Ingeniería de Telecomunicación

Proyecto Fin de Carrera

Sistema Web para Gestión de Propuestas y Votaciones y su aplicación a Conferencias

---

Autor:
Moisés Hiraldo Martos

Tutor:
Antonio Luque Estepa

2012/13

Agradecimientos

A mis padres, Salvador y María del Carmen, por su apoyo incondicional durante mi larga etapa de estudiante, que ahora toca a su fin.

A mi profesor, y tutor del presente proyecto, Antonio Luque. Por su docencia, por su inspiración como ingeniero y como persona, y por concederme la oportunidad de realizar este proyecto.

A toda la comunidad del software libre, por su impagable contribución al conocimiento, al desarrollo de Internet y de la ingeniería en general; por ende, a mi formación como ingeniero, que desemboca en el proyecto que aquí se trata.

Finalmente, a la educación pública. Sin ella, como tantos otros compañeros, yo no habría llegado hasta aquí. Mas no debe olvidarse que recorrer este duro camino no es un privilegio vacío, sino una libertad fundamental llena de sacrifico y esfuerzo. Privar a una persona de la oportunidad de emprenderlo no constituye, por tanto, más que apología de la ignorancia, la mediocridad y el estancamiento social.

Índice general

Índice de figuras

Índice de cuadros

Índice de algoritmos

Capítulo 1

Introducción

1.1. Motivación

El presente proyecto surge cuando Antonio Luque, tutor del mismo y miembro del Comité de Web e Información del IEEE-IES, se plantea la necesidad de adaptar el proceso de propuestas y selección de conferencias de la organización a las nuevas tecnologías web.

La Industrial Electronics Society, perteneciente al IEEE, es una asociación que abarca un diverso rango de actividades técnicas dedicadas a la aplicación de las ciencias eléctrica y electrónica, para la mejora de procesos industriales y de fabricación. Dichas actividades técnicas abordan los últimos desarrollos en sistemas de control informatizados, robótica, comunicaciones y automatización en fábrica, fabricación flexible, adquisición de datos y procesamiento de señales, sistemas de visión, y electrónica de potencia. La asociación actualiza continuamente su programa de actividades técnicas en función de las necesidades de la industria moderna.

Una de las actividades técnicas de mayor importancia consiste en la organización y el patrocinio de conferencias internacionales. De esta forma, el IEEE-IES define una guía de pasos a seguir para realizar una propuesta al Comité de Conferencias. Una vez recibida la propuesta, los miembros del IEEE-IES se encargan de estudiarla, tomar una decisión y comunicarla de vuelta al usuario. Hasta ahora, este proceso interno se realizaba mediante listas de correo, un método poco eficiente comparado con las posibilidades que ofrecen las tecnologías actuales. Es por ello que se plantea la necesidad de realizar un proyecto que, haciendo uso de la tecnología web, implemente el proceso de selección de conferencias, de forma que se mejore la experiencia del usuario y la eficiencia en el tratamiento y almacenamiento de la información.

1.2. Objetivo

El objetivo del proyecto es el diseño de un sistema web para la gestión de propuestas y votaciones que, sin pérdida de generalidad, permita implementar el proceso de selección de conferencias que serán patrocinadas por el IEEE-IES. En la figura 1.1 puede verse un diagrama de flujo del proceso.

En primer lugar, la página web debe permitir a un usuario, siguiendo el formato especificado por el IEEE-IES, realizar una propuesta para la celebración de una conferencia. Cuando dicha propuesta se realice, se le notificará mediante correo electrónico al Vice Presidente de Workshops Activities del IEE-IES, que desde la propia web podrá modificar el estado de la propuesta:

- Rechazarla si no cumple el formato especificado.
- Solicitar al usuario que proporcione más información sobre la conferencia para poder tomar una decisión.
- Abrir una votación, de forma que el resto de miembros del IEEE-IES puedan expresar su opinión respecto a la propuesta.
- Aceptar la propuesta cuando considere que se ha alcanzado un consenso entre los miembros.

Además, es objeto del proyecto el diseño e implementación de la base de datos necesaria para almacenar y mantener toda la información relativa al proceso. El propio flujo ya permite intuir qué información será imprescindible almacenar:

- Cualquier usuario que vaya a realizar una propuesta, incluyendo algunos de sus datos básicos como el nombre y el correo electrónico.
- Miembros del IEEE-IES susceptibles de participar en las votaciones.
- Las propuestas que se realicen, con sus posibles revisiones, votaciones y decisiones asociadas.

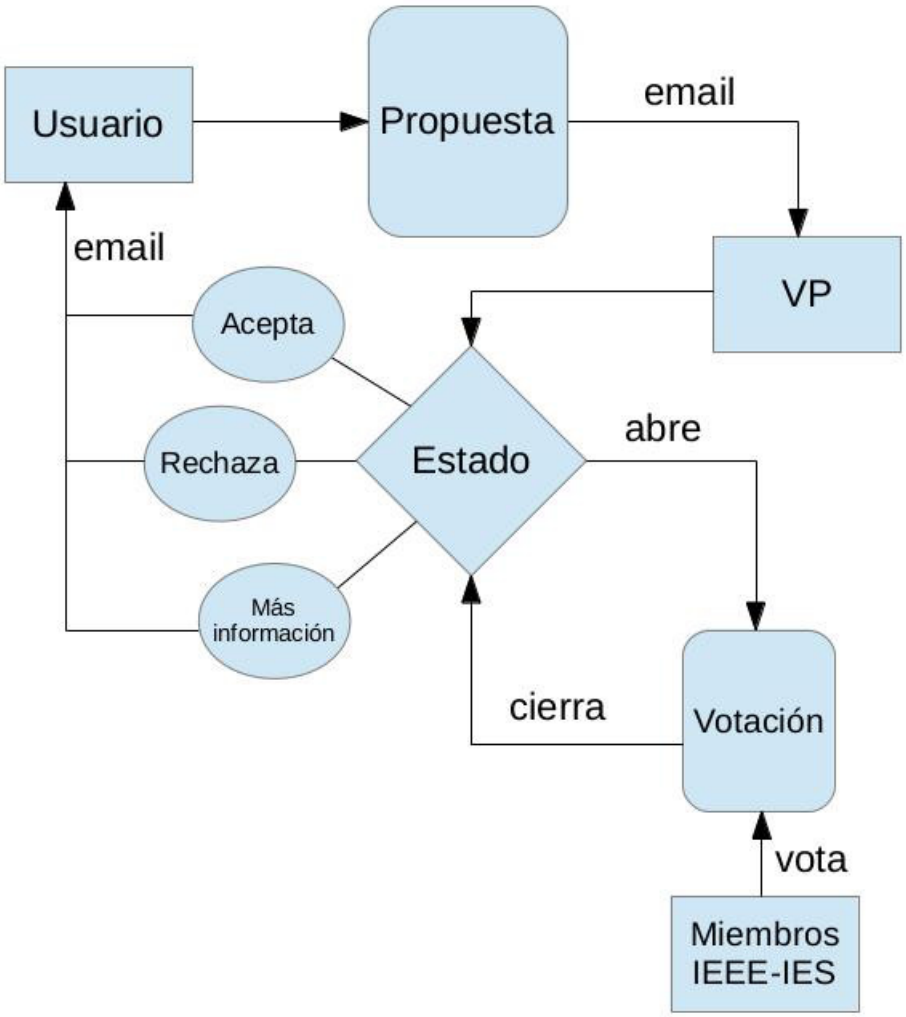

Figura 1.1: Flujo del proceso

Capítulo 2

Estado del Arte

Antes de comenzar el diseño del sistema web, es necesario realizar un estudio de las distintas partes que lo componen, valorando las tecnologías disponibles actualmente para la implementación de cada una y eligiendo las que mejor se ajusten a las necesidades del proyecto.

2.1. Gestor de la base de datos: MySQL

Un Sistema de Gestión de Bases de Datos (SGBD) es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en la base de datos, además de proporcionar las herramientas para añadir, borrar modificar y analizar los datos. La información es accesible usando herramientas específicas de interrogación y de generación de informes, o bien mediante aplicaciones al efecto. Por último, proporcionan métodos para mantener la integridad de los datos, administrar el acceso de usuarios a dichos datos y recuperar la información si el sistema se corrompe. Entre las numerosas soluciones que ofrece el mercado, se elige el paquete MySQL con licencia GPL. El soporte de la comunidad, las ventajas del software libre y su buena integración y amplio uso en aplicaciones web, hacen de MySQL un gestor ideal para una base de datos pequeña como la del presente proyecto.

Las principales características que ofrece MySQL son:

- Arquitectura con motor de almacenamiento insertable.
- Múltiples motores de almacenamiento:
  - InnoDB
  - MyISAM
  - NDB (MySQL Cluster)

    - Memoria
    - Archivo
    - CSV

- Replicación MySQL para mejorar el rendimiento de las aplicaciones y la escalabilidad.
- Particiones MySQL para mejorar el rendimiento y la gestión de aplicaciones de bases de datos grandes.
- Procedimientos almacenados para mejorar la productividad de los desarrolladores.
- Vistas para asegurar que la información sensible no se ve comprometida.
- Esquema de funcionamiento para el control de consumo de recursos a nivel de usuario/aplicación.
- Esquema de información para facilitar el acceso a los metadatos.
- Conectores MySQL (ODBC, JDBC,. NET, etc) para la creación de aplicaciones en múltiples lenguajes de programación.

2.2. Presentación de la información: HTML/CSS

Una parte esencial del proyecto consistirá en estructurar la información para que los usuarios puedan acceder a ella a través de la Web. En la actualidad, el lenguaje de marcado HTML es el más utilizado para estructurar los documentos destinados a visualizarse en un navegador Web, lo cual le convierte en la única elección posible.

El estándar HTML está desarrollado por el W3C (World Wide Web Consortium), siendo su versión más reciente la 4.01. Básicamente, HTML es un lenguaje que mediante etiquetas permite definir la estructura de un documento para su visualización en el navegador. Además, se usará CSS, un lenguaje de hojas de estilo estandarizado también por el W3C, que permitirá separar la definición de la estructura del documento (HTML) de su estilo y formato (CSS).

2.3. Contenido dinámico: PHP

Con el fin de implementar el flujo descrito en el objetivo del proyecto, será necesario un lenguaje de programación que genere dinámicamente para cada caso el contenido a presentar al usuario. Para ello, deberá ser capaz tanto de leer como de modificar la información almacenada en la base de datos, por lo que lo ideal es un lenguaje de programación que se ejecute en el lado servidor. PHP se ajusta perfectamente a estos requerimientos, presentando además las ventajas de ser software libre y tener un amplio soporte de la comunidad.

Aunque JSP también es una opción válida y una tecnología muy extendida, PHP tiene una curva de aprendizaje más rapída por su sintaxis sencilla y flexible, y es más rápido en la ejecución cuando no se sobrepasa un límite de conexiones concurrentes. La modularidad y escalabidad que proporciona JSP supondrían ventajas en proyectos de más envergadura. En definitiva, PHP es la opción que mejor se ajusta a las necesidades del problema.

En resuemen, PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante. Está desarrollado por el PHP Group y actualmente su versión más reciente es la 5.4. Entres sus principales características destacan:

- Orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- Es considerado un lenguaje fácil de aprender, ya que en su desarrollo se simplificaron distintas especificaciones, como es el caso de la definición de las variables primitivas, ejemplo que se hace evidente en el uso de php arrays.
- El código fuente escrito en PHP es invisible al navegador web y al cliente, ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.

- Capacidad de expandir su potencial utilizando módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su sitio web oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Tiene manejo de excepciones (desde PHP5).
- Debido a su flexibilidad ha tenido una gran acogida como lenguaje base para las aplicaciones WEB de manejo de contenido, y es su uso principal.

2.4. Motor de Plantillas: Smarty

Smarty es un motor de plantillas para PHP, es decir, separa el código PHP, como lógica de negocios, del código HTML, como lógica de presentación, y genera contenidos web mediante la colocación de etiquetas Smarty en un documento. Se encuentra bajo la Licencia Pública General Reducida de GNU. De esta forma, separando las tareas de programación del contenido dinámico en PHP y estructuración del documento HTML, se consiguen varias ventajas como una mayor modularidad del proyecto y mejor legibilidad del código. Algunas de las principales características que ofrece Smarty son:

- Expresiones regulares.
- Bucles foreach, while.
- Sentencias condicionales if, elseif, else...
- Modificadores de variables.
- Funciones creadas por el usuario.
- Evaluación de expresiones matemáticas en la plantilla.

2.5. Servidor web HTTP: Apache

HTTP (Hypertext Transfer Protocol) es el protocolo usado en cada transacción de la World Wide Web. HTTP fue desarrollado por el World Wide Web Consortium y la Internet Engineering Task Force, colaboración que culminó en 1999 con la publicación de una serie de RFC, el más importante de ellos es el RFC 2616 que especifica la versión 1.1. HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

Por lo tanto, para el funcionamiento del sistema web será necesario un servidor que atienda las peticiones del cliente, ejecutando el lenguaje de contenido dinámico (PHP) y enviando el documento HTML generado a través del protocolo HTTP. Siendo el servidor web empleado en más de la mitad de los sitios web del mundo, Apache se presenta indiscutiblemente como la única opción.

El servidor HTTP Apache es un servidor web HTTP de código abierto, que implementa el protocolo HTTP/1.12 y la noción de sitio virtual. Se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation. Entres sus principales ventajas se encuentran su modularidad, su disponibilidad en múltiples plataformas y el excelente soporte debido al gran número de desarrolladores y usuarios.

