Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Objetivos Generales

El portal *Core: Sistema Colaborativo para la Gestión de Conferencias* es una aplicación web que trata de facilitar la gestión y el mantenimiento de las distintas conferencias que se llevan a cabo en la comunidad RedIRIS.

En 1988 el *Plan Nacional de Investigación y Desarrollo*[1] puso en marcha un programa horizontal especial para la Interconexión de los Recursos InformáticoS (IRIS) de las universidades y centros de investigación. A partir de 1991, cuando se considera finalizada una etapa de promoción y lanzamiento, IRIS se tranforma en lo que es actualmente RedIRIS: la red académica y de investigación nacional que sigue siendo patrocinada por el Plan Nacional de I+D. En la actualidad, RedIRIS se integra como un departamento con autonomía e identidad propias en el seno de la entidad pública empresarial Red.es, adscrita al Ministerio de Industria, Turismo y Comercio.

RedIRIS organiza a lo largo del año distintas conferencias y congresos, en los que participan

[1] *El Plan Nacional de I+D+I constituye el eje estratégico de la política española de Investigación Científica, Desarrollo e Innovación Tecnológica para su periodo de aplicación. Con este Plan se busca contribuir a la generación de conocimiento, de manera que esté al servicio de la sociedad y se logre así la mejora del bienestar.*

personas de diferentes nacionalidades por lo que se ha tenido en cuenta el soporte para la Internacionalización de la aplicación como base fundamental del proyecto.

Este proyecto se ha realizado en colaboración con la asociación TERENA. TERENA ofrece un foro donde colaborar, innovar y compartir conocimientos con el fin de fomentar el desarrollo de la tecnología de Internet, la infraestructura y los servicios a ser utilizados por la comunidad de la investigación y la educación.

La aplicación que aquí se presenta, está escrita principalmente en el lenguaje de programación PHP (PHP Hipertext Preprocessor), destacando además el *Script de Generación de Eventos* escrito en bash.

En este documento se detalla en primer lugar, qué es la Internacionalización (capítulo 2), en qué se diferencia de la Localización, por qué es necesaria y cómo dar soporte a una aplicación.

En el siguiente capítulo, nos adentraremos en los requisitos necesarios para montar la aplicación, profundizando en las tecnologías utilizadas en el proyecto y el por qué de las mismas.

En el capítulo 4 expondremos las distintas conclusiones a las que hemos llegado tras la realización de este proyecto, así como el presente y el futuro del mismo.

Por último, como anexos se encuentran el manual de instalación de la aplicación y el manual de usuario.

1.2. Antecedentes

A pesar de la amplia utilización de sistemas telemáticos para realizar el control y difusión de conferencias, no existe ningún producto, ni comercial, ni libre, que haya destacado en esta labor.

Como norma general, las instituciones o empresas que afrontan la tarea de realizar un congreso, conferencia o evento similar, optan por la creación de software adaptado a sus necesidades. Esto es debido a la carencia de software de calidad y flexible a las necesidades particulares de cada usuario.

Aún así, podemos encontrar productos dedicados a la administración y publicación de conferencias. Su creación generalmente está asociada a la necesidad de dar soporte a algún evento o similar. Podemos destacar los siguientes:

- **YaCoMaS: Yet Another Conference Management System**

  Nace como el sistema gestor de conferencias para el evento "Festival GNU/Linux y Software Libre" (http://www.fsl.udg.mx). Es capaz de controlar todo el flujo de información que genera una conferencia: registro de ponentes, aceptación de articulos y talleres, sistema de administración etc.

- **Joomla Confereces Abstract Submission Component**

  En este caso nos encontramos con un módulo para el CMS[2] *Joomla* que realiza la tarea de envío y aceptación de artículos. La utilización de este software está ligada al uso del gestor *Joomla* y su última actualización data de octubre del 2006.

- **COD: Conference Organizing Distribution**

  Existe un módulo para la gestión de conferencias para otro CMS, en este caso para el popular *Drupal*. Este módulo aún está en desarrollo pero cuenta con el apoyo de entidades como la NASA o la Fundación Apache para incluirlo como su sistema de conferencias.

[2] Acrónimo de *Content Management System*, Sistema de Gestión de Contenidos. Permite la creación y administración de contenidos principalmente en páginas web.

De los paquetes de software mencionados con anterioridad hemos de citar COD como el más prometedor, pero aún carece de la estabilidad y robustez para plantear su instalación como una alternativa seria.

Además hemos de destacar que ninguno de los sistemas anteriores está provisto de capacidad de internacionalización, requisito indispensable en algunos eventos, el cual ha sido incluido como base en el diseño de *Core*.

Cabe destacar también, que *Core* es la única que integra herramientas para la creación de distintos eventos, trabajando de forma colaborativa tanto moderadores, como ponentes y organizadores.

Capítulo 2

Internacionalización

2.1. Introducción

La Web se ha convertido a lo largo de los años en la principal herramienta de disfusión de información para una audiencia muy variada y de gran tamaño que requiere de un procesamiento de información sencillo.

La información que se muestra al usuario está compuesta de partes diferentes que han de trabajar en conjunto de forma coordinada para que la información sea accesible y universal, es decir, estas partes que integran la Web han de funcionar bajo cualquier circunstancia, en cualquier país, con cualquier idioma y cultura.

Por este motivo, podríamos definir la Internacionalización como un proceso a través del cual se van a diseñar sitios Web adaptables a diferentes idiomas y regiones sin necesidad de realizar grandes cambios en el código fuente de la aplicación.

La utilización de formatos y protocolos que no establezcan fronteras a los diferentes idiomas, siste-

mas de escritura, códigos y otras convenciones locales, es esencial para hablar de Internacionalización en un sitio Web.

La Internacionalización es conocida, además, como I18N. Esta abreviatura se debe a que hay 18 letras entre la primera letra, I, y la última, N.

El W3C[1] inicia la Actividad de Internacionalización en un intento por asegurar que estos formatos y protocolos puedan utilizarse de forma universal en todos los idiomas y en todos los sistemas de escritura. Por lo tanto, la creación de un sitio Web internacional permite garantizar su utilización universal incluyendo todos los idiomas y culturas.

Usuarios de distintos países y con diferentes culturas necesitan servicios adaptados correctamente para procesar información usando su idioma de origen, su sistema de escritura, su sistema de medida, sus calendarios y otras reglas y convenciones culturales.

La especificación de un conjunto particular de convenciones culturales es importante para que un sitio Web procese correctamente la información que intercambia con el usuario. Hay muchas preferencias que un sitio Web debe ofrecer para que sea considerado fácil de usar y aceptable por los usuarios a nivel mundial.

Al existir un variado número de preferencias de circunstancias culturales y de idioma, es importante utilizar identificadores basados en el idioma y lugar como referente para recoger información sobre las preferencias de los usuarios. Por ejemplo HTML usa el atributo *lang* para indicar el idioma de segmentos de contenido.

[1] Abreviatura de *World Wide Web Consortium*. Consorcio internacional que produce estándares para la *World Wide Web*. Está dirigida por Tim Berners-Lee, el creador original de *URL* (*Uniform Resource Locator*, Localizador Uniforme de Recursos), *HTTP* (*HyperText Transfer Protocol*, Protocolo de Transferencia de HiperTexto) y *HTML* (Lenguaje de Marcado de HiperTexto) que son las principales tecnologías sobre las que se basa la Web.

Otra cuestión importante en el área de Internacionalización es hacer frente a los problemas de codificación en documentos (X)HTML. Se trata de un método para transformar una secuencia de bits en una secuencia de caracteres. Los servidores envían documentos HTML a los clientes (navegadores) como cadenas de bits; a su vez, los clientes los interpretan como una secuencia de caracteres.

El método de conversión va desde una simple transformación hasta algoritmos y esquemas complejos. Una forma de solucionar problemas de codificación es servir todas las páginas en un conjunto de caracteres, por ejemplo UTF-8, un conjunto de longitud variable que utiliza grupos de bits para representar el estándar Unicode para el alfabeto de varios idiomas. UTF-8 puede representar los caracteres de una amplia variedad de idiomas. Los navegadores envían de vuelta los datos en la misma codificación que la página que contiene la información.

En un principio la Actividad de Internacionalización promovió el uso de Unicode/ISO-10646 para identificar y describir caracteres. Unicode se ha usado de forma generalizada ya que asegura que los datos puedan ser manejados uniformemente y de forma que puedan ser mostrados y, en definitiva, manipulados sin miedo a alteraciones.

Conceptos de Internacionalización y Localización

Por Internacionalización entendemos la operación por la que un programa o conjunto de programas se convierten en un paquete, el cual es capaz de soportar múltiples idiomas. Este es un proceso de generalización, es decir, dar soporte a una aplicación para que pueda ser localizada.

Se pueden usar diversas técnicas para internacionalizar una aplicación, algunas de ellas normalizadas. GNU Gettext ofrece uno de esos estándares.

La Localización es un proceso de particularización, por el que dado un conjunto de programas

previamente internacionalizados, se les da toda la información necesaria para que pueda adaptarse a un idioma y cultura específicos. El entorno de programación pone varias funciones a disposición de los programadores que permiten esta configuración en tiempo de ejecución. La descripción formal del conjunto específico de hábitos culturales de algunos países, junto con todas las traducciones asociadas, se denomina localidad, en inglés *locale*, para un idioma y país específicos.

Al igual que la Internacionalización con I18N, la Localización es conocida también como L10N. Esta abreviatura se debe a que hay 10 letras entre la primera, L y la última, N.

En resumen, la Internacionalización es el proceso que hay que seguir para que una aplicación pueda ser *localizada*.

2.2. GNU Gettext

Usualmente, las aplicaciones están escritas y documentadas en inglés. y se usa el inglés en tiempo de ejecución para interactuar con el usuario. Esto es cierto no solo desde dentro de GNU[2], sino además en muchas aplicaciones de software propietario y libre.

El uso de un lenguaje común es más útil para la comunicación entre desarrolladores, mantenedores y usuarios provenientes de todos los países. Por otro lado, la mayoría de las personas se sienten menos cómodas trabajando en inglés que con su propio lenguaje, y prefieren por tanto utilizar su lengua materna en el trabajo del día a día.

GNU Gettext ofrece a los programadores, traductores e incluso a los usuarios, un amplio conjunto de herramientas integradas y documentación. Específicamente, GNU Gettext es un conjunto de herramientas que proporcionan un entorno de desarrollo para producir mensajes multi-lenguaje. Estas

[2]Acrónimo recursivo de "*GNU is Not Unix*". El proyecto GNU fue creado en 1983 por Richard Stallman, con el objetivo de crear un sistema operativo completamente libre.

herramientas incluyen una serie de convenciones sobre cómo deben estar escritas las aplicaciones para soportar catálogos de mensajes, una cierta organización de la estructura de directorios y archivos para los propios catálogos de mensajes, una biblioteca que le de soporte a la recuperación de los mensajes traducidos y algunos programas que gestionan de diversas formas los conjuntos de cadenas traducibles, o las cadenas ya traducidas.

Para una total distribución multilingüe, hay muchas cosas para traducir que van más allá de los mensajes de salida.

La traducción es sólo un aspecto más de la Localización. Otros aspectos de Internacionalización son los servicios del sistema, que se manejan en la librería *libc*[3] de GNU.

GNU Gettext trabaja con dos tipos de ficheros. Los ficheros `.po` (*Portable Object*) y los ficheros `.mo`(*Machine Object*) los cuales se explicarán más adelante. Este paradigma está inspirado en el estándar *NLS de Uniforum*[4], implementado por Sun MicroSystems en su sistema operativo Solaris.

Son muchos los atributos que se necesitan para definir un país de las convenciones culturales. Estos atributos incluyen junto a la lengua del país, el formato de la fecha y la hora, la representación de los números, los símbolos de moneda, etc. Estos atributos se denominan normas de localización del país. La localización representa los conocimientos necesarios para apoyar los atributos nativos del país.

[3]Contiene las librerías estándar que se utilizan en casi todos los programas del sistema. Incluye entre otras la bxiblioteca C estándar y la biblioteca estándar de matemáticas.
[4]*Native Language Support*, Soporte de Lenguaje Nativo. http://uniforum.org

2.2.1. Introducción a los ficheros .PO

El conjunto de herramientas de GNU Gettext ayuda a los programadores y traductores en la producción, actualización y uso de ficheros de traducción, principalmente los ficheros `.po`, ficheros de texto editables. En esta sección se explicará el formato de los mismos.

Un fichero `.po` se compone de muchas entradas. Cada una de ellas es una relación entre la cadena original sin traducir y su correspondiente traducción. Todas las entradas en un determinado fichero `.po` generalmente se refieren a un único proyecto, y todas las traducciones se expresan en una sola lengua de destino.

Un fichero `.po` tiene la siguiente estructura:

```
1 <espacio en blanco>
2 #Comentarios del traductor
3 #.Comentarios extraídos
4 #: Referencia al fichero
5 #, Bandera...
6 #| msgid Cadena previa sin traducir
7 msgid Cadena-sin-traducir
8 msgstr Cadena-traducida
```

Figura 2.1: Estructura de un fichero `.po`

La estructura general de un fichero `.po` debe ser bien entendida por el traductor. Una entrada simple, donde queremos traducir una cadena del inglés al español de un fichero llamado `ejemplo.c` en la línea 1 sería:

```
1 #: ejemplo.c:1
2 msgid "Hello world!"
3 msgstr "Hola mundo!"
```

Figura 2.2: Ejemplo de una entrada en un fichero `.po`

Las entradas comienzan con algunos espacios en blanco de manera opcional. Por lo general, cuando se genera a través de las herramientas de GNU Gettext, exactamente existe una línea en blanco entre diferentes entradas.

A continuación le siguen los comentarios, los cuales empiezan siempre con el carácter '#'. Existen básicamente dos tipos de comentarios: aquellos dónde al carácter '#' les sigue un espacio en blanco y los que no lo tienen.

Los comentarios que tienen justo detrás un espacio en blanco, son comentarios introducidos y mantenidos por el traductor.

Los comentarios que no tienen un espacio en blanco, son observaciones que se crean y se mantienen automáticamente por las herrmanientas de gettext. Existen varios tipos:

- `#.`: Contienen observaciones dadas por el programador, dirigidas al traductor. Estos comentarios se obtuvieron mediante la herramienta `xgettext` del código fuente del programa.
- `#:`: Contienen referencias al código fuente del programa; por ejemplo, de qué fichero se ha extraído la cadena y en qué línea.
- `#,`: Estas líneas son especiales, ya que no son ignoradas completamente por los programas como los comentarios lo son en general. La lista separada de banderas es usada por el programa `msgfmt` para dar al usuario algunos mensajes de diagnóstico mejores.

  Actualmente existen dos formas de banderas distintas:

  - `fuzzy`, que puede ser generada por `msgmerge` o por el traductor e indica que la cadena `msgstr` puede no ser una traducción correcta.
  - `c-format` y `no-c-format`, que sólo pueden ser introducidas por el programa `xgettext`, las cuales indican que las cadenas (sin traducir y las traducidas) deben ser cadenas en formato C o no, respectivamente.

    Al igual que para C, estas banderas existen para otros lenguajes de programación como Python, Lisp, Java, C#, Shell, Perl, PHP, y otros muchos.
- `#|`: Contiene la anterior cadena sin traducir para la que el traductor dio una traducción.

Todos los comentarios, ya sean de la naturaleza que sean , son opcionales.

Después de los espacios en blanco y los comentarios, las entradas muestran dos series de caracteres. En primer lugar la cadena sin traducir tal y como aparece en el código original y, a continuación, la traducción de esta cadena.

La cadena original se introduce por la palabra clave `msgid`, y la traducción por `msgstr`. Las dos cadenas, sin traducir y traducida, se citan de diversas maneras, mediante delimitadores de cadena `"` y delimitadores de escape \. Pero el traductor no tiene porqué prestar atención a esto.

Las cadenas `msgid`, así como los comentarios automáticos, son producidos y gestionados por otras herramientas de GNU Gettext, y el modo PO no proporciona los medios necesarios para que el traductor pueda alterarlos. Lo único que puede hacer es suprimirlos, pero esto conlleva la supresión de toda la entrada. Por otro lado, las cadenas `msgstr`, así como los comentarios del traductor, son realmente signficativos para este, de manera que el modo PO proporciona al traductor pleno control sobre éstos.

Además es posible tener entradas en las que se especifica un contexto.

```
1  <espacio en blanco>
2  # Comentarios de traductor
3  #. Comentarios extraídos del fichero de entrada
4  #: Referencia
5  #, Banderas
6  #| msgctxt contexto-anterior
7  #| msgid cadena-anterior-sin-traducir
8  msgctxt contexto
9  msgid cadena-sin-traducir
10 msgstr cadena-traducida
```

Figura 2.3: Estructura de un fichero `.po` con contexto

El contexto sirve para que no haya cadenas ambiguas. Es posible tener varias entradas con la misma cadena sin traducir en un fichero `.po`, a condición de que cada una tenga un contexto diferente. Hay que tener en cuenta que una cadena vacía y un contexto `msgctxt` ausente no significan lo mismo.

Un tipo diferente de entradas se utiliza para las traducciones de la forma plural.

```
1  <espacio en blanco>
2  # Comentarios de traductor
3  #. Comentarios extraídos del fichero de entrada
4  #: Referencia
5  #, Banderas
6  #| msgid cadena-singular-anterior-sin-traducir
7  #| msgid\_plural cadena-plural-anterior-sin-traducir
8  msgid cadena-singular-sin-traducir
9  msgid\_plural cadena-plural-sin-traducir
10 msgstr[0] cadena-traducida-caso-0
11 ...
12 msgstr[N] cadena-traducida-caso-N
```

Figura 2.4: Estructura de un archivo `.po` con cadenas en plural

Aquí también se podría especificar un contexto antes de `msgid`, como se explicó anteriormente.

La `cadena-anterior-sin-traducir` es insertada opcionalmente por el programa `msgmerge`, a la vez que marca el mensaje como `fuzzy`. Esto ayuda al traductor a ver los cambios que fueron realizados por los desarrolladores en la cadena sin traducir.

A continuación se muestra un ejemplo del inglés al alemán de este tipo de traducciones.

```
1  msgid ""
2  msgstr ""
3  "Project-Id-Version: pigs\n"
4  "Report-Msgid-Bugs-To: \n"
5  "POT-Creation-Date: 2003-10-23 04:50+0200\n"
6  "PO-Revision-Date: 2003-11-01 23:40+0100\n"
7  "Last-Translator: Danilo Segan <danilo@kvota.net>\n"
8  "Language-Team: Serbian (sr) <danilo@kvota.net>\n"
9  "MIME-Version: 1.0\n"
10 "Content-Type: text/plain; charset=UTF-8\n"
11 "Content-Transfer-Encoding: 8bit\n"
12 #"Plural-Forms: nplurals=2; plural=n != 1;\n"
13 #: pigs.php:19
14 msgid "" "This is how the story goes.\n"
15 msgstr "" "Und so geht die Geschichte.\n"
16 #: pigs.php:21
17 #, php-format
18 msgid "%d pig went to the market\n"
19 msgid_plural "%d pigs went to the market\n"
20 msgstr[0] "%d Schwein ging zum Markt\n"
21 msgstr[1] "%d Schweine gingen zum Markt\n"
```

Figura 2.5: Ejemplo de un fichero `.po` con traducciones de cadenas en plural

Para cada una de las cadenas traducidas y sin traducir, se respeta la sintaxis del lenguaje para una

cadena de caracteres, incluyendo las comillas y las secuencias de escape. Cuando llegue el momento de escribir varias líneas, no se deben utilizar caracteres de escape de salto de línea, como puede ser \n. Basta con poner las frases separadas por comillas seguidas una de la otra.

```
1  msgid ""
2  "Here is an example of how one might continue a very long string\n"
3  "for the common case the string represents multi-line output.\n"
4  msgstr ""
5  "Aquí tenemos un ejemplo de cómo puede uno continuar una cadena muy larga\n"
6  "para el caso común de que la cadena representa una salida multi-línea.\n"
```

Figura 2.6: Ejemplo de traducción de multi-líneas en un fichero `.po`

En este ejemplo, la cadena vacía se utiliza para permitir una mejor adaptación de la H de la palabra *Here* y la f de la palabra *for*. Pero la clave en este ejemplo está en `msgid` seguida de tres cadenas destinadas a ser concatenadas. Concatenando la cadena vacía no cambia el resultado general de la cadena, pero es una manera de cumplir con la necesidad de `msgid` a ser seguido por una cadena sin dejar de mantener la presentación de múltiples líneas justificadas a la izquierda, ya que se trata de una disposición más limpia del texto.

Sin embargo, la cadena vacía se podría haber omitido, pero sólo si la siguiente cadena se traslada justo a continuación de `msgid`. No es estrictamente necesario utilizar el carácter de escape \n para separar las líneas, ya que esta separación puede llevarse a cabo después de cualquier carácter, pero así el ejemplo queda más claro.

Hay que distinguir cuidadosamente entre el final de las líneas marcadas como "\n", dentro de las comillas, que forma parte de la cadena representada, y el final de las línes del fichero `.po`, fuera de las comillas, lo que no tiene ninguna incidencia sobre las cadenas que se quieren representar.

En resumen, las cadenas de salida, las líneas en blanco y las observaciones pueden ser usadas libremente. Los comentarios deben comenzar por el carácter #, donde los comentarios de los traductores

son seguidos por algunos espacios en blanco, y los que no son seguidos por estos, serán generados y gestionados por herramientas de GNU Gettext.

2.2.2. Preparando las cadenas a traducir

Antes de que las cadenas puedan ser marcadas para ser traducidas, a veces deben ser ajustadas. Por lo general, la preparación de una cadena se realiza justo antes del marcado.
Para preparar una cadena debe tenerse en cuenta lo siguiente:

1. Uso correcto del inglés.

   Hay que evitar el uso de abreviaturas o atajos, ya que puede conducir a error al traductor, de manera que las traducciones realizadas no sean correctas.

2. Oraciones completas.

   Esto es muy importante, ya que en muchos idiomas, la declinación de algunas palabras en la frase, depende del género o el número del resto de palabras en la frase.

