferentes idiomas para los contenidos

Prioridad: 2

Usuarios: ADM

Descripción: Se implementará una funcionalidad que permita al administrador introducir el contenido en diferentes idiomas para que pueda ser mostrado al usuario final en el idioma que haya elegido.

Entrada: El contenido introducido en varios idiomas (esta traducción la debe hacer el administrador manualmente).

Salida: El contenido en el idioma que especifique el visitante.

PRT-001: Creación y administración de foros

Prioridad: 2

Usuarios: ADM

Descripción: El sitio web contará con un foro que será gestionado por el administrador, tanto en la moderación como en su creación y mantenimiento, lo que incluye bloqueo de usuarios, creación de subforos accesibles sólo a usuarios registrados, etc. Los foros podrán incluir filtros por palabras y funcionalidades similares. Podrán existir foros privados sólo para usuarios registrados.

Entrada: Opciones de configuración insertados en el back office de la web.

Salida: Foro con las funcionalidades típicas y con las opciones especificadas por el administrador.

PRT-002: Participación en foros

Prioridad: 2

Usuarios: ADM, MUS, INV

Descripción: Todos los usuarios del sitio web podrán participar en los foros. Para ello no hará falta registro, sino que se seguirá el modelo típico de introducción de comentarios en blogs: el usuario introduce, además de su comentario, un nick, un email, una palabra antispam o *captcha* (ya que el acceso será libre) y, opcionalmente, un sitio web. En caso de que el usuario esté registrado (un músico) y haya iniciado sesión, esos datos se rellenarán automáticamente. Habrá que hacer notar de algún modo qué usuarios, de entre los participantes, están registrados. También podrá incluir algún sistema para que los mismos usuarios puedan avisar de contenido inapropiado.

Entrada: Datos de los comentarios del foro, introducidos libremente en el formulario web. Incluyen un captcha y otros sistemas para evitar spam.

Salida: Los comentarios visibles en el foro por otros usuarios.

PRT-003: Administrar listas de distribución para newsletters

Prioridad: 1

Usuarios: ADM

Descripción: El administrador ha de poder crear y modificar listas de emails para el envío de newsletters. Los propios usuarios podrán añadir su correo a estas listas de distribución y darse de baja de ellas cuando quieran, pero el sistema debe proporcionar al administrador la posibilidad de añadir y quitar direcciones de esas listas. Podrán existir varias listas, dependiendo de si se quiere que una cierta información esté restringida a un grupo reducido de usuarios (por ejemplo, los registrados).

Entrada: Las direcciones de email introducidas por los usuarios con las modificaciones que el administrador quiera hacer en la lista completa a través de un formulario web.

Salida: La lista de distribución, que usará el administrador posteriormente para enviar emails con noticias a los usuarios que lo soliciten.

PRT-004: Creación, envío y mantenimiento de newsletters

Prioridad: 1

Usuarios: ADM

Descripción: El administrador contará con la posibilidad de redactar newsletters y enviarlos a los usuarios que lo hayan solicitado a través de una lista de distribución ya creada. Debe poder también archivar los diferentes números de una newsletter para tener el histórico y para que sigan siendo accesibles a los usuarios.

Entrada: Contenido de las newsletters y listas de distribución a las que enviarlo. Opciones de configuración por parte del administrador.

Salida: La newsletter llegará a los usuarios que lo hayan solicitado y se podrá consultar desde la web.

PRT-005: Envío de música al administrador

Prioridad: 2

Usuarios: MUS, INV

Descripción: Es habitual que los oyentes que son también músicos deseen proponer al administrador de contenidos temas o álbumes para ser programados en la radio. Se

facilitará funcionalidad para permitir a cualquier usuario hacer una propuesta que permita hacer llegar un archivo (en formato mp3, zip u otros) o una simple petición al administrador. El fichero podrá ser enviado o se podrá proporcionar un enlace a éste. El administrador tendrá la posibilidad de gestionar dichas propuestas: descargarse los ficheros, contestar al usuario si proporciona un email, eliminar o archivar la propuesta, etc.

Posteriormente, el administrador procesará todas las peticiones y decidirá si introduce esa música en la lista de reproducción de la radio, así como la información en el Catálogo.

Entrada: Formulario web rellenado por el usuario con datos como: nombre, descripción o mensaje, formación y enlaces a los archivos mp3.

Salida: Un email diario con la lista de propuestas que será remitido al administrador.

PRT-006: Envío de música al músico

Prioridad: 2

Usuarios: ADM, MUS, INV

Descripción: Los oyentes de la radio en muchos casos serán también músicos de cualquier lugar del mundo. En un momento dado pueden querer interpretar un tema que un artista tiene en el catálogo de Barcelona Jazz Radio y compartir su versión con el artista. Para ello se proporcionará la posibilidad de asociar un contenido de tipo audio a alguno de los siguientes contenidos: artistas, temas y partituras. Esta asociación deberá incluir el enlace al archivo mp3 (u otros formatos) que el usuario deberá introducir. No se dará soporte de hosting —tal vez en el futuro con cuentas Premium—. Existirá también la posibilidad de notificar enlaces rotos, así como de mantenimiento por parte del administrador.

Entrada: Datos de la propuesta de envío de música a un músico, rellenando un formulario web.

Salida: La asociación de un archivo de música, que será accesible por el músico, ya que estará asociado a su perfil.

PRT-007: Subscripción RSS a contenidos dinámicos

Prioridad: 1

Usuarios: ADM, MUS, INV

Descripción: El sitio web contará con RSS de los posts, que se convertirán en un archivo XML para que cualquiera pueda acceder a ellas o sindicar el RSS en su propia página web.

Entrada: El conjunto de posts.

Salida: El XML generado a partir de los posts y el acceso a éste.

PRT-008: Mensaje resumen de alertas al administrador

Prioridad: 2

Usuarios: SIS

Descripción: Cada cierto tiempo (configurable) el administrador debe recibir resúmenes vía email sobre la actividad o estadísticas del sitio web. Para ello se configurará previamente la funcionalidad.

Entrada: Los datos de configuración (dirección de email, tipo de resumen, frecuencia...) introducidos en un formulario web.

Salida: Un email enviado a la cuenta que decida el administrador cada vez que se cumpla el tiempo estipulado. Este email contendrá datos actualizados con los contenidos o estadísticas que se hayan indicado.

CAT-001: Creación y edición de artista

Prioridad: 1

Usuarios: ADM, MUS

Descripción: Los artistas del Catálogo (músicos y bandas) son creados por el administrador, que luego podrá dar acceso a los usuarios músicos que crea oportuno para que hagan ciertas modificaciones sobre las fichas. La ficha de un músico incluirá unos datos básicos, una fotografía, los instrumentos que toca, información relevante en Internet, etc. Algunos de los datos de los artistas podrán ser modificados desde el MLE, que podrá acceder a ellos y crear nuevos. La ficha de un artista podrá estar creada pero no publicada si el administrador lo decide así, por ejemplo por no tener suficientes datos para mostrar.

Entrada: Datos del artista introducidos en un formulario web del back office (algunos campos serán obligatorios) para su creación o modificación.

Salida: La ficha del artista creada o modificada a partir de esos datos.

CAT-002: Administrar enlaces útiles de un artista

Prioridad: 1

Usuarios: ADM, MUS

Descripción: Según lo especificado en el requisito CAT-001, la ficha del artista tendrá enlaces de interés a páginas desde las que se pueda complementar la información del músico o banda. El número de enlaces será ilimitado y el propio formulario deberá permitir la creación, edición y supresión de éstos. Cada enlace contará con una breve descripción (por ejemplo, *Entrevista al músico X para la revista Y*).

Entrada: Modificación de la parte de la ficha del artista relacionada con los enlaces.

Salida: Los enlaces introducidos en la ficha quedan asociados a la ficha del artista.

CAT-003: Subir o enlazar imágenes de un artista

Prioridad: 1

Usuarios: ADM, MUS

Descripción: Un artista tendrá asociada una serie de fotografías en las que aparezca. Éstas se almacenarán en algún tipo de galería conjunta, de modo que se pueda acceder a ellas a través de las fichas de los artistas así como por otros medios. La web debe proporcionar un formulario desde el que poder subir las imágenes y a la vez enlazarlas a uno o más artistas.

Entrada: Conjunto de imágenes a incluir y artista al que enlazarlas.

Salida: Las imágenes estarán asociadas a artistas y serán visibles por los usuarios.

CAT-004: Subir o enlazar partituras de un artista

Prioridad: 2

Usuarios: ADM, MUS

Descripción: Dada una parte del catálogo como puede ser un tema, el administrador o el músico podrán incluir partituras para que los usuarios las descarguen y las toquen con sus instrumentos. Dichas partituras quedarían relacionadas con el artista y/o con un tema.

Entrada: Una partitura en algún formato binario y un artista o tema.

Salida: Ciertas partes del Catálogo tendrán asociadas partituras que se podrán descargar.

CAT-005: Subir o enlazar temas de un artista (podcast)

Prioridad: 2

Usuarios: ADM, MUS

Descripción: Esta funcionalidad es análoga a la de las partituras (CAT-004), con la diferencia de que el formato será un mp3 (o bien un enlace al archivo).

Entrada: Uno o varios temas mp3 o sus respectivos links.

Salida: Ciertas partes del catálogo tendrán asociados mp3 disponibles para ser escuchados o descargados.

CAT-006: Visualización de artista

Prioridad: 1

Usuarios: ADM, MUS, INV

Descripción: Debe establecerse la forma de visualizar los datos de la ficha del artista, donde se expone información relacionada con él. El acceso a la ficha se podrá hacer desde varios sitios de la web (desde una lista de músicos, desde la ficha de un álbum, desde un post que lo enlace...). Habrá una fotografía, una pequeña descripción y la discografía del artista que suena en Barcelona Jazz Radio, así como una lista de posts relacionados. Dependiendo de si el artista es de tipo músico o banda se mostrará la lista de instrumentos que toca o la formación habitual, respectivamente.

Entrada: El acceso a la ficha del artista por parte del usuario.

Salida: La ficha del artista con la vista que se deba mostrar.

CAT-007: Creación y edición de álbum

Prioridad: 1

Usuarios: ADM

Descripción: El administrador creará los álbumes del artista y los asociará a éste. El álbum contendrá información como el año de edición, el sello discográfico, la duración, la lista de temas, la imagen de la portada, etc. La creación y modificación de álbumes se podrá hacer también desde el MLE.

Entrada: Datos introducidos en un formulario web del back office.

Salida: El álbum queda creado o modificado según los datos del formulario, y quedará asociado al artista.

CAT-008: Visualización de álbum

Prioridad: 1

Usuarios: ADM, MUS, INV

Descripción: Debe existir la forma de visualizar la información almacenada de un álbum: título, portada, lista de temas ordenados, año de edición, enlaces relacionados con los artistas... El acceso se hará desde distintos sitios de la web, como la ficha de un artista, un post relacionado...

Entrada: El acceso a la ficha del álbum por parte del usuario.

Salida: La ficha del álbum.

CAT-009: Featured Artists

Prioridad: 1

Usuarios: ADM, MUS, INV, SIS

Descripción: Desde la página principal se podrá acceder a una lista de todos los artistas que existen en el Catálogo ordenados alfabéticamente y separados por letra. Si estos artistas tienen una ficha activa, se podrá visualizar a través de un enlace en el propio nombre del artista. Opcionalmente, y si el músico así lo ha decidido, se podrá mostrar de alguna forma si el usuario está online en un momento dado.

Entrada: Acceso a la página con la lista de artistas, que será generada automáticamente por el sistema.

Salida: Visualización de la página *Featured Artists* por parte de todos los usuarios, con acceso a sus fichas de artista.

PLT-001: Playlist

Prioridad: 1

Usuarios: ADM, MUS, INV, SIS

Descripción: En este caso, la playlist se refiere a la zona del sitio web en la que se muestran el tema que se está reproduciendo actualmente y los últimos temas que han sonado (el número de temas a mostrar será elegible por el administrador, hasta un máximo de 10). Esto se hace automáticamente a partir del XML proporcionado por Live365, que incluye los 10 últimos temas de la playlist. Dicha playlist se irá actualizando sin que el usuario necesite hacer nada. Por otro lado, la información de la playlist incluirá enlaces a las fichas de los músicos que han participan en cada uno de los tres temas.

Entrada: Fichero XML e información del catálogo.

Salida: Una parte de la web (bajo el título de *Playlist*) que muestra la información de la canción que se reproduce actualmente y de las anteriores.

SYNC-001: Comunicación entre servicios y MLE

Prioridad: 1

Usuarios: SIS

Descripción: MLE necesita conectarse a la base de datos de la web, pero, por norma general, los hosting's no permiten el acceso remoto a mysql, por temas de seguridad, gracias a la tecnología SOAP, podemos de comunicar MLE (local) y mysql (remoto) mediante unos servicios escritos en php, ya que, al ejecutarse en el servidor, sí que tiene acceso a la base de datos de Drupal.

Tanto la carga, como el guardado de cualquier información, se realizan mediante peticiones a estos servicios.

Para cumplir este requisito, deberemos poder pasar, cualquier información necesaria de MLE a los servicios y de los servicios a MLE.

No incluye el acceso a la base de datos de drupal, requisito SYNC-002.

Entrada: Información a cargar de, o guardar en, la BD.

Salida: Datos si se carga información o cierto/falso para informar del resultado de la operación.

SYNC-002: Comunicación entre servicios y la BD Drupal

Prioridad: 1

Usuarios: SIS

Descripción: Transforma la estructura lógica de MLE en la física de Drupal, es decir, es una capa entre MLE y Drupal que hace de traductor entre los dos sistemas. Por ejemplo, cuando se crea un objeto en MLE, ejemplo un artista, se debe grabar en la base de datos de Drupal, de una manera en que Drupal entienda esos datos y sea capaz de reproducirlos, es decir, a partir de esa información, crear una sección en la web con el contenido que nosotros hemos introducido y los links pertinentes, para poder acceder a ella. A parte, al crear un nuevo elemento en MLE, al introducirlo en Drupal, se deberán actualizar unos contadores internos de Drupal y crear/actualizar diversos elementos de Drupal, que veremos más detalladamente en la fase de implementación.

Entrada: Datos a guardar o cargar en la BBDD de Drupal.

Salida: Datos cargados o éxito de la operación.

3.3. Requisitos no funcionales

La intención de este Proyecto de Final Carrera es que la aplicación web resultante pueda ser usada por un buen número de usuarios que quieran escuchar la radio. Por lo tanto, se espera que cumpla una alta cantidad de los siguientes requisitos:

3.3.1. Usabilidad

Hace referencia a la rapidez y facilidad con que las personas llevan a cabo las tareas propias a través del uso de la aplicación. En este caso, es fácil de manejar, la organización por menús es intuitiva, tanto para el usuario como para el administrador, y el resto de la navegación es clara. Todo esto se ha elaborado conforme a un documento con las normas básicas que debía seguir la web en cuanto a navegación y aspecto.

Para conseguir un alto grado de usabilidad hay que centrarse en cuatro puntos:

- **Una aproximación al usuario**: para desarrollar un producto usable, se tiene que conocer y entender a las personas que representan a los usuarios actuales o potenciales, así como trabajar con ellos.
- **Un amplio conocimiento del contexto de uso:** un producto se considera fácil de aprender y usar en términos del tiempo que toma el usuario para llevar a cabo su objetivo, el número de pasos que tiene que realizar para ello, y el éxito que tiene en predecir la acción apropiada para llevar a cabo. Para desarrollar productos usables hay que entender los objetivos del usuario y hay que conocer los trabajos y tareas del usuario que el producto automatiza, modifica o embellece.
- **El producto ha de satisfacer las necesidades del usuario**: los usuarios son personas ocupadas intentando llevar a cabo una tarea. Se va a relacionar usabilidad con productividad y calidad. El hardware y el software son las herramientas que ayudan a la gente ocupada a realizar su trabajo y a disfrutar de su ocio.
- **Son los usuarios**, y no los diseñadores y los desarrolladores, los que determinan cuándo un producto es fácil de usar.

Esta definición de usabilidad forma parte del artículo sobre usabilidad de la Universidad de Zaragoza (autor: Alejandro Floría Cortés) (Dirección WEB: http://www.sidar.org/recur/desdi/traduc/es/visitable/quees/usab.htm).

Me ha parecido conveniente incluirla en esta memoria ya que define perfectamente el concepto de Usabilidad y es perfectamente aplicable al contexto de este proyecto.

3.3.2. Funcionalidad

- **Adecuación:** capacidad del producto software para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuario especificados.
- **Exactitud:** capacidad del producto software para proporcionar los resultados o efectos correctos o acordados, con el grado necesario de precisión.
- **Interoperabilidad:** capacidad del producto software para interactuar con uno o más sistemas especificados.
- **Seguridad de acceso**: capacidad del producto software para proteger información y datos de manera que las personas o sistemas no autorizados no puedan leerlos ni modificarlos, al tiempo que no se deniega el acceso a las personas o sistemas autorizados.
- **Cumplimiento funcional**: capacidad del producto software para adherirse a normas, convenciones o regulaciones en leyes y prescripciones similares relacionadas con funcionalidad.

3.3.3. Fiabilidad

La gran mayoría de módulos empleados se han probado durante mucho tiempo. Esto asegura que todos o la mayoría de los errores que pudiesen tener estar subsanados.

- **Madurez:** capacidad del producto software para evitar fallar como resultado de fallos en el software.
- **Tolerancia a fallos:** capacidad del software para mantener un nivel especificado de prestaciones en caso de fallos software o de infringir sus interfaces especificados.
- **Capacidad de recuperación:** capacidad del producto software para restablecer un nivel de prestaciones especificado y de recuperar los datos directamente afectados en caso de fallo.
- **Cumplimiento de la fiabilidad:** capacidad del producto software para adherirse a normas, convenciones o regulaciones relacionadas con al fiabilidad.

3.3.4. Eficiencia

- **Comportamiento temporal:** capacidad del producto software para proporcionar tiempos de respuesta, tiempos de proceso y potencia apropiados, bajo condiciones determinadas.
- **Utilización de recursos:** capacidad del producto software para usar las cantidades y tipos de recursos adecuados cuando el software lleva a cabo su función bajo condiciones determinadas.
- **Cumplimiento de la eficiencia:** capacidad del producto software para adherirse a normas o convenciones relacionadas con la eficiencia.

3.3.5. Mantenibilidad

Este requisito se cumple de manera especial ya que Drupal funciona de forma modular y es código abierto. Además, permite que se actualicen sus elementos con gran facilidad y se cumple con las características que se indican a continuación:

- **Capacidad para ser analizado**: es la capacidad del producto software para serle diagnosticadas deficiencias o causas de los fallos en el software, o para identificar las partes que han de ser modificadas. Al funcionar mediante módulos esta tarea es muy sencilla.

- **Capacidad para ser cambiado:** capacidad del producto software que permite que una determinada modificación sea implementada. Al funcionar mediante módulos esta tarea es muy sencilla.
- **Estabilidad:** capacidad del producto software para evitar efectos inesperados debidos a modificaciones del software. Se procura que los módulos empleados hayan sido suficientemente probados para que los errores que pudieran contener se hayan solucionado con el paso del tiempo.
- **Capacidad para ser probado:** capacidad del producto software que permite que el software modificado sea validado.
- **Cumplimiento de la mantenibilidad:** capacidad del producto software para adherirse a normas o convenciones relacionadas con la mantenibilidad.

3.3.6. Multiusuario

El sistema permite ser utilizado por varios usuarios a la vez. Esto supone que en un momento dado dos personas puedan acceder a un mismo recurso, es decir habrá *concurrencia.* Esta característica será especialmente conflictiva en el caso de la playlist, como se verá más adelante.

3.3.7. Seguridad

Se realizan de forma segura: el login de usuarios, la visibilidad de cada parte de la página según el rol del usuario y en las partes especialmente delicadas de envió de información.

3.3.8. Acceso público

Se permite el acceso a usuarios no registrados (invitados) a prácticamente todas las partes de la página, como son las vistas de artistas, álbumes, imágenes, posts, etc. Los usuarios que hayan hecho login tienen permisos para modificar todas estas partes.

3.3.9. Idioma

Como se ha comentado, el idioma de la interfaz y de los contenidos es el inglés. Para mantener la coherencia, la zona de administración o *backoffice* se ha hecho también en ese idioma.

3.3.10. Requisitos para poder alojar el sitio web en un hosting compartido

Capacidad de almacenamiento

Haciendo unos simples cálculos podemos ver sobre qué orden de magnitud se mantendrán los valores de espacio de disco de un sitio web como éste. Sabiendo que, en general, una imagen ocupa más que mil palabras, no tendremos demasiado en cuenta el texto del sitio web, sino que nos centraremos en elementos más pesados.

La cantidad de artistas que existía en la página inicial era de unos 500 sin contar bandas y agrupaciones de dos o más músicos. Por lo tanto, haciendo cálculos generosos se puede esperar que con el tiempo la web albergue del orden de 1.000 artistas.

En caso de que tuvieran una media de 5 álbumes cada uno (que es más que probable que no se llegue a esas cifras), habría unos 5.000 álbumes.

Si hay una media de 3 fotografías por artista (aunque muchos artistas no llegarán a tener fotografía) y dos por álbum, la página podría llegar a alcanzar las 13.000 imágenes.

Los posts también pueden incluir imágenes. Supongamos que en un cierto tiempo se llegasen a escribir 1.000 posts con dos imágenes cada uno, llegando a las 15.000 imágenes totales.

El gestor de contenidos utilizado, Drupal, suele generar, para cada imagen una versión reducida y una miniatura. Podrían añadir a una imagen la mitad de su peso. El único problema que supone todo esto es el espacio, ya que no son números exagerados para una base de datos. A una media de 120 KB por imagen, 180 KB con las otras versiones reducidas, el espacio total ocupado sería de unos 2,6 GB, lo cual para un hosting medio hoy en día no es nada, ya que se están ofreciendo del orden de 500 GB. Incluso suponiendo que en PFCs futuros se incluyesen extensas galerías de imágenes o vídeos musicales alojados en el mismo servidor, la capacidad de almacenamiento no parece que pueda ser conflictiva.

Ancho de banda y capacidad de proceso

El otro problema a tener en cuenta es el de las visitas que pueda tener el sitio web. Se espera que aumenten considerablemente las visitas en comparación con la web anterior, ya que ahora habrá muchos más contenidos.

El número de páginas visitadas por día actualmente es de unas 300. Suponiendo que se llegase a multiplicar ese valor por 20 con los nuevos contenidos, las páginas visitadas pasarían a ser 6.000. Dividiendo por ejemplo por 8 horas (no podemos suponer que las 24 horas del día se reciba la misma cantidad de visitas) tendríamos 750 páginas vistas por hora, lo que en cada minuto supondría 12,5. Es decir, en un segundo se estarían sirviendo 0,2 páginas. Cualquier servicio de hosting puede soportar eso.

Suponiendo que una página media, incluidas varias imágenes, tuviera un peso de 100 KB, 6.000 páginas darían una tasa de transferencia mensual de 17 GB. Hoy en día se pueden conseguir 25 GB de transferencia a precios muy reducidos.

Playlist

El caso de la playlist del sitio web merece ser tratado a parte, ya que se envía cada vez que se carga un página y se actualiza cada vez que empieza a sonar un tema nuevo, lo que significa que todos los visitantes de la web harán una misma petición a la vez. Cuando se hable en detalle de la playlist en la sección 8.6 se expondrá cómo se ha tratado ese problema para hacer la transferencia y la carga del sistema lo más pequeñas posible.

En cualquier caso, la playlist más grande que se puede enviar (configurando el máximo número de temas mostrados) ocupa del orden de 2,5KB. Analicemos los dos casos por separado.

1. Envío de la playlist al cargar una página nueva: Se ha contado como media que una página con imágenes pudiese tener 100KB. Se incluya o no la playlist, es un cambio que no afecta para nada a los resultados obtenidos en el apartado anterior.
2. Actualización automática de la playlist: Un tema de jazz puede durar una media de 4 minutos, así que la playlist se enviaría cada 4 minutos a todos los usuarios conectados en ese momento. Generalmente un usuario no dedica más de un minuto de media a la visualización de una página. Si antes hemos calculado que se sirven 0,2 páginas cada segundo, eso significaría una media de 12 usuarios conectados a la vez. A 2,5 KB cada uno, la transferencia cada 4 minutos sería de 30 KB, lo cual no es problema para el servidor. Eso son 450 KB cada hora, que, repetido durante 24 horas al mismo ritmo serían 10,5 MB al día y 315 MB al mes. Añadidos a los 17 GB de las páginas vistas se llegaría a unos 17,3 GB de transferencia al mes.

A pesar de ser cálculos basándose en un crecimiento del número de visitas muy optimista y en tamaños exagerados (la playlist considerada es de 9 temas, pero una playlist con 3 temas ocupa 1 KB), los números son más que admisibles para un hosting normal.

3.3.11. Requisitos SEO

SEO son las siglas de *Search Engine Optimizer* (Optimización para Motores de Búsqueda). Empezó tratándose de un conjunto de técnicas surgidas de la necesidad de atraer visitantes a las páginas web a partir de las consultas realizadas a los buscadores de Internet. Poco después se convirtió en una profesión y actualmente es todo un arte. El posicionamiento en buscadores requiere conocimientos de muchas disciplinas diferentes, desde cuestiones relacionadas con el lenguaje escrito o el diseño y colocación de las distintas partes que forman la página hasta los entresijos de los servidores web.

El objetivo de SEO es conseguir que una página web aparezca en las primeras posiciones de los resultados de una búsqueda en los grandes *web crawlers* (aunque en la práctica la inmensa mayoría de técnicas SEO están diseñadas para Google). Los buscadores no suelen considerar esta práctica como negativa siempre que no se violen sus políticas.

Los buscadores ordenan las páginas según su importancia en una especie de *ranking*. En el caso de Google, esa puntuación inicialmente se limitaba a un número conocido como *PageRank*, que era una valoración de una web en función del valor de las webs que tenían enlaces hacia ella. Hoy en día, Google utiliza muchos más criterios para devolver resultados además del PageRank.

Lo curioso del caso es que el comportamiento interno de los buscadores es algo desconocido para las personas que intentan posicionar una web, por lo que las técnicas SEO se basan meramente en suposiciones derivadas de los cambios en los resultados después de hacer modificaciones en las webs. Pero lo realmente divertido es que actualmente ni los propios diseñadores de los algoritmos de valoración son capaces de predecir cómo se posicionará una web, ya que esos algoritmos están basados en sistemas caóticos y cualquier pequeño cambio puede hacer variar los resultados de una búsqueda.

Para este sitio web se cuenta con unas especificaciones bien definidas que se deben cumplir para mejorar el posicionamiento. Pero antes veamos algunas técnicas comunes.

Redacción apropiada de contenidos internos

Aunque no hay que redactar todo el contenido como si sólo los buscadores fueran a leerlo, hay que cuidar la densidad de palabras y frases clave y mantener un equilibrio entre lo que los visitantes y los buscadores van a entender. Existen herramientas que ayudan a conocer la densidad de las palabras clave.

Optimización técnica de la página

Para la elección correcta del contenido de las etiquetas meta se pueden usar herramientas como:

- *Keyword Suggestion Tool*: da la frecuencia con que un término o frase es buscada por día. http://www.digitalpoint.com/tools/suggestion/
- *Keyword Selector Tool*: da el número de veces que un término o frase ha sido buscado en el último mes. http://inventory.overture.com/d/searchinventory/suggestion/
- *SubmitExpress*: varias herramientas que validan la densidad de palabras clave. http://www.submitexpress.com/analyzer/

Aspectos a tener en cuenta:

- Evitar la mala ortografía de las palabras clave que suelen ocurrir al teclear rápido. Por ejemplo: *compar* en vez de *comprar*.

- Usar url's que contengan frases clave. Por ejemplo: www.example.com/products/phones en lugar de www.example.com?q=523.
- No usar guiones bajos en nombres de urls largas, sino guiones-normales (Google acepta ambos tipos desde julio de 2007).
- Usar palabras clave en el título de la página. La mayoría de expertos considera que éste es el principio SEO más importante a la hora de posicionarse.
- Utilizar título y etiquetas meta diferentes para cada página.
- Usar cabeceras ***(h1, h2, h3...)*** para definir importancia en el contenido y colocar en ellas las palabras clave.
- No duplicar contenido.
- Utilizar palabras clave en el ***anchor text*** de los enlaces.
- Rellenar todas las etiquetas ***alt*** de las imágenes y ***title*** de los enlaces.
- No utilizar frames.
- Imprescindible un *sitemap* visual (para el usuario) y otro en XML (para los buscadores).
- Utilizar un archivo *robots.txt para evitar la indexación de contenido no deseado.*
- En cualquier página web sobre posicionamiento SEO se pueden encontrar más técnicas como estas por docenas: SEOmoz.com, tiene un ranking de técnicas de posicionamiento elaborado por los más prestigiosos SEO a nivel internacional.

Actualizaciones, creación de datos

Los buscadores indexan con más frecuencia las webs que cambian de contenido continuamente; por lo tanto si se mantiene un contenido de calidad y un nivel de creación de datos aceptable, la escalada en los buscadores se hará más rápidamente.

Link building

Este es uno de los puntos con más importancia del SEO. La construcción de enlaces hacia un sitio es lo hace aumentar el PageRank. Eso sí, la página que enlaza debe tener un buen PageRank.

En este punto toca hacer referencia a un artículo que es toda una autoridad en el tema: *101 Link Building Tips to Market Your Website*, por Aaron Wall y Andy Hagans. http://www.seobook.com/archives/001792.shtml.

Redacción de contenidos externos

Consiste en escribir artículos, reportajes, comentarios y respuestas en foros con contenido interesante y de calidad, siempre enlazando con la firma del pie de página, hacia tu sitio web. Esta técnica estaba mal vista por Google por la facilidad con que se puede crear spam en los blogs y por eso hace varios años Matt Cutts y su equipo crearon los atributos HTML *nofollow*, aplicables a los enlaces. Estos enlaces seguirán atrayendo visitas a la web (las de cualquiera que vea el enlace y esté interesado), pero no serán seguidos por el buscador, haciendo que no se compute el PageRank. Esta técnica también ha sido muy útil para evitar los enlaces pagados.

Red social

Es importante crear una comunidad virtual que genere contenido cambiante y dinámico para la web (foros, posts, intercambio de contenidos entre el usuario y el sitio web...).

Si se dispone de más recursos, pueden construirse varias webs "paralelas" con utilidades o de temática relacionada, que a fin de cuentas redirijan tráfico a la web principal.

Be famous

El sitio web tiene que ser conocido entre los usurarios que estén interesados en este tipo de contenidos. Por ejemplo, organizando eventos donde darse a conocer. Es bueno que se hable del sitio, incluso si hablan mal.

Seguimiento de resultados

Hay que mantenerse al tanto del tráfico que genera la página web y cómo es ese tráfico. Es muy aconsejable utilizar herramientas como Google Analytics. Se tienen que corregir "imperfecciones" de tráfico, palabras clave con bajo rendimiento, etc.

Pero hay que ser cauto porque un cambio de palabras clave puede tardar mucho tiempo en hacerse notar, para bien o para mal, y se tienen que analizar bien los resultados antes de lanzarse a un cambio radical.

Google también proporciona otras herramientas para webmasters, las *Webmaster Tools*, donde se puede enviar un sitemap, estudiar las palabras clave más eficientes, ver los posibles errores SEO cometidos y recibir avisos de Google si ha ocurrido algo indeseado. Por ejemplo, que el sitio haya sido excluido del buscador. En tal caso, desde allí se puede pedir una reconsideración.

Promociones offline

La promoción offline se puede convertir en visitas online a una web. Publicar en otros medios de comunicación (periódicos, revistas, radio, etc.), asistir a conferencias y congresos. Aquél que lea un artículo de la web en el periódico, o a quien le haya gustado una buena conferencia, es muy probable que visite su sitio web, o publique una entrada en su web haciéndole referencia. El boca a oreja sigue siendo efectivo. Incluso en técnicas SEO.

Especificaciones para el PFC

Los puntos bien definidos para el proyecto de barcelonajazzradio son los siguientes:

- Los diferentes epígrafes de cada página se escribirán para que haya una repetición pertinente (sin spam) de las palabras clave.
- Es importante que en el código que lean los buscadores, en especial Google, se encuentre en primer lugar un título H1 que contenga el nombre principal (del artista o del álbum). Antes de ese título no debería haber nada de contenido.
- Los títulos de las páginas deberían ser de este estilo: "barcelona jazz radio – nombre de artista", "barcelona jazz radio – título de álbum"...
- La URL física de cada una de las páginas también debería incluir esos nombres.
- El apartado *Featured Musicians* actuará en parte de sitemap de RW.
- El hosting de la web hay que configurarlo para que el alojamiento sea directo (sin la redirección actual).
- Todos los links externos (tanto los de la lista del menú principal como los de las páginas de contenido) incluirán un atributo *nofollow* para que los buscadores no continúen buscando información a través de ese link.
- Las imágenes de artista, banda y álbum incluirán también el ALT. Asimismo es recomendable que los nombres de los jpg sean descriptivos, tanto para artistas como para álbumes.

3.3.12. Diseño de la interfaz gráfica de RW

En cuanto al diseño del sitio web, también estaba definido con mucho detalle en un documento que incluía medidas y capturas de pantalla que indicaban la apariencia que debía mostrar al usuario, además de un esquema navegacional. Veamos algunas partes de este documento.

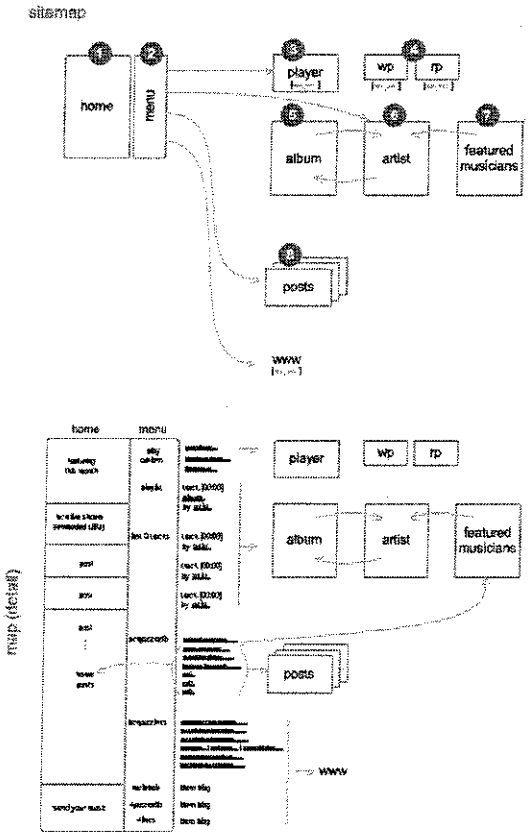


Figura 3.1: Sitemap simple y detallado

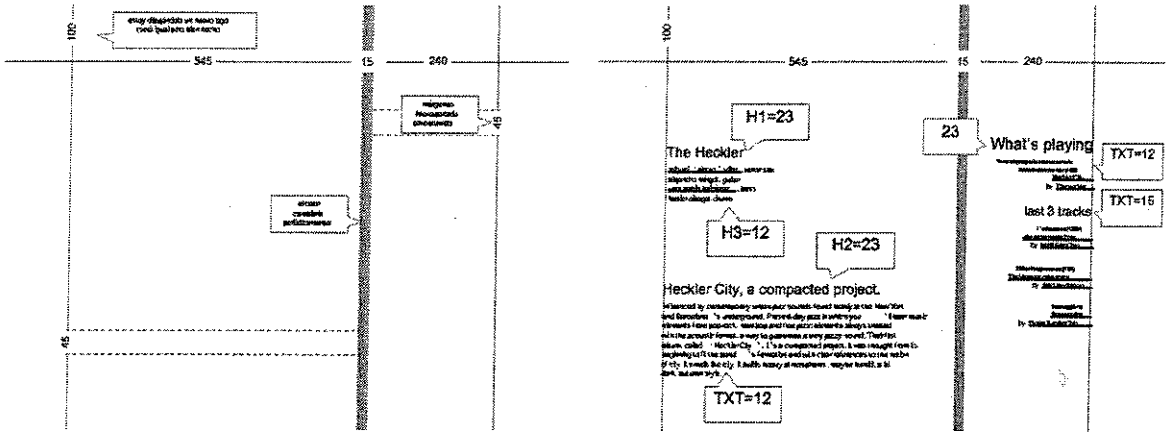


Figura 3.2: Medidas y tipos de letra

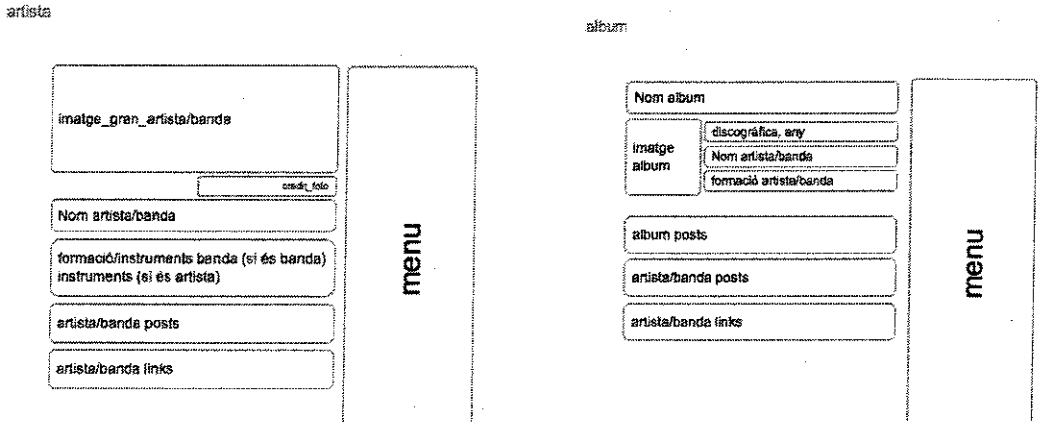


Figura 3.3: Estructura de las fichas de artista y álbum

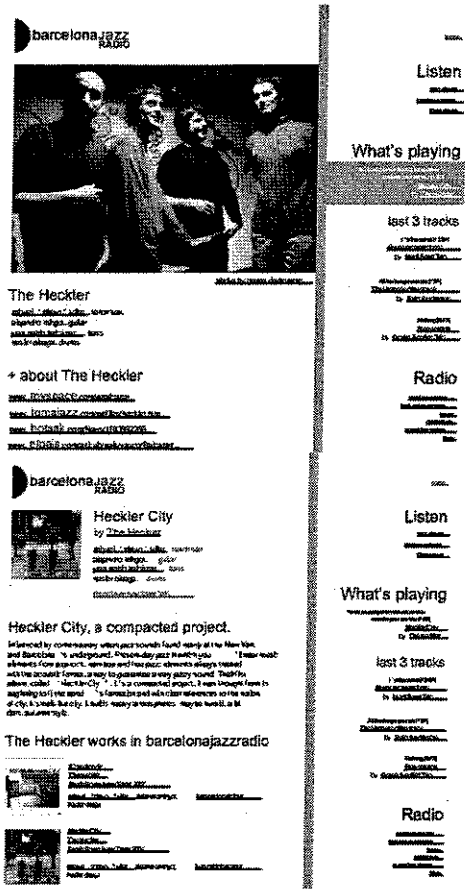


Figura 3.4: Diseño de las fichas de artista y álbum

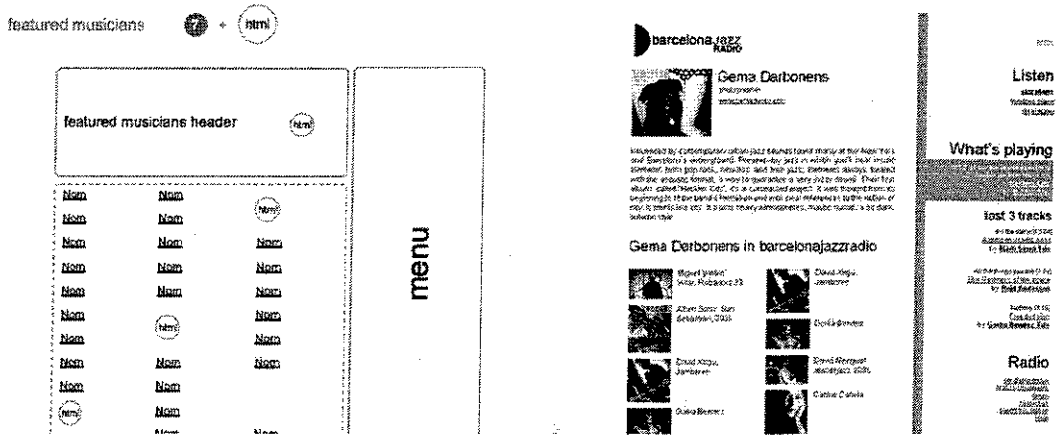


Figura 3.5: Featured musicians y galería de fotógrafo

4. Planificación y costes del proyecto

Una vez establecidos los objetivos e identificados los diferentes requerimientos del sistema, es el momento de enumerar y describir las tareas necesarias para satisfacerlos y desarrollar el proyecto. A partir de aquí se hace una previsión del tiempo y coste necesarios para llevarlo a cabo.

Aunque la planificación y las estimaciones iniciales no sean muy ajustadas, hacer una planificación inicial tiene varias ventajas, de entre las que destacan:

- Asegurar el proceso de desarrollo, estableciendo los recursos y el tiempo necesarios para su finalización.
- Adquirir compromisos de tiempos y plazos más o menos fiables para cada una de las fases del proyecto.
- Llevar el seguimiento del proceso, identificar desviaciones, retardos en las fechas de entrega, etc.
- Facilitar la toma de decisiones.

En resumen, es mucho mejor tener una planificación poco precisa que no tener ninguna.

4.1. Planificación

En proyectos grandes y complejos, donde durante algún momento de su desarrollo intervienen distintas personas y/o distintos recursos, es necesario llevar una gestión, control y asignación de recursos bien planificada. Más aún si las tareas que componen el desarrollo están interrelacionadas entre sí.

Para llevar a cabo este tipo de planificación resultan de gran utilidad técnicas como los diagramas de Pert o Gantt. Sin embargo, en este caso se ha preferido usar otras herramientas similares que cumplen los mismos propósitos. El proyecto consta de tres personas trabajando en áreas bastante diferenciadas, y además de la gestión del tiempo, que también se ha hecho, lo más importante era saber en cada momento en qué estaba trabajando cada uno.

Como se ha explicado en la sección 2.6, los métodos utilizados para mantenernos informados y comunicados han sido dos:

- Informes semanales al director del proyecto para que conociera los avances y las dificultades. Este informe consistía en una hoja de cálculo que incluía las distintas tareas a las que se había dedicado tiempo durante la semana y la cantidad de horas

dedicadas a cada una. Acompañando a la hoja de cálculo debía ir un email explicando detalladamente lo que se había hecho y lo que se pensaba hacer durante la semana siguiente. De este modo, la previsión concreta de horas se hacía de forma semanal.

- Una hoja de cálculo adaptada a nuestras necesidades. En esta hoja de cálculo estaban detalladas las tareas principales y las subtareas, usando dos categorías diferentes, de modo que podían ocultarse las subtareas y ver qué tareas principales estaban terminadas o a medias. También se podía saber fácilmente en qué estaba trabajando cada uno, mostrar sólo las tareas no terminadas de uno de los tres, ver la fecha estimada de finalización o las dependencias que creaba en otras tareas el no tener una finalizada. Si había algo que los demás debían saber, se podía añadir un comentario y así todos estaban informados echando un rápido vistazo.

La estimación de tiempo se ha realizado examinando atentamente los datos obtenidos durante el análisis de requisitos y teniendo presentes los objetivos. No obstante, dada la poca experiencia en la construcción y gestión de sistemas de este tipo, estas estimaciones pueden resultar poco ajustadas a la realidad.

Aún así, hacer la estimación permitió acotar tanto la duración probable del proyecto como el coste aproximado del mismo.

4.1.1. Identificación de las etapas del proyecto

Como al empezar el proyecto ya había una idea bastante clara de hacia dónde iría encaminado, lo primero que se hizo fue un **aprendizaje** de las herramientas que se utilizarían, a saber, PHP, MySQL, Javascript y sobre todo Drupal. Aunque había usado alguna vez PHP para alguna práctica y para algún proyecto propio, necesitaba unos conocimientos más profundos, y sobretodo comprender su relación con Drupal. De igual manera tuve que aprender MySQL que no lo conocía bastante bien, por lo que sólo hubo que pulir algunos detalles. En cuanto a Javascript, realmente necesitaba empezar de cero, lo mismo que con Drupal, que no había utilizado nunca.

Para llevar todo esto a cabo, el director del proyecto había elaborado una guía de autoaprendizaje, compartida en los Grupos de Google, que permitiría a cualquier persona que desconociera estas tecnologías aprenderlas en pocas semanas.

A continuación hubo una toma de contacto con el proyecto y con los **requisitos** preestablecidos. Eso se hizo en varias reuniones mientras terminaba la fase de formación. Se estudiaron los requisitos globales del proyecto y se asignaron a los PFC's de forma que fueran lo más independientes posible.

En la parte de **especificación** se definieron los modelos conceptuales, de casos de uso y de comportamiento. Fue relativamente rápido porque algunas partes ya estaban determinadas por los otros PFCs y sólo hacía falta complementarlos y adaptarlos. Aún así, se dedicó el tiempo necesario, ya que esa documentación serviría para los pasos siguientes.

El **diseño** también dependía en buena parte de los otros proyectos. Las bases de datos estaban bastante elaboradas y sólo hubo que añadir algunas tablas y relacionarlas con el resto. La elección de las distintas tecnologías ya estaba hecha, pero había que ver cómo se adaptaban las especificaciones del proyecto a esas tecnologías. Básicamente estudiar módulos existentes, ver qué funcionalidades ya quedaban cubiertas sin necesidad de implementarlas y cuáles no.

A partir de ese momento venía la **implementación**. Se codificaron los requisitos y se volvieron a modificar iterativamente algunos modelos de datos a medida que se descubrían formas más eficientes de resolver problemas que las que se habían propuesto inicialmente.

Finalmente se pasaron algunos **juegos de prueba** muy exhaustivos y se validó la solución final.

El último paso ha sido la **documentación**, es decir, la creación de esta memoria. Algunas secciones ya se habían elaborado con anterioridad, como el análisis de los requisitos o partes de la implementación y juegos de prueba.

4.1.2. Tareas y dedicación estimada

La dedicación que se acordó con el director del proyecto fue de 750h de trabajo, en la tabla que tenemos a continuación (Tabla 4.1), se observa la distribución de la carga de trabajo en media, digo en media, ya que esta tabla indica la dedicación media, pero en la realidad se ha dado el caso de uno o varios días hacer más o menos horas de lo que me tocaba o por razones personales (problema de salud) o laborales.

En la tabla siguiente se puede observar las horas que se han dedicado realmente al proyecto (**negrita**) y las horas estimadas (subrayado).

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1-oct	8-oct	15-oct	22-oct	29-oct	5-nov	12-nov	19-nov	26-nov	3-dic	10-dic	17-dic	24-dic	31-dic	7-ene	14-ene	21-ene	28-ene	4-feb	11-feb	18-feb	25-feb	3-mar	10-mar	17-mar
20	**20**	**17**	**19**	**23**	**20**	**25**	**19**	**16**	**13**	**25**	**25**	**15**	**5**	**20**	**15**	**15**	**15**	**25**	**25**	**20**	**20**	**15**	**15**	**15**
20	20	20	20	20	20	20	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15

26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	
24-mar	31-mar	7-abr	14-abr	21-abr	28-abr	5-mayo	12-mayo	19-mayo	26-mayo	2-jun	9-jun	16-jun	23-jun	30-jun	7-jul	14-jul	21-jul	28-jul	4-ago	11-ago	18-ago	25-ago	1-sep	Totales
15	**15**	**23**	**18**	**17**	**20**	**15**	**20**	**20**	**20**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**40**	**40**	**15**	**35**	**0**	**0**	**45**	**815**
15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	770

Tabla 4.1: Distribución de la carga de trabajo en semanas

Veamos ahora las distintas tareas identificadas y el tiempo que se pensaba dedicar a cada una de ellas.

Formación

De forma resumida, esto es lo que proponía la guía de autoaprendizaje inicial:

- Drupal I: Drupal sin usar Drupal (50 horas)
    - Lectura rápida del libro de Drupal (VANDYK, WESTGATE, 2007): El objetivo es tener identificadas todas las partes y más adelante saber dónde recurrir cuando haga falta.
    - Mirar los vídeos de demostración disponibles en la página oficial de Drupal.
    - Documentación y recursos online de Drupal.
- PHP + MySQL (30 horas)

  - Leer un manual o libro de PHP (COGGESHALL, 2005) de la misma forma que con el de Drupal.
  - Instalar el entorno necesario.
  - Hacer unos ejercicios propuestos para poner a prueba los conocimientos.
- Drupal II: Extender funcionalidad de Drupal con módulos existentes (35 horas)
  - Instalación de Drupal.
  - Registro en drupal.org.
  - Pruebas básicas.
  - Instalar y probar módulos.
- Drupal III: Extender funcionalidad con módulos propios (40 horas)
  - Releer el capítulo del libro de Drupal (VANDYK, WESTGATE, 2007) llamado *Writing a module.*
  - Crear módulos siguiendo las directrices del libro.
- Drupal IV: Theming (30 horas)
  - Releer la documentación del libro y ampliarla con otra.
  - Crear un tema nuevo, tomando otro como modelo.
  - Probar la aplicación del tema y ver qué problemas puede haber con ciertos módulos.
- Drupal: Recursos en paralelo para Drupal II, III y IV (30 horas)
  - Utilizar una lista de enlaces que se encuentra en los mismos Grupos.

Las tareas de auto aprendizaje se dividieron de la forma que se puede ver en la Tabla 4.2. Como se empezaron a estudiar los requisitos antes de terminar la formación, se alargó la formación una semana más para poder dedicar tiempo a ambas cosas.

Tarea	Horas
Drupal I	**50**
Pro Drupal Development	30
Vídeos de demostración	15
Documentación, recursos online...	5
PHP + MySQL	**30**
Manual de PHP (+ MySQL + Javascript...)	7
Ejercicios PHP	23
Drupal II	**35**
Instalar Drupal y hacer pruebas básicas	10
Instalar y probar módulos	15
Recursos paralelos	10
Drupal III	**40**
Crear módulos propios	30
Recursos paralelos + requisitos	10
Drupal IV	**30**
Themes: crear, modificar, adaptar...	27
Recursos paralelos	3
Drupal V (opcional)	**30**
Otras pruebas	30
TOTAL	**215**

Tabla 4.2: Dedicación estimada a la autoformación

Requisitos

Como se ha indicado, la definición de los requisitos se llevó a cabo sobre todo durante la etapa de formación. Fueron reuniones con el director del proyecto y con los otros dos estudiantes de las que salieron la lista y la descripción de esos requisitos.

Entre las reuniones y la redacción del documento explicativo del funcionamiento del sistema y de los requisitos, se dedicaron aproximadamente unas **55 horas** a esta parte del proyecto.

También salió de esta etapa la planificación de las siguientes tareas, así como los tiempos estimados a dedicarles.

Especificación

La especificación dependía mucho de los modelos conceptuales ya existentes proporcionados por el administrador de la radio, por lo que se estimó que el tiempo a dedicar sería de unas **25 horas**.

Diseño

Para el diseño también se contaba con una buena parte construida, así como con varios documentos, con lo que se hizo una previsión de unas **20 horas**.

Implementación

Para la implementación se calculó el tiempo restante descontando al tiempo total los valores anteriores, el tiempo de los juegos de prueba y la documentación, obteniendo una dedicación del orden de **365 horas.**

Juegos de prueba

Se estimó que unas **30 horas** para hacer pruebas y corregir errores serían suficientes.

Documentación

A la redacción de la memoria se dedicarían por lo menos las dos últimas semanas, unas **60 horas.**

Tiempo de coordinación

Este tiempo es el que ha dedicado el director de proyecto a coordinar el proyecto, reuniones, etc, el tiempo estimado es de unas **66 horas. No se tiene en cuenta en esta planificación ya que es tiempo que ha dedicado el director y no el alumno. Lo mostramos para que posteriormente hacer el coste económico del proyecto.**

4.1.3. Comparación entre dedicación real y estimada.

El recuento total de horas previstas y reales se puede observar en la Tabla 4.3.

Tarea	Estimado	Real
Formación	215	240
Requisitos	55	55
Especificación	25	30
Diseño	20	25
Implementación	365	380

Juegos de prueba	30	30
Documentación	60	55
	770	**815**

Tabla 4.3: Dedicación estimada por tareas

El tiempo dedicado, como era de esperar, no coincidió con lo que se había planeado.

Donde se ha alejado más planificación ha sido fundamentalmente en dos puntos clave:

- La formación: Esto ha sucedido ya que la formación es donde más se ha tenido que investigar, a veces se ha tenido que abandonar una línea de investigación debido a que se ha llegado a un camino sin salida y se han tenido que encontrar otros, aumentando así el consumo de horas.