2.6. Resumen de las tecnologías elegidas

En la tabla 2.1 se recogen las tecnologías que se usarán en el presente proyecto.

Función	Tecnología empleada	Versión
Gestor de la base de datos	MySQL	5.5.29
Presentación de la información	HTML/CSS	4.0.1/2.1
Generación de contenido dinámico	PHP	5.4.11
Motor de plantillas	Smarty	3.0.8
Servidor Web	Apache	2.4.3

Cuadro 2.1: Tecnologías elegidas

2.7. Integración de las tecnologías

En la figura 2.1 puede verse cómo se integran las tecnologías que se usarán para la implementación del sistema web. El usuario, desde un navegador web, realiza una

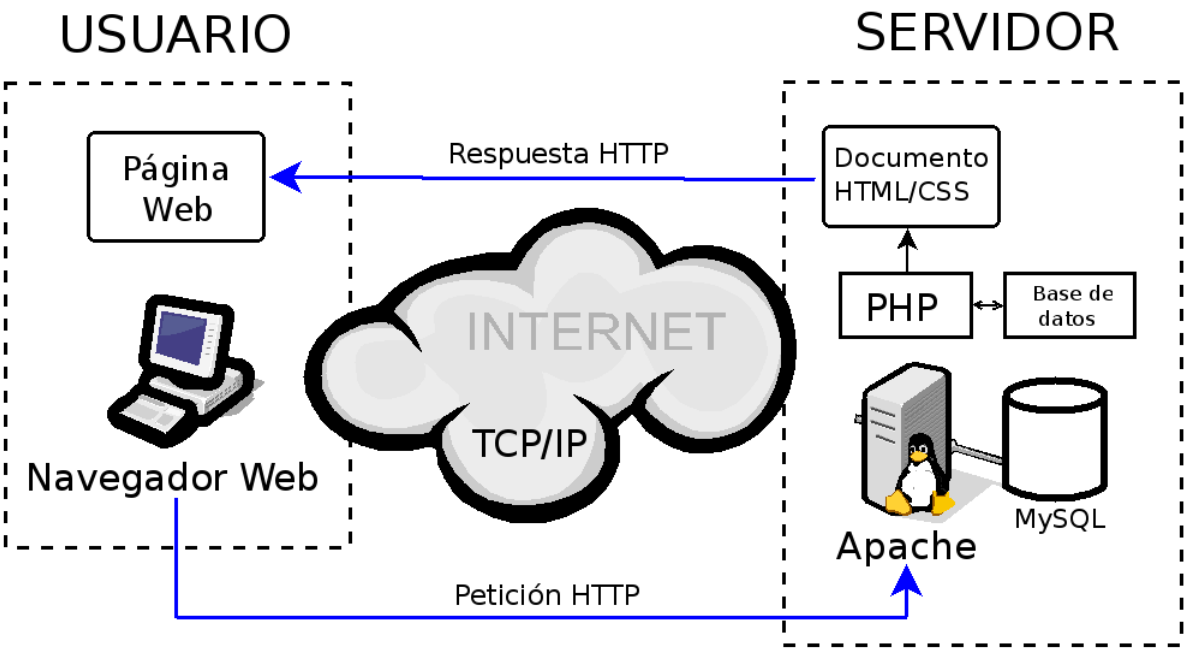

Figura 2.1: Integración de las tecnologías

petición usando el protocolo HTTP. La petición es atendida por el servidor Apache ejecutando el lenguaje de contenido dinámico. PHP se encargará, según la petición del usuario y obteniendo o modificando información de la bases de datos MySQL cuando sea necesario, de generar el documento HTML/CSS. Dicho documento será enviado por el servidor Apache, de nuevo usando el protocolo HTTP, para que finalmente el usuario pueda visualizarlo en el navegador web.

Capítulo 3

Planificación

En este capítulo se describen los recursos utilizados para la realización del proyecto, así como la planificación temporal llevada a cabo en el mismo.

3.1. Recursos

3.1.1. Humanos

- D. Antonio Luque Estepa, profesor del Departamento de Ingeniería Electrónica de la Universidad de Sevilla y miembro del Comité de Web e Información del IEEE-IES, en calidad de tutor del proyecto.
- Moisés Hiraldo Martos, alumno de la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla, autor del proyecto.

3.1.2. *Hardware*

Servidor para puesta en marcha y pruebas del sistema web:

- Ordenador de sobremesa con procesador AMD Athlon 64 3500+ a 2200 MHz, 2 GB de memoria RAM y sistema operativo GNU/Linux Slackware 14.0.

Equipo personal de trabajo:

- Portátil con procesador AMD Turion 64 x2 a 1600 MHz, 1 GB de memoria RAM y sistema operativo GNU/Linux Debian Wheezy.

3.1.3. *Software*

Implementación y administración de la base de datos:

- Gestor de base de datos MySQL.
- Administrador web de MySQL phpMyAdmin.

Implementación del servidor web:

- Servidor HTTP Apache.
- Módulo PHP.

Pruebas del sistema web:

- Navegadores web Mozilla Firefox y Google Chrome.

Edición de código:

- Editor de texto Emacs.

Elaboración de imágenes, flujos y diagramas:

- Editores de diagramas LibreOffice Draw, Calligra Flow y Dia.
- Editor de imágenes GIMP.

Elaboración de la memoria y presentación:

- Editor gráfico LyX para composición de textos en LaTeX.
- Editor de presentaciones LibreOffice Impress.

3.2. Planificación temporal

Para la realización del proyecto se ha elaborado una planificación temporal basada en la siguiente división de tareas:

1. Análisis del problema y elección de las tecnologías: análisis detallado del problema, definición del objetivo del proyecto y búsqueda de las tecnologías disponibles para implementarlo.
2. Documentación teórica: estudio y formación teórica necesaria para acometer el diseño de las distintas partes del sistema web.

3. Diseño de la base de datos: modelado teórico de la base de datos e implementación en MySQL.

4. Diseño PHP: programación del código encargado de generar el contenido dinámico del sistema.

5. Diseño HTML/CSS: programación de la estructura y presentación de la página web.

6. Implementación y pruebas: instalación de las tecnologías necesarias para la implementación del sistema, puesta en marcha y pruebas de funcionamiento.

7. Elaboración de la memoria y presentación.

Suponiendo una duración aproximada de 400 horas, en la siguiente tabla se presenta la carga temporal y porcentual de cada una de las tareas.

Tarea	Porcentaje	Tiempo
Análisis y elección de las tecnologías	2 %	8 h
Documentación teórica	15 %	60 h
Diseño de la base de datos	8 %	32 h
Diseño PHP	30 %	120 h
Diseño HTML/CSS	20 %	80 h
Implementación y pruebas	10 %	40 h
Elaboración de la memoria	15 %	60 h

Cuadro 3.1: Planificación temporal

Capítulo 4

Diseño

Para el diseño del sistema web, se aprovechará la modularidad innata que ya se presentó al elegir las tecnologías que implementarán el sistema. De esta forma, se abordará cada una de las partes del mismo con un análisis teórico hasta llegar a la solución propuesta, para finalmente aplicar dicha solución con la tecnología correspondiente.

Se comenzará con el diseño de la base de datos, ya que es imprescindible conocer de qué elementos se compone para generar el código del contenido dinámico. A continuación se creará la estructura de la web en PHP, y por último se diseñará el aspecto visual de la página con HTML/CSS, gracias a que el motor de plantillas Smarty nos permite separar las dos tareas.

4.1. Diseño de la base de datos

4.1.1. Modelo entidad-relación

El primer paso para el diseño de la base de datos será elaborar un diagrama con las entidades que la componen, las relaciones entres ellas y sus atributos. En la figura 4.1 puede verse el modelo entidad-relación de la base de datos.

La entidad People modela a los usuarios que interactúan con el sistema web, con sus correspondientes atributos: nombre, email, contraseña... Mientras que a través de la entidad User Types se modelan los tipos de usuario existentes y sus correspondientes permisos, mediante atributos binarios, que determinarán qué acciones pueden llevar a cabo en la web. De esta forma, la relación entre las entidades es de n a 1, es decir, que un usuario puede ser de un único tipo.

Las entidades Proposals y Revisions modelan las propuestas de los usuarios y sus distintas versiones, siendo la segunda una entidad débil de la primera y guardando

una relación 1 a n (esto significa que una versión sólo puede existir perteneciendo a una propuesta, y que una propuesta puede tener varias versiones). De manera genérica, la entidad REVISIONS tendrá los atributos de una propuesta que puedan cambiar con cada versión (comentarios, fecha de subida...), mientras que la entidad PROPOSALS recogerá los atributos fijos como puedan ser el nombre o la fecha de comienzo del evento.

Por último, las entidades VOTE TYPES y DECISION TYPES modelan qué tipo de votos y decisiones podrán tomarse sobre las propuestas, completando las tres relaciones que pueden establecerse entre usuarios y propuestas:

- Proponer: un usuario puede proponer varias versiones de distintas propuestas (relación 1 a n).
- Votar: los usuarios que tengan los permisos adecuados pueden votar varias versiones de distintas propuestas (relación n a m).
- Decidir: un usuario, en principio sólo el administrador, puede decidir sobre las distintas versiones de una propuesta. Aunque se ha supuesto una relación 1 a n, podría cambiar si se considera oportuno otorgar permisos de decisión a otros usuarios.

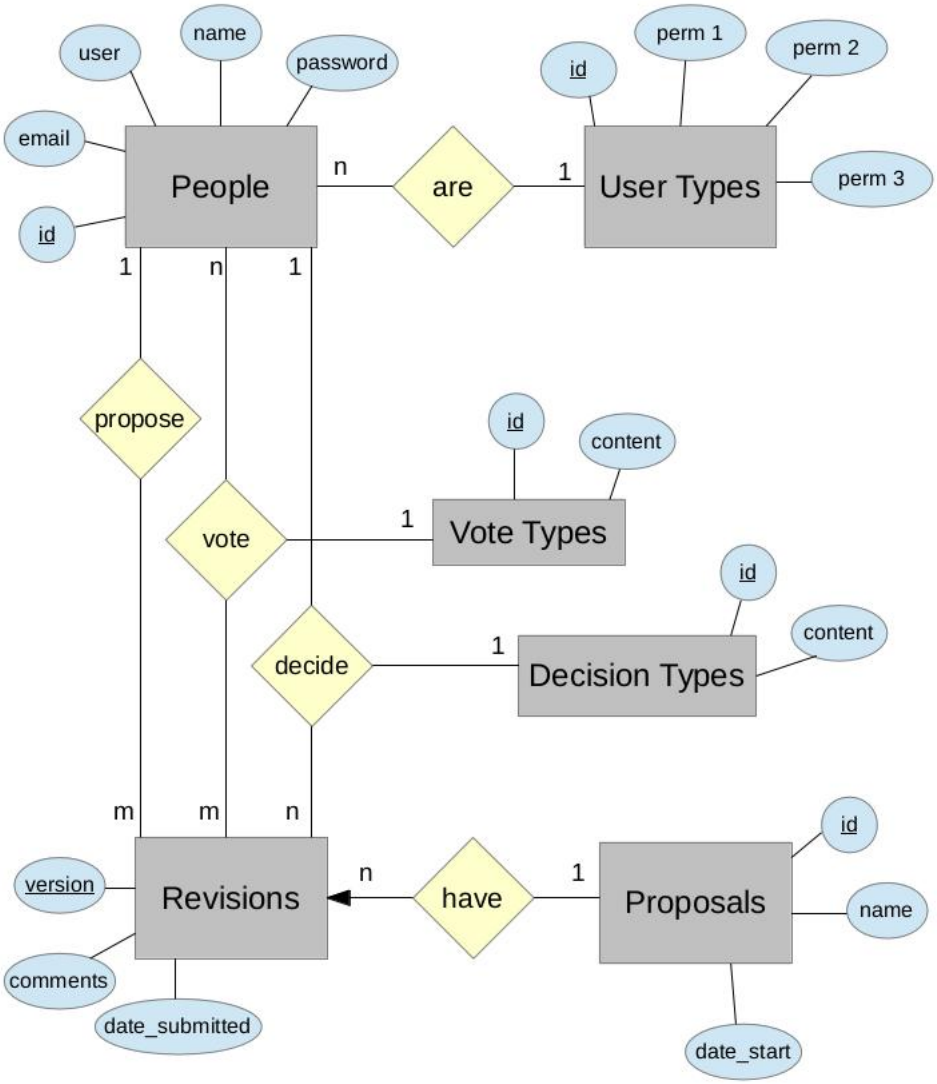

Figura 4.1: Modelo entidad-relación de la base de datos

4.1.2. Conversión en tablas

En general, cada entidad y cada relación se convierten en una tabla a la hora de implementar la base de datos, pero pueden realizarse algunas simplificaciones:

- La relación entre las entidades PEOPLE y USER TYPES es equivalente a un atributo en la tabla PEOPLE que indique el tipo de usuario.
- La relación entre las entidades PEOPLE y REVISIONS es equivalente a añadir un atributo a la tabla PROPOSALS que contenga la id del usuario que la propuso.
- La relación entre las entidades REVISIONS y PROPOSALS es equivalente a un atributo en la tabla REVISIONS que contenga la id de la propuesta a la que pertenece.