   Usualmente en otros idiomas, hay más interdependencias entre las palabras que en inglés. La consecuencia es que pedir a un traductor para traducir medias frases y a continuación, la combinación de éstas por medio de frases de concatenación, no funciona. Esta es la razón por la que los traductores necesitan frases completas.

3. Dividir en párrafos.

   Las cadenas deben limitarse a un sólo párrafo. La cadena a traducir no debe ser de más de 10 líneas. La razón es que, cuando se producen cambios en la cadena a traducir, el traductor se enfrenta a la tarea de traducir toda la cadena completa. Tal vez una sola palabra haya cambiado, pero si la cadena es demasiado larga, será un trabajo tedioso, de manera que el traductor tendrá que revisar la cadena entera hasta dar con la palabra que haya cambiado.

4. Utilizar formato de cadenas en lugar de concatenación de cadenas.

A veces, para la formación de cadenas se utilizan funciones de concatenación, por ejemplo:

```
strcpy (s, "Replace ");
strcat (s, object1);
strcat (s, "with ");
strcat (s, object2);
strcat (s, "?");
```

Figura 2.7: Ejemplo en lenguaje C de concatenación de cadenas

Con el fin de presentar la cadena completa al traductor, y además porque en algunos idiomas el traductor podría querer cambiar el orden de las variables *object1* y *object2,* es necesario utilizar el formato de cadenas:

```
sprintf (s, "Replace \%s with \%s?", object1, object2);
```

Figura 2.8: Ejemplo en lenguaje C de cadena con formato

2.2.3. Creación de una plantilla .PO: *xgettext*

Después de preparar los ficheros fuente, el programador crea un fichero PO plantilla. En esta sección se explica cómo utilizar `xgettext` para este cometido.

`xgettext` crea un fichero llamado *`nombredominio.po`*. A continuación, debemos renombrar este fichero a *`nombredominio.pot`*. ¿Por qué no es `xgettext` quien se encarga directamente de crear este fichero? La respuesta es, por razones históricas. Cuando `xgettext` fue especificado, la distintición entre un fichero PO y un fichero plantilla PO era exclusivamente la bandera *fuzzy* y la extensión `.pot` no estaba en uso en ese momento.

Invocación de `xgettext`

El programa `xgettext` se encarga de extraer las cadenas marcadas de los ficheros de entrada dados. Su invocación es la siguiente:

xgettext [*options*] [*inputfile*]

Veamos las opciones más importantes que podemos invocar con `xgettext`.

Localización de los ficheros de entrada:

- `-f` *`file`*`, --files-from=`*`file`*: Lee los nombres de los ficheros de entrada del fichero *file*.
  Si el fichero de entrada es '-', se leerá de la entrada estándar.
- `-D` *`directory`*`, --directory=`*`directory`*: Añade un directorio a la lista de directorios. Los ficheros fuentes serán buscados en relación a esta lista de directorios, sin embargo, el fichero `.po` resultante será escrito con relación al directorio actual.

Localización de los ficheros de salida:

- `-d` *`name`*`, --default-domain=`*`name`*: Utilizará como fichero de salida *name.po* en lugar de *messages.po*, el nombre por defecto.
- `-o` *`name`*`, --output=`*`file`*: Escribirá la salida en el fichero especificado, en lugar de *name.po* o de *messages.po*.
  Si el fichero de salida es '-' o '/dev/stdout/', el resultado será escrito en la salida estándar.
- `-p , --output-dir=`*`dir`*:Los archivos de salida serán colocados en el directorio *dir*.

Elección del lenguaje de los ficheros de entrada

- `-L` *name*`, --language=`*name*: Especifica el lenguaje de los ficheros de entrada. Los lenguajes soportados son C, C++, OjectiveC, PO, Python, Lisp, EmacsLisp, librep, Scheme, Smalltalk, Java, JavaProperties, C#, awk, YCP, Tcl, Perl, PHP, GCC-source, NXStringTable, RST y Glade.
- `-C, --c++`: Esto es simplemente una abreviatura de `--language=C++`.

Interpretación de los ficheros de entrada:

- `--from-code=`*name*: Especifica la codificación de los ficheros de entrada. Esta opción es necesaria solamente si algunas cadenas sin traducir, o sus correpondientes comentarios contienen caracteres que no son ASCII.
  Por defecto, se supone que los ficheros de entrada están en ASCII.

Modo de operación:

- `-j, --join-existing`: Une los mensajes con un fichero ya existente.
- `-x` *file*`, --exclude-file=`*file*: Las entradas especificadas en el fichero *file* no son extraídas. Para que esta opción pueda llevarse a cabo, el fichero debe tener extensión PO o POT.
- `-c [`*tag*`], --add-comments[=`*tag*`]`: Coloca los bloques de comentarios con la etiqueta *tag* en los ficheros de salida.

Opciones específicas del lenguaje:

- `-a, --extract-all`: Extrae todas las cadenas. Esta opción tiene efecto con la mayoría de los lenguajes (C, C++, ObjectiveC, Shell, Python, Lisp, EmacsLisp, librep, Java, C#, awk, Tcl, Perl, PHP, GCC-Source y Glade).

- `-k` *`keywordspec`*`, --keyword[=`*`keywordspec`*`]`: El programa `gettext` posee distintos alias (palabras clave, *keywords*) que pueden ser llamados en lugar de `gettext` en los ficheros fuente de nuestra aplicación.

  Estos alias (que se especifican con *keywordspec*) que se esperan, a menos que hayan sido explícitamente deshabilitados son dependientes del lenguaje. Algunos ejemplos son:

  - **Java**: *GettextResource.gettext:2, GettextResource.ngettext:2,3, ngettext:1,2 y getString.*
  - **C#**: *GetString y GetPluralString:1,2.*
  - **Perl**: *%gettext, $gettext, dgettext:2, dcgettext:2, ngettext:1,2, dngettext:2,3, dcngettext:2,3 y gettext_ noop.*
  - **PHP**: *_ , dgettext:2, dcgettext:2, ngettext:1,2, dngettext:2,3 y dcngettext:2,3.*
  - **Glade**: *label, title, text, format, copyright, comments, preview_ text y tooltip.*

Detalles de la salida:

- `--force-po`: Escribe siempre un fichero de salida, incluso si no se han encontrado cadenas que extraer.
- `-i, --indent`: Escribe el fichero `.po` utilizando un estilo sangrado.
- `--no-location`: No escribe los comentarios '#: nombre-fichero:línea' que indican dónde se encontró la cadena extraída.
- `-n, --add-location`: Genera los comentarios '#: nombre-fichero:línea' en el fichero de salida. Esta opción se ejecuta por defecto.
- `-w` *`number`*`, --width=`*`number`*: Fija el ancho de página. Las cadenas largas en los archivos de salida serán dividias en múltiples líneas, con el fin de garantizar que la anchura de cada una de las líneas (igual al número de columnas de la pantalla) es menor o igual al número dado.

- `--no-wrap`: Con esta opción, aquellas líneas cuya anchura sea superior a la anchura de la página de salida, no serán divididas en varias líneas.

- `-s, --sort-output`: Genera la salida ordenada alfabéticamente. Hay que tener en cuenta que si esto es así, al traductor le será más difícil saber el contexto del mensaje.

- `-F, --sort-by-file`: Ordena la salida en función de la localización de los ficheros.

- `--omit-header`: No escribe la cabecera '`msgid`' de las entradas. Esto es útil para propósitos de prueba. Con esta opción, se garantiza que dos llamadas a `xgettext` con las mismas opciones y los mismos ficheros, producen iguales resultados.

- `--copyright-holder=`*`string`*: Establece los derechos de autor del fichero de salida. La cadena *string* debe ser el titular de los derechos de autor del paquete. Debemos tener en cuenta, que las cadenas `msgstr` extraídas de las fuentes del paquete, pertenecen a estos derechos de autor.

  Los traductores están a la espera de transferir o renunciar a los derechos de autor para su traducción, con el fin de que los mantenedores de los paquetes puedan distribuirlos sin riesgo jurídico. Si la cadena *string* está vacía, los derechos de autor se marcan de dominio público, obligando a los traductores a renunciar a sus derechos de autor, para que de nuevo, los mantenedores puedan distribuir las traducciones sin riesgo jurídico.

  El valor por defecto de esta cadena es 'Free Software Foundation, Inc.', simplemente debido a que `xgettext` se utilizó por primera vez en el proyecto GNU.

- `--foreign-user`: Omite la cadena 'Free Software Foundation, Inc.' en el fichero de salida. Esta opción es equivalente a '`--copyright-holder=''`'. Esto puede ser útil para aquellos paquetes que no pertenezcan al proyecto GNU, pero quieran que sus traducciones sean de dominio público.

- `--msgid-bugs-address=`*`email@address`*: Establece la dirección a la que hay que dirigirse para enviar informes de error. Es la dirección de correo electrónico o URL a la que los traductores

deberán informar de los errores producidos en las cadenas sin traducir.Por ejemplo:

- Cadenas que no son frases completas.
- Cadenas ambiguas que pueden llevar a confusión y necesitan un contexto para poder ser entendidas.
- Cadenas que pueden suponer una notación inválida sobre la fecha, el tiempo o el dinero.
- Problemas en las formas plurales.
- Escritura incorrecta del inglés.
- Formato incorrecto de las cadenas.

Esta dirección puede ser una dirección de correo electrónico personal o de una lista de correo donde los traductores pueden escribir sin necesidad de estar suscritos a ella. También puede ser la URL de una página web donde los traductores pueden ponerse en contacto con los programadores.

El valor por defecto es la cadena vacía, lo que significa que los traductores no tendrán ninguna pista para saber a donde entregar los informes de error. Esta opción es muy importante.

- `-m [`*`string`*`], --msgstr-prefix[=`*`string`*`]`: Utiliza la cadena *string* o la cadena vacía si no se especifica, como prefijo para las entradas `msgstr`.
- `-M [`*`string`*`], --msgsrt-sufix[=`*`string`*`]`: Utiliza la cadena *string* o la cadena vacía si no se especifica, como sufijo para las entradas `msgstr`.

Salida informativa:

- `-h, --help`: Muestra por pantalla todas las opciones disponibles para ejecutar con `xgettext` y finaliza el programa.
- `-V, --version`: Muestra por pantalla la versión instalada de *xgettext* y sale del programa.

2.2.4. Creación de un fichero .PO: *msginit*

Al comenzar una nueva traducción, el traductor crea un fichero llamado *lang*.po (donde *lang* es el código del lenguaje al que se está traduciendo), como una copia del fichero *messages.pot*, plantilla creada con las modificaciones comentadas anteriormente (comentarios iniciales y cabecera de entrada).

```
1  msgid ""
2  msgstr ""
3  "Project-Id-Version: PACKAGE VERSION\n"
4  "Report-Msgid-Bugs-To: \n"
5  "POT-Creation-Date: 2007-11-03 20:19+0100\n"
6  "PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
7  "Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
8  "Language-Team: LANGUAGE <LL@li.org>\n"
9  "MIME-Version: 1.0\n"
10 "Content-Type: text/plain; charset=<charset>\n"
11 "Content-Transfer-Encoding: 8bit\n"
```

Figura 2.9: Ejemplo de comentarios iniciales y cabecera de un archivo .po antes de ser modificado

La forma más fácil de inicializar los valores, es haciendo uso del programa `msginit`. Por ejemplo:

```
1  $ cd PACKAGE-VERSION
2  $ cd po
3  $ msginit
```

Figura 2.10: Inicialización de comentarios y cabeceras de forma automática mediante msginit

Otra alternativa es modificar esta información manualmente. Para esto, copiamos el fichero *messages.pot* a *lang.po* y a continuación editamos los comentarios iniciales y la cabecera.

Invocación del programa `msginit`

El programa `msginit` crea un nuevo fichero `.po`, inicializando la metainformación[5] con valores tomados del entorno del usuario.

`msginit [`*option*`]`

Figura 2.11: Invocación del programa msginit

[5]La metainformación es información acerca de la información. El término metainformación hace referencia a cualquier dato que se use para añadir identificación, descripción y localización a un recurso electrónico en la red. En nuestro caso, información sobre nuestro fichero `.po`. Toda la cabecera de nuestro fichero es metainformación.

A continuación, al igual que hicimos con xgettext, analizaremos las distintas opciones de `msginit`, para así poder hacer un buen uso del mismo, lo que nos ahorrará bastante trabajo.

Localización del fichero de entrada:

- `-i` *`inputfile`*`, --input=`*`inputfile`*: El fichero de entrada debe tener la extensión `.pot`.
  Si no se especifica ningún fichero de entrada, se busca en el directorio actual un fichero con dicha extensión. Si se especifica '-', se lee de la entrada estándar.

Localización del fichero de salida:

- `-o` *`file`*`, --output=`*`file`*: Se escribirá en el fichero *file* especificado.
  Si no se especifica ningún fichero de salida, se generará uno cuyo nombre dependerá de la opción '`--locale`' o de la configuración local del usuario. Si se especifica '-', el resultado de la ejecución se mostrará en la salida estándar.

Detalles de los ficheros de salida:

- `-l` *`ll_CC`*`, --locale=`*`ll_CC`*: Fija la información de localización. *ll* corresponde al código del lenguaje, mientras que *CC* corresponde al código del país. Con el comando '`locale -a`' podremos ver todos los *locale* que tengamos instalados en el sistema. Por defecto, la información de localización será la del entorno del usuario del sistema.
  Más adelante se tratará esto más detenidamente.

- `-w` *`number`*`, --width=`*`number`*: Esta opción es equivalente a la opción '`--width=`*`number`*' del comando `xgettext` explicado en la sección anterior.

- `--no-wrap`: Esta opción es equivalente a la opción '`--no-wrap`' del comando `xgettext` explicado en la sección anterior.

Salida informativa:

- `-h, --help`: Muestra por pantalla todas las opciones disponible a ejecutar con el comando `msginit`.

- `-V, --version`: Muestra la versión instalada del programa y lo finaliza.

Completando la cabecera de entrada

Los comentarios iniciales *"SOME DESCRIPTIVE TITLE*", "YEAR" y "FIRST AUTHOR <EMAIL@ADDRESS>, YEAR", deben ser sustituidos por información del paquete. Esto se puede hacer en cualquier editor de texto, como VIM o Emacs.

Existen otros editores especiales para ficheros `.po`, como GTranslator[6]

```
1  # SOME DESCRIPTIVE TITLE.
2  # Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
3  # This file is distributed under the same license as the PACKAGE package.
4  # FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
5  #
6  #, fuzzy
7  msgid ""
8  msgstr ""
9  "Project-Id-Version: PACKAGE VERSION\n"
10 "Report-Msgid-Bugs-To: \n"
11 "POT-Creation-Date: 2007-11-28 12:46+0100\n"
12 "PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
13 "Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
14 "Language-Team: LANGUAGE <LL@li.org>\n"
15 "MIME-Version: 1.0\n"
16 "Content-Type: text/plain; charset=CHARSET\n"
17 "Content-Transfer-Encoding: 8bit\n"
```

Figura 2.12: Cabecera de un archivo .po

Es necesario que rellenemos los siguientes campos:

- `Project-Id-Version`: Nombre y versión del paquete.
- `Report-Msgid-Bugs-To`: Este campo se rellena por el programa *xgettext* con la opción '`--msgid-bugs-address=email@address`'. Contiene la dirección de correo electrónico o la URL a donde se pueden enviar los errores producidos en las cadenas sin traducir.

[6]Herramienta enmarcada dentro del Translation Project para la Internacionalización del Software Libre. Puede emplearse desde el escritorio GNOME. Ofrece una serie de facilidades, como la navegación entre las frases del texto a traducir, el almacenamiento del autor y versión de la traducción, corrección ortográfica, que unidas a una interfaz gráfica a base de ventanas, colocan al traductor en un entorno amistoso en el que trabajar. Aunque ofrece una opción de "traducción automática", ésta no es otra cosa que la búsqueda de las palabras de forma independiente en un diccionario, sin prestar atención al contexto de la frase completa. Al encontrar el término buscado en el diccionario, le ofrece al usuario la posibilidad de aceptar o rechazar la traducción localizada.

- `POT-Creation-Date`: Este campo se rellena automáticamente por *xgettext* con la fecha y la hora del día en que se ejecutó.

- `PO-Revision-Date`: Este campo no es necesario rellenarlo. Será rellenado por el editor de ficheros PO utilizado cuando se salve el fichero.

- `Last-Translator`: Aquí introduciremos el nombre del traductor y su dirección de correo electrónico.

- `Language-Team`: En este campo introducimos el nombre en inglés de la lengua a la que queremos traducir el texto, y además, la dirección de correo electrónico o la URL de la página oficial del *Equipo de Traducción*[7] del que forma parte el traductor.

  Antes de comenzar con la traducción, es recomendable ponerse en contacto con el equipo de traducción al que pertenezca el traductor, no sólo para asegurar que no se esté duplicando el trabajo, sino también para coordinar las dificultades lingüíticas que puedan surgir.

- `Content-Type`: Debemos sustituir '*CHARSET*' por la codificación de caracteres usado en nuestro *locale*, o por UTF-8. Este campo es necesario para el correcto funcionamiento de `msgmerge` y de `msgfmt`, así como para los usuarios cuya localidad de codificación difiera de la nuestra, (véase la conversión de caracteres).

  La codificación de nuestro sistema podemos obtenerla ejecutando el comando '`locale charmap`'. Si el resultado de éste es 'C'o 'ANSI_X3.4-1968', que es equivalente a ASCII, significa que la localidad de nuestro sistema no está configurada correctamente. En este caso, deberíamos preguntar a nuestro Equipo de Traducción qué codificación debemos usar. ASCII sólo es utilizable para el Latín.

  Dado que los ficheros PO deben ser portables hacia sistemas operativos con menos facilidades de Internacionalización, las codificaciones que pueden ser utilizadas están limitadas a aquellos

[7] Conjunto de personas encargadas de la traducción de un determinado país e idioma. En http://translationproject.org podemos encontrar todos los equipos de traducción.

soportados tanto por GNU *libc* como por GNU *libiconv*[8].

A continuación se muestra una lista de codificaciones con sus correspondientes idiomas.

- `ISO-8859-1`: Afrikaans, Albanés, Alemán, Bretón[9], Catalán, Córnico[10], Danés, Estonio, Español, Feroés[11], Finlandés, Francés, Gallego, Groenlandés, Holandés, Inglés, Indonesio, Irlandés, Islandés, Malayo, Manés[12], Noruego, Occitano[13], Portugués, Sueco, Tagalo[14], Uzbeko, Valón[15] y Vasco.
- `ISO-8859-2`: Bosnio, Croata, Checo, Húngaro, Polaco, Rumano, Serbio, Eslovaco y Esloveno.
- `ISO-8859-3`: Maltés.
- `ISO-8859-5`: Macedonio y Serbio.
- `ISO-8859-6`: Arábico.
- `ISO-8859-7`: Griego.
- `ISO-8859-8`: Hebreo.
- `ISO-8859-9`: Turco.
- `ISO-8859-13`: Letón, Lituano y Maorí.
- `ISO-8859-14`: Galés.
- `ISO-8859-15`: Alemán, Catalán, Danés, Español, Finlandés, Francés, Gallego, Irlandés, Italiano,Portugués, Sueco, Valón y Vasco.

[8] Libreía de conversión de caracteres del proyecto GNU.

[9] La base del bretón (ar brezhoneg) es una lengua céltica al igual que el galés o el gaélico. Ha sido muy influenciado por el francés en tal modo que parte del léxico proviene de esta lengua. El bretón se habla esencialmente en el oeste de Bretaña (en una zona denominada Baja Bretaña).

[10] Lengua Oficial de Cornualles, condando administrativo y ceremonial que ocupa gran parte de la península del mismo nombre, consituyendo el extremo sudoccidental de Inglaterra

[11] Lengua escandinava hablada por unas 48.000 personas en las Islas Feroe y otras 12.000 en Dinamarca

[12] El idioma manés es una lengua celta que se habla en la isla de Man, una pequeña isla del Mar de Irlanda que constituye una dependencia autogobernada de la corona británica que no pertenece al Reino Unido.

[13] El occitano o lengua de oc es una lengua romance de Europa. Es hablada por entre dos y diez millones de personas en el sur de lo que hoy es Francia (al sur del río Loira), así como en Italia en los Valles Occitanos de los Alpes del Piamonte y el valle de Arán (en el noroeste de Cataluña, España; donde tiene carácter de lengua oficial).

[14] Lengua de origen neomalayo, base del idioma Filipino.

[15] Lengua románica del norte, de la misma rama que el francés. Se habla en Valonia, al sur de Bélgica.

- `KOI8-R`: Ruso.
- `KOI8-U`: Ucraniano.
- `KOI8-T`: Tayiko[16].
- `CP1251`: Búlgaro y Bielorruso.
- `GB2312, GBK y GB18030`: Chino simplificado.
- `BIG5 y BIG5-HKSCS`: Chino tradicional.
- `EUC-JP`: Japonés.
- `EUC-KR`: Coreano.
- `TIS-620`: Tailandés.
- `GEORGIAN-PS`: Georgiano.
- `UTF-8`: Para cualquier idioma, incluidos todos los anteriores.

En cualquier caso, y para garantizar la total portabilidad, se recomienda el uso del juego de caracteres UTF-8.

El nombre del juego caracteres puede escribirse tanto en minúsuculas como en mayúsculas, pero siempre es preferible en mayúsuculas.

- `Content-Transfer-Encoding`: Debe ser de 8 bits.
- `Plural-Forms`: Este campo es opcional. Sólo es necesario si el archivo PO contiene cadenas en plural, las cuales se pueden encontrar buscando la cadena '*msgid_plural*' en el fichero.

[16]El tayiko es una lengua iraní de la misma rama que el persa hablada por más de cuatro millones de personas. Es el idioma oficial de Tayikistán, aunque también se habla en países limítrofes.

2.2.5. Actualización de ficheros .PO existentes: *msgmerge*

El programa `msgmerge` mezcla dos ficheros .po de estilo Uniforum en uno solo.

`msgmerge [`*option*`]` *def*`.po` *ref*`.pot`

Figura 2.13: Invocación del programa msgmerge

El fichero `def.po` es un fichero existente de tipo .po con las traducciones anteriores que se conservarán siempre y cuando coincidan con las nuevas; los comentarios se conservarán, pero los comentarios extraídos y las posiciones del fichero no lo serán.