- La implementación: Alguna vez se ha tenido que replantear la programación de ciertas piezas del PFC ya que conceptualmente la programación era correcta pero a la práctica era mejor usar otro tipo de implementación. Retrasando así su implementación.

Gracias a los informes semanales, contamos con una relación detallada del tiempo dedicado a cada tarea. Los nombres de las tareas provienen en gran parte de la especificación, el diseño y la implementación, y por lo tanto se explicarán en detalle en capítulos siguientes.

4.2. Análisis económico

Una vez se tiene el cómputo de horas dedicadas al proyecto, se puede llevar a cabo el estudio económico de éste. En este aparatado sólo se tienen en cuenta los datos de este PFC.

4.2.1. Coste de personal

Respecto al desarrollo, son necesarios al menos dos perfiles: el de analista de sistemas y el de programador. Además, en todo proyecto suele haber un perfil de jefe de proyecto, quien dirige las operaciones del resto del equipo y que en este caso será el director del proyecto.

En la actualidad dichos perfiles cobran sueldos del orden de los que se muestran en la Tabla 4.4, donde se suponen una media de 240 días laborables por año y un coste de seguridad social a cargo de la empresa de un 33%.

Rol	Sueldo anual bruto	Coste anual	Coste/día
Jefe de proyecto	42.000 €	55.860 €	232.75 €
Analista	30.000 €	39.900 €	166.25 €
Programador	21.000 €	27.930 €	116.37 €

Tabla 4.4: Coste diario por rol

El proyecto dura, sin contar la fase inicial de formación, 600 horas. Como tenemos el cómputo de las horas dedicadas a reuniones (66 horas) tomaremos ese valor como tiempo

de dedicación del jefe de proyecto. Respecto a las 600 horas del tiempo que he de distribuir entre los roles de analista y programador, estimo lo siguiente:

- Analista = Horas Requerimientos + Horas Especificación + Horas Diseño + Horas documentación = 55 + 30 + 25 + 25 = 135 horas.
- Programador = Horas Implementación + Horas Pruebas + Horas documentación = 380 + 30 + 30 = 440 horas

Los costes totales de personal se observan en la Tabla 4.5.

Rol	Horas	Días	Coste/día	Total
Jefe de proyecto	66	8,25	232.75 €	**1921 €**
Analista	135	16,88	166.25 €	**2806 €**
Programador	440	55	116.37 €	**6400 €**
TOTAL				**11127 €**

Tabla 4.5: Coste de personal

4.2.2. Coste de hardware

Durante la realización del proyecto se ha utilizado:

- Dominio web y hosting con PHP y MySQL: Se ha contratado un servicio de bajo coste para poder realizar el desarrollo. El servicio dura un año y su precio es 27,14 €.
- Toshiba Satellite Pro, Intel Core 2 Duo 2'16Ghz, 3 GB RAM, 500 GB HD, monitor 20'': 895,63 €. El equipamiento informático posee una vida media de 3 años, con lo cual amortizaremos solo el porcentaje correspondiente al uso durante el proyecto (5 meses). Por lo tanto, el % imputable es de (5/36)*100 = 13,89 %. El coste total nos da, pues, 124,12 €

El coste total de hardware es: 27,14 + 124,12 = **151,26 €**

4.2.3. Coste de software

Para la realización del proyecto se ha utilizado el siguiente software en la estación de trabajo:

- Windows Vista: 114,84 €
- TextMate 1.5.6 con licencia para un usuario: 48.75 €
- Adobe Photoshop CS2: 415 €
- Microsoft Office 2004: 540,55 €
- Herramientas gratuitas: Firefox 2 y 3, Cyberduck (cliente FTP), FolderShare, Skype...

El software contenido en el servidor ya viene incluido en el precio de su alquiler contabilizado en el apartado correspondiente.

El coste para el software tiene un periodo de amortización de 3 años. Análogamente al apartado anterior, procederemos al cálculo del coste del software atribuible a este PFC (Tabla 4.6).

Software	Precio software	Plazo amortización	% imputable por 5 meses	Coste
Windows Vista	114,84	36	13,89 %	15,95€
TextMate	48,75	36	13,89 %	6,77€
Adobe Photoshop	415	36	13,89 %	57,64€
Microsoft Office	540,55	36	13,89 %	75,08€
				155,44€

Tabla 4.6: Coste de software

4.2.4. Coste de ocupación

Se ha partido de un coste de alquiler de 600 € para una oficina donde llevar a cabo el proyecto. Este precio incluye los servicios de acceso a Internet, teléfono y electricidad.

Como en este proyecto somos 3 personas trabajando el coste de ocupación debe repartirse entre los tres miembros del grupo para así calcular el coste atribuible a este PFC (ver Tabla 4.7).

Coste oficina	Capacidad	Duración proyecto	Coste
600 €	3 personas	5 meses	**1.000€**

Tabla 4.7: Coste de ocupación

4.2.5. Coste total

Finalmente, calculamos el coste total en la Tabla 4.8.

Concepto	Coste
Coste de personal	11.788,89€
Coste de hardware	151,26€
Coste de software	155,44€
Coste de ocupación	1.000€
	13.095,59€

Tabla 4.8: Coste Total

5. Especificación

Una vez definidos los requisitos que debe tener la aplicación, comienza la fase de especificación.

Esta etapa define básicamente qué se quiere desarrollar, dejando a un lado cómo se hará y, por ello, es una de las partes más importantes del desarrollo del software.

Por un lado, la especificación representa el contrato con el cliente y, por otro, es la base fundamental de todas las etapas siguientes de la ingeniería del software.

Para la realización del documento de especificación, se utiliza la notación *UML* (Unified Model Language). Esta notación permite realizar el modelado de los sistemas de información. Es la herramienta de representación de modelos orientados a objetos más conocida y utilizada actualmente. Este lenguaje gráfico permite visualizar, especificar, construir y documentar un sistema software.

Hay dos aspectos que pueden distinguirse en un sistema:

- El aspecto estático representa los conceptos significativos del dominio del sistema (el contexto en el que sistema se ejecutará). Además, describe cuales son las funcionalidades para cada agente o actor que intervenga.
- El aspecto dinámico describe los aspectos de un sistema que cambian con el tiempo. En concreto describe las interacciones entre los objetos, los estados de estos, las transiciones entre los estados, los eventos que se producen y las operaciones que se ejecutan.

Por ello, el documento de especificación se suele dividir en cuatro partes. Para representar el aspecto estático del sistema se emplean:

- Modelo Conceptual de Datos
- Modelo de Casos de Uso

En cambio, para representar el aspecto dinámico del sistema se emplean:

- Modelo de Comportamiento
- Modelo de Estados

5.1. Modelo de casos de uso

La especificación de los casos de uso de un sistema permite describir la secuencia de acciones que tiene un evento determinado. Se deben indicar todos los actores que

intervendrán en el evento y se deben tratar cada uno de los casos que se puedan dar desde el punto de vista del usuario.

Por lo tanto, describe la funcionalidad del sistema independientemente de su implementación.

Los componentes del modelo de casos de uso son las siguientes:

- Definición de actores
- Diagrama de casos de uso
- Especificación de los casos de uso

5.1.1. Definición de actores

En el sistema que se está implementando, como se ha especificado en el entorno del PFC (sección 2.2), intervienen cuatro actores: administrador (ADM), músico (MUS), invitado (INV) y sistema (SIS).

Administrador : Usuario registrado encargado de la introducción de contenidos.

Músico : Usuario registrado con ciertos permisos de introducción de contenidos.

Invitado : Usuario no identificado.

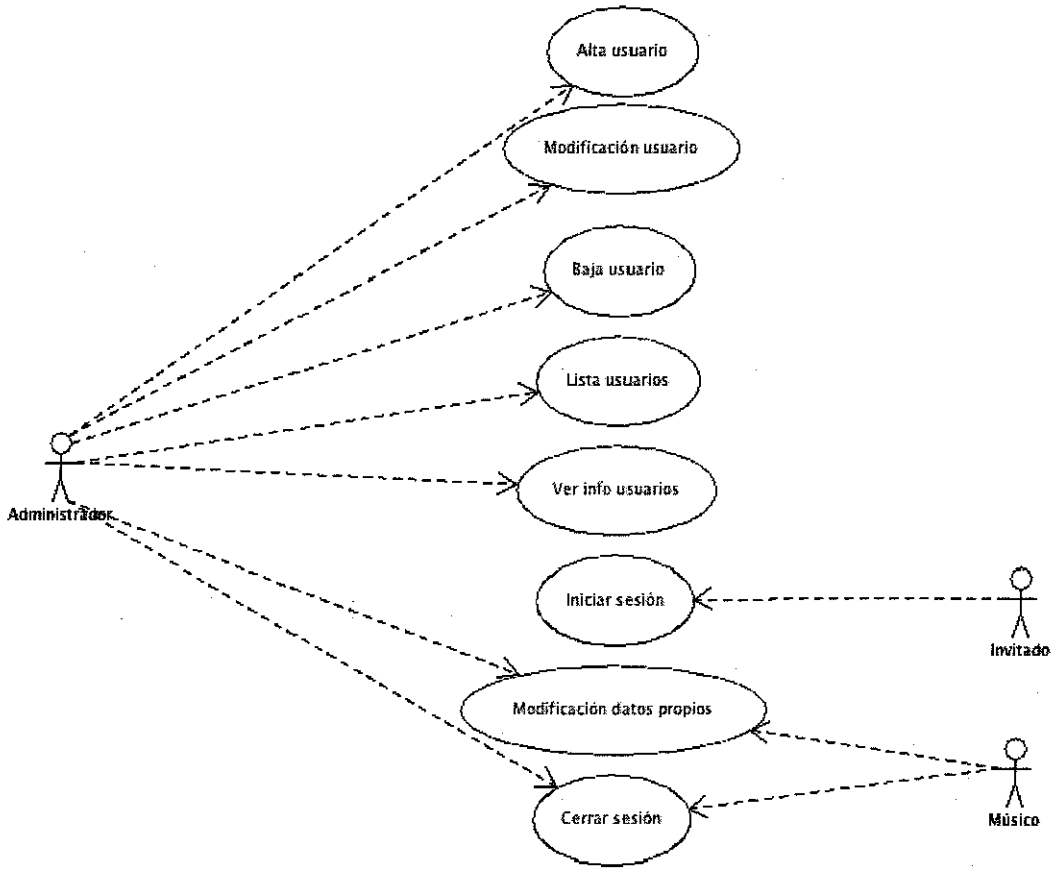
Sistema : Actor que realiza funciones automáticas.

5.1.2. Diagramas y especificación de casos de uso

Debido al gran número de casos de uso existentes, se han agrupado en diferentes paquetes para facilitar su comprensión. Los paquetes son:

- Gestión de usuarios
- Funciones de negocio
- Estadísticas y configuración
- Sincronización

Gestión de usuarios



Caso de uso: Alta usuario

Actores: Administrador

Descripción: Permite dar de alta a un nuevo usuario de tipo músico.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que el administrador desea dar de alta a un usuario músico en la aplicación.	
	2. Solicita los datos del nuevo usuario.
3. El administrador introduce los datos solicitados.	
	4. Se comprueba que los datos introducidos sean correctos.
	5. Se crea el nuevo usuario.

Cursos alternativos:

4a: En caso de que los datos introducidos no sean válidos (por ejemplo, si el usuario existe) o no se introduzcan todos los necesarios, el sistema lo comunicará al administrador para que los vuelva a introducir.

Caso de uso: Modificación usuario

Actores: Administrador

Descripción: Permite modificar los datos de un usuario de tipo músico.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que el administrador desea modificar los datos de un usuario músico. Para ello se solicita la lista de usuarios.	2. El sistema muestra la lista de usuarios.
3. El administrador selecciona el usuario que desea modificar y da la orden de modificación.	4. El sistema recibe la orden y muestra los datos actuales del usuario y la posibilidad de modificarlos.
5. El administrador modifica los datos que desea.	6. El sistema comprueba que los datos son correctos y los almacena.

Cursos alternativos:

6a: En caso de que los datos introducidos no sean válidos o no se introduzcan todos los necesarios, el sistema lo comunicará al administrador para que los vuelva a introducir.

Caso de uso: Baja usuario

Actores: Administrador

Descripción: Permite dar de de baja a un usuario músico existente.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que el administrador desea eliminar a un usuario del sistema. Para ello solicita el listado de usuarios.	2. El sistema muestra la lista de usuarios.
3. El administrador selecciona el usuario a dar de baja y da la orden.	4. El sistema recibe la orden y la ejecuta. 5. El sistema muestra la lista de usuarios sin el que ha sido dado de baja.

Caso de uso: Lista usuarios

Actores: Administrador

Descripción: Permite visualizar la lista de usuarios del sistema.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que el administrador desea ver todos los usuarios y da la orden al sistema para que los muestre.	
	2. El sistema recibe la orden y muestra la lista con todos los usuarios.

Caso de uso: Ver información de usuarios

Actores: Administrador

Descripción: Permite ver la información de los usuarios.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que el administrador desea ver la información de un usuario y pide al sistema que le muestre la lista de usuarios.	
	2. El sistema muestra la lista de usuarios.
3. El administrador selecciona al usuario del cual quiere ver la información y da la orden al sistema para que la muestre.	
	4. El sistema muestra la información del usuario seleccionado.

Caso de uso: Iniciar sesión

Actores: Invitado

Descripción: Permite a un usuario invitado que se identifique y pase a ser considerado músico o administrador.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que un invitado quiere identificarse.	
	2. Solicita los datos del usuario (nombre de

	usuario y contraseña).
3. El invitado introduce los datos solicitados.	4. Se comprueba que los datos introducidos sean correctos.
	5. Se permite al músico o al administrador acceder como tal.

Cursos alternativos:

4a: En caso de que los datos introducidos no sean válidos o no se introduzcan todos los necesarios, el sistema lo comunicará al invitado para que los vuelva a introducir.

Caso de uso: Modificación de datos propios

Actores: Administrador, Músico

Descripción: Permite a un usuario registrado modificar sus propios datos.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que el usuario registrado desea modificar su información de registro y lo solicita al sistema.	
	2. El sistema muestra los datos del usuario.
3. El usuario modifica los datos y da la orden de modificarlos.	
	4. Se comprueba que los datos introducidos sean correctos.
	5. Se modifican los datos del usuario.

Cursos alternativos:

4a: En caso de que los datos introducidos no sean válidos o no se introduzcan todos los necesarios, el sistema lo comunicará al usuario para que los vuelva a introducir.

Caso de uso: Cerrar sesión

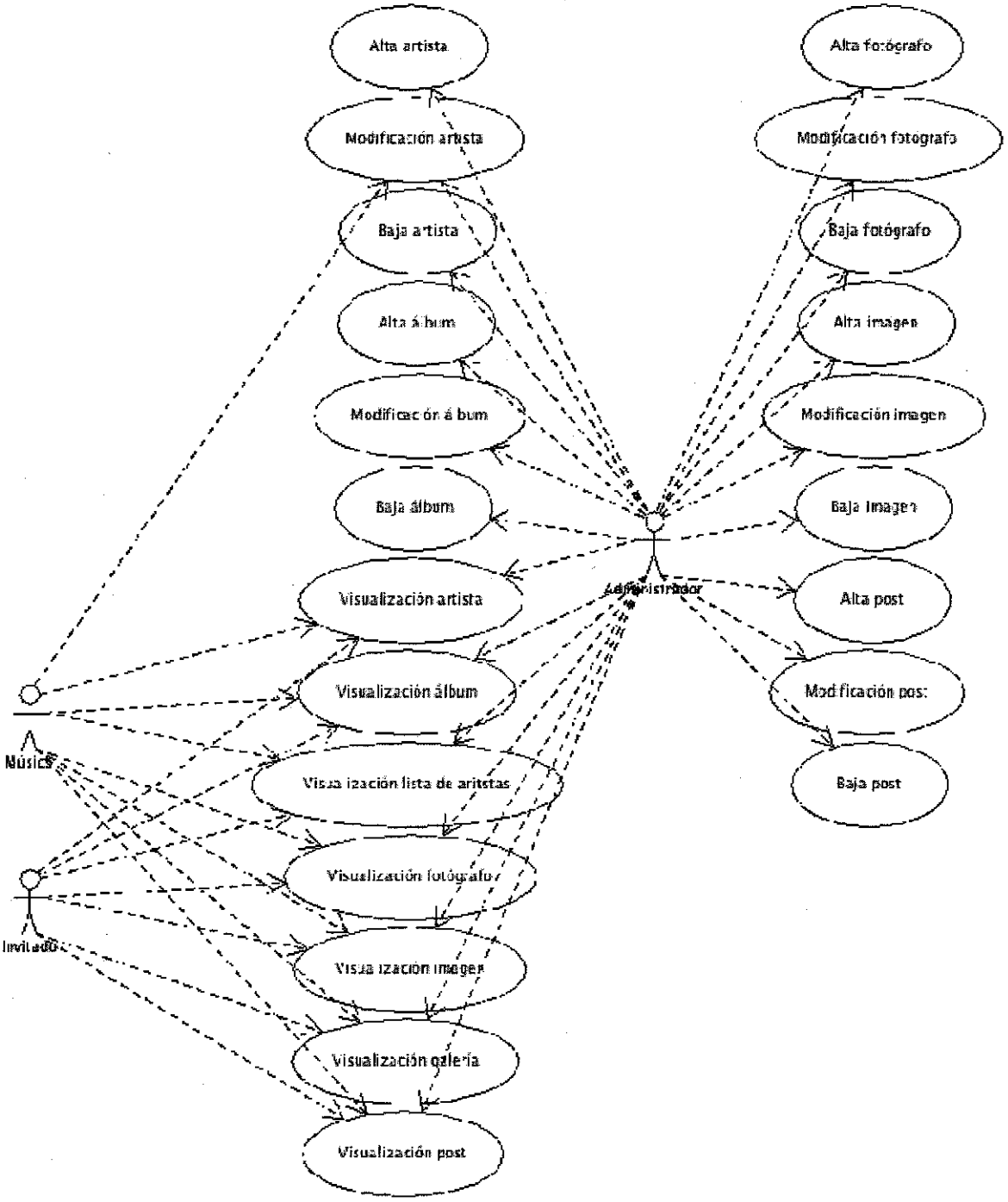
Actores: Administrador, Músico

Descripción: Permite a un usuario registrado e identificado dejar de ser reconocido como tal y pasar a ser invitado.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que el usuario identificado quiere dejar de estarlo y lo comunica al sistema.	
	2. El sistema cierra la sesión del usuario.

Funciones de negocio



Nota: Dado que las altas, modificaciones, bajas y visualizaciones de los distintos elementos del Catálogo musical son todas parecidas y extenderían innecesariamente este documento, se describirán sólo los casos del artista y se supondrá que álbumes, fotógrafos e imágenes tienen una descripción análoga. Lo mismo para los posts del blog. En caso de que haya que especificar diferencias para alguno de esos elementos se indicará con una nota al pie de página.

Caso de uso: Alta artista

Actores: Administrador

Descripción: Permite dar de alta a un nuevo artista (músico o banda) en el sistema.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que el administrador desea dar de alta a un nuevo artista en la aplicación.	
	2. Solicita los datos del nuevo artista.
3. El administrador introduce los datos solicitados.	
	4. Se comprueba que los datos introducidos sean correctos.
	5. Se crea el nuevo artista.

Cursos alternativos:

4a: En caso de que los datos introducidos no sean válidos (por ejemplo, si el artista existe) o no se introduzcan todos los necesarios, el sistema lo comunicará al administrador para que los vuelva a introducir.

4b:[1] Si en la lista de instrumentos (caso del músico) o en la formación (caso de la banda) se introduce un instrumento inexistente, se creará y se añadirá al sistema.

Caso de uso: Modificación artista

Actores: Administrador, Músico[2]

Descripción: Permite modificar los datos de un artista (músico o banda).

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que el administrador desea modificar los datos de un artista. Para ello se solicita la lista de artistas[3].	2. El sistema muestra la lista de artistas.
3. El administrador selecciona el artista que desea modificar y da la orden de	

[1] El curso alternativo 4b es válido tanto para artistas como para álbumes, ya que también cuentan con una formación.

[2] El músico sólo puede modificar artistas (cuando se trata del artista que lo representa en el Catálogo). El resto de modificaciones sólo puede hacerlas el administrador.

[3] Sólo existe en el sistema una lista para artistas. En el caso de álbumes, imágenes, fotógrafos o posts, para encontrarlos se hará bien a partir de la ficha de un artista o banda, bien usando el buscador o bien desde la página principal (sólo posts). Esto también es válido para los casos de uso de baja de artista y visualización de artista.

modificación.	4. El sistema recibe la orden y muestra los datos actuales del artista y la posibilidad de modificarlos.
5. El administrador modifica los datos que desea.	6. El sistema comprueba que los datos son correctos y los almacena.

Cursos alternativos:

6a: En caso de que los datos introducidos no sean válidos o no se introduzcan todos los necesarios, el sistema lo comunicará al administrador para que los vuelva a introducir.

6b:[4] Si en la lista de instrumentos (caso del músico) o en la formación (caso de la banda) se introduce un instrumento inexistente, se creará y se añadirá al sistema.

Caso de uso: Baja artista

Actores: Administrador

Descripción: Permite dar de de baja a un artista (músico o banda) existente.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que el administrador desea eliminar a un artista del sistema. Para ello solicita el listado de artistas.	
	2. El sistema muestra la lista de artistas.
3. El administrador selecciona el artista a dar de baja y da la orden.	
	4. El sistema recibe la orden y la ejecuta.
	5. El sistema muestra la lista de artistas sin el que ha sido dado de baja.

Caso de uso: Visualizar artista

Actores: Administrador, Músico, Invitado

Descripción: Permite ver la información de un artista (músico o banda).

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que el usuario desea ver la información de un artista y pide al sistema que le muestre la lista de artistas.	

[4] El curso alternativo 6b es válido tanto para artistas como para álbumes, ya que también cuentan con una formación.

	2. El sistema muestra la lista de artistas.
3. El administrador selecciona el artista del cual quiere ver la información y da la orden al sistema para que la muestre.	
	4. El sistema muestra la información del artista seleccionado.

Caso de uso: Visualizar lista de artistas[5]

Actores: Administrador, Músico, Invitado

Descripción: Permite ver la lista de todos los artistas (músicos y bandas) del sistema.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que el usuario desea ver la información de un artista y pide al sistema que le muestre la lista de artistas.	
	2. El sistema muestra la lista de artistas.

Caso de uso: Visualizar galería de imágenes[6]

Actores: Administrador, Músico, Invitado

Descripción: Permite ver las galerías de imágenes de artistas o fotógrafos.

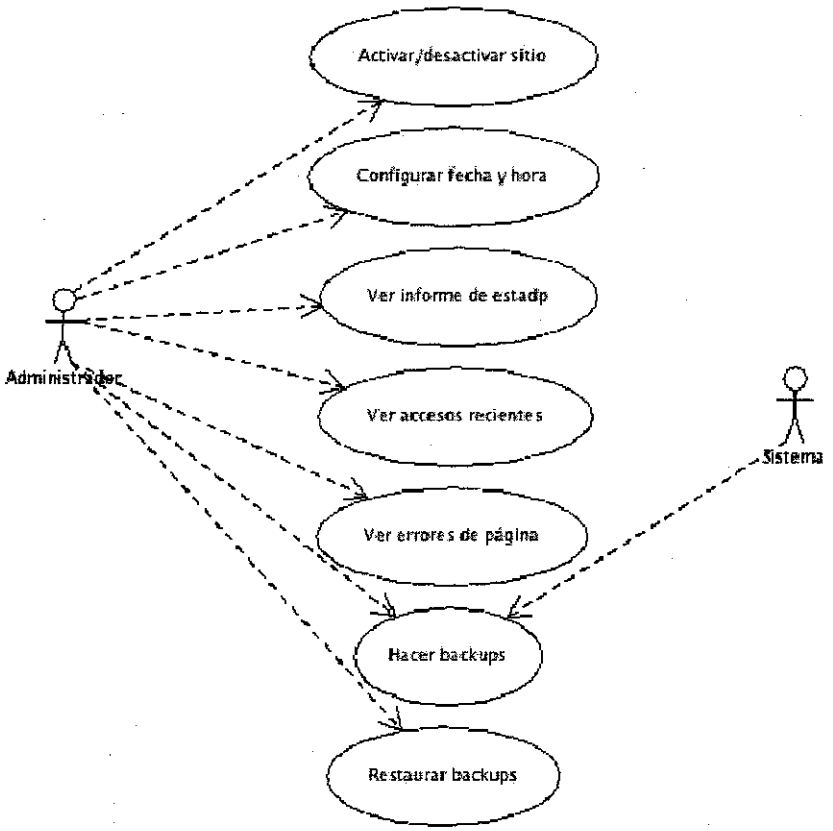
Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que el usuario desea ver una galería de imágenes de un artista o fotógrafo, y se lo pide al sistema desde una de sus fichas.	
	2. El sistema muestra la galería de imágenes.

[5] Este caso de uso es exclusivo para artistas.

[6] Este caso de uso es exclusivo para galerías de imágenes.

Estadísticas y configuración



Caso de uso: Activar/desactivar sitio

Actores: Administrador

Descripción: Permite que se active o desactive el sitio de forma temporal.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que el administrador desea desactivar el sitio para realizar alguna modificación importante y lo solicita.	
	2. El sistema pregunta si se desea activar o desactivar.
3. El administrador confirma lo que desea.	
	4. El sistema activa o desactiva según lo que le indique el comercial.

Caso de uso: Configurar fecha y hora

Actores: Administrador

Descripción: Permite que se modifique la fecha y la hora de la aplicación.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que el administrador desea modificar la fecha o la hora d la aplicación y lo solicita al sistema.	
	2. El sistema pregunta qué hora y fecha desea introducir.
3. El administrador introduce los datos correctos.	
	4. El sistema guarda los datos modificados.

Caso de uso: Ver informe de estado

Actores: Administrador

Descripción: Permite ver en qué estado se encuentra la aplicación, base de datos, etc.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que el administrador desea comprobar el estado en que se encuentra el sitio y lo solicita.	
	2. El sistema muestra un informe sobre el estado actual de la aplicación, bases de datos, etc.

Caso de uso: Ver accesos recientes

Actores: Administrador

Descripción: Permite ver quién ha accedido al sitio recientemente. Útil en el caso de que se produzca algún ataque o intento de ataque, ya que queda registrado quiénes estaban conectados en ese momento.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que el administrador desea ver el listado de los últimos accesos y lo solicita al sistema.	
	2. El sistema muestra el listado de los últimos accesos.

Caso de uso: Ver errores de página no encontrada

Actores: Administrador

Descripción: Permite ver qué direcciones se han pedido que hayan dado ese error. Es útil para identificar enlaces erróneos.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que el administrador desea comprobar qué direcciones han dado errores de página no encontrada.	
	2. El sistema muestra el listado de estas direcciones.

Caso de uso: Hacer backups

Actores: Administrador, Sistema

Descripción: Permite realizar copias de seguridad del sitio y las bases de datos.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que el administrador desea hacer una copia de seguridad de toda la información.	
	2. Solicita qué datos se quieren copiar.
3. El administrador elige lo que quiere copiar.	
	4. Se hacen las copias de seguridad.

Cursos alternativos:

1a: En el caso de que se programen copias automáticas, es el propio sistema el que las hace sin necesidad del administrador.

Caso de uso: Restaurar backups

Actores: Administrador

Descripción: Permite restaurar una copia de seguridad realizada.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que el administrador desea restaurar una copia de seguridad existente.	
	2. Solicita la copia a restaurar.
3. El administrador elige la copia y confirma que la quiere restaurar.	

Sincronización

Caso de uso: Sincronizar sitio

Actores: Administrador, Sistema

Descripción: Permite que se sincronice el sistema con MLE.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1. El caso de uso comienza en el momento en que el administrador desea activar el proceso de sincronización del sitio.	
	2. El sistema sincroniza ambos sistemas.
	3. El sistema devuelve a MLE los datos sincronizados.

5.2. Modelo conceptual

El modelo conceptual es la representación de los conceptos de la realidad que resultan significativos en el dominio del sistema que se quiere desarrollar.

Dicho modelo permite expresar restricciones o reglas aplicadas a los objetos del modelo, haciéndolo íntegro y consiguiendo que se corresponda adecuadamente con el dominio del problema.

Este modelo muestra principalmente:

- *Clases de objetos:* describen un conjunto de objetos con:
    - Las mismas propiedades
    - Idéntica relación con otros objetos
    - Semántica común
    - Comportamiento común
- *Asociaciones de objetos:* representan la relación entre dos o más objetos.
- *Atributos de las clases de objetos:* propiedades o características que son compartidas por todos los objetos de la misma clase.

Además, el modelo conceptual se divide en las siguientes partes:

- Clases y atributos.
- Diagrama de clases.
- Restricciones de integridad.

5.2.1. Clases y atributos

Las clases del dominio, junto con sus atributos se muestran en la Figura 5.1.

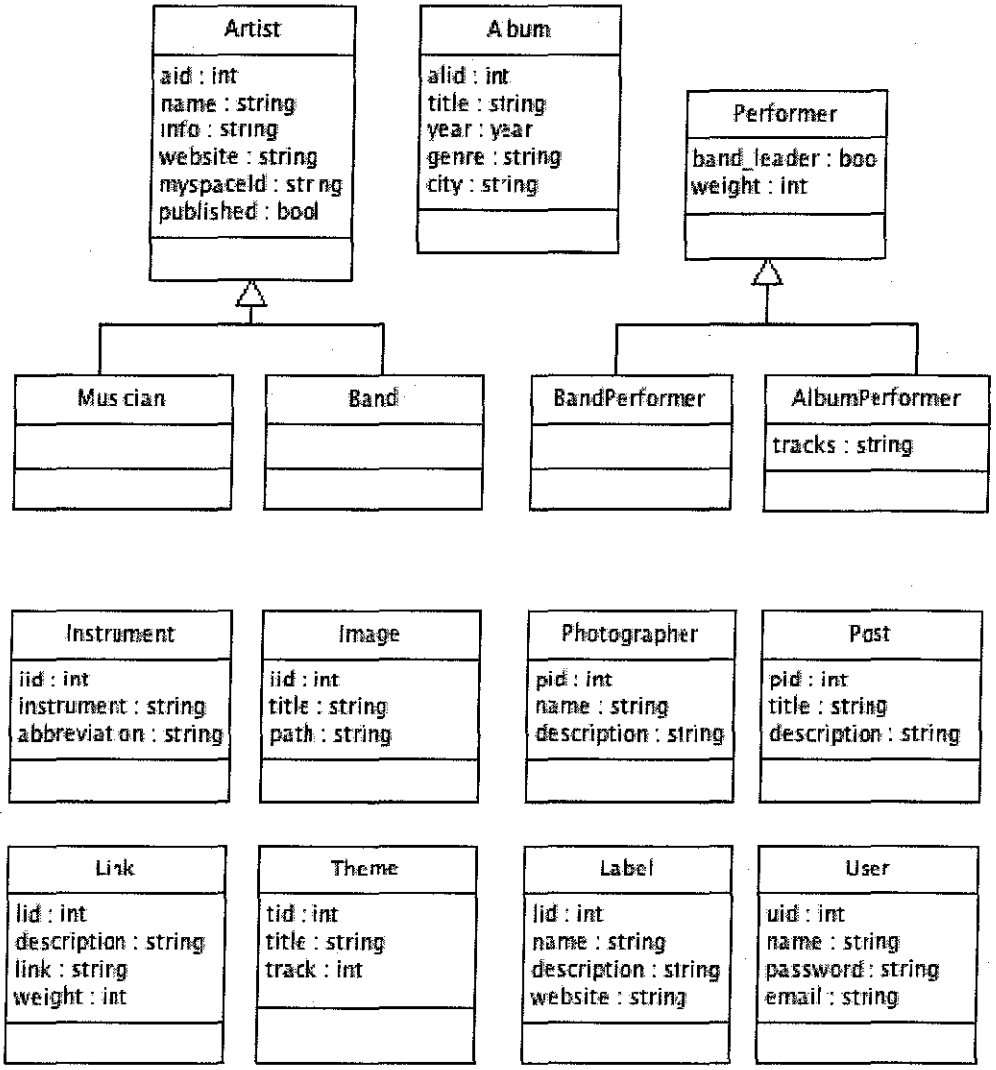


Figura 5.1: Clases y atributos

5.2.2. Diagrama de clases

El diagrama de la Figura 5.2 muestra las distintas clases principales y sus relaciones entre ellas.

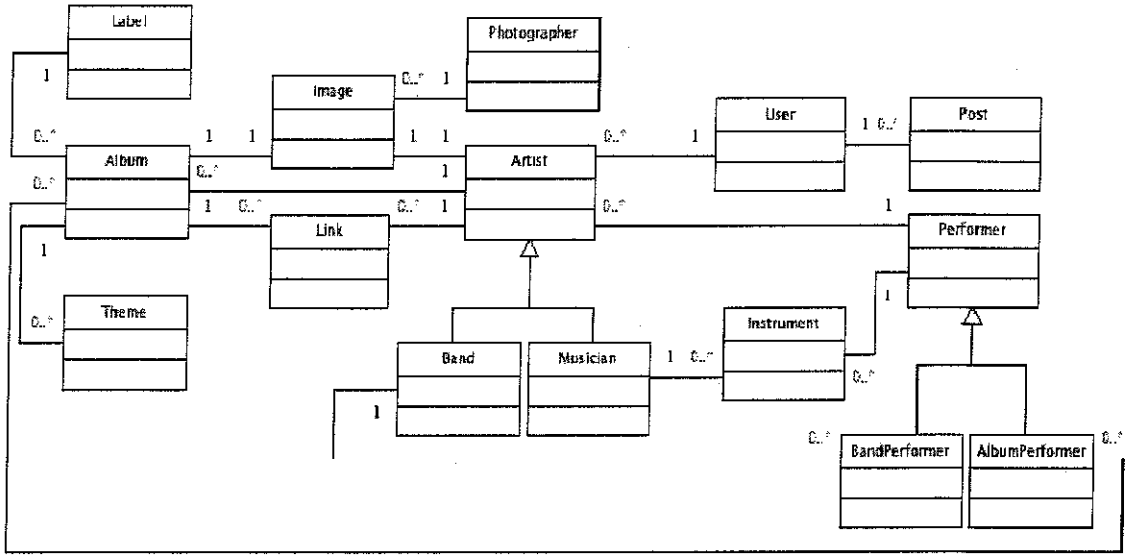


Figura 5.2: Diagrama de clases

5.2.3. Restricciones de integridad

El modelo conceptual tiene unas restricciones textuales de integridad que no pueden expresarse con la notación UML. A continuación se listan las restricciones de clave que definen el modo mediante el cual se identifica cada una de las entidades del modelo.

Artist:	aid
Album:	alid
Instrument:	iid
Image:	iid
Photographer:	pid
Post:	pid
Link:	lid
Theme:	tid
Label:	lid
User:	uid

Existen otras restricciones referentes al modelo, como las que aparecen en el proceso de normalización. La clase Performer era una clase asociativa que provenía de la unión de artistas con instrumentos. A la vez, sus subclases BandPerformer y AlbumPerformer tienen una asociación con Band y Album respectivamente. Al normalizar el diagrama obtenemos las restricciones textuales:

1. Para cada Artist, Instrument y Band sólo puede existir una instancia de BarndPerformer.
2. Para cada Artist, Instrument y Album sólo puede existir una instancia de AlbumPerformer.

5.3. Modelo de comportamiento

El modelo de comportamiento muestra cómo actúan los actores del sistema de una forma más concreta que en el modelo de casos de uso. Este modelo se puede descomponer en dos partes:

- Diagramas de secuencia
- Contratos de las operaciones

5.3.1. Diagramas de secuencia

Este diagrama es uno de los más efectivos para expresar la interacción entre objetos en un sistema. Para cada caso de uso se examina su descripción para modelar un diagrama de secuencia. El diagrama de secuencia contiene detalles de la implementación del escenario (objetos, clases y mensajes pasados entre objetos).

En un diagrama de secuencia se representan: con líneas discontinuas verticales, los objetos que intervienen en el escenario y con vectores horizontales, los mensajes pasados entre objetos. Estos vectores horizontales se dibujan cronológicamente desde la parte superior a la inferior mientras que la distribución horizontal de los objetos es arbitraria.

A continuación se indican los diagramas de secuencia que se han incluido. Se han elegido algunos significativos para no alargar excesivamente esta sección.

1. Alta usuario
2. Listar usuarios
3. Alta artista
4. Modificación artista
5. Baja artista
6. Visualización artista
7. Visualización lista de artistas

Alta usuario

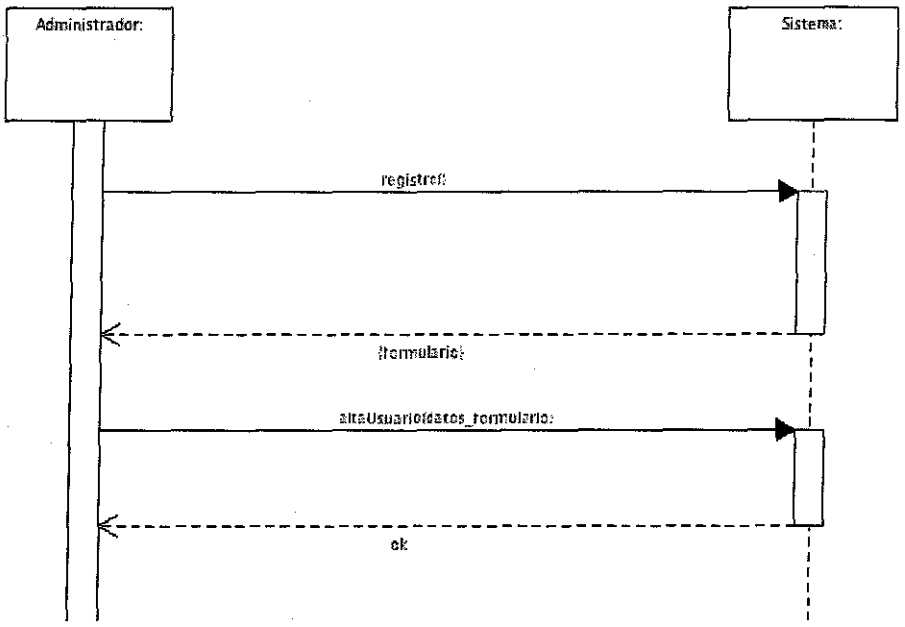


Figura 5.3: Alta usuario

Datos del formulario: {uid: int; name: string; password: string; email: string}

Listar usuarios

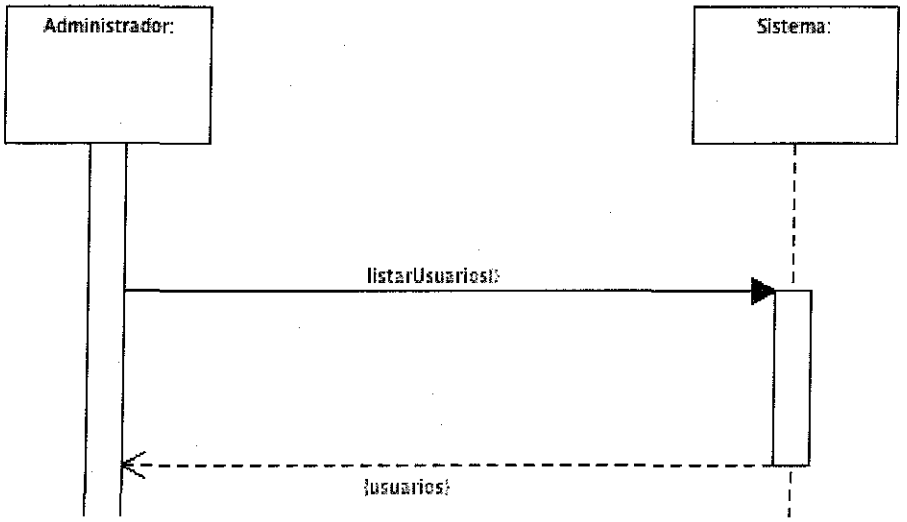


Figura 5.4: Listar usuarios

Alta artista

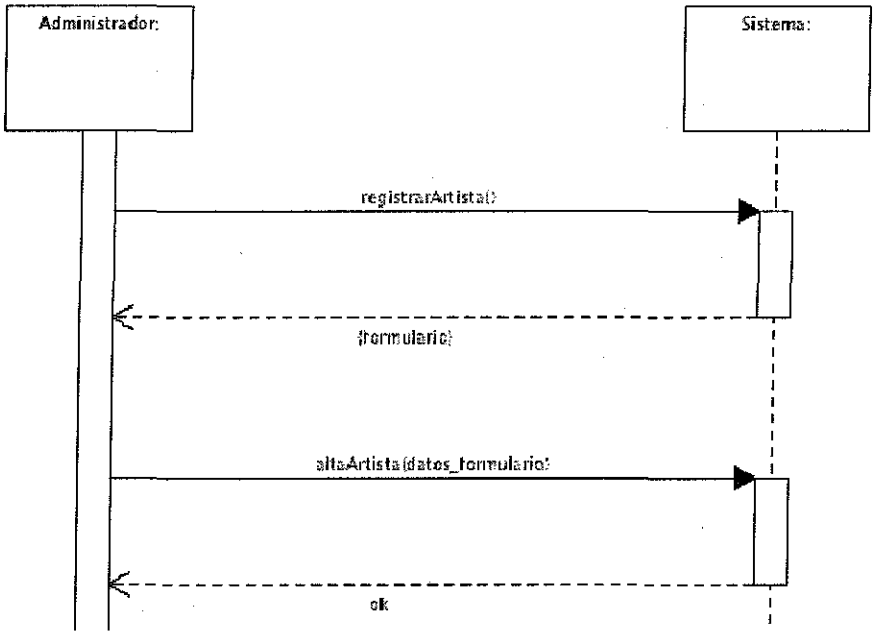


Figura 5.5: Alta artista

Datos del formulario: {aid: int; name: string; info: string; website: string; myspaceId: string; published: bool; *{lista de links}*; *{lista de instrumento o performers}*}

Modificación artista

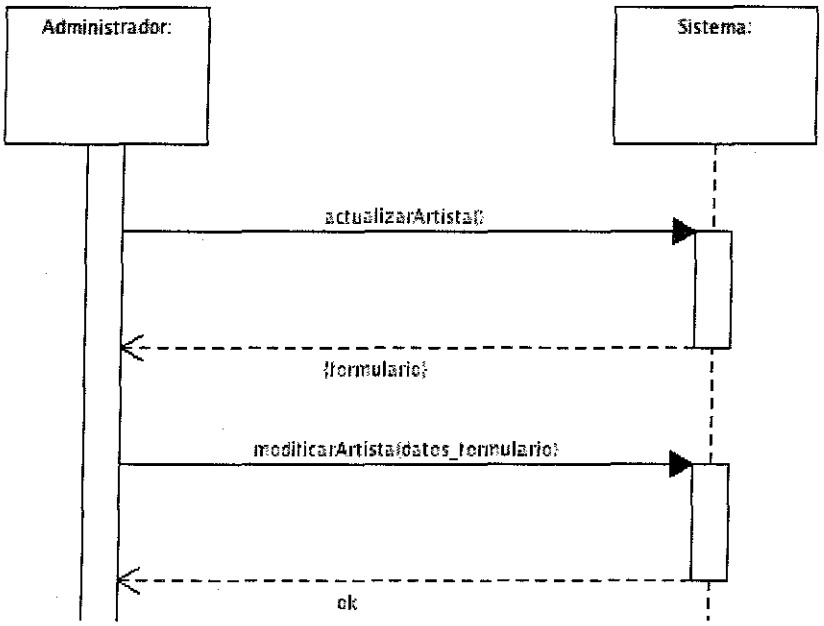


Figura 5.6: Modificación artista

Datos del formulario: {aid: int; name: string; info: string; website: string; myspaceId: string; published: bool; {*lista de links*}; {*lista de instrumento o performers*}}

Baja artista

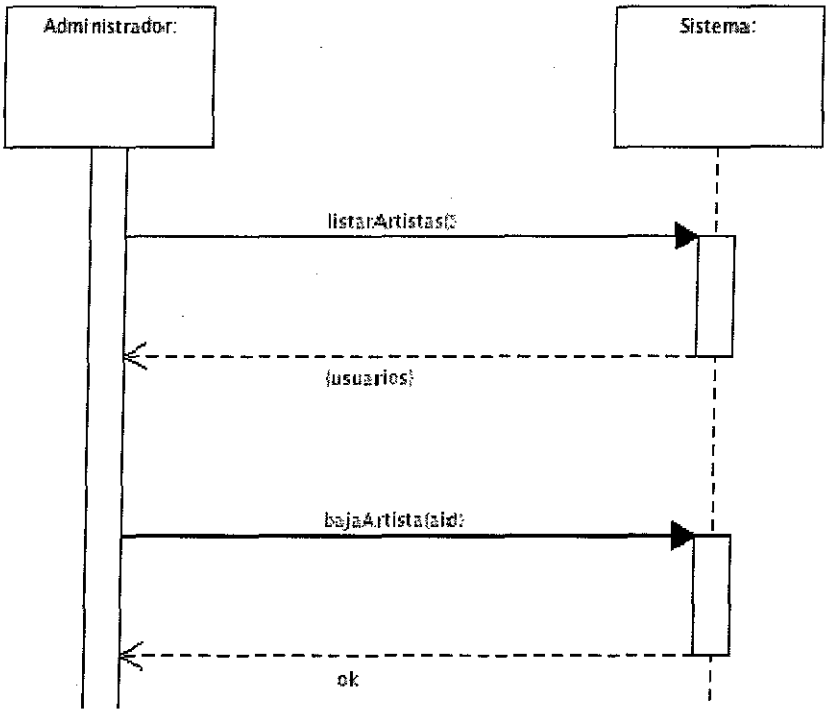


Figura 5.7: Baja artista

Visualización artista

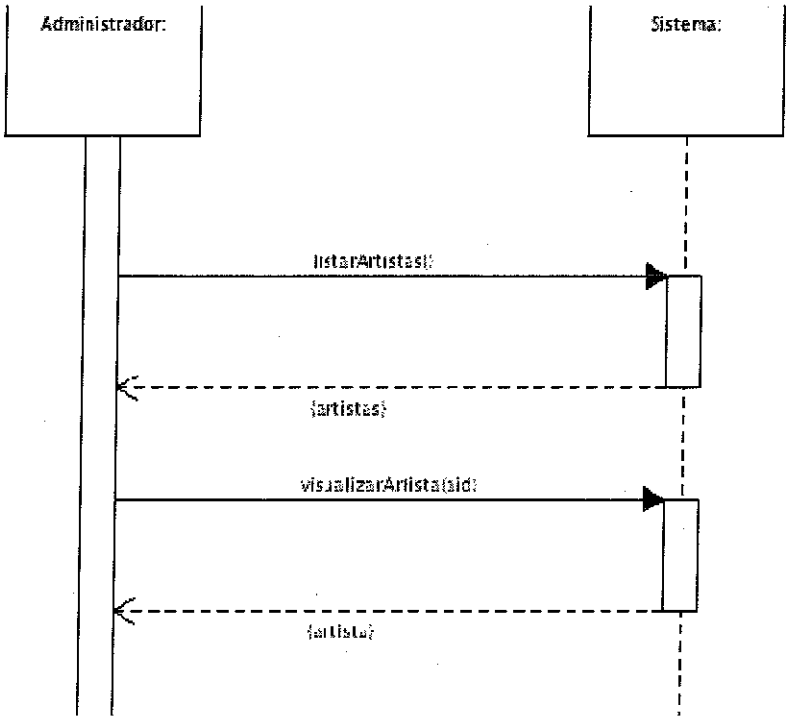


Figura 5.8: Visualización artista

Visualización lista de artistas

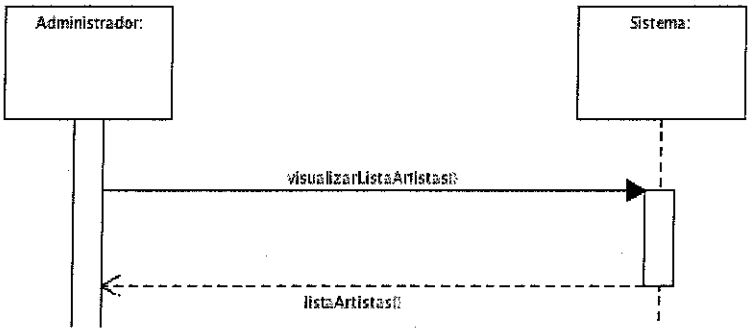


Figura 5.9: Visualización lista de artistas

5.3.2. Contrato de las operaciones

Los contratos de las operaciones describen el comportamiento del sistema en cuanto a cambios de estado de la base de la información y de las salidas que el sistema proporciona cuando se invoca una operación.

La descripción que se realiza en el contrato de una operación se centra en qué hace la operación y no en explicar cómo lo hace. Los contratos de las operaciones constan primordialmente de precondición, postcondición y salidas.

Caso de uso: Alta usuario

Operación: altaUsuario

Responsabilidades: Recibe los datos de un nuevo usuario y lo da de alta en el sistema.

Precondiciones: No existe un usuario con el mismo nombre.

Postcondiciones: Da de alta una nueva instancia de la clase User.

Salida: Se muestra un mensaje diciendo que el usuario ha sido dado de alta correctamente.

Caso de uso: Listar usuarios

Operación: listarUsuarios

Responsabilidades: Muestra la lista de usuarios del sistema.

Precondiciones: ---.

Postcondiciones: Muestra la lista de usuarios.

Salida: Se muestra por pantalla una lista con los usuarios.

Caso de uso: Alta artista

Operación: altaArtista

Responsabilidades: Recibe los datos de un nuevo artista y lo da de alta en el sistema.

Precondiciones: No existe un artista con el mismo nombre.

Postcondiciones: Da de alta una nueva instancia de la clase Artist.

Salida: Se muestra un mensaje diciendo que el artista ha sido dado de alta correctamente.

Caso de uso: Modificación artista

Operación: modificarArtista

Responsabilidades: Recibe los datos actualizados de un artista existente y los sustituye.

Precondiciones: Existe el artista que se va a modificar.

Postcondiciones: Modifica la instancia de la clase Artist.

Salida: Se muestra un mensaje diciendo que el artista ha sido actualizado correctamente.

Caso de uso: Baja artista

Operación: bajaArtista

Responsabilidades: Elimina del sistema un artista.

Precondiciones: Existe el artista que se va a dar de baja.

Postcondiciones: Da de baja una instancia de la clase Artist.

Salida: Se muestra un mensaje diciendo que el usuario ha sido dado de baja correctamente.

Caso de uso: Visualización artista

Operación: visualizarArtista

Responsabilidades: Muestra por pantalla la información de un artista del sistema.

Precondiciones: Existe el artista que se va a visualizar.

Postcondiciones: Muestra la información del artista.

Salida: Se muestra por pantalla la información del artista.

Caso de uso: Visualización lista de artistas

Operación: visualizarListaArtistas

Responsabilidades: Muestra por pantalla la lista de artistas del sistema.

Precondiciones: ---

Postcondiciones: Muestra la lista de los artistas del sistema.

Salida: Se muestra por pantalla la lista de los artistas.

5.4. Modelo de estados

Tanto los artistas como los álbumes, imágenes, posts y fotógrafos pueden estar publicados o no, y los posts además pueden ser promocionados a la página principal.

En la se muestra el diagrama de estados para los posts, que es el más completo.

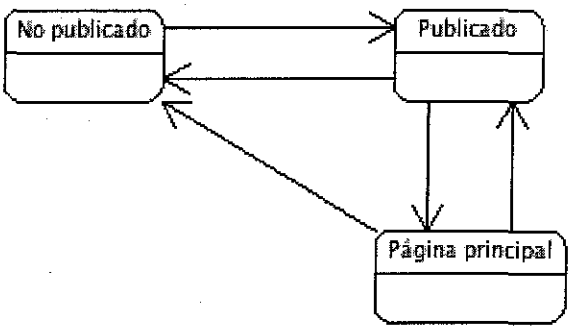


Figura 5.10: Diagrama de estados

6. Diseño

Una vez finalizada la etapa de especificación del proyecto, el siguiente paso es la fase de diseño. En esta fase se estudiará y decidirá cómo se va a resolver el problema partiendo del análisis del capítulo anterior. Por tanto, a continuación se analizarán las diversas maneras de abordar la solución al problema. Hay que recordar que esta solución puede no ser la única solución válida.

Antes de empezar a escribir código se debe pensar en ciertas cosas como:

- ¿Cuál es la mejor arquitectura para el sistema?
- ¿Cuál es la mejor herramienta para desarrollarlo?
- ¿Cómo diseñar la base de datos más óptima?
- ¿Cómo se diseñan las clases fundamentales?

También se debe tener en cuenta que el análisis no debe satisfacer tan solo las necesidades actuales sino que debe estar preparada para posibles cambios futuros que el cliente pueda desear, sin tener que reescribir todo el código. Esto implica que debe ser flexible.