Realizando las simplicaciones anteriores, y a la espera de definir los atributos concretos, genéricamente habrá que implementar las siguientes tablas:

PEOPLE (id, user, password, email, role...);
USER_TYPES (role, perm1, perm2, perm3...);
PROPOSALS (id, proposer, name, date_start...);
REVISIONS(version, proposal, comments, date_submitted...);
VOTES_TYPES (id, content...);
DECISION_TYPES (id, content...);
VOTES (vote, proposal, version, user);
DECISIONS (decision, proposal, version, user);

4.1.3. Implementación en MySQL

Partiendo del diseño general de la base de datos, se definen los atributos específicos que se consideran necesarios para el caso concreto de las conferencias del IEEE-IES, así como los comandos para implementar cada una de las tablas en MySQL.

4.1.3.1. Tabla PEOPLE

Los atributos considerados en el diseño general son suficientes, sólo se tendrá en cuenta que el nombre de usuario debe ser único en la web.

Algoritmo 4.1 Implementación de la tabla PEOPLE

```
CREATE TABLE IF NOT EXISTS `people`(
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `user` varchar(20) NOT NULL,
    `password` varchar(20) NOT NULL,
    `email` text NOT NULL,
    `name` text NOT NULL,
    `role` int(11) NOT NULL,
    PRIMARY KEY (`id`),
    UNIQUE KEY `user` (`user`)
);
```

4.1.3.2. Tabla USER_TYPES

Se considera que los usuarios pueden llevar a cabo cuatro acciones fundamentales en la web: realizar propuestas, ver propuestas, votar propuestas y decidir propuestas. Para cada una de estas acciones se define un permiso en forma de atributo binario, siendo el atributo *role* la clave primaria que identifica a cada tipo de usuario.

Algoritmo 4.2 Implementación de la tabla USER_TYPES

```
CREATE TABLE IF NOT EXISTS `user_types`(
    `role` int(11) NOT NULL,
    `CANSUBMIT` tinyint(1) NOT NULL,
    `CANVIEWSUBM` tinyint(1) NOT NULL,
    `CANVIEWSTATUS` tinyint(1) NOT NULL,
    `CANVOTE` tinyint(1) NOT NULL,
    `CANDECIDE` tinyint(1) NOT NULL,
    PRIMARY KEY (`role`)
);
```

En principio, se considera que existen tres tipos de usuarios: usuario por defecto (puede realizar propuestas), miembro del IEEE-IES (puede realizar, ver y votar propuestas) y administrador (todos los permisos). Tendrán los roles 0, 1 y 2 respectivamente.

Algoritmo 4.3 Inserción de datos en la tabla USER_TYPES

```
INSERT INTO `user_types` (`role`, `CANSUBMIT`, `CANVIEWSUBM`, `CANVIEWSTATUS`, `
    CANVOTE`, `CANDECIDE`)
    VALUES (0, 1, 1, 0, 0, 0), (1, 1, 1, 1, 1, 0), (2, 1, 1, 1, 1, 1);
```

4.1.3.3. Tabla PROPOSALS

Se considera que los atributos fijos de una conferencia son su fecha de inicio, el nombre completo, un identificador único de pocos caracteres, y la URL de la web.

Algoritmo 4.4 Implementación de la tabla PROPOSALS

```
CREATE TABLE IF NOT EXISTS `proposals`(
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `date_start` date NOT NULL,
    `proposer` int(11) NOT NULL,
    `fullname` text NOT NULL,
    `confid` varchar(20) NOT NULL,
    `url` varchar(120) NOT NULL,
    PRIMARY KEY (`id`),
    UNIQUE KEY `confid` (`confid`)
);
```

4.1.3.4. Tabla REVISIONS

La clave primaria de la tabla REVISIONS estará formada por los atributos *version* y *conferencie*, haciendo este último referencia a una id de la tabla PROPOSALS (deberá actualizarse o borrarse si cambia o desaparece una conferencia, de manera que no haya incongruencias en la base de datos). Además, se añade el atributo *close_date*, que indica si una revisión está abierta a votación conteniendo su fecha de cierre.

Algoritmo 4.5 Implementación de la tabla REVISIONS

```
CREATE TABLE IF NOT EXISTS `revisions`(
    `conference` int(11) NOT NULL,
    `version` int(11) NOT NULL,
    `comments` text NOT NULL,
    `date_submitted` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `close_date` date NOT NULL,
    PRIMARY KEY (`conference`, `version`),
    FOREIGN KEY (`conference`) REFERENCES `proposals`(`id`) ON UPDATE CASCADE ON
        DELETE CASCADE
);
```

4.1.3.5. Tabla VOTE_TYPES

Sólo es necesario tener un identificador y el contenido de cada voto.

Algoritmo 4.6 Implementación de la tabla VOTE_TYPES

```
CREATE TABLE IF NOT EXISTS `vote_types`(
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `content` text NOT NULL,
    PRIMARY KEY (`id`)
);
```

Y se consideran, en principio, cuatro tipos de votos:

Algoritmo 4.7 Inserción de datos en la tabla VOTE_TYPES

```
INSERT INTO `vote_types`(`id`, `content`)
    VALUES (1, 'I agree'), (2, 'I agree, if a reduced fee is available'), (3, 'I
        disagree'), (4, 'No opinion');
```

4.1.3.6. Tabla DECISION_TYPES

Sólo es necesario tener un identificador, la decisión a tomar y su estado asociado.

Algoritmo 4.8 Implementación de la tabla DECISION_TYPES

```
CREATE TABLE IF NOT EXISTS `decision_types`(
    `decision` int(11) NOT NULL AUTO_INCREMENT,
    `action` text NOT NULL,
    `content` text NOT NULL,
    PRIMARY KEY (`decision`)
);
```

Se consideran, en principio, tres tipos de decisiones:

Algoritmo 4.9 Inserción de datos en la tabla DECISION_TYPES

```
INSERT INTO `decision_types` (`decision`, `action`, `content`)
    VALUES (1, 'Accept', 'Accepted'), (2, 'Reject', 'Rejected'), (3, 'Ask for revised
         proposal', 'Asked for revised proposal');
```

4.1.3.7. Tabla VOTES

La clave primaria estará compuesta por los atributos *vote*, *person*, *conference* y *version*, haciendo referencia a las claves correspondientes de las tablas VOTE_TYPES, PEOPLE, PROPOSALS y REVISIONS, respectivamente. Además, en el atributo *time_cast* se almacenará la fecha y hora en la que se emitió el voto.

Algoritmo 4.10 Implementación de la tabla VOTES

```
CREATE TABLE IF NOT EXISTS `votes`(
    `conference` int(11) NOT NULL,
    `version` int(11) NOT NULL,
    `person` int(11) NOT NULL,
    `vote` int(11) NOT NULL,
    `comments` text NOT NULL,
    `time_cast` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
        CURRENT_TIMESTAMP,
    PRIMARY KEY (`conference`, `version`, `person`, `vote`),
    FOREIGN KEY (`conference`, `version`) REFERENCES `revisions`(`conference`, `
        version`) ON UPDATE CASCADE ON DELETE CASCADE,   FOREIGN KEY (`person`)
        REFERENCES `people`(`id`) ON UPDATE CASCADE ON DELETE CASCADE
);
```

4.1.3.8. Tabla DECISIONS

La clave primaria estará compuesta por los atributos *decision*, *conference* y *version*, haciendo referencia a las claves de las tablas DECISION_TYPES, PROPOSALS y REVISIONS, respectivamente. Además, en el atributo *time_cast* se almacenará la fecha y hora en la que se emitió la decisión.

Algoritmo 4.11 Implementación de la tabla DECISIONS

```
CREATE TABLE IF NOT EXISTS `decisions`(
    `conference` int(11) NOT NULL,
    `version` int(11) NOT NULL,
    `decision` int(11) NOT NULL,
    `person` int(11) NOT NULL,
    `time_cast` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
        CURRENT_TIMESTAMP,
    `comments` text NOT NULL,
    PRIMARY KEY (`conference`, `version`, `decision`),
    FOREIGN KEY (`conference`, `version`) REFERENCES `revisions`(`conference`, `
        version`) ON UPDATE CASCADE ON DELETE CASCADE
);
```

4.1.3.9. Resumen de tablas y atributos

En el siguiente cuadro se recogen las tablas que se deben crear, con sus correspondientes atributos:

TABLA	ATRIBUTOS
PEOPLE	id, user, password, email, name, role
USER_TYPES	role, SUBMIT, VIEWSUBM, VIEWSTATUS, VOTE, DECIDE
PROPOSALS	id, proposer, date_start, fullname, confid, url
REVISIONS	conference, version, comments, date_submitted, close_date
VOTE_TYPES	id, content
DECISION_TYPES	id, action, content
VOTES	conference, version, person, vote, comments, time_cast
DECISIONS	conference, version, decision, person, time_cast

Cuadro 4.1: Tablas y atributos de la base de datos

4.2. Motor de plantillas Smarty

Smarty es un motor de plantillas para PHP que permite separar el código que genera el contenido dinámico del lenguaje de etiquetado HTML que estructura el documento. Para ello, se definen dos tipos de archivos:

- Archivos con extenión *tpl*: en ellos se define la estructura de la página web en HTML. Además, se puede hacer uso de variables y sentencias de control propias de Smarty.

- Archivos con extensión *php*: contienen el código para generar el contenido dinámico de la página, y sólo se hace referencia a las plantillas que definen la estructura del documento a través de sentencias Smarty.

De esta forma, se consiguen separar las tareas de diseño y programación de la web.

4.2.1. Setencias Smarty en PHP

Para asignar valores a variables de Smarty en PHP, tan sólo hay que iniciar una nueva instancia del objeto *Smarty* y usar el método *assign*, tal y como se muestra en el siguiente ejemplo:

Algoritmo 4.12 Setencias Smarty en PHP

```
1 $smarty=new Smarty;
2 $smarty->assign("VariableName",$Value);
3 $smarty->display("plantilla.tpl");
```

Siendo *VariableName* el nombre de la variable y *$Value* el valor asignado a dicha variable. De esta forma, ya estará lista para ser usada en la plantilla HTML, que se mostrará en el documento usando el método *display.*

4.2.2. Setencias Smarty en HTML

Las sentencias Smarty en HTML deben encerrarse entre llaves. Además de los bucles típicos en programación (if, while...), en el presente proyecto se usarán básicamente las sentencias para incluir archivos y mostrar variables:

Algoritmo 4.13 Sentencias Smarty en HTML

```
1 {include file="header.tpl"}
2 {$VariableName}
```

Las sentencias anteriores incluyen la plantilla *header.tpl* e imprime por pantalla el valor de *$VariableName* que se haya asignado en el código PHP.

4.3. Contenido dinámico: PHP

El objetivo del código PHP será leer y/o modificar la base de datos de acuerdo a las peticiones de los usuarios, generando la información pertinente de modo que pueda ser estructurada y presentada en un documento HTML.

4.3.1. Conceptos básicos

PHP es un lenguaje de programación muy simple, con una sintaxis bastante similar al lenguaje C en todo lo que se refiere a definición de variables, funciones y estructuras de control, de forma que sólo será necesario definir algunos conceptos nuevos relativos al paradigma de la programación web.

Uno de ellos es el concepto de sesión, entendida como un período de tiempo que un usuario determinado pasa interactuando con la web. El manejo de sesiones se realiza de forma automática por parte de PHP, de forma que sólo es necesario iniciarla con la función *session_start* y usar la variable global *$_SESSION* para almacenar los datos relativos a la sesión de usuario.

El otro es el concepto de formulario, que es el método que usarán los usuarios para enviar datos al servidor. Dichos datos son fácilmente accesibles en PHP sin más que usar las variables globales *$_POST* y *$_GET*, según los respectivos métodos HTML. De esta forma, en gran parte de los casos, el pseudocódigo de los archivos PHP será el que se presenta en la figura 4.2.

Se mostrará el formulario al usuario. Cuando se reciba la información, se comprobará si contiene errores, y en caso positivo se volverá a mostrar el formulario informando de los errores. En caso negativo, se procesará la información, se leerá o modificará la información de la base de datos y por último se enviará dicha información a la plantilla HTML correspondiente a través de variables Smarty.

4.3.2. Conexión con la base de datos

El primer paso es, necesariamente, escribir el código que permita conectarse a la base de datos y definir las funciones requeridas para realizar las operaciones elementales, puesto que se usarán continuamente durante la implementación del proceso general. Para ello se usará la extensión *mysqli*, que permite acceder a la funcionalidad proporcionada por MySQL desde PHP de forma muy simple, tal y como se muestra en el algoritmo 4.14.