El fichero `ref.pot` es el último fichero .po creado (generalmente con xgettext), cualquier traducción o comentario se eliminará, sin embargo los comentarios puntuales y las posiciones dentro del fichero se conservarán. En donde no haya una coincidencia exacta, se utilizará el *método de comparación difusa*[17] para obtener mejores resultados.

A continuación veremos qué opciones se pueden utilizar con este programa:

Localización de los ficheros de entrada:

- `-D` *directory*`, --directory=`*directory*: Añade un directorio a la lista de directorios. Los ficheros fuentes serán buscados en relación a esta lista de directorios, sin embargo, el fichero `.po` resultante será escrito con relación al directorio actual.
- `-C` *file*`, --compendium=`*file*: Especifica una librería adicional de traducciones de mensajes.

Modo de operación:

- `-U, --update`: Actualiza el fichero def.po. No hace nada si ya se ha actualizado dicho fichero.

Localización del fichero de salida:

[17] Este método consiste en añadir la bandera fuzzy a las cadenas que parece no estar correctamente traducidas o simplemente están sin traducir.

- `-o` *`file`*`, --output-file=`*`file`*: Escribe la salida en el fichero especificado. Si no se especifica ninguno o se trata de '-', los resultados serán mostrados por la salida estándar.

Localización del fichero de salida en el modo *update*: el resultado de la ejecución será escrito en el fichero `def.po`.

- `--backup=`*`control`*: Crea una copia de seguridad del fichero `def.po`.
- `--suffix=`*`suffix`*: Ignora el sufijo habitual de la copia de seguridad.
  El método de control de versiones puede ser seleccionado mediante la opción '`--backup`' o a través de la variable de entorno VERSION_CONTROL.
  Los valores de esta variable pueden ser:
  - `none, off`: Nunca se hacen copias de seguridad, a menos que se especifique la opción '`--backup`'.
  - `numbered, t`: Crea copias de seguridad numeradas.
  - `existing, nil`: Crea copias de seguridad numeradas siempre y cuando existan ya copias de seguridad de este tipo. Si no es así, las crea sin numerar.
  - `never, simple`: Crea siempre copias de seguridad simples.

  El sufijo por defecto para las copias de seguridad es ' ~ ', a menos que se especifique uno con la opción '`--suffix`' o se cambie el valor de la variable de entorno SIMPLE_BACKUP_SUFFIX.

Modificadores de operación:

- `-m, --multi-domain`: Aplica ref.pot a cada uno de los dominios de def.po.
- `-N, --no-fuzzy-matching`: No utiliza la bandera *fuzzy* cuando no encuentra una ocurrencia exacta de la cadena. Esto acelera considerablemente la operación.
- `--previous`: Mantiene los *msgid* previos, marcados con '#|', al añadir la marca *fuzzy* a dichos mensajes.

Detalles de la salida:

Se utilizan las mismas opciones que en las llamadas a *xgettext*:

- `--force-po`
- `-i, --indent`
- `--no-location`
- `--add-location`
- `--strict`
- `-w` *number*, `--width=`*number*
- `--no-wrap`
- `-s, --sort-output`
- `-F, --sort-by-file`

Salida informativa:

- `-h, --help`: Muestra por la salida estándar todas las opciones posibles y después finaliza el programa.
- `-V, --version`: Muestra por la salida estándar la versión utilizada de *msgmerge* y después sale del programa.
- `-v, --verbose`: Aumenta la cantidad de mensajes informativos mostrados.
- `-q, --quiet, --silent`: Suprime los indicadores de progreso

2.2.6. Manipulación de ficheros .PO

A veces es necesario manipular los ficheros .po mejorando la realización automática de los mismos. GNU gettext incluye un juego de herramientas completo para este fin.

Cuando se combinan dos paquetes en un único paquete, el fichero resultante `.pot`, será la concatenación de los ficheros `.pot` de los paquetes de entrada. Así, el mantenedor debe concatenar las dos traducciones existentes de los paquetes en un único catálogo de traducción para cada idioma. Esto se realiza mejor utilizando el programa `msgcat`. Es entonces cuando los traductores deben hacer frente a los posibles conflictos que puedan surgir durante la fusión.

Cuando un traductor se hace cargo del trabajo de traducción de otro traductor, pero utiliza otra localización, deberá convertir el catálogo a la codificación específica que utilice. Para ello podrá utilizar la función `msgconv`.

Cuando un mantenedor toma un archivo de origen con la etiqueta de mensajes de otro paquete, puede obtener las traducciones existentes para este archivo de origen sin tener que duplicar el trabajo. Una forma de hacer esto es mediante `msggrep`, y otra es crear una fichero plantilla para ese fichero de origen y usar `msgmerge`.

Cuando un traductor desea ajustar algunos catalogos de traducción a un catálogo especial de dialectos o de otrografía - por ejemplo, alemán escrito en Suiza, frente a alemán escrito en Alemania - necesita aplicar algunas reglas de procesamiento del texto a cada mensaje del catálogo. La herramienta que nos permite hacer esto se llama `msgfilter`.

Otro uso de `msgfilter` es producir aproximadamente un archivo .pot para un fichero .po ya hecho. Tenga en cuenta que el archivo original .pot pudo haber tenido diferentes comentarios, o mensajes en plural, por lo que es mejor usar el archivo .pot original si es que este está disponible.

Existe una herramienta `msgexec`, que nos sirve para comprobar las traducciones, por ejemplo, comprobar la ortografía de acuerdo a las normas o utilizar un corrector ortográfico no interactivo.

Cuando herramientas de terceros crean un fichero .po o .pot, a veces no se pueden evitar los duplicados. Pero las herramientas de GNU gettext dan un error cuando encuentran duplicados msgid en el mismo fichero y con el mismo dominio. Para combinar duplicados podemos usar la herramienta `msguniq`.

El programa `msgcomm`, es una herramienta más general, para mantener o desechar los duplicados que se dan en los diferentes ficheros.

El programa `msgcmp` se usa para comprobar si el catálogo de mensajes está completamente traducido.

El programa `msgattrib` se utiliza para seleccionar y extraer sólo las cadenas marcadas como *fuzzy* o las que no están traducidas dentro del catálogo de mensajes.

El programa `msgen` es útil como primer paso para preparar los catálogos de traducción en inglés. Lo que hace es copiar las cadenas de *msgid* a *msgstr*.

2.2.6.1. Concatenación de ficheros .po: msgcat

Para concatenar varios ficheros PO válidos en un compedio de ficheros, pueden usarse los programas `msgcomm` o `msgcat`, este último preferido.

```
msgcat -o compendium.po file1.po file2.po
```

Figura 2.14: Invocación del programa msgcat para concatenar archivos .po

Por defecto, el programa *msgcat* acumulará traducciones divergentes para la misma cadena. Estas ocurrencias serán marcadas como *fuzzy,* difusas, y serán decoradas llamativamente para captar la atención del traductor.

Haciendo una llamada a `msgcat` con el fichero *file1*.po

```
#: src/hello.c:200
#, c-format
msgid "Report bugs to <%s>.\n"
msgstr "Comunicar ‘bugs’ a <%s>.\n"
```

Figura 2.15: Ejemplo de concatenación de ficheros .po: fichero de entrada 1

y el fichero *file2*.po:

```
#: src/bye.c:100
#, c-format
msgid "Report bugs to <%s>.\n"
msgstr "Comunicar \"bugs\" a <%s>.\n"
```

Figura 2.16: Ejemplo de concatenación de ficheros .po: fichero de entrada 2

tendremos como resultado en el fichero *compedium*.po lo siguiente:

```
#: src/hello.c:200 src/bye.c:100
#, fuzzy, c-format
msgid "Report bugs to <%s>.\n"
msgstr ""
"#-#-#-#-#  file1.po  #-#-#-#-#\n"
"Comunicar ‘bugs’ a <%s>.\n"
"#-#-#-#-#  file2.po  #-#-#-#-#\n"
"Comunicar \"bugs\" a <%s>.\n"
```

Figura 2.17: Ejemplo concatencación de ficheros .po: fichero de salida

El traductor deberá resolver este conflicto a mano. Es decir, debe decidir cuál de las dos es la traducción apropiada o incluso hacer una traducción nueva, si así lo cree necesario. Además deberá eliminar el marcador de líneas y la marca de *fuzzy*.

Si el traductor sabe de antemano, que la primera traducción encontrada es la correcta, puede usar la opción `--use-first`:

```
msgcat --use-first -o compendium.po file1.po file2.po
```

Un buen compedio de ficheros no debe contener nunca cadenas difusas o sin traducir. Si los ficheros de entrada son *dirty*[18], se deben preprocesar los ficheros de entrada o preprocesar la salida con el comando '`msgattrib -translated -no-fuzzy`'.

2.2.7. Introducción a los ficheros .MO

La mejor descripción del formato generado de un fichero `.mo` es la siguiente figura que se muestra más adelante.

Las dos primeras palabras sirven para la identificación del fichero:

- El número mágico es un identificador de los ficheros GNU MO. Esta constante es almacenada en el orden de bytes de la máquina que los genera, por tanto, el número mágico es en realidad dos números: 0x950412de y 0xde120495, dependiendo de si la máquina es *little endian* o *big endian*.
- La segunda palabra describe la actual revisión del formato del archivo. Por ahora la revisión es 0. Esto podría cambiar en futuras versiones, y se asegura que los lectores de los ficheros MO puedan distinguir nuevos formatos de los antiguos, a fin de que ambos puedan ser tratados correctamente. La versión se mantiene separada del número mágico, en lugar de utilizar distintos números mágicos para los diferentes formatos, especialmente porque /etc/magic no se actualiza con frecuencia.

A este identificador le sigue un número que indica la cantidad de punteros a tablas posteriores en el fichero, permitiendo que los ficheros `.mo` sean ampliados sin necesidad de ser recompilados. Esto es muy útil para la posterior inserción de algunos bits de bandera, inidicaciones acerca de caracteres utilizados, nuevos cuadros, etc.

[18] Un fichero se considera *dirty,* sucio, cuando contiene cadenas marcadas como *fuzzy*.

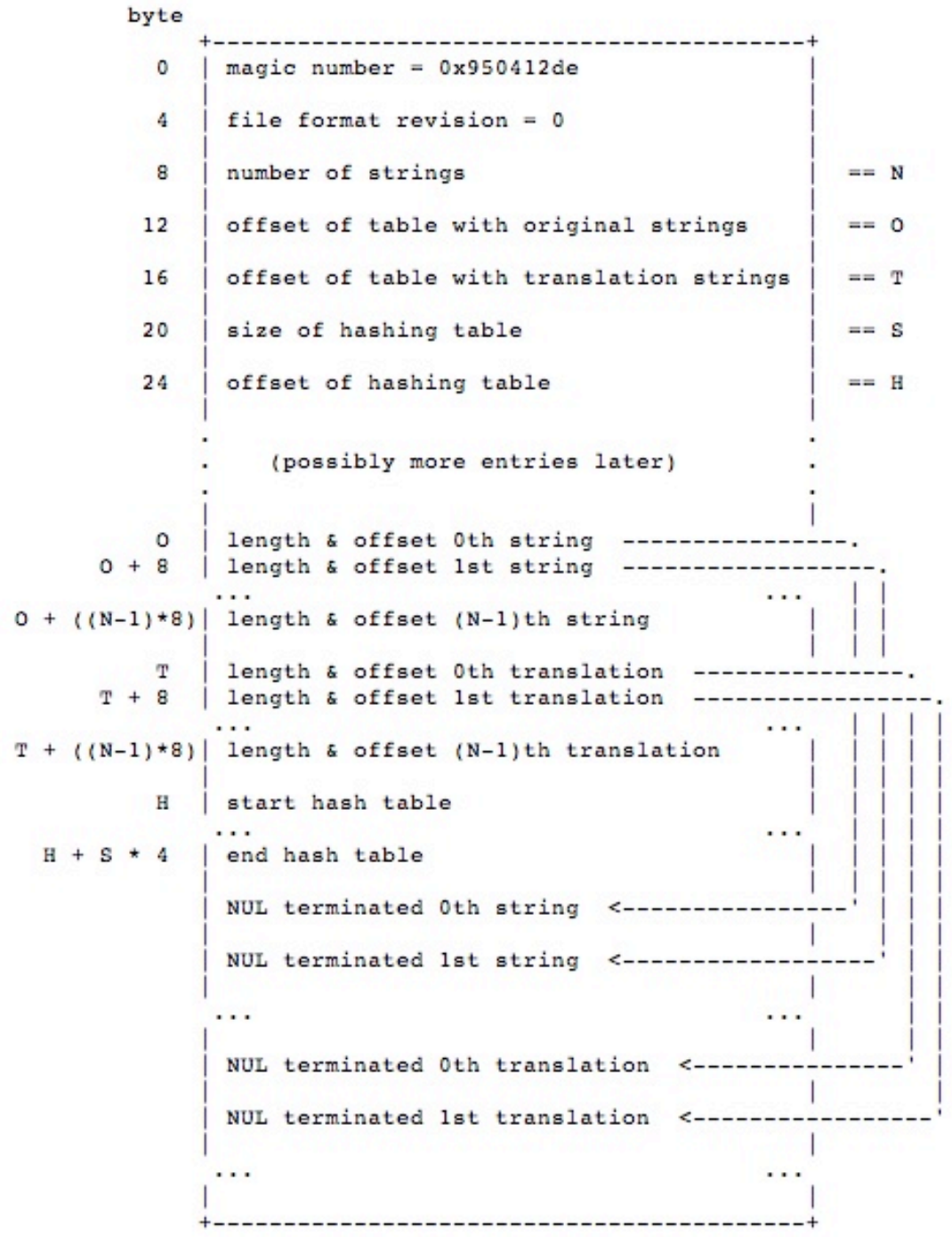


Figura 2.18: Formato de un fichero binario .mo

En las posiciones offset O y offset T de la figura, se pueden encontrar dos tablas de descriptores de cadena. En ambas, cada descriptor utiliza dos enteros de 32 bits, uno para la longitud de la cadena y otro para el *offset* o desplazamiento de la cadena dentro del fichero `.mo`, contando los bytes desde el principio del fichero.

La primera tabla contiene los descriptores de las cadenas originales, y está ordenada según un orden creciente lexicográfico de las cadenas originales. La segunda tabla contiene los descriptores de las cadenas traducidas, y es paralela a la primera tabla: para encontrar la traducción correspondiente a una entrada de la primera tabla, debe acceder a la segunda tabla con el mismo índice.

Tener las cadenas originales ordenadas permite el uso del algoritmo de la Búsqueda Binaria[19], siempre que el fichero `.mo` no contenga una tabla de dispersión o cuando la tabla proporcionada por el fichero no sea practicable. Además tiene otra ventaja, y es que algunas cadenas vacías en el fichero `.po` son traducidas por gettext a información del sistema adjuntándolo al fichero `.mo`, y la cadena vacía necesariamente ocupa la primera posición tanto de la primera tabla como de la segunda, haciendo muy sencilla la búsqueda de información del sistema.

El tamaño de la tabla de dispersión, S, puede ser cero. En ese caso, la tabla de dispersión en sí misma no está contenida en el fichero `.mo`. A veces es preferible esto, ya que esta tabla ocupa espacio en el disco y no gana en velocidad.

En cuanto a las cadenas que siguen, cada una termina con el carácter <NUL>, que no es tenido en cuenta en la longitud de la cadena.

[19]El algoritmo de la Búsqueda Binaria es un algoritmo recursivo que consiste en dividir un vector en dos partes e ir comparando el elemento buscado con el elemento central del vector. Como el vector está ordenado crecientemente, si el elemento buscado es mayor que el elemento central, se descartará el sub-vector que queda a la izquierda (que contiene elementos menores que el elemento central, y por tanto menores que el elemento buscado) y se volverá a aplicar el algoritmo sobre el sub-vector que queda a la derecha. Si el elemento buscado es menor que el elemento central el sub-vector que se descarta es el derecho y se vuelve a aplicar el algoritmo sobre el izquierdo. Si el elemento buscado es igual al elemento central el algoritmo finaliza.

El programa `msgfmt` posee una opción que permite seleccionar la alineación de las cadenas del fichero `.mo`. Con esta opción, las cadenas están separadas y alineadas de tal forma que empiezan con un desplazamiento que es múltiplo del valor de alineación. Esto lo veremos más adelante. En algunas máquinas RISC, este alineamiento aceleraría mucho el proceso.

Las forma plural de la cadena original se almacena a continuación de la forma singular de la cadena original, separándolas por el byte <NUL>. La longitud de la cadena que aparece en el descriptor las incluye a ambas. Sin embargo, sólo la forma singular de la cadena original es relevante a la hora de la búsqueda en la tabla de dispersión. Las variantes de la forma plural de la cadena traducida se almacenan consecutivamente, separadas por medio de <NUL>. La longitud de la cadena que aparece en el descriptor, al igual que antes, las incluye a todas.

2.2.8. Produciendo ficheros binarios .MO

En este apartado veremos cómo producir los ficheros binarios `.mo`.

El programa `msgfmt` genera un catálogo binario de mensajes a partir de la descripción de la traducción textual contenida en el fichero `.po`.

```
msgfmt [opción] fichero.po
```

Figura 2.19: Invocación del programa msgfmt

Veamos las distintas opciones para compilar los archivos `.po`.

Localización del fichero de entrada:

- `-D` *`directory`*`, --directory=`*`directory`*:añade el directorio *directory* a la lista de búsqueda

de ficheros de entrada.

Si el fichero de entrada es '–', se leerá de la entrada estándar.

Localización del fichero de salida:

- `-o` *`file`*`, --output-file=`*`file`*: Escribe la salida en el fichero *file* especificado.
- `--strict`: Escribe en estilo Uniforum estricto. Actualmente, esto sólo afecta a la denominación del archivo de salida. Si no se da esta opción el nombre del archivo de salida es el mismo que el nombre del dominio. Si el modo estricto Uniforum está habilitado el sufijo `.mo` se añadirá al nombre del archivo si no está ya presente.

  Si el fichero de salida es '–', se escribe en la salida estándar.

Interpretación del archivo de entrada:

- `-c, --check`: Hace todas las revisiones implicadas por `--check-format`, `--check-header` y `-check-domain`.
- `--check-format`: Revisa las cadenas de formato dependientes del idioma.

  Si la cadena representa una cadena con formato similar a funciones como *printf*[20], entonces tanto la cadena original como la traducida deben tener el mismo número de especificadores de formato '%', y además deben coincidir en el tipo de especificador.

  Normalmente, el programa `xgettext` decide automáticamente si una cadena es una cadena de formato o no. Sin embargo, este algoritmo no es perfecto. Se podría considerar que es una cadena de formato sin estar dentro de una llamada a *printf*, por lo que podrían producirse errores en la llamada a `msgfmt`.
- `--check-header`: Verifica la presencia y contenido de la línea de encabezado.
- `--check-domain`: Revisa si hay conflictos entre las instrucciones del dominio y la opción `--output-file`.

[20] Función del lenguaje C que se utiliza para mostrar mensajes por la salida estándar.

- `-C, --check-compatibility`: Revisa si el programa `msgfmt` de GNU se comporta como el `msgfmt` de X/Open.
- `-f, --use-fuzzy`: Utiliza entradas difusas en el archivo de salida. Esta opción es usualmente incorrecta, ya que significa que los mensajes no han sido validados por un traductor humano.

Detalles en el fichero de salida:

- `-a` *`number`*`, --alignment=`*`number`*: Alinea las cadenas con un offset que será múltiplo de *number*.
- `--no-hash`: El fichero binario no incluirá la tabla de hash.

Salida informativa:

- `-h, --help`: Muestra la ayuda, que contiene entre otras opciones las detalladas aquí y sale del programa.
- `-V, --version`: Muestra la versión instalada del programa y lo finaliza.
- `--statistics`: Muestra estadísticas acerca de las traducciones.
- `-v, --verbose`: Aumenta el número de mensajes de información al ejecutar el comando `msgmt`.

2.2.9. Estructura de directorios: *locale*

La librería GNU Gettext busca los archivos compilados (`.mo`) con las definiciones del lenguaje en un directorio y sus subdirectorios. Estos directorios siguen un patrón concreto, que nos ayudará a tener organizados los archivos de lenguaje.

El nombre del directorio raíz puede ser el que deseemos, pero por convenio se utiliza `locale`. En este directorio debemos crear un nuevo directorio por cada idioma del que queramos dotar a nuestra

aplicación. El nombre de estos directorios deben ser el código locale de cada país e idioma.

Como explicamos anteriormente, con el comando '`locale -a`' podremos saber qué conjunto de localidad tenemos instalado en nuestro sistema.

A su vez, esos subdirectorios contendrán otro subdirectorios, denominados `LC_MESSAGES`, los cuales contendrán los archivos compilados.

A continuación se muestra un ejemplo para mostrar esto con más claridad.

```
$ tree
.
|-- aplicacion.php
`-- locale
    |-- de_DE
    |   `-- LC_MESSAGES
    |-- en_GB
    |   `-- LC_MESSAGES
    `-- es_ES
        `-- LC_MESSAGES
```

Figura 2.20: Ejemplo de estructura de directorios *locale*

2.2.10. Uso en PHP

El idioma se selecciona ajustando la localización con la función `setlocale()`, aunque en ocasiones también han de ajustarse las variables de entorno *LANG* o *LANGUAGE* a través de `putenv()`.

El código del "locale" que usemos ha de especificar el idioma y país, por ejemplo: es_ES, es_MX, en_GB o en_US. Pero debemos tener en cuenta la localidad que utiliza nuestro sistema. Esto lo veremos y trataremos más a fondo en el capítulo siguiente.

Un ejemplo en lenguaje PHP sería:

```php
<?php
/**
* setlocale fija la información de localización
* El primer parámetro que se le pasa se utiliza para especificar
*la categoría de las funciones afectadas por el ajuste de
*localización. LC_ALL afecta a todas las funciones.
*/

setlocale(LC_ALL, 'es_ES');

/**
* bindtextdomain define la ruta para el dominio 'messages'
*nombre de nuestros ficheros de traducción compilados .mo
*/

bindtextdomain('messages', './locale');

/**
* Llamada a gettext mediante su alias _()
*/

echo _('Hello') . "n";
?>
```

Figura 2.21: Ejemplo de uso de gettext en PHP

Si no se encuentra la localización correcta o hay algún problema a la hora de ajustarla, el mensaje mostrado será el identificador, es decir, la cadena *msgid* correspondiente.