6.1. Arquitectura

Antes de empezar con la definición de la arquitectura del sistema y establecer por tanto su estructura, es necesario identificar las propiedades que deberá tener el sistema. Las propiedades de la arquitectura son las siguientes:

- **Cambiable:** Capacidad de ser extensible, portable, reestructurable, mantenible, etc.
- **Interoperable:** Capacidad de dos o más entidades de intercambiar funcionalidades y datos.
- **Reusable:** Capacidad de aprovechar lo que se tiene para conseguir lo que se quiere.
- **Eficiente:** Buen uso de los recursos hardware, alto rendimiento y bajos tiempos de respuesta.
- **Fiable:** Capacidad de tolerancia a fallos, robustez frente a usos incorrectos, pérdidas de conexión y tratamiento de errores inesperados.
- **Probable:** Facilidad para ser testeado.
- **Integridad conceptual:** Armonía, simetría y predictibilidad.

- **Usabilidad:** Facilidad para ser utilizado.

Existen distintos tipos de estilos o patrones arquitectónicos. Sin embargo, prácticamente todos son clasificables entre los siguientes:

1. **Centrado en datos:** Arquitectura especialmente útil en sistemas que requieren acceso a datos compartidos. Estos datos se guardan en un sitio a modo de repositorio. Los clientes se conectan a este repositorio para acceder y actualizar los datos.
2. **Flujo de datos:** Este estilo arquitectónico se acostumbra a emplear para sistemas que, dados unos datos, realizan una serie de transformaciones para obtener datos de salida. Dentro de este estilo suele distinguirse dos tipos de patrones de datos.
    a. **Tuberías y filtros:** Este patrón se compone de un grupo de procesos o subsistemas (denominados filtros) que están conectados por tuberías por donde transmiten los datos de un componente al siguiente. Cada filtro trabaja independientemente de aquellos que se encuentran en el flujo de entrada o de salida.
    b. **Secuencia por lotes:** Este patrón descompone el sistema en una secuencia de programas independientes que transmiten los datos entre ellos. Se orienta a procesos que se ejecutan de forma diferida.
3. **Llamada y retorno:** Este es el patrón arquitectónico más utilizado y que se emplea tradicionalmente en grandes sistemas software. Se distinguen 3 tipos:
    a. **Programa principal y subrutina:** Es el típico estilo de la programación tradicional resultado de un análisis y diseño descendente. Los componentes se agrupan en programa principal y subrutinas. Se favorece la *cambiabilidad* pero se desfavorece la eficiencia. Una subrutina es un componente que al recibir el control y los datos de uno de sus padres, lo transmite hacia sus hijos. El retorno del control se hace en sentido contrario.
    b. **Orientación a objetos:** En este estilo se encapsulan los datos y mecanismos necesarios para manipularlos. Un subconjunto de estos mecanismos se pone a disposición del resto de componentes como servicios públicos. Este estilo también facilita la *cambiabilidad* y reusabilidad pero desfavorece la eficiencia.
    c. **Por capas**: Los componentes de este patrón se organizan en forma de capas. Cada componente únicamente puede comunicarse con componentes de la misma capa o de capas contiguas. Este patrón, al igual que los anteriores facilita la *cambiabilidad* y *portabilidad* pero desfavorece la eficiencia.
4. **Modelo – Vista – Controlador**: Este patrón se utiliza en sistemas software interactivos con interfaces de usuario muy flexibles, donde la información proporcionada por el sistema y su comportamiento tienen que reflejar inmediatamente las manipulaciones de la información. También son útiles cuando se quiere mostrar la misma información en varias pantallas o ventanas. La aplicación de este patrón suele resultar bastante ineficiente en recursos.

Tal y como se puede deducir de las arquitecturas comentadas, no existe ninguna que favorezca la eficiencia a la vez que la *cambiabilidad* y *portabilidad*.

Por tanto y dado que no existe ninguna solución que se adapte a la perfección a las necesidades, **se ha optado por una solución hibrida** basada en el patrón arquitectónico por capas:

- **Capa de presentación:** Responsable de la presentación al usuario y de la interacción con el mismo.
- **Capa de dominio:** Responsable de la implementación de las funcionalidades y de la lógica del sistema.
- **Capa de datos:** Responsable de la comunicación con el *SGBD.*

Dado que el sistema debe poder ser accesible por más de un usuario simultáneamente el sistema debe estar distribuido. Para llevar a cabo esta distribución se ha utilizado arquitectura de cliente/servidor. Esta arquitectura se compone de:

- **Servidor:** Es un ordenador que lleva a cabo un servicio que habitualmente requiere gran potencia de procesamiento, memoria o espacio en disco.
- **Cliente:** Es un ordenador que solicita los servicios que proporciona el servidor y también lleva a cabo algún tipo de procesamiento.
- **Software intermedio:** Es una capa de software cuya función es la de hacer transparente la comunicación entre servidor y cliente.

El conjunto de sistemas de información cliente/servidor que utilizan las tecnologías web son denominados habitualmente aplicaciones web.

Existen tres modelos arquitectónicos de aplicaciones web según el libro de UML (CONALLEN, 1999):

- **Thin web client:** Este esquema solo necesita un explorador estándar con capacidad para presentar formularios en HTML, links y botones de envío de formularios. Toda la lógica del sistema reside y es ejecutada en el servidor y el cliente solo recibe la capa de presentación en forma de página HTML. Los únicos protocolos de conexión entre el cliente y el servidor son HTTP y HTTPS.
- **Thick web client:** Este patrón amplía al anterior permitiendo al cliente ejecutar parte de la lógica de la aplicación. Esto se consigue gracias a la utilización de HTML dinámico, applets java o controles ActiveX que se incrustan en las páginas que recibe el cliente en formato HTML o XML. De todas maneras, la comunicación con el servidor sigue haciéndose mediante los protocolos HTTP o HTTPS.
- **Web delivery:** Estas aplicaciones utilizan además de HTTP otros protocolos para comunicarse, como RMI, IIOP, etc. Estos protocolos se utilizan para hacer invocaciones sobre objetos remotos.

Una vez planteados los posibles patrones web, **se ha decidido optar por el patrón *Thick web client***, teniendo en cuenta la arquitectura en tres capas que se decidió utilizar anteriormente. De este modo, la capa de de datos y la capa de dominio residirán físicamente en el servidor y la capa de presentación se distribuye entre el cliente y el servidor. **Para cierta parte del proyecto, sin embargo, se ha utilizado el modelo *Web delivery*.**

6.2. Capa de presentación

La capa de presentación se encarga de recoger todos los datos proporcionados por el usuario, pedir al sistema que procese sus peticiones y devolverle la respuesta.

El diseño de esta capa se podría dividir en dos fases:

1. **Diseño interno**: Es donde se diseñan los mecanismos que recogen los datos proporcionados por los usuarios, se procesan sus peticiones y se les da respuesta. Estos mecanismos permiten la comunicación entre la interfaz de usuario y las reglas de negocio, transmitiendo las peticiones del usuario y facilitando los datos que, en cada momento, da este.

2. **Diseño Externo:** Consiste en el diseño de todo lo que el usuario podrá ver y todo con lo que podrá interactuar. Es importante procurar que el diseño permita que la información se presente de manera agradable, atractiva y de fácil lectura, simplificando así las tareas de los comerciales y atrayendo a los clientes. El resultado son las diferentes pantallas de la aplicación web, algunas de las cuales se pueden ver en el manual de usuario incluido en la memoria.

6.3. Capa de dominio

La capa de dominio es la capa intermedia encargada de llevar a cabo la lógica del negocio y realizar todos los cálculos. Esta capa conoce cómo resolver las peticiones de los usuarios pero ignora dónde se guarda la información y como se presenta al usuario.

Las tareas de las que es responsable la capa son las siguientes:

- Recoger las órdenes del usuario procedentes de la capa superior (capa de presentación)
- Ejecutar estas órdenes recibidas, realizando para ello los cálculos necesarios y modificando el estado de la aplicación
- Consultar y actualizar el estado de los datos del sistema a través de la comunicación con la capa inferior (capa de datos)

Las clases de esta parte del sistema se han podido observar en el modelo conceptual de la especificación (sección 5.2). Aún así, en la etapa del diseño ya hay que tener en cuenta la arquitectura sobre la que se basará el sistema. En este caso, hay que adaptar el modelo a la infraestructura que proporciona Drupal. Eso hace que aparezcan otras clases, que hay que tener en cuenta.

La principal de ellas, y la única que tendremos en cuenta aquí, es la clase Node. Como se ha explicado en la sección 7.9, Drupal identifica como nodos todos los tipos de contenido. Por lo tanto, lo único que debemos añadir al diagrama de clases es la clase Node y su relación con la clase User, ya que un nodo siempre tiene un autor. Además, aprovecharemos para mostrar que el usuario de tipo Musician está asociado a uno o más artistas, de los que puede modificar sus datos. Se pueden ver estos cambios en la Figura 6.1.

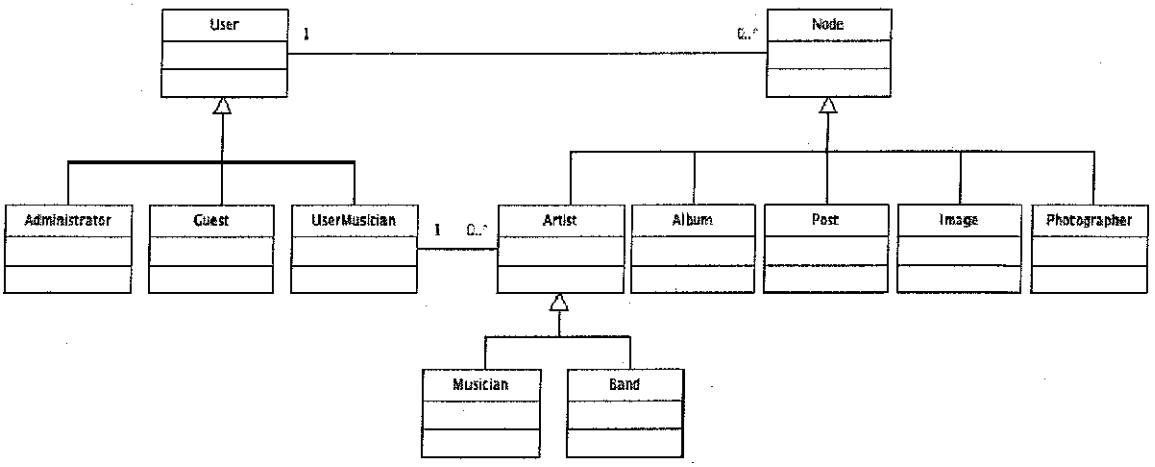


Figura 6.1: Clases adicionales al diagrama

6.4. Capa de datos

Esta capa interrelaciona las reglas de negocio con el sistema de gestión de bases de datos. Su diseño es complejo, pues abarca un número importante de decisiones. Dada su complejidad lo mejor es dividir el problema.

Por esto, realizaremos el diseño de la base de datos mediante la suma de diferentes componentes: modelo conceptual, modelo lógico y modelo físico.

6.4.1. Modelo Conceptual

Este modelo consiste en una descripción a alto nivel de la estructura de la base de datos sin tener en cuenta el Sistema Gestor de Base de Datos que se vaya a utilizar más adelante.

Permite ver de manera gráfica todas las relaciones que hay derivadas de la definición del análisis de requerimientos realizado anteriormente.

Basándonos en nuestro modelo conceptual podemos realizar el diseño de nuestra base de datos de una forma mucho más sencilla.

El modelo conceptual del sistema puede verse en la figura de la sección 5.2, a la que hay que añadir los últimos cambios de la Figura 6.1.

6.4.2. Modelo lógico

El modelo lógico se basa en el modelo conceptual para dar como resultado el esquema lógico.

Este esquema es una descripción de la estructura de la base de datos en términos que pueda procesar un *SGBD*.

Por tanto, el diseño lógico depende del tipo de *SGBD* que se vaya a utilizar y en nuestro caso también del gestor de contenido utilizado en este proyecto , como se ha comentado con anterioridad, el sistema utilizado Drupal condiciona la representación lógica y física de la capa de datos, pues el sistema tiene clases y relaciones necesarias para su funcionamiento estándar como CMS, con los cuales el modelo propio de la solución construida no tiene más remedio que relacionarse. No obstante, no se va a especificar aquí esa parte de la capa de datos que es propia de Drupal. A pesar de ello, en la sección 8.8, que hace referencia a la sincronización entre MLE y RW se especifica la parte de la capa de datos propia del CMS con que es necesario interaccionar para poder realizar la sincronización.

En la Figura 6.2 y la Figura 6.3 se muestran algunas de las tablas (a modo de ejemplo) que conforman el modelo lógico del sistema, concretamente la de artistas (sirve tanto para músicos como para bandas) y la de álbumes.

	Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
☐	aid	int(11)		UNSIGNED	No		auto_increment	
☐	nid	int(11)		UNSIGNED	No			
☐	names	varchar(255)	utf8_general_ci		No			
☐	picture	varchar(255)	utf8_general_ci		Si	*NULL*		
☐	website	varchar(255)	utf8_general_ci		Si	*NULL*		
☐	myspace	varchar(255)	utf8_general_ci		Si	*NULL*		
☐	is_band	tinyint(1)			No	0		

Figura 6.2: Tabla `catalogue_artist`

	Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
☐	alid	int(11)		UNSIGNED	No		auto_increment	
☐	nid	int(11)		UNSIGNED	No	0		
☐	lid	int(11)		UNSIGNED	No	0		
☐	aid	int(11)		UNSIGNED	No	0		
☐	title	varchar(255)	utf8_general_ci		No			
☐	year	year(4)			No	0000		
☐	cover_path	varchar(255)	utf8_general_ci		No			
☐	genre	varchar(255)	utf8_general_ci		No			
☐	city	varchar(255)	utf8_general_ci		No			
☐	buy_url	varchar(255)	utf8_general_ci		No			

Figura 6.3: Tabla `catalogue_album`

6.4.3. Modelo físico

El modelo físico es la aplicación del modelo lógico a un *SGBD* determinado. En la aplicación web de este proyecto se decidió utilizar *MySQL* como Sistema Gestor de Bases de Datos.

El resultado de este modelo es un código SQL, dado que cada tabla que aparece en el modelo lógico se traduce a una sentencia SQL.

A continuación se muestran las sentencias SQL de creación de tablas del fichero `catalogue.install` que, como se ha explicado en la sección 7.9.3, contiene todas las instrucciones SQL que se ejecutan en el momento de instalar el módulo desarrollado, así como a la hora de desinstalarlo.

```
CREATE TABLE `catalogue_artist` (
        `aid` int(11) unsigned NOT NULL auto_increment,
        `nid` int(11) unsigned NOT NULL,
        `names` varchar(255) NOT NULL default '',
        `website` varchar(255) default NULL,
        `myspace` varchar(255) default NULL,
        `is_band` tinyint(1) NOT NULL default '0',
        PRIMARY KEY  (`aid`),
        FULLTEXT KEY `names` (`names`)
)

CREATE TABLE `catalogue_album` (
        `alid` int(11) unsigned NOT NULL auto_increment,
        `nid` int(11) unsigned NOT NULL default '0',
        `lid` int(11) unsigned NOT NULL default '0',
        `aid` int(11) unsigned NOT NULL default '0',
        `title` varchar(255) NOT NULL default '',
        `year` year(4) NOT NULL,
        `cover_path` varchar(255) NOT NULL default '',
        `genre` varchar(255) NOT NULL default '',
        `city` varchar(255) NOT NULL default '',
        `buy_url` varchar(255) NOT NULL default '',
        PRIMARY KEY  (`alid`),
        KEY `nid` (`nid`),
```

```
        FULLTEXT KEY `title` (`title`)
)

CREATE TABLE `catalogue_label` (
        `lid` int(11) NOT NULL auto_increment,
        `label` varchar(250) NOT NULL default '',
        `description` varchar(250) NOT NULL default '',
        `website` varchar(250) NOT NULL default '',
        PRIMARY KEY  (`lid`)
)

CREATE TABLE `catalogue_photographer` (
        `nid` int(11) unsigned NOT NULL default '0',
        `names` varchar(255) NOT NULL default '',
        `website` varchar(255) default NULL,
        PRIMARY KEY  (`nid`)
)

CREATE TABLE 'catalogue_instrument' (
        'iid' int(10) unsigned NOT NULL auto_increment,     /* Instrument ID */
        'instrument' varchar(255) NOT NULL default '',
        'abbreviation' varchar(4) default '',
        PRIMARY KEY ('iid')
)

CREATE TABLE 'subwidget_links' (
        'swid' int(11) unsigned NOT NULL AUTO_INCREMENT,
        'wid' int(11) unsigned NOT NULL default '0' REFERENCES widget('wid') ON DELETE CASCADE,
        'link' varchar(255) NOT NULL default '',
        'description' varchar(255) NOT NULL default '',
        'weight' tinyint(4) default '0',
        PRIMARY KEY ('swid')
)

CREATE TABLE 'subwidget_instruments' (
        'swid' int(11) unsigned NOT NULL AUTO_INCREMENT,
        'wid' int(11) unsigned NOT NULL default '0' REFERENCES widget('wid') ON DELETE CASCADE,
        'iid' int(11) unsigned NOT NULL default '0' REFERENCES catalogue_artist('aid'),
        'weight' tinyint(4) default '0',
        PRIMARY KEY ('swid')
)

CREATE TABLE 'subwidget_band_performers' (
        'swid' int(11) unsigned NOT NULL AUTO_INCREMENT,
        'wid' int(11) unsigned NOT NULL default '0' REFERENCES widget('wid') ON DELETE CASCADE,
        'aid' int(11) unsigned NOT NULL default '0' REFERENCES catalogue_artist('aid') ON DELETE,
```

```
        'iid' int(11) unsigned NOT NULL default '0' REFERENCES catalogue_artist('aid'),
        'weight' tinyint(4) default '0',
        'band_leader' tinyint(1) NOT NULL default '0',
        PRIMARY KEY ('swid')
)

CREATE TABLE 'subwidget_album_performers' (
        'swid' int(11) unsigned NOT NULL AUTO_INCREMENT,
        'wid' int(11) unsigned NOT NULL default '0' REFERENCES widget(wid) ON DELETE CASCADE,
        'aid' int(11) unsigned NOT NULL default '0' REFERENCES catalogue_artist(aid) ON DELETE,
        'iid' int(11) unsigned NOT NULL default '0' REFERENCES catalogue_artist(aid),
        'interviene' varchar(256) default '',
        'weight' tinyint(4) default '0',
        'band_leader' tinyint(1) NOT NULL default '0',
        PRIMARY KEY ('swid')
)

CREATE TABLE `catalogue_theme` (
        `tid` int(11) unsigned NOT NULL auto_increment,
        `alid` int(11) unsigned NOT NULL default '0',
        `title` varchar(255) NOT NULL default '',
        `path` varchar(255) NOT NULL default '',
        `track` int(11) NOT NULL default '0',
        `fecha` int(11) NOT NULL,
        PRIMARY KEY  (`tid`)
)
```

Para ver gráficamente las tablas y su relación, en el diagrama de la Figura 6.4 está definido el modelo físico de la base de datos para las tablas más relevantes del proyecto.

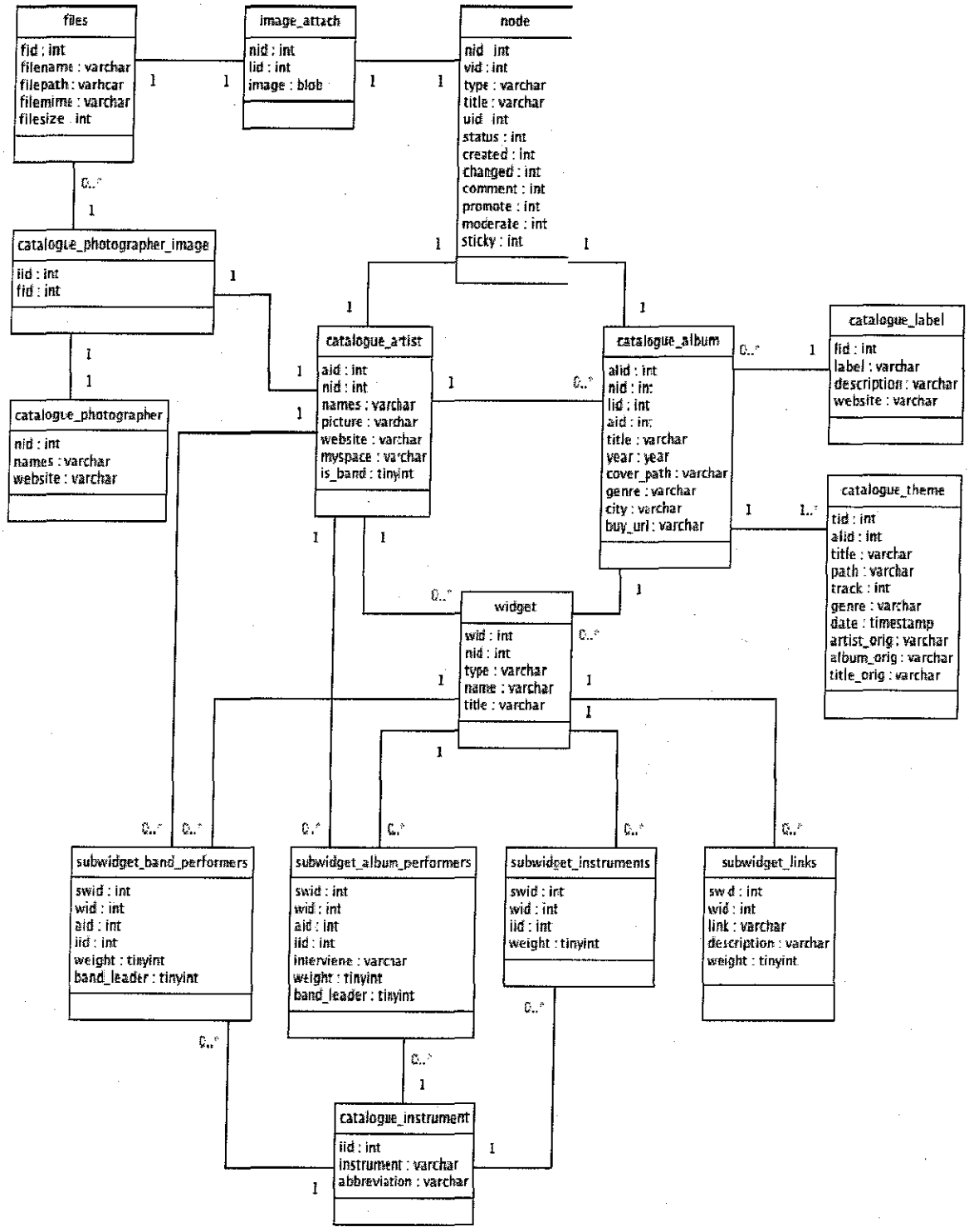


Figura 6.4: Modelo físico de datos

Se observan algunos elementos nuevos:

- Los nodos con los que trabaja Drupal.
- Los archivos (files) y las imágenes, que se gestionan con un módulo de Drupal que las adjunta a los nodos.

- Los widgets y subwidgets, que provienen de un módulo de Drupal desarrollado para este PFC. Se hablará detenidamente de ellos en el capítulo de implementación (8.5), pero podemos adelantar que sirven para añadir a los nodos listas de tamaño variable de un cierto tipo de *ítems* (en nuestro caso los ítems serán de 4 tipos distintos: instrumentos asociados a un músico , formación asociada a una banda , formación asociada a un álbum , links adicionales asociados a un músico o banda),

7.Sistemas gestores de contenidos (CMS)

7.1. Introducción

Antes de hablar propiamente de los CMS habría que decir que para el proyecto actual no se tenía constancia de que existiera ningún software estándar que resolviera las necesidades de la aplicación web que se iba a elaborar. Como se partía de la radio de Live365, se observaron otras radios similares, tanto de jazz como no de jazz. Algunas de ellas no tienen ni siquiera una web de soporte, sino que utilizan el mismo perfil de Live365 como lugar de presentación. De las que poseen web propia, principalmente las más serias, lo que tienen es un acceso a la radio y algo de información sobre algunos artistas y álbumes. Las que más profundizan en la radio muestran horarios de emisión, playlists, etc. Pero no vimos ninguna que tuviera una base de datos con el catálogo completo de los artistas y los álbumes emitidos como aquí se pretende, y mucho menos enlazada con la música que suena en ese momento.

Realizar un sitio web como éste puede ser un trabajo complicado y muy laborioso si no se dispone de las herramientas adecuadas. En el pasado las herramientas eran básicamente editores que permitían generar una página, que evolucionaron para incorporar el control de la estructura de la web y otras funcionalidades, pero en general estaban enfocadas más a la creación que al mantenimiento. En los últimos años se ha desarrollado el concepto de sistema de gestión de contenidos (Content Management Systems o CMS). Se trata de herramientas que permiten crear y mantener un web con facilidad, encargándose de los trabajos más tediosos que hasta ahora ocupaban el tiempo de los administradores de las webs.

Teniendo en cuenta el ahorro que supone la utilización de estas herramientas, y el coste de desarrollarlas, sería lógico esperar que su precio fuera muy elevado. Eso es cierto para algunos productos comerciales, pero existen potentes herramientas de gestión de contenidos de acceso libre, disponibles bajo licencias de código abierto.

Los gestores de contenidos proporcionan un entorno que posibilita la actualización, mantenimiento y ampliación de la web con la colaboración de múltiples usuarios. En cualquier entorno virtual ésta es una característica importante, que además puede ayudar a crear una comunidad cohesionada que participe más de forma conjunta.

Como consecuencia de ello se planteó para este proyecto una solución basada en el desarrollo sobre una plataforma CMS que, por las características que requeríamos (inserción de los elementos del Catálogo en la BD, creación de posts e imágenes relacionados con ellos, etc.), parece lo más conveniente. Se descarta, pues, un desarrollo a medida partiendo de cero que requeriría mucho tiempo montando una estructura base que nos alejaría de las partes realmente importantes del proyecto. Aún así, se hará necesario desarrollar algunos módulos para extender la funcionalidad básica del CMS.

El único problema que puede suponer el uso de un CMS es a la hora de gestionar la inserción de datos partiendo directamente de los mp3, ya que el MLE (Music Library Editor) deberá acceder y modificar la base de datos de la misma forma que lo hace el CMS, para no aportar inestabilidad y datos contradictorios.

7.2. Concepto CMS

Los sistemas de gestión de contenidos (Content Management Systems o CMS) son elementos de software que se utilizan principalmente para facilitar la gestión de webs, ya sea en Internet o en una intranet, y por eso también son conocidos como gestores de contenido web (Web Content Management o WCM). Hay que tener en cuenta, sin embargo, que la aplicación de los CMS no se limita sólo a las webs.

Se propone una división de la funcionalidad de los sistemas de gestión de contenidos en cuatro categorías: creación de contenido, gestión de contenido, publicación y presentación.

7.2.1. Creación de contenido

Un CMS aporta herramientas para que los creadores sin conocimientos técnicos en páginas web puedan concentrarse en el contenido. Lo más habitual es proporcionar un editor de texto WYSIWYG, en el que el usuario ve el resultado final mientras escribe, al estilo de los editores comerciales, pero con un rango de formatos de texto limitado. Esta limitación tiene sentido, ya que el objetivo es que el creador pueda poner énfasis en algunos puntos, pero sin modificar mucho el estilo general del sitio web.

Hay otras herramientas como la edición de los documentos en XML, utilización de aplicaciones ofimáticas con las que se integra el CMS, importación de documentos existentes y editores que permiten añadir marcas, habitualmente HTML, para indicar el formato y estructura de un documento.

Un CMS puede incorporar una o varias de estas herramientas, pero siempre tendría que proporcionar un editor WYSIWYG por su facilidad de uso y la comodidad de acceso desde cualquier ordenador con un navegador y acceso a Internet.

Para la creación del sitio propiamente dicho, los CMS aportan herramientas para definir la estructura, el formato de las páginas, el aspecto visual, uso de patrones, y un sistema modular que permite incluir funciones no previstas originalmente.

7.2.2. Gestión de contenido

Los documentos creados se depositan en una base de datos central donde también se guardan el resto de datos de la web, cómo son los datos relativos a los documentos (versiones hechas, autor, fecha de publicación y caducidad, etc.), datos y preferencias de los usuarios, la estructura de la web, etc.

La estructura de la web se puede configurar con una herramienta que, habitualmente, presenta una visión jerárquica del sitio y permite modificaciones. Mediante esta estructura se puede asignar un grupo a cada área, con responsables, editores, autores y usuarios con diferentes permisos. Eso es imprescindible para facilitar el ciclo de trabajo (workflow) con un circuito de edición que va desde el autor hasta el responsable final de la publicación. El CMS permite la comunicación entre los miembros del grupo y hace un seguimiento del estado de cada paso del ciclo de trabajo.

7.2.3. Publicación

Una página aprobada se publica automáticamente cuando llega la fecha de publicación, y cuando caduca se archiva para futuras referencias. En su publicación se aplica el patrón definido para toda la web o para la sección concreta donde está situada, de forma que el resultado final es un sitio web con un aspecto consistente en todas sus páginas. Esta

separación entre contenido y forma permite que se pueda modificar el aspecto visual de un sitio web sin afectar a los documentos ya creados y libera a los autores de preocuparse por el diseño final de sus páginas.

7.2.4. Presentación

Un CMS puede gestionar automáticamente la accesibilidad del web, con soporte de normas internacionales de accesibilidad como WAI, y adaptarse a las preferencias o necesidades de cada usuario. También puede proporcionar compatibilidad con los diferentes navegadores disponibles en todas las plataformas (Windows, Linux, Mac, Palm, etc.) y su capacidad de internacionalización lo permite adaptarse al idioma, sistema de medidas y cultura del visitante.

El sistema se encarga de gestionar muchos otros aspectos como son los menús de navegación o la jerarquía de la página actual dentro del web, añadiendo enlaces de forma automática. También gestiona todos los módulos, internos o externos, que incorpore al sistema. Así por ejemplo, con un módulo de noticias se presentarían las novedades aparecidas en otro web, con un módulo de publicidad se mostraría un anuncio o mensaje animado, y con un módulo de foro se podría mostrar, en la página principal, el título de los últimos mensajes recibidos. Todo eso con los enlaces correspondientes y, evidentemente, siguiendo el patrón que los diseñadores hayan creado.

7.3. Necesidad de un CMS

En el apartado anterior se han presentado bastantes motivos para ver la utilidad de un sistema que gestione un entorno web, pero se podría pensar que no es necesario para un web relativamente pequeño o cuando no se necesitan tantas funcionalidades. Eso sólo podría ser cierto para un web con unas pocas páginas estáticas para el que no se prevea un crecimiento futuro ni muchas actualizaciones, lo que no es muy realista. En cualquier otro caso, la flexibilidad y escalabilidad que permiten estos sistemas justifican su utilización en prácticamente cualquier tipo de web.

Muchos usuarios particulares utilizan CMS gratuitos para elaborar y gestionar sus webs personales, obteniendo webs dinámicos llenos de funcionalidades. El resultado que obtienen es superior al de algunas empresas que se limitan a tener páginas estáticas que no aportan ningún valor añadido.

Éstos son algunos de los puntos más importantes que hacen útil y necesaria la utilización de un CMS:

- *Inclusión de nuevas funcionalidades en el sitio web.* Esta operación puede implicar la revisión de multitud de páginas y la generación del código que aporta las funcionalidades. Con un CMS eso puede ser tan simple como incluir un módulo realizado por terceros, sin que eso suponga muchos cambios en la web. El sistema puede crecer y adaptarse a las necesidades futuras.
- *Mantenimiento de gran cantidad de páginas.* En una web con muchas páginas hace falta un sistema para distribuir los trabajos de creación, edición y mantenimiento con permisos de acceso a las diferentes áreas. También se tienen que gestionar los metadatos de cada documento, las versiones, la publicación y caducidad de páginas y los enlaces rotos, entre otros aspectos.
- *Reutilización de objetos o componentes.* Un CMS permite la recuperación y reutilización de páginas, documentos, y en general de cualquier objeto publicado o almacenado.
- *Páginas interactivas.* Las páginas estáticas llegan al usuario exactamente como están almacenadas en el servidor web. En cambio, las páginas dinámicas no existen en el servidor tal como se reciben en los navegadores, sino que se generan según las peticiones de los usuarios. De esta manera cuando por ejemplo se utiliza un

buscador, el sistema genera una página con los resultados que no existían antes de la petición. Para conseguir esta interacción, los CMS conectan con una base de datos que hace de repositorio central de todos los datos de la web.

- *Cambios del aspecto de la web*. Si no hay una buena separación entre contenido y presentación, un cambio de diseño puede comportar la revisión de muchas páginas para su adaptación. Los CMS facilitan los cambios con la utilización, por ejemplo, del estándar CSS (Cascading Style Sheets u hojas de estilo en cascada) con lo que se consigue la independencia de presentación y contenido.
- *Consistencia de la web*. La consistencia en un web no quiere decir que todas las páginas sean iguales, sino que hay un orden (visual) en vez de caos. Un usuario nota enseguida cuándo una página no es igual que el resto de las de la misma web por su aspecto, la disposición de los objetos o por los cambios en la forma de navegar. Estas diferencias provocan sensación de desorden y dan a entender que la web no ha sido diseñada por profesionales. Los CMS pueden aplicar un mismo estilo en todas las páginas con el mencionado CSS, y aplicar una misma estructura mediante patrones de páginas.
- *Control de acceso*. Controlar el acceso a un web no consiste simplemente al permitir la entrada al web, sino que comporta gestionar los diferentes permisos a cada área del web aplicado a grupos o individuos.

7.4. CMS comerciales y de código abierto

Se puede hacer una primera división de los CMS según el tipo de licencia escogido. Por una parte están los CMS comercializados por empresas que consideran el código fuente un activo más que tienen que mantener en propiedad, y que no permiten que terceros tengan acceso a él. Por la otra tenemos los de código fuente abierto, desarrollados por individuos, grupos o empresas que permiten el acceso libre y la modificación del código fuente.

La disponibilidad del código fuente posibilita que se hagan personalizaciones del producto, correcciones de errores y desarrollo de nuevas funciones. Este hecho es una garantía de que el producto podrá evolucionar incluso después de la desaparición del grupo o empresa creadora.

Algunas empresas también dan acceso al código, pero sólo con la adquisición de una licencia especial o después de su desaparición. Generalmente las modificaciones sólo pueden hacerlas los mismos desarrolladores, y siempre según sus prioridades.

Los CMS de código abierto son mucho más flexibles en este sentido, pero se podría considerar que la herramienta comercial será más estable y coherente al estar desarrollada por un mismo grupo. En la práctica esta ventaja no es tan grande, ya que los CMS de código abierto también están coordinados por un único grupo o por empresas, de forma similar a los comerciales.

Utilizar una herramienta de gestión de contenidos de código abierto tiene otra ventaja que hace decidirse a la mayoría de usuarios: su coste. Habitualmente todo el software de código abierto es de acceso libre, es decir, sin ningún coste en licencias. Sólo en casos aislados se hacen distinciones entre empresas y entidades sin ánimo de lucro o particulares. En comparación, los productos comerciales pueden llegar a tener un coste que sólo una gran empresa puede asumir.

En cuanto al soporte, los CMS comerciales acostumbran a dar soporte profesional, con un coste elevado en muchos casos, mientras que los de código abierto se basan más en las comunidades de usuarios que comparten información y solución a los problemas. Las formas de soporte se pueden mezclar, y así encontramos CMS de código abierto con empresas que ofrecen servicios de valor añadido y con activas comunidades de usuarios. En el caso comercial también sucede, pero el coste de las licencias hace que el gran público

se decante por otras opciones y por lo tanto las comunidades de soporte son más pequeñas.

Un problema que acostumbra a tener el software de código abierto es la documentación, generalmente escasa, dirigida a usuarios técnicos o mal redactada. Este problema se agrava en el caso de los módulos desarrollados por terceros, que no siempre incorporan las instrucciones de su funcionamiento de forma completa y entendible.

En el mercado hay CMS de calidad tanto comerciales como de código abierto. Muchos CMS de código abierto están poco elaborados (aunque en plena evolución), pero también lo encontramos entre los comerciales. En definitiva, un buen CMS de código abierto es mucho más económico que su homólogo comercial, con la ventaja de disponer de todo el código fuente y de una extensa comunidad de usuarios.

Por todos estos motivos, y como apuesta por la filosofía del software libre, en este trabajo sólo se presentan algunos CMS de código abierto.

7.5. Presente y futuro de los CMS

En la actualidad, a parte de la ampliación de las funcionalidades de los CMS, uno de los campos más interesantes es la incorporación de estándares que mejoran la compatibilidad de componentes, facilitan el aprendizaje al cambiar de sistema y aportan calidad y estabilidad.

Algunos de estos estándares son: CSS, que permite la creación de hojas de estilo; XML, un lenguaje de marcas que permite estructurar un documento; XHTML, que es un subconjunto del anterior orientado a la presentación de documentos vía web; WAI, que asegura la accesibilidad del sistema; y RSS, para sindicar contenidos de tipo noticia. También las aplicaciones que rodean los CMS acostumbran a ser estándar (de facto), como los servidores web Apache e ISS; los lenguajes PHP, Perl y Python; y las bases de datos MySQL y PostgreSQL. La disponibilidad para los principales sistemas operativos de estas aplicaciones y módulos, permite que los CMS puedan funcionar en diversas plataformas sin muchas modificaciones.

Sobre el futuro de los CMS, se apunta que:

- Los CMS se convertirán en un artículo de consumo, cuando los productos se hayan establecido y más soluciones lleguen al mercado. Eso provocará una disminución de los precios en los productos comerciales y una mayor consistencia en las funcionalidades que ofrecen.
- En este entorno, muchas empresas que implementan webs tendrán que cerrar.
- Muchos proyectos fracasarán por no ajustarse a los estándares y no entender conceptos como usabilidad, arquitectura de la información, gestión del conocimiento y contenido.
- El campo de los gestores de contenido madurará hasta conseguir un alto grado de consistencia y profesionalismo.
- Se adoptarán estándares en el almacenaje, estructuración y gestión del contenido.
- Se producirá una fusión entre gestión de contenidos, gestión de documentos y gestión de registros.

También se puede añadir la incorporación de sistemas de e-learning y gestión del conocimiento, y en los entornos de intranet corporativa, la posibilidad de acceder a otras fuentes de datos como por ejemplo sistemas de soporte de decisiones (*Decision Support Systems* o DSS). El campo de los CMS de código abierto tendría que seguir un desarrollo similar.

7.6. Criterios de selección

Antes de empezar el proceso de selección de un CMS concreto, hay que tener claros los objetivos de la web, teniendo en cuenta al público destinatario, y estableciendo una serie de requerimientos que tendría que poder satisfacer el CMS.

La siguiente lista está basada en las funciones principales de los CMS expuestas anteriormente y una recopilación de los requerimientos básicos de una web:

- **Código abierto.** Por los motivos mencionados anteriormente, el CMS tendría que ser de código fuente abierto (o libre).
- **Arquitectura técnica.** Tiene que ser fiable y permitir la escalabilidad del sistema para adecuarse a futuras necesidades con módulos. También tiene que haber una separación de los conceptos de contenido, presentación y estructura que permita la modificación de uno de ellos sin afectar a los otros. Es recomendable, pues, que se utilicen hojas de estilo (CSS) y patrones de páginas.
- **Grado de desarrollo.** Madurez de la aplicación y disponibilidad de módulos que le añaden funcionalidades.
- **Soporte.** La herramienta tiene que tener soporte tanto por parte de los creadores como por otros desarrolladores. De esta manera se puede asegurar de que en el futuro habrá mejoras de la herramienta y que se podrá encontrar respuesta a los posibles problemas.
- **Posición en el mercado y opiniones.** Una herramienta poco conocida puede ser muy buena, pero hay que asegurarse de que tiene un cierto futuro. También son importantes las opiniones de los usuarios y de los expertos.
- **Usabilidad.** La herramienta tiene que ser fácil de utilizar y aprender. Los usuarios no siempre serán técnicos, por lo tanto hace falta asegurar que podrán utilizar la herramienta sin muchos esfuerzos y sacarle el máximo rendimiento.
- **Accesibilidad.** Para asegurar la accesibilidad de una web, el CMS tendría que cumplir un estándar de accesibilidad. El más extendido es WAI (*Web Accessibility Initiative*) del World Wide Web Consortium.
- **Velocidad de descarga.** Teniendo en cuenta que no todos los usuarios disponen de líneas de alta velocidad, las páginas se tendrían que cargar rápidamente o dar esa opción.
- **Funcionalidades.** No se espera que todas las herramientas ofrezcan todas las funcionalidades, ni que éstas sean las únicas que tendrá finalmente la web. Entre otras:
    - Editor de texto WYSIWYG a través del navegador. El administrador elige y el sistema procede en consecuencia.
    - Herramienta de búsqueda.
    - Comunicación entre los usuarios (foros, correo electrónico, chat).
    - Noticias.
    - Artículos.
    - Ciclo de trabajo (*workflow*) con diferentes perfiles de usuarios y grupos de trabajo.
    - Fechas de publicación y caducidad.
    - Webs personales.
    - Carga y descarga de documentos y material multimedia.

- Avisos de actualización de páginas o mensajes en los foros, y envío automático de avisos por correo electrónico.
- Envío de páginas por correo electrónico.
- Páginas en versión imprimible.
- Personalización según el usuario.
- Disponibilidad o posibilidad de traducción al catalán y al castellano.
- Soporte de múltiples formados (HTML, Word, Excel, Acrobat, etc.).
- Soporte de múltiples navegadores (Internet Explorer, Firefox, etc.).
- Soporte de sindicación (RSS, NewsML, etc.).
- Estadísticas de uso e informes.
- Control de páginas caducadas y enlaces rotos.

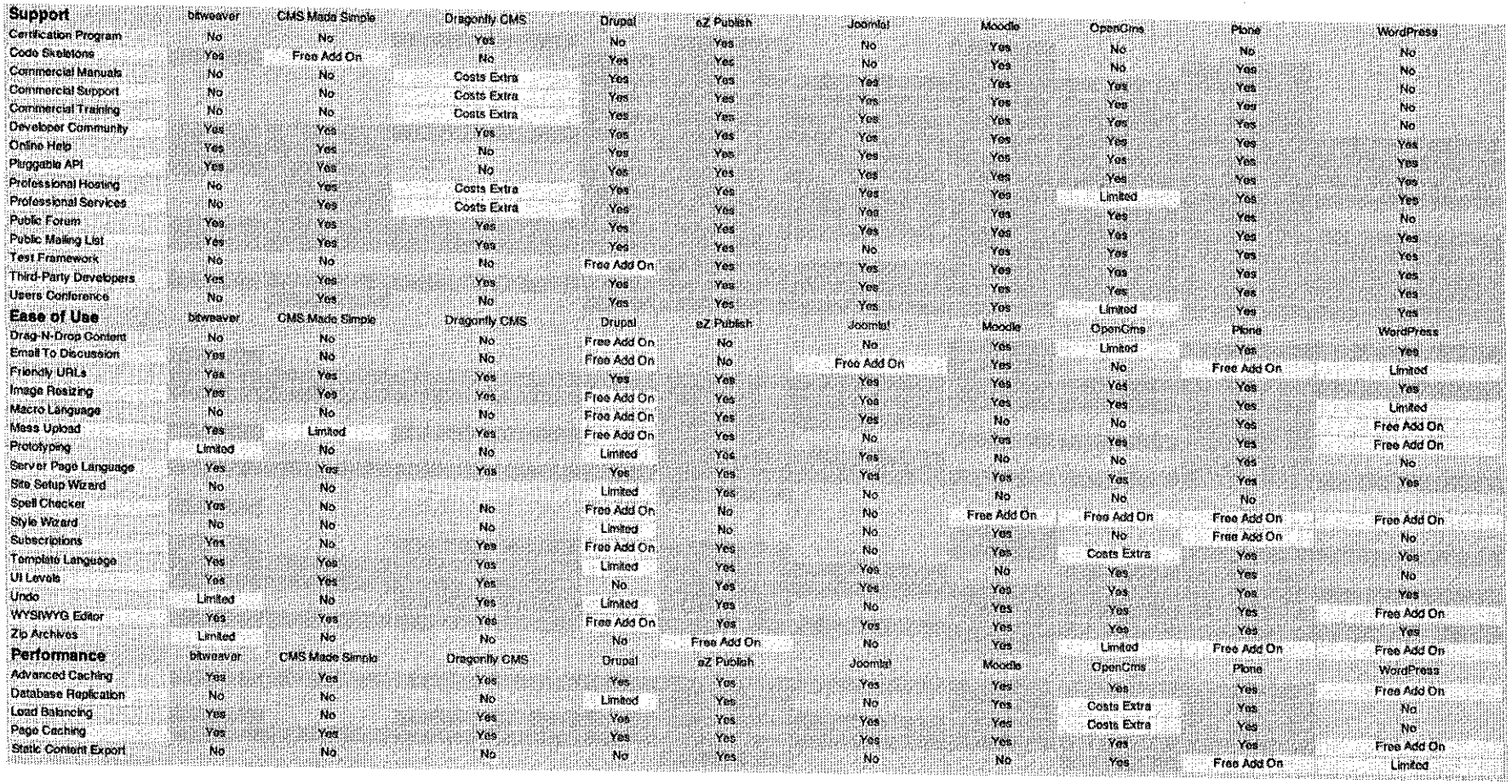
7.7. Estudio de los CMS

Existen decenas de CMS en el mercado, cada uno pensado para distintos usos, y es imposible conocerlos todos y hacer una comparativa exhaustiva. Por eso se han seleccionado algunos de los más conocidos o utilizados siguiendo opiniones de diversas fuentes, y mediante el servicio de comparación de CMS de www.cmsmatrix.com se ha obtenido la siguiente tabla. En ella se estudian, para cada CMS:

- Requisitos del sistema
- Seguridad
- Soporte
- Facilidad de uso
- Rendimiento
- Gestión
- Interoperabilidad
- Flexibilidad
- Aplicaciones incorporadas
- Capacidad para el e-Comercio

Product	bitweaver 2	CMS Made Simple 1.2.5	Dragonfly CMS 9.0.6.1	Drupal 6.2	eZ Publish 3.10 (PHP4) and 4.0 (PHP5)	Joomla! 1.5.3	Moodle 1.8	OpenCms 7.0.4	Plone 3.0	WordPress 2.2.1
Last Updated	3/20/2008	5/28/2008	8/24/2008	4/10/2008	11/19/2007	5/31/2006	9/12/2007	3/25/2008	9/8/2007	7/25/2007
System Requirements	bitweaver	CMS Made Simple	Dragonfly CMS	Drupal	eZ Publish	Joomla!	Moodle	OpenCms	Plone	WordPress
Application Server	PHP 4.1.0+		Apache recommended, any server that supports PHP and MySQL	PHP 4.3.5+	None	Any that supports PHP (Apache recommended)	PHP 4.3.3+	Tomcat, JBoss, Resin 3, Websphere 6	Zope	Apache recommended, any server that supports PHP and MySQL
Approximate Cost				Free		Free	0	Free	Free	Free
Database	Any ADOdb engine	MySQL, Postgres	MySQL recommended	MySQL, Postgres	MySQL, PostGreSQL, Oracle, MSSQL	MySQL	MySQL, Oracle, MSSQL, Postgres	AS400	Zope	MySQL version 4.0 or greater
License	GNU LGPL	GNU GPL + optional Commercial license	GNU GPL	GNU GPL	GNU GPL	GNU/GPL v2	GNU GPL	GNU LGPL	GNU GPL	GNU GPL
Operating System	Any	Any	Linux, Windows	Any	Linux, Windows	Any	Any	Any	Any	OS Independent
Programming Language	PHP, Java Script, DHTML	PHP	PHP 4.3 or later	PHP	PHP	PHP	PHP 4.3 or later	Java 1.4. +	Python	PHP version 4.2 or greater
Root Access	No	No	No	No	No	No	No	No	No	No
Shell Access	No	No	No	No	No	No	No	No	Yes	No
Web Server	Any	Any php enabled server	Apache	Apache, IIS	Apache	Apache	Any	IIS, Apache	Apache, Nginx, IIS, (+anything that can proxy)	Apache, mod_rewrite
Security	bitweaver	CMS Made Simple	Dragonfly CMS	Drupal	eZ Publish	Joomla!	Moodle	OpenCms	Plone	WordPress
Audit Trail	Yes	Yes	No	Yes	Yes	No	Yes	Yes	Yes	Limited
Captcha	Yes	Free Add On	No	Free Add On	Free Add On	Free Add On	No	Yes	Free Add On	No
Content Approval	Yes	No	Limited	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Email Verification	Yes	No	Yes	Yes	Yes	Yes	Yes	No	Yes	Free Add On
Granular Privileges	Yes	No	Limited	Yes	Yes	No	Yes	Yes	Yes	Yes
Kerberos Authentication	No	No	No	No	No	No	Yes	No	Free Add On	No
LDAP Authentication	Yes	No	No	Free Add On	Yes	Yes	Yes	Costs Extra	Yes	No
Login History	Limited	No	No	Yes	Yes	Yes	Yes	Yes	Free Add On	Free Add On
NIS Authentication	No	No	No	No	No	No	No	No	Free Add On	No
NTLM Authentication	No	No	No	Free Add On	Free Add On	No	No	No	Free Add On	No
Pluggable Authentication	Yes	No	No	Yes	Yes	Yes	Yes	Costs Extra	Yes	Yes
Problem Notification	Yes	No	Yes	No	Costs Extra	No	Yes	Yes	Free Add On	Free Add On
Sandbox	Yes	Limited	No	No	Yes	No	Yes	Yes	Yes	Limited
Session Management	Yes	No	Yes	Yes	Yes	Yes	Yes	No	Free Add On	Free Add On
SMB Authentication	No	No	No	No	Free Add On	No	No	No	Free Add On	No
SSL Compatible	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SSL Logins	Yes	No	No	No	Yes	Yes	Yes	Yes	Free Add On	Free Add On
SSL Pages	Yes	No	No	No	Yes	Yes	Yes	Yes	No	Limited
Versioning	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Free Add On

Figura 7.1: Comparativa CMS. Requisitos del sistema y seguridad



Support	bitweaver	CMS Made Simple	Dragonfly CMS	Drupal	eZ Publish	Joomla!	Moodle	OpenCms	Plone	WordPress
Certification Program	No	No	Yes	No	Yes	No	Yes	No	No	No
Code Skeletons	Yes	Free Add On	No	Yes	Yes	No	Yes	No	Yes	No
Commercial Manuals	No	No	Costs Extra	Yes	Yes	Yes	Yes	Yes	Yes	No
Commercial Support	No	No	Costs Extra	Yes	Yes	Yes	Yes	Yes	Yes	No
Commercial Training	No	No	Costs Extra	Yes	Yes	Yes	Yes	Yes	Yes	No
Developer Community	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Online Help	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Pluggable API	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Professional Hosting	No	Yes	Costs Extra	Yes	Yes	Yes	Yes	Limited	Yes	Yes
Professional Services	No	Yes	Costs Extra	Yes	Yes	Yes	Yes	Yes	Yes	No
Public Forum	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Public Mailing List	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Test Framework	No	No	No	Free Add On	Yes	Yes	Yes	Yes	Yes	Yes
Third-Party Developers	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Users Conference	No	Yes	No	Yes	Yes	Yes	Yes	Limited	Yes	Yes
Ease of Use	bitweaver	CMS Made Simple	Dragonfly CMS	Drupal	eZ Publish	Joomla!	Moodle	OpenCms	Plone	WordPress
Drag-N-Drop Content	No	No	No	Free Add On	No	No	Yes	Limited	Yes	Yes
Email To Discussion	Yes	No	No	Free Add On	No	Free Add On	Yes	No	Free Add On	Limited
Friendly URLs	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Image Resizing	Yes	Yes	Yes	Free Add On	Yes	Yes	Yes	Yes	Yes	Limited
Macro Language	No	No	No	Free Add On	Yes	Yes	No	No	Yes	Free Add On
Mass Upload	Yes	Limited	Yes	Free Add On	Yes	No	Yes	Yes	Yes	Free Add On
Prototyping	Limited	No	No	Limited	Yes	Yes	No	No	Yes	No
Server Page Language	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Site Setup Wizard	No	No		Limited	Yes	No	No	No	No	
Spell Checker	Yes	No	No	Free Add On	No	No	Free Add On	Free Add On	Free Add On	Free Add On
Style Wizard	No	No	No	Limited	No	No	Yes	No	Free Add On	No
Subscriptions	Yes	No	Yes	Free Add On	Yes	No	Yes	Costs Extra	Yes	Yes
Template Language	Yes	Yes	Yes	Limited	Yes	Yes	No	Yes	Yes	No
UI Levels	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
Undo	Limited	No	Yes	Limited	Yes	No	Yes	Yes	Yes	Free Add On
WYSIWYG Editor	Yes	Yes	Yes	Free Add On	Yes	Yes	Yes	Yes	Yes	Yes
Zip Archives	Limited	No	No	No	Free Add On	No	Yes	Limited	Free Add On	Free Add On
Performance	bitweaver	CMS Made Simple	Dragonfly CMS	Drupal	eZ Publish	Joomla!	Moodle	OpenCms	Plone	WordPress
Advanced Caching	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Free Add On
Database Replication	No	No	No	Limited	Yes	No	Yes	Costs Extra	Yes	No
Load Balancing	Yes	No	Yes	Yes	Yes	Yes	Yes	Costs Extra	Yes	No
Page Caching	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Free Add On
Static Content Export	No	No	No	No	Yes	No	No	Yes	Free Add On	Limited