Se ha definido la función *db_connect*, que a través de la función *mysqli_connect* realiza la conexión con la base de datos con los parámetros nombre de la base de

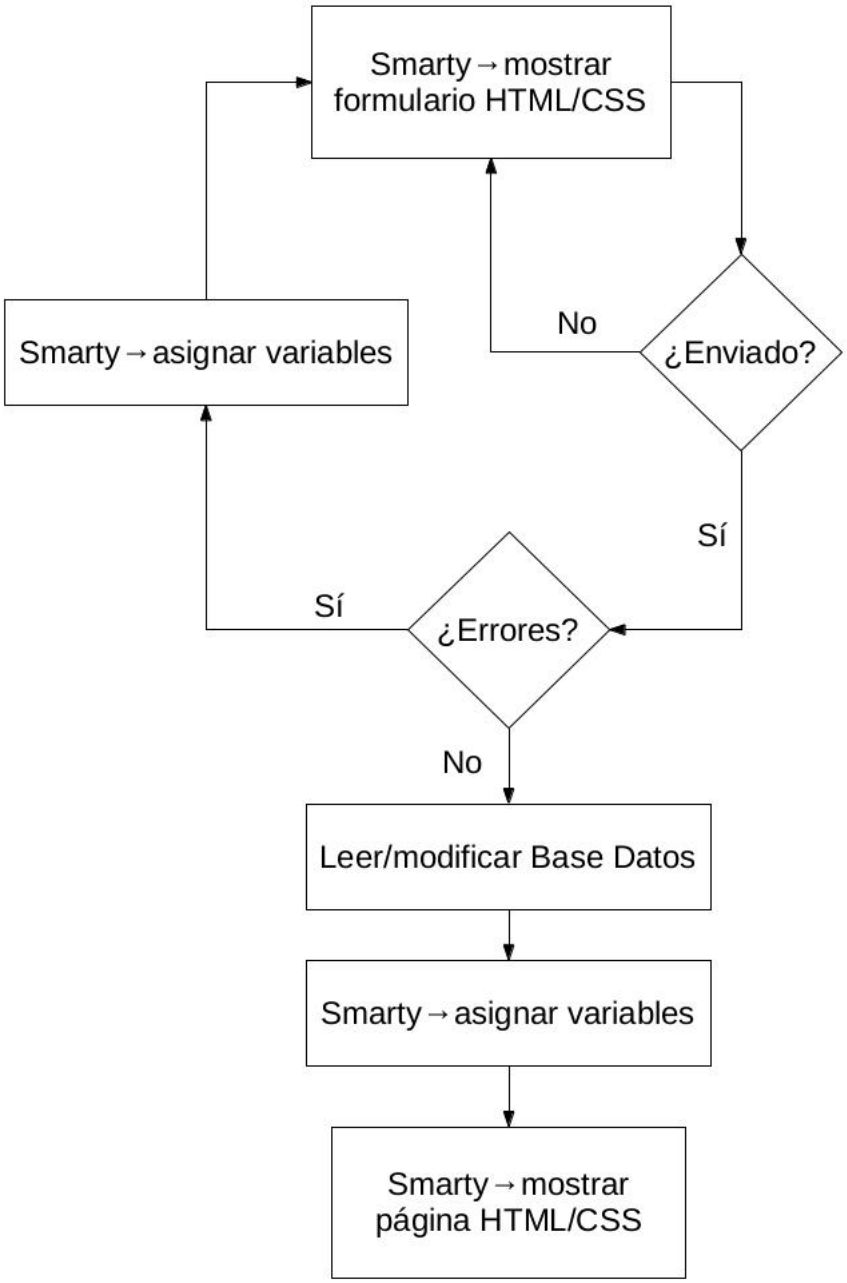

Figura 4.2: Pseudocódigo genérico de los archivos PHP

Algoritmo 4.14 Conexión con la base de datos

```php
<?php

// Database conecction parameters

global $db_conn;
$db_name="tech";
$db_host="";
$db_user="tech_conf";
$db_passwd="";

// Database functions

function db_connect() {
    global $db_name,$db_host,$db_user,$db_passwd,$db_conn;
    $db_conn= mysqli_connect($db_host,$db_user,$db_passwd,$db_name);
    mysqli_select_db($db_name);
    if($db_conn==false) {
        print("<H1>Error al conectar a la base de datos.</H1>");
        echo mysqli_connect_errno().": ".mysqli_connect_error();
        return false;
    }
    return $db_conn;
}
```

datos, host (por defecto local), usuario de la base de datos y contraseña. Si hay algún error, lo imprime por pantalla, y en caso contrario devuelve el conector.

Ahora sólo queda definir una serie de funciones que permitan realizar consultas a la base de datos y otras operaciones genéricas, como obtener el número de resultados o el propio resultado de una consulta. Dichas funciones, mostradas en el algoritmo 4.15, quedarán definidas en el archivo *config.php*, que será incluído en el resto para que puedan ser usadas en cualquier momento.

4.3.3. Funciones y variables globales

Además de las funciones para realizar operaciones con la base de datos, será interesante definir globalmente otras funciones y variables que se usarán continuamente durante la implementación del proceso, de forma que el código resultante sea más simple y menos extenso. Dichas funciones, al igual que las relativas a la base de datos, estarán en el fichero *config.php*.

4.3.3.1. Variables globales

Se definen en variables globales la URL de la web, el directorio donde se almacenarán los archivos relativos a las propuestas y la dirección remitente en los correos de notificaciones, así como la de destino en caso del administrador (VP).

Algoritmo 4.15 Funciones de consulta a la base de datos

```
function db_query($query) {
    global $db_name,$db_conn;
    $idx=mysqli_query($db_conn,$query);
    if($idx==false) {
        print("<H1>Error al ejecutar una consulta en la base de datos</H1>");
        print $query."<BR>";
        echo mysqli_errno().": ".mysqli_error();
        return false;
    }
    return $idx;
}

function db_numrows($idx) {
    if($idx) return mysqli_num_rows($idx);
    else return 0;
}

function db_result($idx,$row,$field) {
    mysqli_data_seek($idx,$row);
    $fields=mysqli_fetch_array($idx);
    return $fields[$field];
}

function db_numfields($conn) {
    return mysqli_field_count($conn);
}

function db_fieldname($idx,$num) {
    $field=mysqli_fetch_field_direct($idx,$num);
    return $field->name;
}

function db_error($conn) {
    return mysqli_error($conn);
}

function db_close($conn) {
    return mysqli_close($conn);
}

function db_fetch_row($idx,$row) {
    if($row>mysqli_num_rows($idx)-1) return false;
    if(mysqli_data_seek($idx,$row)==false) return false;
    return mysqli_fetch_row($idx);
}

function db_fetch_array($idx,$row) {
    if(mysqli_data_seek($idx,$row)==false) return false;
    return mysqli_fetch_array($idx);
}

function db_fetch_next_row($idx) {
    return mysqli_fetch_row($idx);
}
```

Algoritmo 4.16 Variables globales

```
// Global variables

session_start();
if (isset($_SESSION['mark']) == false) {
    session_regenerate_id(true);
    $_SESSION['mark']=true;
}

global $db_conn;
global $user,$password;

global $cb_path;
$cb_path="/conferences/tech";

global $cb_attached;
$cb_attached="/var/www/conferences/test/attach";

global $from_address;
$from_address="vp-workshops@ieee-ies.org";

global $vpw_address;
$vpw_address="vp-workshops@ieee-ies.org";

global $user_logged;
$user_logged=is_user_logged();

// Smarty variables

global $smarty;
$smarty=new Smarty;
$smarty->assign("is_user_logged",$user_logged);
$today=date("l, F d, Y",time());
$smarty->assign('today', date("l, F d, Y",time())); $smarty->assign('titulo',"IEEE
    IES Technically (Co-)sponsored Conferences");

if ($user_logged) {
    $user=$_SESSION["user"];
    $idx=db_query("select name from people where user='$user';");
    $name=db_result($idx,0,0);
    $smarty->assign("username",$name);
}
```

Además, se iniciará una sesión de Smarty, y se asignará valor a algunas variables como la fecha actual, el título de la página o el nombre del usuario en caso de que haya iniciado sesión. Por último, se llamará a la funcion de PHP *session_start* para que inicie o reanude una sesión cada vez que el usuario abra una página de la web. Las citadas definiciones pueden verse en el algoritmo 4.16.

4.3.3.2. Funciones relativas al usuario

Hay tres funciones básicas a las que se recurrirá constantemente durante el flujo de un usuario por la web: comprobar si ha iniciado sesión, comprobar si es un usuario válido y comprobar que tiene permisos para realizar una acción determinada.

La primera se implementa con la función *is_user_logged*, que simplemente com-

Algoritmo 4.17 Funciones relativas al usuario

```
function is_user_logged() {
    global $_SESSION;
    if(isset($_SESSION["user"]) && $_SESSION["user"]!="") return true;
    else return false;
}

function user_auth() {
    global $user,$password;
    if(isset($_SESSION["user"])) {
        $user=$_SESSION["user"];
        $password=$_SESSION["password"];
        // Check if valid
        $idx=db_query("select * from people where user='$user' and password='
            $password';");
        if(db_numrows($idx)!=1) {
            session_redirect("login.php");
            exit;
        }
    }
    else session_redirect("login.php");
}

function check_perms($perm) {
    global $user;
    $idx=db_query("select role from people where user='$user';");
    $role=db_result($idx,0,0);
    $idx=db_query("select * from user_types where role='$role';");
    $permission=db_result($idx,0,$perm);
    return $permission;
}
```

prueba si hay un usuario definido en la variable global de PHP *$_SESSION* (esta variable se crea automáticamente con la función *session_start* usada en el apartado anterior), devolviendo verdadero o falso.

La segunda se implementa con la función *user_auth*, que realiza una consulta a la base de datos comprobando que existe una fila en la tabla PEOPLE que contiene el usuario y contraseña, redireccionando a la página de login en caso negativo.

La última se implementa con la función *check_perms*. Esta función recibe como parámetro uno de los atributos definidos en la tabla USER_TYPES, comprueba el rol del usuario actual y devuelve el valor binario de dicho atributo, que indica si se tiene o no permiso para realizar una acción determinada.

En el algoritmo 4.17 puede verse el código correspondiente a dichas funciones.

4.3.3.3. Funciones relativas a las propuestas

Durante el flujo del proceso, será necesario comprobar en numerosas ocasiones si una propuesta está abierta a votación, si ya se ha tomado una decisión sobre ella o si un usuario concreto ha votado la versión más reciente. Para ello se implementan tres funciones.

Algoritmo 4.18 Funciones relativas a las propuestas

```
function is_proposal_open($id) {
    $votation_running=false;
    $idx=db_query("select * from revisions where conference='$id';");
    $v=db_numrows($idx)-1;
    $closedate=db_result($idx,$v,"close_date");
    if(is_earlier($closedate)) $votation_running=true;
    return $votation_running;
}

function is_proposal_decided($id) {
    $idx=db_query("select * from revisions where conference='$id';");
    $v=db_numrows($idx)-1;
    $idx=db_query("select * from decisions where conference='$id' and version='$v';")
        ;
    $numd=db_numrows($idx);
    if($numd>=1) return true;
    else return false;
}

function has_voted($person,$conference,$version) {
    $idx=db_query("select * from votes where person='$person' and conference='
        $conference' and version='$version';");
    if(db_numrows($idx)>=1) return true;
    return false;
}
```

La primera, *is_proposal_open*, compara la fecha actual con el atributo *close_date* de la última revisión de la propuesta que se recibe como parámetro. Este atributo toma la fecha del final de la votación, o el valor "0000-00-00" en caso de que esté cerrada (en cuyo caso la comparación de la función devolvería cero).

La segunda, *is_proposal_decided*, simplemente comprueba si hay una fila en la tabla DECISIONS con una decisión sobre la última revisión de la propuesta.

La tercera, *has_voted*, devuelve verdadero si el usuario pasado como parámetro ha votado la versión de la conferencia también pasada como parámetro, y falso en caso contrario.

En el algoritmo 4.18 se detalla el código de las tres funciones.

4.3.3.4. Otras funciones de uso general

A continuación se definen cuatro funciones de propósito general que también serán usadas en numerosas ocasiones.

La primera, *session_redirect*, redirige la sesión actual de un usario a la página que recibe como parámetro.

La segunda, *is_earlier*, devuelve verdadero si la fecha actual es anterior a la que recibe como parámetro, y falso en caso contrario.

La tercera, *template_process*, sustituye las variables del array que recibe como parámetro en una plantilla genérica abierta desde archivo, que será devuelta con el

Algoritmo 4.19 Otras funciones de uso general

```
function session_redirect($loc) {
    global $cb_path;
    $string="http://".getenv('HTTP_HOST').$cb_path."/".$loc;
    header("Location: $string");
    print("\n\n");
    exit;
}

function is_earlier($datecomp) {
    if($datecomp=='0000-00-00') return 0;
    $timestamp_comp=strtotime($datecomp);
    $today=strtotime(date("Y-m-d",time()));
    if($today <= $timestamp_comp) return 1;
    else return 0;
}

function template_process($tpl_file,$array_vars) {
    if(($handle=fopen($tpl_file,"r"))==FALSE) {
        print "file not found: $tpl_file";
        exit;
    }
    $template_source = fread($handle, filesize($tpl_file));
    foreach($array_vars as $var=>$val) {
        $template_source=preg_replace("/\{\\\$".$var."\}/",$val,$template_source);
    }
    return $template_source;
}

function escape_string($string){
    global $db_conn;
    return htmlspecialchars(mysqli_real_escape_string($db_conn, $string));
}

?>
```

fin de enviarse en una notificación por email.

Por último, se define la función *escape_string*, que elimina caracteres especiales y etiquetas HTML de las cadenas de caracteres con el fin de evitar la inyección de código.

La definición de las funciones de uso general puede verse en el algoritmo 4.19.

Además, se definirá en el archivo *formprocess.php* una función para comprobar que los campos de un formulario enviado son correctos. Esta función utiliza un array global que contiene un array por cada campo del formulario, conteniendo el tipo entre otros campos el nombre, el tipo, el valor y un mensaje de error. De esta forma, la función comprueba campo por campo del formulario según el tipo, añadiendo el mensaje correspondiente el campo error del array en caso de encontrarlo y devolviendo uno como resultado.