Capítulo 3

Core: Sistema Colaborativo para la Gestión de Conferencias

3.1. Requisitos de la aplicación

Es necesario tener en cuenta, a lo largo de este capítulo, que el proyecto ha sido realizado sobre la plataforma *Ubuntu Linux 7.10 Gutsy Gibbon*[1]. Por tanto, se asume la instalación de ciertos paquetes previos y otros requisitos del sistema, así como la jerarquía de directorios que posee este sistema.

3.1.1. Servidor Apache

El Servidor HTTP del Proyecto Apache es un servidor HTTP de código abierto multiplataforma. Un servidor web es un programa que implementa el protocolo HTTP. Este protocolo está diseñado para transferir lo que llamamos hipertextos.

El servidor web se encarga de mantenerse a la espera de peticiones HTTP llevadas a cabo por un cliente HTTP, más conocido como navegador. Este navegador realiza una petición al servidor y éste le contesta con el contenido que el cliente solicita.

El cliente es el encargado de interpretar el código HTML que le llega del servidor, el cuál sólo se

[1]Distribución Linux basada en Debian GNU/Linux.

limita a transferir el código de la página sin llevar a cabo ninguna interpretación de la misma.

Podemos diferenciar distintas aplicaciones web:

- Aplicaciones del lado del cliente: El cliente web es el encargado de ejecutarlas en la máquina del usuario. El servidor proporciona el código de las aplicaciones al cliente y éste las ejecuta mediante el navegador. Son aplicaciones tipo JavaScript.
- Aplicaciones del lado del servidor: El servidor web ejecuta la aplicación y ésta una vez ejecutada, genera código HTML que posteriormente es enviado al cliente por medio del protocolo HTTP.

La aplicación que aquí se presenta es una aplicación en el lado del servidor escrita en lenguaje PHP y que más adelante se explicará con más detalle.

3.1.1.1. Instalación y configuración del Servidor Apache con soporte SSL

El protocolo SSL[2] es un protocolo criptográfico que proporciona comunicaciones seguras en Internet.

Este protocolo proporciona autenticación y privacidad de la información entre extremos de la comunicación (cliente y servidor) mediante el uso de la criptografía. Habitualmente, sólo el servidor es autenticado, es decir, se garantiza su identidad, mientras que el cliente se mantiene sin autenticar.

Este protocolo permite a las aplicaciones cliente-servidor comunicarse de una forma diseñada para prevenir escuchas, la falsificación de la identidad de la otra parte (comúnmente conocido como *phishing*, en el caso del servidor) y mantener la integridad del mensaje.

[2]Secure Socket Layer

El protocolo SSL implica una serie de fases básicas:

1. Negociar entre ambas partes (cliente y servidor), el algoritmo que se usará en la comunicación.
2. Intercambio de claves públicas y autenticación basada en certificados digitales.
3. Cifrado del tráfico basado en *cifrado simétrico*[3].

La aplicación *Core*, proporciona a sus usuarios una conexión segura en el momento de autenticarse para poder hacer los cambios pertinentes en la información privada, así como la administración de presentaciones y ponencias ligadas a una sesión.

Para ello, a continuación se detalla la instalación del Servidor Apache con soporte SSL.

En primer lugar debemos descargar el código fuente para su posterior instalación. Esta tarea podemos realizarla mediante varias opciones: en Linux, con el Gestor de Paquetes Synaptic o mediante línea de comandos con el comando '`sudo apt-get install apache2-src`' .También podremos descargarlo de la página web oficial del Proyecto Apache.

El código fuente vendrá empaquetado en un archivo comprimido y se ubicará en '`/usr/src`'. Una vez descargado, procederemos de la siguiente manera:

[3]Método criptográfico que usa una misma clave para cifrar y para descifrar mensajes. Las dos partes que se comunican han de ponerse de acuerdo de antemano sobre la clave a usar. Una vez ambas tienen acceso a esta clave, el remitente cifra un mensaje usándola, lo envía al destinatario, y éste lo descifra con la misma.Un buen sistema de cifrado pone toda la seguridad en la clave y ninguna en el algoritmo.

```
#Descomprimimos el fichero que contiene el código fuente
~$ cd /usr/src/
~/usr/src/ gunzip apache2.tar.gz
~/usr/src/ tar -xvf apache2.tar
~/usr/src/ cd apache2
~/usr/src/apache2 ls
ABOUT_APACHE      BuildBin.dsp   config.sub    httpd.dsp       LAYOUT              modules
NWGNUmakefile     ROADMAP        VERSIONING    acinclude.m4    buildconf           configure
httpd.spec        libhttpd.dsp   mpm-event     os              server Apache.dsw   CHANGES
configure.in      include        LICENSE       mpm-prefork     patch-stamp         srclib
apachenw.mcp.zip  config.guess   docs          INSTALL         Makefile.in         mpm-worker
README            support        build         config.layout   emacs-style         InstallBin.dsp
Makefile.win      NOTICE         README.platforms              test
#A continuación ejecutamos el script configure habilitando las siguientes opciones
~/usr/src/apache2/ ./configure --enable-so --enable-ssl --enable-mime-magic
~/usr/src/apache2/ make
~/usr/src/apache2/ sudo make install
```

Figura 3.1: Instalación de Apache 2 con soporte SSL

Como vemos, la instalación de Apache 2 es muy sencilla, pero debemos tener en cuenta que debe configurarse con las opciones indicadas.

- `--enable-so`: Esta opción es muy importante a la hora de configurar Apache 2. Esta opción habilita la capacidad de cargar nuevos módulos para el servidor de manera dinámica.
- `--enable-ssl`: Esta opción carga el módulo SSL. Con este módulo podremos crear aplicaciones web bajo el protocolo HTTPS.
- `--enable-mime-magic`: Con esta opción se determinarán los tipos MIME mediante números mágicos.

3.1.1.2. Creación de un certificado SSL para el proceso de desarrollo de la aplicación

Para configurar un servidor seguro es necesario enviar una solicitud de certificado (incluyendo la clave pública), una prueba de la identidad de la compañía, y el pago correspondiente a una Autoridad de Certificación (Certificate Authority, CA). La CA verifica la solicitud del certificado y su identidad, y posteriormente envía un certificado para el servidor seguro.

Pero también podemos crear un certificado auto-firmado. Pero este certificado sólo nos servirá durante el desarrollo de nuestra aplicación, ya que los certificados auto-firmados no deben usarse en

la mayoría de los entornos de producción. Los certificados auto-firmados no son aceptados automáticamente por los navegadores de los usuarios. Los navegadores solicitarán al usuario que acepte el certificado para crear la conexión segura.

Cuando tengamos el certificado auto-firmado, o el certificado firmado por una CA , necesitaremos instalarlo en nuestro servidor seguro.

Muchos navegadores web que soportan SSL tienen una lista de CAs cuyos certificados aceptan automáticamente. Si un navegador encuentra un certificado autorizado por una CA que no está en su lista, el navegador le preguntará al usuario si desea aceptar o denegar la conexión.

Nosotros hemos generado un certificado auto-firmado para nuestro servidor, pero debemos tener en cuenta que ese certificado no proporciona la misma funcionalidad que un certificado firmado por una CA. La mayoría de los navegadores web no reconocen automáticamente los certificados auto-firmados, y éstos además no proporcionan ninguna garantía acerca de la identidad de la organización que está proporcionando el sitio web. Un certificado firmado por una CA proporciona estas dos importantes características a un servidor seguro. El proceso para obtener un certificado de una CA es realmente fácil.

A grandes rasgos, consta de:

1. Crear un par de claves, pública y privada.
2. Crear una solicitud de certificado basado en la clave pública. La solicitud de certificado contiene información sobre nuestro servidor y la compañía que lo aloja.
3. Enviar la solicitud de certificado, junto con los documentos que prueban nuestra identidad, a una CA.
4. Cuando la CA esté segura de que tiene todo lo que necesitamos, recibiremos un certificado digital.

5. Instalar el certificado en el servidor seguro y soportar transacciones seguras.

Tanto para obtener un certificado de una CA, como para crearnos el nuestro propio, lo primero que debemos hacer es generar un CSR (Certificate Signing Request), o lo que es lo mismo, una Petición de Firma de Certificado. Para ello debemos ejecutar el siguiente comando:

```
1  #Creamos un directorio en /usr/local/src donde almacenaremos los ficheros que se generarán con las
       ... claves para el certificado
2
3  ~$ mkdir /usr/local/src/keygen
4  ~$ cd /usr/local/src/keygen
5
6  #A continuación generamos la clave privada RSA
7
8  ~/usr/local/src/keygen openssl genrsa -des3 -out server.key 1024
9
10 #Este comando produce la siguiente salida:
11
12 Generating RSA private key, 1024 bit long modulus
13 ..................................................++++++
14 ...................++++++
15 e is 65537 (0x10001)
16 Enter pass phrase for server.key:
```

Figura 3.2: Generación de la clave privada del servidor seguro

El comando nos pide introducir una frase de paso. Es muy importante recordarla porque nos será solicitada más adelante en el proceso.

Para mayor seguridad, ésta debería contener, al menos, ocho caracteres. La longitud mínima al especificar la opción `-des3` es de cuatro caracteres. Debe incluir números y/o signos de puntuación, y no debería ser una palabra que se pudiera encontrar en un diccionario. Además, recuerde que su frase de paso distingue mayúsculas de minúsculas.

Una vez insertada, debemos volver a escribirla para verificarla. Cuando esto ocurra, se generará la clave del servidor y se almacenará en el archivo `server.key`.

Un desafortunado efecto secundario del paso de frase de la clave privada, es que Apache la solicitará cada vez que queramos arrancar el servidor. Obviamente esto no es necesariamente conveniente,

porque no siempre habrá alguien cerca para introducir la frase de paso, por ejemplo después de reiniciar el sistema o de un posible accidente.

Es posible eliminar la encriptación Triple-DES de la clave, con lo que la frase de paso no nos será de nuevo solicitada. Si la clave privada ya no está encriptada , es fundamental que este archivo sólo pueda ser leído por el usuario root. Eliminar esto es sólo recomendable en el entorno de desarrollo, ya que si se hiciera en un entorno de producción podría haber bastantes problemas de seguridad. Para eliminarlo podemos ejecutar el siguiente comando:

```
openssl rsa -in server.key -out server.pem

#Debemos introducir la frase de paso y obtendremos el siguiente mensaje

writing RSA key
```

Figura 3.3: Eliminación de la encriptación Triple-DES de la clave privada de un entorno de desarrollo

Ahora ya estamos listos para crear el CSR, que normalmente será enviado a una CA para que lo firme, pero como hemos explicado antes, para un entorno de desarrollo como el nuestro, esto no será necesario y lo firmaremos nosotros mismos.

```
#Ejecutamos el siguiente comando para crear el CSR

~/usr/local/src/keygen openssl req -new -key server.key -out server.csr

#Cuando esto comience, el script nos pedirá cierta información. La mayoría es meramente
#informativa, pero hay otra que no. He aquí lo que se pide:

Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:
Email Address []:

Please enter the following 'extra' attributes to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Figura 3.4: Generación del CSR (Certificate Signing Request)

La entrada Common Name es la más importante. Aquí debemos poner el nombre de nuestro servidor, tal y como aparece en la entrada ServerName del fichero de configuración de Apache, httpd.conf.

Como estamos creando un certificado auto-firmado para un entorno de desarrollo, utilizaremos para esta entrada `127.0.0.1`, que es la IP por defecto asociada a localhost.

Una vez creada la petición, debemos crear el certificado auto-firmado. Para ello, ejecutamos los siguiente:

```
1  openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
2
3  #Este comando generará la siguiente salida, donde nuevamente se solicitará la frase de paso:
4
5  Signature ok
6  subject=/C=ES/ST=Spain/L=Seville/O=RGZC/OU=RGZC/CN=Rosario/emailAddress=rgzuniga@cica.es
7  Getting Private key Enter pass phrase for server.key:
```

Figura 3.5: Creación del certificado auto-firmado

Tras crear el certificado, debemos copiar, como usuario `root`, el fichero `server.crt` al directorio `/etc/ssl/certs` y los ficheros `server.key` y `server.pem` al directorio `/etc/ssl/private`.

Debemos tener en cuenta que estos ficheros sólo pueden ser accedidos por el usuario `root`, por tanto le cambiamos los permisos:

```
1  #Como usuario root
2  chmod 400 /etc/ssl/certs/server.*
3  chmod 400 /etc/ssl/private/server.*
```

Por último debemos configurar el servidor para que acepte peticiones a páginas https.

Tenemos que especificarle a Apache que debe escuchar además de por el puerto 80, por el puerto 443, puerto por el que escucha el protocolo HTTPS. Editando el fichero `httpd.conf`:

```
1  #Debajo de la línea Listen *:80 añadimos
2
3  Listen *:443
```

Como queremos escuchar por ambos puertos, debemos especificar un VirtualHost para cada uno de ellos.

Confguramos en primer lugar el puerto 80:

```
#Configuración del VirtualHost para el puerto 80

<VirtualHost 127.0.0.1:80>
DocumentRoot "/usr/local/apache2/htdocs"
ServerName 127.0.0.1
#Aquí especificamos el email del administrador del servidor
ServerAdmin r.gzuniga.canivell@gmail.com
SSLEngine off
</VirtualHost>
```

Figura 3.6: Configuración del VirtualHost del puerto 80

Configuramos a continuación el puerto 443:

```
#Configuración del VirtualHost para el puerto 443

<VirtualHost 127.0.0.1:443>
DocumentRoot "/usr/local/apache2/htdocs"
ServerName 127.0.0.1
ServerAdmin r.gzuniga.canivell@gmail.com
ErrorLog /usr/local/apache2/logs/error_ssl_log
TransferLog /usr/local/apache2/logs/access_ssl_log
SSLEngine on
SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire

#A continuación debemos especificar la ruta hacia el certificado y las claves que hemos creado

SSLCertificateFile /etc/ssl/certs/server.crt

#Si queremos ingresar la frase de paso cada vez que iniciemos el servidor debemos agregar:

SSLCertificateKeyFile /etc/ssl/private/server.key

#Si por el contrario, en nuestro entorno de desarrollo creemos conveniente que no es necesario
#y es más fácil, agregaremos la siguiente línea:

SSLCertificateKeyFile /etc/ssl/private/server.pem
</VirtualHost>
```

Figura 3.7: Configuración del VirtualHost del puerto 443

Con esto, hemos finalizado la instalación y configuración de nuestro servidor. Para comprobar su correcta instalación, abrimos nuestro navegador preferido y nos dirigimos a `http://localhost`. Debe aparecernos un mensaje como el siguiente:

Figura 3.8: Mensaje de éxito en la instalación de Apache 2

3.1.2. Servidor PostgreSQL

PostgreSQL es un servidor de bases de datos relacionales, liberado bajo *licencia BSD*[4].

Las principales características de PostgreSQL son:

- Multihilo y multiusuario.
- Amplia variedad de tipos nativos..Entre ellos caben destacar los números de precisión arbitraria, el texto de largo ilimitado, direcciones IP y MAC, arrays y la posibilidad de creación de propios tipos de datos del usuario.
- Disparadores o *triggers*, Vistas, Integridad Transaccional, y funciones (que pueden ser definidas en varios lenguajes además del lenguaje propio PL/PgSQL) entre otras.

A continuación se explicará la instalación necesaria para nuestra aplicación.

3.1.2.1. Instalación e inicialización del sistema

Para comenzar, debemos asegurarnos de que las dependencias para la instalación de PostgreSQL estén debidamente instaladas. En cualquier caso, a la hora de la configuración, si alguna de estas

[4]Licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution). Pertenece al grupo de licencias de Software Libre.

dependencias no estuviera disponible, el script de configuración nos avisará de ello.

Descargamos la última versión de PostgreSQL de **http://www.postgresql.org/** y procedemos de la siguiente manera:

```
1  ~/usr/local/src/ tar -zxvf postgresql-8.2.5.tar.gz
2  ~/usr/local/src/ cd postgresql-8.2.5
3
4  #Para obtener más ayuda sobre cómo configurarlo ejecutaremos './configure --help'
5  #El directorio de instalación por defecto es '/usr/local/pgsql'
6  #pero se puede especificar un directorio de instalación con la opción '--prefix=DIR'
7
8  ~/usr/local/src/postgresql-8.2.5 ./configure
9  ~/usr/local/src/postgresql-8.2.5 sudo make
10 ~/usr/local/src/postgresql-8.2.5 sudo make install
```

Figura 3.9: Instalación del Servidor PostgreSQL

Una vez tenemos todo instalado correctamente, procedemos a la inicialización del sistema.

Creamos en primer lugar un usuario en el sistema llamado `postgres` con la contraseña que deseemos. Este usuario es necesario para la inicializacion del servidor, ya que debe existir estrictamente una base de datos denominada postgres en la que se inicializará el servidor. Posteriormente podremos añadir tantos usuarios al sistema PostgreSQL como queramos, así como dotarlos con los privilegios que deseemos para no tener que trabajar con el usuario postgres.

Como hemos comentado anteriormente, a la hora de ejecutar el script '`./configure`' podemos especificar el directorio de instalación de PostgreSQL. Nosotros lo hemos instalado en el directorio por defecto para este cometido '`/usr/local/pgsql`'.

En el directorio pgsql creamos un nuevo directorio, `data/`, que será el que albergue todo nuestro servidor. A continuación se muestran los comandos ejecutados para realizar esta acción:

```
#Añadimos un nuevo usuario al sistema

adduser postgres

#Creamos el directorio data/

mkdir /usr/local/pgsql/data

#Hacemos que postgres sea su propietario

chown postgres /usr/local/pgsql/data

#Cambiamos al usuario postgres

su - postgres

#Creamos el clúster de bases de datos que contendrá las bases de datos que vayamos creando
#La opción -E sirve para indicar la codificación por defecto al crear nuevas bases de datos
#La opción -D sirve para indicar el directorio donde queremos albergar las bases de datos

/usr/local/pgsql/bin/initdb -E UTF-8 -D /usr/local/pgsql/data
```

Figura 3.10: Inicialización del Sistema Gestor de Bases de Datos PostgreSQL

Una vez creado el clúster, es necesario que iniciemos el servidor para poder empezar a operar con nuestras bases de datos.

Para ello, existe un comando en el directorio bin de pgsql llamado `pg_ctl`. Este comando tiene los siguientes usos:

- `pg_ctl start` *opciones*: Se utiliza para lanzar el servidor PGSQL. La opción más habitual es -D para indicar el directorio donde se albergan las bases de datos.
- `pg_ctl stop` *opciones*: Sirve para parar el servidor PGSQL. Si al lanzarlo se utilizó la opción -D, para pararlo es necesario indicar el mismo directorio.
- `pg_ctl restart` *opciones*: Sirve para reiniciar el servidor.
- `pg_ctl reload` *opciones*: Sirve para recargar los archivos de configuración del servidor.
- `pg_ctl status` *opciones*: Imprime el estado del servidor (si está en ejecución, parado, etc).
- `pg_ctl kill SIGNALNAME PID`: Sirve para mandar una señal al proceso cuyo PID se indique. Estas señales pueden ser HUP, INT, QUIT, ABRT, TERM, USR1 y USR2.

El comando que más utilizaremos será:

`/usr/local/pgsql/bin/pg_ctl -D start /usr/local/pgsql/data`,

indicando el directorio como hemos comentado antes.

Otra forma de lanzar el servidor es con el comando:

`/usr/local/pgsql/bin/postgres -D /usr/local/pgsql/data`

Esta forma es equivalente a la anterior, por tanto, podremos usar una forma u otra indistintamente.

Una vez arrancado el servidor, podremos crear bases de datos, usuarios y realizar todo tipo de gestiones tanto con los comandos proporcionados por el SGBD como ejecutando sentencias SQL y otros comandos por la línea de comandos de pgsql. Para ello, basta con ejecutar `/usr/local/pgsql/bin/psql`[5] y se nos abrirá el prompt de psql.

3.1.3. PHP 5

PHP (acrónimo de 'PHP: Hypertext Pre-processor') es un lenguaje de programación usado normalmente para la creación de páginas web dinámicas. Se trata de un lenguaje interpretado, esto es, un lenguaje que se ejecuta por medio de un intérprete, en lugar de ser compilado.

Su interpretación y ejecución se da en el servidor web, en el cual se encuentra almacenado el script, y el cliente sólo recibe el resultado de la ejecución. Cuando el cliente hace una petición al servidor para que le envíe una página web, generada por un script PHP, el servidor ejecuta el intérprete de PHP, el cual procesa el script solicitado que generará el contenido de manera dinámica, pudiendo modificar el contenido a enviar, regresando el resultado al servidor, y éste al cliente.

La similitud con otros lenguajes más comunes de programación estructurada, como C++ o Perl, permiten a la mayoría de los programadores experimentados crear aplicaciones complejas con una

[5]Para más información sobre este comando, ejecutar `/usr/local/pgsql/bin/psql --help`

curva de aprendizaje muy suave. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones y prácticas.

Además también es posible crear aplicaciones con una interfaz gráfica para el usuario (también llamada GUI), utilizando la extensión PHP-Qt o PHP-GTK. Asimismo, puede ser usado desde la línea de comandos, de la misma manera que Perl y Python pueden hacerlo.

PHP permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, PostgreSQL, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite, lo que nos permite la creación de aplicaciones web muy robustas.

Además tiene la capacidad de ser ejecutado por la mayoría de los sistemas operativos tales como UNIX (Linux, Mac OS X) y Windows y puede interactuar con los servidores web más populares ya que existe en versión CGI[6], módulo para Apache, e ISAPI[7].

Por último, antes de pasar a la instalación requerida para nuestra aplicación, es necesario destacar que, desde su puesta en marcha en 1995 se ha extendido exponencialmente, convirtiéndose en el lenguaje de scripting de la web de referencia en el mundo del software libre, siendo así una alternativa perfecta a otros lenguajes de scripting, accesible a todos.

3.1.3.1. Instalación y configuración

Una vez descargados los ficheros fuente de **http://php.net**, pasamos a explicar la configuración e instalación que se ha llevado a cabo para que nuestra aplicación funcione.

A continuación se muestran los comandos ejecutados para su configuración.

[6] Acrónimo de Common Gateway Intarface, que permite a un cliente solicitar datos de un programa ejecutado en un servidor web. CGI especifica un estándar para transferir datos entre cliente y programa. Las aplicaciones que se ejecutan en el lado del servidor se denominan CGIs.

[7] ISAPI (Internet Server Application Programming Interface) es una interfaz de programación de aplicaciones de servidor para internet. Desarrollado por Process Software y Microsoft Corporation, que se utiliza en lugar de CGI.