Figura 7.2: Comparativa CMS. Soporte, facilidad de uso y rendimiento

Management	bitweaver	CMS Made Simple	Dragonfly CMS	Drupal	eZ Publish	Joomla!	Moodle	OpenCms	Plone	WordPress
Advertising Management	No	Free Add On	Yes	Free Add On	Free Add On	Yes	No	No	Free Add On	No
Asset Management	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Limited
Clipboard	No	No	No	No	No	No	Yes	No	Yes	No
Content Scheduling	Yes	No	Yes	Free Add On	Yes	Yes	No	Yes	Yes	Free Add On
Content Staging	No	Free Add On	No	Free Add On	Yes	No	Yes	Limited	Free Add On	No
Inline Administration	Yes	No	No	Yes	Yes	Yes	Yes	Limited	Yes	No
Online Administration	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Package Deployment	Yes	No	No	No	Yes	No	Yes	Limited	Yes	No
Sub-sites / Roots	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	No
Themes / Skins	Yes	Yes	No	Yes	Yes	Yes	Yes	No	Yes	Yes
Trash	No	No	No	No	Yes	Yes	No	Yes	Free Add On	No
Web Statistics	Yes	Free Add On	Yes	Yes	Free Add On	Yes	Yes	No	Free Add On	Free Add On
Web-based Style/Template Management	No	Yes	No	Yes	Yes	Yes	Yes	Limited	Yes	Yes
Web-based Translation Management	Yes	Yes	No	Yes	Yes	Free Add On	Yes	No	Yes	Limited
Workflow Engine	Free Add On	Limited	No	Limited	Yes	No	Yes	Free Add On	Yes	No
Interoperability	bitweaver	CMS Made Simple	Dragonfly CMS	Drupal	eZ Publish	Joomla!	Moodle	OpenCms	Plone	WordPress
Content Syndication (RSS)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Costs Extra	Yes	Yes
FTP Support	No	No	No	Limited	No	Yes	Yes	No	Yes	Free Add On
iCal	No	No		Free Add On	No	No	Yes	No	Free Add On	
UTF-8 Support	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
WAI Compliant	Limited	No	No	Limited	Yes	No	Yes	Limited	Yes	Limited
WebDAV Support	No	No	No	No	Yes	No	Yes	Yes	Yes	No
XHTML Compliant	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Flexibility	bitweaver	CMS Made Simple	Dragonfly CMS	Drupal	eZ Publish	Joomla!	Moodle	OpenCms	Plone	WordPress
CGI-mode Support	No	No	Yes	Yes	Yes	Yes	Yes	No	Yes	No
Content Reuse	Yes	No	No	Limited	Yes	Yes	Yes	Yes	Yes	No
Extensible User Profiles	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Interface Localization	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Metadata	Free Add On	Limited	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Multi-lingual Content	Yes	No	Yes	Yes	Yes	Free Add On	Yes	Yes	Yes	Free Add On
Multi-lingual Content Integration	Limited	Free Add On	No	Free Add On	Yes	Free Add On	Yes	Yes	Yes	Free Add On
Multi-Site Deployment	No	No	No	Yes	Yes	Free Add On	Yes	Yes	Yes	No
URL Rewriting	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes

Figura 7.3: Comparativa CMS. Gestión, interoperabilidad y flexibilidad

Built-in Applications	bitweaver	CMS Made Simple	Dragonfly CMS	Drupal	eZ Publish	Joomla!	Moodle	OpenCms	Plone	WordPress
Blog	Yes	Free Add On	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
Chat	Yes	Free Add On	Free Add On	Free Add On	Free Add On	Free Add On	Yes	No	Free Add On	Free Add On
Classifieds	No	No	Free Add On	Free Add On	Free Add On	Free Add On	Yes	No	Free Add On	Free Add On
Contact Management	Yes	No	Yes	Free Add On	Yes	Yes	Yes	No	Free Add On	Free Add On
Data Entry	No	Free Add On	Yes	Free Add On	Yes	Free Add On	Yes	No	Free Add On	No
Database Reports	No	No	No	No	Limited	Free Add On	Yes	Costs Extra	Yes	No
Discussion / Forum	Yes	Free Add On	Yes	Yes	Yes	Free Add On	Yes	Free Add On	Free Add On	Free Add On
Document Management	No	No	Free Add On	Limited	Yes	Free Add On	Yes	No	Yes	No
Events Calendar	Yes	Free Add On	Free Add On	Free Add On	Yes	Free Add On	Yes	Costs Extra	Yes	Free Add On
Events Management	No	No	Free Add On	Free Add On	Free Add On	Free Add On	Yes	No	Yes	No
Expense Reports	No	No	No	No	No	Free Add On	Yes	No	Free Add On	No
FAQ Management	No	Free Add On	Free Add On	Yes	Free Add On	Yes	Yes	Free Add On	Free Add On	Free Add On
File Distribution	Yes	Free Add On	Free Add On	Free Add On	Yes	Free Add On	Yes	No	Yes	Free Add On
Graphs and Charts	Limited	No	No	No	Free Add On	Free Add On	Yes	No	Free Add On	No
Groupware	Limited	No	No	Free Add On	No	Free Add On	Yes	No	Free Add On	No
Guest Book	Yes	Free Add On	Free Add On	Free Add On	Free Add On	Free Add On	Yes	No	Free Add On	Free Add On
Help Desk / Bug Reporting	No	No	Free Add On	Free Add On	Yes	Free Add On	Yes	No	Free Add On	No
HTTP Proxy	No	No	No	No	Free Add On	No	No	No	Free Add On	No
In/Out Board	No	No	No	No	No	No	No	No	Free Add On	No
Job Postings	No	Free Add On	No	Free Add On	Yes	Free Add On	Yes	No	Free Add On	No
Link Management	No	Free Add On	Free Add On	Free Add On	Yes	Yes	Yes	Costs Extra	Free Add On	Free Add On
Mail Form	No	Free Add On	Yes	Free Add On	Yes	Yes	Yes	Yes	Yes	Yes
Matrix	No	No	No	No	No	No	Yes	Yes	Free Add On	Free Add On
My Page / Dashboard	Yes	No	Yes	Free Add On	Limited	No	No	No	No	No
Newsletter	Yes	Free Add On	Yes	Free Add On	Free Add On	Free Add On	Yes	No	Yes	Yes
Photo Gallery	Yes	Free Add On	Yes	Free Add On	Yes	Free Add On	Yes	Free Add On	Free Add On	Free Add On
Polls	No	Free Add On	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Free Add On
Product Management	No	No	Free Add On	Free Add On	Yes	Yes	Yes	Free Add On	Free Add On	Free Add On
Project Tracking	No	No	Free Add On	Free Add On	No	Free Add On	Yes	Costs Extra	Yes	No
Search Engine	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Free Add On	No
Site Map	No	Yes	No	Free Add On	Yes	Free Add On	Yes	Yes	Yes	Yes
Stock Quotes	No	No	Free Add On	Free Add On	No	No	No	Yes	Yes	Free Add On
Surveys	No	No	Yes	Free Add On	Free Add On	Free Add On	Free Add On	No	Free Add On	No
Syndicated Content (RSS)	Yes	Free Add On	Yes	Yes	Yes	Yes	Yes	No	Free Add On	Free Add On
Tests / Quizzes	No	Free Add On	Free Add On	Free Add On	Free Add On	Free Add On	Yes	Costs Extra	Yes	Yes
Time Tracking	No	No	No	Free Add On	No	No	Yes	No	Free Add On	Free Add On
User Contributions	Yes	No	Yes	Yes	Yes	Yes	Yes	No	Free Add On	Free Add On
Weather	No	No	Free Add On	Free Add On	Free Add On	No	Yes	No	Yes	Yes
Web Services Front End	No	No	No	Limited	No	Yes	No	No	Free Add On	Free Add On
Wiki	Yes	No	No	Free Add On	Yes	Free Add On	No	No	No	Free Add On
							Yes	No	Free Add On	Free Add On

Figura 7.4: Comparativa CMS. Aplicaciones incorporadas

Commerce	bitweaver	CMS Made Simple	Dragonfly CMS	Drupal	eZ Publish	Joomla!	Moodle	OpenCms	Plone	WordPress
Affiliate Tracking	No	No	Yes	Free Add On	No	Free Add On	No	No	No	No
Inventory Management	No	No	No	Free Add On	No	Free Add On	Yes	No	Free Add On	No
Pluggable Payments	Yes	No	No	Free Add On	Yes	Free Add On	Yes	No	Free Add On	No
Pluggable Shipping	Yes	No	No	Free Add On	Yes	Free Add On	No	No	Free Add On	No
Pluggable Tax	No	No	No	Free Add On	Yes	Free Add On	No	No	Free Add On	No
Point of Sale	No	No	No	No	No	Free Add On	No	No	Free Add On	No
Shopping Cart	Yes	No	Free Add On	Free Add On	Yes	Free Add On	No	Free Add On	Free Add On	No
Subscriptions	No	No	No	Free Add On	Free Add On	Free Add On	No	No	Free Add On	No
Wish Lists	No	No	No	Free Add On	Yes	Free Add On	Yes	No	No	No

Figura 7.5: Comparativa CMS. Comercio

7.8. Elección de un CMS. Drupal

Las razones que llevaron a la elección de Drupal fueron varias. Para empezar, es uno de los que cubrían mejor los requisitos citados en este capítulo. Pero tal vez el más importante de ellos es la facilidad con que se añaden funcionalidades nuevas y la gran cantidad de módulos preexistentes que se pueden hallar en su repositorio. Además, cuenta con mucha documentación, aunque a veces más dispersa de lo que uno desearía.

Si se observa la tabla de la comparativa se puede ver que un número elevado de características no son resueltas por Drupal directamente desde el *core*, sino que necesita de la ayuda de otros módulos, desarrollados por la comunidad OpenSource. Si lo comparamos con otros gestores como Moodle o eZ Publish, que implementan casi todas las funcionalidades de serie, podría parecer un CMS algo limitado. Pero en realidad el poder instalar sólo lo que a uno le convenga ayuda a no querer abarcar más de lo necesario, hecho que incurriría en potenciales problemas de seguridad.

Otros gestores de contenidos como CMS Made Simple tienen una cierta popularidad debido a su facilidad de uso, pero no ofrecen todo lo que se podría esperar.

Existen sitios web como http://www.opensourcecms.com que permiten probar una larga lista de CMS antes de instalarlos. Se hicieron algunas pruebas con los más conocidos, entre los que estaba Joomla!, y se acabó eligiendo Drupal por la facilidad de uso, la flexibilidad, el soporte y el rendimiento, así como por el uso de la plataforma LAMP (Linux, Apache, MySQL y PHP), muy extendida y fácil de encontrar en servicios de hosting.

Por si fuera poco, Drupal ganó en 2007 el Premio al Mejor CMS OpenSource, de PacktPublishing. Pero nunca es recomendable fiarse demasiado de los premios, pues Joomla! y WordPress ganaron el mismo año otras categorías de ese premio y ya habían ganado en 2006 y 2005 respectivamente el mismo premio que Drupal. En cualquier caso, eso sitúa a Drupal en una de las primeras posiciones y en la elección de muchos desarrolladores.

Otra comparativa interesante es la que hace IBM, en la que se dan razones por las que ellos mismos decidieron usar Drupal, http://www.ibm.com/developerworks/ibm/library/i-osource1/. Eligen una lista de CMS diferente a la nuestra, que incluye también el framework Ruby on Rails. Empezando por los requisitos previos del sistema (Figura 7.6) y siguiendo por un análisis de las funcionalidades (Figura 7.7), llegan a las conclusiones siguientes.

	Web server	Database	Language
Ruby on Rails	Apache, FastCGI	MySQL, PostgreSQL, SQLite, Oracle, SQL Server, DB2, Firebird	Ruby
Drupal	Apache IIS	MySQL, PostgreSQL	PHP
Mambo	Apache IIS	Apache IIS	PHP
Typo3	Apache IIS	Apache IIS	PHP
Movable Type	Apache, Jetty, Tomcat, IIS	Apache, Jetty, Tomcat, IIS	Perl
Word Press	Apache, mod_rewrite	Apache	PHP
Text Pattern	Apache	Apache	PHP

Figura 7.6: Comparativa IBM. Requisitos

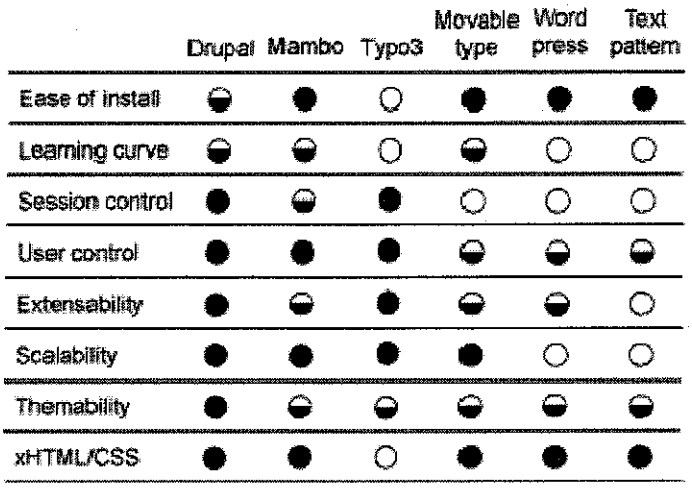


Figura 7.7: Comparativa IBM. Funcionalidades

Ruby on Rails (ROR) se descartó directamente porque obligaba a crear un CMS casi desde cero, ya que ROR no es un gestor de contenidos sino un framework.

Como lo que pretendían era una implementación sencilla, necesitaban un control de acceso de usuarios efectivo y una gestión de sesiones y usuarios robusta y flexible. Además, la velocidad de la implementación mejoraba si había una infraestructura modular cargada de funcionalidades y respaldada por una comunidad de desarrolladores.

Otro aspecto clave eran la escalabilidad y la extensibilidad, lo que lo hacen adecuado para sitios que se prevé que van a crecer en el futuro o que pronto van a incrementar el número de usuarios concurrentes.

La facilidad en el ajuste de la presentación de contenidos era algo crucial, ya que requerían una gran flexibilidad durante el diseño y los futuros ajustes. Además, era importante el uso de buenas prácticas con respecto al diseño web: XHTML, CSS y diseño accesible.

Y Drupal era el único que ofrecía todo esto junto. Como en nuestro sitio web estas funcionalidades también están entre las deseadas, Drupal parece la mejor elección.

7.9. Drupal

7.9.1. Introducción

Drupal es un Sistema Gestor de Contenidos (CMS) de código abierto, escrito en PHP y creado por Dries Buytaert que permite al administrador crear contenidos de distintas clases, organizarlos, cambiar la forma en que se muestran, automatizar tareas de administración y gestionar usuarios y visitantes.

Sin conocimientos de programación se puede montar rápidamente un sitio web con Drupal, pero la gran ventaja que tiene es su extensibilidad y su facilidad para ampliar funcionalidades creando código nuevo de forma totalmente modular.

Funciona en varios sistemas operativos y se ejecuta sobre la plataforma LAMP.

La instalación básica de Drupal, conocida como el *core* tiene varias características similares a las de otros CMS: registro y mantenimiento de cuentas de usuario, menús de administración, *feeds RSS*, estructura visual modificable, gestión de privilegios flexible, autentificación de usuarios, creación de blogs, foros, encuestas y la posibilidad de crear comunidades web interactivas.

Los contenidos pueden ser creados por usuarios registrados o anónimos y pueden ser encontrados por fecha, categoría, usuario, búsquedas... También se incluye en el core un

sistema de taxonomías que permite organizar el contenido de forma sencilla a base de etiquetas.

Además, mantiene un registro detallado de las operaciones realizadas por los usuarios.

El core proporciona varios módulos que pueden ser activados o desactivados por el administrador:

- Creación y edición de contenido por varios usuarios.
- Funciones avanzadas de búsqueda.
- Comentarios, foros y encuestas.
- Perfiles de usuario.
- Sistema de menús multinivel.
- Feeds RSS y agregador de feeds.
- Restricciones diversas de acceso (IP, roles de usuario, email...).
- Estadísticas de acceso y registro de operaciones.
- Caché de contenidos y la posibilidad de desactivar módulos automáticamente en caso de sobrecarga.
- URLs descriptivas (www.example.com/user/root en lugar de www.example.com?user=1).
- Herramientas como triggers y acciones.
- Notificaciones de actualizaciones de seguridad.
- Soporte para OpenID.

Además, el core de Drupal está traducido actualmente a 44 idiomas además del inglés, incluso algunos lenguajes de derecha a izquierda.

Una de las características principales de Drupal es que los contenidos se generan siempre de forma dinámica. Prácticamente todo lo necesario para mostrar una página se encuentra almacenado en la base de datos. No existen páginas estáticas guardadas en archivos con la extensión .html, sino que en el mismo momento de recibir la petición por parte del usuario, Drupal crea la página tomando los contenidos necesarios de la BD, siguiendo una serie de reglas para saber qué coger y de dónde.

Como inconveniente, Drupal, aunque se ejecuta perfectamente sobre PHP5, no está orientado a objetos, con lo cual no aprovecha al máximo la potencia del lenguaje. Es probable que eso cambie en el futuro.

7.9.2. Módulos

Como se ha comentado, la forma de extender la funcionalidad de Drupal consiste en crear módulos que se adapten al core o bien instalar otros módulos hechos por la comunidad de desarrolladores. Instalar un módulo consiste en descomprimir el archivo que lo contiene en un lugar concreto del árbol de directorios. De hecho, hay varios lugares donde Drupal detectará el módulo: en el directorio donde están los módulos del core, en un directorio específico para el sitio web que se está creando o bien en un directorio que sirva para una instalación multi-sitio.

Una vez allí, Drupal buscará en el directorio del módulo un archivo llamado `nombre_modulo.info`, en el que se especifica el nombre, la versión, etc., y lo añadirá automáticamente a la lista de módulos disponibles, permitiendo instalarlo desde la zona de administración para empezar a usar sus funciones.

Instalar un módulo consiste en ejecutar las instrucciones que se encuentran en un archivo definido por el programador con el nombre `nombre_modulo.install`. Básicamente son sentencias SQL que crean las tablas necesarias en la base de datos.

Una vez instalado, un módulo se puede volver a desinstalar o bien desactivar.

- Desactivándolo se eliminará su presencia en las páginas web del sitio y su funcionalidad.
- Desinstalándolo se borrarán sus tablas de la BD y con ellas todos los datos con los que trabajaba (las sentencias de eliminación de tablas también hay que especificarlas en el archivo `.install`).

La implementación del módulo comienza en el archivo `nombre_modulo.module`. Al tratarse de PHP, se pueden incluir todos los archivos que se quiera para complementar el código del módulo.

7.9.3. Nodos

En Drupal cualquier tipo de contenido que pueda tener un título y un cuerpo, un autor, una fecha de creación y edición, etc. es llamado *nodo*. Un nodo puede ser, por ejemplo, un post de un blog, una imagen, una encuesta o un mensaje en un foro.

Dichos nodos pueden tener el estado de publicados o no publicados, pueden ser promocionados a la página principal, ordenados, archivados, buscados, modificados, comentados por otros usuarios, votados o eliminados. Tratarlos a todos como un mismo tipo de entidades da una gran consistencia a Drupal.

No sólo eso, sino que un desarrollador puede crear nuevos tipos de contenido (un vídeo, una postal, un producto para vender...) mediante el código, cuyas instancias serán nodos. Esos nodos tendrán campos diferentes unos de otros (un producto tendrá un precio mientras que una postal tendrá un destinatario), que deberán ser definidos por el programador a través del código. Drupal ofrece una API para formularios que facilita la labor de introducción y recuperación de datos.

Además, existe un módulo, CCK, ideado para que se puedan añadir campos a esos tipos de contenido sin tener que escribir código, como un campo de texto, un select o un botón. Otros módulos utilizan el módulo CCK para crear nuevos tipos de campos. El módulo Date, por ejemplo, permite añadir con CCK un campo de fecha. Además, desde Drupal 5 la creación de tipos de contenido nuevos se puede hacer desde la interfaz.

Los nodos tienen distintas formas de mostrarse: completo, resumen, lista... Y por supuesto, todas ellas se pueden modificar. Así, por ejemplo, en la página principal o en el resultado de una búsqueda se podría mostrar un resumen de cada nodo (*teaser* en lenguaje Drupal) y en un bloque de un lateral se mostraría una lista de los nodos más votados, que consistiría en el título del nodo. Al hacer clic en el título, tanto en la lista como en el resumen, se accedería al nodo completo, en el que se vería toda la información, así como los comentarios de los usuarios, etc.

7.9.4. Bloques

En distintas partes de una web Drupal ofrece la posibilidad de crear lo que llama *bloques*. Un bloque viene a ser una caja en la que se puede insertar contenido, incluidos nodos. Un bloque *no* es un nodo. Los bloques se pueden activar o desactivar, ordenar, mostrar sólo a un cierto tipo de usuarios (p.e., los autentificados), permitir a los propios usuarios que decidan si quieren verlos o no, etc.

El contenido de esos bloques se puede definir desde el código o bien desde la interfaz, pudiendo aplicar filtros en caso de estar definiendo una lista de nodos. Los lugares habituales para colocar nodos son las barras laterales o bien las zonas superior o inferior del contenido, pero eso queda a elección del administrador.

7.9.5. Usuarios

Los usuarios tampoco son tipos de nodos para Drupal, sino entidades aparte. Un usuario cuenta en general con una dirección de correo, un nombre de usuario y un password. A partir de ahí, el administrador decide si quiere que tenga más campos. El usuario principal, el administrador general de Drupal, tiene el id de usuario 1.

Un usuario puede pertenecer a uno o más *roles*. Un rol define un conjunto de permisos de acceso, modificables desde la interfaz. Por ejemplo, es normal que un usuario con rol "autentificado" tenga más permisos que uno que sea "invitado", y otro de rol "administrador" pueda realizar tareas que ninguno de los otros tenga permiso para ejecutar.

Incluso se puede definir desde el código que un usuario determinado vea ciertos tipos de contenido o tenga ciertos privilegios. En cualquier momento está disponible la variable global `$user`. Eso permite también modificar el perfil o cambiarle los privilegios.

El registro de usuarios, así como el control de sesiones, es una de las partes que más trabajo ahorran al desarrollador. Desde hacer login a recibir automáticamente emails de confirmación de registro o de cambio de password, pasando por tener que pedir permiso al administrador para poder registrarse, Drupal se encarga de gestionar todo eso y más, incluyendo autentificación externa con LDAP, etc.

7.9.6. Hooks

Si hay algo que define la forma de trabajar con Drupal, ya sea para implementar nuevos módulos o para modificar los existentes son los *hooks*. Un hook viene a ser algo parecido a lo que en muchos lenguajes de programación se llama *callback*. Es decir, es una función que se dispara de forma asíncrona en cuanto se cumple algún tipo de condición (normalmente conocida como *evento*).

Dichos hooks son funciones con unos nombres determinados y deben ser implementados por el programador en caso de que quiera modificar el comportamiento de Drupal en algún sentido.

Durante la ejecución de Drupal existen determinados momentos en los que el sistema pregunta a los módulos si quieren hacer algo. Por ejemplo, supongamos que estamos creando un módulo `producto` que añade a Drupal un nuevo tipo de contenido también llamado `producto`. El formulario de inserción y edición de un producto será idéntico al de cualquier tipo de nodo básico: un título y un cuerpo. Si queremos añadir un campo `precio`, lo primero que debemos hacer es implementar el hook `form()`, que en este caso será una función llamada `producto_form()`, según el convenio de Drupal para los hooks. En ella, y siguiendo las directrices de la API de formularios de Drupal, podremos definir no sólo el tipo de campo (será un campo de texto) sino incluso decirle que tiene que ser numérico y que muestre el campo de otro color en caso de que alguien escriba un valor no numérico y haga clic en enviar. Para esto último habrá que implementar el hook `producto_validate()` y hacer la comprobación correspondiente.

Del mismo modo, en el momento en que el usuario entre correctamente los datos y acepte el formulario, Drupal preguntará al módulo `producto` si quiere hacer algo más. Ése es el momento en el que se llamará al hook `producto_insert()`, implementado por el desarrollador. Como Drupal sólo se preocupará de guardar la parte básica del nodo, nosotros deberemos decirle dónde guardar los campos que hemos añadido, es decir, el precio. Aquí haremos una inserción en una tabla de la base de datos que habremos creado para tal uso.

A continuación, Drupal pretenderá mostrarnos el nodo recién insertado, lo cual significa que nuestro navegador hará una petición para ver el nodo. Como se ha dicho, Drupal lo almacena todo en la BD, por lo tanto irá a recuperar los datos almacenados. Pero sólo sabrá obtener el nombre y el cuerpo del producto, que son las únicas partes que sabía

guardar. Por eso hará una llamada al hook `producto_load()` para dar al módulo la oportunidad de cargar los datos que falten. El hook hará una consulta a la BD para obtener el precio del producto y se lo devolverá a Drupal.

Una vez que tenga todos los datos, Drupal llamará a las funciones de visualización. Nuevamente, sabrá cómo mostrar el título y el cuerpo del nodo pero no el precio, por lo que llamará a la función `producto_view()`, que se encargará de eso.

En caso de editar el nodo, en lugar del hook `insert()` se llamará al hook `update()`, y si lo que se quiere es eliminarlo, se hará uso del hook `delete()`.

Esto es, en resumen lo que hay que hacer para crear y mantener un nuevo tipo de nodo desde el código. Igual que existen estos hooks, hay otros para modificar formularios existentes (hook `form_alter()`), o para cambiar el comportamiento de nodos definidos por otro módulo (hook `nodeapi()`). En el ejemplo explicado, el tipo de nodo lo definía el mismo módulo.

De esta forma, el usuario puede acceder al flujo de ejecución en casi cualquier momento y definir comportamientos añadidos, lo cual le da un grado de libertad muy alto y una potencia y una simplicidad que no tienen otros CMS.

7.9.7. Taxonomías

La categorización de contenidos es algo que han puesto de moda los blogs y que ha demostrado que funciona muy bien. Es habitual encontrar en las entradas de todo tipo de blogs unas etiquetas que indican a qué categorías pertenecen. Por ejemplo: cultura, deportes, informática, varios...

Se trata de *taxonomías*. Existe una taxonomía, que se podría llamar "categorías" y que incluiría varios *términos*, que son las palabras que dan nombre a las etiquetas mencionadas. Cuando el administrador del blog escribe una nueva entrada, puede elegir los términos de la taxonomía "categorías" a los que quiere asociar esa entrada. Si no existe el término que necesita, lo puede crear en ese momento.

Una vez publicada la entrada, cada una de las etiquetas seleccionadas aparecerá en algún lugar en forma de enlace. Al hacer clic en ese enlace se mostrará una página que contendrá todas las entradas asociadas a ese término de la taxonomía.

Drupal cuenta con un elaborado sistema de taxonomías. En muchos casos, con tener una lista de términos es más que suficiente, pero hay ocasiones en las que es necesario definir varias taxonomías. Por ejemplo, en una web sobre películas puede hacer falta una taxonomía para el género (drama, acción, comedia, aventuras...) y otra para la edad recomendada (todos los públicos, mayores de 7, de 13, de 18...). Naturalmente, se podrían tener todos esos términos juntos en una única categoría, pero tiene mucho más sentido y conserva más la lógica de la web separarlos en dos.

Drupal permite incluso crear jerarquías y dependencias entre taxonomías, términos sinónimos o relacionados, términos fijos o variables (que se puedan crear en el mismo momento de asignarlos), etc. Y tiene la opción de editar los vocabularios y los propios términos, añadiéndoles una descripción.

De este modo, los contenidos de la web pueden quedar muy correctamente organizados si se utilizan las taxonomías con un cierto cuidado.

7.9.8. Theming

Se entiende por *theming* la forma de presentar los contenidos del sitio web. Drupal, como otros gestores de contenidos, separa totalmente la capa de dominio de la capa de presentación. Esto permite al administrador cambiar en cualquier momento la apariencia de la web sin tener que modificar nada del contenido. Para conseguirlo, se utilizan los llamados *template languages* como Smarty, PHPTAL o XTemplate, bien conocidos por los

desarrolladores web. Estos lenguajes básicamente proporcionan al programador la posibilidad de incrustar contenidos dinámicos dentro del código HTML.

Drupal ofrece, además, una capa intermedia de abstracción para la comunicación con estos lenguajes: los llamados *theme engines*. El más utilizado, que fue creado a medida por los desarrolladores y que viene por defecto con la instalación básica de Drupal, es PHPTemplate. Como su nombre indica, está totalmente basado en el mismo lenguaje que Drupal, por lo que con él no sólo se evita el paso intermedio de parsing necesario para los otros lenguajes sino que se le da al programador avanzado la flexibilidad de utilizar cualquier dato accesible a través de la API de Drupal, no sólo los que el lenguaje o el motor de theming le permitan.

Para poner un ejemplo sencillo, el código siguiente mostraría los enlaces primarios y secundarios de Drupal usando PHPTemplate:

```
<div id="top-nav">
  <?php if (count($secondary_links)) : ?>
    <ul id="secondary">
    <?php foreach ($secondary_links as $link): ?>
      <li><?php print $link?></li>
    <?php endforeach; ?>
    </ul>
  <?php endif; ?>
  <?php if (count($primary_links)) : ?>
    <ul id="primary">
    <?php foreach ($primary_links as $link): ?>
      <li><?php print $link?></li>
    <?php endforeach; ?>
    </ul>
  <?php endif; ?>
</div>
```

Se puede observar cómo se mezcla el código HTML con el de PHP formando una plantilla, pudiendo utilizar bucles, condiciones, etc., simplemente insertando el código PHP entre los símbolos `<?php` y `?>`, como es habitual. Lo interesante de este código es el uso de unas ciertas variables que no han sido definidas previamente en el mismo archivo.

Los *themes* en Drupal son un conjunto de archivos que hacen que los contenidos se muestren de una u otra manera. Se pueden descargar e instalar themes prediseñados, de los cuales unos cuantos vienen con la instalación de Drupal, o bien modificar los existentes o crear otros propios.

Al final todo consiste en decirle a Drupal, a través de esos archivos con extensión .tpl.php, cómo debe mostrar cada parte de la web. Para ello hay que saber elegir los archivos y sus nombres, ya que Drupal buscará el archivo más específico que pueda aplicar basándose en varios criterios, como por ejemplo la URL. Si no lo encuentra, buscará otro más general, y así hasta dar con uno que le indique la forma de presentar los contenidos. Por ejemplo, para saber cómo debe mostrar la URL http://example.com/?q=node/1/edit, Drupal buscará los archivos en este orden:

```
page-node-edit.tpl.php
page-node-1.tpl.php
page-node.tpl.php
page.tpl.php
```

Cada uno de los archivos de template que vienen con Drupal, como `page.tpl.php`, `node.tpl.php`, `block.tpl.php`, `comment.tpl.php` o `box.tpl.php` tiene una lista de variables predeterminadas a las que se puede acceder (por ejemplo, `$date` en el archivo `node.tpl.php`). Pero por supuesto, el usuario puede definir las suyas propias.

El usuario puede también redefinir funciones de theming, que son las que están en el código fuente y generan esas variables, de modo que la flexibilidad a la hora de decidir qué mostrar es casi absoluta.

Sobre cómo mostrarlo, en última instancia son los archivos de hojas de estilo en cascada (CSS) los que se encargan de eso. Por lo tanto, Drupal ofrece grandes facilidades para crear, modificar y manipular el theming de un sitio web.

7.9.9. Aprendizaje de Drupal

Sobre la dificultad de aprendizaje de Drupal, el propio creador del CMS, Dries Buytaert, publicó una gráfica (Figura 7.8) en la que se podía apreciar la complejidad en los distintos niveles.

La iniciación de un usuario que no pretenda programar módulos o temas de presentación, sino hacerlo todo desde la interfaz, es relativamente sencilla, y consistiría en la superación del umbral "soy un asco". Es decir, comprender algunos de los conceptos más básicos explicados en esta sección y saber llevarlos a cabo sin ayuda de programación.

Si ese usuario quiere llegar más lejos, retocando algunas características de los temas de visualización, personalizando algunas partes de la web, instalando módulos de idioma, etc. llegaría a un nivel alto de usuario (umbral "soy la leche"). Esta parte puede ser más costosa para un usuario no programador, porque ya incluye algunos conceptos avanzados.

El nivel superior consiste en utilizar nuevos tipos de contenido (ya sea creados con CCK o programáticamente), entender el concepto de hook, desarrollar módulos y temas, etc.

Tal vez habría que hacer dos gráficas distintas, una para programadores y otra para no programadores, ya que para una persona que no conozca PHP, HTML, Javascript o CSS el nivel superior sería directamente inalcanzable, mientras que para alguien acostumbrado a todos estos lenguajes no sería un problema superar el segundo nivel con una cierta tranquilidad, mientras que el nivel avanzado sería el verdadero punto de inflexión de la complejidad, que se reduciría a medida que ascienden las habilidades.

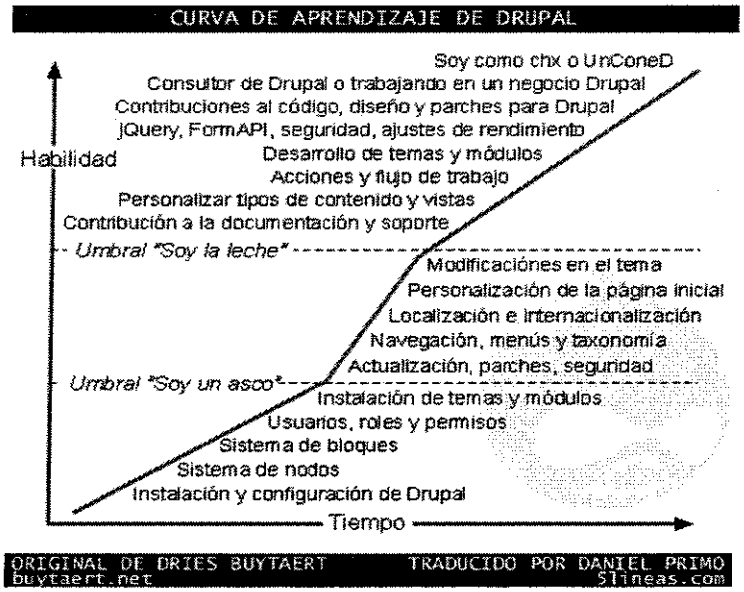


Figura 7.8: Curva de aprendizaje de Drupal

7.9.10. Comunidad y módulos

La comunidad de Drupal es una de las más activas tras un CMS. En la página principal de Drupal, www.drupal.org, hay registrados unos 300.000 usuarios, y alrededor de 2000 personas han hecho una petición para cuentas de desarrollo.

Hay listas de correo, foros y grupos de discusión, y se pueden encontrar miles de páginas en las que se explican detalles del funcionamiento de Drupal.

En cuanto a los módulos creados por desarrolladores, existen actualmente del orden de 2000, organizados en varias categorías:

- 3rd party integration (303)
- Administration (239)
- CCK (165)
- Commerce / advertising (82)
- Community (170)
- Content (401)
- Content display (430)
- Developer (160)
- e-Commerce (66)
- Evaluation/rating (66)
- Event (42)
- File management (56)
- Filters/editors (143)
- Import/export (66)
- Javascript Utilities (101)
- Location (39)
- Mail (104)
- Media (149)
- Multilingual (25)
- Organic Groups (53)
- Paging (19)
- RDF (15)
- Search (64)
- Security (57)
- Syndication (60)
- Taxonomy (123)
- Theme related (116)
- User access/authentication (139)
- User management (127)
- Utility (424)

- Views (95)

Prácticamente cualquier elemento que pueda encontrarse en una web es susceptible de estar en alguno de estos módulos en forma de varias funcionalidades: soporte para diversos idiomas, creación de newsletters, Ajax, comercio electrónico, editores WYSIWYG, Captchas, autentificación segura, envío de emails, valoración de los usuarios, RSS...

8.Implementación

8.1. Decisiones de implementación.

Una vez especificado y diseñado el sistema se pasa al desarrollo del código. Dado que se trabaja con un sistema gestor de contenidos, Drupal, la implementación depende en gran medida de su forma de funcionar y de cómo Drupal gestiona sus procesos. Por lo tanto, cualquier adición a su funcionalidad deberá hacerse en forma de módulo, siguiendo sus directrices para la implementación de módulos.

Para crear y mantener el Catálogo musical se ha creado un módulo llamado **`catalogue`**. Como se ha dicho, se ha trabajado en inglés contemplando la posibilidad de poder contribuir a la comunidad de Drupal ofreciendo parte del resultado de este PFC.

Además, dado que algunas partes del módulo **`catalogue`** necesitaban una funcionalidad muy concreta, se ha desarrollado otro pequeño módulo llamado `widget`, a partir de un módulo de ejemplo.

8.2. Estudio y elección de módulos

Son varios los módulos que se han utilizado, además de los nuestros, para dar al sitio web toda la funcionalidad que requería. Esta es la lista de los utilizados. De los dígitos de la versión, los primeros, 5.x, hacen referencia a la versión de Drupal, mientras que los siguientes son la versión del propio módulo (los del core sólo tienen versión de Drupal).

- **Core**
  - Color 5.7: Permite cambiar la paleta de colores en algunos themes.
  - Comment 5.7: Ofrece la posibilidad de que los usuarios añadan comentarios a los nodos.
  - Contact 5.7: Activa formularios de contacto para usuarios y para el sitio.
  - Help 5.7: Gestiona los mensajes de ayuda que se muestran al usuario.
  - Menu 5.7: Permite a los administradores modificar los menús de navegación del sitio web.
  - Path 5.7: Permite renombrar URLs. Es necesario para usar el módulo Pathauto.
  - Profile 5.7: Para cambiar el perfil de los usuarios.
  - Search 5.7: Activa las búsquedas del sitio.
  - Taxonomy 5.7: Activa la categorización de contenidos.

    - Upload 5.7: Permite a los usuarios adjuntar archivos al contenido.

- CCK
    - Content 5.x-1.6-1: Permite a los administradores definir nuevos tipos de contenido.
    - Number 5.x-1.6-1: Permite definir campos numéricos.
- AHAH
    - AHAH Forms 5.x-1.5-5: Paquete de utilidades para crear formularios dinámicos. Lo utilizamos para el módulo Widget.
- Image
    - Image 5.x-2.0: Permite subir, visualizar y cambiar el tamaño de las imágenes.
    - Image Attach 5.x-2.0: Permite adjuntar fácilmente nodos del tipo image a otros tipos de nodo.
    - Image Gallery 5.x-2.0: Ordena y muestra galerías de imágenes basadas en categorías.
    - Image Import 5.x-2.0: Permite importar grupos de imágenes desde un directorio del servidor.
- Mail
    - Mime Mail 5.x-1.0: Para emails con HTML y adjuntos.
    - Simplenews 5.x-1.2: Envía newsletters a emails suscritos.
- Views
    - Views 5.x-1.6: Permite crear vistas específicas de listas de nodos.
    - Views Theme Wizard 5.x-1.6: Creación de themes para vistas.
    - Views UI 5.x-1.6: Permite crear vistas desde la interfaz de usuario.
- Otros
    - Pathauto 5.x-2.2: Crea alias para las URLs. Sirve para hacerlas más fáciles de leer a los buscadores.
    - Simplenews Template 5.x-1.1: Extiende la funcionalidad de Simplenews.
    - TinyMCE 5.x-1.9: Añade a los campos de texto propiedades WYSIWYG.
    - Token 5.x-1.11: API para añadir variables que se sustituyen por su valor. Necesario para usar Pathauto.

8.3. La implementación con Drupal

En la sección 7.9 se han dado ya algunos detalles sobre los métodos de desarrollo con Drupal, así como conceptos básicos (nodos, hooks...), y en la 10.1 se explica cómo poner a punto una instalación básica de Drupal, cómo y dónde hay que instalar los módulos, etc.

Principalmente, al ir a desarrollar un módulo con Drupal hay que crear varios ficheros necesarios:

- `mimodulo.info`: Contiene información básica del módulo, como el nombre, la versión, y los nombres de los módulos de los que depende.

- `mimodulo.install`: En él hay dos funciones: una, `mimodulo_install()`, que hace las peticiones SQL que crean las tablas que necesitará utilizar el módulo para funcionar (además de las que ya proporciona Drupal), y otra, `mimodulo_uninstall()`, que sirve para eliminar esas tablas en el momento en que el usuario decida no seguir usando el módulo.
- `mimodulo.module`: Éste es el fichero principal, donde empieza el código del módulo y donde irá a buscar Drupal las funciones y las implementaciones de los hooks. Este fichero puede contener *includes* de otros ficheros para organizar mejor la estructura del módulo.

En el momento de pensar en la creación de un módulo, uno debe preguntarse si necesitará crear tipos nuevos de contenido (por ejemplo, productos para vender, noticias para un periódico online, etc.) o si va a elaborar un módulo funcional que añada características a otros tipos de contenido (por ejemplo, dar la posibilidad de votar nodos). Dependiendo de la elección se deberán implementar unos hooks u otros.

8.3.1. Crear un tipo de nodo

Si lo que se pretende es crear un tipo de nodo nuevo, en el fichero `mimodulo.module` deben implementarse las funciones que complementen el funcionamiento de un nodo. Si, por ejemplo, se va a crear un tipo de nodo para recetas de cocina, en el fichero `mimodulo.install` se habrá creado una tabla `receta`. Esta tabla contendrá una clave foránea que apuntará al identificador de la tabla `node` (desde ahora, `nid`) y otros campos necesarios para almacenar una receta.

Para que, desde la interfaz, se pueda trabajar con recetas, hay que implementar varios hooks. Recordemos que los hooks son funciones que tienen que llamarse obligatoriamente `receta_nombrehook`. Por ejemplo, `receta_insert()`:

- ***form:*** Permite modificar el formulario de creación y edición del nodo, pudiendo añadir y eliminar campos. Para una receta se podría añadir un campo para ingredientes, otro para el número de personas para los que se han calculado...
- ***insert:*** Sirve para guardar en la base de datos la información extra añadida al formulario. Es llamado después de haber insertado la información básica del nodo y permitiría ejecutar sentencias SQL del tipo `insert` para guardar en la tabla `receta` los campos de ingredientes, número de personas, etc.
- ***update:*** Igual que el insert, pero para la modificación de un nodo existente (que comparte formulario con la inserción).
- ***delete:*** Se llama cuando se elimina un nodo, y sirve para que el programador pueda eliminar de sus tablas el resto de información que haya añadido al crear el nodo.
- ***load:*** Justo antes de visualizar un nodo, Drupal carga su información de la base de datos. Implementar este hook permite cargar los datos que no son básicos en el nodo y que están en las tablas específicas del módulo que se está creando. Debe incluir consultas SQL a la tabla `receta`.
- ***view:*** Indica a Drupal cómo debe mostrar por pantalla los nodos del tipo que se está creando, ya que no sabría cómo y dónde mostrar los ingredientes, por ejemplo.

Sobre las consultas a la base de datos, Drupal tiene una capa de abstracción intermedia que sirve, entre otras cosas, para *limpiar* el texto de las consultas, evitando así problemas de inyección SQL.

Un ejemplo de consulta para el hook load sería como sigue (para los inserts y updates el funcionamiento es el mismo):

```
$receta = db_fetch_object(db_query('SELECT * FROM {receta} WHERE nid = %d',
```

```
        $node->nid));
```

Se puede observar cómo se llama a unas funciones específicas de Drupal que empiezan por 'db', cómo los nombres de la tabla están entre claves y cómo se pasan parámetros a la consulta con un comportamiento parecido al de la función `printf()` de C.

8.3.2. Modificar nodos existentes

Si lo que se pretende es crear un módulo funcional, entonces el hook que se debe implementar, y que incluye todos los acabados de explicar, es el hook nodeapi. Este hook se ejecuta cada vez que se hace alguna operación en un nodo, y da la oportunidad a otros módulos de irrumpir en la ejecución y alterar ciertos comportamientos de ese nodo. La función recibe como parámetros el nodo, la operación que se está procesando y un booleano que indica si el nodo es completo o un resumen (*teaser*).

A los atributos de la variable `node` se puede acceder con toda normalidad, pero lo diferente de la función en comparación con los otros hooks explicados anteriormente es que la operación viene en otra variable, por lo que la estructura de la implementación de este hook contiene necesariamente una sentencia `switch` para elegir qué hacer cada vez. Es decir, algo así:

```
function mimodulo_nodeapi(&$node, $op, $teaser=FALSE) {
    ...
    switch($op) {
        case insert:
            ...
            break;
        case update:
            ...
            break;
        case load:
            ...
            break;
        ...
    }
    ...
}
```

Como detalle, la variable `node` se pasa por referencia, ya que así se le pueden hacer las modificaciones pertinentes.

8.3.3. Otros hooks

Hay varios hooks más en Drupal que permiten realizar acciones distintas. Algunos de los más importantes serían:

- ***menu:*** Asocia urls con funciones, de modo que cuando alguien acceda a una dirección concreta se ejecutará una determinada función.
- ***access:*** Sirve para definir permisos de acceso.
- ***block:*** Permite declarar bloques.

- ***validate:*** Se llama antes de una inserción o una actualización de un nodo, y sirve para comprobar que los datos introducidos son correctos.
- ***form_alter:*** Con él se puede acceder a formularios concretos y hacer modificaciones. Puede servir para poner valores por defecto en ciertos casos, etc.

La lista completa de hooks se puede encontrar en la dirección http://api.drupal.org/api/group/hooks/

Otros módulos, como el Views, incluyen una API que añade otros hooks al sistema, que pueden ser implementados por los módulos creados.

8.3.4. Creando varios tipos de nodo

En el caso de que un módulo necesite crear varios tipos de nodo en vez de uno solo, la forma correcta de hacerlo es crear un fichero `mimodulo.module` con varios includes de ficheros llamados *tiponodo*`.inc`. La extensión no es obligatoria, se puede usar `php` en lugar de `inc`.

En el fichero general del módulo se implementarían hooks más generales, como el menú, así como funciones para la creación del formulario de administración del módulo. También debe incluir un nuevo hook llamado node_info, que es el que indica los distintos tipos de nodo que creará el módulo.

En los ficheros concretos de cada tipo de nodo estarían los hooks de inserción, actualización, carga, borrado, etc. El prefijo para la implementación de estos hooks debe ser el nombre de cada fichero secundario (o, mejor dicho, el nombre indicado en el hook node_info), no el del módulo. De esta forma se diferencia el comportamiento de los hooks para cada tipo de nodo.

8.4. Implementación del módulo `Catalogue`

El módulo `Catalogue` se ha implementado como un módulo que añade varios tipos de nodo, por lo que se ha creado un fichero llamado `catalogue.module` en el que se incluyen varios otros ficheros:

- *cat_musician:* Implementa músicos.
- *cat_band:* Implementa bandas de músicos.
- *cat_album:* Para los álbumes.
- *cat_photo:* Para el tratamiento de fotógrafos.
- *cat_common:* Incluye algunas funciones útiles para todos los demás ficheros.
- *cat_views:* Donde se incluyen las implementaciones de las diferentes vistas en Drupal.

Hay algunos ficheros más que dependen del módulo `Widget`, por lo que se explicarán en la sección 8.5.

Es importante destacar que aunque se ha separado la implementación de hooks de músico y de banda, la base de datos sólo tiene una tabla para artistas (ver figura de la página 97). Esto se ha hecho así porque la información de cada uno de ellos es algo distinta, y utilizar el mismo formulario para ambos supondría hacer una serie de artimañas del lado del cliente que complicarían mucho la implementación e introducirían posibles agujeros de seguridad.

La parte de la playlist, aunque forma parte del módulo, se explicará a parte en la sección 8.6.

8.4.1. Fichero `catalogue.module`

En este fichero se incluye el resto de ficheros del módulo, así como varios hooks y funciones importantes. Los hooks tienen el prefijo `catalogue`.

El hook help muestra un pequeño texto informativo en el menú de administración de los módulos.

El hook node_info indica al sistema los tipos de nodo que se van a crear. En este caso, `cat_musician`, `cat_band`, `cat_album` y `cat_photo`.

El hook perm incluye los distintos tipos de permiso para los usuarios: crear álbum, ver álbum, editar artista propio...

El hook menu incluye la asignación de funciones a ciertas rutas, entre las que se encuentran el formulario de administración del módulo, diversas funciones de *typeahead* o la función que devuelve la playlist.

La función que define el formulario de administración, `catalogue_admin_settings()`, permite modificar dos parámetros de la playlist. Por un lado, la url de la playlist XML proporcionada por Live365 y, por otro, el número de últimas pistas reproducidas a mostrar en el bloque de la playlist en la página. Usando la misma API de formularios de Drupal se define una función para validar los datos introducidos en dicho formulario, `catalogue_admin_settings_validate`. Esta función también está implementada en este fichero.

El hook block define los diversos bloques que se quiere que vayan incluidos en el módulo por defecto, como el de la Playlist o el de los links para escuchar la radio con distintos reproductores.

Con el hook form_alter se le dice a Drupal que los únicos nodos que deben ir por defecto a la portada del sitio web son los posts. Eso se hace modificando la casilla que traen por defecto los formularios de todos los nodos. Su valor puede ser cambiado manualmente en el formulario en el momento de crear o editar, pero por defecto estará activada o desactivada según el tipo de nodo.

El resto de funciones incluidas en este fichero son implementaciones de hooks y funciones proporcionadas por el módulo `Views`, que sirven para crear listas de nodos. En nuestro caso, necesitamos crear en las fichas de artistas (tanto músicos como bandas) una lista de álbumes que tienen asociados, una lista de imágenes y una lista de posts, y lo mismo para los álbumes. Los fotógrafos también incluyen una galería de imágenes. Con estas funciones se crean unas vistas por defecto, que no son más que consultas a la base de datos especificadas de una forma concreta. El resto de la implementación de estas vistas está en los ficheros secundarios del módulo.

8.4.2. Ficheros `cat_musician.php` y `cat_band.php`

Ambos ficheros son similares en estructura, pero tienen detalles que los diferencian, para distinguir entre ambos tipos de artista.

Ambos implementan el hook form, que crea el formulario de inserción y actualización de los datos del artista. Tienen un nombre, una descripción, un website y un id de MySpace comunes, así como una imagen que se les puede adjuntar gracias al módulo `image`. También tienen una lista de enlaces que se implementa con el módulo `Widget` (sección 0) el cual, como se ha dicho, sirve para crear listas de tamaño variable de distintos tipos de elementos. Luego los formularios tienen la parte diferente que obligó a separar los músicos y las bandas en ficheros diferentes, y son los instrumentos y las formaciones, también implementados con el módulo `Widget`. Un músico tiene una lista de instrumentos, mientras que una banda tiene una formación compuesta por músicos e instrumentos. Ambas listas pueden ser rellenadas automáticamente marcando una casilla del formulario. Si se marca

esta casilla se intentará crear una lista de instrumentos o una formación a partir de la información de los álbumes del artista.

El hook insert llena la tabla `catalogue_artist` de la base de datos con la información del formulario, además de crear un término con el nombre del artista para la taxonomía Artists, que servirá para, por ejemplo, asignarle posts o imágenes.

El hook update hace lo mismo que el insert pero además incluye la opción de rellenar instrumentos o formaciones a partir de sus álbumes (en el insert no tenía sentido, ya que un artista inexistente no tiene aún álbumes). Se hacen, además, algunas modificaciones relacionadas con la comunicación entre MLE y la base de datos para los temas de un artista, que también están incluidas dentro de las responsabilidades de este PFC.

En los ficheros de músicos y bandas hay, respectivamente, las funciones que insertan los instrumentos y las formaciones de forma automática.

El hook load devuelve los datos del artista obtenidos de la BD, pero además se encarga de seleccionar los elementos correctos para las vistas (del módulo `Views`): posts, álbumes e imágenes del artista.

Se ha decidido que el hook delete se encargue de eliminar no sólo el artista sino también todos sus álbumes y su nombre de las formaciones en las que se encontraba.

El hook view implementa la forma de mostrar el nodo tanto si es teaser (resumen) como si no.

Finalmente, hay una función que define las búsquedas en la base de datos para campos de typeahead.

8.4.3. Fichero `cat_album`

El fichero `cat_album` es parecido a los anteriores. En su formulario se define, además del título del álbum y una descripción, un campo para elegir el autor. Esto se ha implementado con un campo de typeahead que va mostrando los nombres de los artistas a medida que se escribe. También tiene año, género, un website y una formación de músicos parecida a la de la banda, pero añadiendo a los músicos y los instrumentos la lista de temas en los que interviene cada uno, e incluye del mismo modo la posibilidad de rellenar esta formación automáticamente con los datos del autor del álbum (músico o banda).