Algoritmo 4.20 Función para comprobar formularios I

```
<?php

function form_process() {
    global $_POST;
    global $fields;
    global $smarty;
    $error=0;

    // Check each field for correctness
    foreach($fields as $field) {

        // Reassign default values
        $smarty->assign($field["value"],$_POST[$field["name"]]);

        switch($field["type"]) {
            case "text":
                if($field["compul"] && (!isset($_POST[$field["name"]]) || $_POST[
                    $field["name"]]=="")) {
                    $smarty->assign($field["error_text"],"* Please fill the field ".
                        $field["disp_name"]);
                    $error=1;
                }
                break;
            case "url": // Check if valid url
                if($field["compul"] && (!isset($_POST[$field["name"]]) || $_POST[
                    $field["name"]]=="" || $_POST[$field["name"]]=="http://")) {
                    $smarty->assign($field["error_text"],"* Please fill the field ".
                        $field["disp_name"]);
                    $error=1;
                }
                break;
            case "email":
                if($field["compul"] && (!isset($_POST[$field["name"]]) || $_POST[
                    $field["name"]]=="")) {
                    $smarty->assign($field["error_text"],"* Please fill the field ".
                        $field["disp_name"]);
                    $error=1;
                }
                if(!check_email_address($_POST[$field["name"]])) {
                                    $smarty->assign($field["error_text"],"* The
                    email address ".$_POST[$field["name"]]." is not valid. Please
                    enter a valid email address");
                $error=1;
                }
                break;
```

Algoritmo 4.21 Función para comprobar formularios II

```
case "date": // Check if valid date
                if($field["compul"] && (!isset($_POST[$field["name"]]) || $_POST[
                    $field["name"]]==="")) {
                    $smarty->assign($field["error_text"],"* Please fill the field ".
                        $field["disp_name"]);
                    $error=1;
                }
                $sub_date=$_POST[$field["name"]];
                if(strtotime($sub_date)==FALSE && strtotime(str_replace('/', '-',
                    $sub_date))==FALSE) {
                    $smarty->assign($field["error_text"],"* Please enter the date as
                        yyyy-mm-dd");                    $error=1;
                }
                break;
            case "file":
                if($field["compul"] && (!isset($_FILES[$field["name"]]["name"]) ||
                    $_FILES[$field["name"]]["name"]=="")) {
                    $smarty->assign($field["error_text"],"* Please fill the field ".
                        $field["disp_name"]);
                    $error=1;
                    return $error;
                }
                if($field["mime"] && $field["mime"]!=$_FILES[$field["name"]]["type"]
                    ) {
                    $smarty->assign($field["error_text"],"* Please submit a file of
                        the appropriate format at ".$field["disp_name"]);
                    $error=1;
                    return $error;
                }
                break;
            }
        }
    return $error;
}


function check_email_address($email) {
    // First, we check that there's one @ symbol, and that the lengths are right
    if (!ereg("^[^@]{1,64}@[^@]{1,255}$", $email)) {
    // Email invalid because wrong number of characters
    // in one section or wrong number of @ symbols
        return false;
    }
    // Split it into sections to make life easier
    $email_array = explode("@", $email);
    $local_array = explode(".", $email_array[0]);
    for ($i = 0; $i < sizeof($local_array); $i++) {
        if(!ereg("^(([A-Za-z0-9!#$%&'*+/=?^_`{|}~-][A-Za-z0-9!#$%&'*+/=?^_
            `{|}~\.-]{0,63})|(\"[^(\\|\")]{0,62}\"))$", $local_array[$i])) {
            return false;
        }
    }
    // Check if domain is IP. If not, it should be valid domain name
    if (!ereg("^\[?[0-9\.]+\]?$", $email_array[1])) {
        $domain_array = explode(".", $email_array[1]);
        if (sizeof($domain_array) < 2) {
        return false; // Not enough parts to domain
        }
        for ($i = 0; $i < sizeof($domain_array); $i++) {
            if (!ereg("^(([A-Za-z0-9][A-Za-z0-9-]{0,61}[A-Za-z0-9])|([A-Za-z0-9]+))$"
                , $domain_array[$i])) {
                return false;
            }
        }
    }
return true;
}
```

Algoritmo 4.22 Registro de usuario: definición de variables

```
<?php

require("Smarty/Smarty.class.php");
require("config.php");
require("formprocess.php");

$fields=array(
    array("name"=>"name",
        "type"=>"text",
        "error_text"=>"name_error",
        "disp_name"=>"Name",
        "value"=>"name_value",
        "compul"=>1),
    array("name"=>"email",
        "type"=>"email",
        "error_text"=>"email_error",
        "disp_name"=>"Email",
        "value"=>"email_value",
        "compul"=>1),
    array("name"=>"username",
        "type"=>"text",
        "error_text"=>"username_error",
        "disp_name"=>"User",
        "value"=>"username_value",
        "compul"=>1),
    array("name"=>"password",
        "type"=>"text",
        "error_text"=>"password_error",
        "disp_name"=>"Password",
        "value"=>"password_value",
        "compul"=>1)
);

if(is_user_logged()) session_redirect("logout.php");
```

4.3.4. Registro de usuario

Definidas las funciones generales, ya se puede comenzar a implementar el flujo de un usuario por la página web. Antes de realizar cualquier acción, debe registrarse en el sistema. El código que gestiona dicho proceso se escribirá en el archivo *register.php*.

En primer lugar, se incluyen los archivos necesarios para usar las funciones generales, se define el array que contendrá los valores del formulario para ser comprobado, y se cierra la sesión del usuario en caso de que esté abierta (con una página que se creará más adelante), como se muestra en el algoritmo 4.23.

A continuación, si se ha recibido el formulario, se llama a la función *form_process* en busca de errores. En caso de que no los haya, se recuperan los valores y se comprueba que no es un usuario duplicado buscando filas en la tabla PEOPLE con el mismo nombre de usuario o email. Si hubiera algún error o duplicado, se asignará el mensaje correspondiente a una variable de Smarty y se volverá a mostrar el formulario de registro, esta vez informando sobre los errores (ver algoritmo 4.23).

Finalmente, si todo es correcto, se almacena el usuario en la base de datos, se

Algoritmo 4.23 Registro de usuario: comprobación de errores

```
1  if(isset($_POST["send"])) {
2
3      $err=form_process(); // Look for errors in the resgistration fields
4      if($err) {
5      $smarty->assign("global_error","Please correct the errors below and submit the
           form again");
6      $smarty->display("registerform.tpl");
7      exit;
8      }
9
10     $email=escape_string($_POST["email"]);
11     $name=escape_string($_POST["name"]);
12     $username=escape_string($_POST["username"]);
13     $password=escape_string($_POST["password"]);
14
15     // Check if duplicated
16     $idx=db_query("select * from people where email='$email';");
17     if(db_numrows($idx)>=1) {
18         $smarty->assign("global_error","There is already an user with the address ".
               $email." in the system. If you have forgotten your password, you can <a
               href=\"forget.php\">recover it</a>");
19         $smarty->display("registerform.tpl");
20         exit;
21     }
22     $idx=db_query("select * from people where user='$user';");
23     if(db_numrows($idx)>=1) {
24     $smarty->assign("global_error","There is already an user with the user name ".
           $email." in the system. Please use a different one.");
25     $smarty->display("registerform.tpl");
26     exit;
27     }
```

Algoritmo 4.24 Registro de usuario: almacenamiento y notificación

```
    // Store the user
    db_query("insert into people (user,email,name,password,role) values     ('
        $username','$email','$name','$password',0);"); // 0 means normal user

    // Send confirmation email
    $tpl_vars=array();
    $tpl_vars["username"]=$username;
    $tpl_vars["name"]=$name;
    $tpl_vars["password"]=$password;
    $tpl_vars["email"]=$email;
    global $from_address;
    $message_tosend=template_process("msg/register.txt",$tpl_vars);
    mail($email,"IEEE IES Conferences",$message_tosend,"From: $from_address\r\n");
    $smarty->display("registersubmitted.tpl");
    exit;
    }

$smarty->display("registerform.tpl");

?>
```

le envía un email de notificación usando la plantilla correspondiente y se muestra la página de registro completado, tal y como muestra el código del algoritmo 4.24.

4.3.5. Login/logout de usuario

Una vez que el usuario queda registrado en el sistema, puede conectarse usando su nombre de usuario y contraseña. El código que gestiona el proceso se define en el archivo *login.php*, y puede verse en el algoritmo 4.25, constando simplemente de una función que busca una entrada en la tabla PEOPLE que contenga los atributos usuario y contraseña que se correspondan con los del formulario enviado. Si son correctos, se inicia una sesión y se redirecciona a la página principal. En caso contrario, se vuelve a mostrar el formulario de login con un mensaje de error.

El código que gestiona el logout se define en el archivo *logout.php*, limitándose básicamente a vaciar los parámetros usuario y contraseña de la sesión actual (véase el algoritmo 4.26).

4.3.6. Recuperación de usuario y contraseña

Si el usuario olvida su nombre de usuario y contraseña, podrá recuperarlos introduciendo su email en un formulario. El código del archivo *forget.php,* mostrado en el algoritmo 4.27, se encargará de buscar la fila en la tabla de usuarios que se corresponda con el email, obtener los datos y enviarlos por email usando la plantilla correspondiente. En caso de no existir, se mostrará un mensaje de error asignándolo a una variable de Smarty.

Algoritmo 4.25 Login de usuario

```php
<?php

require("Smarty/Smarty.class.php");
require("config.php");

if(isset($_POST["send"])) {
    // Check user and password
    $user=escape_string($_POST["user"]);
    $password=escape_string($_POST["pass"]);
    $idx=db_query("select * from people where user='$user' and password='$password';");
    if(db_numrows($idx)==1) {
        // Start session
        session_start();
        $_SESSION["user"]=escape_string($_POST["user"]);
        $_SESSION["password"]=escape_string($_POST["pass"]);
        session_regenerate_id(true);
        session_redirect("index.php");
        exit;
    }
    else { // Login incorrect
        $smarty->assign("global_error","User / password combination not found.");
                                $smarty->display("login.tpl");
        exit;
    }
}

$smarty->display("login.tpl");

?>
```

Algoritmo 4.26 Logout de usuario

```php
<?php

require("Smarty/Smarty.class.php");
require("config.php");

$_SESSION["user"]="";
$_SESSION["password"]="";

$smarty->assign("is_user_logged",false);
$smarty->display("logout.tpl");

?>
```

Algoritmo 4.27 Recuperación de usuario y contraseña

```php
<?php

require("Smarty/Smarty.class.php");
require("config.php");

if(isset($_POST["send"])) {
    $email=escape_string($_POST["email"]);
    $idx=db_query("select * from people where email='$email';");
    if(db_numrows($idx)==1) {
        // Get user information
        $fullname=db_result($idx,0,"name");
        $username=db_result($idx,0,"user");
        $email=db_result($idx,0,"email");
        $password=db_result($idx,0,"password");
        $tpl_vars=array();
        $tpl_vars["username"]=$username;
        $tpl_vars["name"]=$name;
        $tpl_vars["password"]=$password;
        $tpl_vars["email"]=$email;
        global $from_address;

        // Send email
        $message_tosend=template_process("msg/forget.txt",$tpl_vars);
        mail($email,"IEEE IES Conferences",$message_tosend,"From: $from_address\r\n")
            ;
        $smarty->display("recovered.tpl");
        exit;
        }
    else {
        $smarty->assign("global_error","Email not found.");
        $smarty->display("forget.tpl");
        exit;
    }
}

$smarty->display("forget.tpl");

?>
```

4.3.7. Realización de una propuesta

Para realizar una propuesta, el usuario sólo tiene que rellenar un formulario con los datos relativos a la misma. Además de los ya definidos como atributos en la base de datos, deberá subir un PDF con información de la conferencia en el formato especificado por el IEEE-IES. El código encargado de gestionar el almacenamiento de la propuesta se definirá en el archivo *submit.php*.

Antes de realizar cualquier operación, se comprobará que el usuario tiene permisos para realizar propuestas con la función *check_perms*. Una vez comprobado, se definirá el array que almacenará los datos del formulario para su comprobación y se obtendrá el nombre y el correo electrónico del usuario que realiza la propuesta, como se muestra en el algoritmo 4.29.

A continuación, se comenzará a procesar los datos del formulario una vez recibidos. Como siempre, se llamará a la función *form_process* en busca de posibles errores. Una vez obtenidos los datos de la conferencia, como se muestra en el algoritmo 4.29, se comprobará que:

- Otro usuario no haya realizado una propuesta usando el mismo identificador corto para la conferencia, que debe ser único.
- En caso de que el mismo usuario esté realizando una nueva revisión de la misma propuesta, se comprobará que la última revisión ya está decidida. En caso contrario, no se permitirá realizar una nueva hasta que se decida.

Por último, una vez comprobado que los datos son correctos, se almacenará la propuesta, en caso de que sea la primera vez que se realiza, y la revisión correspondiente, con el atributo *close_date* tomando el valor "0000-00-00" (votación cerrada por defecto). El PDF se almacenará en la ruta definida en las variables globales, siguiendo una estructura de directorios en árbol determinada por el identificador corto de la conferencia y la versión de la revisión. Como en el caso del registro, se le enviará un email de notificación al usuario, además de otro al administrador (VP). El código del proceso está detallado en el algoritmo 4.30.

También se permitirá al usuario subir directamente una nueva revisión de una propuesta ya realizada, siguiendo el enlace adecuado. La única diferencia, señalada en el algoritmo 4.31, es que los parámetros de la conferencia se obtendrán a partir del identificador pasado como parámetro en el enlace URL, y sólo será necesario obtener del formulario los datos propios de una nueva revisión.

El código completo puede verse en el archivo *submitrevision.php*.