```
~/usr/local/src tar -zxvf php-5.2.5.tar.gz
~/usr/local/src cd php-5.2.5

#A continuación se ejecuta el script de configuración con todas las opciones necesarias para su
     ...correcta instalación

~/usr/local/src/php-5.2.5 ./configure --prefix=/usr/share --with-apxs2=/usr/local/apache2/bin/apxs
--build=i486-linux-gnu --host=i486-linux-gnu --mandir=/usr/share/man --disable-debug
--with-regex=php --disable-rpath --disable-static --with-pic --with-layout=GNU
--with-pear=/usr/share/php --enable-calendar --enable-sysvsem --enable-sysvshm
--enable-sysvmsg --enable-ftp --with-gettext --enable-mbstring --enable-shmop --enable-sockets
--enable-wddx --with-libxml-dir=/usr --with-zlib --with-kerberos=/usr --with-openssl=/usr
--enable-soap --enable-zip --with-mime-magic=/etc/magic.mime --with-exec-dir=/usr/lib/php5/libexec
--without-mm --with-curl=shared,/usr --with-zlib-dir=/usr --with-gd=shared,/usr/local
--with-jpeg-dir=shared,/usr --with-xpm-dir=shared,/usr/X11R6 --with-png-dir=shared,/usr
--with-freetype-dir=shared,/usr --with-pgsql=/usr/local/pgsql

#Ejecutamos los comandos de instalación

~/usr/local/src/php-5.2.5 sudo make && make install
```

Figura 3.11: Configuración e instalación de PHP 5

Ahora se explicarán las opciones más importantes, y porqué son necesarias en nuestra aplicación:

- `--prefix=[DIR]`: Se utiliza para definir el directorio donde queremos que se instale PHP. Si no se especifica, se instalará en el directorio por defecto, /usr/local.
- `--with-apxs2=[ruta de apxs]`: Sirve para que php se compile como módulo de Apache 2, y por tanto éste pueda ejecutar nuestros scripts escritos en PHP. Debe especificarse la ruta del archivo binario apxs. Si Apache 2 fue compilado e instalado como se explicó en la sección 3.2.1, este fichero se encontrará en /usr/local/apache2/bin/.
- `--with-pear=[DIR]`:Instala PEAR en el directorio que se especifica.Es necesario ya que como veremos más adelante, son necesarios muchos paquetes.
- `--with-gettext`: Permite que se hagan llamadas a gettext dentro de nuestros scripts PHP. Imprescindible para el soporte de internacionalización.
- `--enable-mbstring`: Permite el uso de funciones para cadenas multibyte.
- `--enable-shmop`: Habilita el uso de Shmop, un conjunto de funciones que permiten a PHP leer, escribir, crear y borrar de forma sencilla segmentos de memoria compartida de tipo UNIX.

- `--enable-wddx`: Habilita el soporte para WDDX[8].
- `--with-zlib`: Instala zlib[9], que es requerida por otras librerías como las librerías de imágenes jpeg, png, etc.
- `--with-openssl`: Permite usar las funciones openssl desde PHP.
- `--with-gd`: Librería necesaria para el manejo de imágenes.
- `--with-jpeg-dir=[DIR]`: Librería para el manejo de imágenes en formato jpeg. Depende de libgd.
- `--with-png-dir=[DIR]`: Librería para el manejo de imágenes en formato png. Depende también de libgd.

Por último, es necesario que Apache sea capaz de ejecutar el analizador de PHP cuando el cliente solicita una página php, y para ello tenemos que añadir la siguiente línea al fichero de configuración `httpd.conf`:

```
1 #Para que Apache analice páginas en php, debemos agregar la siguiente línea al fichero httpd.conf
2 AddType application/x-httpd-php .php
3 #Si queremos que Apache además, analice otro tipo de ficheros con el analizador de PHP añadiremos
      ...más extensiones, por ejemplo:
4 AddType application/x-httpd-php .php .phtml .php3
5 AddType application/x-httpd-php-source .phps
6 #Estos dos últimos ejemplos son opcionales, sin embargo, es obligatorio añadir la primera línea si
      ... queremos que funcione Apache con PHP.
```

Figura 3.12: Configuración de Apache con PHP5

[8]WDDX es un estándar XML para el intercambio de información estructurada entre distintos lenguajes de programación.Al permitir fácil y claramente el intercambio de datos entre distintos lenguajes, WDDX podría convertirse en una herramienta muy útil de la web de servicios, ya que una empresa que desee proveer un determinado servicio puede utilizar el lenguaje que quiera y entregar como resultado datos WDDX para que otras empresas usando otros lenguajes puedan tomarlo y utilizarlo.

[9]Librería de compresión de datos

3.1.3.2. Administración de bases de datos: phpPgAdmin

`phpPgAdmin` es una aplicación web escrita en PHP para administrar bases de datos PostgreSQL. Esta aplicación nos permite buscar, crear, borrar y modificar bases de datos, tablas, vistas, secuencias, funciones, dominios y usuarios.

Nos permite estar conectados a más de un servidor de bases de datos y la posibilidad de exportar nuestras bases de datos. Podemos exportar tanto los datos solamente, como sólo la estructura e incluso ambas, dándonos la posibilidad de mostrarlo o descargarlo.

También ofrece diferentes asistentes para que la tarea de administrar nuestras bases de datos sea más asequible.

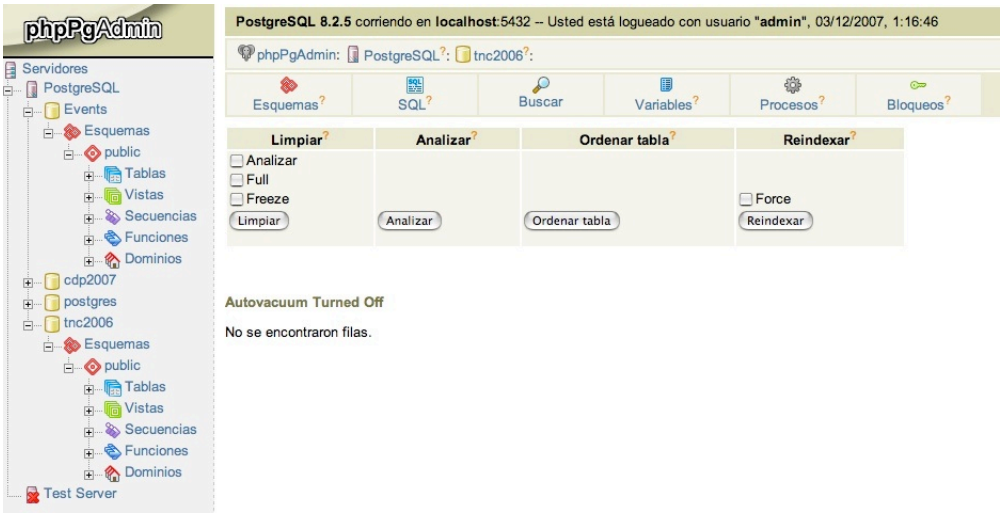


Figura 3.13: Administración de BBDD con phpPgAdmin

Esta aplicación ofrece una interfaz gráfica bastante amigable, que facilita la tarea de gestionar nuestras bases de datos, mostrándolas de forma muy nítida. Es por ello que se eligió esta herramienta para gestionar las bases de datos de nuestra aplicación.

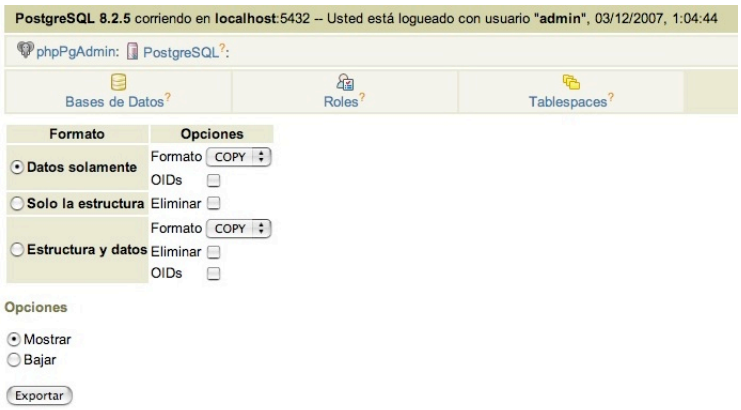


Figura 3.14: Exportar BBDD en phpPgAdmin

Figura 3.15: Ejemplo de búsqueda en phpPgAdmin

3.1.4. Paquetes PEAR y PECL

Además de todas las opciones utiliadas anteriormente en la configuración de PHP, este lenguaje de programación, sin duda completo y robusto, ofrece un amplio repertorio de módulos o extensiones, los cuales facilitan la labor del programador, pudiendo reutilizar código creado por otros programadores. Esto se puede conseguir gracias a los paquetes PEAR y PECL.

PEAR, *The PHP Extension and Application Repository* es un entorno de desarrollo y sistema de distribución para componentes de código PHP. El proyecto PEAR fue fundado por *Stig S. Bakken* en 1999 para promover la reutilización de código que realizan tareas comunes.

El proyecto tiene como metas:

- promover una biblioteca de código bien estructurada.
- mantener un sistema de distribución y mantenimiento de paquetes de código.
- promover un estilo de codificación estándar.

En definitiva consiste en un repertorio bastante amplio de bibliotecas de código PHP que permiten hacer ciertas tareas de manera más rapida y eficiente reutilizando código escrito previamente por otras personas. Generalmente, estas bibliotecas contienen clases en archivos PHP que posteriormente se incluyen en nuestro script PHP y se usan sin muchas complicaciones. Además PEAR proporciona una página web y listas de correo donde dar soporte a la comunidad de usuarios PHP/PEAR.

PECL, *The PHP Extension Community Library*, es un repositorio para extensiones PHP, que proporciona un directorio de todas las extesiones conocidas y facilidades de descarga y desarrollo de nuevas extesiones para PHP.

A continuación se detalla la información sobre los paquetes utilizados en nuestra aplicación, lo que nos ha facilitado la tarea de desarrollo de la misma.

- **Auth**: Proporciona métodos para la creación de un sistema de autenticación usando PHP.
- **Archive_Tar**: Posibilita el uso de archivos `.tar` en PHP. Soporta creación, extracción, adición y listado de archivos `.tar`. El soporte para Gzip estará disponible si PHP fue instalado con la extensión zlib.
- **Console_Getopt**: Analizador de las opciones en la línea de comandos. Es una implementación en PHP de *getopt*[10] y soporta opciones tanto largas como cortas. Es necesario para poder usar el paquete PEAR.

[10] analizador de las opciones de las líneas de comandos para un fácil análisis por parte de los procedimientos del shell y para identificar opciones legales

- **DB**:*Database Abstraction Layer*. Proporciona una capa de abstracción de la base de datos.
- **Date**: Conjunto de clases genéricas para la representación y manipulación de fechas, horas y zonas horarias sin la necesidad de usar *timestamps*[11].
- **Event_Dispatcher**: Este paquete actúa como una tabla de informes. Es usado para notificar información que se considere importante. Es requerido por el paquete LiveUser.
- **Fileinfo**: Las funciones de este módulo de PECL intentan adivinar el tipo de contenido y la codificación de un archivo, buscando ciertas secuencias de bytes *mágicos* en posiciones específicas dentro del fichero. Utiliza la base de datos /etc/magic.
- **HTML_Common**: Este paquete proporciona métodos para manipular atributos y visualizar código HTML. El paquete HTML_QuickForm depende de este paquete.
- **HTML_QuickForm**:Este paquete proporciona métodos para la creación y validación dinámica de formularios HTML.
- **HTTP**: Clase con métodos estáticos que implementan el protocolo HTTP y que posee distintas funcionalidades como cambiar el formato de la fecha o redirecciones HTTP.
- **HTTP_Download**: Proporciona una interfaz para enviar ficheros u otra información relevante a distintos clientes HTTP.
- **HTTP_Header**: Esta clase proporciona métodos para modificar cabeceras HTTP y códigos de estado. Este paquete depende del paquete HTTP.
- **LiveUser**: Conjunto de clases que permiten trabajar con autenticación de usuario y gestión de permisos.
- **LiveUser_Admin**: Este paquete fue creado con la intención de ser usado con el paquete LiveUser. Está compuesto por todas las clases necesarias para administrar los datos usados

[11]En el ámbito de los sistemas Unix, medida universal de tiempo, siendo un número que nos indica la cantidad de milisegundos que han transcurrido desde las 00:00 horas del 1 de enero de 1970 GMT.

por el paquete anterior. Con este paquete seremos capaces de añadir, editar, ver o borrar cosas como: permisos, usuarios, grupos y otras entidades que existen en LiveUser.

- **Mail**: Este paquete define una interfaz para la implementación de sistemas de correo. Proporciona además útiles funciones para el soporte de múltiples sistemas de correo. Depende del paquete Net_SMTP.
- **Net_SMTP**: Proporciona una implementación del protocolo SMTP. Este paquete depende del paquete Net_Socket.
- **Net_Socket**: Interfaz para implementar *sockets*[12] TCP.
- **PEAR**: Es la base del sistema PEAR. Todos los paquetes dependen de él.
- **Text_Password**: Permite crear contraseñas.
- **Var_Dump**: Proporciona métodos para el volcado estructurado de la información sobre una variable.
- **XML_Parser**: Analizador de XML basado en la extensión xml que se intala con PHP. Soporta dos modos de operación básicos: *func* y *event*.

3.1.5. El motor de plantillas Smarty

Smarty Template Engine es un motor de plantillas para el lenguaje de programación PHP. Concretamente, esta herramienta facilita la manera de separar la lógica de la aplicación y el contenido en la presentación.

Las plantillas poseen la extensión `.tpl` y funcionan de una forma distinta a los scripts PHP. La diferencia fundamental, es que PHP es un lenguaje interpretado, mientras que Smarty debe ser compilado.

[12]Socket designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiarse cualquier flujo de datos, generalmente de manera fiable y ordenada.Un socket queda definido por una dirección IP, un protocolo y un número de puerto.

La mejor descripción está en una situación donde la aplicación del programador y la plantilla del diseñador juegan roles diferentes, y en la mayoría de los casos no son la misma persona. Smarty no intenta separar completamente la lógica de la plantilla. No existe un problema entre la lógica y su plantilla, bajo la condición de que esta lógica es estrictamente para su presentación. Se trata de mantener la aplicación lógica fuera de la plantilla de presentación y viceversa. Esto tiene como finalidad, tener un objeto más manipulable y escalable en el futuro.

Las características más destacables de la tecnología Smarty son:

- Es extremadamente rápido.
- No analiza gramaticalmente la plantilla, sólo la compila una vez. Sólo se recompilan aquellas plantillas que fueron modificadas.
- Posibilidad de crear funciones y modificadores de variables personalizadas, de modo que el lenguaje de la plantilla es altamente extensible.
- Sintaxis de etiquetas delimitadoras que facilitan la configuración de la plantilla.
- Las sentencias selectivas como if, elseif, o los bucles son interpretados por el analizador de PHP, de forma que el código de dichas sentencias puede ser todo lo complejo o simple que se desee.
- Permite un anidamiento ilimitado de estas sentencias.
- Es posible incrustar código PHP en la plantilla, aunque no es recomendado.
- Soporte de *caching*[13] incrustado y funciones de manipulación de caché.
- Arquitectura modular, facilidad para añadir *plug-ins*[14].

[13]Término que se refiere a almacenar de forma local en nuestra aplicación datos remotos de forma que se reducen las transferencias de la red.

[14]En informática, aplicación que interactúa con otra aplicación para aportarle una funcionalidad, generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal a la que le añade nuevas utilidades. También conocido con otros términos en inglés como add-on o add-in.

3.1.5.1. Instalación y configuración de *Smarty Template Engine*

La instalación de Smarty es muy sencilla, pero hemos de tener en cuenta algunos aspectos a la hora de montar nuestra aplicación.

En primer lugar, debemos tener en cuenta que necesitamos un servidor web corriendo con PHP 4.0.6 o superior, en nuestro caso servidor Apache 2 corriendo con PHP 5.

Descargamos de la página web de Smarty, **http://smarty.php.net/download.php** la última versión del software y a continuación realizamos lo siguiente:

```
1  ~/usr/local/src tar -zxvf Smarty-2.6.18.tar.gz cd Smarty-2.6.18
2
3  #Listamos el directorio para ver el contenido
4  ~/usr/local/src cd Smarty-2.6.18
5  ~/usr/local/src/Smarty-2.6.18 ls
6  BUGS  ChangeLog  INSTALL  QUICK_START  RELEASE_NOTES  demo  misc  COPYING.lib  FAQ
7  NEWS  README     TODO     libs         unit_test
8  ~/usr/local/src/Smarty-2.6.18 ls libs/
9  Config_File.class.php Smarty_Compiler.class.php  internals  Smarty.class.php  debug.tpl  plugins
10
11 #Creamos el directorio Smarty-2.6.18 en el directorio donde tengamos instalado PHP, en nuestro
       ...caso /usr/local/share/php
12
13 ~/usr/local/share/php sudo mkdir Smarty-2.6.18
14
15 #Y a continuación copiamos todo el directorio libs de Smarty en el nuevo directorio que acabamos
       ...de crear.
16
17 sudo cp -R /usr/local/src/Smarty-2.6.18/libs/ /usr/local/share/php/Smarty-2.6.18
18
19 #Y por último creamos el siguiente enlace simbólico, que más adelante explicaremos por qué es
       ...necesario.
20
21 sudo ln -s /usr/local/share/php/Smarty-2.6.18 /usr/local/share/php/Smarty
```

Figura 3.16: Instalación de *Smarty Template Engine*

Debido a su estructura modular, podemos añadir todas las funcionalidades que necesitemos.

En nuestro caso, para poder darle soporte completo para la Internacionalización es necesario que instalemos el módulo smarty-gettext, lo que nos permitirá extraer las cadenas que queramos traducir de las plantillas de presentación.

Lo descargamos de la siguiente dirección web, **http://sourceforge.net/projects/smarty-gettext** y procedemos de la siguiente manera:

```
1  #Descomprimimos el archivo como hicimos anteriormente con la aplicación principal
2
3  ~/usr/local/src tar smarty-gettext-1.0b1.tar.gz
4  ~/usr/local/src cd smarty-gettext-1.0b1
5  ~/urs/local/src/smarty-gettext-1.0b1 ls
6  COPYING         ChangeLog       README          block.t.php     tsmarty2c.php
7
8  #Ahora solo basta con copiar los ficheros block.t.php y tsmarty2c.php en los directorios
       ...correspondientes.
9  #block.t.php en el directorio \emph{plugins} de Smarty
10 #y el fichero tsmarty2c.php por comodidad lo alojaremos en /usr/local/share/php por motivos que se
       ... explicarán más adelante
11
12 sudo cp /usr/local/src/smarty-gettext-1.0b1/block.t.php /usr/local/share/php/Smarty-2.6.18/plugins
13 sudo cp /usr/local/src/smarty-gettext-1.0b1/tsmarty2c.php /usr/local/share/php
```

Figura 3.17: Instalación de módulos en Smarty

Una vez que tenemos instalados todos nuestros requisitos previos para el uso de Smarty, vamos a pasar a la configuración general para poderlo usar en nuestra aplicación.

En primer lugar, Smarty requiere de cuatro directorios por defecto llamados `templates/`, `templates_c/`, `configs/` y `cache/`. Cada uno de estos directorios son para definir las propiedades de las clases de Smarty.

- **$template_dir**: Nombre del directorio que almacena las plantillas. Si no se proporciona el path, se buscará en '`./templates`' por defecto, es decir, en el mismo directorio del script php que se esté ejecutando en ese momento.
- **$compile_dir**: Nombre del directorio que almacena las plantillas una vez compiladas. El valor por defecto es '`./templates_c`', de modo que si no se especifica un nuevo valor, se buscará en el mismo directorio del script PHP que se esté ejecutando. Es importante que este directorio tenga permiso de escritura para el servidor web. Más adelante se explicará cual es la mejor manera de conceder permisos a este directorio.
- **$config_dir**: Nombre del directorio donde se almacenan los archivos de configuración. Por defecto es '`./configs`'.

- **$cache_dir**: Nombre del directorio donde se cachean las plantillas. Es decir, se buscará en este directorio si se ha cargado previamente una plantilla. Por defecto es el directorio '`./cache`'. Al igual que el directorio $compile_dir, debe tener permisos de escritura para el servidor web.

Es recomendable que estos directorios no estén contenidos en el *DocumentRoot*[15] del servidor por motivos de seguridad.

Dada la estructura de nuestra aplicación, decidimos que tendríamos una estructura de plantillas general, dentro del directorio `_system/` de la misma, y además una estructura de plantillas por cada conferencia que se crea nueva, ajustada a dicha conferencia. Entraremos en más detalle de la organización de la aplicación en la siguiente sección.

Para que la ejecución de la aplicación sea correcta por tanto, debemos ejecutar los siguientes comandos para que se pueda escribir en los directorios `templates_c` y `cache`.