El hook insert introduce en la tabla de álbumes la información del formulario, además de crear un término para la taxonomía de álbumes. También inserta los valores necesarios en la tabla de sellos discográficos, `catalogue_label` y rellena la formación si se ha indicado así en el formulario.

El hook update es análogo al insert excepto porque se incluye información de la conexión de MLE a la base de datos.

La función `auto_album_performers()` es la encargada de llenar la formación.

El hook load no tiene diferencias importantes con los de los artistas, a excepción de que en las vistas sólo están los posts (no hay álbumes ni imágenes como en los artistas).

Eliminar un álbum con el hook delete supone borrar también su formación.

El hook view hace lo mismo que el de artista, diferenciando entre teaser y nodo completo.

Por último, también cuenta con una función de typeahead.

8.4.4. Fichero `cat_photo`

La implementación de los hooks en este fichero de fotógrafos es más sencilla que en los otros, ya que no se incluyen campos añadidos que hagan uso del módulo `Widget` ni otros campos a parte de uno para el website del fotógrafo.

El insert y el update añaden el nombre del fotógrafo a la taxonomía Photographers y el load introduce información en la vista que servirá de galería de imágenes del fotógrafo. Por lo demás, todo funciona de la misma forma que en los anteriores.

8.4.5. Fichero `cat_common`

Este fichero tiene algunas funciones comunes o que sirven a varios otros archivos, como consultas a la base de datos por nombre, funciones de autocomplete (typeahead) que no estuvieran en otro sitio (para sellos discográficos, por ejemplo), así como algunas búsquedas de términos en taxonomías concretas no ofrecidas por Drupal o una función para validar URLs.

8.4.6. Fichero `cat_views`

Este fichero tiene algunas funciones comunes para dar soporte a los tipos de contenidos nuevos , en concreto aportan funcionalidad sobre las diferentes vistas que se tienen que implementar en Drupal como por ejemplo , mostrar los álbumes de un artista , los post de un álbum...

8.5. Implementación del módulo `Widget`

Éste es un módulo desarrollado específicamente para este PFC, aunque partiendo de un módulo de ejemplo.

Hay varios elementos en el módulo `Catalogue` que necesitan crear grupos de tamaño variable de elementos iguales (una especie de arrays dinámicos pero con elementos de formulario). Por ejemplo, los artistas tienen que incluir una lista de enlaces a páginas web relacionadas con ellos. A priori no se puede saber cuántos enlaces va a necesitar un artista determinado, por lo que sería ideal añadir una funcionalidad en los formularios que permitiera crear tantos como se desee.

El problema era que añadir campos nuevos a un formulario de forma dinámica era algo que había que hacer desde el lado de cliente, lo cual suponía que Drupal, del lado del servidor, no llegaba a saber que se habían creado esos nuevos campos y no era capaz de recoger sus valores. Para conseguir este efecto se estuvieron buscando y probando varios módulos y soluciones con funcionalidades Ajax sin resultado.

Finalmente, se dio con el módulo `AHAH Forms Framework` (http://drupal.org/project/ahah_forms), creado por Tao Starbow. Este módulo permite actualizar fragmentos de páginas sin tener que recargar la página completa. Hace uso de la FormAPI de Drupal, lo que le permite conectar el lado de cliente con el de servidor de una forma bastante natural. Hace uso de Javascript y evita al usuario la necesidad de utilizar ese lenguaje. Hay que recalcar la utilidad de este módulo, que ha hecho que sea incluido en el core de Drupal 6.

El módulo traía de ejemplo un pequeño módulo llamado `Widget` que precisamente creaba una lista de enlaces, justo lo que se necesitaba. El funcionamiento es sencillo (ver imagen en la página 207): hay un subformulario dentro del formulario general en el que se puede escribir un enlace y una descripción para éste. Una vez llenados los campos, se hace clic en el botón de actualizar y, sin recargar la página entera, ese enlace va a parar a una tabla justo encima, dejando libre el subformulario para insertar otros enlaces. Así se pueden añadir todos los que se quiera, con la opción extra de poder ordenarlos por *peso* (algo muy propio de Drupal) y de eliminarlos una vez que están en la lista.

Como era exactamente lo que se estaba buscando se utilizó directamente. Pero luego se observó que esa funcionalidad podía ser muy útil para otros aspectos de la web. Concretamente, un artista puede tocar uno o más instrumentos, y esos instrumentos a menudo se quieren mostrar en un orden concreto. Lo mismo ocurre con las formaciones de

músicos para bandas y álbumes: de entrada no se puede saber cuántos músicos intervendrán.

De modo que se estudió y modificó completamente el módulo para convertirlo en algo más genérico, que sirviera para cualquier tipo de elemento, construyendo una pequeña API que debía implementar cada uno de los tipos nuevos de *subwidgets* que se quisiera añadir al sistema. Lo que hay que hacer ahora para crear un nuevo tipo de subwidget es simplemente crear un nuevo fichero con el nombre del subwidget, incluirlo en el módulo con un `include` e implementar todas las funciones necesarias para que funcione. De cara a que todo esto se refleje en la base de datos, sus tablas deben ser creadas por el módulo al instalarse.

De esta forma, no sólo se pueden incluir varios widgets del mismo tipo en un formulario sino también de tipos distintos, ya que se ha añadido la opción de identificarlos por tipo y por nombre. Además, ahora se permite que a la hora de recoger los datos del formulario se cuente con la posibilidad de obtener, en vez del valor introducido en el formulario, un valor de la base de datos en función de ése. Por ejemplo, en nuestras tablas de formaciones los valores almacenados para instrumentos y músicos no son nombres, sino identificadores de sus tablas. Se puede indicar que en lugar de guardar el nombre, que es lo que se ve en el formulario, se obtenga de la base de datos el identificador de ese nombre y se inserte.

Veamos la implementación por partes.

8.5.1. Fichero `widget.module`

Cuando se quiere insertar un elemento en un formulario con Drupal, el aspecto que tiene ese elemento en el hook form es de este estilo:

```
$form ['title'] = array(
    '#type' => 'textfield',
    '#title' => t('Name'),
    '#required' => TRUE,
    '#default_value' => $node->title,
    '#maxlength' => 255,
    '#weight' => -5
  );
```

Cuando se inserta un elemento que va a utilizar el módulo `Widget`, la forma que tiene es así:

```
$form['musician_links']['additional_links'] = array(
      '#type' => 'widget',
      '#title' => 'Links',
      '#weight' => 4,
      '#process' => array(
            'widget_expand' => array(
                  $node, 'links'
            )
      )
);
```

De esta forma, se delega el proceso de este elemento en la función `widget_expand()`, ubicada en el fichero `widget.module`. Esta función se encarga de añadir a estos elementos la propiedad `#ahah_binding`, que es la que conecta con las funciones Javascript definidas por `AHAH Forms`. A continuación, se construye el elemento del formulario, que consiste en una

tabla con los subwidgets ya añadidos y un pequeño formulario para añadir nuevos. Los cuatro pasos para crear este elemento son:

- Conseguir los valores introducidos en el formulario.
- Comprobar si son correctos y procesar la adición del nuevo subwidget a los ya existentes, así como procesar las posibles eliminaciones.
- Convertir los valores en elementos de formulario (tabla superior).
- Añadir el formulario de introducción de datos.

El módulo incluye algunas funciones para la conexión con Javascript y luego pasa a la implementación del hook nodeapi que, como se ha explicado, sirve para que módulos que no crean tipos de nodo puedan manipular nodos creados por otros módulos. Las operaciones que implementa esta función son:

- ***insert:*** Crea un elemento nuevo en la tabla `widget` de la base de datos y tantos elementos en la tabla `suwidget_`*tipowidget* (por ejemplo, `subwidget_instruments`) como se hayan añadido. Para ello llama a las funciones específicas de cada subwidget, que son las que le dan los nombres de los campos que se van a utilizar, los valores a recoger (directos u obtenidos de la base de datos a partir de ellos), etc. Además, se ha incluido la posibilidad de hacer que no se haga nada. Esto puede ser útil si el desarrollador quiere saltarse los valores introducidos en el formulario por alguna razón. En nuestro caso se hace cuando queremos que los instrumentos o formaciones se rellenen automáticamente ignorando cualquier valor del formulario.
- ***update:*** Hace lo mismo que el insert pero eliminando antes los valores existentes de la base de datos.
- ***delete:*** Borra de la base de datos el widget y los subwidgets seleccionados.
- ***load:*** Carga de la base de datos los valores del widget y sus subwidgets.

Finalmente, se incluyen algunas funciones de theming.

8.5.2. Ficheros de subwidgets

Cada uno de los ficheros que representan un subelemento de un widget tiene que implementar varias funciones que son llamadas por el módulo `Widget`, entre las que se encuentran:

- Funciones para comprobar y validar los datos introducidos.
- Una función que crea los elementos de la tabla del formulario.
- Una función que crea el elemento de introducción de datos del formulario.
- Funciones de theming.
- Una función que devuelve los campos del subwidget.

La más interesante es ésta última, ya que no sólo da el nombre del campo y su tipo para que el módulo `Widget` sepa de dónde obtener los datos sino que además se incluyen las consultas opcionales que hay que hacer a la base de datos para conseguir el valor que se desea asociar a cada campo.

8.6. Implementación de la playlist

En todas las páginas del sitio, en el menú lateral, aparece la lista de reproducción de la radio, que incluye el tema que está sonando y los *n* temas anteriores (*n* está entre 0 y 9 y su valor es configurable desde las opciones del módulo `Catalogue`). Esta lista está sincronizada con la emisión de la radio y tiene una cuenta atrás del tema que está sonando

y que en el momento en que termina el tema actual vuelve a hacer una petición al servidor para conseguir la nueva lista.

8.6.1. Lado del servidor

Esto significa que cada vez que alguien visite una página del sitio hará una petición para obtener esa lista y cada vez que termine de sonar un tema en la radio todas las personas que estén conectadas a la web en ese momento pedirán al servidor la nueva lista de reproducción. Por lo tanto, pueden producirse retrasos importantes si hay varias personas conectadas, además de que pueden sobrecargar el servidor.

Para evitar estos problemas se ha diseñado un protocolo de comunicación que reduzca al máximo el número de conexiones. Pero antes veamos los diferentes elementos que intervienen en el proceso:

- **Cliente**: Está visualizando la página con un contador que tiende a 0.
- **Servidor**: Sabe cómo mostrar la playlist, dónde está la lista original y cuántos temas anteriores hay que mostrar. Para poder generar la playlist para el cliente, lo primero que debe conocer es la lista de temas.
- **Live365**: Ofrece una playlist en formato XML que incluye el tema que suena actualmente y los 9 anteriores. La playlist está siempre en una URL concreta y tiene un campo en el que indica el número de segundos que falta para que empiece un nuevo tema. Este valor lo actualiza cada vez que alguien pide ese fichero XML.

Por lo tanto, el orden lógico de proceder en el peor de los casos, que es cuando el contador de la playlist llega a cero y todos los usuarios pedirán la playlist, sería el siguiente:

1. El cliente está leyendo una página en el momento en que va a empezar a sonar un nuevo tema en la radio, y hace una petición al servidor para obtener la nueva lista.
2. El servidor recibe la petición y pide a Live365 su lista en XML.
3. Una vez recibida la lista, el servidor genera una nueva lista con los *n* temas según la configuración del sitio web.
4. El servidor envía esta lista al cliente, que la actualiza para reflejar el cambio de tema.

Este modelo es correcto, pero es altamente ineficiente, ya que cada vez que cambie el tema y varios clientes le pidan la playlist el servidor, éste tendrá que conectarse a Live365 tantas veces como clientes se la pidan para acabar dándoles a todos prácticamente lo mismo. La única diferencia en la lista que les dará es el tiempo que falta para que empiece a sonar otro tema, ya que los primeros que lleguen tendrán un número de segundos mayor que los últimos.

Un sistema que reduciría eso sería hacer que el servidor sólo tenga que acceder a Live365 la primera vez, que se guarde la lista de reproducción y desde entonces sólo tenga que dársela al resto de clientes. Para optimizarlo más, el servidor no generará la lista de reproducción a partir del XML de Live365 cada vez sino que se creará una playlist propia que tenga sólo el número de temas necesario.

Por lo tanto, sólo quedan tres detalles a resolver en el lado del servidor:

- ¿Cómo sabe el cliente que la lista que tiene almacenada sigue siendo válida y que aún no tiene que pedir una lista actualizada a Live365?
- ¿Cómo se dice a cada cliente el momento en que tiene que volver a hacer la petición, dado que ahora se tiene una misma playlist para todos?
- ¿Cómo se bloquea al resto de usuarios mientras el primero hace que el servidor pida el XML a Live365?

Para resolver lo primero se ha optado por lo siguiente: cuando el servidor recibe el XML, en él encuentra el número de segundos que faltan para el próximo tema, pongamos 180. En ese momento el servidor mira la hora actual y le suma esos 180 segundos, de modo que ahora puede trabajar con horas absolutas en lugar de relativas y sólo con mirar si ha pasado la hora indicada sabe si su lista ha quedado obsoleta.

Pero el cliente sigue trabajando con tiempo relativo, ya que lo único que quiere saber es cuándo tiene que realizar la siguiente petición, así que eso no responde a la segunda pregunta. La solución consiste en que, cuando unos segundos más tarde (5 segundos, por ejemplo) otro cliente pida la playlist, el servidor le devolverá por un lado la lista de temas y por el otro el tiempo en segundos que falta para que termine el tema actual (175 segundos), que calculará haciendo una resta entre la hora de actualización que tiene guardada y la hora actual.

La tercera pregunta ya corresponde a temas de tecnologías de implementación. Se pensó en la posibilidad de controlar la concurrencia con semáforos. PHP incluye funciones para trabajar con semáforos, pero nos encontramos con que el servidor en el que teníamos alojado el sitio de desarrollo no había compilado PHP con la opción `--enable-sysvsem` requerida, por lo que nos vimos obligados a buscar alternativas, ya que era muy probable que otros hostings tampoco dispongan de semáforos. Se pensó en utilizar una variable que hiciese las veces de semáforo (bloqueado/desbloqueado), para lo cual se necesitaba memoria compartida. De nuevo, PHP proporciona esa funcionalidad pero el hosting no la tenía (esta vez, la opción de compilación era `--enable-sysvshm`).

La solución se encontró en el almacenamiento en un fichero de una playlist intermedia que tuviera los datos provenientes de Live365 y el tiempo de actualización en valor absoluto. Cada vez que hubiese una petición masiva de la playlist, el primer proceso en llegar abriría el fichero para lectura/escritura. Eso hace que el fichero quede automáticamente bloqueado para otras lecturas, de modo que el resto de clientes queda esperando. Con el fichero abierto, se comprueba si hay que actualizarlo o no. Como el tiempo está en valor absoluto, el servidor comprueba si la hora que marca ha pasado ya y, si es así, pide a Live365 una nueva lista de reproducción en XML y genera la nueva playlist intermedia. Cuando la tiene, y una vez cambiado el tiempo al número de segundos que faltan para el nuevo tema, se la da al cliente y se cierra el fichero. En ese momento, el segundo cliente, que estaba esperando a tenerlo libre, lo abre para L/E, dejándolo bloqueado. Comprueba si la hora actual es posterior a la que indica ese fichero y, al ver que no (es decir, que está actualizado), se cierra el fichero, dando paso al tercer cliente, y se le da al segundo la playlist con el nuevo tiempo calculado. Lo mismo ocurrirá para todos los que vengan luego, de modo que se elimina el largo tiempo de petición de la playlist a Live365 y en muy poco tiempo todos ellos pueden obtener los mismos datos.

En caso de que alguien esté navegando y tenga que cargar la playlist por acabar de visitar una nueva página, el proceso será idéntico. Como la playlist seguirá actualizada, se le dará la que hay en el servidor pero con el tiempo restante adaptado al momento de la petición.

En cuanto a la generación de la playlist intermedia, la estructura es también un XML parecido al de Live365 y consiste en lo siguiente:

```
<Playlist>
	<Refresh>1214374285</Refresh>
	<PlaylistEntry>
		<Title>
			<name>IMAGINATION (Van Heusen-Burke)</name>
			<url></url>
		</Title>
		<Artist>
			<name>Tete Montoliu</name>
```

```
                <url>/musician/tete-montoliu</url>
            </Artist>
            <Album>
                <name>Lush Life</name>
                <url>/album/lush-life</url>
            </Album>
            <Seconds>318</Seconds>
        </PlaylistEntry>
        <PlaylistEntry>
            <Title>
            <name>Venado tuerto</name>
            <url></url>
            </Title>
            <Artist>
                <name>Jordi Bonell, Dani Pérez</name>
                <url></url>
            </Artist>
            <Album>
                <name>Duo</name>
                <url></url>
            </Album>
            <Seconds>371</Seconds>
        </PlaylistEntry>
</Playlist>
```

En este caso la playlist cuenta con el tema actual y con un único tema anterior, pero si hubiese más se verían debajo de los dos. Como ésta es la playlist que almacena el servidor, el tiempo de actualización (campo `Refresh`) es un timestamp en formato Unix. Si fuese la que se envía al cliente, el valor sería menor o igual a 318, ya que es el tiempo que dura el tema que está sonando actualmente. El resto sería idéntico.

Otro hecho a observar es que el primer tema contiene una URL para el músico y otra para el álbum, mientras que el segundo carece de esas URLs. Las URLs son las que vuelven a unir la información que ha viajado hacia Live365 dentro de los metadatos de los MP3 y la que ha ido directamente a la base de datos del sitio web. El servidor genera estas URLs con la información que encuentra en la base de datos a partir de los nombres del artista y del álbum.

Si, por lo que sea, no existe o no se encuentra esa información en la BD, los campos se dejan vacíos. También se dejan vacíos si el artista o el álbum no están publicados. Actualmente no se pone una URL para los temas por las razones que se han dado en la sección 2.3.4, pero en el futuro se podría querer poner, por lo que se ha añadido el campo en el XML.

8.6.2. Lado del cliente

El otro problema que supone la playlist es que tiene que actualizarse de forma asíncrona, de modo que tiene que poder saber cuándo pedir una nueva lista de reproducción. Ya se ha explicado que la playlist que recibe el cliente contiene el número de segundos que faltan para la próxima actualización. Esta información es suficiente para generar unos eventos utilizando un lenguaje del lado del cliente como es Javascript con las funciones

`setTimeout()` para el momento de actualización y `setInterval()` para el contador de segundos que se decrementa hasta llegar a 0.

Haciendo uso concretamente de jQuery, se generan unas llamadas Ajax que acceden a una URL del servidor. Utilizando el hook menu del fichero `catalogue.module` se relaciona esa URL con la función que devuelve la playlist. Esta lista de reproducción está en formato XML y el mismo código Javascript lo parsea y obtiene los datos para crear la playlist que se mostrará al usuario. El usuario verá la lista de temas, álbumes y artistas junto con el tiempo que dura cada uno. En caso de que el artista y el álbum tengan URL se mostrará un enlace que llevará a sus respectivas fichas, como se ve en la Figura 8.1.

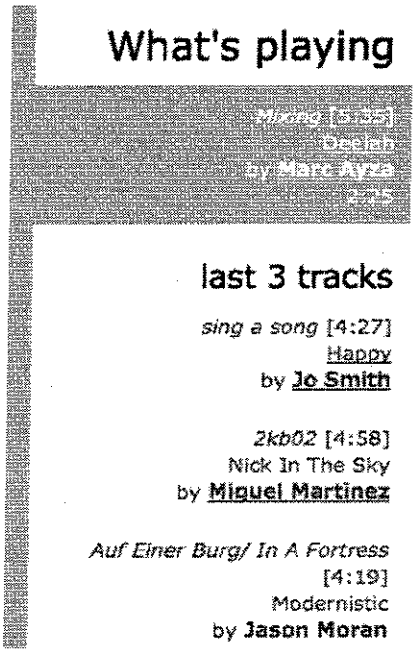


Figura 8.1: Playlist

El código Javascript está dividido en dos ficheros, uno de cliente que se encarga de hacer las peticiones en el momento que toque y otro de servidor, generado por PHP y necesario para mostrar la playlist al inicio del proceso, cuando se carga la página.

8.7. Implementación de la capa de usuario y SEO

La capa de usuario se genera por varias vías usando Drupal.

- Las funciones de theming de los módulos indican qué mostrar, en qué orden, etc.
- Los ficheros .tpl de los themes indican a Drupal cómo mostrarlo
- Los ficheros .css de los themes dan forma a la estructura y a cada elemento concreto de la página

Para este proyecto se ha utilizado un theme de SEOposition (www.socialseo.com) adaptado a nuestras necesidades. Se ha modificado parte de la estructura, tamaños, tipos de letra, colores, etc. Es un theme optimizado para las necesidades SEO básico de los sitios web, en el que los bloques de la web están colocados de la forma más correcta.

De hecho, los requisitos SEO se implementan principalmente en la capa de usuario, ya que es la misma que *perciben* los buscadores cuando leen una web. Para cumplir algunos de los requisitos se ha instalado el módulo Pathauto junto con los que requiere para funcionar, Path y Token. Con ellos se generan URLs amigables tanto para humanos como para buscadores.

La playlist también se ha adaptado a las características pedidas en la sección 3.3.12, aunque aquí el código no se ha generado totalmente con las funciones que proporciona Drupal para ello sino que ha habido que hacerlo de una forma más *ad hoc*, ya que intervenía también una parte de Javascript utilizando funciones para modificar la estructura DOM del documento. El DOM o Document Object Model es la construcción lógica de un documento realizado con lenguajes de etiquetado como XML o XHTML.

8.8. Implementación de la sincronización entre MLE y Drupal

8.8.1. Introducción

Se puede decir que el catálogo musical de la radio está contenido en dos lugares: por un lado, en una base de datos de artistas, álbumes, temas etc. y, por otro, en las etiquetas o tags ID3 de la librería de temas musicales en formato mp3 a los que hace referencia dicho catálogo. En cierto modo, el conjunto de etiquetas ID3 contenidas en los diferentes ficheros mp3 puede ser considerado como una base de datos distribuida en archivos de audio sin integridad referencial asegurada.

El catálogo musical que se encuentra representado en la base de datos es accesible públicamente y en formato HTML a través del sitio web o RW (Radio Website), que ha sido implementado mediante el sistema gestor de contenidos Drupal. Así pues, el Catálogo está almacenado en la base de datos que crea y gestiona Drupal para el sitio web RW.

Ambas representaciones del Catálogo deben ser mantenidas en sincronía: colección de etiquetas ID3 y base de datos de RW. El sistema encargado de ello es la aplicación de escritorio MLE (Music Library Editor), que permite al usuario extraer la información almacenada en las etiquetas ID3 de la librería musical, editarla y completarla con información adicional no incluida en dichas etiquetas e insertarla en la base de datos. Adicionalmente la información del Catálogo se puede crear y editar también desde RW, siendo entonces MLE el sistema encargado de actualizar esos cambios en la colección de etiquetas ID3. MLE permite modificar dichos datos de forma sincronizada en ambas partes y así mismo es capaz de detectar cuanto los datos han sido modificados en RW y replicar dichas modificaciones en los tags ID3.

El escenario operativo más probable es que dicho sitio web se aloje en un servidor compartido contratado a un proveedor de servicios de Internet o, cuando menos, en una máquina no administrada por la persona que introduce los datos en el Catálogo a partir de los mp3 mediante MLE. Habitualmente, los ISPs deshabilitan por cuestiones de seguridad el acceso remoto a las bases de datos, en especial en su servicio de alojamiento en servidores compartidos.

Esto supone un problema a la hora de mantener sincronizadas las dos representaciones del Catálogo, ya que para realizar las operaciones de sincronización es preciso que MLE tenga acceso remoto a la base de datos y es de esperar que éste esté deshabilitado.

Elección del procedimiento

En resumen, lo que se hace de forma muy general, es enviar unos datos desde MLE a una interfaz que trata esos datos y los inserta/actualiza en Drupal, esta interfaz debe tener en cuenta la reglas que tiene Drupal de mantener la integridad de su BBDD y aplicar estas mismas reglas, tal y como un usuario lo haría si utilizara la interfaz de formularios de Drupal. Estas regla sobre cómo Drupal gestiona estas reglas se explica más extensamente en el capítulo 8.8.3.

En un principio, se plantearon dos posibles soluciones:

- ***Sincronización***: El MLE trabaja sobre las etiquetas de los archivos mp3 y sobre una base de datos local. A continuación, se hace una copia de esa BD en el servidor compartido y se activa un programa de sincronización entre la copia de la BD local y la BD del sitio web RW. El último paso consiste en hacer una copia local de la BD ya sincronizada en el servidor compartido.

- ***Acceso remoto***: El MLE trabaja igualmente sobre las etiquetas de los archivos mp3 pero en lugar de utilizar una BD local, utiliza directamente la BD remota de RW usando como intermediario algún sistema alojado en el servidor y, por tanto, con acceso a la misma.

En ambos casos es necesario construir una capa intermedia de comunicación, o bien el programa de sincronización, o bien el sistema intermediario. Se ha optado por la segunda opción por varias razones que se exponen a continuación.

Por un lado, periódicamente se selecciona un subconjunto de los temas de la librería musical para ser emitido en la radio. A estos subconjuntos de temas en formato mp3 se les llama playlist. Copias de esos mp3 son subidas al servidor de streaming de la radio bifurcando el flujo de los datos del sistema. Ahora el catálogo está ubicado, total o parcialmente, en tres sitios diferentes: 1) la librería musical o colección original de mp3, 2) la base de datos del sitio web y 3) los mp3 alojados en el servidor de streaming para su emisión en la radio. Añadir una cuarta versión de esos datos no parece lo más sensato. Una nueva versión del Catálogo crearía un sistema demasiado complejo y multiplicaría las consecuencias negativas.

Como se ha dicho es necesario mantener la información del Catálogo en sincronía. Añadir una base de datos local que se sincroniza con la del sitio web mediante procesos manuales introduce, como mínimo, retrasos importantes en los procesos de sincronización, dando lugar a problemas y errores durante esos períodos en los que no todas las versiones están sincronizadas.

El último argumento consiste en la complejidad y dificultad de uso que para el usuario supone el primer sistema, ya que ha de realizar unas operaciones manuales que requieren tiempo y una destreza técnica que no es razonable esperar que posea.

Por estas razones se ha optado por construir un sistema intermediario entre MLE y la base de datos de RW, que se ejecute en el servidor y proporcione así el acceso remoto a la base de datos que de forma directa no es posible por estar deshabilitado por el ISP.

Conceptos Previos

Antes de empezar, se van a explicar varios conceptos necesarios para la comprensión de esta documentación.

- ***Servicio:*** Servicios web con protocolo SOAP implementados en PHP con las librerías nuSOAP. Se ejecutan en el servidor web y ofrecen a MLE las funciones que necesita para acceder a la base de datos de RW, estos servicios a su vez llamarán a uno o más "sprocs".

- ***Procedimiento almacenado (sproc):*** Función escrita en PHP que hace una cierta transacción en la BBDD de Drupal, NO hay que confundir con un Stored Procedure de MySQL. Aunque este nombre pueda causar confusión, se decidió llamarlo así porque originalmente estaba previsto implementar esta capa de acceso a los datos mediante stored procedures, pero que por restricciones impuestas por el entorno de ejecución no fue posible y se hubo de implementar como una serie de funciones en código PHP.

- ***Interfaz de conexión:*** Es la capa PHP, donde están implementados todos los Sprocs y servicios necesarios para que se pueda ejecutar la sincronización.

Tecnología usada en capa intermedia entre MLE y la base de datos de RW

El sistema intermediario que se ha decidido construir ha de ofrecer a MLE una serie de servicios que le permitan realizar las operaciones de inserción, actualización, eliminación y consulta sobre la base de datos de RW que necesita.

Las características de este sistema son pues propias de una arquitectura orientada a servicios. Tras una investigación de las posibilidades existentes y teniendo en cuenta que Drupal está implementado en PHP y que este lenguaje de scripting se ofrece prácticamente en todos los servicios de alojamiento en servidores compartidos, se optó por implementar unos webservices en PHP en el lado servidor, utilizando el protocolo SOAP para la comunicación entre servidor y cliente.

Se ha elegido utilizar para la implementación las librerías NuSOAP que proveen de un kit de herramientas para implementar webservices bajo PHP. NuSOAP está en una fase madura de desarrollo, no necesita módulos adicionales y es muy fácil su instalación.

8.8.2. Esquema de funcionamiento de la reingeniería inversa de los procesos de Drupal.

¿Por qué una reingeniería inversa?

Respecto a la función de los servicios web, hay algo importante a destacar, y es que Drupal, necesita hacer uso de entidades propias, representadas en una multitud de tablas de la BD. Estas entidades no serían necesarias en caso de tener un esquema de base de datos que representase exclusivamente el diagrama de clases propio del Catálogo, pero deben ser tenidas en cuenta a la hora de hacer modificaciones externas a la BD.

Por ejemplo, si desde RW se inserta un artista en la base de datos, no sólo se creará una nueva entidad artista, sino que Drupal añadirá o modificará varias de sus propias entidades, como pueden ser nodos, autores de contenido, etc. Además, el diseño del RW incluye aún otros elementos que no coinciden del todo con el diseño de MLE y que también son modificados en las distintas operaciones de la BD. Si esto no se tiene en cuenta a la hora de introducir artistas desde MLE, se producirán desajustes en el esquema de funcionamiento de Drupal que luego no permitirán mostrar correctamente la información e introducirán fallos de seguridad y de contenido.

Por lo tanto, los servicios web deben hacer una doble función cada vez que se pretende insertar, modificar o eliminar algún dato desde MLE: por un lado, actualizar la base de datos de acuerdo con el diagrama de clases del Catálogo y por el otro, actualizar el resto de clases de Drupal y de RW. Lo mismo ocurre con las consultas; para el MLE sólo es pedir un artista, pero para el servicio web supone hacer varias consultas complementarias.

Esto requiere un estudio de todos los cambios que se efectúan en la base de datos cada vez que se ejecuta un caso de uso del catálogo. El resultado de este estudio será el conjunto de operaciones a realizar por cada servicio cuando es llamado por MLE.

Ejecución en la base de datos

En el momento en que es llamado un servicio, éste debe hacer varias operaciones en la base de datos para devolver el resultado correspondiente. Para agilizar el proceso, se pensó en usar stored procedures en la ejecución de los servicios. Los stored procedures (sprocs o SP) son rutinas almacenadas en la propia base de datos, y entre sus ventajas están la seguridad y el rendimiento. En nuestro caso, además, servirían para ocultar a la lógica de los servicios los detalles de implementación específicos de Drupal.

El uso de stored procedures permite hacer consultas más complejas o que reciban un mayor número de parámetros y aún así se ejecutarán más deprisa que haciéndolas del modo tradicional. Al estar más cerca de los datos, sus ejecuciones son de muy bajo nivel y por tanto, más rápidas. Además, el lenguaje SQL proporciona sentencias condicionales, bucles, etc. que permiten hacer lo mismo que se haría con varias llamadas a la BD pero de forma mucho más ágil.

Pero, como se ha comentado antes, las versiones del software del servidor no permitían utilizar stored procedures, por lo que finalmente hubo que saltarse este paso y hacer las llamadas directamente desde la capa de datos de los servicios.

Funcionamiento de los servicios de accesos a la base de datos

Como ya se ha dicho, los accesos a la base de datos que MLE requiere serán tanto de inserción, modificación y borrado, como de consulta. Así pues, los servicios que se desean implementar deben proporcionar a MLE una capa de abstracción tal que le permita realizar esas operaciones sobre la base de datos sin necesidad de lidiar con las especificidades del esquema de la base de datos propio de Drupal, ni con los detalles propios de configuración del alojamiento compartido. Es decir, se desea proporcionar unos servicios que proporcionen acceso a la base de datos remota en condiciones parecidas a las que tendría de tratarse de una base de datos local.

Debido a la distribución de tareas entre los miembros del equipo de proyecto y otros requerimientos temporales marcados por la planificación, la implementación de MLE se ha programado antes de la implementación de estos servicios. Por ello, la primera versión de MLE utilizaba el acceso a una base de datos local, pero la implementación, en lenguaje Java, se ha realizado teniendo en cuenta que el código de acceso a la base de datos se localice en unos métodos concretos para separarlos al máximo del resto de la lógica. De esta forma, hacer los cambios necesarios para sustituir el acceso a una base de datos local por el acceso a la base de datos remota de RW a través de los servicios ha requerido mínimas y controladas modificaciones.

NuSOAP hace la transferencia de datos usando el estándar XML (eXtensible Markup Language). El esquema típico de funcionamiento es el siguiente:

- El cliente (MLE) realiza llamadas a los servicios implementados en PHP que se ejecutan en el servidor compartido a través de Internet, encapsulando los datos que desea enviar a la base de datos en formato XML.
- El servicio llamado recoge y desencapsula los datos.
- El servicio ejecuta las operaciones solicitadas sobre la base de datos.
- El servicio recoge la información solicitada de la base de datos, así como otros datos resultantes de las operaciones solicitadas, los encapsula nuevamente en formato XML y los devuelve al cliente de la misma forma.
- El cliente recibe y extrae los datos devueltos y continúa la ejecución con las operaciones siguientes.

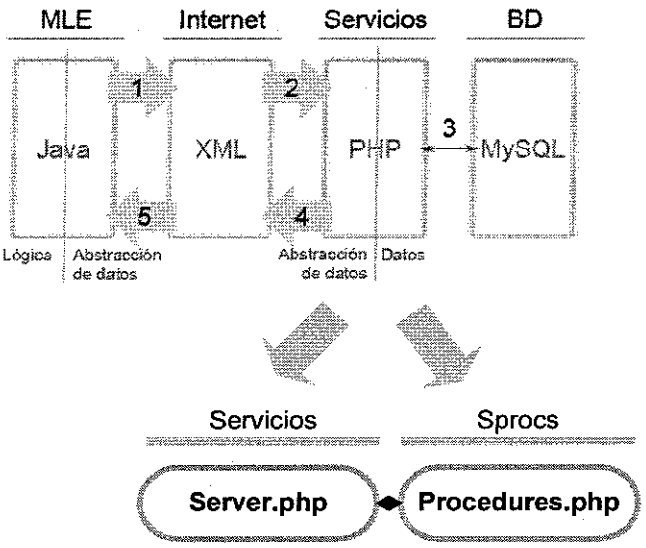


Figura 8.2 : Esquema general del proceso de sincronización

El lado del cliente (MLE) debe implementar una capa de abstracción que reciba peticiones muy generales y, en vez de convertirlas en una o varias consultas a una base de datos local, las envíe al servidor mediante llamadas a los webservices, encapsulando los datos intercambiados en formato XML (1).

Al recibir la respuesta (5), extraerá los datos obtenidos en formato XML (ya sean valores devueltos de consultas o resultados de las operaciones de inserción, actualización o borrado solicitadas) y continuará la ejecución utilizándolos como corresponda.

Por su lado, el servidor recibirá las peticiones generales en XML (2), su capa de abstracción las desglosará y usará los parámetros para construir peticiones concretas que ejecutará en la base de datos (3). Los resultados los pasará a formato XML y los enviará de vuelta al cliente (4).

La capa de abstracción del lado del MLE sirve para separar la lógica del programa del acceso a la BD, evitando que la capa de dominio necesite conocer los detalles de almacenamiento de los datos.

La capa de abstracción del lado de los servicios web tiene dos funciones:

- Las peticiones de consulta, inserción o eliminación de datos viajan a través de Internet, lo que supone un tiempo de respuesta muchísimo mayor que si se accediera a una BD local. Para reducir ese tiempo, una buena medida es reducir al máximo el número de conexiones, y eso se consigue haciendo las operaciones de los servicios lo más abstractas posible, de modo que en una única petición XML se envíen todos los datos necesarios para resolver una operación, lo que incluye la operación y todos sus parámetros. El envío de más datos en una petición es un problema mucho menor en este caso que el hecho de hacer peticiones cortas y numerosas, ya que la cantidad de datos enviada en cada petición es más bien pequeña. El MLE delega, pues, en el servicio la descomposición de la petición en operaciones más pequeñas.

- Una vez recibida la petición, el servicio decide cuántas y qué operaciones necesita realizar en la base de datos, y hace las llamadas necesarias para poder devolver los resultados a través de XML (de nuevo, devolverá juntos todos los resultados para

evitar múltiples conexiones). El modo en que se ejecutarán finalmente esas consultas en la BD se explica en los siguientes apartados.

8.8.3. ¿Cómo se ha hecho la reingeniería?

Con anterioridad, se ha comentado el porqué es necesario realizar una reingeniería inversa de la BD de Drupal. En este capítulo se expone el método utilizado para realizar dicha reingeniería.

Básicamente y de forma resumida, se investigó la forma que tiene Drupal de implementar su funcionalidad, para ello se plantearon unas pruebas genéricas (creación, actualización, eliminación de un nodo) y se comparó el estado de la BD antes de la operación y justo después de ésta. Con este primer paso observamos que entidades y que tablas de la BD creaba/actualizaba/utilizaba el CMS. Un segundo paso fue investigar los posibles detalles de implementación que no se deducían del primer paso como es el caso de cómo Drupal incrementa los identificadores de las entidades, entre otros.

Como resultado de esta investigación se vio que Drupal constaba de nodos y que éstos eran la unidad básica de contenido y cada vez que se crea un nodo se rellenan una serie de tablas en la BD.

¿Qué se tiene que sincronizar?

Antes se ha expuesto que se debe mantener una sincronía entre dos bases de datos, por un lado, una base de datos de artistas (que pueden ser un músico o una banda), sus álbumes, las imágenes tanto de los artistas como de los álbumes y los temas. Y, por otro, las etiquetas o tags ID3 de la librería de temas musicales en formato mp3 a los que hace referencia dicho catálogo.

La piedra filosofal de estos datos son los temas, éstos deben estar perfectamente sincronizados con los tags ID3 de los MP3 ya que en esta premisa se basa la sincronización, este concepto se explica de una forma más extendida en el capítulo 0.

¿Cómo gestiona Drupal los nodos?

Como se ha comentado, los nodos son la unidad básica de contenido en Drupal y están identificados de forma única por un identificador llamado ***nid,*** cada nodo al ser actualizado crea una copia llamada revisión que sirve para poder tener un historial de los cambios de cada nodo.

También cada nodo se puede relacionar con otros, sean de su mismo tipo o no, la implementación de éstas relaciones se realizará mediante taxonomías que son el mecanismo estándar que nos proporciona Drupal para en un primera instancia categorizar el contenido (es decir los nodos) y en segunda instancia nos permite relacionar nodos (menos los nodos de tipo imagen que éstos se relacionarán mediante otro método proporcionado por Drupal llamado attach_image, éste explicado más adelante).

De la misma forma que los nodos se pueden relacionar entre sí también se pueden categorizar por temas usando taxonomías.

Cada nodo, tiene una url asociada que se debe transformar es una url estructurada de forma que los principales buscadores puedan interpretar de forma más eficiente, a éstas url se les denominan coloquialmente *"url's limpias"*. La sincronización debe gestionar las transformación de *"url's Drupal"* a *"url's limpias"*.

Secuencias en Drupal

La versión de Drupal utilizada en este proyecto gestiona de un modo muy especial los campos autoincrementables (como por ejemplo: identificador del nodo, identificador de ficheros, identificador de usuarios...).

Para hacer esta gestión de contadores, Drupal utiliza una tabla auxiliar llamada ***sequences*** para almacenar los últimos valores de los campos autos numéricos, el gestor de BD *NO* se encarga de actualizar la secuencia automáticamente, ésta se debe ir actualizando y gestionando mediante programación.

Por tanto, a la hora de insertar un nodo, un fichero o una revisión de un nodo se debe consultar este valor, usarlo y actualizarlo incrementándolo en 1 unidad mediante programación.

Por ejemplo, si el último nodo insertado tiene por nid = 100, la tabla ***sequences*** contendrá la tupla (nid, 100), pues si se insertara un nuevo nodo se debería incrementar el contador y actualizar la tabla sequences sustituyendo 100 por 101, es decir la tupla (nid, 101).

Hay que destacar que las versiones posteriores del CMS no siguen este método sino que el gestor de las bases de datos ya utiliza los campos autoincrementables y prescinden de la tabla ***sequences*** aquí comentada.

Direcciones URL limpias

Al sincronizar la información con MLE, nos debemos cerciorar que las url hacia esa información sean "limpias" y que cumpla con los requisitos SEO especificados en el documento de diseño.

Para conseguir esto, se debe actualizar la tabla path_auto, en concreto se inserta la url que apunta al nodo que se crea/actualiza y por otra parte se inserta el destino, este destino debe contener la url "limpia" hacia el contenido sincronizado.

Consideraciones sobre las revisiones en Drupal

Las revisiones es una funcionalidad estándar que se instala por defecto al instalar Drupal. Y cuya función es crear una copia previa del nodo de Drupal, antes de sufrir una modificación, para así mantener un historial de las modificaciones de los nodos.

Hay que destacar que tal y como está diseñada la sincronización entre la BD de Drupal las revisiones NO TIENEN SENTIDO ya que si por ejemplo actualizamos un artista o álbum... no se crea dicha ninguna copia sino que SÓLO se actualiza el nodo el cuestión y nada más.

Esto es debido a que el contenido de RW se actualiza con mucha frecuencia y se ha estimado que se crearían muchas versiones de cualquier nodo , esto probablemente sería un problema ya que el servidor de hosting tiene espacio limitado y además sería de gran dificultad para el administrador gestionar tantas revisiones.

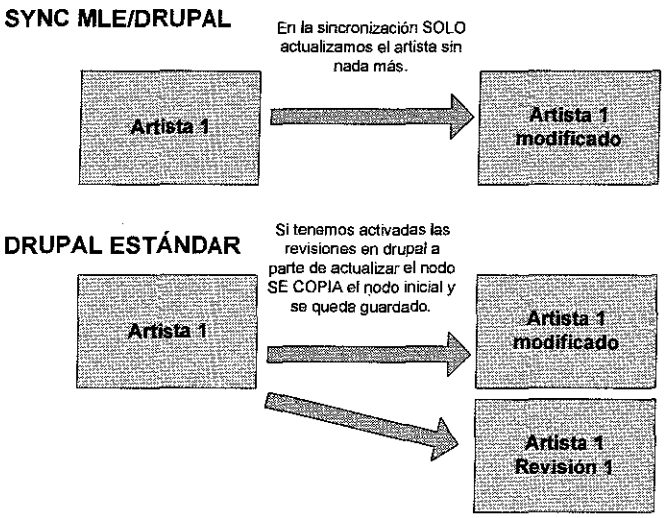


Figura 8.3: Comparación entre el funcionamiento estándar de revisiones con el usado en este PFC

¿Cómo Drupal gestiona los nodos de tipo imagen?

El tema de la creación y actualización de imágenes es un tema clave en la sincronización, se debe tener en cuenta que las imágenes son un tipo de nodo más en Drupal y se deben tratar como tal, a continuación se muestra un esquema del criterio que se sigue respecto al comportamiento de las imágenes:

En la sincronización si *SE CREA* un artista o álbum con una fotografía:

1. Debemos buscar si la imagen está subida (si existe el nodo imagen en Drupal mediante el nombre físico de la imagen).
2. Si la imagen existe en Drupal:
    a. Se debe cambiar el nombre de la imagen que MLE nos proporciona añadiendo la numeración _nn que corresponda (por ejemplo si nos pasa el nombre imagen.jpg se cambiará a imagen_01.jpg)
    b. Se crea el nodo tipo "Image" para el nuevo nombre de imagen (por ejemplo imagen_01.jpg). La imagen físicamente aun no existe en el file system hasta que la suba MLE por ftp
    c. Se asigna la imagen_01.jpg al nodo (artista o álbum) nuevo que se ha creado
    d. Se devuelve al servicio el nombre "nuevo": imagen_01.jpg
3. Si la imagen existe en Drupal:
    a. Se crea nuevo nodo tipo "Image" (sin cambiar el nombre que MLE proporciona)para imagen.jpg
    b. Se asigna la imagen.jpg al nodo nuevo que se ha creado.
    c. Se devuelve al servicio el nombre: imagen.jpg (En este caso coincide con el que nos pasan por parámetro al sproc)

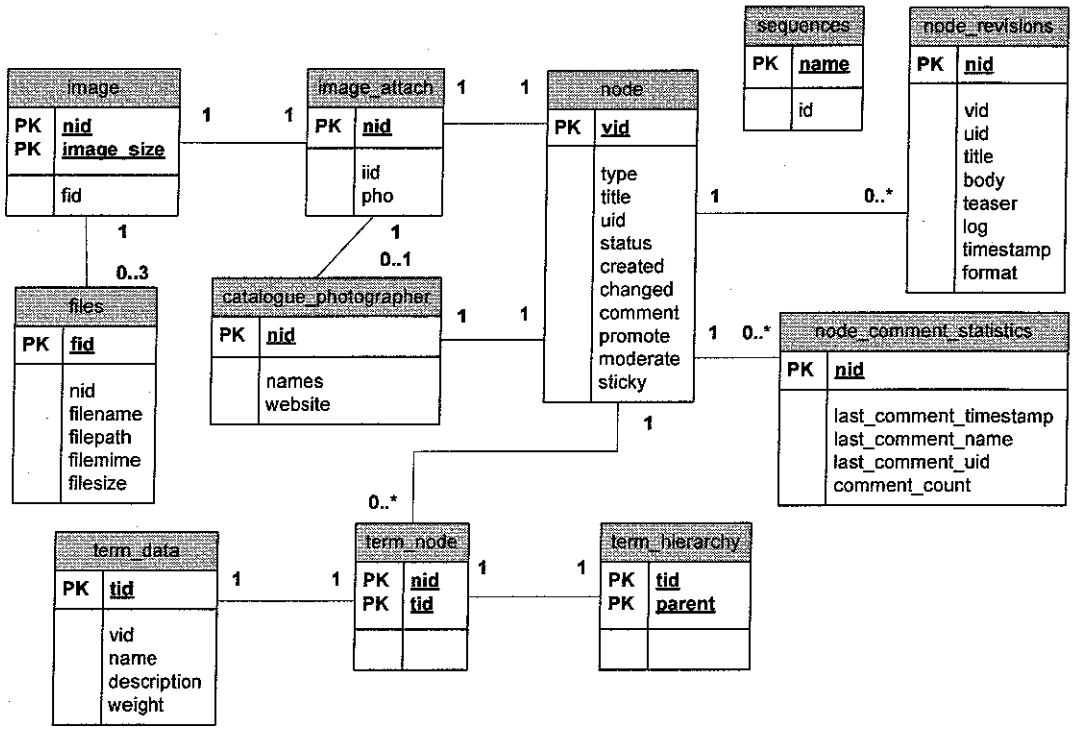
En la sincronización si *SE ACTUALIZA* un artista o álbum con una fotografía:

1. Debemos buscar si la imagen está subida (si existe el nodo imagen en Drupal).
2. Si la imagen existe en Drupal:
    a. Se debe cambiar el nombre de la imagen que MLE nos proporciona añadiendo la numeración _nn que corresponda (por ejemplo si nos pasa el nombre imagen.jpg se cambiará a imagen_01.jpg)

b. Se crea el nodo tipo "Image" para el nuevo nombre de imagen (por ejemplo imagen_01.jpg). La imagen físicamente aun no existe en el file system hasta que la suba MLE por ftp
c. Se asigna el nuevo nodo imagen (imagen_01.jpg) al nodo (álbum o artista) que se está actualizando **(Desasignando PERO NO BORRANDO la imagen anterior que seguirá existiendo en DRUPAL como un nodo más).**
d. Se devuelve al servicio el nombre "nuevo": imagen_01.jpg

3. Si la imagen no existe en Drupal:
   a. Se crea nuevo nodo tipo "Image" (sin cambiar el nombre que MLE proporciona)para imagen.jpg
   b. Se asigna el nuevo nodo imagen (imagen.jpg) al nodo (álbum o artista) que se está actualizando **(Desasignando PERO NO BORRANDO la imagen anterior que seguirá existiendo en DRUPAL como un nodo más).**
   c. Devolvemos al servicio el nombre: imagen.jpg (En este caso coincide con el que nos pasan por parámetro al sproc)

A continuación se muestra la parte de la BD que se ven implicadas a la hora de tratar una imagen en la sincronización.



Las imágenes al tratarse de un nodo más están relacionadas con la tabla nodo y sus agregadas (taxonomías y revisiones).Por otro lado la relación entre las imágenes y los nodos se implementa en la tabla image_attach y como cada imagen es un conjunto de archivos (se debe recordar que una imagen tiene 3 versiones: tamaño original, tamaño thumb y tamaño preview) está relacionada con la tabla files.

¿Cómo Drupal gestiona las taxonomías?

Ideas básicas sobre taxonomías

La capacidad del categorizar los contenidos del sitio es uno de los puntos más fuertes de Drupal. Cada sistema de categorización es un "vocabulario" y Cada "vocabulario" contiene una lista de términos.

Cada "vocabulario" se puede asignar a diferentes tipos de contenido. Por ejemplo, un "vocabulario" para los foros, otro para las imágenes, otro para las páginas, etc. De esta manera, al editar cada contenido, aparecerá la lista de términos para seleccionar el que hay que asociar al contenido.

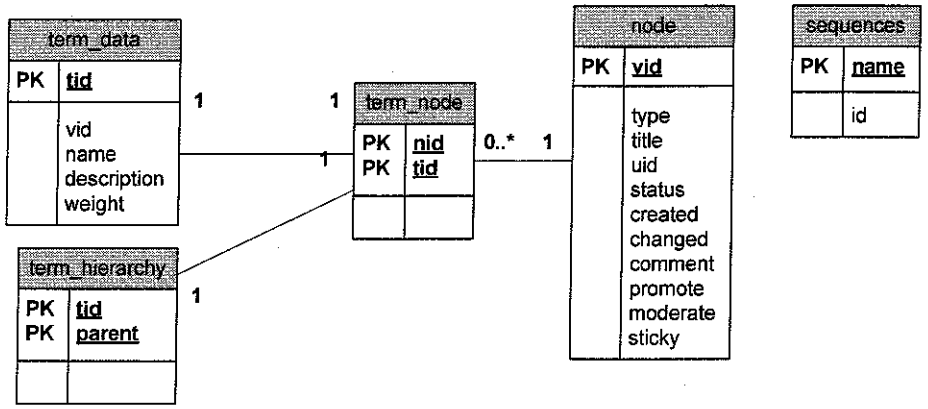
Cada "vocabulario" puede organizar sus términos de diferente manera:

- Sin jerarquía: una simple lista de términos (noticas, opiniones, trucos).
- Jerarquías simples: por ejemplo un vocabulario "recetas" podría contener "arroces", "pastas", "carnes", "aves", etc. A su vez el término "carnes" podría contener "cordero", "ternera", etc.
- Jerarquía múltiple: Un término puede tener diferentes "padres" en la jerarquía. Por ejemplo, "arroz con pollo" podría estar tanto en "aves" como en "arroces"
- Requerido: hacer obligatorio escoger al menos un término del vocabulario.
- Selección múltiple: se puede escoger más de un término para el nodo.
- Etiquetado libre: los términos se crean al vuelo, escribiéndose directamente en la edición del nodo.

Taxonomías y sincronización

Tal y como se hace en Drupal, la sincronización debe mantener las taxonomías. En concreto se debe tener en cuenta en los siguientes casos:

- Creación de Artista: Se crea un término asociado a este artista con el mismo nombre que el artista.
- Actualización del Artista: Sólo si se modifica el nombre del artista se actualizará también su término relacionado, con el nuevo nombre de artista. Hay que destacar que se modificará el término que ya existía, no se crea uno nuevo y se elimina el anterior.
- Creación de un Álbum: Se crea un término asociado a este álbum con el mismo título que el álbum.
- Actualización del Álbum: Sólo si cambia el título del álbum se actualizará también su término relacionado, con el nuevo título de álbum. Hay que destacar que se modificará el término que ya existía, no se crea uno nuevo y se elimina el anterior.



La tabla donde están implementados los términos es la tabla term_data ésta consta de un identificador y un nombre de término, por tanto como se ha comentado antes modificando el nombre de un término no habrá que modificar sus relaciones con otros nodos.

La tabla que implementa la relación entre términos y nodos es la tabla term_node.

La tabla term_hierarchy implementa la jerarquía padres-hijos entre nodos que en nuestro caso no será utilizada ya que las relaciones no serán padre-hijo.

En el diagrama también aparece la tabla de secuencias, esto es debido a que a la hora de crear un término se debe actualizar la tabla de sequences.

La entidad "Tema"

Antes se ha comentado la importancia que tienen los temas en la sincronización, ya que el proceso de sincronización se basa en esta entidad para ejecutar la sincronización. Para una mayor compresión vamos a comentar como se han implementado los temas en Drupal y qué criterios se han planteado para mantener la sincronía entre los dos sistemas.