Algoritmo 4.28 Realización de una propuesta: definición de variables

```
<?php

require("Smarty/Smarty.class.php");
require("config.php");
require("formprocess.php");

// Check perms for submitting proposals user_auth();
if(!check_perms("CANSUBMIT")) session_redirect("index.php");

$fields=array(
    array("name"=>"conference",
        "type"=>"text",
        "error_text"=>"conf_error",
        "disp_name"=>"Conference Name",
        "value"=>"conf_value",
        "compul"=>1),
    array("name"=>"confid",
        "type"=>"text",
        "error_text"=>"confid_error",
        "disp_name"=>"Short conference ID",
        "value"=>"confid_value",
        "compul"=>1),
    array("name"=>"confdates",
        "type"=>"date",
        "error_text"=>"confdates_error",
        "disp_name"=>"Conference dates",
        "value"=>"confdates_value",
        "compul"=>0),
    array("name"=>"confurl",
        "type"=>"url",
        "error_text"=>"confurl_error",
        "disp_name"=>"Conference URL",
        "value"=>"confurl_value",
        "compul"=>1),
    array("name"=>"cfp",
        "type"=>"file",
        "mime"=>"application/pdf",
        "error_text"=>"cfp_error",
        "disp_name"=>"Call for papers",
        "compul"=>1),
    array("name"=>"comments",
        "type"=>"text",
        "error_text"=>"comments_error",
        "disp_name"=>"Comments",
        "value"=>"comments_value",
        "compul"=>0),
);

$close=date("0000-00-00");

// Get proposer name and email $idx=db_query("select * from people where user='$user
    ';");
$fullname=db_result($idx,0,"name");
$smarty->assign("name_value",$fullname);
$email=db_result($idx,0,"email");
$smarty->assign("email_value",$email);
$proposerid=db_result($idx,0,"id");
```

Algoritmo 4.29 Realización de una propuesta: comprobación de datos

```
if(isset($_POST["send"])) {
    $err=form_process(); // Look for errors in the proposal fields
    if($err) {
        $smarty->assign("global_error","Please correct the errors below and submit
            the form again");
        $smarty->display("proposalform.tpl");
        exit;
    }

    $time_received=date(DATE_ISO8601);
    $confid=addslashes(strtoupper($_POST["confid"]));
    $conference=addslashes($_POST["conference"]);
    $conf_date=addslashes($_POST["confdates"]);
    $conf_url=addslashes($_POST["confurl"]);
    $comments=addslashes($_POST["comments"]);

    // Check if duplicated
    $version=0;
    $idx=db_query("select * from proposals where confid='$confid';");
    if(db_numrows($idx)>=1) {
        $proposer=db_result($idx,0,"proposer");
        if($proposer==$proposerid) {
        // Check that previous version was decided and revision requested
        $id=db_result($idx,0,"id");
        $idx=db_query("select * from revisions where conference='$id';");
        $version=db_numrows($idx)-1;
        if(is_proposal_decided($id)) $version++;
        else {
            $smarty->assign("global_error","You have already submitted the proposal
                for $confid. Please do not submit it again until you have a reply.");
            $smarty->display("proposalform.tpl");
            exit;
        }
        }
        else {
            $smarty->assign("global_error","Someone has already submitted a proposal
                for $confid. Please, use another short ID.");
            $smarty->display("proposalform.tpl");
            exit;
        }
    }
```

Algoritmo 4.30 Realización de una propuesta: almacenamiento y notificación

```
    // Store the request
    if($version==0) $idx=db_query("insert into proposals (date_start,proposer,
        fullname,confid,url) values ('$conf_date','$proposerid','$conference','
        $confid','$conf_url');");
    $idx=db_query("select id from proposals where confid='$confid';");
    $id=db_result($idx,0,0);
    $idx=db_query("insert into revisions (conference, version, date_submitted,
        close_date, comments) values ('$id','$version', '$time_received', '$close', '
        $comments');");

    // Store attached files
    mkdir($cb_attached."/".$confid."/".$version,0770,true);
    $uploadfile=$cb_attached."/".$confid."/".$version."/cfp.pdf";
    move_uploaded_file($_FILES["cfp"]["tmp_name"],$uploadfile);

    // Send confirmation email
    $tpl_vars=array();
    global $from_address,$email,$fullname,$vpw_address;
    $tpl_vars["confid"]=$confid;
    $tpl_vars["name"]=$fullname;
    $message_tosend=template_process("msg/submit.txt",$tpl_vars);
    mail($email,"IEEE IES Conference proposal",$message_tosend,"From: $from_address\r
        \n");
    $message_tosend=template_process("msg/submit-vpw.txt",$tpl_vars);
    mail($vpw_address,"IEEE IES New conference proposal",$message_tosend,"From:
        $from_address\r\n");
    $smarty->display("proposalsubmitted.tpl");
    exit;
}

$smarty->display("proposalform.tpl");

?>
```

Algoritmo 4.31 Realización de una propuesta: nueva revisión

```
// Get proposal id
if (isset($_GET["id"])){
$id=intval($_GET["id"]);
$idx=db_query("select * from proposals where proposer='$proposerid' and id='$id';");
$confid=db_result($idx,0,"confid"); $smarty->assign("confid",$confid);
}
```

Algoritmo 4.32 Apertura de una propuesta a votación

```php
<?php

require("Smarty/Smarty.class.php");
require("config.php");

// Check perms for deciding proposals
user_auth();
if(!check_perms("CANDECIDE")) session_redirect("index.php");

// Get proposal id and version
$id=intval($_GET["id"]);
$idx=db_query("select * from revisions where conference='$id';");
$version=db_numrows($idx)-1;

// Get close date
$today=date("Y-m-d",time());
$close=date("Y-m-d",strtotime($today."+1 month"));

if(is_proposal_open($id) && !is_proposal_decided($id)) {
    $msg="This proposal is already open for voting.";
}
elseif (is_proposal_decided($id)) {
    // Reopen votation
    db_query("update revisions set close_date='$close' where conference='$id' and
        version='$version';");
    $msg="The proposal has been opened for voting, even when it was already decided."
        ;
}
else {
    // Open votation
    db_query("update revisions set close_date='$close' where conference='$id' and
        version='$version';");
    $msg="The proposal has been opened for voting.";
}

$smarty->assign("msg",$msg); $smarty->display("openclose.tpl");

?>
```

4.3.8. Apertura/cierre de votaciones

La apertura y cierre de votaciones se realiza de forma muy simple modificando el atributo *close_ date* de las revisiones. Para abrir la última revisión de una propuesta a votación, no hay más que calcular una fecha de cierre y almacenarla en dicho atributo. En principio se ha considerado un período de votaciones de un mes. Si la propuesta ya estaba decidida, se considerará que puede volver abrirse a votación si el administrador así lo desea, aunque se advertirá de ello con un mensaje cuando se realice la operación.

Como siempre, será muy importante comenzar comprobando que el usuario de la sesión en marcha tiene permisos para decidir sobre las propuestas. El código del proceso completo se muestra en el algoritmo 4.32.

Para cerrar la votación, no habrá más que volver a poner el atributo *close_ date* de la última revisión de la propuesta a cero, siguiendo los mismos preceptos de seguridad

Algoritmo 4.33 Cierre de una votación

```php
<?php

require("Smarty/Smarty.class.php");
require("config.php");

// Check perms for deciding proposals
user_auth();
if(!check_perms("CANDECIDE")) session_redirect("index.php");

// Get last revision
$id=intval($_GET["id"]);
$idx=db_query("select * from revisions where conference='$id';");
$version=db_numrows($idx)-1;
$close=date("0000-00-00");

// Check if proposal is open for votation
if(is_proposal_open($id)) {
    //Close votation
    db_query("update revisions set close_date='$close' where conference='$id' and
        version='$version';");
    $msg="Voting closed for this proposal.";
}
else $msg="The proposal was not open.";

$smarty->assign("msg",$msg); $smarty->display("openclose.tpl");

?>
```

considerados en el caso de la apertura y como se muestra en el algoritmo 4.33.

4.3.9. Votación de propuestas

El código que gestiona el proceso de votación estará recogido en el archivo *vote.php*. Análogamente al resto de acciones, se comenzará comprobando que el usuario de la sesión actual tiene permisos para votar. Seguidamente, se obtendrán los datos de la propuesta que se desea votar a partir de la id que se pasa como parámetro (ver algoritmo 4.35).

A continuación se comprobará que la propuesta está abierta a votación y que todavía no se ha decidido. Además, se obtendrán los tipos de voto de la tabla VOTE_TYPES para asignarlos a variables Smarty, con el fin de mostrarlos al usuario para que pueda elegir su voto, tal y como se muestra en el algoritmo 4.35.

Cuando se haya recibido el voto, se llamará a la función *cast_vote* para almacenarlo. Dicha función, mostrada en los algoritmos 4.36 y 4.37, comenzará obteniendo el tipo de voto y su contenido, así como la información del votante y los posibles comentarios.

Para finalizar, se almacenará el voto (actualizándose si el usuario ya había votado anteriormente) y se enviará un email de notificación. Si no se hubiera recibido el voto,

Algoritmo 4.34 Votación de propuestas: definición de variables

```php
<?php

require("Smarty/Smarty.class.php");
require("config.php");

// Check perms for voting proposals
user_auth();
if(!check_perms("CANVOTE")) session_redirect("index.php");

global $id,$version,$confid,$confname;
// Get proposal id
if(isset($_POST["submit"])) $id=intval($_POST["id"]);
else $id=intval($_GET["id"]);

// Get proposal information
$idx=db_query("select * from proposals where id='$id';");
if(db_numrows($idx)<1) exit;
$confid=db_result($idx,0,"confid");
$confname=db_result($idx,0,"fullname");
$idx=db_query("select * from revisions where conference='$id';");
$version=db_numrows($idx)-1;
$today=date("Y-m-d",time());

$smarty->assign("confid",$confid);
$smarty->assign("conftitle",$confname);
```

Algoritmo 4.35 Votación de propuestas: comprobación y tipo de votos

```php
$error=0;

// Check proposal is open and not decided
if(!is_proposal_open($id)) {
    $msg="This proposal is not open for voting at this moment.";
    $error=1;
}
elseif (is_proposal_decided($id)) {
    $msg="The proposal was already decided.";
    $error=1;
}
else {
    // Display voting options
    $idx=db_query("select * from vote_types;");
    $num=db_numrows($idx);
    for($i=0;$i<$num;++$i) {
        $a[$i]["id"]=db_result($idx,$i,"id");
        $a[$i]["content"]=db_result($idx,$i,"content");
    }
    $smarty->assign("vote_types",$a);
    $smarty->assign("id",$id);
}
```

Algoritmo 4.36 Votación de propuestas: obtención del voto

```
if(isset($_POST["submit"])) {
    cast_vote();
    exit;
}

function cast_vote() {
    global $id,$version,$confid,$confname,$_POST,$user,$smarty;
    // Get vote id
    $idvote=intval($_POST["vote"]);
    // Get vote content
    if($idvote<=0) exit;
    $idx=db_query("select * from vote_types where id='$idvote';");
    if(db_numrows($idx)<1)  exit;
    $vote_content=db_result($idx,0,"content");
    // Get user information
    $idx=db_query("select * from people where user='$user';");
    $iduser=db_result($idx,0,"id");
    $fullname=db_result($idx,0,"name");
    $email=db_result($idx,0,"email");
    $comments=addslashes($_POST["comments"]);
```

el usuario estará esperando que se le muestren las opciones de voto, y se llamará a Smarty para hacerlo.

4.3.10. Decisión de una propuesta

El código que gestiona el proceso de toma de decisiones será prácticamente idéntico al de las votaciones, y se encontrará en el archivo *decide.php*. De nuevo, se comenzará comprobando tanto que el usuario tiene permisos para decidir como que la propuesta no estaba ya decidica, y se obtendrán los tipos de decisiones de la tabla DECISION_TYPES, exactamente igual que en caso de las votaciones.

La principal diferencia se encontrará en la función *cast_decision* (véase el algoritmo 4.38) que almacena la decisión cuando se recibe, que además tiene que cerrar la votación y en este caso, enviar un email al usuario que realizó la propuesta.

4.3.11. Visionado de propuestas

En el archivo *proposals.php* se recoge el código que prepara el visionado general de las propuestas, en el que únicamente se mostrará el nombre, el identificador corto, el estado y algunos botones de acción según los permisos del usuario (votar, decidir...). Para que la lista no sea demasiado extensa, se supondrá una división por páginas, obteniendo la página que se desea ver mediante un parámetro pasado por URL.

El primer paso será, como siempre, comprobar que el usuario tiene permisos para ver la lista de propuestas. A continuación, como se detalla en el algoritmo 4.40, se obtendrán las variables necesarias: el número de página a mostrar, el identificador

Algoritmo 4.37 Votación de propuestas: almacenamiento del voto

```
    // Check for duplicate
    if(has_voted($iduser,$id,$version)) {
        db_query("update votes set conference='$id', version='$version', person='
            $iduser', vote='$idvote',comments='$comments'  where person='$iduser' and
             conference='$id' and version='$version';");
    } else {
    db_query("insert into votes (conference,version,person,vote,comments) values ('
        $id','$version','$iduser','$idvote','$comments');");
    }

    // Send confirmation email
    $tpl_vars=array();
    global $from_address,$email,$fullname;
    $tpl_vars["confid"]=$confshort;
    $tpl_vars["name"]=$fullname;
    $tpl_vars["vote"]=$vote_content;
    $message_tosend=template_process("msg/submit.txt",$tpl_vars);
    mail($email,"IEEE IES Voting confirmation",$message_tosend,"From: $from_address\r
        \n");
    $smarty->assign("comments",$comments);
    $smarty->assign("content",$vote_content);
    $smarty->assign("confname",$confname);
    $smarty->display("votecast.tpl");
}

$smarty->assign("error",$error); $smarty->assign("msg",$msg); $smarty->assign("confid
    ",$confid); $smarty->display("vote.tpl");

?>
```

de usuario, todas las propuestas ordenadas por identificador, el número de páginas totales... Además, se define el parámetro *$P*, cuyo valor determina cuántas propuestas se muestran por página.