```
1 chown user:group DocumentRoot/CORE/_system/templates_c
2 chmod 770 DocumenRoot/CORE/_system/templates_c
3 chown user:group DocumentRoot/CORE/_system/cache
4 chmod 770 DocumentRoot/CORE/_system/cache
```

Figura 3.18: Configuración de permisos de los directorios requeridos por Smarty

Generalmente, el usuario y el grupo del servidor suelen ser *nobody*, pero para otros sistemas operativos el usuario y el grupo por defecto puede ser *www* o *daemon*. Podremos comprobar nuestro usuario y nuestro grupo del servidor en el fichero *httpd.conf*.

3.1.5.2. Interacción con el código PHP de la aplicación

Una vez vista toda la instalación y configuración de Smarty, es hora de ver cómo interactúa con nuestro código PHP.

[15] Directorio donde se almacenan los documentos web. Directorio a partir del cual cuelga todo el árbol de directorios de nuestra aplicación web.

Supongamos que queremos una cabecera y un pie de página que se repite a lo largo de toda nuestra aplicación. ¿De qué sirve repetir el mismo código en cada uno de nuestros ficheros php? Para evitar esto se utilizan las plantillas Smarty. Supongamos que en nuestra cabecera debe aparecer una barra de navegación que cambia dinámicamente conforme vamos visitando las distintas partes que nos ofrece nuestra aplicación. Esto es posible de una manera muy sencilla con la herramienta Smarty.

Debemos tener una instancia de la clase Smarty en nuestro código PHP. Por otro lado, debemos tener un fichero `.tpl`, con el código que queremos evitar repetir, pero que sin embargo queremos un mismo resultado en todas las páginas de nuestra aplicación.

Desde nuestro script php podremos asignar valores a variables con la función `assign()` que luego se utilizarán en la plantilla. Para poder visualizarla, deberemos llamar a la función `display()`.

A continuación se mostrará un ejemplo, que nos permitirá ver todo esto de una forma más nítida.

```
1 {* Smarty *}
2 Hello, {$name}!
```

Figura 3.19: Ejemplo de fichero `.tpl`: index.tpl

```
1  <?php
2  // Cargamos la librería Smarty
3  require('Smarty.class.php');
4  $smarty = new Smarty;
5  //Debemos tener definidos las variables de los directorios Smarty como se comentó
6  //anteriormente.
7  $smarty->template_dir = $_SERVER['DOCUMENT_ROOT'].'/CORE/_system/templates/';
8  $smarty->compile_dir = $_SERVER['DOCUMENT_ROOT'].'/CORE/_system/templates_c/';
9  $smarty->config_dir = $_SERVER['DOCUMENT_ROOT'].'/CORE/_system/configs/';
10 $smarty->cache_dir = $_SERVER['DOCUMENT_ROOT'].'/CORE/_system/cache/';
11 //Asignamos a la variable 'name' el valor 'Rosario'
12 $smarty->assign('name','Rosario');
13 $smarty->display('index.tpl');
14 ?>
```

Figura 3.20: Ejemplo de interacción de Smarty y PHP

En primer lugar debemos cargar la librería de Smarty, de manera que podamos crear objetos. A continuación es necesario definir los directorios templates, templates_c, configs y cache.

Por último se asigna a la variable 'name' de la plantilla, el valor 'Rosario' y se muestra el resultado con la funcón display(). Con esta forma de programar, no es necesario repetir código si queremos que en muchas de nuestras páginas aparezca la cadena 'Hello" seguido del nombre que deseemos, ya que sólo debemos pasar el valor de la variable 'name'.

3.2. Modelo de la aplicación

Existen tres elementos básicos que conforman el modelo de la aplicación:

Usuarios

Los usuarios son las personas que finalmente utilizarán la aplicación. Podemos distinguir tres tipos de usuarios: usarios adminitradores, usuarios registrados (ponentes) y usuarios no registrados.

Los usuarios administradores podrán acceder a todas las funcionalidades que ofrece el sistema, previa autenticación. Los usuarios no registrados, por el contrario sólo podrán visualizar la información.

Los usuarios registrados son aquellas personas que forman parte de la conferencia. Estos usuarios pueden formar parte de una o varias sesiones o presentaciones. Además pueden tener distintos roles dentro de cada sesión: moderador (*chairman*) o ponentes (*speakers*), e incluso ambas. Cada usuario registrado podrá modificar su perfil, el cual consta de la siguiente información:

- Nombre.
- Apellidos.

- Organización a la que pertenece.
- Posición dentro de la organización.
- Imagen identificativa (opcional).
- País.
- Descripción de su perfil.

Sesiones

Las conferencias se dividen en diferentes sesiones a lo largo de los días en que ésta está vigente. Cada sesión además está compuesta de distintas presentaciones. Cada sesión estará vinculada a un día, una hora y una sala concretas. De cada sesión se almacena la siguiente información:

- Título de la sesión.
- Descripción de la sesión.
- Tipo de sesión : normal, evento especial, etc.
- Localización de la sesión (sala donde dará lugar dicha sesión).
- Categoría de la sesión, de manera que se puedan agrupar las sesiones. Por ejemplo: Wireless, IPv6, etc.
- Ficheros de video que los usuarios pueden descargar.
- Fecha en que tendrá lugar la sesión.
- Hora de inicio y fin de la sesión.
- Moderador de la sesión (*chairman*).

Presentaciones

Cada sesión está formada por distintas presentaciones. Estas presentaciones están vinculadas con uno o más ponentes, los cuales a su vez pueden estar vinculados a una o más presentaciones.

La aplicación ofrece a los usuarios registrados la posibilidad de subir ficheros que contengan las distintas ponencias. Podemos distinguir los papeles de las ponencias (*papers*), y las transparencias (*slides*) que se usan en las mismas.

De cada presentación se almacena la siguiente información:

- Título.
- Resumen (*abstract*).
- Ponentes de la presentación.
- Autores de la presentación.

3.2.1. Jerarquía de Clases y Directorios

La aplicación *Core* cuenta con una jerarquía de clases genéricas que forman la base de la aplicación. Estas clases se alojan en el directorio `_system/classes` de nuestra aplicación.

Figura 3.21: Árbol de directorios de _system

A continuación explicaremos de forma general las funcionalidades que aportan estas clases.

- **TERENA_DB**: Esta clase crea una conexión con el sistema gestor de bases de datos mediante funciones del paquete PEAR::DB. Realiza una conexión singleton, lo que asegura una sola conexión con el mismo, obteniendo siempre el mismo objeto. Se aloja en el fichero TERENA_DB.class.php
- **PEARDB_Exception**: Esta clase se utiliza para empaquetar los posibles errores que puedan suceder durante las conexiones al SGBD.
- **TERENA_Visitor**: Esta clase permite la conexión al sistema a usuarios visitantes, sin la necesidad de que estén autenticados.
- **TERENA_Smarty**: Esta clase crea una instancia de Smarty, inicializando las variables template_dir, compile_dir, y cache_dir, necesarias para la estructura de plantillas de la aplicación, como vimos en la sección 3.2.5.
- **TERENA_Mailer**: Esta clase utiliza las funciones del paquete PEAR::Mail. Inicializa los parámetros necesarios para la configuración del sistema de correo electrónico.
- **TERENA_Exception**: Esta clase empaqueta todos los posibles errores que puedan producirse dentro de nuestra aplicación, capturando las posibles excepciones y mostrando mensajes de error al usuario.

Dentro del directorio `pages` podemos encontrar diferentes clases que representan las distintas páginas que podemos encontrar en nuestra aplicación.

Cada sección muestra un tipo de página distinta, es decir, para la sección Events de la aplicación, existe una clase denominada TERENA_Events_Page que define la estructura de la página. Todas estas clases dependen de una clase base denominada TERENA_Base_Page, la cual define la estruc-

tura base de todas las páginas.

Figura 3.22: Directorio de páginas

En el directorio `dao`, siglas referentes a *Data Access Object*[16], encontramos las clases necesarias para representar de manera lógica los datos de la aplicación almacenados en el SGBD.

El directorio `forms` alberga las clases que representan los distintos formularios utilizados en la aplicación.

Figura 3.23: Directorio de formularios

[16]Componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una Base de datos o un archivo.

3.2.2. Otros directorios

Existen otros directorios en la aplicación que juegan un papel muy importante.

Directorio locale

Como comentábamos en el capítulo 2, para la Internacionalización existe un convenio por el cual debemos utilizar un sistema de directorios para organizar los catálogos de mensajes.

El directorio raíz, de estos subdirectorios se denomina locale, y se ubica en el directorio raíz (CORE) de nuestra aplicación.

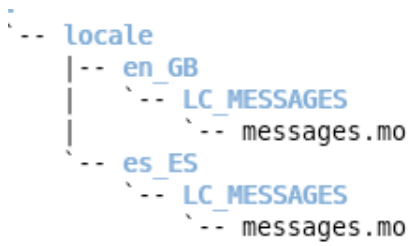


Figura 3.24: Directorio locale

Para que la localización funcione correctamente, debemos indicarle a nuestra aplicación de dónde tomar los catálogos de mensajes. Para ello, existe un script denominado localization.php que le indica a la aplicación el dominio de dichos catálogos.

Además se ha creado una cookie de lenguaje, de modo que si un usuario elige un idioma, la próxima vez que acceda a la aplicación la verá en el idioma elegido. Esta cookie cambia cada vez que el usuario elige un idioma distinto al establecido.

```php
<?php
//Default Language: en_GB

$lang = 'en_GB.utf8';

/*Si la cookie 'lang' no está definida, la definimos con el
*idioma por defecto, el inglés.
*Si el usuario selecciona un idioma, la cookie cambiará.
*/
if (!isset($_COOKIE['lang'])) {
        setcookie('lang', $lang, time()+60*60*24*30, '/');
        if(isset($_GET['lang']))
        {
                $lang = $_GET['lang'].".utf8";
                setcookie('lang','', time()-3600, '/');
                setcookie('lang', $lang, time()+60*60*24*30, '/');
        }
/*Si la cookie ya existía, y el usuario elige otro idioma,
*se procede de manera análoga.
*/
}else{
        $lang = $_COOKIE['lang'];
        if(isset($_GET['lang']))
        {
                $lang = $_GET['lang'].".utf8";
                setcookie('lang','', time()-3600, '/');
                setcookie('lang', $lang, time()+60*60*24*30, '/');
        }
}
putenv("LC_ALL=$lang");
setlocale(LC_ALL, $lang);
$path=$_SERVER['DOCUMENT_ROOT']."/CORE/locale";

//Seleccionamos el dominio del catálogo 'messages'

bindtextdomain("messages", $path);
textdomain("messages");
?>
```

Figura 3.25: Script de localización: *localization.php*

Directorio events

Este directorio es bastante importante en la estructura de nuestra aplicación. En él se albergan todos los directorios de los distintos eventos.

Cada conferencia se identifica como un evento. Todos los directorios de eventos poseen la misma estructura, la cual es tomada de un directorio llamado plantilla, que se proporciona con la aplicación.

3.2.3. Estructura de las Bases de Datos

La aplicación *Core* posee distintas bases de datos con diferentes cometidos, los cuales pasamos a explicar a continuación.

En primer lugar podemos encontrar la base de datos **Events**. Esta base de datos alberga los nombres de las diferentes conferencias de nuestro portal. Posee una sola tabla donde se almacena el nombre de la conferencia, las fechas de inicio y fin, y la localización (ciudad donde se celebra la conferencia en cuestión).

Esta BD[17] nos servirá entre otras cosas para mostrar una lista con todas las conferencias posibles, diviéndolas en Eventos Próximos y Eventos Pasados, ofreciendo al usuario la posibilidad de visualizar información sobre conferencias ya pasadas.

Columna	Tipo de dato	No Nulo	Predeterminado	Acciones	
id_event	integer	NOT NULL	nextval('events_id_event_seq'::regclass)	Modificar	Eliminar
event_name	text	NOT NULL		Modificar	Eliminar
short_name	character varying	NOT NULL		Modificar	Eliminar
start_date	date			Modificar	Eliminar
end_date	date			Modificar	Eliminar
location	character varying	NOT NULL		Modificar	Eliminar

Figura 3.26: Tabla *events* de la BD Events

Como podemos observar, existe además de los campos mencionados anteriormente, un campo llamado short_name, que más adelante se explicará su cometido.

Cada conferencia poseerá su propia BD, que se generará automáticamente mediante el **Script de Generación de Eventos**, que toma como plantilla una estructura que se aplica a todas las conferencias.

[17] Base de Datos

A continuación se explica la estructura de esta plantilla:

- Tabla **persons**: En ella almacenamos los datos personales de los usuarios.

Columna	Tipo de dato	No Nulo	Predeterminado	Acciones	
person_id	integer	NOT NULL	nextval(('"persons_person_id_seq"'::text)::regclass)	Modificar	Eliminar
fname	character varying(80)	NOT NULL	'::character varying	Modificar	Eliminar
lname	character varying(80)	NOT NULL	'::character varying	Modificar	Eliminar
position	character varying(50)			Modificar	Eliminar
org	character varying(255)			Modificar	Eliminar
zip	character varying(10)			Modificar	Eliminar
country	character varying(3)			Modificar	Eliminar
phone	character varying(15)			Modificar	Eliminar
fax	character varying(15)			Modificar	Eliminar
mobile	character varying(15)			Modificar	Eliminar
email	character varying(150)	NOT NULL		Modificar	Eliminar
profile	text			Modificar	Eliminar
address	character varying(255)			Modificar	Eliminar
city	character varying(90)			Modificar	Eliminar
updated	timestamp(0) without time zone	NOT NULL	now()	Modificar	Eliminar
title	character varying(15)			Modificar	Eliminar
gender	character(1)			Modificar	Eliminar

Figura 3.27: Tabla *persons*

- Tabla **accounts**: En esta tabla se almacenan datos relativos a la cuenta del usuario, como la contraseña (encriptada bajo el algoritmo MD5[18], o el nivel de usuario. Este nivel se fija para determinar el tipo de usuario: si es administrador o no.

[18] Acrónimo de Message Digest 5, es un algoritmo de reducción criptográfico de 128 bits. La codificación del MD5 de 128 bits es representada típicamente como un número de 32 dígitos hexadecimal.

Columna	Tipo de dato	No Nulo	Predeterminado	Acciones
person_id	integer	NOT NULL		Modificar Eliminar
passwd	character(32)	NOT NULL		Modificar Eliminar
level	smallint	NOT NULL	1	Modificar Eliminar
token	character(32)	NOT NULL	0	Modificar Eliminar
passwd_sent	timestamp(0) without time zone			Modificar Eliminar

Figura 3.28: Tabla *accounts*

Esta tabla junto con la de persons se relacionan de la siguiente manera:

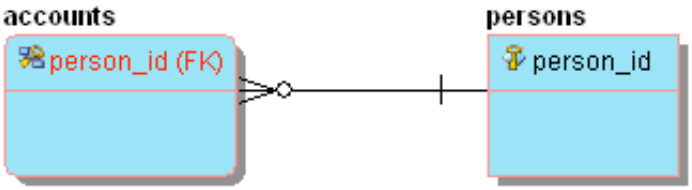


Figura 3.29: Modelo ERD: tablas *accounts* y *persons*

- Tabla **presentations**: En esta tabla se almacenan datos relativos a las presentaciones, como puede ser el título, un breve resumen (abstract) o los autores.

Columna	Tipo de dato	No Nulo	Predeterminado	Acciones
pres_id	integer	NOT NULL	nextval(('"presentations_pres_id_seq"'::text)::regclass)	Modificar Eliminar
title	character varying	NOT NULL		Modificar Eliminar
abstract	text			Modificar Eliminar
updated	timestamp(0) without time zone	NOT NULL	now()	Modificar Eliminar
authors	text			Modificar Eliminar

Figura 3.30: Tabla *presentations*

- Tabla **roles**: En esta tabla se almacenan los distintos roles que pueden jugar los usuarios.

- Tabla **files**: Esta tabla se usa para almacenar los ficheros binarios que se utilizan en la aplicación, así como datos relativos a los mismos. Por ejemplo el tamaño, el tipo de fichero (imagen, presentación pdf, etc), el nombre o el propietario del mismo.

Columna	Tipo de dato	No Nulo	Predeterminado	Acciones	
role_id	integer	NOT NULL	nextval(('"roles_role_id_seq"'::text)::regclass)	Modificar	Eliminar
name	character varying	NOT NULL		Modificar	Eliminar

Figura 3.31: Tabla *roles*

Columna	Tipo de dato	No Nulo	Predeterminado	Acciones	
file_id	integer	NOT NULL	nextval(('"files_file_id_seq"'::text)::regclass)	Modificar	Eliminar
object_id	oid	NOT NULL		Modificar	Eliminar
filesize	integer	NOT NULL	0	Modificar	Eliminar
mimetype	character varying(255)	NOT NULL		Modificar	Eliminar
filename	character varying(255)	NOT NULL		Modificar	Eliminar
updated	timestamp(0) without time zone	NOT NULL	now()	Modificar	Eliminar
active	boolean	NOT NULL	true	Modificar	Eliminar
owner_id	bigint			Modificar	Eliminar

Figura 3.32: Tabla *files*

- Tabla **filetypes**: Esta tabla almacena el nombre del tipo de fichero pero desde un punto de vista lógico en la aplicación. Es decir, tipos de ficheros creados para saber lo que se está manejando en cada momento. Estos tipos definidos son: Paper, Person picture, Extenden abstract, Slide y Housekeeping Slide.

Columna	Tipo de dato	No Nulo	Predeterminado	Acciones	
type_id	integer	NOT NULL	nextval(('"filetypes_type_id_seq"'::text)::regclass)	Modificar	Eliminar
name	character varying(50)	NOT NULL		Modificar	Eliminar

Figura 3.33: Tabla *filetypes*

- Tabla **pres_people**: Esta tabla relaciona a los usuarios con las presentaciones y los roles, es decir, un usuario forma parte de una presentación con un determinado rol.
- Tabla **person_files**: Esta tabla relaciona a los usuarios con los ficheros y los tipos de fichero definidos.
- Tabla **pres_files**: Esta tabla relaciona las presentaciones con los ficheros que se asocian a ella.
- Tabla **locations**: Esta tabla almacena las distintas localizaciones que puede tener una sesión, es decir, las distinta salas en las que se puede celebrar una sesión.

Columna	Tipo de dato	No Nulo	Predeterminado	Acciones	
location_id	integer	NOT NULL	nextval(('"locations_location_id_seq"'::text)::regclass)	Modificar	Eliminar
name	character varying	NOT NULL		Modificar	Eliminar
streaming_url	character varying			Modificar	Eliminar
comments	text			Modificar	Eliminar
abbr	character varying(5)			Modificar	Eliminar
capacity	smallint			Modificar	Eliminar

Figura 3.34: Tabla *locations*

- Tabla **timeslots**: Esta tabla almacena los horarios de los distintos días de la conferencia.

Columna	Tipo de dato	No Nulo	Predeterminado	Acciones	
timeslot_id	integer	NOT NULL	nextval('timeslots_timeslot_id_seq'::regclass)	Modificar	Eliminar
start_time	timestamp(0) without time zone	NOT NULL		Modificar	Eliminar
end_time	timestamp(0) without time zone	NOT NULL		Modificar	Eliminar

Figura 3.35: Tabla *timeslots*

- Tabla **sessions**: Esta tabla almacena toda la información relevante de una sesión.

Columna	Tipo de dato	No Nulo	Predeterminado	Acciones	
sess_id	integer	NOT NULL	nextval('sessions_sess_id_seq'::regclass)	Modificar	Eliminar
type_id	integer	NOT NULL	1	Modificar	Eliminar
title	character varying(255)	NOT NULL		Modificar	Eliminar
description	text			Modificar	Eliminar
location_id	smallint			Modificar	Eliminar
cat_id	smallint			Modificar	Eliminar
locked	boolean	NOT NULL	false	Modificar	Eliminar
updated	timestamp(0) without time zone	NOT NULL	now()	Modificar	Eliminar
timeslot_id	integer			Modificar	Eliminar
videofile	character varying			Modificar	Eliminar
attendees	integer			Modificar	Eliminar
comments	text			Modificar	Eliminar

Figura 3.36: Tabla *sessions*

A continuación se muestra el modelo ERD que muestra como el concepto de la sesión abarca todos los demás, estando todos relacionados con ella.

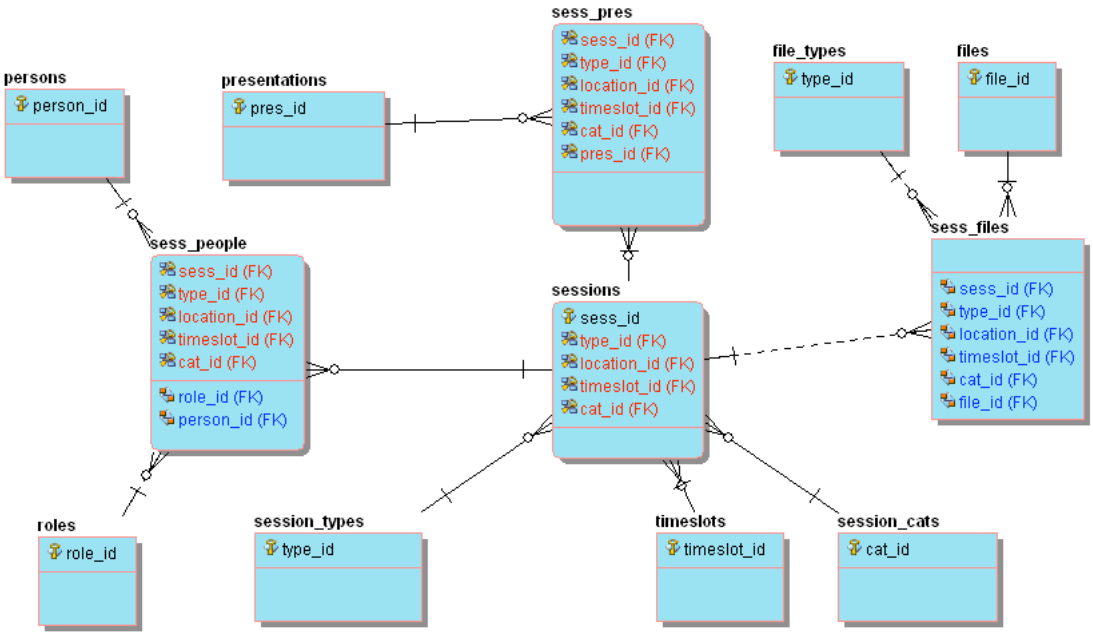


Figura 3.37: Modelo ERD: sessions, presentations, people, timeslots

También se muestra el modelo ERD que relaciona las tablas pres_people, person_files y pres_files.

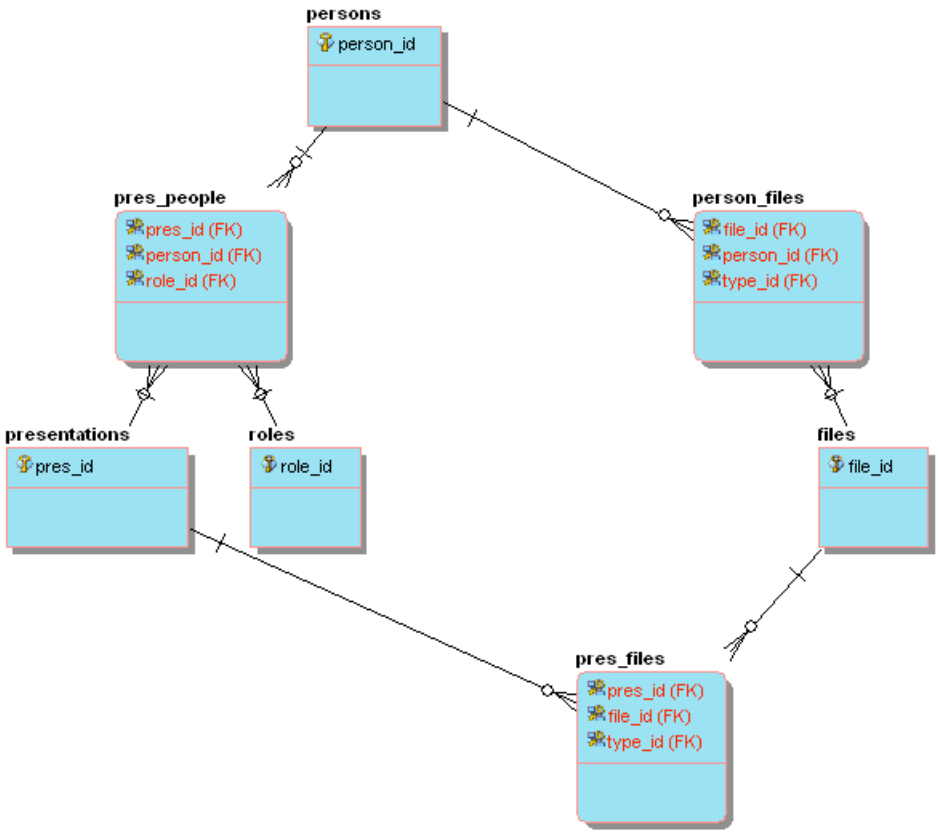


Figura 3.38: Modelo ERD: pres_people, person_file y pres_files

3.2.4. Script de Generación de Eventos

Como se comentó anteriormente, *Core* trabaja con el concepto de evento. Cada evento posee una jerarquía de directorios propia, así como una base de datos propia.

El *Script de Generación de Eventos* nace con la finalidad de automatizar la tarea de crear cada uno de estos eventos. Es un script escrito en Bash, que genera los directorios y la base de datos necesarios para albergar todo un evento.

A la hora de crear este script, se ha tenido en cuenta la fiabilidad y la tolerancia a los fallos. Es decir, si se produce un fallo en algún momento de la ejecución del script, se realizarán las funciones necesarias para restaurar el sistema y dejarlo como estaba antes de ejecutar el script y se produjera el fallo.

Se definen las siguientes funciones:

```
#Create the db for the congress
function createDb {
        echo "Creating the congress' database"
        createdb -h localhost -p 5432 -U admin -q $1
        if [ $? != 0 ]
        then
                 echo "There was an error while creating the database {$1}"
                exit 1
        else
                echo "OK. DB created"
        fi
}
function undoCreateDb {
        dropdb -h localhost -p 5432 -U admin -q $1
}
function insertTemplate {
        echo "Importing template to the database"
        psql -h localhost -p 5432 -U admin -q $1 < plantilla.sql
        if [ $? != 0 ]
        then
                echo "There was an error while importing the template"
                return 2
        else
                echo "OK. Template DB inserted"
      fi
}
```

Figura 3.39: Funciones generadoras y modificadoras de la BD de la conferencia

Como podemos observar, existen tres funciones modificadoras de la base de datos de cada evento o conferencia.

La función **createDb** no es más que un envoltorio de la función createdb que proporciona PostgreSQL. Debemos especificar el host donde se ubica nuestro SGBD, el puerto por donde debemos leer, (el de PostgreSQL es el puerto 5432), el usuario que será el propietario de la base de datos y el nombre de la base de datos que se pasa como parámetro a la función.

Por otro lado está la función **undoCreateDb**, que se utiliza para deshacer la función anterior en caso de que se produzca algún error.

La función **insertTemplate** se encarga de establecer la estructura de la BD, ejecutando los comandos SQL ubicados en el fichero `plantilla.sql`. Es necesario que en el SGBD exista un usuario denominado `admin`, pues se ha establecido así en nuestra aplicación.

```
1  function insertEventIndex {
2          echo "Creating index in the Events DB"
3          comando="INSERT INTO events (event_name, short_name, start_date, end_date, location)
               ...VALUES ('"$1"', '"$2"', '"$3"', '"$4"', '"$5"')"
4          psql -h localhost -p 5432 -U admin -d Events -c "$comando"
5          if [ $? != 0 ]
6          then
7                 echo "There was an error while creating index in the Events DB."
8                 return 3
9          fi
10 }
11 function createShortName {
12        #Create the acronym of the congress to create the directory with this name
13        WSP_IFS=$'\x20'
14        for i in $1
15        do
16                 name=$name${i:0:1}
17         done
18         #Then we translate it to lowercase
19         name="`echo $name|tr '[A-Z]' '[a-z]'`"
20         #append the year to the short name
21         name=$name$year
22 }
```

Figura 3.40: Funciones modificadoras de la BD Events

La función **insertEventIndex** se encarga de añadir la nueva conferencia a la base de datos Events. A esta función se le pasan cinco parámetros que son: nombre del evento (event_name), nombre corto (short_name), fecha de inicio (start_date), fecha de finalización, (end_date) y localización del evento (location).

La función **createShortName** se encarga de crear el nombre corto de la conferencia. Este nombre se utilizará como nombre del directorio que albergará la conferencia. El nombre corto es en realidad el acrónimo del nombre largo de la conferencia, seguido del año en que se realiza la conferencia.

```
1  function createDirectory {
2        echo "Creating congress' directory"
3        cp -R ../plantilla ../events/$1 > /dev/null
4        #todo lo que mande a dev/null no hace nada. Todo lo que redirijamos a la de dev/null va a la
             ... nada
5        if [ $? != 0 ]
6        then
7                echo "An error occured while creating the congress' directory"
8                return 4
9        fi
10       echo "Changing permissions"
11       chmod -R 755 ../events/$1
12       chown daemon:daemon ../events/$1/core/templates_c
13       chmod 770 ../events/$1/core/templates_c
14       chown daemon:daemon ../events/$1/core/cache
15       chmod 770 ../events/$1/core/cache
16 }
17 function undoCreateDirectory {
18         rm -R ../events/$1/
19 }
20 function replaceNameFromTemplate {
21         echo "Replacing congress template name to definitive name"
22         for i in $(grep -R -l tnc2006 ../events/$1)
23         do
24                 perl -pi -e 's/tnc2006/'${1}'/g' $i
25         done
26 }
```

Figura 3.41: Funciones generadoras y modificadoras del directorio de la conferencia

Por último están las funciones generadoras del directorio de la conferencia.

La función **createDirectory** crea el directorio de la conferencia desde la plantilla. A continuación le cambia los permisos, para que puedan ser leídos los subdirectorios y scripts, y además da permiso de escritura al usuario *daemon* de los directorios `templates_c` y `cache` de Smarty, como vimos en la

sección 3.2.5. La función **undoCreateDirectory** deshace la función anterior en caso de que se haya producido algún error.

La función **replaceNameFromTemplate**, reemplaza el nombre de la conferencia plantilla por el nombre de la nueva conferencia en todos los scripts y directorios donde debe aparecer.
Por último se muestra el resto del script, indicando el orden en que se ejecutan las funciones una vez que es llamado mediante `./generate.sh`.

```
echo "Welcome to the Core Congresses' Generation Script"
echo "Please insert congress' name"
        read -e long_name
echo "Insert congress' starting date (YYYY-MM-DD)"
        read start_date
echo "Insert congress' ending date (YYYY-MM-DD)"
       read end_date
echo "Please insert congress' city location"
       read city_location
echo "Please insert congress' country location"
        read country_location
        location="$city_location,$country_location"
        year=${start_date%"-"*} year=${year%"-"*}
#Create the short name
createShortName "$long_name"
#Check the name isn't repeat
        sql="SELECT short_name FROM events WHERE event_name='"${name}"'"
        repeat="`psql -h localhost -p 5432 -U admin -d Events -c "$sql" -t`"
        if [ -n "$repeat" ]
        then
                echo "Please insert another name"
                read long_name
                createShortName "$long_name"
                sql="SELECT short_name FROM events WHERE event_name='"${name}"'"
                repeat="`psql -h localhost -p 5432 -U admin -d Events -c "$sql" -t`"
        fi
#Create the database
createDb "$name"
#Import template to the database
insertTemplate "$name"
if [ $? != 0 ]
then
         undoCreateDb "$name"
         exit 1
fi
insertEventIndex "$long_name" "$name" "$start_date" "$end_date" "$location"
 if [ $? != 0 ]
then
         undoCreateDb "$name"
         exit 1
fi
createDirectory "$name"
if [ $? != 0 ]
then
        undoCreateDb "$name"
        exit 1
fi
replaceNameFromTemplate "$name"
if [ $? != 0 ]
then
        undoCreateDb "$name"
        undoCreateDirectory "$name"
        exit 1
fi
echo "...:.:...:.:..Done!..:.:...:.:..."
```

Figura 3.42: Script de Generación de Eventos

Capítulo 4

Conclusiones

Este proyecto ha sido decisivo para hacerme ver el esfuerzo y la madurez necesaria para afrontar una tarea de esta envergadura.

Un primer aspecto observado al realizar este proyecto, ha sido la necesidad de documentarse e investigar antes de realizar ninguna implementación. Si bien es importante llevar a la práctica los conceptos asimilados, más importante es realizar un estudio exhaustivo de las tecnologías que se van a utilizar, para así poder tomar las decisiones correctas cuando fuese necesario.

El proyecto que aquí se presenta es una aplicación compleja y robusta, que ha requerido mucho tiempo y esfuerzo. El hecho de trabajar con distintos *frameworks* de desarrollo me ha aportado, además de una visión global de lo que supone un proyecto de esta envergadura, es el conocimiento de nuevas tecnologías y la posibilidad de aprovechar herramientas que han sido creadas y testadas por otras personas en favor de la *comunidad* de desarrolladores, en nuestro caso de la tecnología PHP.

Quisiera destacar la necesidad de la *Internacionalización*. Además del simple hecho de que los usuarios prefieren trabajar en su lengua nativa como se comentaba en el capítulo 2, la Internacio-

nalización favorece el crecimiento de las empresas, dado que la dimensión empresarial depende de la dimensión del mercado al que accede o se dirige y tiene una relación de doble flujo con el conocimiento, tanto tecnológico, como de gestión.

Hace que crecimiento, estrategia y estructura deban evolucionar paralelamente incorporando una mayor orientación al cliente, y una posición que permite y obliga a absorber nuevas capacidades de innovación. Ya que los clientes, usuarios de distintos países y con diferentes culturas necesitan servicios adaptados correctamente para procesar información usando su idioma de origen, su sistema de escritura, su sistema de medida, sus calendarios y otras reglas y convenciones culturales.

La herramienta GNU Gettext es una herramienta eficaz y sencilla de utilizar, pero por su sencillez, la tarea de Internacionalización de una aplicación de esta envergadura resulta una tarea bastante tediosa, especialmente al tener que adaptar la aplicación siguiendo una serie de reglas establecidas. Quizás debería trabajarse la forma de automatización de esta herramienta, con respecto a las aplicaciones que se pretenden internacionalizar.

Me gustaría destacar de este proyecto, la capacidad de abstracción y de seguridad en mí misma que he conseguido. Enfrentarse a un proyecto real, y a una serie de problemas que aparentemente no tenían solución, me ha servido para madurar tanto intelectual como profesionalmente. Un aspecto fundamental y decisivo en este proceso de maduración que he sufrido ha sido la posibilidad de trabajar con otras personas de diferentes países, lo cual enriquece mucho. Hace ver que es muy importante saber trabajar en equipo para que todo un proyecto salga adelante.

Hoy día este proyecto se sigue desarrollando, intentando mejorarlo y adaptándolo a nuevas tecnologías. Como se comentó en la introducción, esta aplicación pretende ser un sistema robusto para la gestión de conferencias organizadas por RedIRIS, y una de las mejoras más importantes que se

plantean es modificar la forma de autenticación para usar PAPI[1].

Por último me gustaría recalcar el papel que el Software Libre ha jugado en el desarrollo de este proyecto. Utilizar tecnologías libres, ha aportado una base de conocimiento y fuente de información inagotable, sin la que no podría haberse realizado este proyecto.

[1]PAPI es un sistema desarrollado por RedIRIS que provee un control de acceso a recursos de información electrónica restringidos. Intenta mantener la autenticación como un aspecto local de la organización a la que pertenezca el usuario, mientras que da a los proveedores de información total control sobre los recursos que ofertan.

Capítulo 5

Anexos

5.1. Manual de Instalación

La instalación de *Core* es compleja, especialmente porque requiere la previa instalación de unos requisitos mínimos que hay que tener en cuenta (sección 3.2).

Instalación de los requisitos básicos

En primer lugar debemos instalar el servidor Apache 2 con soporte SSL, tal y como se explicó en la sección 3.2.1.

A continuación deberemos instalar el sistema gestor de base de datos PostgreSQL, tal y como se planteó en la sección 3.2.2.

La instalación de PHP 5 llega en este punto, teniendo en cuenta que debemos indicar entre otras opciones `--with-apxs2` y `--with-pgsql`, tal y como se expuso en la sección 3.2.3.

Una vez instalado PHP, podemos proceder a instalar los paquetes PEAR necesarios. Para ello utilizaremos el siguiente comando autenticándonos como usuario root del sistema:

```
1 sudo pear install nombre-paquete
```

Figura 5.1: Ejecución del comando *pear*

El nombre-paquete corresponderá con el nombre del paquete que deseemos instalar. Si el paquete disponible es una versión beta o alpha, deberemos añadir la cadena -beta o -alpha al nombre del paquete. Por ejemplo, para instalar el paquete I18N, cuya versión es beta deberemos ejecutar: `sudo pear install I18N-beta`.

Para los paquetes PECL, se procede de la misma manera, pero ejecutando el comando `pecl install`.

Por último debemos instalar el motor de plantillas de Smarty. Para ello, debemos proceder como se explicó en la sección 3.2.5.

Instalación de Core: Sistema Colaborativo para la Gestión de Conferencias

Una vez instalados todos los requisitos básicos de la aplicación, pasamos a explicar la instalación de *Core*.

En primer lugar debemos tener localizado nuestro DocumentRoot, en nuestro caso se encuentra en /usr/local/apache2/htdocs. Esta variable se puede ver y modificar en el fichero de configuración de apache httpd.conf.

A continuación copiamos todo el directorio de nombre CORE (suminisitrado en el CD), a nuestro DocumentRoot. Es muy importante que el directorio CORE sea accesible tanto en lectura como en ejecución.

```
#Para realizar las tareas que a continuación se describen, es necesario ser superusuario, root.
#por tanto podremos autenticarnos con 'sudo su' o anteponer 'sudo' al comando que queramos
     ...ejecutar

#Copiar el directorio CORE al DocumentRoot.

cp -R CORE /usr/local/apache2/htdocs

#Cambiar los permisos recursivamente.