La tabla catalogue_theme

La tabla ***catalogue_theme*** es la tabla de Drupal donde se implementa la relación entre los mp3 que gestiona MLE y la BBDD Drupal. Esta tabla no es estándar y se crea cuando se instala el módulo *catálogo* que se ha implementado en este proyecto y consta de los siguientes campos:

Campo	Comentario
IdTema	identificador interno de Drupal, éste debe ser trasparente a MLE
IdAlbum	Contiene el id del álbum de Drupal, así es como se relaciona un tema con su álbum.
Titulo	Se debe mantener este campo por si los MP3 se reripean y pierde la información de sus tags ID3, podamos recuperar esta información.
Path	Es la ruta física del tema, se usa como identificador para el MLE, para Drupal debe ser transparente.
Track	Necesitamos mantener este campo por si los MP3 se reripean y pierde la información de sus tags ID3, podamos recuperar esta

	información.
Fecha	Es el timestamp del tema, ***la sincronización, a muy grandes rasgos, consiste en comparar este campo con el timestamp de los archivos MP3, detectar los posibles cambios y actuar al respecto para mantener la sincronía.***

Tabla 8.1: Tabla catalogue_theme

Criterios en la sincronización respecto a los temas

Inicialmente se planteó como criterio general que todos los campos de los temas estuvieran representados como una única entidad en la BBDD tuvieran una relación directa entre los campos de los ID3 que MLE modificaba. Es decir, que todos los tags ID3 que se modificaran en los MP3 estuvieran también replicados y sincronizados en la tabla de temas en Drupal.

Pero al final la decisión final fue mantener solo lo necesario en la tabla de temas en Drupal y no replicar campos que estuvieran en otras tablas (p. ej.: el autor de un tema será el autor del álbum donde pertenecen esos temas). *Esto es debido a que así evitamos contenido del catálogo replicado y mantenemos la BBDD lo más normalizada dentro de nuestras posibilidades.*

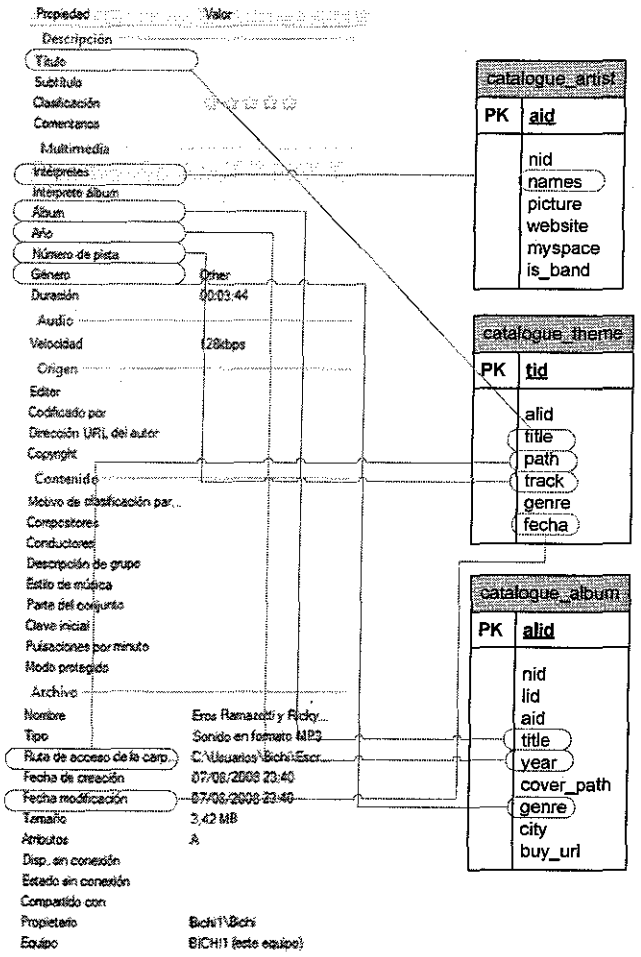


Figura 8.4 : Relación entre campos de la BD y los tags ID3 de los MP3

A continuación, se muestran los casos en que se actualizan los timestamp de los temas en la BD, para que MLE pueda detectar los cambios:

- Al actualizar el nombre de un artista (independientemente de si es músico o banda) se deben actualizan TODOS los temas de TODOS los álbumes en los que ese artista sea autor.
- Al actualizar el título del álbum, el autor, discográfica, año publicación y género se actualizan TODOS los temas de ESE álbum.
- Al actualizar un tema se actualiza el campo ID3 *timestamp* de cada MP3.

Los servicios

Como se observa en la Figura 8.2. La capa de servicios tiene dos partes una que implementa el servicio y la comunicación (servicio propiamente dicho) y otra que implementa el acceso a los datos o la capa de datos o la capa que permite al servicio abstraerse de las particularidades de implementación de la BD por Drupal (las sprocs). En este apartado comentaremos que son los servicios y cuáles han sido necesarios crear para realizar la comunicación entre MLE y Drupal.

A continuación, se exponen los servicios (webservices) que se han creado para implementar el interfaz de conexión entre MLE y la BD de RW gestionada por Drupal. Dichos servicios son ofrecidos desde un servidor cuya dirección se debe introducir en las opciones de configuración de MLE para que éste sepa localizarlos.

Lista de servicios
get_instrumentos
get_artista
get_artista_name
get_artistas
get_album
get_temas
get_fotografos
get_instrumentos_res
get_discograficas
buscar_formacion
save_artista
save_album
save_instrumentos
test_bd

Tabla 8.2: Lista de servicios necesarios para que MLE pueda realizar la sincronización

Descripción detallada de los servicios

get_instrumentos	
Entrada	No hay
Salida	Entero id

	Texto nombre
	Texto abreviatura
Descripción	Este servicio se encarga de cargar los instrumentos existentes en Drupal.

get_instrumentos_res	
Entrada	No hay
Salida	Lista de instrumentos
Descripción	Se devuelve una lista con todos los instrumentos que hay en la BD de Drupal.

get_artista	
Entrada	Entero id
Salida	Entero id Texto nombre Texto foto_url Texto foto_autor Texto web Texto myspace Texto url1 Texto url2 Booleano es_banda
Descripción	Se cargan los datos principales de un artista (sea músico o banda), la entrada id se refiere al identificador del artista.

get_artista_name	
Entrada	Texto nombre_artista
Salida	Entero id_artista Texto nombre_artista Texto foto_url_artista Texto web Texto myspace Texto url1 Texto url2 Booleano es_banda
Descripción	Se carga el artista con el nombre que se pasa en la entrada.

get_artistas	
Entrada	No hay
Salida	Lista de artistas
Descripción	Se cargan todos los artistas de uno en uno, devolviendo los datos

	necesarios de la salida.

get_album	
Entrada	String ruta
Salida	Texto nombre_artista_autor Texto titulo_album Texto anyo Texto cover_url Texto genero Texto tienda_url Texto descripcion Entero id_artista_autor Entero id_Discografica
Descripción	Dado una ruta del álbum, se cargan todos los datos de dicho álbum.

get_temas	
Entrada	No hay
Salida	Texto titulo_tema Texto artista_autor_tema Texto genero Texto nombre_album Texto ruta_tema_HDD Entero date Entero id_Album
Descripción	Se devuelve de uno en uno todos los temas que existen en la BD Drupal.

get_fotografos	
Entrada	No hay
Salida	Lista de fotógrafos
Descripción	Se devuelve una lista con todos los fotógrafos que hay en la BD de Drupal.

get_discograficas	
Entrada	No hay
Salida	Lista de discográficas
Descripción	Se devuelve una lista con todas las discográficas que hay en la BD de Drupal.

buscar_formacion	
Entrada	Texto id_artista

	Texto id_album
Salida	Entero id Entero id_musico Entero id_instrumento Entero id_album Texto interviene Booleano bl
Descripción	Teniendo en cuenta el id_artista y comprobando si es una banda o no, se busca la formación de ese artista, en el caso de que no sea artista se busca un álbum suyo y devolvemos la formación según la especificación de la salida.

save_artista

Entrada	Entero id Texto nombre Texto foto_url Texto foto_autor Texto web Texto myspace Texto url1 Texto url2 Booleano es_banda
Salida	Lista de timestamps de los temas actualizados por un cambio en el nombre del artista y el nombre de la foto si se ha introducido al guardar.
Descripción	Según el id de la entrada, se crea el artista o se actualiza. Es de suma importancia actualizar el timestamp de los temas al modificar el nombre del artista.

save_album

Entrada	Texto nombre_artista_autor Texto titulo Texto anyo Texto cover_url Texto genero Texto tienda_url Texto descripcion Entero id_Artista_autor Entero id_Discografica
Salida	Cierto todo a ha ido bien Falso ha habido un error devuelve el nombre de la foto si es necesario
Descripción	Este servicio se encarga de crear un álbum. Hay que tener en cuenta que se debe actualizar el timestamp de los temas al modificar el título del álbum, el género, el nombre del artista o año.

save_instrumentos	
Entrada	Entero id Texto nombre Texto abreviatura
Salida	Cierto todo a ha ido bien Falso ha habido un error
Descripción	Dada una entrada concreta se crea un instrumento en Drupal.

Test_bd	
Entrada	No hay
Salida	Cierto si hay conexión, Falso si ha habido un error
Descripción	Solo se comprueba si existe conexión con la BD de Drupal.

Los sprocs[7]

Las sprocs deben realizar sobre la base de datos las operaciones de creación, modificación o consulta que solicitan los servicios, replicando las operaciones que sobre la misma se hubiesen hecho en caso de que la operación solicitada por el servicio, hubiese sido realizada directamente desde Drupal.

A continuación, se documenta el detalle de cada uno de los sprocs en una serie de tablas donde se incluye el nombre de la función PHP que lo implementa, una descripción de la función que realizan, los parámetros de entrada que se han de proporcionar a la función, los resultados que devuelve y un subdiagrama de clases que muestra las tablas de la base de datos con las que interacciona.

Lista de sprocs
sp_test_connection
sp_get_instrumentos_res
sp_get_instrumentos
sp_get_temas
sp_get_tema
sp_get_temas_album
sp_get_album
sp_get_formacion_album
sp_get_artista
sp_get_artista_name
sp_get_artistas
sp_get_discograficas
sp_get_fotografos
sp_delete_formacion
sp_buscar_formacion

[7] *Se llaman sproc en referencia a los stored procedures o procedimientos almacenados que se pueden crear y ejecutar dentro de la propia base de datos, porque originalmente se deseaban implementar de esa manera. Por restricciones de las versiones de PHP y MySQL instaladas en el hosting de desarrollo, finalmente se implementaron mediante funciones PHP.*

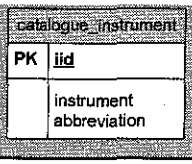
sp_save_formacion
sp_save_instrumentos
sp_save_artista
sp_save_album

Tabla 8.3: Lista de sprocs que necesitan que serán llamados por los servicios

Descripción detallada de los sprocs

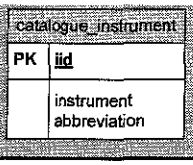
sp_test_connection	
Entrada	No hay
Salida	Cierto o falso.
Operaciones que se realizan en Drupal	Se comprueba que haya acceso a la BD haciendo un consulta simple a la tabla de instrumentos, si hay conexión se retorna cierto si no falso.

Diagrama de clases que interviene



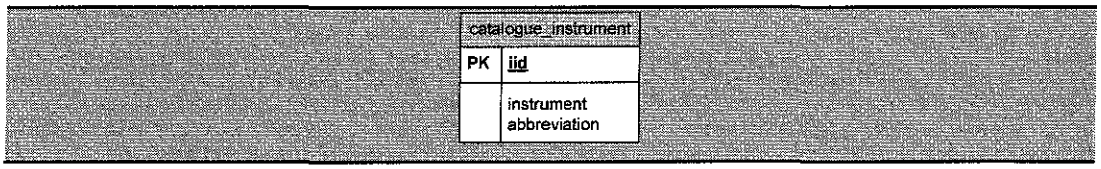
sp_get_instrumentos_res	
Entrada	No hay
Salida	Se devuelve una lista de instrumentos
Operaciones que se realizan en Drupal	Recoger y devolver una lista con los datos de la tabla de instrumentos.

Diagrama de clases que interviene



sp_get_instrumentos	
Entrada	No hay
Salida	Se devuelve una lista de instrumentos
Operaciones que se realizan en Drupal	Recoger y devolver una lista con los datos de la tabla de instrumentos pero en este caso en concreto se devuelven los datos de la consulta directamente (son generar la lista propiamente dicha) ya que por especificaciones del MLE simplifica mucho la programación.

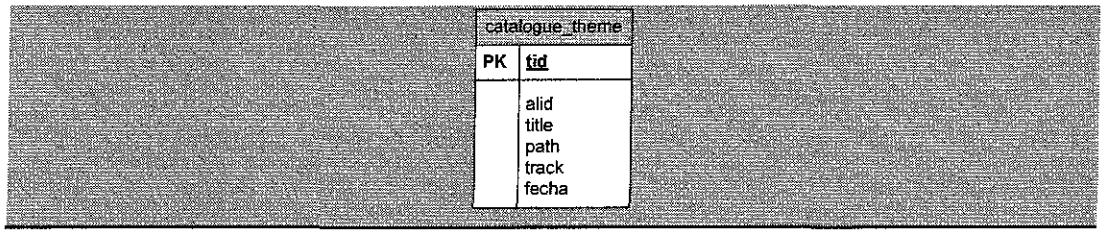
Diagrama de clases que interviene



sp_get_temas

Entrada	No hay
Salida	Lista con las rutas de los temas
Operaciones que se realizan en Drupal	Recoger y devolver una lista con el campo ruta de la tabla de temas.

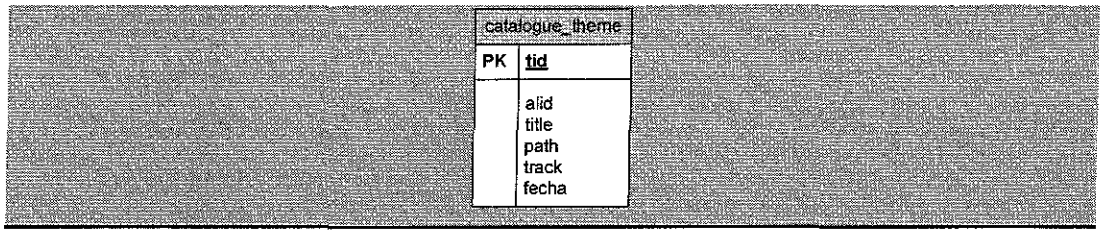
Diagrama de clases que interviene



sp_get_tema

Entrada	Ruta de un tema
Salida	Tema que tiene la ruta de la entrada
Operaciones que se realizan en Drupal	Recoger y devolver el tema con la ruta que nos pasan en la entrada

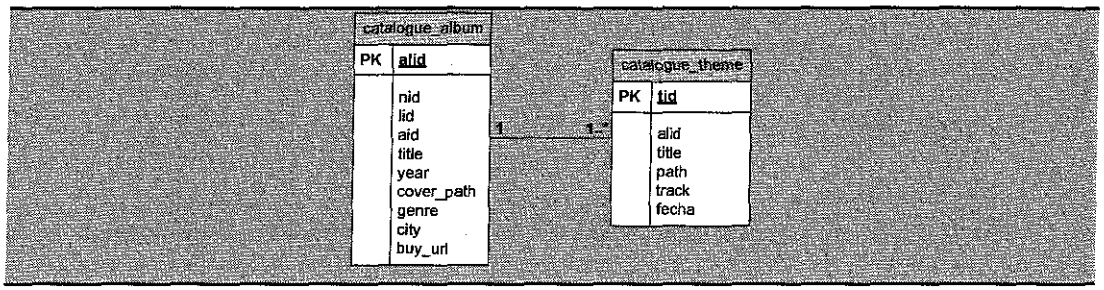
Diagrama de clases que interviene



sp_get_temas_album

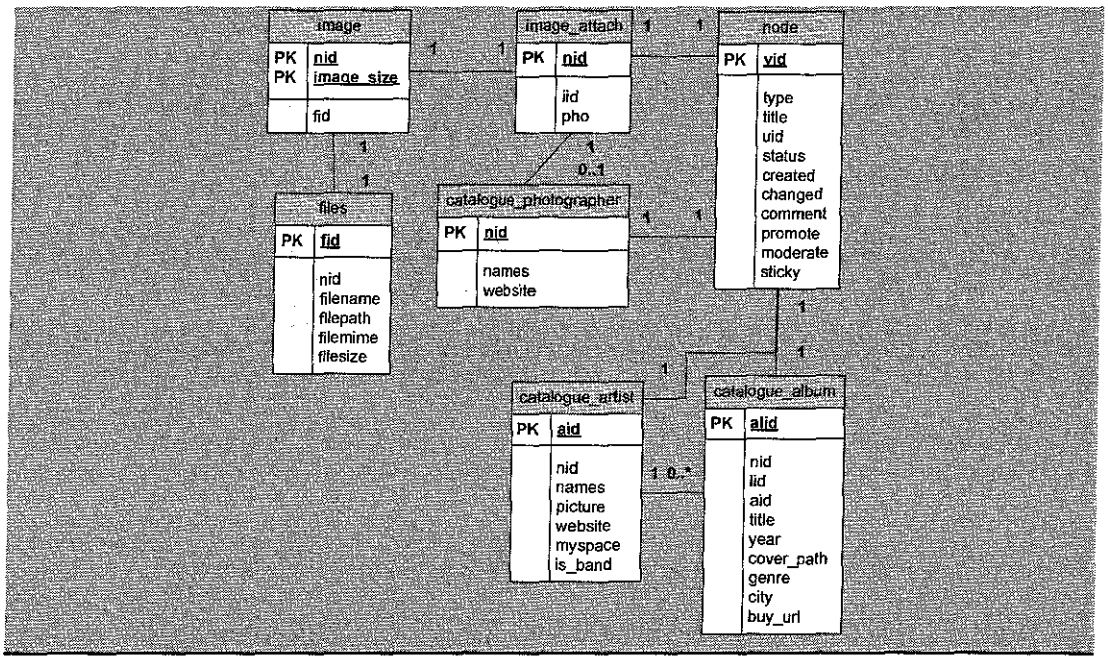
Entrada	Nombre del álbum
Salida	Lista con los temas del álbum que tiene por nombre la entrada.
Operaciones que se realizan en Drupal	Se cargan y devuelven los temas del álbum con el nombre de la entrada. Lo que se hace es buscar la entrada (que es el nombre del álbum) en la ruta y esto funciona ya que los MP3's se guardan en un carpeta con el nombre del álbum.

Diagrama de clases que interviene



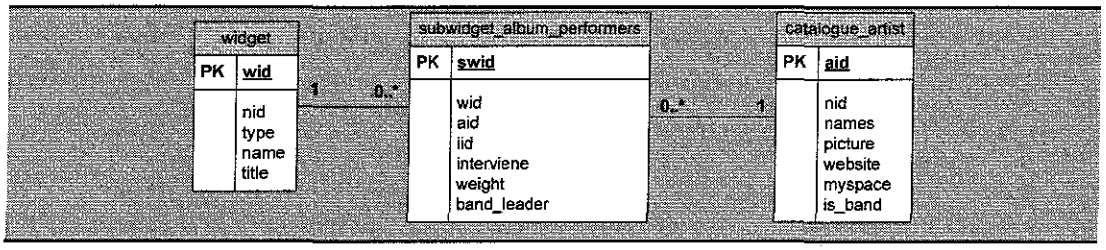
sp_get_album	
Entrada	Identificador del álbum
Salida	Álbum identificado por la entrada.
Operaciones que se realizan en Drupal	Se buscan los datos del álbum, también se debe recuperar ruta de la imagen de la carátula de ese álbum y los datos del autor del álbum.

Diagrama de clases que interviene



sp_get_formacion_album	
Entrada	Identificador del álbum
Salida	Lista con la formación del álbum
Operaciones que se realizan en Drupal	Se buscan los datos de la formación del álbum y para cada componente se buscan los datos del artista asociado y se devuelve.

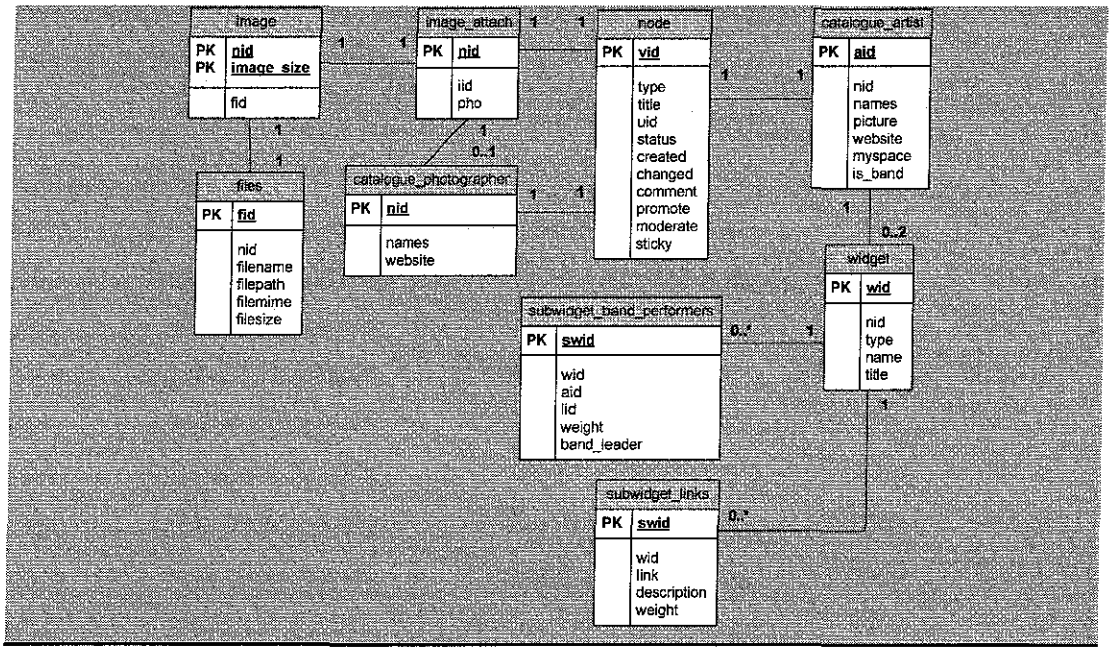
Diagrama de clases que interviene



sp_get_artista

Entrada	Identificador del artista
Salida	Se devuelve el artista con el identificador de entrada y se indica si el artista está en una formación o no.
Operaciones que se realizan en Drupal	Se buscan los datos del artista, también se debe recuperar ruta de la imagen de la fotografía del artista (si tiene), los links adicionales que no están en los datos básicos del artista y por último se debe buscar si ese artista pertenece a una formación o no.

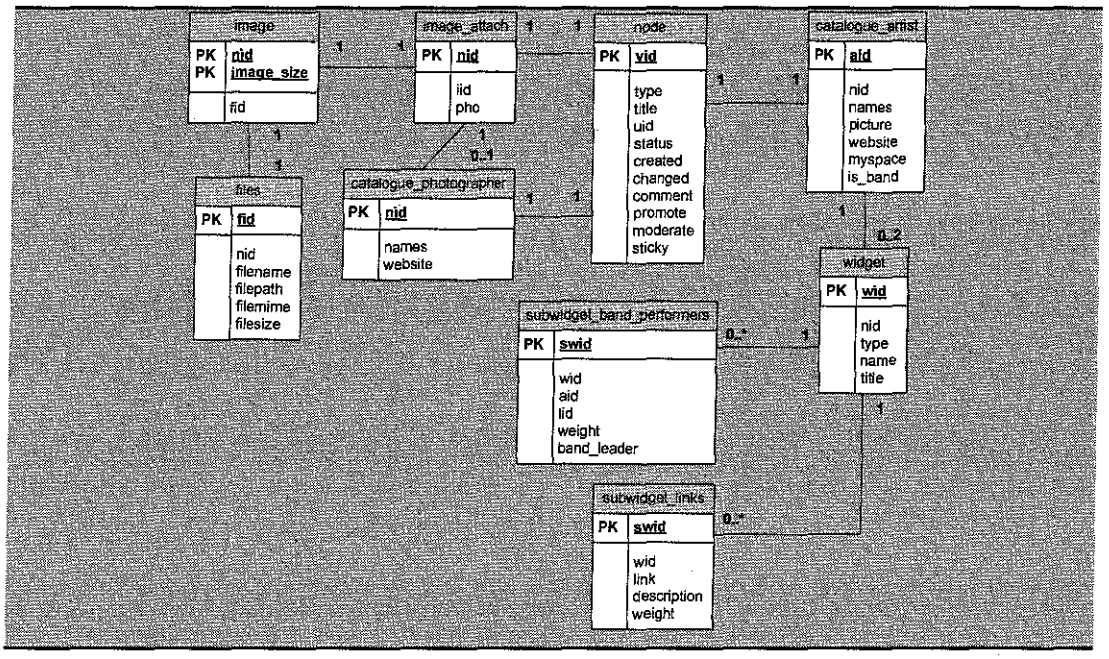
Diagrama de clases que interviene



sp_get_artista_name

Entrada	Nombre del artista
Salida	Se devuelve el artista con el nombre de entrada y se indica si el artista está en una formación o no.
Operaciones que se realizan en Drupal	Se buscan los datos del artista, también se debe recuperar ruta de la imagen de la fotografía del artista (si tiene), los links adicionales que no están en los datos básicos del artista y por último se debe buscar si ese artista pertenece a una formación o no.

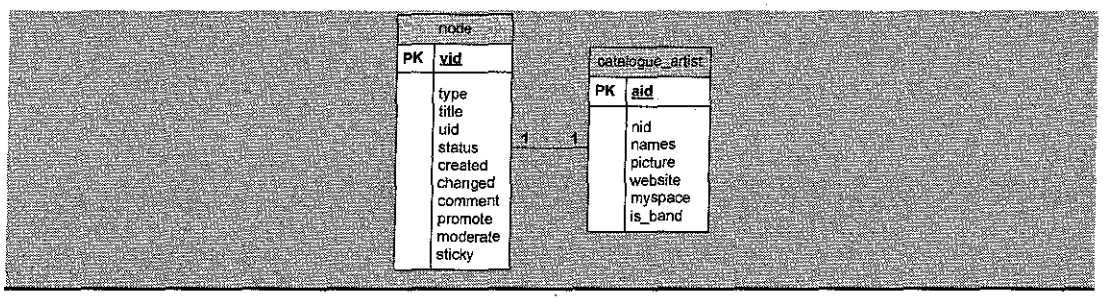
Diagrama de clases que interviene



sp_get_artistas

Entrada	No hay
Salida	Se devuelven el identificador, nombre y si es banda o no de TODOS los artistas existentes en Drupal.
Operaciones que se realizan en Drupal	Se buscan los datos de todos los artistas y se devuelven según la especificación de la salida.

Diagrama de clases que interviene



sp_get_discograficas

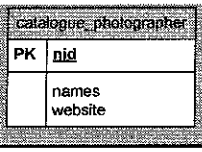
Entrada	No hay
Salida	Lista de instrumentos
Operaciones que se realizan en Drupal	Se buscan los datos de todos los instrumentos y se devuelven en una lista.

Diagrama de clases que interviene

sp_get_fotografos

Entrada	No hay
Salida	Lista de fotógrafos
Operaciones que se realizan en Drupal	Se buscan los datos de todos los fotógrafos y se devuelven en una lista.

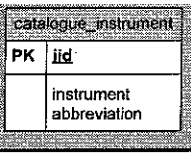
Diagrama de clases que interviene



sp_save_instrumentos

Entrada	Datos de un instrumento, incluyendo el identificador de instrumento.
Salida	Id del instrumento
Operaciones que se realizan en Drupal	Se crea un instrumento con los datos que se facilitan en la entrada en el caso de el id de instrumento de la entrada es -1 y si no es -1 se busca el instrumento y se modifica con los datos de entrada.

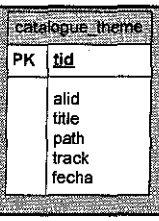
Diagrama de clases que interviene



sp_save_tema

Entrada	Datos de un tema, incluyendo el identificador de instrumento.
Salida	El timestamp del tema
Operaciones que se realizan en Drupal	Se crea un tema con los datos que se facilitan en la entrada en el caso de el id del tema de la entrada es -1 y si no es -1 se busca el tema y se modifica con los datos de entrada.

Diagrama de clases que interviene



sp_save_artista	
Entrada	Datos de un artista, incluyendo su identificador.
Salida	El nombre de la foto con la que MLE debe subir la fotografía del artista, ésta puede ser nula si el artista no tiene foto. Por otra parte si se modifica el nombre del artista (solo en caso de actualización) se debe devolver una lista con los temas de los álbumes de ese artista.
Operaciones que se realizan en Drupal	- Caso crear artista (identificador de artista -1) Según los datos de la entrada se deben mirar si el artista debe aparecer como publicado en Drupal. En el caso de que se cumpla la siguiente condición: Si el nombre del artista, su foto, el fotógrafo de la foto y la web o myspace vengan informados el artista aparecerá como publicado en barcelonajazzradio. Posterior a esto se crea un nodo de tipo artista, si el artista tiene links auxiliares se deben crear los links asociados a ese artista siguiendo la especificación del módulo widget. De mismo modo si el artista tiene formación (porque sea una banda) se debe proceder de la misma manera. El siguiente paso es crear el término taxonómico asociado al artista. Si el artista tiene foto, se debe crear un nodo de tipo imagen y asociarlo mediante los mecanismos que proporciona el modulo image_attach para relacionar la foto al artista y además se tiene que crear una relación entre el fotógrafo y la fotografía. - Caso actualizar artista (identificador de artista distinto de -1) Primeros se debe encontrar el nodo el identificador de la entrada. Posteriormente a esto se debe actualizar el nodo de tipo artista, si el artista tiene links auxiliares se deben actualizar los links asociados a ese artista siguiendo la especificación del módulo widget. De mismo modo si el artista tiene formación

	(porque sea una banda) se debe proceder de la misma manera. En el caso de que varíe el nombre del artista se debe actualizar el término taxonómico asociado a ese artista y devolver una lista con todos los temas donde ese artista es autor. Si el artista tiene foto, se debe crear un nodo de tipo imagen y asociarlo mediante los mecanismos que proporciona el modulo image_attach para relacionar la foto al artista y además se tiene que crear una relación entre el fotógrafo y la fotografía. La fotografía que había antes de la actualización no se elimina pero si se "desasocia" al nodo artista en cuestión.

Diagrama de clases que interviene

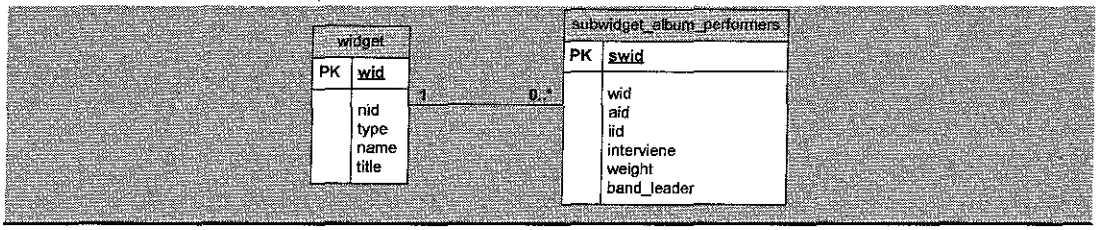
El diagrama que interviene es idéntico al que el capítulo 8.8.4

sp_save_album	
Entrada	Datos de un álbum, incluyendo su identificador.
Salida	Identificador del álbum y el nombre de la foto con la que MLE debe subir la fotografía del álbum, ésta puede ser nula si el álbum no tiene foto.
Operaciones que se realizan en Drupal	- Caso crear álbum (identificador de álbum -1) Se debe crea un nodo de tipo álbum, si el álbum tiene links auxiliares se deben crear los links asociados a ese álbum siguiendo la especificación del módulo Widget. De mismo modo si el álbum tiene formación se debe proceder de la misma manera. El siguiente paso es crear el término taxonómico asociado al álbum. Si el álbum tiene foto, se debe crear un nodo de tipo imagen y asociarlo mediante los mecanismos que proporciona el modulo image_attach para relacionar la foto al álbum y además se tiene que crear una relación entre el fotógrafo y la fotografía. - Caso actualizar álbum (identificador de álbum distinto de -1) Primero se debe encontrar el nodo el identificador de la entrada. Posteriormente a esto se debe actualizar el nodo de tipo álbum, si el álbum tiene links auxiliares se deben actualizar

	los links asociados a ese álbum siguiendo la especificación del módulo widget. De mismo modo si el álbum tiene formación se debe proceder de la misma manera. En el caso de que varíe el nombre del álbum se debe actualizar el término taxonómico asociado a ese álbum. Si el álbum tiene foto, se debe crear un nodo de tipo Imagen y asociarlo mediante los mecanismos que proporciona el modulo image_attach para relacionar la foto al álbum y además se tiene que crear una relación entre el fotógrafo y la fotografía. La fotografía que había antes de la actualización no se elimina pero sí se "desasocia" al nodo álbum en cuestión.
Diagrama de clases que interviene	
El diagrama que interviene es idéntico al que el capítulo 8.8.4.	

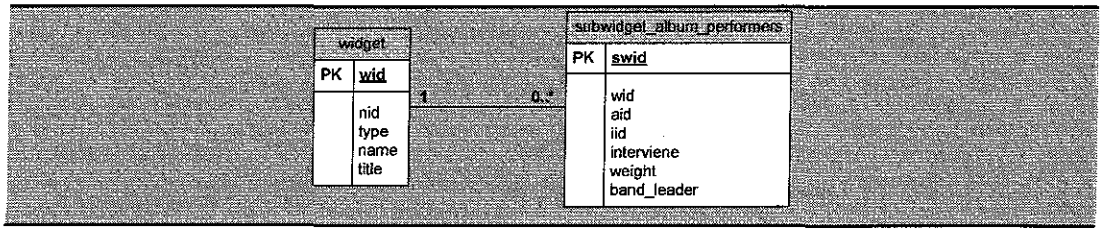
sp_delete_formacion

Entrada	Identificador del álbum.
Salida	No hay
Operaciones que se realizan en Drupal	Se buscan a los componentes de la formación de un álbum y se eliminan
Diagrama de clases que interviene	



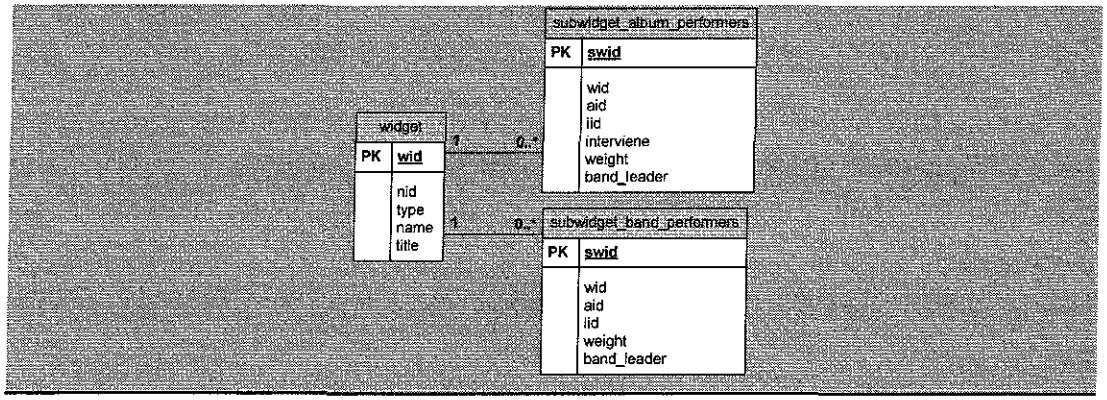
sp_save_formacion

Entrada	Datos de entrada de un componente de una formación, incluyendo el identificador del álbum
Salida	No hay
Operaciones que se realizan en Drupal	Buscamos el álbum y si existe, se añade el componente a la formación de ese álbum.
Diagrama de clases que interviene	



sp_buscar_formacion	
Entrada	Identificador de artista o de álbum
Salida	Lista de componentes de la formación
Operaciones que se realizan en Drupal	Buscamos la formación del artista y si no existe se busca formación en alguno de sus álbumes.

Diagrama de clases que interviene



Relación entre servicios y sprocs creados

A continuación se encuentra una lista en la que se puede observar que sproc son llamados en cada servicio. Cabe comentar que la mayoría de veces un servicio llama sólo a un sproc, pero hay excepciones, generalmente los sproc que más funcionalidad abarcan llaman a más de un sproc.

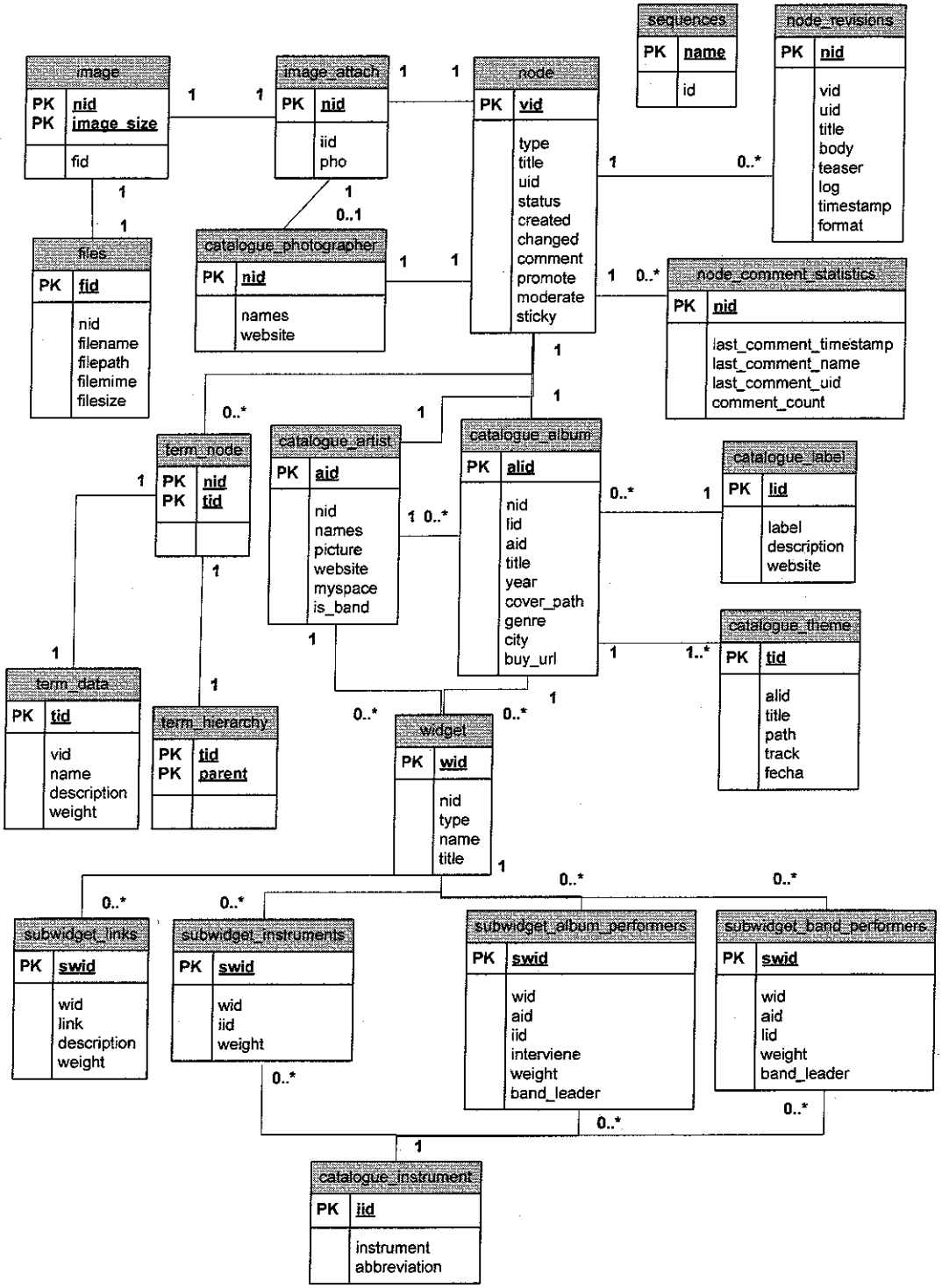
Servicio	Sprocs
Test_db	sp_test_connection
get_instrumentos	sp_get_instrumentos
get_artista	sp_get_artista
get_artista_name	sp_get_artista_name
get_artistas	sp_get_artistas
get_fotografos	sp_get_fotografos
get_instrumentos_res	sp_get_instrumentos_res
get_discograficas	sp_get_discograficas
save_artista	sp_save_artista
save_instrumentos	sp_save_instrumentos
get_album	sp_get_temas_album sp_get_album

	sp_get_formacion_album sp_get_artista
buscar_formacion	sp_buscar_formacion
save_album	sp_save_album sp_save_tema sp_delete_formacion sp_save_formacion
get_temas	sp_get_temas

Tabla 8.4: Relación entre servicios y sprocs

8.8.4. Diagrama de los elementos implicados en la sincronización

En el siguiente diagrama de la BD se muestran las entidades que se ven implicadas en la sincronización. Hay que destacar que este diagrama es un subconjunto de toda la BD que gestiona Drupal.



8.8.5. Consideraciones de implementación

En la solución implementada se deben tener en consideración varios aspectos de implementación que se han tomado:

- Como norma general, no se utilizan valores booleanos sino enteros con valores 1 que representa el valor cierto y 0 que representa el valor falso.
- Cuando se deba modificar un timestamp, éste se actualiza con el valor de la fecha del sistema al inicio de cada sproc, no justo antes de cada llamada a la BD.
- La forma en que Drupal trata el tipo de dato timestamp es peculiar, en concreto, lo que utiliza son enteros y no el tipo de dato TIMESTAMP estándar que se utilizaría en MySQL, por tanto para grabar este dato en los tags id3 de los MP3's se debe hacer una conversión de entero a Timestamp.
  En los sprocs se ha decidido implementarlo de la misma manera que lo hace Drupal con el objetivo de mantener el mismo criterio en ambos sistemas.

8.8.6. Posibles mejoras

La solución aquí propuesta para la sincronización funciona correctamente, pero se propuso en una fase poco madura del proyecto, a medida que nuestros conocimientos se han ido completando se ha visto otras posibles maneras que podrían ser una solución que funcione igual pero que más sencilla de implementar y más estandarizada con el funcionamiento Drupal, en concreto proponemos que se investigue:

- Utilizar el API de DRUPAL para realizar la programación de los servicios. En concreto la funcionalidad DRUPAL_EXECUTE, esta funcionalidad permite ejecutar un formulario mediante programación pasándole todos los parámetros necesarios. Se adjuntan varios links informativos sobre el tema:
  http://api.drupal.org/api/function/drupal_execute/5
  http://drupal.org/node/115522
  http://www.lullabot.com/articles/quick_and_dirty_cck_imports

- En la actualidad MLE sube las imágenes hacia el hosting por el protocolo FTP, la idea es integrar éstas con las subidas de todos los otros datos, es decir subir las imágenes utilizando la tecnología SOAP, la API NUSOAP de Java soporta esta funcionalidad. Se adjuntan varios links informativos:
  http://www.forosdelweb.com/f68/confusion-con-nusoap-455204/
  http://www.nociondigital.com/webmasters/php-tutorial-servicios-web-con-php-nusoap-detalle-168.html

- En la implementación de los sprocs se utiliza el timestamp (que es un campo más de la tabla) para indicar a un registro de una tabla que ha sido modificado. Este valor toma al iniciar el sproc que está en curso, sería ideal que se cogiera justo antes de hacer la consulta a la BD.

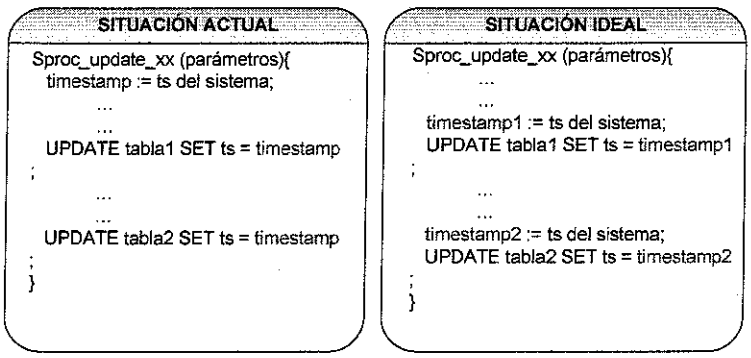


Figura 8.5: Diferentes formas de utilización del Timestamp

- Actualmente la solo el administrador puede modificar datos en la BD, por tanto no es necesario crear un control de concurrencia para que no se puedan modificar datos que se han modificado por otro usuario después de que fueron leídos.
  Pero podría darse el caso de que en un futuro cercano el sistema sea mantenido por más de un Administrador, pudiéndose crear conflictos en el caso de que más de una persona accede a modificar los mismos datos y a la vez. Por esta razón es muy interesante implementar el control de concurrencia.

8.9. Implementación de los requerimientos.

A continuación se explica a modo de resumen, como se ha planteado la implementación de cada requerimiento según lo explicado a lo largo de la sección 8.

CON-001: Insertar y editar contenido (entradas de blog)

Para la implementación de este requisito se utilizó el módulo Story de Drupal, ya que se adaptaba exactamente a la funcionalidad requerida. Las modificaciones que se tuvieron que implementar fueron ocultar varios componentes del formulario de edición de una entrada de blog. Posteriormente se tuvo que implementar un template específico para que se mostraran los datos según especificó el administrador de la radio.

CON-002: Categorización de contenidos

La versión de Drupal instalada en el hosting tiene esta funcionalidad estándar. Por lo tanto este requerimiento queda implementado mediante el propio Drupal.

CON-003: Búsqueda de contenidos

La versión de Drupal instalada en el hosting tiene esta funcionalidad estándar. Por lo tanto este requerimiento queda implementado mediante el propio Drupal.

CON-004: Archivo de contenidos

La versión de Drupal instalada en el hosting tiene esta funcionalidad estándar. Por lo tanto este requerimiento queda implementado mediante el propio Drupal.

USR-001: Creación de cuenta de usuario

Se deben crear los roles y los permisos de usuarios, luego se deben asociar cada permiso a cada rol y luego cada rol a cada usuario, después hay campos que se han tenido que añadir mediante programación.

USR-003: Autentificación (login)

La versión de Drupal instalada en el hosting tiene esta funcionalidad estándar. Por lo tanto este requerimiento queda implementado mediante el propio Drupal.

PRT-003: Administrar listas de distribución para newsletters

Se deben instalar y probar varios módulos externos, después de concluir esa investigación se observó que existía uno que cumplía todas las expectativas tanto en calidad como en funcionalidad y se instaló en la versión de desarrollo. En concreto se instalaron los siguientes módulos:

- **Simplenews:** Permite crear newsletters y enviarlas, archivarlas.
- **Mime Mail:** Permite que al crear la newsletter sea HTML y no texto plano.
- **Simplenews Template:** Permite crear templates para luego usarlos al crear una newsletter.

PRT-004: Creación, envío y mantenimiento de newsletters

Idem caso anterior.

PRT-007: Subscripción RSS a contenidos dinámicos

La versión de Drupal instalada en el hosting tiene esta funcionalidad estándar. Por lo tanto este requerimiento queda implementado mediante el propio Drupal.

CAT-001: Creación y edición de artista

Se ha creado un módulo específico: **catalogue** donde se implementa esta funcionalidad (Sección: 8.4).

CAT-002: Administrar enlaces útiles de un artista

Se ha utilizado el módulo **Widget** (Sección: 8.5) diseñado e implementado especialmente para este proyecto. Hay que dejar claro que el módulo Widget es un submódulo del catalogue, como antes se ha comentado está desarrollado también especialmente para este proyecto.

CAT-003: Subir o enlazar imágenes de un artista

Para esta implementación se han tenido que hacer bastantes modificaciones.

- Primero se ha tenido que instalar el módulo **image** (Sección: 8.2).
- Para poder tener disponibles el submódulo **image_attach** (Sección: 8.2) con el que se puede asociar una imagen a un nodo. Luego se modificó el formulario para que se adaptara al diseño del administrador de la radio.
- También se usó el módulo **Taxonomy** (Sección: 8.2 y 8.8.3) para asociar

mediante taxonomías un fotógrafo a una imagen.

CAT-006: Visualización de artista

Se ha implementado el hook_view del módulo **catalogue** (Sección: 8.4) y se han diseñado templates con la definición html del sitio (Sección: 8.7). Se tenía a disposición un esquema del diseño proporcionado por el administrador del radio website.

CAT-007: Creación y edición de álbum

Se ha creado un módulo específico: **catalogue** donde se implementa esta funcionalidad (Sección: 8.4).

CAT-008: Visualización de álbum

Se ha implementado el hook_view del módulo **catalogue** (Sección: 8.4) y se han diseñado templates con la definición html del sitio (Sección: 8.7). Se tenía a disposición un esquema del diseño proporcionado por el administrador del radio website.

SYNC-002: Comunicación entre servicios y la BD Drupal

En el servidor se ha creado un archivo que implementa las funciones y pasos necesarios para poder crear la conexión entre Drupal y MLE, extensamente explicado en la Sección 8.8.

9. Juegos de prueba

9.1. Juegos de prueba específicos de RW

A continuación, se muestra la relación de juegos de prueba realizados sobre RW.

Me gustaría dejar constancia de que absolutamente todos los errores que se muestran en la tabla siguiente han sido subsanados en una segunda versión.