Seguidamente se recorrerá mediante un bucle el rango de propuestas dentro de la página seleccionada, obteniendo para cada una la información a mostrar, y asignando los datos a un array que será enviado a la plantilla HTML a través de Smarty (ver algoritmo 4.40).

También se definirán en el array los permisos del usuario, con el fin de determinar las acciones disponibles que se mostrarán, para finalmente asignar toda la información a variables Smarty y mostrar la plantilla HTML correspondiente, de la forma mostrada en el algoritmo 4.41.

Además, también se le permitirá al usuario la opción de ver únicamente las propuestas que él mismo ha realizado. El código es exactamente igual al ya expuesto, salvo que en la consulta a la base de datos para obtener las propuestas se añade la condición de coincidencia entre el identificador del usuario y el realizador de la propuesta. El código completo puede verse en el archivo *yourproposals.php*.

Algoritmo 4.38 Decisión de una propuesta

```
function cast_decision() {
    global $id,$version,$confid,$confname,$_POST,$user,$smarty;

    // Get decision id and comments
    $user=$_SESSION["user"];
    $iddecision=intval($_POST["decision"]);
    $comments=addslashes($_POST["comments"]);
    // Get decision content
    if($iddecision<=0) exit;
    $idx=db_query("select * from decision_types where decision='$iddecision';");
    if(db_numrows($idx)<1)  exit;
    $decision_content=db_result($idx,0,"content");
    $idx=db_query("select * from people where user='$user';");
    $iduser=db_result($idx,0,"id");
    // Insert decision into the db
    db_query("insert into decisions (conference,version,decision,person,comments)
        values     ('$id','$version','$iddecision','$iduser','$comments')");
    // Close votation
    if(is_proposal_open($id))  {
        $close=date("0000-00-00");
        db_query("update revisions set close_date='$close' where conference='$id' and
            version='$version';");
    }
    // Send email to organizer
    $idx=db_query("select * from proposals where id='$id';");
    $proposer=db_result($idx,0,"proposer");
    $idx=db_query("select * from people where id='$proposer';");
    $proposername=db_result($idx,0,"name");
    $proposermail=db_result($idx,0,"email");
    $tpl_vars=array();
    global $from_address;
    $tpl_vars["confid"]=$confid;
    $tpl_vars["name"]=$proposername;
    $tpl_vars["content"]=$decision_content;
    $message_tosend=template_process("msg/decided.txt",$tpl_vars);
    mail($proposermail,"IEEE IES Voting confirmation",$message_tosend,"From:
        $from_address\r\n");
    $smarty->assign("comments",$comments);
    $smarty->assign("content",$decision_content);
    $smarty->assign("confname",$confname);
    $smarty->display("decidecast.tpl");
}
```

Algoritmo 4.39 Visionado de propuestas: definición de variables

```php
<?php

require("Smarty/Smarty.class.php");
require("config.php");

// Check user perms
user_auth();
if(!check_perms("CANVIEWSUBM")) session_redirect("index.php");

$P=5; // Proposals per page

// Get page
if (isset($_GET["page"])){
    $page=intval($_GET["page"]);
    if ($page>1) $prevpage=$page-1;
} else { // By default, show first page
    $page=1;
}

// Get user id
$idx=db_query("select id from people where user='$user';");
$iduser=db_result($idx,0,0);

// Get proposals
$idx=db_query("select * from proposals order by proposals.id DESC;");
$num=db_numrows($idx);
if ($num>$P*$page) $nextpage=$page+1;
$pages=ceil($num/$P);
```

Algoritmo 4.40 Visionado de propuestas: obtención de propuestas

```php
// Get the information of proposals on the selected page
for($i=$P*($page-1);($i<$P*$page)&&($i<$num);$i++) {
    // Get proposals information
    $id=db_result($idx,$i,"id");
    $name=db_result($idx,$i,"fullname");
    $confid=db_result($idx,$i,"confid");
    $date=db_result($idx,$i,"date_start");
    $proposer=db_result($idx,$i,"proposer");

    $j=$i-$P*($page-1);
    $a[$j]=array("name"=>$name, "confid"=>$confid, "date"=>$date, "id"=>$id);

    // Get proposals status and user perms
    if(is_proposal_decided($id) && !is_proposal_open($id)) {
        $a[$j]["status"]="Decided";
        $a[$j]["candecide"]=0;
        $a[$j]["canvote"]=0;
    } elseif (is_proposal_open($id)) {
        $a[$j]["status"]="Voting";
    } else {
    $a[$j]["status"]="Pending";
    }
```

Algoritmo 4.41 Visionado de propuestas: permisos y asignación de variables Smarty

```
    if(check_perms("CANVIEWSUBM") || $proposer==$iduser) {
        $a[$j]["canviewsubm"]=1;
    }
    if(check_perms("CANVIEWSTATUS") || $proposer==$iduser) {
        $a[$j]["canviewstatus"]=1;
    }
    if(check_perms("CANVOTE") && is_proposal_open($id)) {
        $a[$j]["votationid"]=$id;
        $a[$j]["canvote"]=1;
    }
    if(check_perms("CANDECIDE") && !is_proposal_decided($id)) {
        $a[$j]["candecide"]=1;
        if(is_proposal_open($id)) {
            $a[$j]["votationid"]=$id;
            $a[$j]["canclose"]=1;
        } else {
            $a[$j]["canopen"]=1;
        }
    }
}

// Display proposals
$smarty->assign("proposals",$a);
$smarty->assign("page",$page);
$smarty->assign("pages",$pages);
$smarty->assign("nextpage",$nextpage);
$smarty->assign("prevpage",$prevpage);
$smarty->display("proposals.tpl");

?>
```

Algoritmo 4.42 Visionado detallado de una propuesta: definición de variables

```php
<?php

require("Smarty/Smarty.class.php");
require("config.php");

// Check user perms
user_auth();
if(!check_perms("CANVIEWSUMB")) session_redirect("index.php");

// Get proposal id
$id=intval($_GET["id"]);
$smarty->assign("id",$id);
// Get user id
$idx=db_query("select id from people where user='$user';");
$iduser=db_result($idx,0,0);
// Get proposal information
$idx=db_query("select * from proposals where id='$id';");
if(db_numrows($idx)!=1) exit;
$proposer=db_result($idx,0,"proposer");
$fullname=db_result($idx,0,"fullname");
$conf=db_result($idx,0,"confid");
$date_start=db_result($idx,0,"date_start");
$url=db_result($idx,0,"url");
$idx=db_query("select * from revisions where conference='$id';");
$last_version=db_numrows($idx)-1;
// Get proposer information
$idx=db_query("select * from people where id='$proposer'");
$proposername=db_result($idx,0,"name");
$proposer_email=db_result($idx,0,"email");
$smarty->assign("proposer",$proposername);
$smarty->assign("proposer_email",$proposer_email);
```

4.3.12. Visionado detallado de una propuesta

El código que gestiona el visionado detallado de una propuesta se recoge en el archivo *proposal.php*. Como en la mayoría de los casos, será necesario comenzar comprobando que el usuario tiene permisos para ver la propuesta. A continuación se obtendrá la información de la propuesta mediante consultas a la base de datos, tal y como se comienza en el algoritmo 4.43.

Mediante un parámetro enviado por URL, se le permitirá al usuario seleccionar la revisión de la propuesta que desea visionar. Por defecto se mostrará la última versión, tal y como se procede en el algoritmo 4.43.

A continuación se contarán los votos asociados a la revisión de la propuesta que se desea visionar, almacenando en un array la información del contenido y el votante para asignarlo a una variable Smarty (ver algoritmo 4.44).

Finalmente, como se muestra en el algoritmo 4.45, se obtendrá el estado de la propuesta comprobando si está decidida o abierta a votación, y se asignarán a variables Smarty los permisos del usuario para determinar qué opciones se mostrarán.

Algoritmo 4.43 Visionado detallado de una propuesta: versión

```
// Get proposal version
if (isset($_GET["version"])){
    $version=intval($_GET["version"]);
    if ($version<$last_version) $next_version=$version+1;
    if ($version>0) $prev_version=$version-1;
} else {
    // By default, show last version
    $version=$last_version;
    if ($version>0) $prev_version=$version-1;
}

// Get version information
$comments=db_result($idx,$version,"comments"); $submitted=db_result($idx,$version,"
    date_submitted");
// Send information to Smarty $smarty->assign("fullname",$fullname);
$smarty->assign("confid",$conf);
$smarty->assign("url",$url);
$smarty->assign("date_start",$date_start);
$smarty->assign("comments",$comments);
$smarty->assign("date_subm",$submitted);
$smarty->assign("version", $version);
$smarty->assign("prev_version",$prev_version);
$smarty->assign("next_version",$next_version);
```

Algoritmo 4.44 Visionado detallado de una propuesta: recuento de votos

```
// Get vote types
$idxvt=db_query("select * from vote_types;");
$num=db_numrows($idxvt);
for($j=0;$j<$num;++$j) {
    $idchoice=db_result($idxvt,$j,"id");
    $totalvotes[$idchoice]=0;
    $namevotes[$idchoice]=db_result($idxvt,$j,"content");
}

// Count votes
$idx=db_query("select * from revisions where conference='$id';");
$numvotations=db_numrows($idx);
if($numvotations>=1) {
    $idx2=db_query("select * from votes where conference='$id' and version='$version
        ';");
    $num=db_numrows($idx2);
    for($i=0;$i<$num;++$i) {
        $idperson=db_result($idx2,$i,"person");
        $idx3=db_query("select name from people where id='$idperson';");
        $v[$i]["person"]=db_result($idx3,0,"name");
        $idchoice=db_result($idx2,$i,"vote");
        $idx3=db_query("select content from vote_types where id='$idchoice';");
        $v[$i]["vote"]=db_result($idx3,0,"content");
        $v[$i]["comments"]=db_result($idx2,$i,"comments");
        $totalvotes[$idchoice]++;
    }
    $smarty->assign("v",$v);
    $smarty->assign("totalvotes",$totalvotes);
    $smarty->assign("namevotes",$namevotes);
}
```

Algoritmo 4.45 Visionado detallado de una propuesta: estado y permisos

```
1  if((is_proposal_decided($id) && !is_proposal_open($id)) || $version<$last_version) {
       $idx=db_query("select decision from decisions where conference='$id' and
       version='$version';");
2      $decision=db_result($idx,0,0);
3      $idx=db_query("select content from decision_types where decision='$decision';");
4      $status=db_result($idx,0,0);
5      $smarty->assign("status",$status);
6  } elseif (is_proposal_open($id)) {
7      $smarty->assign("status","Voting");
8  } else {
9      $smarty->assign("status","Pending");
10 }
11
12 // Send user permissions to Smarty
13 if(check_perms("CANVIEWSTATUS") || $proposer==$iduser) $smarty->assign("canviewstatus
       ",1);
14 if(check_perms("CANVIEWSTATUS"))     $smarty->assign("canviewvotes",1);
15 if(check_perms("CANVIEWSUBM") || $proposer==$iduser)     $smarty->assign("canviewsubm"
       ,1);
16 if(check_perms("CANVOTE") && is_proposal_open($id) && $version==$last_version)
       $smarty->assign("canvote",1);
17 if(check_perms("CANDECIDE") && !is_proposal_open($id) && !is_proposal_decided($id) &&
        $version==$last_version) $smarty->assign("canopen",1);
18 if(check_perms("CANDECIDE") && is_proposal_open($id) && $version==$last_version)
       $smarty->assign("canclose",1);
19
20 $smarty->display("proposal.tpl");
21
22 ?>
```

4.3.13. Resumen: listado y función de archivos PHP

En la siguiente tabla se recoge un listado con el nombre de todos los archivos PHP que generan el contenido dinámico de la web y la función específica de cada uno de ellos.

Nombre del archivo	Función
index.php	Muestra la página principal de la web
config.php	Contiene definición de variables y funciones globales
formprocess.php	Busca errores en los formularios
register.php	Gestiona el proceso de registro de un nuevo usuario
forget.php	Permite recuperar usuario y contraseña
login.php	Permite a un usuario hacer login
logout.php	Permite a un usuario hacer logout
submit.php	Procesa la realización de una nueva propuesta
submitrevision.php	Procesa la realización de una nueva revisión de propuesta
open.php	Abre una propuesta a votación
close.php	Cierra la votación de una propuesta
vote.php	Gestiona la votación de una propuesta
decide.php	Gestiona la decisión de una propuesta
proposals.php	Muestra todas las propuestas
yourproposals.php	Muestra las propuestas realizadas por el usuario
proposal.php	Muestra los detalles de una propuesta concreta

Cuadro 4.2: Listado y función de archivos PHP

4.4. Estructura y diseño de la página: HTML/CSS

Una vez generado el contenido dinámico, para finalizar el diseño sólo queda definir cómo se estructura la información que se mostrará en la página web. Como ya se explicó, se usará el lenguaje de marcado HTML para estructurar la información, en las plantillas Smarty a las que se hace referencia en el código PHP, y CSS para definir el estilo de las distintas partes de la estructura de la página web.

4.4.1. Estructura principal

Aunque en general será necesario implementar plantillas para cada una de las distintas funciones que ofrece el sistema web (registro de usuarios, realización de propuestas, votaciones...), toda la página seguirá una estructura común. Esta estructura consiste básicamente en un encabezado, un menú de navegación lateral y el pie de página, quedando el resto del espacio libre para el contenido que ofrezcan las distintas funciones. El resultado puede verse en la figura 4.3.