chmod -R 755 /usr/local/apache2/htdocs/CORE
chown daemon:daemon /usr/local/apache2/htdocs/CORE/_system/templates_c
chmod 770 /usr/local/apache2/htdocs/CORE/_system/templates_c
chown daemon:daemon /usr/local/apache2/htdocs/CORE/_system/cache
chmod 770 /usr/local/apache2/htdocs/CORE/_system/cache
```

Figura 5.2: Cambio de permisos en CORE

Es muy importante que los directorios Smarty templates_c y cache tengan además permiso de escritura para el usuario *daemon* y su grupo (usuario que ejecuta el servidor), por lo que deberemos modificar también estos permisos. Como se comentó anteriormente, este usuario se puede consultar y modificar en el fichero de configuración httpd.conf de Apache.

Cada directorio pertenciente a un evento también posee unas plantillas Smarty, por lo que debemos cambiar los permisos también a cada directorio templates_c y cache de cada evento.

En los eventos que se proporcionan en el CD como ejemplo, deberemos cambiar, tal y como se muestra arriba, los permisos a los directorios 'events/tnc2006/core/templates_c' y 'events/tnc2006/core/cache'. Y así con todos los eventos proporcionados.

Debemos tener en cuenta, que dependiendo de cómo tengamos montado nuestro sistema, las rutas hacia directorios y scripts incluidos desde otros ficheros php, puden diferir de las que se proporcionan en el CD, aunque se garantiza que si se han seguido los pasos indicados en esta documentación, esas rutas no necesitan modificación.

Con esto ya tenemos nuestra aplicación lista. Ahora sólo nos queda saber cómo funciona y para

ello, en la siguiente sección se proporciona un Manual de Usuario totalmente ilustrado.

5.2. Manual de Usuario

Este manual se divide en dos partes. Por una parte se mostrarán las funcionalidades permitidas a un usuario administrador y por otra las funcionalidades de un usuario registrado, que evidentemente son más restringidas.

A continuación se muestra la página de bienvenida:

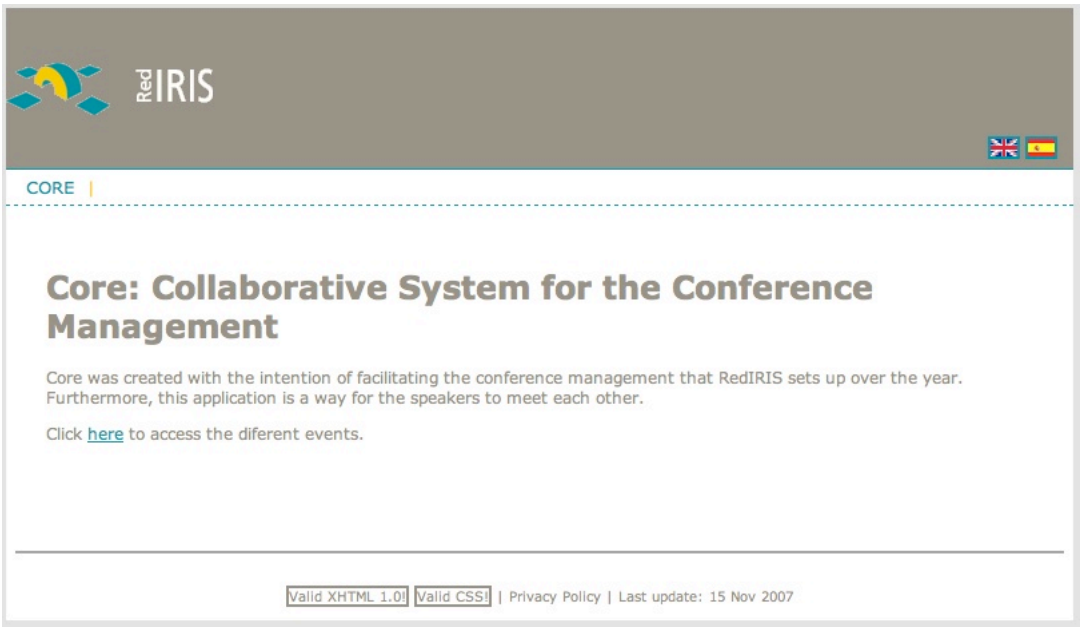


Figura 5.3: Página de bienvenida de Core

Para entrar en la página que nos muestra los eventos debemos pulsar en el enlace 'here' que aparece debajo de la introducción. Cuando pulsemos se nos abrirá una página como la siguiente:

Figura 5.4: Página principal de eventos. Eventos próximos

En esta página se muestran las próximas conferencias dependiendo de la fecha de las mismas con respecto a la fecha actual. En el momento en que la fecha de una conferencia venza, dicha conferencia pasará a formar parte de los eventos pasados, los cuales puede ser accedidos desde esta página.

Pulsando sobre el nombre de la conferencia accederemos a la página principal de la conferencia desde la que podremos acceder a su programa.

En el programa pueden visualizarse tanto los ponentes como las ponencias de cada sesión, pero por defecto se visualizará el nombre de los ponentes, debido a que el nombre de las ponencias es a veces demasiado largo, lo que hace que el programa no se vea de manera clara.

Para identificarnos en el sistema, pulsamos en el enlace "speakers can login". Dependiendo del nivel del usuario, éste podrá acceder a todas las funcionalidades del sistema o sólo a las que le correponden como un usuario registrado sin privilegios.

Figura 5.5: Página de eventos pasados.

Usuarios Administradores

Una vez nos identificamos como usuarios administradores (cuyo nivel de permisibilidad debe ser igual o superior a 5), aparecerán las posibles modificaciones que se le puede dar a una página determinada. A continuación se muestran diferentes capturas de las páginas que puede modificar un administrador. Estas páginas son idénticas a las que puede visualizar un usuario registrado sin privilegios de administración, es decir, lo visualiza sin los comandos propios de un administrador.

Tanto los usuarios no registrados, como los registrados verán la misma información, con la salvedad de que los usuarios registrados podrán modificar su perfil.

En el modo administrador, es muy fácil modificar el programa de una conferencia. En la figura 5.13 se muestran los detalles de administrador en la página del programa de la conferencia. Como podemos observar, aparecen *checkboxes* en cada una de las celdas del horario. Éstos se utilizan para intercambiar dos sesiones. Podemos verlo en la figura 5.14.

Capturas

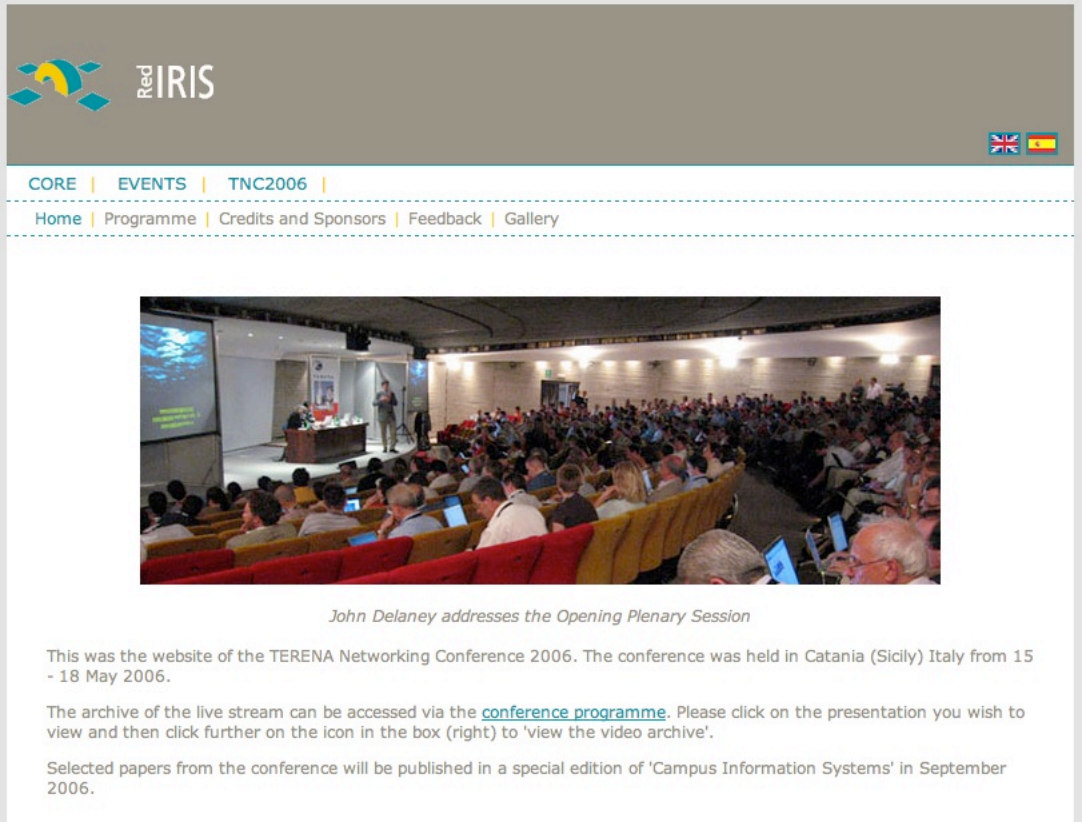
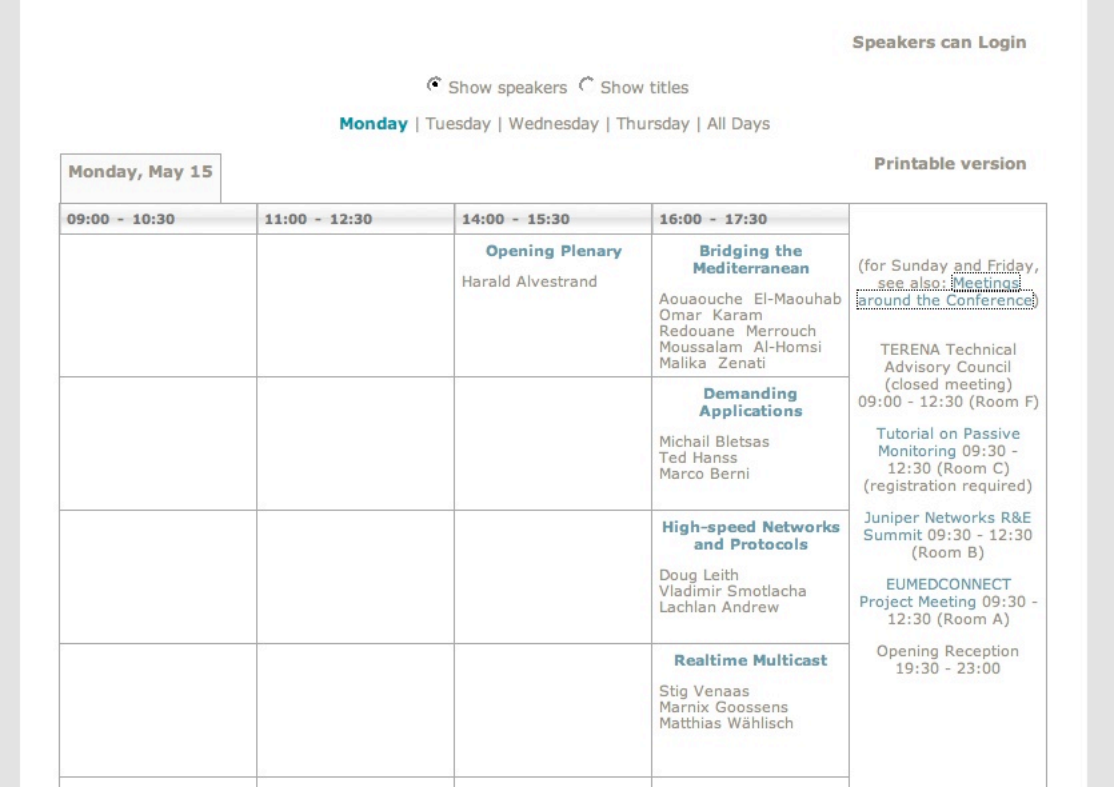


Figura 5.6: Página principal de una conferencia.



Speakers can Login

Show speakers Show titles

Monday | Tuesday | Wednesday | Thursday | All Days

Monday, May 15

Printable version

09:00 - 10:30	11:00 - 12:30	14:00 - 15:30	16:00 - 17:30	
		Opening Plenary Harald Alvestrand	**Bridging the Mediterranean** Aouaouche El-Maouhab Omar Karam Redouane Merrouch Moussalam Al-Homsi Malika Zenati	(for Sunday and Friday, see also: Meetings around the Conference) TERENA Technical Advisory Council (closed meeting) 09:00 - 12:30 (Room F) Tutorial on Passive Monitoring 09:30 - 12:30 (Room C) (registration required) Juniper Networks R&E Summit 09:30 - 12:30 (Room B) EUMEDCONNECT Project Meeting 09:30 - 12:30 (Room A) Opening Reception 19:30 - 23:00
			Demanding Applications Michail Bletsas Ted Hanss Marco Berni	
			High-speed Networks and Protocols Doug Leith Vladimir Smotlacha Lachlan Andrew	
			Realtime Multicast Stig Venaas Marnix Goossens Matthias Wählisch	

Figura 5.7: Programa de una conferencia.

Figura 5.8: Página de *login*.

Figura 5.9: Lista de todas las sesiones y las correspondientes presentaciones.

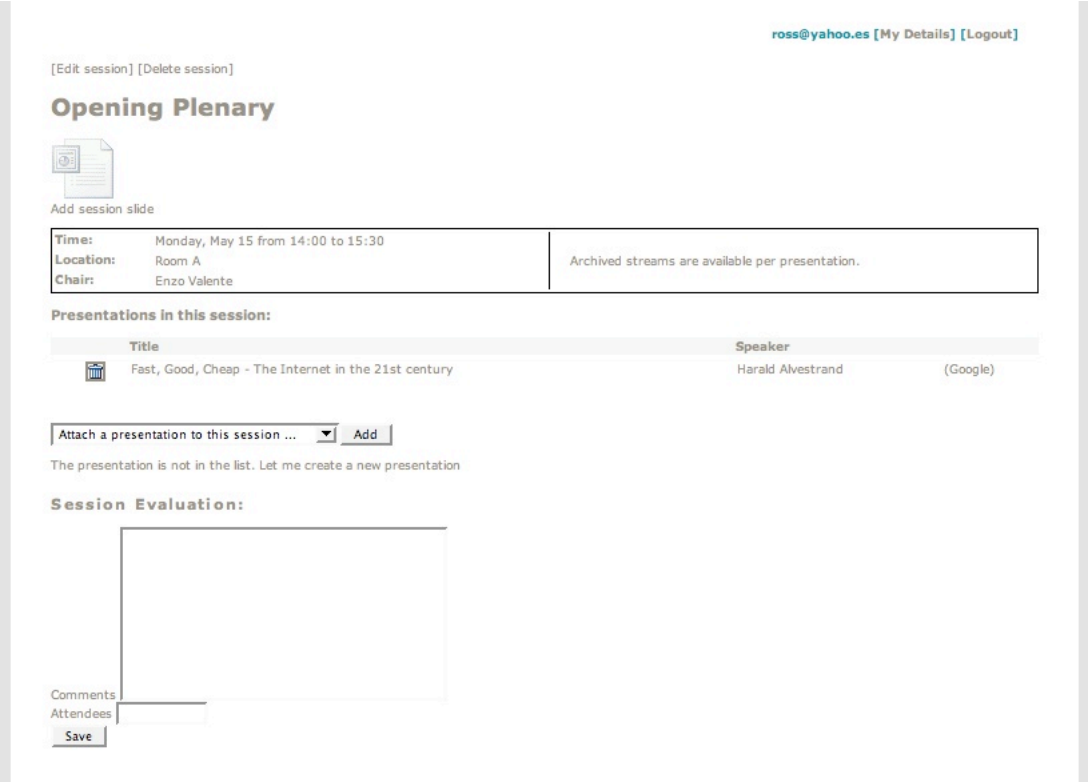


Figura 5.10: Detalles de una sesión. Modo administrador.

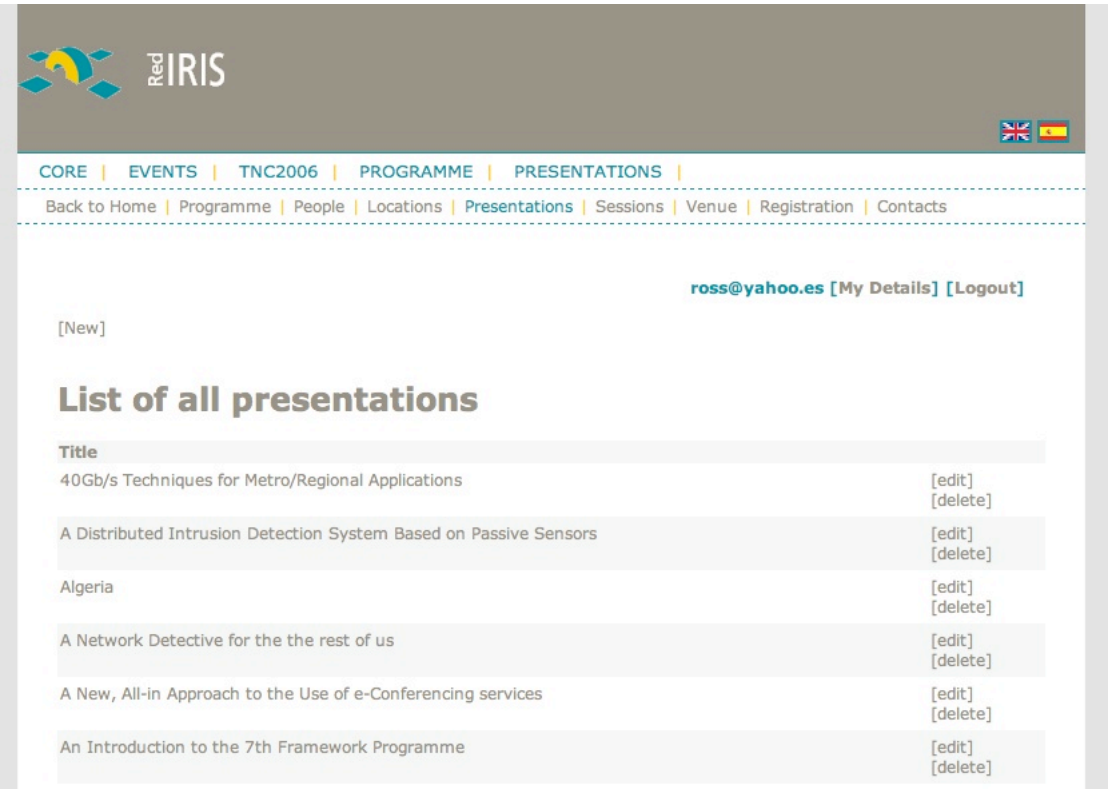


Figura 5.11: Lista de todas las presentaciones. Modo administrador.

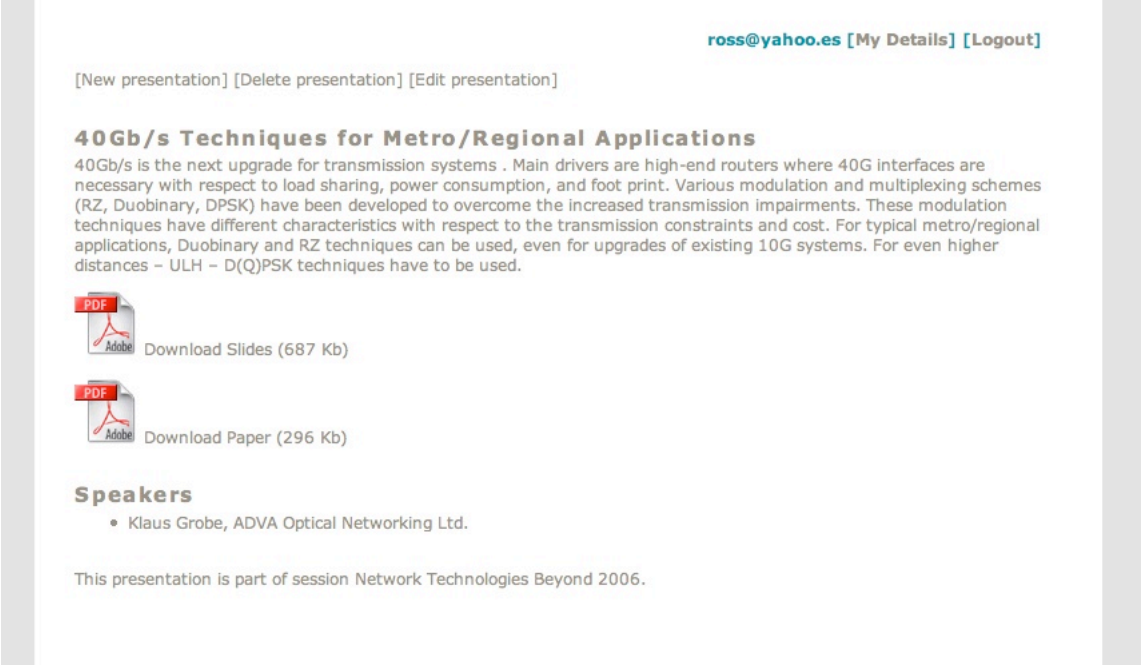


Figura 5.12: Detalles de una presentación. Modo administrador.

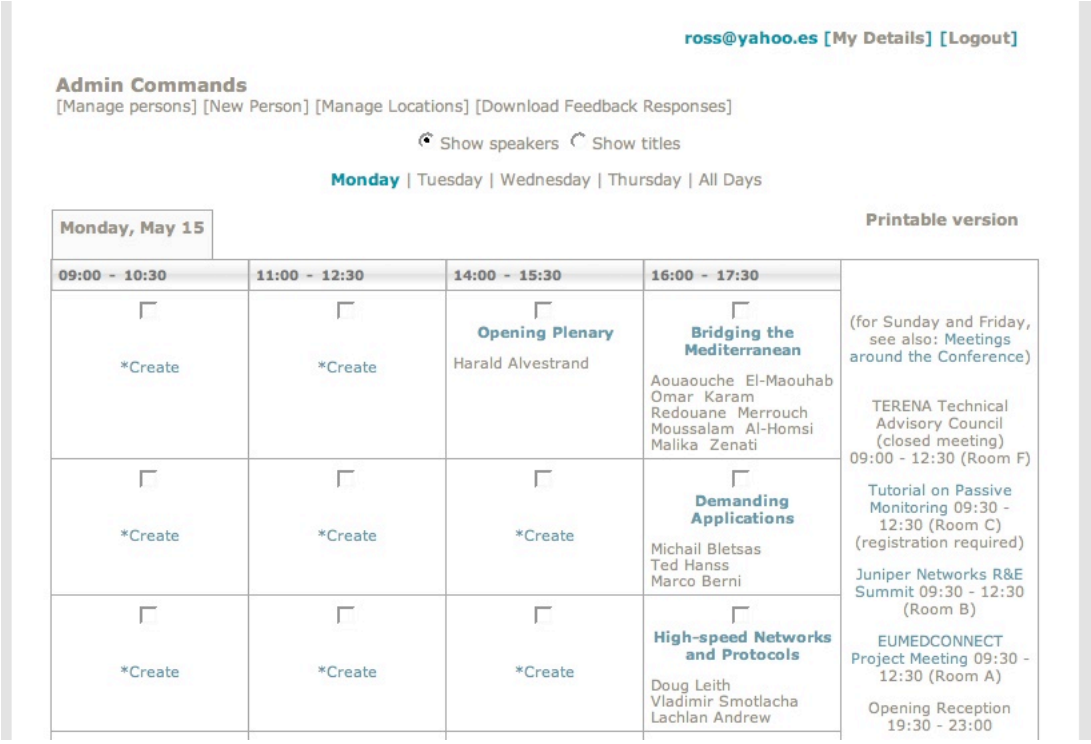


Figura 5.13: Programa de una conferencia. Modo administrador.

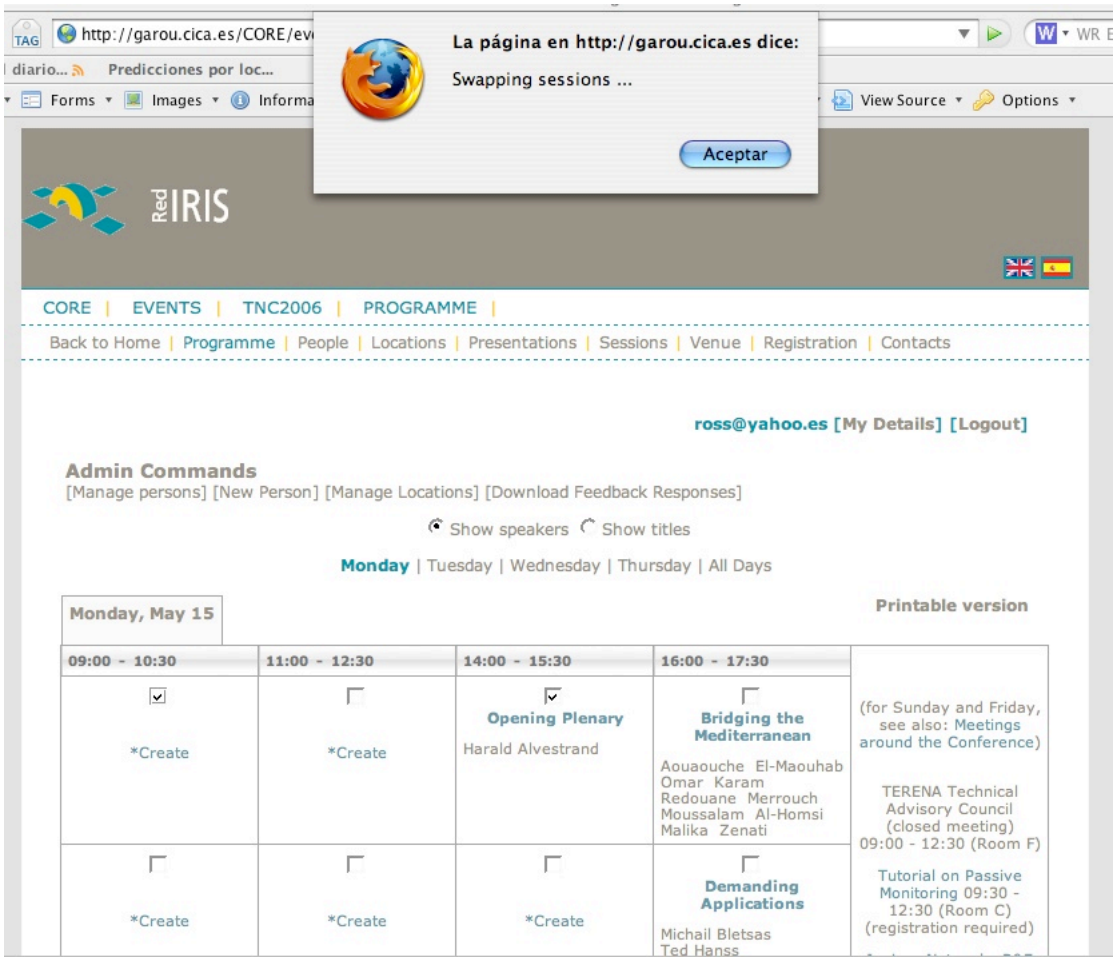


Figura 5.14: Detalle de intercambio de sesiones. Modo administrador.

Bibliografía

[W3C] Guía breve de Internacionalización
http://www.w3c.es/divulgacion/guiasbreves/internacionalizacion

[Translation Project] The Translation Project
http://translationproject.org/

[GNU Gettext] GNU Gettext Utilities
http://www.gnu.org/software/gettext/manual/gettext.html

[Apache] The Apache Software Foundation
http://apache.org

[PostgreSQL] PostgreSQL: The world's most advanced open source database
http://www.postgresql.org

[PHP] PHP HyperText Pre-processor
http://php.net

[PEAR] The PHP Extension and Application Repository
http://pear.php.net

[PECL] The PHP Extension Community Library
http://pecl.php.net

[Smarty] Smarty: Template Engine
http://smarty.php.net

[Bash] Advanced Bash Scripting Guide, Mendel Cooper
http://tldp.org/LDP/abs/html/

[RedIRIS] RedIRIS: Red Española de I+D
http://rediris.es

[Terena] Terena: Networking the networkers
http://terena.org

[PAPI] PAPI RedIRIS
http://papi.rediris.es