Categoría	Prueba	Comprobar que	Resultado
General	Insertar caracteres raros en los campos de texto.	Los campos de texto deben procesar correctamente los caracteres extraños, los que no estén definidos (ej.: caracteres cirílicos se deben eliminar).	**La transliteración no se ejecuta correctamente, Drupal elimina por defecto los caracteres que no entiende, en concreto las url's no se crean limpias, por tanto no se cumple la especificación SEO.**
General	Insertar valores extraños (letras en los años, url's incorrectas…).	Cuando se insertan años con letras, el dato no sea aceptado y cuando se inserten url's de diferentes tipos (con IP, .com.ar, etc.) tampoco te lo permita el sistema.	**La fecha aparecer correctamente pero la url no se abre de forma correcta ya que los caracteres raros no se procesan bien.**
General	Insertar valores largos y cortos.	Al insertar una descripción muy larga se inserte correctamente.	**La prueba es satisfactoria. Ya que se visualiza correctamente el resultado.**
General	Insertar valores que ya existían.	El sistema avisa de que el nodo no se ha insertado porque ya existe uno con el mismo nombre.	**El sistema avisa correctamente de nodo duplicado.**
Crear Músico	Añadir descripción (general info), Website, myspace id, enlaces e instrumentos.	Se crea una página de músico. Sobre todo se tienen que añadir los links e instrumentos de forma correcta.	**La prueba es satisfactoria. El artista se crea de forma correcta.**

Crear Músico	Comprobar que se ha creado la página de músico, se muestran los datos insertados y los enlaces funcionan correctamente.	La visualización sea la correcta según la especificación que se nos proporcionó.	**La prueba es satisfactoria ya que los resultados se muestran correctamente, aunque hay algún detalle de theming a arreglar pero damos la prueba por correcta.**
Crear Músico	Comprobar que se lista correctamente en Featured Musicians.	Al ir a la lista de Featured Artist aparece, si está publicado con su respectivo link y si no con el link "deshabilitado".	**La prueba es satisfactoria. Al entrar como administrador parece el link correcto ya que el artista está publicado al modificar lo a no publicado ya no aparece el link.**
Crear Músico	Crear nuevamente el mismo músico y comprobar mensaje de error al entrar un músico ya existente. Probar cambiando mayúsculas y minúsculas.	Se no crea un músico ni una banda con el mismo nombre, el sistema no debe dejar crearlo.	**La prueba es satisfactoria. No se crea el músico nuevo sí que se crea si ponemos el artista con una letra diferente (sea mayúscula o minúscula).**
Editar un músico	Localizar y editar el músico anteriormente creado.	Al ir al músico y modificar la descripción, todo se modifica correctamente.	**La prueba es satisfactoria.**
Editar un músico	Cambiar descripción, cambiar un enlace, borrar un enlace existente, añadir un nuevo enlace.	Eliminar todos los enlaces para posteriormente visualizar, luego crearlos de nuevo, visualizar y si todo es correcto pues modificar un enlace en concreto y visualizar.	**Todo correcto pero al eliminar todos los links de golpe, al actualizar no los elimina.**

Editar un músico	Eliminar un instrumento y añadir otro.	La ficha del artista se actualiza correctamente y ver como se visualiza dicho artista.	**La prueba es satisfactoria. A diferencia de los links Los cambios se guardaron sin problemas.**
Editar un músico	Añadir instrumentos en distinto orden, repetidos, etc. y comprobar que en la ficha la información es correcta.	La ficha del artista se actualiza correctamente y ver como se visualiza dicho artista.	**La prueba es satisfactoria. Al visualizarlo se ven en el orden correcto.**
Eliminar músico	Eliminar el músico y comprobar que se actualiza en Featured Musicians.	Al eliminar un músico, desaparece del Featured Artist.	**La prueba es satisfactoria. Deja de aparecer en el Featured.**
Eliminar músico	Tratar de eliminar un músico con álbumes asignados y comprobar que no es posible y se informa del motivo.	Da un mensaje por pantalla y no deja eliminar el músico, al eliminar todos sus álbumes si que debería dejar eliminarlo.	**El músico se elimina aunque tenga álbumes asociados.**
Eliminar músico	Tratar de eliminar un músico con banda asignada y comprobar que no es posible y se informa del motivo.	Ídem que el caso anterior.	**La prueba es satisfactoria. El músico si que se elimina correctamente.**
Añadir un Story a la página de músico	Crear un Story asignado a una página de músico.	Al visualizar el artista se debe ver esa Story debajo de su ficha.	**La prueba es satisfactoria. En la ficha del artista sus posts asociados se muestran en el orden que se ha especificado.**

Añadir un Story a la página de músico	Modificar el Story (contenido y posición) y comprobar que se actualiza correctamente.	Se añadirá un Story más al músico y con un orden inferior al anterior y al visualizar la ficha del músico se deben ordenar de forma correcta.	**La prueba es satisfactoria. En la ficha del artista sus posts asociados se muestran en el orden que se ha especificado.**
Añadir un Story a la página de músico	Cambiar el músico al que se asigna y comprobar que desaparece de la ficha del primero y aparece en la del segundo.	Al editar el Story anterior (cambiando su taxonomía artista) se asigna correctamente al nuevo músico.	**La prueba es satisfactoria. En la ficha del artista sus posts asociados se muestran en el orden que se ha especificado.**
Añadir un Story a la página de músico	Borrar el Story creado y comprobar que se elimina de la página de músico.	Al eliminar el Story anterior se desasigna del artista con el que estaba relacionado.	**La prueba es satisfactoria. En la ficha del artista sus posts asociados se muestran en el orden que se ha especificado y sólo aparecen los que deben aparecer.**
Crear un álbum nuevo	Crear el álbum, con descripción, imagen y formación.	Al crearlo se deben guardan todos los datos correctamente.	**La prueba es satisfactoria.**
Crear un álbum nuevo	Comprobar que se crea la página de álbum.	El álbum anterior se visualiza correctamente según las especificaciones de diseño del sitio.	**La prueba es satisfactoria. La ficha del álbum se visualiza de forma correcta.**

Crear un álbum nuevo	Comprobar que los enlaces a páginas de artistas de la formación, enlaces externos y enlaces a otros álbumes funcionan correctamente.	Se visualiza un artista con formación, enlaces externos y álbumes asociados, al ir a cualquier artista de la formación (estando publicado) se debe mostrar la vista de ese álbum. Si vamos a uno de sus links nos debe dirigir correctamente a la página de destino. Y por último si vamos a un álbum nos debe dirigir a su ficha.	**Todo funciona correctamente cuando se trata de contenido interno, si vamos a un link externo aparece una página Drupal "Página no encontrada".**
Crear un álbum nuevo	Crear un álbum con el mismo nombre que un músico.	Se debería crear el álbum sin ningún problema ya que ese caso se puede dar.	**La prueba es satisfactoria. El nuevo nodo se crea de forma correcta.**
Crear un álbum nuevo	Comprobar las fichas del músico y del álbum para ver que los posts, álbumes, fotografías, etc. están en su sitio.	La visualización de las fichas debe ser igual a la proporcionada en el documento de especificación de diseño.	**La prueba es satisfactoria.**
Crear un álbum nuevo	Utilizar el Fill it.	Comprobar que se le asigna una formación cumpliendo con el algoritmo especificado en la especificación.	**No se rellena la formación de forma correcta.**
Editar un álbum existente	Localizar el álbum.	Se puede localizar el álbum desde la administración al mismo álbum o desde la ficha de su autor.	**La prueba es satisfactoria. El álbum está donde debe estar y se puede acceder a él según los requerimientos del administrador de la radio.**

Editar un álbum existente	Cambiar descripción, cambiar formación, label e imagen.	Los cambios se hacen efectivos, la imagen se carga correctamente, creando un nuevo nodo imagen y, por último, la formación se actualiza correctamente.	**La prueba es satisfactoria. La imagen se crea como nodo de forma correcta y se relaciona con el álbum.**
Editar un álbum existente	Comprobar que la página de álbum se actualiza.	Al visualizar dicho álbum los cambios sean efectivos.	**La prueba no es satisfactoria. Visualización correcta excepto el modo resumen.**
Editar un álbum existente	Comprobar que en la página de artista se actualiza.	Al modificar el autor de un álbum, y ir a la ficha del nuevo autor ésta contiene el álbum asignado, quedando desasignado de su antiguo autor.	**La prueba es satisfactoria. Al ver la ficha del artista aparecen todos sus álbumes asociados.**
Editar un álbum existente	Utilizar el Fill it.	Comprobar que se le asigna una formación cumpliendo con el algoritmo especificado en la especificación.	**No se rellena la formación de forma correcta.**
Añadir un Story a la página de álbum	Crear un Story asignado a una página de álbum y comprobar que se visualiza correctamente.	Se crea un Story asignándolo a un álbum creado con anterioridad, al visualizar la ficha del álbum debe aparecer la nueva Story.	**La prueba es satisfactoria. Al ver la ficha del álbum aparecen todos sus posts asociados.**

Añadir un Story a la página de álbum	Modificar el Story (contenido y posición) y comprobar que se actualiza correctamente.	Al modificar la Story anterior, los cambios han sido efectivos y que aparece correctamente en la ficha del álbum.	**La prueba es satisfactoria. Al ver la ficha del álbum aparecen todos sus posts asociados, los que se deben ver ni más ni menos.**
Añadir un Story a la página de álbum	Cambiar el álbum al que se asigna y comprobar que desaparece de la ficha del primero y aparece en la del segundo.	Al modificar el álbum del Story y se va a su ficha, éste debe aparecer en su ficha y no aparecer en su álbum asociado anterior.	**La prueba es satisfactoria. Al ver la ficha del álbum aparecen todos sus posts asociados.**
Añadir un Story a la página de álbum	Borrar este Story.	Debe desaparecer de la ficha del álbum asociado.	**La prueba es satisfactoria. Al ver la ficha del álbum aparecen todos sus posts asociados y los eliminados dejan de aparecer.**
Eliminar un álbum	Eliminar un álbum.	No se eliminan sus Stories asociadas, pero sí que se desasigna de su autor (mirando la ficha del autor).	**La prueba es satisfactoria. Los posts asociados se quedan sin relacionar.**
Eliminar un álbum	Comprobar que las páginas de artista se actualizan.	Ídem caso anterior.	**La prueba es satisfactoria. Los posts asociados se quedan sin relacionar.**
Crear una banda nueva	Añadir descripción (general info), Website, myspace id, enlaces y formación.	Se crea la banda correctamente.	**La prueba es satisfactoria. Ídem artistas.**

Crear una banda nueva	Comprobar que se ha creado la página banda, se muestran los datos insertados y los enlaces funcionan correctamente.	Al visualizar la banda aparece en su lugar y su ficha también debe aparecer acorde con el documento de diseño.	**La prueba es satisfactoria. Ídem artistas.**
Crear una banda nueva	Comprobar que se lista correctamente en Featured Musicians.	Al ir a la lista de artistas aparece la banda si está publicada y "deshabilitada" si no lo está, esta funcionalidad es idéntica a la de músico.	**La prueba es satisfactoria. Los posts asociados se quedan sin relacionar. Ídem artistas.**
Crear una banda nueva	Crear nuevamente la misma banda y comprobar mensaje de error al entrar una banda ya existente.	No se crea la banda y el sistema te informa de la causa.	**La prueba es satisfactoria. Ídem artistas.**
Crear una banda nueva	Utilizar el Fill it.	Comprobar que se le asigna una formación cumpliendo con el algoritmo especificado en la especificación.	**No se rellena la formación de forma correcta.**
Editar una banda	Localizar y editar la banda anteriormente creada.	Los cambios quedan reflejados en el sistema.	**La prueba es satisfactoria. Ídem artistas.**
Editar una banda	Cambiar descripción, cambiar un enlace, borrar un enlace existente, añadir un nuevo enlace.	El sistema de añadir links funciona correctamente.	**Los links si se eliminan todos a la vez, el sistema no se entera de los cambios, se ha tenido que eliminar también la caché.**

Editar una banda	Eliminar un músico y añadir otro.	La formación se modifica, eliminando el componente de la banda especificado y añadiendo el nuevo.	**Los links si se eliminan todos a la vez, el sistema no se entera de los cambios, se ha tenido que eliminar también la caché.**
Editar una banda	Comprobar que los cambios se reflejan en la página de banda correspondiente.	Al visualizar los cambios se reflejan en la ficha de la banda.	**Los links si se eliminan todos a la vez, el sistema no se entera de los cambios, se ha tenido que eliminar también la caché. Haciéndolo de uno en uno funciona correctamente.**
Editar una banda	Reordenar músicos.	Al cambiar el orden de aparición de los componentes de la banda, al visualizarla los componentes quedan correctamente ordenados.	**La prueba es satisfactoria. Ídem álbumes.**
Editar una banda	Utilizar el Fill it.	Comprobar que se le asigna una formación cumpliendo con el algoritmo especificado en la especificación.	**No se rellena la formación de forma correcta.**
Añadir un post a la página de banda	Crear un Story asignado a una página de banda (artista).	Se debe crear la relación entre el post (Story) y la banda de forma correcta y coherente con el sistema.	**La prueba es satisfactoria. Ídem artistas.**
Añadir un post a la página de banda	Comprobar que se visualiza correctamente.	Al visualizar la banda deben aparecer todos sus posts asociados.	**La prueba es satisfactoria. Ídem artistas.**

Añadir un post a la página de banda	Modificar el Story (contenido y posición) y comprobar que se actualiza correctamente.	Cuando hay más de un post asociado a una banda, al modificar el orden de aparición, los posts se ordenan de la forma especificada.	**La prueba es satisfactoria. Ídem artistas.**
Añadir un post a la página de banda	Borrar este Story y comprobar que se elimina de la página de banda.	Al eliminar el post, éste debe desaparecer de la ficha de la banda.	**La prueba es satisfactoria.**
Eliminar una banda	Eliminar la banda y comprobar que se actualiza en Featured Musicians.	Si una banda tiene álbumes asociados no se puede eliminar, al eliminar esos álbumes se debe poder eliminar la banda y por tanto no debería aparecer en el Featured Artist.	**La prueba es satisfactoria. Ídem artistas.**
Crear un fotógrafo	Crear un fotógrafo, incluir su descripción.	Se crea la ficha del fotógrafo correspondiente, comprobando que permite insertar código HTML en la descripción, para posteriormente visualizar este código HTML correctamente en su ficha.	**La prueba es satisfactoria. Su ficha se crea correctamente y la descripción se muestra de forma correcta.**
Crear un fotógrafo	Comprobar que es posible asignar imágenes al mismo desde página de artista y de álbum.	Al ir a la ficha de un álbum o artista con fotografía principal y al cambiar el fotógrafo, la imagen aparece en la galería del nuevo fotógrafo y queda eliminada del antiguo.	**La prueba es satisfactoria. Al editar un artista modificando el fotógrafo, esta foto aparece de forma correcta en la galería del fotógrafo.**

Editar un fotógrafo	Modificar la información sobre un fotógrafo.	Los cambios se actualizan correctamente en el sistema.	**La prueba es satisfactoria.**
Editar un fotógrafo	Comprobar que se actualiza en la página de fotógrafo.	Al visualizar la ficha del fotógrafo cumple con los requisitos del documento de diseño.	**La prueba es satisfactoria. Cumple con las especificaciones de diseño y de SEO.**
Eliminar un fotógrafo	Eliminar un fotógrafo sin imágenes asignadas.	Se elimina correctamente del sistema.	**La prueba es satisfactoria. El nodo se elimina correctamente.**
Eliminar un fotógrafo	Comprobar que no se puede eliminar un fotógrafo con imágenes asignadas.	Al asignarle un fotógrafo a una imagen y al intentar eliminarlo, el sistema no lo permite y se muestra un mensaje informando de lo sucedido.	**El nodo se elimina correctamente de todas formas, se debe validar antes.**
Crear una página (full HTML)	Crear una página con tipo de inserción full HTML.	El código HTML insertado en un nodo se visualiza correctamente, y no debería mostrar tags HTML sino solo el resultado.	**La prueba es satisfactoria. Ídem fotógrafos.**
Imágenes	Crear una imagen desde un artista o álbum.	Al crear un álbum o artista se debe crea la imagen asociada, creando así un nuevo nodo imagen correctamente asignado al artista o álbum, esto se puede comprobar visualizando la ficha de un artista.	**La prueba es satisfactoria. La imagen se asigna de forma correcta y se desasigna también de forma correcta.**

Imágenes	Asignar una imagen a un fotógrafo y comprobar que la galería se crea correctamente.	Ya sea desde la edición de un artista o desde la edición de una imagen, al asignarle un fotógrafo la imagen pasa a la galería de dicho fotógrafo.	**La prueba es satisfactoria. La imagen se asigna de forma correcta y se desasigna también de forma correcta incluyendo su fotógrafo.**
Imágenes	Eliminar una fotografía.	Al eliminar una fotografía, se elimina de la ficha del artista y también se elimina de la galería del fotógrafo asignado.	**La prueba es satisfactoria. La imagen se elimina de forma correcta y se desasigna también de forma correcta de su fotógrafo.**
Imágenes	Crear una imagen desde el formulario estándar.	Se crea una imagen sin que quede asignada a ningún otro nodo, sea del tipo que sea.	**La prueba es satisfactoria. El nodo se crea correctamente.**
Imágenes	Editar una imagen.	Al editar una imagen, ésta carga la nueva imagen del HDD seleccionada por el usuario y se visualiza correctamente.	**La prueba es satisfactoria. El nodo se modifica correctamente, al visualizar la imagen se ve la nueva fotografía.**
Imágenes	Cambiarle el nombre a una imagen desde el formulario de un artista.	Al editar un artista se modifica el título de la imagen en su ficha y también en la visualización de la fotografía.	**El nodo imagen no queda modificado, no obstante, si lo modificamos desde el nodo imagen propiamente dicho, funciona correctamente.**

Imágenes	Editar la imagen de un álbum y asignarle fotógrafo.	La fotografía debe quedar relacionada correctamente con el fotógrafo y con el álbum, si visualizáramos sólo la imagen debería aparecer también en la galería del fotógrafo.	**La prueba es satisfactoria. Al visualizar el fotógrafo la imagen aparece en su galería y al visualizar el álbum la foto aparece en su ficha.**

9.2. Juegos de prueba específicos de MLE

En esta sección podemos ver una relación de las pruebas realizadas tanto satisfactoriamente como si no lo han sido sobre MLE y los elementos a comprobar tras realizar cada prueba.

Categoría	Prueba	Comprobar que	Resultado
Comprobar la actualización de la librería	Creación por primera vez del *libreria.txt (primera vez que se abre la librería).*	Se reconocen todos los álbumes en el disco y todos los mp3 comprobar que se crean en el archivo *libreria.txt* sólo los álbumes y sus mp3 que están editados. Que están contenidos en el disco duro y en la BD.	**La prueba es satisfactoria. El fichero se crea correctamente.**
Comprobar la actualización de la librería	Cambiar el path de la librería (opciones) y luego abrir librería Hacer una copia de la librería de origen, añadiendo álbumes y artistas nuevos y eliminando algunos de los que había editados en el original y dejando algunos álbumes que no estén editados en ambas librerías.	Comprobar que se reconocen todos los álbumes y mp3 en el nuevo path. Que los álbumes del antiguo path que ya no están en el nuevo, no aparecen (siguen en la BD, pero MLE no los hace accesibles -navegación-). Si hay álbumes ya editados en el BD, que se marcan como editados. Que los artistas ya editados y de alta en la base de datos siguen accesibles (typeahead).	**La prueba es satisfactoria.**
Comprobar la actualización de la librería	Añadir un álbum nuevo. Navegar hasta el álbum.	Lo identifica como no editado y lo hace accesible. Navegación. Aparece como no editado.	**La prueba es satisfactoria. Destacar que va lento.**

Comprobar la actualización de la librería	Eliminar del HDD un álbum no editado.	No es navegable, No aparece en el combo Ir_a.	**La prueba es satisfactoria. Funcionamiento correcto.**
Comprobar la actualización de la librería	Eliminar del HDD un álbum editado.	No es navegable, No aparece en el combo Ir_a.	**La prueba es satisfactoria. Funcionamiento correcto.**
Comprobar la actualización de la librería	Añadir un mp3 nuevo a un álbum ya editado. Navegar hasta el álbum.	El álbum sale como no editado pero recoge los datos de la BD del resto del álbum. Cuando se accede se identifican los mp3 nuevos y se distinguen de los ya editados. Se guardan los datos correctamente (comprobar volviendo al álbum).	**La prueba es satisfactoria. Funcionamiento correcto.**
Comprobar la actualización de la librería	Añadir un mp3 re-ripeado a un álbum ya editado. Navegar hasta el álbum.	El álbum se navega como editado y se abre como editado pero recoge los datos de la BD de todo el álbum incluido el mp3 re-ripeado. Cuando se accede se identifica el mp3 re-ripeado y se distinguen de los ya editados pero con los datos de la BD. Se guardan los datos correctamente (comprobar volviendo al álbum). Al cambiar de álbum sin guardar pregunta descartar cambios.	**La prueba es satisfactoria. Funcionamiento correcto.**

Comprobar la actualización de la librería	Re-ripear completamente un álbum ya editado. Navegar hasta el álbum.	Identifica los temas re-ripeados y pone los datos previamente editados para los temas pero los temas aparecen como no editados. Es el usuario quien tiene que acordarse de que el álbum hay que ir a editarlo de nuevo aunque en MLE aparece como editado pero sus temas como no editados.	**La prueba es satisfactoria. Funcionamiento correcto.**
Otras opciones de configuración	Provocar un error en los parámetros y luego elegir la opción descartar cambios. Luego volver a mostrar la pantalla de opciones.	La segunda vez que se muestra el formulario, todas las opciones aparecen con los valores originales. Probar todos los campos.	**No funciona en todos los campos.**
Otras opciones de configuración	Provocar un error en los parámetros y luego elegir la opción aceptar cambios. Luego volver a mostrar la pantalla de opciones.	La segunda vez que se muestra el formulario, todas las opciones aparecen con los valores modificados, no con los originales. Probar todos los campos.	**No funciona en todos los campos.**
Otras opciones de configuración	Introducir un directorio para la librería inexistente.	Da el aviso correspondiente y ofrece las tres opciones: aceptar, descartar, cancelar.	**La prueba es satisfactoria.**
Otras opciones de configuración	Introducir un directorio para la librería correcto.	Guarda las opciones, verificar cambios abriendo de nuevo el formulario.	**La prueba es satisfactoria.**

Otras opciones de configuración	Introducir opciones de conexión al servicio erróneas.	Da el aviso correspondiente y ofrece las tres opciones: aceptar, descartar, cancelar.	**La prueba es satisfactoria.**
Otras opciones de configuración	Introducir opciones de conexión al servicio correctas.	Guarda las opciones, verificar cambios abriendo de nuevo el formulario.	**La prueba es satisfactoria.**
Otras opciones de configuración	Introducir opciones de conexión a FTP erróneas*.	Da el aviso correspondiente y ofrece las tres opciones: aceptar, descartar, cancelar.	**La prueba es satisfactoria.**
Otras opciones de configuración	Introducir opciones de conexión a FTP correctas*.	Guarda las opciones, verificar cambios abriendo de nuevo el formulario.	**La prueba es satisfactoria.**
Comprobar el funcionamiento de los botones de navegación	Moverse a través de la librería con los diferentes botones de navegación.	Comprobar que realmente se sitúe en el álbum correspondiente. Comprobar que se recorren todos y en el orden establecido (orden alfabético artista/álbum) navegando todos, editados y no editados.	**No cumple la especificación del administrador de la radio.**
Comprobar el funcionamiento de los botones de navegación	Ir al primer álbum de la lista y clicar en los botones de anterior (los 4 primeros botones, flechas en dirección <) e ir al último y clicar en los botones de siguiente (>).	Comprobar que no nos movemos del primer álbum o del último.	**La prueba es satisfactoria. Vamos al último y al primero.**

Comprobar el funcionamiento de los botones de navegación	Imaginemos que X es un álbum no editado y V es editado. 1) VVVVVVXXXXXX si estamos en la última V y le pedimos el siguiente editado (no existe). - igualmente si pedimos un anterior no editado (no existe todos los anteriores son editados). 2) XXXXXXVVVVVV en esta situación, si pedimos el siguiente no editado y todos son editados. - igualmente si pedimos el anterior editado este no existe.	En ninguno de los cuatros caso se ha de mover del álbum en el que está porque no existe un siguiente álbum del tipo que se pide en la dirección en la que se pide.	**La prueba es satisfactoria.**
Comprobar el funcionamiento de los botones de navegación	1) Elegir varios álbumes en el typeahead y navegar hacia ellos. 2) Probar los tres botones siguientes. 3) Probar los tres botones anteriores.	Comprobar que realmente nos vamos al álbum seleccionado. Comprobar que se desplaza al álbum siguiente del tipo solicitado. Comprobar que se desplaza al álbum siguiente del tipo solicitado.	**El typeahead va muy lento hay que optimizarlo.**

Gestión de imágenes	Subir una imagen de álbum.	Que desde Drupal se ve en la ficha del álbum la nueva imagen. Qué ocurre si hay una imagen anterior (que no se queden perdidas en el hosting) si está previsto en RW que haya otras imágenes del álbum, que la existente se convierta en una de las otras.	**La prueba es satisfactoria.**
Gestión de imágenes	Subir una imagen de un artista.	Que desde Drupal se ve en la ficha del artista la nueva imagen. Qué ocurre si hay una imagen anterior (que no se queden perdidas en el hosting): que la existente se convierta en una de las otras.	**La prueba es satisfactoria.**
Abrir i editar álbum	Abrir MLE.	Se carga correctamente toda la lista de discográficas, artistas, músicos, fotógrafos, álbumes e instrumentos en los typeahead. Comprobar que se cargan todos los datos de Drupal y que se ven correctos (acentos, caracteres raros, etc.)(instrumentos fotógrafos discográficas).	**La prueba es satisfactoria.**

Abrir i editar álbum	Abrimos un álbum no editado.	Se cargan todos los campos que se extraen del mp3: artista, álbum, tema, género, año y número de canción. El álbum no tiene un artista asignado y se muestra en el typeahead del artista -- y en rojo si se hace F2 se ve el artista que hay en los tags id3. Se tiene la posibilidad de elegir otro artista y actualizarlo pulsando F2.	**La prueba es satisfactoria.**
Abrir i editar álbum	Abrir un álbum editado.	Se cargan los datos de la BD, se comprueba si algún tema se ha re-ripeado para mostrarlo como no editado. Los datos cargados son correctos. Si hay algún mp3 re-ripeado el álbum se muestra como no editado.	**La prueba es satisfactoria.**

Abrir i editar álbum	Modificar cualquier información de un álbum y comprobar que cualquier botón que nos mueva de álbum nos pregunte si queremos descartar los cambios.	Al editar cualquier campo y cambiar de álbum sin guardar datos se muestra un mensaje para descartar cambios o cancelar. Comprobar que el ciclo del focus es el correcto tanto al editar los álbumes como al administrar un artista que el focus vuelva a su punto de origen. Comprobar que al guardar la información no se nos muestra el mensaje de descartar cambios.	**No lo pregunta siempre (ej. Año del álbum).**

Guardar i actualizar álbum	Guardar varios álbumes con informaciones diversas (un álbum completo uno lo mas vacio posible, otros normales) que estuvieran editados anteriormente y no editados.	Comprobar que se actualizan todos los timestamps de los mp3. Comprobar que no se ha producido ningún error al guardar los álbumes. Comprobar que tras guardar los álbumes si accedemos a ellos aparecen con toda la información introducida. Comprobar que esa información queda bien introducida en los mp3 sin problemas de codificación con los acentos. Los álbumes aparecen en la librería, se puede comprobar con el botón de Ir_a. Comprobar que todos los elementos del álbum y este propiamente dicho se reflejen correctamente en Drupal, tanto su información como sus relaciones con otros elementos como puede ser la tabla *sequence* o *taxonomías* asociadas al álbum, que aparezca en el buscador de la web, etc.	**La prueba es satisfactoria. Guarda los datos correctamente.**

Administrar Artista	Abrir artista.	Comprobar que se muestran correctamente todos los datos. Si el músico ya está en una formación será imposible convertirlo en banda hasta que se borre de esas formaciones (El checkbox en MLE aparece deshabilitado).	**La prueba es satisfactoria.**
Administrar Artista	Crear o actualizar un artista.	Al actualizar un artista todos los temas de sus álbumes tienen que actualizar el nombre y el timestamp. Se guardan correctamente los datos tanto en mp3 como en la BD y se actualizan los typeahead. Comprobar que todos los elementos del álbum y este propiamente dicho se reflejen correctamente en Drupal, tanto su información como sus relaciones con otros elementos como puede ser la tabla *sequence* o *taxonomías* asociadas al álbum, que aparezca en el buscador de la web, etc. Probar que pasa si un músico pasa a ser una banda, se actualiza correctamente i a posteriori se puede añadir una formación a esa banda.	**La prueba es satisfactoria.**

Administrar Instrumentos	Crear e actualizar varios instrumentos.	Comprobar que los datos actualizados se modifican correctamente en la DB. Comprobar que los datos editados se muestran correctamente, ojo con los acentos y caracteres raros. Comprobar que los cambios realizados se ven reflejados en Drupal.	**La prueba es satisfactoria.**
Otras modificaciones inversas desde Drupal	Modificar en Drupal los campos (3 pruebas) título, año y genero.	Verificar que, para ese álbum, sus temas aparecen con no editados (check no marcado pero álbum como editado y cuando vamos a cambiar de álbum sin haber guardado, avisa de si queremos perder los cambios). Comprobar que si se guarda el álbum se cambian los valores id3 del campo cambiado en Drupal en los temas.	**La prueba es satisfactoria.**
Otras modificaciones inversas desde Drupal	Modificar en Drupal un campo de álbum que no sea ninguno de los anteriores.	Verificar que, para todos los temas del álbum aparecen como editados. Comprobar que se da de alta en Drupal como not published.	**La prueba es satisfactoria.**

Otras modificaciones inversas desde Drupal	Crear un artista que no tenga informados (repetir con varias combinaciones de estos campos no informados: private String nombre; private String foto_url; private int foto_autor_id; y uno o ambos de estos dos: private String web; private String myspace;	Verificar que, para todos los álbumes del artista, sus temas aparece con no editados (check no marcado pero álbum como editado y cuando vamos a cambiar de álbum sin haber guardado, avisa de si queremos perder los cambios). Comprobar que si se guarda el álbum se cambian los nombres de artista en los temas. Verificar en la segunda prueba que los álbumes en los que el músico aparece en la formación muestran el cambio de nombre.	**No funciona, hay combinaciones que no son correctas se debe estudiar a fondo para ver el denominador común.**
Otras modificaciones inversas desde Drupal	Modificar en Drupal el nombre de un artista de tipo banda Repetir la prueba con un artista de tipo músico.	Verificar que, para todos los álbumes del artista, sus temas aparece como editados.	**La prueba es satisfactoria.**
Otras modificaciones inversas desde Drupal	Modificar en Drupal un campo de artista (no nombre).	Comprobar que los álbumes –uno o dos- de esa discográfica reflejan el (visualizan) cambio.	**La prueba es satisfactoria.**
Otras modificaciones inversas desde Drupal	Cambiar el nombre de una discográfica.	Comprobar que al abrir el álbum se refleja el cambio.	**La prueba es satisfactoria.**

Otras modificaciones inversas desde Drupal	La discográfica de un álbum.	Comprobar que al abrir el álbum se refleja el cambio.	**La prueba es satisfactoria.**
Otras modificaciones inversas desde Drupal	El instrumento de un músico en la formación de un álbum.	Comprobar que al abrir el álbum se refleja el cambio.	**La prueba es satisfactoria.**
Pruebas de conexión	Abrir un álbum / artista cortar la conexión a internet y probar a guardar.	Comprobar que sale un mensaje de error conforme no hay conexión y no se puede grabar la información.	**La prueba es satisfactoria. El mensaje aparece, el administrador quiere modificar el mensaje de error.**
Pruebas de conexión	Abrir la pantalla de instrumentos cortar la conexión a internet y probar a guardar.	Comprobar que sale un mensaje de error conforme no hay conexión y no se puede grabar la información.	**La prueba es satisfactoria. El mensaje aparece, el administrador quiere modificar el mensaje de error.**

10. Manuales

10.1. Instalación de Drupal

10.1.1. Entorno de trabajo

Para el desarrollo de este PFC se ha instalado Drupal en un hosting compartido. Dado que Drupal funciona sobre la plataforma LAMP (Linux, Apache, MySQL y PHP), ha habido que buscar uno que tuviera este software instalado. Concretamente, éstas son las características del hosting:

- Arquitectura i686
- Sistema Operativo Linux
- Versión de Apache 2.2.8 (Unix)
- Versión de MySQL 5.0.51a
- Versión de PHP 5.2.6
- Versión de Drupal 5.7
- Versión de cPanel 11.23.3-RELEASE

A pesar de utilizar PHP5, Drupal se ejecuta sobre PHP4, es decir, que no aprovecha (hasta la versión Drupal 6 por lo menos) la orientación a objetos de PHP. Eso no impide que se pueda ejecutar en versiones posteriores de PHP y que funcione correctamente.

Cuando se inició el proyecto la versión estable de Drupal era la 5, estando la 6 en fase beta. Pero poco antes de empezar a implementar apareció la primera versión estable de Drupal 6, con lo que hubo que decidir cuál de las dos se utilizaría. Drupal 6 mejoraba bastantes aspectos con respecto a la versión 5, no sólo de rendimiento sino de facilidad de uso y mantenimiento; algunos módulos que antes eran opcionales ahora se habían añadido al core; los contadores internos funcionaban de forma diferente; se pone un límite de usuarios conectados; más registros de acceso; mejor integración con Ajax, utilizado ya en los propios menús de administración; mejoras en las taxonomías y en la inclusión en la web semántica; más facilidad y control en la instalación y desinstalación de módulos, etc.

Pero el problema con las versiones de los módulos de Drupal es que sólo sirven para una misma versión del core, de modo que al pasar de la versión 4 a la 5 hubo que rediseñar todos los módulos desarrollados por terceros y al migrar a la versión 6 ocurriría lo mismo. Como el proyecto tenía que estar terminado en pocos meses, no había tiempo para esperar a que la actualización general estuviera terminada o por lo menos avanzada. Por esa razón se optó por usar Drupal 5.

Por lo tanto, las tecnologías usadas en el lado de servidor están claras: programación con PHP y bases de datos de MySQL. Para el lado de cliente se utilizarán lenguajes como XML, XHTML, CSS y Javascript (concretamente, jQuery). También se aprovecharán estos lenguajes para hacer pequeñas funcionalidades en Ajax.

10.1.2. Instalación básica de Drupal

Archivos y base de datos

La instalación de Drupal es muy sencilla. Simplemente hay que crear una base de datos en la que se alojará (Figura 10.1). Eso puede hacerse desde el CPanel del hosting.

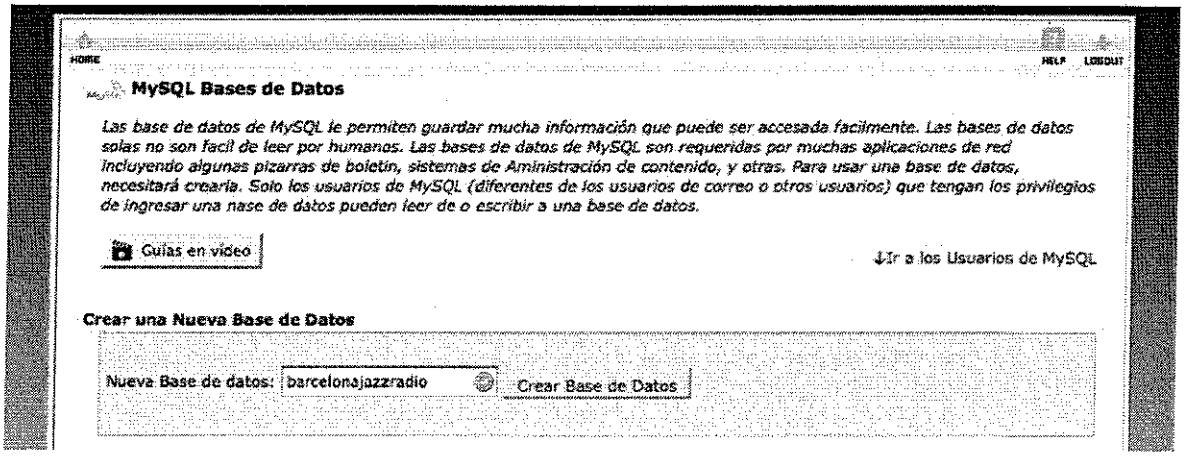


Figura 10.1: Creación de la base de datos

A continuación se le asigna un usuario (Figura 10.2). Se puede utilizar uno que ya esté creado o crear uno nuevo. Es habitual que automáticamente aparezca el nombre de usuario del hosting como prefijo del nombre de la base de datos y del usuario, separados por un guión bajo. Los nombres que necesitaremos a continuación deben incluir ese prefijo.

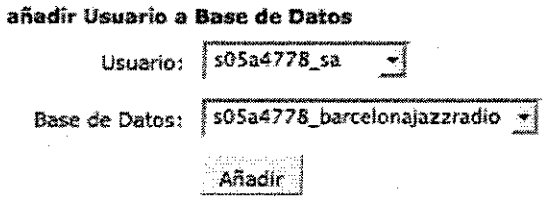


Figura 10.2: Asignar usuario a la base de datos

Y se le dan los privilegios para que pueda actuar con libertad como se muestra en la Figura 10.3.

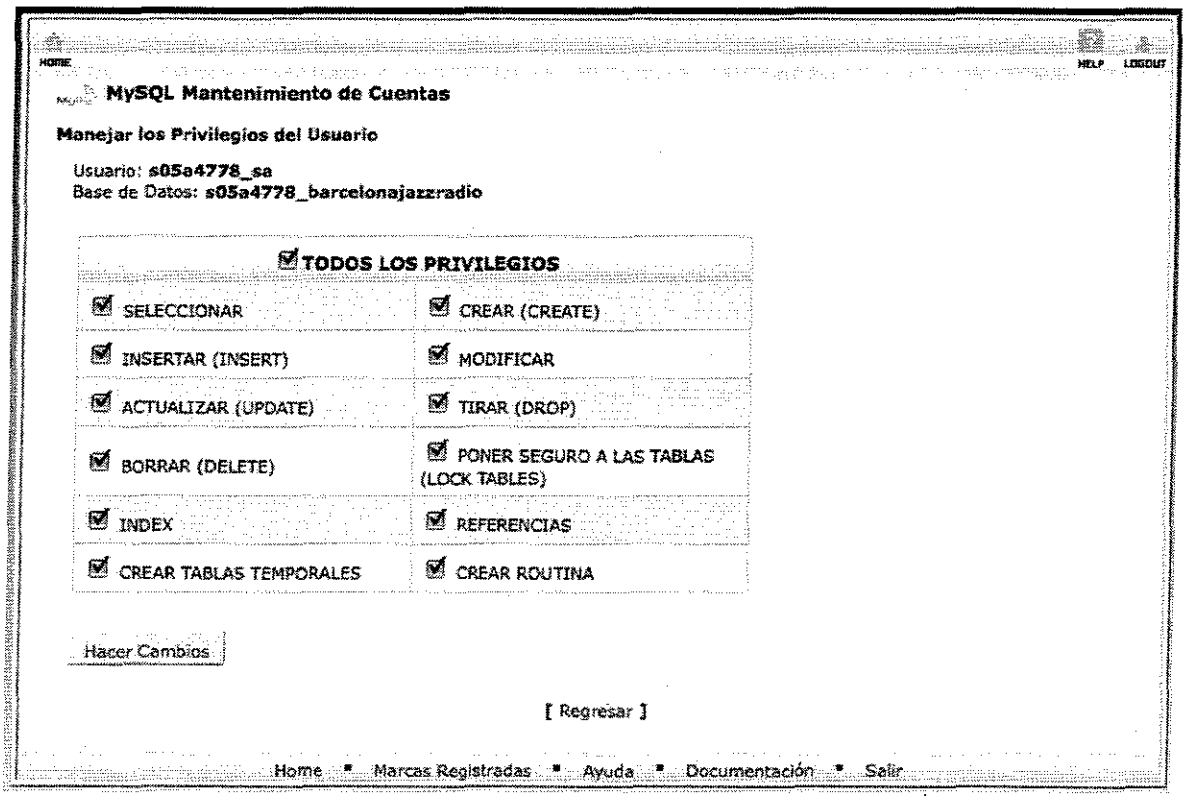


Figura 10.3: Privilegios del usuario de la BD

Se descarga una copia comprimida del sitio de Drupal y se descomprime. Eso dejará en el disco duro la estructura de directorios de Drupal con todos los archivos básicos necesarios. Se renombra el directorio base, que tendrá un nombre como `drupal-5.7`, al nombre deseado, por ejemplo `barcelonajazzradio`, y se sube por FTP a un directorio del hosting al que tenga acceso el servidor web. Se puede subir sólo el contenido del directorio si se quiere acceder directamente a la raíz del servidor.

A continuación se accede a través de un navegador a esa ruta del servidor. Drupal avisará de que no tiene permisos para escribir en el fichero `./sites/default/settings.php`. En este archivo guarda Drupal información importante como la de acceso a la base de datos. Una vez se le den permisos de escritura y se recargue la página, Drupal mostrará un mensaje como el de la Figura 10.4 en el que pedirá esa información al usuario. Hay que tener en cuenta dos cosas para evitar posibles errores:

- Hay que poner el nombre de la base de datos y el del usuario con el prefijo del usuario del hosting. Si no se hace así, recibiremos mensajes de error diciendo que la base de datos o el usuario no existen.
- Es posible que se pueda conectar a la BD y que identifique al usuario, pero que no pueda abrir la conexión con el socket. Añadiendo al archivo `settings.php` la línea `ini_set('mysql.default_socket', '/tmp/mysql.sock');` debería funcionar.

Una vez introducidos los datos correctamente, Drupal estará instalado y se nos recordará que hay que volver a dejar los permisos del archivo `settings.php` como estaban, ya que no hacerlo sería un grave error de seguridad.

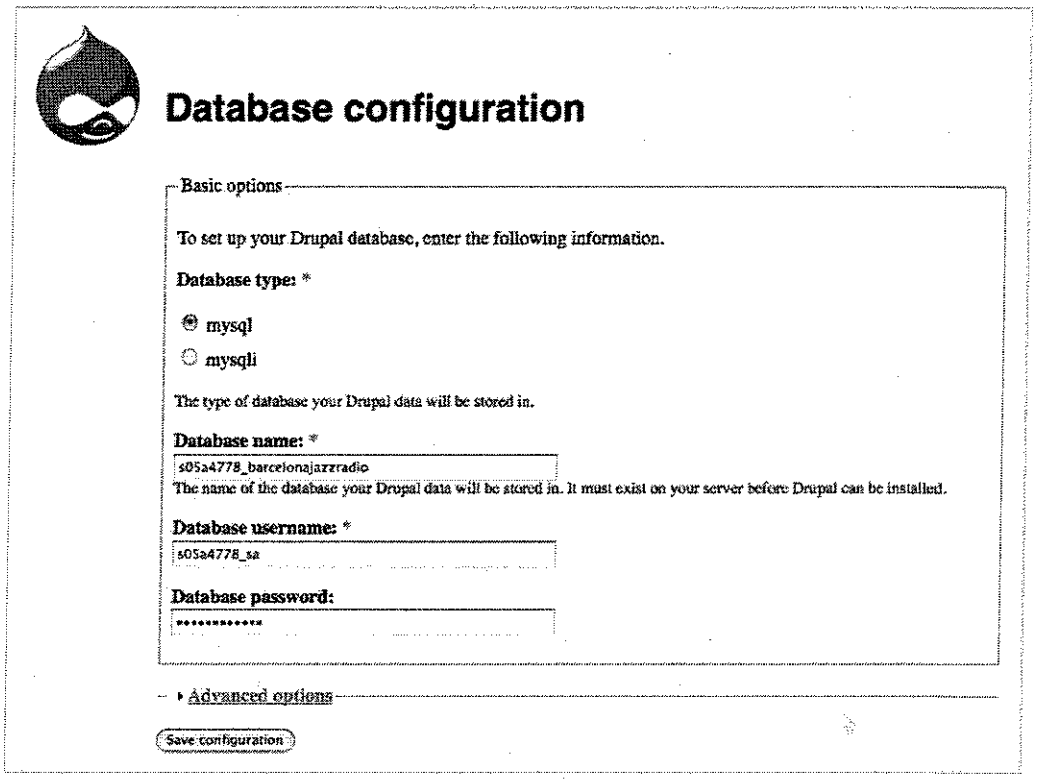


Figura 10.4: Configuración de la base de datos

Después de eso tendremos acceso a nuestro sitio acabado de crear. Yendo a la dirección base del sitio con el navegador llegaremos a la página principal de Drupal como la que se ve en la Figura 10.5.

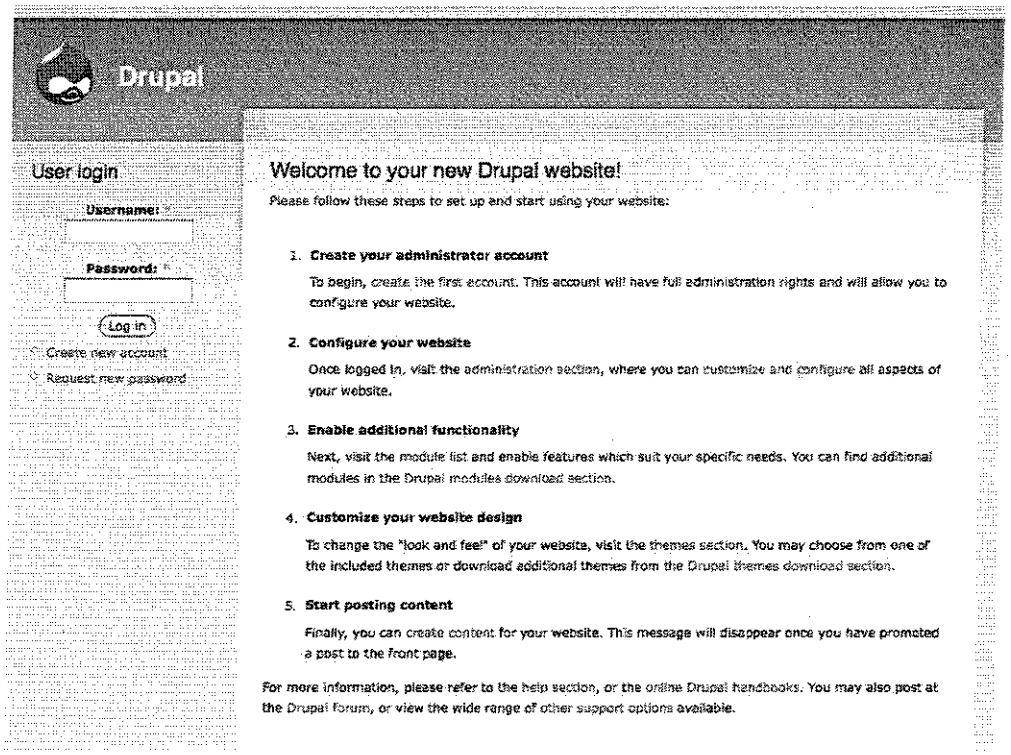


Figura 10.5: Página de inicio de Drupal

A partir de este momento hay varios temas administrativos a resolver. Para empezar hay que crear el usuario principal a partir del enlace proporcionado. Este usuario tendrá todos los permisos en Drupal, y tiene el identificador 1.

A continuación hay que acceder al item *Administer* del menú lateral, lo que nos llevará a una página con cinco elementos de configuración:

- *Content management*: Contenidos, categorías, comentarios, tipos de contenido, RSS...
- *Site building*: Bloques, menús, módulos y temas.
- *User management*: Usuarios, roles, permisos de acceso...
- *Site configuration*: Información y mantenimiento del sitio, sistema de archivos, tema visual de administración, errores...
- *Logs*: Registros, errores de acceso denegado o de página no encontrada y un informe de estado.

Precisamente a este informe de estado hay que ir, ya que es habitual que en esta pantalla se muestre un mensaje de error con una instalación reciente. Concretamente, es probable que haya dos errores:

- El directorio files, donde se guardan los archivos que sube el usuario, no existe. Hay que crearlo y darle permisos de escritura.
- No se ha ejecutado Cron.

Cron

Cron es un script que proporciona Drupal para ejecutar regularmente tareas de mantenimiento. Por ejemplo, actualiza el índice de búsqueda con los nuevos contenidos añadidos. Se puede ejecutar manualmente accediendo a la dirección http://example.com/*ruta_base*`/cron.php` con el navegador. No se mostrará ningún mensaje de confirmación, simplemente una pantalla en blanco indicará que se ha ejecutado correctamente

La otra opción para ejecutar cron es crear un cron job en el sistema operativo. Los hostings suelen permitirlo desde el CPanel. Se puede crear un cron job para que se ejecute cron, por ejemplo, cada hora. Para evitar problemas con las actualizaciones de las webs vía RSS, que en muchos casos se comunican entre ellas a las horas en punto y las horas y media, es recomendable poner el cron, por ejemplo, a las horas y 45 minutos. Esto se haría desde la opción *Cron jobs* del CPanel, accediendo al apartado *Avanzado (estilo Unix)* e insertando lo siguiente, como se ve en la Figura 10.6:

```
45 * * * *    /usr/bin/wget -O - -q http://www.example.com/cron.php
```

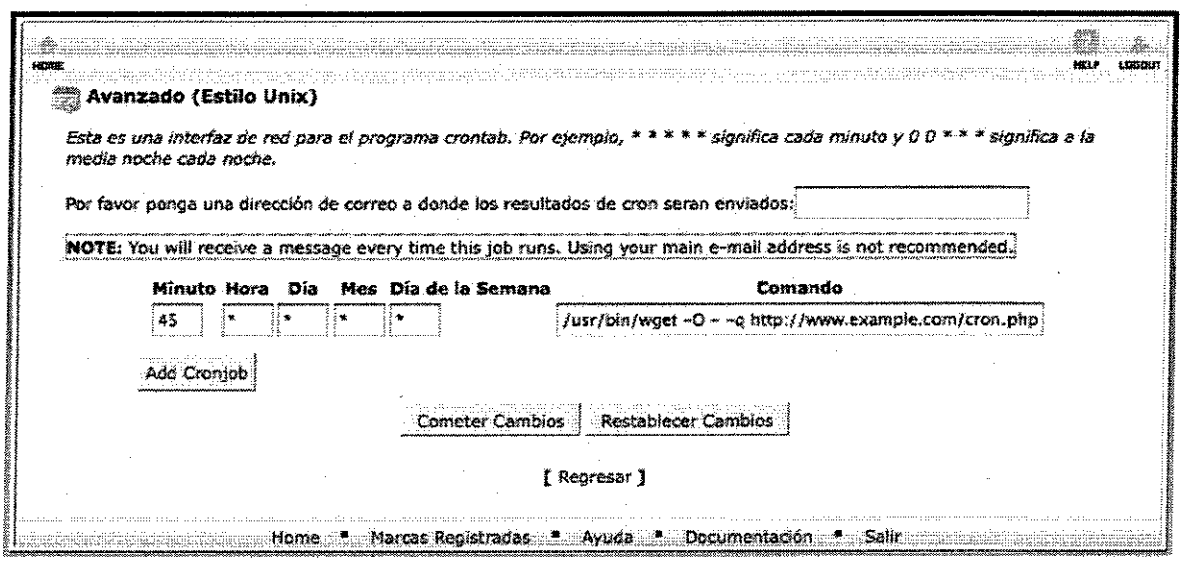


Figura 10.6: Cron job para Cron

De esta forma, el proceso wget se lanzará y visitará la página de Cron como si lo hiciese alguien a través del navegador.

10.1.3. Configuración de Drupal

Una instalación básica de Drupal 5 trae bastantes funcionalidades con las que se puede implementar un gran número de sitios web que no requieran nada demasiado específico. Se pueden crear blogs, ya que permite la publicación de posts, se pueden crear foros, añadir encuestas a la web, etc.

Drupal funciona de forma modular también en el core, lo que significa que existen módulos que ya están instalados en el momento de empezar a trabajar pero que se pueden activar o desactivar. Eso no es cierto para todos los módulos. La instalación de Drupal trae un conjunto de módulos necesarios que no se pueden desactivar y luego otros que vienen de forma opcional. Estos dos grupos de módulos se pueden observar en las Figura 10.7 y Figura 10.8.

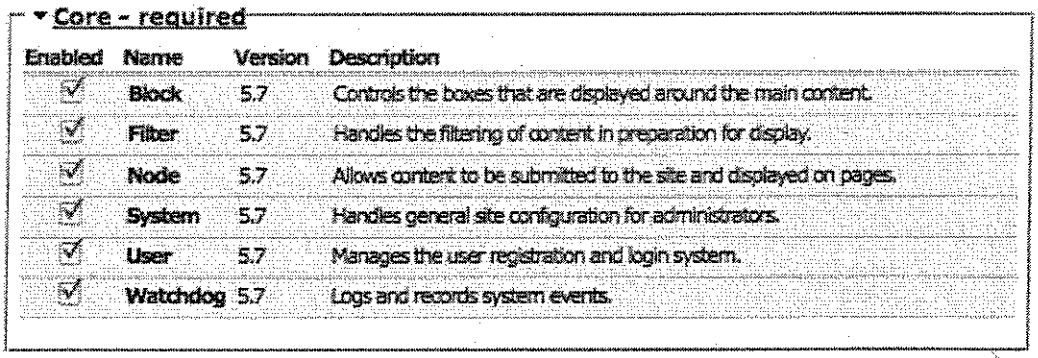
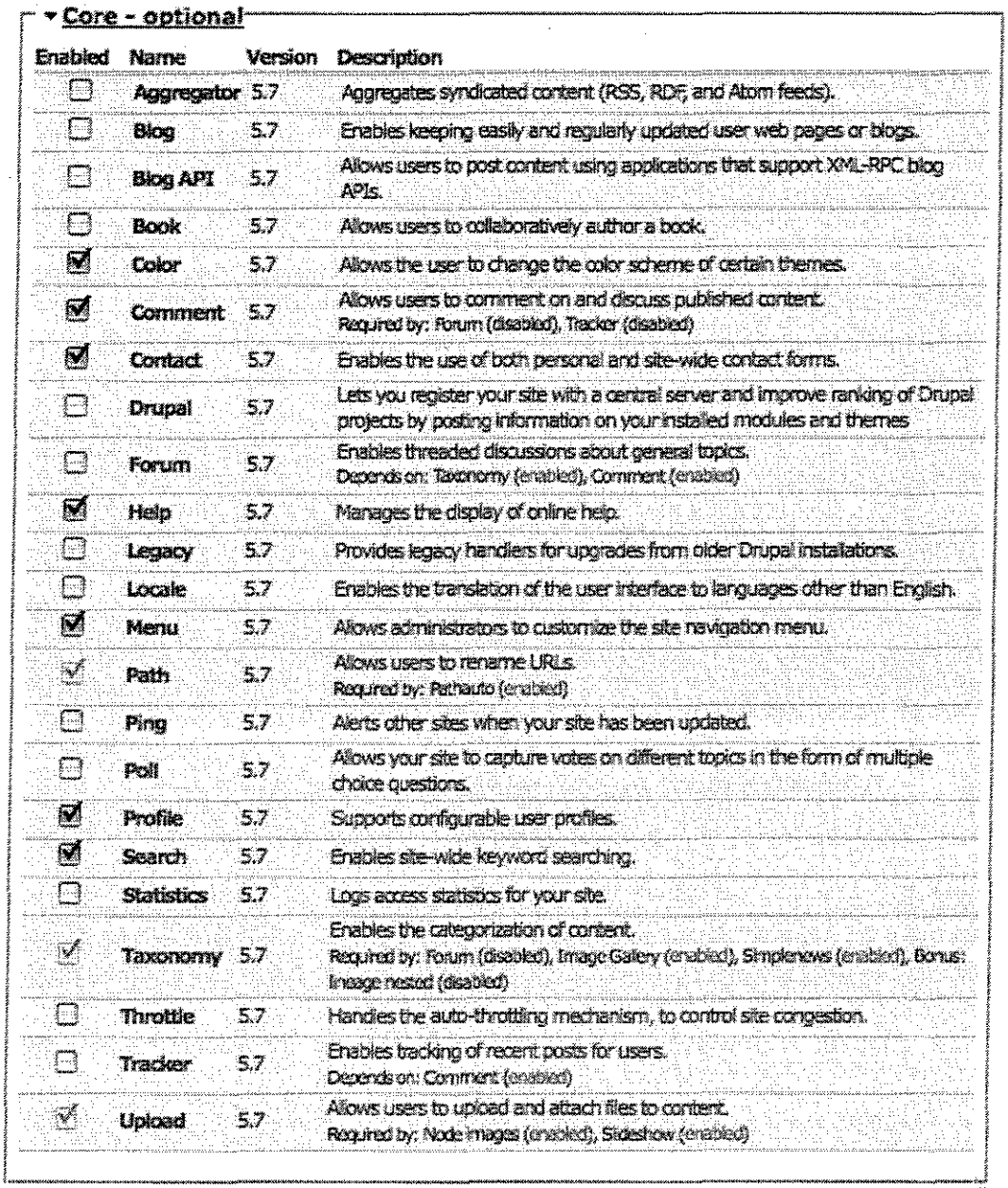


Figura 10.7: Módulos del core necesarios



▾ Core - optional

Enabled	Name	Version	Description
☐	Aggregator	5.7	Aggregates syndicated content (RSS, RDF, and Atom feeds).
☐	Blog	5.7	Enables keeping easily and regularly updated user web pages or blogs.
☐	Blog API	5.7	Allows users to post content using applications that support XML-RPC blog APIs.
☐	Book	5.7	Allows users to collaboratively author a book.
☑	Color	5.7	Allows the user to change the color scheme of certain themes.
☑	Comment	5.7	Allows users to comment on and discuss published content. Required by: Forum (disabled), Tracker (disabled)
☑	Contact	5.7	Enables the use of both personal and site-wide contact forms.
☐	Drupal	5.7	Lets you register your site with a central server and improve ranking of Drupal projects by posting information on your installed modules and themes
☐	Forum	5.7	Enables threaded discussions about general topics. Depends on: Taxonomy (enabled), Comment (enabled)
☑	Help	5.7	Manages the display of online help.
☐	Legacy	5.7	Provides legacy handlers for upgrades from older Drupal installations.
☐	Locale	5.7	Enables the translation of the user interface to languages other than English.
☑	Menu	5.7	Allows administrators to customize the site navigation menu.
☑	Path	5.7	Allows users to rename URLs. Required by: Pathauto (enabled)
☐	Ping	5.7	Alerts other sites when your site has been updated.
☐	Poll	5.7	Allows your site to capture votes on different topics in the form of multiple choice questions.
☑	Profile	5.7	Supports configurable user profiles.
☑	Search	5.7	Enables site-wide keyword searching.
☐	Statistics	5.7	Logs access statistics for your site.
☑	Taxonomy	5.7	Enables the categorization of content. Required by: Forum (disabled), Image Gallery (enabled), Simplenews (enabled), Bonus: Image nested (disabled)
☐	Throttle	5.7	Handles the auto-throttling mechanism, to control site congestion.
☐	Tracker	5.7	Enables tracking of recent posts for users. Depends on: Comment (enabled)
☑	Upload	5.7	Allows users to upload and attach files to content. Required by: Node images (enabled), Slideshow (enabled)

Figura 10.8: Módulos del core opcionales

Como se puede ver, el core incluye obligatoriamente todo lo relacionado con nodos, bloques, usuarios, etc., así como un sistema de *logging* llamado Watchdog.

De forma opcional, en la instalación básica de Drupal se cuenta con un sistema de menús que hace muy sencillo añadir nuevos enlaces a páginas recién creadas; además, incluye blogs, libros (son textos que se pueden escribir de forma colaborativa), encuestas, estadísticas, ayuda, búsqueda, comentarios, RSS... Así como un sistema, Throttle, que permite indicar qué módulos pueden ser desactivados automáticamente si el sistema está sobrecargado. También está la posibilidad de adjuntar archivos a los nodos con el módulo Upload.

En caso de que un módulo requiera de otro para funcionar, se indicará bajo la descripción. Y también cuando un módulo sea requerido por otro. Por ejemplo, el módulo Forum requiere la instalación del Taxonomy y del Comment. Eso se puede ver en los tres módulos

de la imagen, junto con un aviso de si están o no activados. En caso de que ni siquiera estuviesen presentes en el sistema, también se indicaría con un *missing*.

Cuando es necesario extender la funcionalidad de la instalación básica de Drupal, el usuario accede al repositorio de drupal.org y busca entre los cientos de módulos el que le convenga para sus necesidades. Los módulos vienen en archivos que hay que descomprimir dentro del árbol de directorios de Drupal. Hay tres sitios donde se pueden alojar los módulos descargados para que Drupal los encuentre:

- Directorio `/modules`: Es donde están los módulos del core, presentes en el momento de la instalación. No es recomendable si se quiere mantener una estructura limpia con los módulos originales separados de los instalados por el usuario.
- Directorio `/sites/all/modules`: El sitio adecuado si se tiene un solo sitio web o si se trabaja con un sistema multisitio y se quiere que todos los sitios puedan acceder a ese módulo para utilizarlo.
- Directorio `/sites/`*nombresitio*`/modules`: Se utiliza éste directorio si se utiliza un sistema multisitio y se quiere usar el módulo sólo en un sitio concreto.

Una vez están en uno de los tres lugares, Drupal lo detectará cuando el usuario acceda al menú de configuración de módulos. Los módulos instalados estarán en la lista y se podrá marcar la casilla para activarlos.

Una estructura análoga de directorios tienen los themes, cambiando `modules` por `themes` en los tres casos. El administrador puede descargar themes del sitio web de Drupal y descomprimirlos en cualquiera de esos tres directorios. Desde el menú de configuración de los themes se puede cambiar la apariencia del sitio con sólo un clic.

10.2. Instalación de los módulos necesarios

Siguiendo lo que se acaba de explicar, para poder utilizar el módulo `Catalogue` hay que instalar los módulos que éste requieres. La lista de esos módulos se encuentra en el apartado 8.2.

De la mayoría de ellos se ha hecho una instalación estándar, pero algunos ha habido que modificarlos, ya sea desde su interfaz de configuración o desde el propio código. Los cambios más importantes realizados son los siguientes:

- Los módulos `Content` y `Number` de CCK se han utilizado para crear un tipo de datos *weight* para ordenar posts en la portada del sitio web.
- Al módulo `Taxonomy` se le han añadido los vocabularios Artists y Albums, aplicables a los tipos de nodo Image y Story, y el vocabulario Photographers aplicable sólo a los nodos de tipo Image.
- Al módulo `AHAH Forms` se le ha añadido una línea de código para permitir el uso de campos con typeahead.
- El módulo `Widget` que venía con `AHAH Forms` ha sido ampliamente modificado para convertirlo en una herramienta más genérica (ver página 128).
- Los módulos `Image` e `Image Attach` también han sido modificados para cumplir algunos requisitos que habrían requerido mucho esfuerzo de haberse tenido que rodear desde el módulo `Catalogue`, mientras que añadiendo unas líneas a estos módulos se conseguía de forma relativamente fácil.
- El módulo `Views` no se ha modificado pero sí se ha creado una vista desde la interfaz para mostrar en la portada los posts ordenados por peso (el weight creado con CCK).

- Del `Pathauto` sólo destacar que los guiones bajos de las URLs no se han convertido en guiones normales, ya que para Google no suponen actualmente un problema; también, que los nodos de tipo Story tendrán URLs del tipo 'post/[title-raw]'.

Después de esto se puede instalar el módulo `Catalogue`.

10.3. Manual de usuario

Las tareas que se pueden realizar en el sitio web son de dos tipos: administración y consulta. Las de administración, sin incluir la administración propia de Drupal, consisten en crear contenido como músicos, bandas, álbumes, etc., y las de consulta consisten en visualizar estos contenidos.

10.3.1. Creación de contenido

Antes de poder crear contenidos del Catálogo hay que identificarse en el sistema. Como por ahora se pretende que el sitio web no se abra a todo el mundo para el registro, se ha ocultado el cuadro de login de la página. Por lo tanto, para acceder hay que dirigirse a la ruta `/user` dentro del sitio.

Una vez identificado, el administrador podrá crear contenido accediendo al menú *Create content*, inexistente para los usuarios no identificados. Dentro podrá elegir entre los distintos tipos de contenido disponibles: Album, Band, Musician, Image, Page, Newsletter issue, o Story.

El formulario de creación y edición de un músico tiene el aspecto de la Figura 10.9 y la Figura 10.10. Se empieza introduciendo un nombre y una pequeña descripción, que puede añadirse en formato HTML o bien con el editor WYSIWYG.

A continuación, existe la posibilidad de adjuntar una imagen a ese músico, ya sea subiendo una nueva o aprovechando una existente. También en este momento se le puede asignar esa fotografía a un fotógrafo. El título de la imagen servirá como título del nodo imagen que se generará con este procedimiento.

Después de esto viene el apartado de enlaces. Primero, un sitio web, y luego un id de MySpace. Y justo debajo aparece el primer uso del módulo `Widget`, cuyo funcionamiento puede observarse en la Figura 10.11. Escribiendo una URL y una descripción y seleccionando un peso (de -10 a 10), una vez que se le dé al botón *Update* esa línea pasará a la parte superior. Si hacemos lo mismo varias veces los distintos enlaces se irán ordenando según el peso. En otros widgets, como los de instrumentos o formación, se incluye una casilla con un mensaje como "Fill with artist values (will ignore any value below)". Esto significa que marcando esa casilla el sistema intentará buscar valores por defecto para esos campos, sin hacer caso de lo que el usuario inserte. Esos valores pueden ser músicos e instrumentos de una banda para rellenar la formación de un disco suyo, instrumentos que toca un músico para ponerlos en su ficha, etc.

Otro elemento importante es el campo de typeahead. En el formulario de un álbum, por ejemplo (Figura 10.12), encontramos la asignación del álbum a un artista. Para ello se nos presenta un campo de texto en el que podemos empezar a escribir un nombre y se nos mostrarán todos los artistas que empiecen por ese nombre, pudiendo elegir uno de ellos. Estos campos se reconocen por el círculo de la derecha, que gira cuando está buscando.

Una vez que se ha rellenado todo el formulario, se puede hacer clic en Preview, para ver la vista previa del músico, o bien en Submit, para insertarlo directamente en la BD.

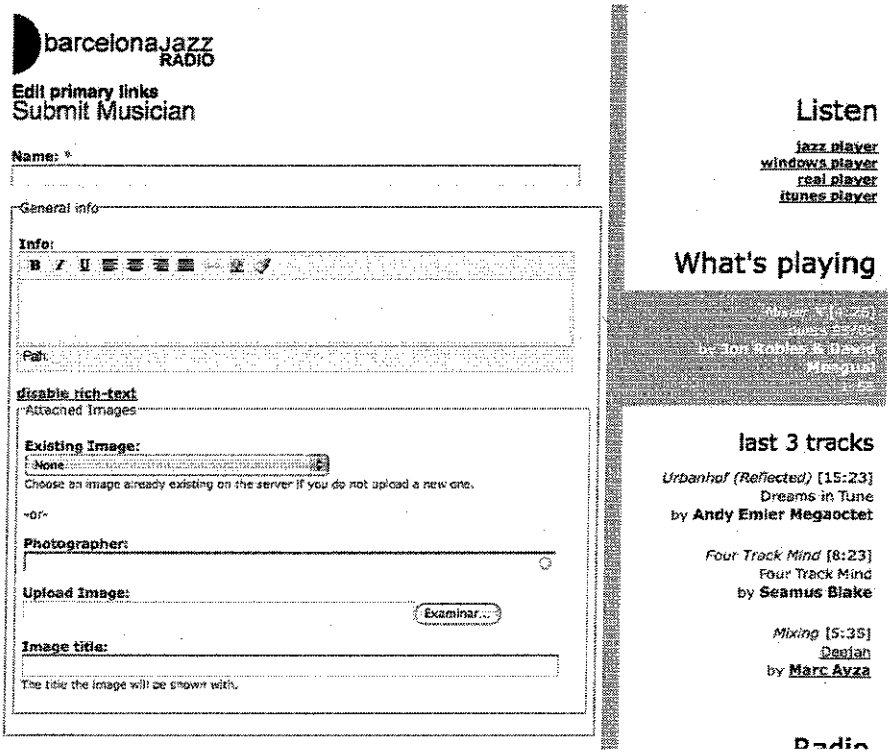


Figura 10.9: Formulario de músico I

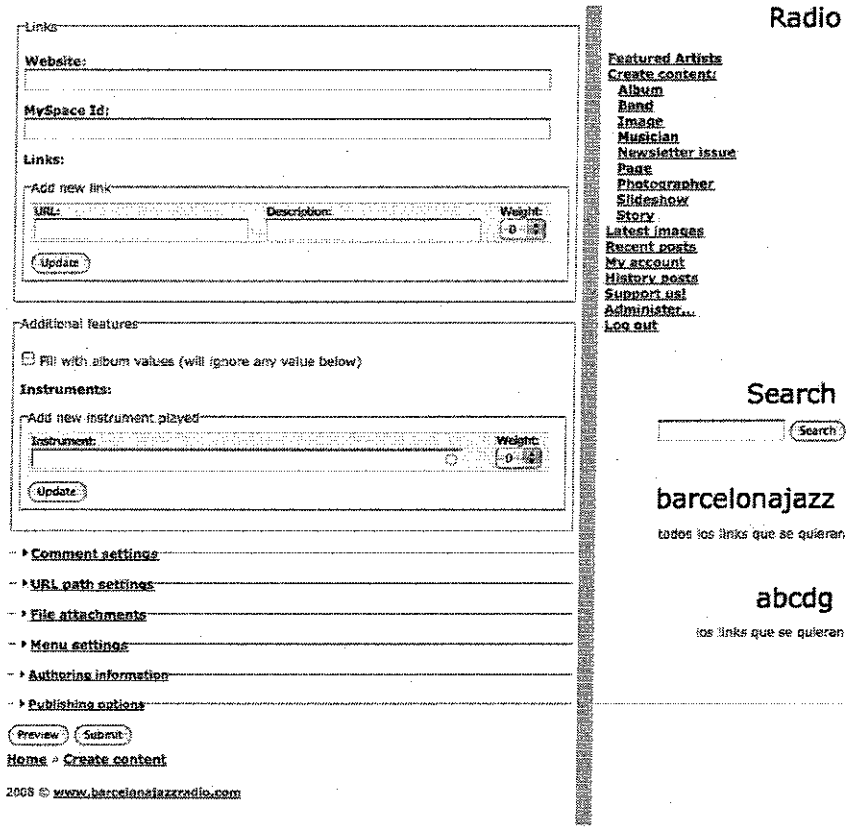


Figura 10.10: Formulario de músico II

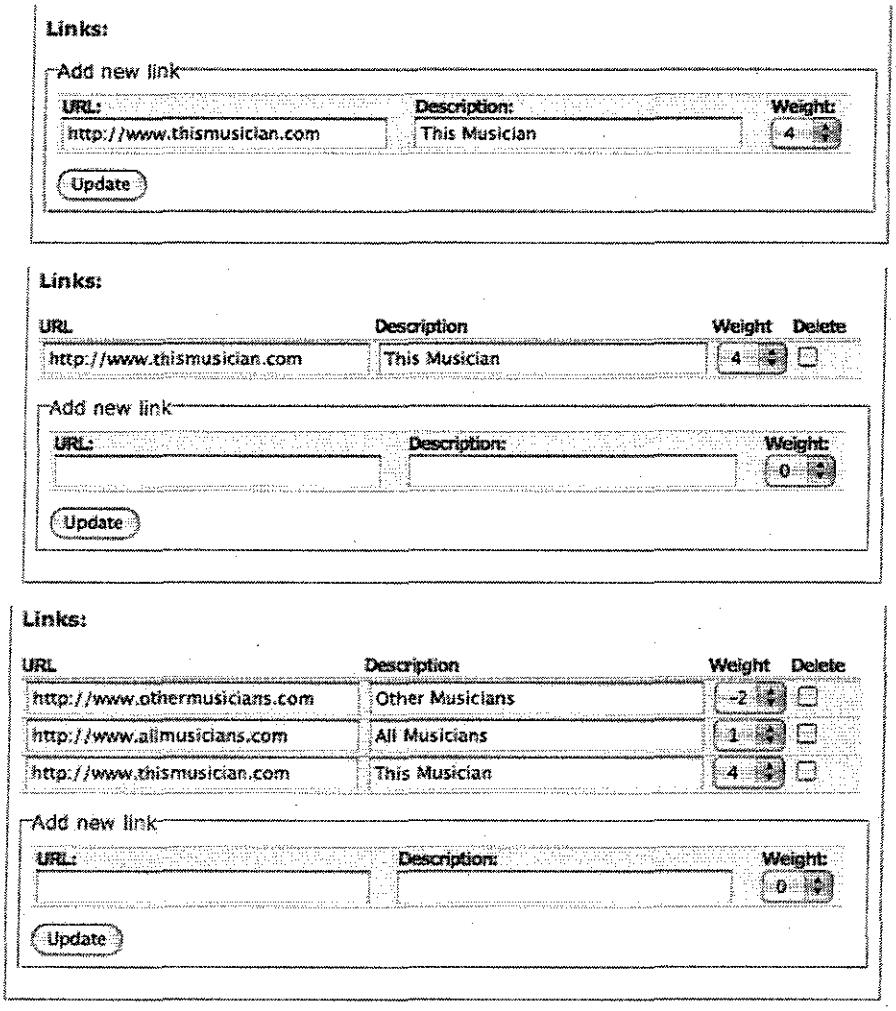


Figura 10.11: Widget de links

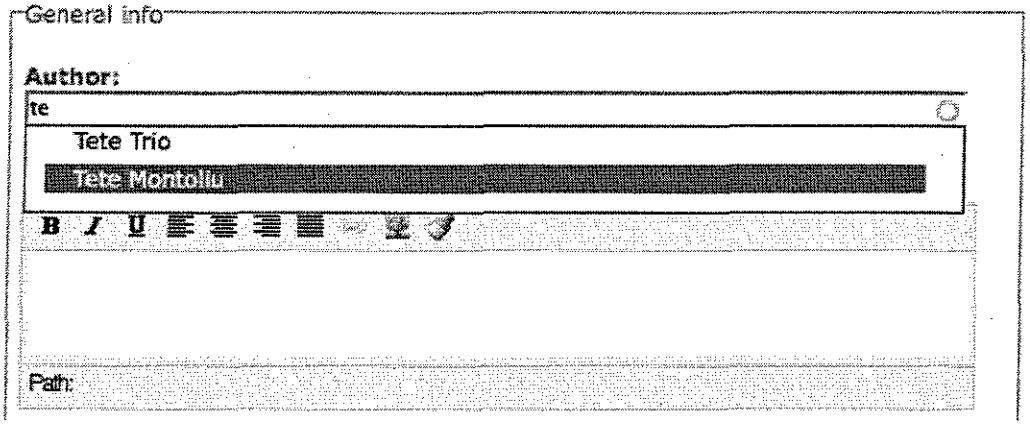


Figura 10.12: Campo de typeahead en el formulario de un álbum

Algunos detalles a tener en cuenta para el administrador a la hora de insertar contenidos:

- Como se ha dicho en la sección 2.3.1, no es fácil decidir qué músicos se ponen en la formación de una banda, ya que puede haber varias formaciones importantes. Por eso es conveniente que el administrador, antes de insertar un álbum de una banda

determinada, se fije en si los componentes del álbum coinciden o no con los de la banda a la que se lo va a asignar. En caso contrario, tal vez desee crear una nueva banda con el mismo nombre que la otra pero con el sufijo (II) y asignar el álbum a ésa en vez de a la otra.

- Si se elimina un artista también se eliminarán todos sus álbumes y desaparecerá de las formaciones en las que estaba.

Si se quieren importar imágenes de forma masiva, el módulo Image Import permite hacerlo muy fácilmente. Sólo hay que subir las imágenes a un directorio del servidor, acceder por los menús a *Administer/Site Configuration/Image Import* y añadir información a todas las imágenes que se mostrarán. Una vez importadas desaparecerán de ese directorio.

10.3.2. Navegación

Una página importante del sitio web es la de *Featured Artists* (Figura 10.13), a la que se accede por el menú de la derecha. En ella se muestran todos los artistas que hay registrados en el sistema, con un enlace a sus fichas. En caso de que el artista tenga la ficha como no publicada, aparecerán los enlaces en un tono más suave si se está identificado como administrador o bien no aparecerán si no se está identificado.

La navegación por el resto del sitio es sencilla. Desde la portada se puede acceder a los posts publicados, los cuales están asignados a músicos o álbumes, a los que se puede acceder desde ellos.

Dentro de la ficha de un artista se pueden encontrar enlaces a sus webs, *teasers* de sus posts, álbumes que ha grabado e imágenes suyas. Desde las imágenes también se puede acceder a la galería del fotógrafo, etc.

También se puede acceder a todos los contenidos utilizando el buscador del sitio web, que incluye un buscador avanzado.

10.3.3. Configuración

El módulo sólo ofrece la posibilidad de configurar dos parámetros, ambos relacionados con la playlist: la URL del XML de Live365 y el número de temas anteriores que se mostrará en el lateral. Para acceder a esta configuración hay que ir al menú de Administración y luego al menú de configuración del módulo `Catalogue`. Una vez allí, aparecerá un formulario como el de la Figura 10.14.

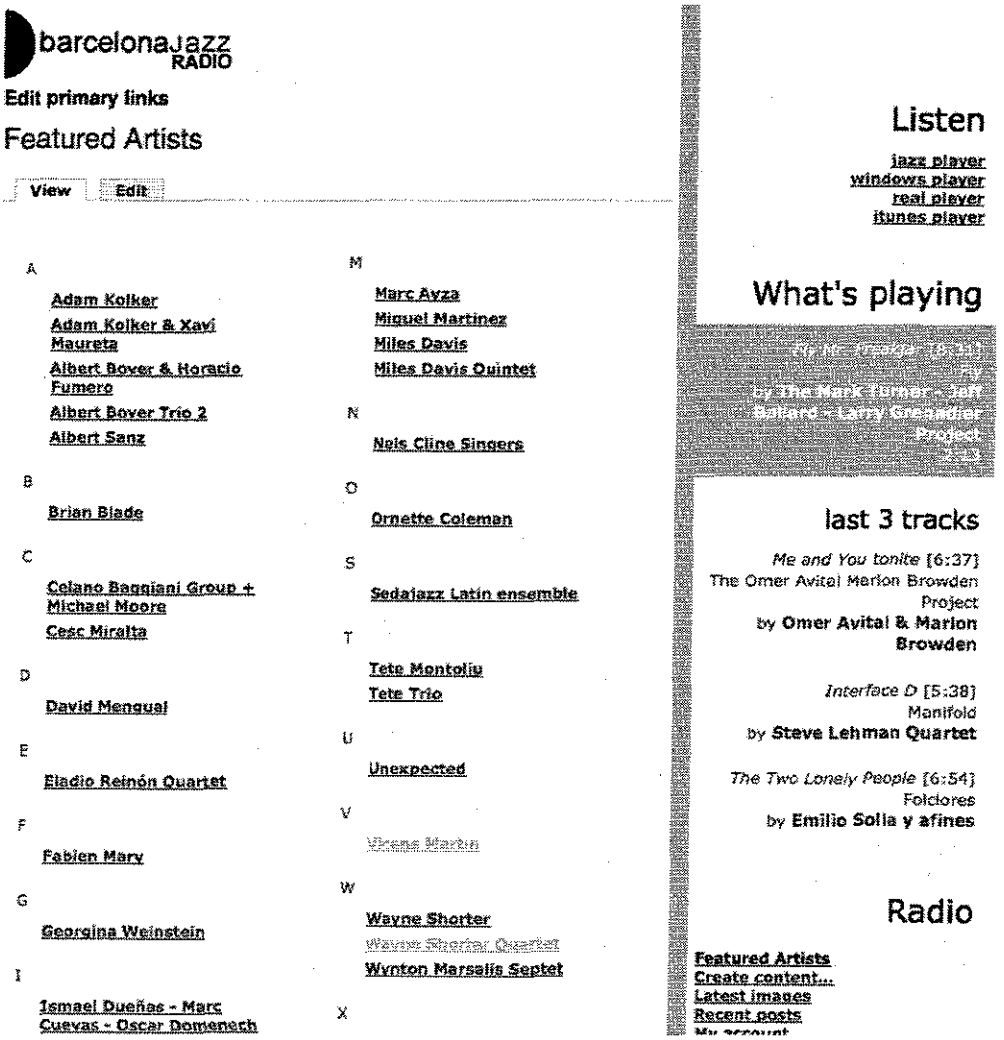


Figura 10.13: Featured Artists

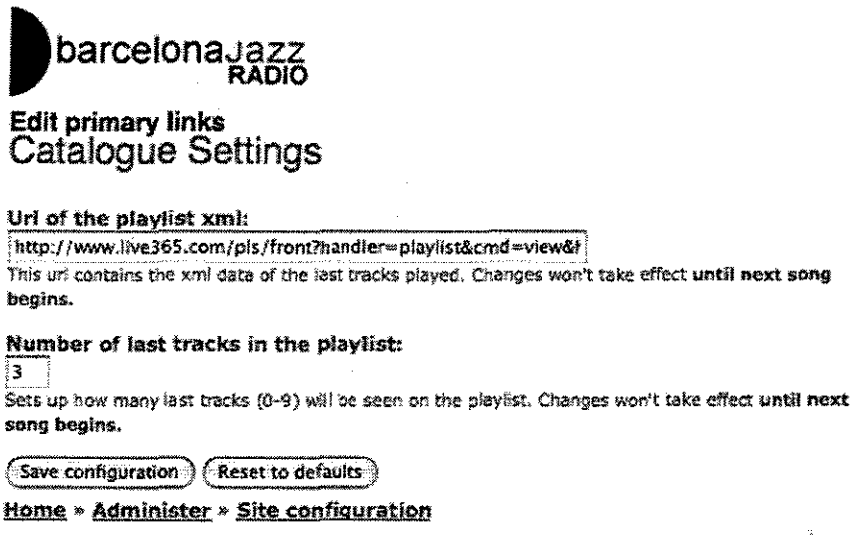


Figura 10.14: Configuración del módulo Catalogue

11. Conclusiones

En este capítulo final, se expone mi experiencia concreta en la realización de este proyecto. En concreto, se exponen los logros conseguidos, errores cometidos y a modo informativo el futuro de este proyecto.

11.1. Objetivos alcanzados

En principio, se ha llegado a los objetivos establecidos al principio del proyecto que básicamente ha sido crear un sistema web que cumpliera una serie de requisitos acordados en la fase de toma de requerimientos donde participó el administrador de la radio, el director de proyecto y los proyectistas.

Cabe recordar que este proyecto consta de dos partes, una primera fase que implementa la funcionalidad base (cms, catálogo musical, sincronización, conexión con Live365...) y poca funcionalidad "2.0", que es la que implementa este proyecto y una segunda fase posterior que se implementará en futuros pfc's.

La primera fase del proyecto deja el camino ***muy llano*** para la implementación de la segunda fase.

11.2. Aportación del proyecto

Este proyecto final de carrera ha sido para mí el colofón de la carrera, una larga lucha de varios años donde he aprendido muchas cosas algunas buenas y otras no tanto. En esta lucha, este proyecto ha sido especialmente útil ya que he aprendido mucho: que es un cms, utilizar php, conceptos técnicos, y un largo etc.

Pero me gustaría destacar que lo más importante que he aprendido ha sido: **trabajar en equipo**, aunque no sea algo tangible, ha dado mucho sentido a este proyecto y aunque he cometido errores, he sabido subsanarlos de una forma más o menos eficaz.

Al principio, creía que sabía trabajar en equipo ya que profesionalmente colaboro con mucha gente, pero siempre en proyectos pequeños/medianos. En cambio este proyecto es de bastante envergadura y me he dado cuenta que no tenía todos los conocimientos/habilidades necesarias para coordinarme con todos mis compañeros de una manera eficaz, pero después de mucho esfuerzo creo que he conseguido aprender a trabajar en proyectos grandes.

Por otra parte, he conseguido experiencia en el mundo del diseño gráfico, que considero una tarea importante a la hora de implementar un sistema, y que poca experiencia había tenido hasta el momento. El diseño de la aplicación ha sido costoso pero muy gratificante tras el esfuerzo, ayuda y aportación de ideas por parte de todos.

He aprendido a utilizar con un nivel muy alto tecnologías tan importante como PHP, CSS, HTML... que seguro me sirven para mi vida profesional.

En menor medida, he aprendido a autoformarme ya que en toda la carrera nos inculcan este método como forma complementaria de aprendizaje.

Lo que más ha motivado tanto a mí como a mis compañeros ha sido entre otras cosas:

- Saber que este proyecto es algo real, que se va a usar y que encima será una web importante dentro de la escena musical barcelonesa.
- Poder tener tiempo para aprender una tecnología nueva que esperamos que de aquí a poco tiempo esté plenamente implantada en muchos sitios web importantes. Ya que en Estados Unidos empresas u organismos estatales como la NASA, MTV, Yahoo Research, Mobile data... lo usan.
- Nuestro primer proyecto serio como ingenieros (bueno mejor dicho casi ingenieros).

En resumen, ha sido un proyecto largo y duro pero que una vez finalizado vemos que el trabajo nos puede dar un empujón a nuestra carrera profesional ya que no hay muchos profesionales en este sector en concreto.

11.3. Problemas encontrados

Los problemas más importantes que nos hemos ido encontrando en este proyecto han sido:

- Falta de método en el trabajo en equipo. A veces ha habido falta de coordinación entre los integrantes del proyecto. Este problema se ha ido subsanando conforme el proyecto ha ido avanzando.
- Falta de conocimiento en el sistema que se estaba modificando, muchas veces nos hemos encontrado con problemas de implementación donde Drupal trata los datos de una manera y se ha tenido que investigar más de lo necesario para poder entender como Drupal gestiona la información.

Personalmente considero que estos problemas son "normales" dentro de un proyecto grande.

11.4. Desarrollo de futuro

Aunque el proyecto ha adquirido los objetivos que se marcaron en un principio, hay que destacar que quedan pendientes para futuras implementaciones todos aquellos requerimientos que en el apartado de requerimientos se han marcado como prioridad 2 o el propio director de proyecto estimó que serían para fututos pfc's, en concreto estas tareas son a grandes rasgos:

- Gestión de usuarios. Cada músico sea un usuario y pueda modificar sus posts, etc...
- Implantación de un foro.
- Gestión del idioma según GeoIP.
- Posts en varios idiomas según el idioma predeterminado del usuario.
- Envío de archivos (música, partituras...) al músico.
- Gestión de temas de álbumes (la gestión se hace a nivel de álbum, no de tema).

Anexo I: Bibliografía

COGGESHALL, John. *La Biblia de PHP 5*. Madrid: Ediciones Anaya Multimedia, 2005.

CONALLEN, Jim. *Modeling Web Applications with UML*, Conallen, Inc., 1999

SHREVES, Ric. *Drupal 5 Themes*. Birmingham, UK: Packt Publishing, 2007.

VANDYK, John K., WESTGATE, Matt. *Pro Drupal Development*. Berkeley, CA: Apress, 2007.

Sitios web

www.drupal.org

www.drupal.org.es

www.wikipedia.com

www.cocinandocondrupal.com

www.drupalhispano.com

... y decenas de páginas, blogs y foros sobre Drupal encontrados vía Google que ofrecen soluciones rápidas a problemas comunes y no tan comunes.

Anexo II: Glosario

Ajax AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Anchor text El anchor text o etiqueta del enlace es la parte visible y clicable de un hiperenlace. Las palabras contenidas en el anchor text pueden determinar la posición de esa página en los motores de búsqueda.

Álbum En el ámbito de este PFC es una grabación realizada por uno o más músicos. Uno de esos músicos es el que tiene el disco a su nombre, mientras que el resto forma parte de su *formación*.

Artista En el contexto de este PFC, es uno de los dos tipos de Artista. Se diferencia del Músico porque está formado por una formación de músicos, mientras que el Músico es un solo componente que toca uno o más instrumentos.

Attach_Image Módulo Drupal que permite agregar una imagen a un tipo de contenido, configurable por BackOffice.

Banda En el contexto de este PFC, es uno de los dos tipos.

Base de datos Una base de datos o banco de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. En la actualidad, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos están en formato digital (electrónico), que ofrece un amplio rango de soluciones al problema de almacenar datos.

BD ver *Base de datos.*

Bitrate En telecomunicación e informática, el término bit rate (en español velocidad binaria, cadencia, tasa o flujo de bits) define el número de bits que se transmiten por unidad de tiempo a través de un sistema de transmisión digital o entre dos dispositivos

digitales. Así pues, el bit rate es la velocidad de transferencia de datos.

Blog Un blog, o en español también una *bitácora*, es un sitio web periódicamente actualizado que recopila cronológicamente textos o artículos de uno o varios autores, apareciendo primero el más reciente, donde el autor conserva siempre la libertad de dejar publicado lo que crea pertinente. El término blog proviene de las palabras *web* y *log* ('log' en inglés = diario). El término bitácora, en referencia a los antiguos cuadernos de bitácora de los barcos, se utiliza preferentemente cuando el autor escribe sobre su vida propia como si fuese un diario, pero publicado en Internet en línea.

Bloque Para Drupal es un fragmento de una página web en el que se muestran contenidos. Pueden incluir nodos o cualquier otro tipo de información.

Buscador web Un motor de búsqueda es un sistema informático que indexa archivos almacenados en servidores web. Un ejemplo son los buscadores de Internet (algunos buscan sólo en la Web pero otros buscan además en News, Gopher, FTP, etc.) cuando se pide información sobre algún tema. Las búsquedas se hacen con palabras clave o con árboles jerárquicos por temas; el resultado de la búsqueda es un listado de direcciones Web en los que se mencionan temas relacionados con las palabras clave buscadas.

Catálogo musical Base de datos que permite catalogar la librería musical registrando los atributos necesario sobre los temas que la componen: título, género, años, álbum, artista, formación musical, etc...

CMS *Content Management System* (o Sistema Gestor de Contenidos) es un programa que permite crear una estructura de soporte (framework) para la creación y administración de contenidos por parte de los participantes principalmente en páginas web.

Consiste en una interfaz que controla una o varias bases de datos donde se aloja el contenido del sitio. El sistema permite manejar de manera independiente el contenido y el diseño. Así, es posible manejar el contenido y darle en cualquier momento un diseño distinto al sitio sin tener que darle formato al contenido de nuevo, además de permitir la fácil y controlada publicación en el sitio a varios editores. Un ejemplo clásico es el de editores que cargan el contenido al sistema y otro de nivel superior que permite que estos contenidos sean visibles a todo el público.

Core En Drupal hace referencia al núcleo, compuesto por unos módulos básicos que le dan sus funciones principales. Suele emplearse como sinónimo de "instalación básica".

CSS Las hojas de estilo en cascada (Cascading Style Sheets, CSS) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores.

La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación.

Drupal Sistema Gestor de Contenidos (o CMS) para sitios Web. Permite publicar artículos, imágenes, u otros archivos y servicios añadidos como foros, encuestas, votaciones, blogs y administración de usuarios y permisos. Drupal es un sistema dinámico: en lugar de almacenar sus contenidos en archivos estáticos en el sistema de ficheros del servidor de forma fija, el contenido textual de las páginas y otras configuraciones es almacenado en una base de datos y se edita utilizando un entorno Web incluido en el producto.

Etiqueta/tag ID3

Metadato que permite registrar atributos de un fichero multimedia siguiendo el estándar ID3. En el ámbito de este proyecto, hacen referencia siempre a las etiquetas que permiten catalogar temas musicales almacenados en ficheros mp3, especificando el título, el álbum, el artista, el género, el año de publicación, así como otros atributos.

Frappr.com Una aplicación web 2.0 que crea mapas de comunidades online.

GeoIP Identificación de dónde se encuentra físicamente una IP.

GeoVisitors Geo Visitors es una herramienta que permite a tus visitantes colocar un pin en el mapa mundi, construyendo de forma visual el origen de tus visitas.

Hook En lenguaje de Drupal viene a ser lo que en otros lenguajes se llama *callback*, es decir, una función que se ejecuta de forma asíncrona cuando se da un cierto evento. Su función es la de permitir al usuario *colgarse* en cualquier momento del flujo de ejecución del programa y hacer modificaciones.

ID3 ID3 es un estándar de facto para incluir metadatos (etiquetas) en un contenedor multimedia, tales como álbum, título o artista. Se utiliza principalmente en ficheros MP3. Ver www.id3.org.

ISP Internet Service Provider. Empresa dedicada a proveer de servicios de Internet, entre ellos, el de alojamiento de aplicaciones y datos en servidores compartidos, así como también en servidores privados virtuales o en servidores dedicados.

iTunes iTunes es una aplicación creada por Apple con el fin de reproducir, organizar y comprar música (es también el nombre común de iTunes Music Store, aunque la palabra "Music" se les ha quedado pequeña, ya que actualmente la tienda vende vídeos musicales, películas, juegos, audiolibros, etc. Por esa razón, desde la versión 7.0 de esta aplicación, se le ha cambiado el nombre a iTunes Store). Es compatible con ordenadores con Mac OS X, Windows 2000, Windows XP o Windows Vista como sistema operativo. Algunas versiones tempranas de iTunes también funcionan con Mac OS 9. Estadísticamente, es el reproductor más usado por los usuarios de reproductores iPod.

iTunes es un sistema basado en SoundJam MP, una popular aplicación de MP3, creada por la compañía Casady & Greene. Apple compró los derechos de SoundJam MP y pronto estrenó la primera versión de iTunes, que era muy similar a SoundJam MP.

Jazz El jazz es un género musical nacido a finales del siglo XIX en Estados Unidos que se expandió de forma global a lo largo de todo el siglo XX.

La historia del jazz se caracteriza por dos rasgos fundamentales:

En primer lugar, tanto por su constante asimilación de otras tendencias musicales estilística o culturalmente ajenas a él, como por su capacidad de mezclarse con otros géneros y crear nuevos estilos musicales, como el rock and roll, que terminarían por evolucionar de forma independiente al jazz.

En segundo lugar, por la sucesión de forma ininterrumpida de un numeroso conjunto de subestilos que, vistos en perspectiva, manifiestan entre algunos de ellos enormes diferencias musicales.

LDAP *Lightweight Directory Access Protocol* (Protocolo Ligero de Acceso a Directorios) es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. LDAP también es considerado una base de datos (aunque su sistema de almacenamiento puede ser diferente) a la que pueden realizarse consultas.

Lenguaje de scripting

Se conoce como lenguaje interpretado a un lenguaje de programación que fue diseñado para ser ejecutado por medio de un intérprete, en contraste con los lenguajes compilados. También se les conoce como lenguajes de script. Muchos autores rechazan la clasificación de lenguajes de programación entre interpretados y compilados, considerando que el modo de ejecución (por medio de intérprete o de compilador) del programa escrito en el lenguaje es independiente del propio lenguaje.

Librería musical Conjunto de temas musicales de diferente artistas y álbumes que han sido seleccionados para ser emitidos por la radio.

Live365 Live365 es un sitio Web de radio por Internet donde los miembros pueden crear su propia emisora de radio en la red o escuchar la de los demás miembros. Actualmente (mediados de 2008) hay unas 11.000 emisoras activas. Algunas de ellas reproducen nichos de géneros que raramente se escuchan en la radio OM/FM tradicional.

Look&feel Este término inglés hace referencia a los diferentes aspectos y funcionamiento de los interfaces gráficos de usuario. Así, un interface basado en ventanas como Windows puede tener un mejor *look and feel* que uno basado en menús desplegables.

MLE Music Library Editor, es una aplicación diseñada específicamente para www.barcelonajazzradio.com mediante la cual es posible la introducción de datos en la web y el sincronizado de esta con la librería musical.

Músico En el contexto de este PFC, es uno de los dos tipos de Artista. Se diferencia de la Banda porque está formado por un solo componente que toca uno o más instrumentos, mientras que la Banda consta de una formación de músicos.

MySpace MySpace es un popular sitio web con una gran comunidad online que ofrece servicios de blog, perfiles personales, búsqueda de amigos, grupos, fotos, música y videos.

Nodo En Drupal cualquier tipo de contenido que pueda tener un título y un cuerpo, un autor, una fecha de creación y edición, etc. es llamado nodo. Un nodo puede ser, por ejemplo, un post de un blog, una imagen, una encuesta o un mensaje en un foro.

NuSOAP NuSOAP es un kit de herramientas (ToolKit) para desarrollar Web Services bajo el lenguaje PHP. Está compuesto por una serie de clases que facilitan el desarrollo de Web Services. Provee soporte para el desarrollo de clientes (aquellos que consumen los Web Services) y de servidores (aquellos que los proveen). NuSOAP está basado en SOAP 1.1, WSDL 1.1 y HTTP 1.0/1.1 http://sourceforge.net/projects/nusoap

PageRank PageRank es una marca registrada y patentada por Google el 9 de enero de 1999 que ampara una familia de algoritmos utilizados para asignar de forma numérica la relevancia de los documentos (o páginas web) indexados por un motor de búsqueda. Sus propiedades son muy discutidas por expertos en optimización de motores de búsqueda. El sistema PageRank es utilizado por el popular motor de búsqueda Google para ayudarle a determinar la importancia o relevancia de una página.

Peer programming

Se llama así a esta técnica de programación que consiste en programar código en grupos de dos. Cada persona mira el código que ha hecho el compañero justo después de que lo escriba. La colaboración, unida al hecho de que cada uno valida al otro, asegura que las rutinas estén más pensadas. También ayuda a asegurar que el código esté mejor documentado.

Playlist Lista de reproducción. Una lista de temas musicales seleccionados para ser reproducidos con algún software. En el contexto de este proyecto tiene varios significados: la lista de reproducción creada por el administrador de contenidos para subir a Live365 (llamada en general programación de la radio); el fichero XML con 10 temas que envía Live365 a los reproductores (llamado en la memoria "XML de Live365"); la versión reducida de ese fichero almacenada en el servidor de barcelonajazzradio.com (playlist del servidor); la versión del XML del servidor que se envía al cliente (playlist de cliente) y, por último, la lista de reproducción que se muestra en el sitio web con enlaces a las fichas de artistas y álbumes (playlist del sitio web).

RealPlayer RealPlayer es un reproductor multimedia multiplataforma perteneciente a la compañía RealNetworks. Ejecuta múltiples formatos multimedia como MP3, MPEG, QuickTime, Windows Media y múltiples versiones de los formatos RealAudio y RealVideo.

Rol Características que comparte un grupo de usuarios dentro de un sitio web, lo que los hace iguales en permisos de cara al sistema.

Ruby on Rails También conocido como RoR o Rails es un framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby, siguiendo el paradigma de la arquitectura Modelo Vista Controlador (MVC). Trata de combinar la simplicidad con la posibilidad de desarrollar aplicaciones del mundo real escribiendo menos código que con otros frameworks y con un mínimo de configuración

RW Radio Website, parte de este proyecto que se encarga de la gestión del sitio web que da soporte a la radio, www.barcelonajazzradio.com.

Streaming Streaming es un término que se refiere a ver u oír un archivo directamente en una página web sin necesidad de descargarlo antes al ordenador. Puede decirse que describe una estrategia sobre demanda para la distribución de contenido multimedia a través del Internet.

SGBD Los Sistemas de Gestión de Bases de Datos (en inglés: Database Management System, abreviado DBMS) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

Sitemap Un site map (o sitemap) es una página web que lista las páginas en un sitio web, organizadas comúnmente de forma jerárquica. Esto ayuda a los visitantes y a los motores de búsqueda a hallar las páginas en un sitio. Los site maps pueden mejorar la optimización para los motores de búsqueda de un sitio asegurándose que todas ellas puedan ser encontradas.

Slide.com Sitio web que permite la creación de pases de diapositivas a partir de fotografías del usuario.

SOAP SOAP (siglas de Simple Object Access Protocol) es un protocolo estándar creado por Microsoft, IBM y otros, actualmente bajo el auspicio de la W3C, que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. SOAP es uno de los protocolos utilizados en los servicios Web.

SP ver *stored procedure.*

sproc ver *stored procedure.*

stored procedure

Un stored procedure es un programa (o función) el cual está físicamente almacenado en un banco de datos o base de datos. La exacta implementación de un stored procedure incluyendo lo que es y cómo es implementado varía de un sistema gestor de bases de datos a otro.

tag ID3 ver *etiqueta ID3*

Taxonomía Es la Ciencia de la clasificación. En el contexto de la Web 2.0 hace referencia a un sistema de etiquetas que permite organizar un sitio web.

Teaser Drupal llama teaser (resumen) a la versión reducida de un nodo que habitualmente se muestra en listas o resultados de una búsqueda.

Tema En el contexto del PFC, una interpretación de un tema real dentro de un álbum. Es decir, una instancia concreta de la composición para un álbum.

Template

language Lenguaje que permite mezclar código estructural con variables que hacen referencia al contenido, de modo que se mejoren el proceso de diseño de una web. Algunos lenguajes de plantillas son Smarty, PHPTAL o XTemplate.

Theme engine A fin de conseguir una mayor separación entre contenido, control y presentación, Drupal no incorpora directamente la gestión de temas dentro del núcleo del mismo, sino que delega esta gestión en módulos externos a los que accede a través de llamadas a funciones que éstos están obligados a implementar. Algunos de estos módulos a su vez son muy genéricos, y en vez de ofrecer un único tema, lo que ofrecen es funcionalidad para trabajar con ellos con plantillas. Este tipo de módulos reciben el nombre de engine theme, que traducido literalmente sería algo así como "motor de temas".

Typepad Un servicio de blogs alojados diseñado y pensado para bloggers profesionales y empresas. TypePad es un medio fácil y de costo razonable que permite crear una presencia importante en la web en poco tiempo. Para crear un blog en TypePad no se requiere ningún conocimiento de orden técnico. Un editor de texto WYSIWYG, un control eficaz anti-spam, capacidades de creación de notas vía móvil, plantillas de diseños alojadas, y un sistema multimedia enriquecido hacen posible la construcción de blogs profesionales en sólo algunos minutos.

Typeahead Es una característica de algunos campos de texto que permiten que el usuario escriba mientras el sistema le sugiere palabras que empiezan por los caracteres que ya ha escrito o bien que los contengan.

UML Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

URI Uniform Resource Identifier, Identificador Uniforme de Recurso, definido en RFC 2396 (Uniform Resource Identifiers: Generic Syntax). Algunos URI pueden ser URL, URN o ambos.

Un URI es una cadena corta de caracteres que identifica inequívocamente un recurso (servicio, página, documento, dirección de correo electrónico, enciclopedia, etc). Normalmente estos recursos son accesibles en una red o sistema.

Un URI consta de las siguientes partes:

* Esquema: nombre que se refiere a una especificación para asignar los identificadores, e.g. urn:, tag:, cid:. En algunos casos también identifica el protocolo de acceso al recurso, por ejemplo http:, mailto:, ftp:.

* Autoridad: elemento jerárquico que identifica la autoridad de nombres (por ejemplo //es.wikipedia.org).

* Ruta: Información usualmente organizada en forma jerárquica, que identifica al recurso en el ámbito del esquema URI y la autoridad de nombres (e.g. /wiki/Uniform_Resource_Identifier).

* Consulta: Información con estructura no jerárquica (usualmente pares "clave=valor") que identifica al recurso en el ámbito del esquema URI y la autoridad de nombres. El comienzo de este componente se indica mediante el carácter '?'.

* Fragmento: Permite identificar una parte del recurso principal, o vista de una representación del mismo. El comienzo de este componente se indica mediante el carácter '#'.

URL URL significa Uniform Resource Locator, es decir, localizador uniforme de recurso. Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.

Los URL fueron una innovación fundamental en la historia de la Internet. Fueron usadas por primera vez por Tim Berners-Lee en 1991, para permitir a los autores de documentos establecer hiperenlaces en la World Wide Web (WWW o Web). Desde 1994, en los estándares de la Internet, el concepto de URL ha sido incorporado dentro del más general de URI (Uniform Resource Identifier - Identificador Uniforme de Recurso), pero el término URL aún se utiliza ampliamente.

Web crawler *ver buscador web.*

Webplayer Reproductor web. Consiste en un reproductor hecho con HTML y otras tecnologías como CSS, Javascript o Flash, que se ejecuta desde un navegador de Internet.

Webservices Un servicio web (en inglés web service) es un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet.

Windows Media Player

Windows Media Player, Reproductor Multimedia de Windows o Reproductor de Windows Media (abreviado frecuentemente WMP) es un reproductor multimedia creado por la empresa Microsoft en 2003.

Se han lanzado varias versiones del reproductor. Para marzo de 2008, la versión 11 es la última existente, que se incluye con Windows Vista, existiendo también una versión para Windows XP.

WYSIWYG Es el acrónimo de *What You See Is What You Get* (en inglés, "lo que ves es lo que obtienes"). Se aplica a los procesadores de texto y otros editores de texto con formato (como los editores de HTML) que permiten escribir un documento viendo directamente el resultado final, frecuentemente el resultado impreso. Se dice en contraposición a otros procesadores de texto, hoy en día poco frecuentes, en los que se escribía sobre una vista que no mostraba el formato del texto, hasta la impresión del documento.

XHTML XHTML, acrónimo inglés de eXtensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto), es el lenguaje de marcado pensado para sustituir a HTML como

estándar para las páginas web. En su versión 1.0, XHTML es solamente la versión XML de HTML, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones, más estrictas, de XML. Su objetivo es avanzar en el proyecto del World Wide Web Consortium de lograr una web semántica, donde la información, y la forma de presentarla estén claramente separadas.

XML Siglas en inglés de Extensible Markup Language (Lenguaje de Marcas Extensible). Es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

Anexo III: Diagrama de clases completo

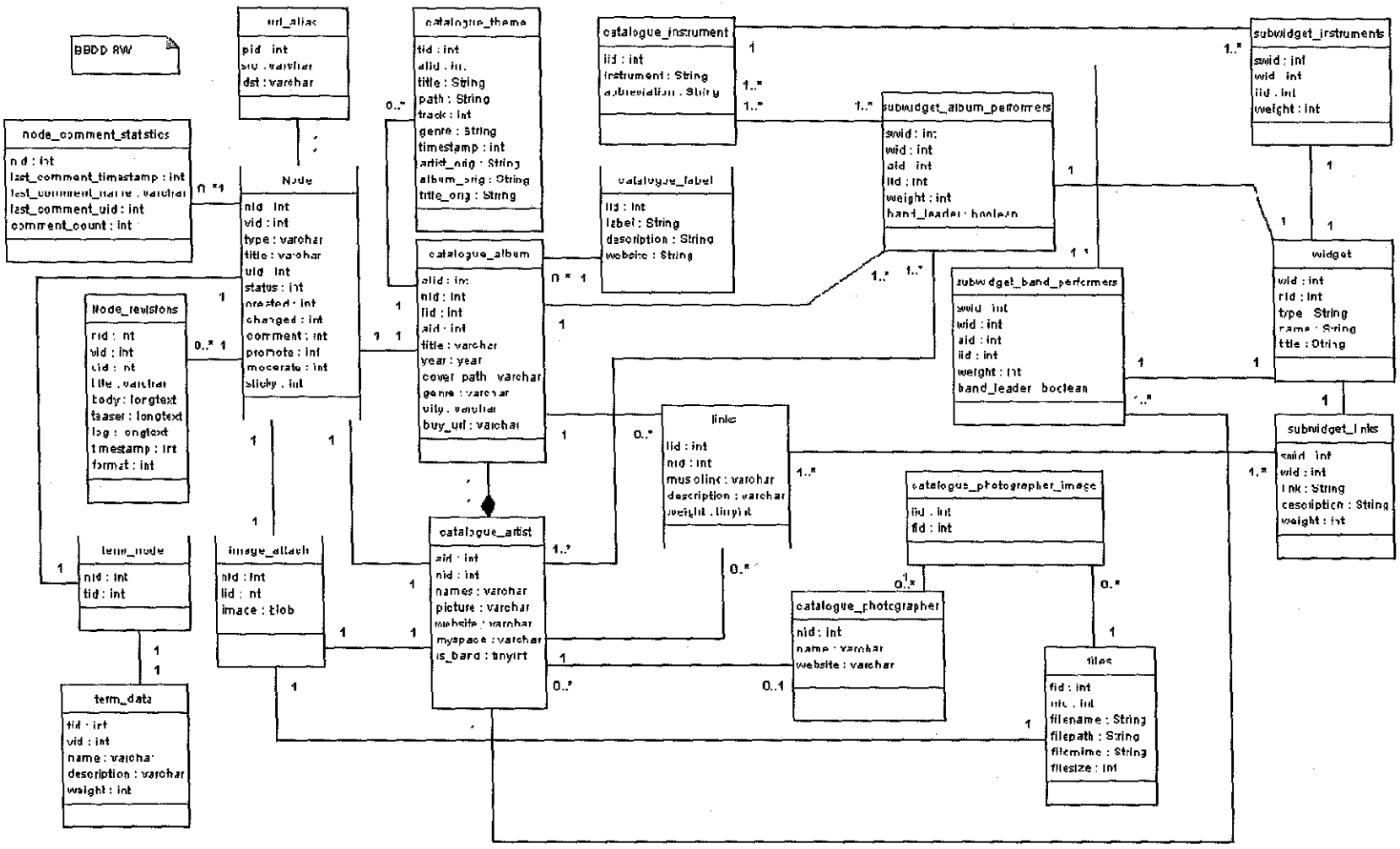


Figura 13.1: Diagrama de clases completo