4.4.1.1. Encabezado

La estructura del encabezado se creará en el archivo *header.tpl*. Antes de nada, se define el encabezado HTML, con información general del documento como el título

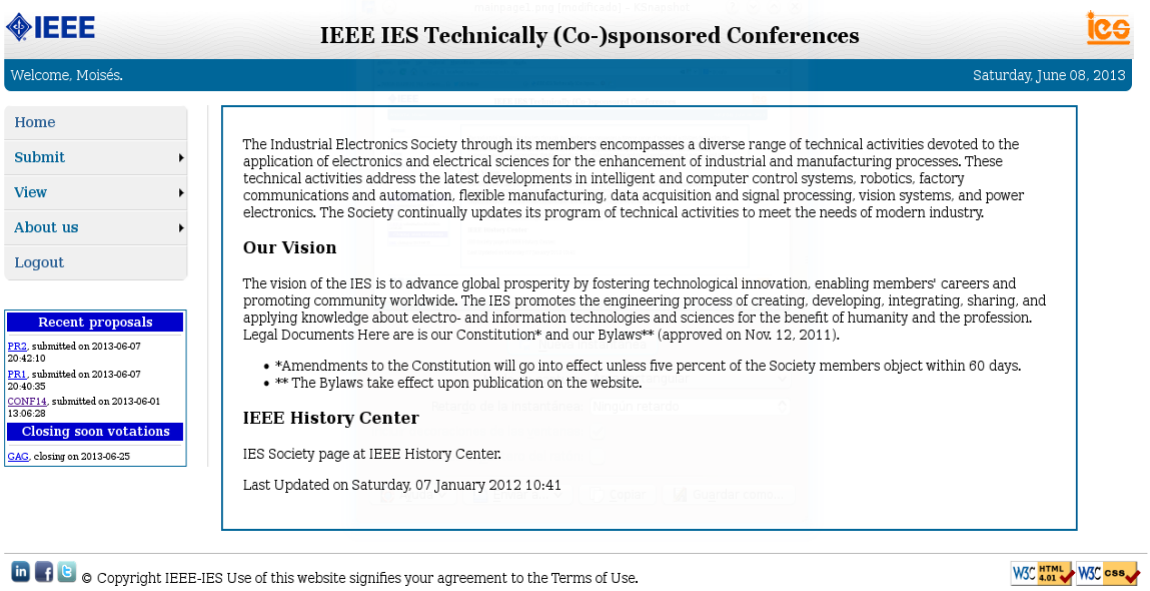

Figura 4.3: Estructura principal de la página web

de la página web y los archivos CSS que se incluyen para determinar el estilo.

A continuación, abriendo ya la etiqueta para el cuerpo de la página, se define el encabezado de la web. Como puede verse en el algoritmo 4.46, estará compuesto por un título, los logos del IEEE y el IES, y una pequeña barra donde se muestra un mensaje de bienvenida con el nombre del usuario y la fecha actual.

El estilo de cada elemento queda definido en el archivo *style.css* como se muestra en el algoritmo 4.47:

4.4.1.2. Menú de navegación

El objetivo del menú de navegación será ofrecer al usuario una interfaz gráfica para navegar por las distintas funciones disponibles en la página web. Se definirá también en el archivo *header.tpl*, a continuación de la cabecera.

La primera parte consistirá en un menú desplegable, que ofrecerá enlaces para realizar las acciones disponibles de cara al usuario, según esté registrado o no y sus permisos. Para ello se hará uso de las listas HTML, como puede apreciarse en el algoritmo 4.48.

Para el estilo del menú desplegable se ha utilizado el framework *Free CSS Drop-Drown Menu*, liberado con licencia dual MIT/GNU GPL, modificándolo ligeramente para adaptarlo al presente proyecto. El código completo puede encontrarse en los archivos *menu.css* y *menustyle.css*.

Algoritmo 4.46 Encabezado de la página web

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"    "http://www.w3.org/
    TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
        <meta content="" name=keywords> <link rel=stylesheet type="text/css" href="
            style.css">
        <link rel=stylesheet type="text/css" href="menu.css"> <link rel=stylesheet
            type="text/css" href="menustyle.css">
        <title>IEEE IES Technically (Co-)sponsored Conferences</title>
    </head>

<body>
    <img class="imgleft" src="images/ieee.png" alt="IEEE"> <img class="imgright" src=
        "images/ies.png" alt="IES">
    <h1>IEEE IES Technically (Co-)sponsored Conferences</h1>
    <div id="topbar" class="topbar">
        {if $is_user_logged}
            <span style="float:left;">Welcome, {$username}.</span>
        {/if}
    {$today}
    </div>
```

Algoritmo 4.47 Estilo del encabezado

```
body {
 margin-left: 3em;
 margin-right: 5em;
}

h1 {
 font-family: Georgia, 'Times New Roman', Times, serif;
 text-align: center;  font-size: 1.6em;
 padding: 0.5em;
 background: #fff url(images/pattern-wavy-grad.png) 0 0 repeat-x;
 margin-bottom:0;
}

.imgleft { float: left; }
.imgright { float: right; }

div.topbar {
 padding: 0.5em;
 width: 98.7%;
 background: #006699;
 margin-bottom: 1em;
 color: white;
 text-align: right;
 border-bottom-left-radius: 0.5em;
 border-bottom-right-radius: 0.5em;
}
```

Algoritmo 4.48 Menú desplegable

```
<div id="nav">
    <div>
        <ul id="menu" class="menu menu-vertical">
            <li class="top"><a href="index.php">Home</a></li>
            {if $is_user_logged}
            <li><span class="dir">Submit</span>
                <ul>
                    <li><a href="submit.php">New proposal</a></li>
                    <li><a href="submitrevision.php">Proposal revision</a></li>
                </ul>
            </li>
            <li><span class="dir">View</span>
                    <ul>
                        <li><a href="proposals.php">Proposals</a></li>
                        <li><a href="yourproposals.php">Your proposals</a></li>
                    </ul>
            </li>
            <li><span class="dir">About us</span>
                <ul>
                    <li><a href="http://www.ieee.org">IEEE</a></li>
                    <li><a href="http://ieee-ies.org">IES</a></li>
                </ul>
            </li>
            <li class="bottom"><a href="logout.php">Logout</a></li>
            {else}
            <li><span class="dir">About us</span>
                <ul>
                    <li><a href="http://www.ieee.org">IEEE</a></li>
                    <li><a href="http://ieee-ies.org">IES</a></li>
                </ul>
            </li>
            <li class="bottom"><a href="login.php">Login</a></li>
            {/if}
        </ul>
    </div>
```

Algoritmo 4.49 Menú de navegación: actividades recientes

```
1     <div style="float:left;">
2         <table class="nav">
3             <tr><th>Recent proposals <tr>
4             <td><div class="separator"></div>
5             {section name=i loop=$recentprop}
6             {if ($is_user_logged && $recentprop[i].canviewsubm)}
7             <tr><td>
8                 <a href="proposal.php?id={$recentprop[i].id}">{$recentprop[i].confid}
                      </a>, submitted on {$recentprop[i].date}
9             {/if}
10            {/section}
11            <tr><th>Closing soon votations
12            <tr><td><div class="separator"></div>
13            {section name=i loop=$closingvot}
14            {if $closingvot[i].canvote}
15            <tr><td><a href="vote.php?id={$recentprop[i].id}">{$closingvot[i].confid}
                  </a>, closing on {$closingvot[i].date}
16            {/if}
17            {/section}
18        </table>
19    </div>
20</div>
21
22<div id="content">
```

La segunda parte del menú de navegación consiste en una tabla con accesos directos a las propuestas más recientes y a las votaciones cercanas a su cierre. Finalmente, se abrirá la etiqueta para el contenido de la página, que vendrá determinado por la plantilla correspondiente a cada función (ver algoritmo 4.49).

El estilo de las tablas del menú de navegación, del propio menú y del cuadro que encerrará el contenido, estará definido en el archivo *style.css*, como se muestra en el algoritmo 4.50.

4.4.1.3. Pie de página

En el pie de página, definido en el archivo *end.tpl*, se incluyen enlaces a perfiles del IEEE-IES y un aviso informativo sobre los términos de uso. Además, se muestran botones de validación tanto HTML como CSS, que enlazan a la herramienta del W3C que permite comprobar que la web está correctamente estructurada conforme a los estándares (ver algoritmo 4.51).

El estilo de los elementos que aparecen en el pie de página queda definido en el archivo *style.css*, conforme al algoritmo 4.52.

4.4.2. Página principal

La página principal simplemente contiene un mensaje de bienvenida en texto simple, dentro del contenido de la estructura principal. La estructura puede verse en el

Algoritmo 4.50 Estilo del menú de navegación

```
#nav {
 float:left;
 width:18%;
 border-right: solid 1px #d9d9d9;
}

table.nav {
 border-style: solid;
 border-width: thin;
 border-color: #006699;
 padding: 0.05em;
 margin-top: 2em;
 width: 90%;
}

table.nav th {
 background-color: #0000cc;
 padding: 0.1em;
 padding-left: 0.4em;
 padding-right: 0.4em;
 color: #fff;
 font-weight: bold;
 font-size: 0.9em;
}

table.nav td {   font-size: 0.7em; }

table.nav td:hover {   background-color: #ffffcc; }

div.separator {
 border-bottom: 1px solid #d9d9d9;
 margin: 0.1em;
}

#content {
 float: left;
 width: 72%;
 min-height: 25em;
 padding: 1.5em;
 padding-top: 1em;
 margin-left: 1em;
 border: 0.15em solid #006699;
}
```

Algoritmo 4.51 Estructura del pie de página

```
</div>

<div id="footer">

<a href="http://www.linkedin.com/groups?gid=3124640" target="_blank"><img src="images
    /linkedin.png" alt="linkedin"></a>
<a href="http://www.facebook.com/pages/IEEE-Industrial-Electronics-Society
    /129894117039826" target="_blank"><img src="images/facebook.png" alt="facebook"
    ></a>
<a href="http://twitter.com/ieee_ies" target="_blank"><img src="images/twitter.png"
    alt="twitter"></a>

&copy; Copyright IEEE-IES Use of this website signifies your agreement to the Terms
    of Use.

<a href="http://jigsaw.w3.org/css-validator/check/referer" target="_black"> <img
    class="validation" src="http://jigsaw.w3.org/css-validator/images/vcss" alt="¡CSS
     Válido!" > </a>
<a href="http://validator.w3.org/check?uri=referer" target="_blank"> <img class="
    validation" src="http://www.w3.org/Icons/valid-html401" alt="Valid HTML 4.01
    Transitional"> </a>

</div>
</body>
</html>
```

Algoritmo 4.52 Estilo del pie de página

```
#footer {
 width: 100%;
 float: left;
 margin-top: 1.5em;
 margin-bottom: 1em;
 padding: 0.5em;
 text-align: left;
 border-top: solid 1px #b9b9b9;
}

#footer img {
 width: 1.5em;
 margin-left: auto;
 margin-right: auto;
}

#footer img.validation {
 width: 5em;
 float: right;
}
```

Algoritmo 4.53 Estructura de la página de registro

```
{include file="header.tpl"}

<p> Please fill in the fields below to register in the system.
<div class="error">{$global_error}</div>

<p> <form enctype="multipart/form-data" method="post" action="register.php">
    <table class="maintable">
        <tr> <td><div class="sep">Personal data</div>
        <tr> <td class="question">Your real name: <td class="reply"><div class="error
            ">{$name_error}</div><input type="text" name="name" size=40 value="{
            $name_value}">
        <tr> <td class="question">Email address: <td class="reply"><div class="error"
            >{$email_error}</div><input type="text" name="email" size=40 value="{
            $email_value}">
        <tr> <td class="question">Preferred username: <td class="reply"><div class="
            error">{$username_error}</div><input type="text" name="username" size=40
            value="{$username_value}">
        <tr> <td class="question">Choose a password: <td class="reply"><div class="
            error">{$password_error}</div><input type="text" name="password" size=40
            value="{$password_value}">
        <tr> <td><td align="right"> <input type="submit" name="send" value="Send">
    </table>
</form>

{include file="end.tpl"}
```

archivo *index.html*.

4.4.3. Registro de usuario

La página de registro de usuario seguirá la estructura definida en la plantilla *registerform.tpl*, que simplemente define un formulario gestionado por el archivo *register.php* y en el que se incluyen mediante una tabla los distintos campos a rellenar por el usuario, estructurada según se presenta en el algoritmo 4.55.

El estilo de los nuevos elementos que aparecen, como siempre, estará definido en el archivo *style.css*. En general, será el mismo para el resto de formularios de la página web (ver algoritmo 4.62).

El resultado puede verse en la figura 4.4.

Cuando el registro se complete, se mostrará una página con un mensaje de información, cuya estructura está definida en el archivo *registersubmitted.php* y se muestra en el algoritmo 4.55.

4.4.4. Login/logout de usuario

La estructura de la página de login, definida en *login.tpl*, será muy parecida a la del registro. En este caso, el formulario sólo tendrá los campos para usuario y contraseña, y estará gestionado por el archivo *login.php*. Se añadirá además un enlace a la opción

Algoritmo 4.54 Estilo de los formularios

```
table.maintable {
 border-style: solid;
 border-width: thin;
 border-color: #d9d9d9;
 padding: 1em;
 border-top-right-radius: 0.5em;
 border-top-left-radius: 0.5em;
 border-bottom-right-radius: 0.5em;
 border-bottom-left-radius: 0.5em;
}

td.question {
 width: 50%;
 padding-left: 0.2em;
 background-color: #eeeeee;
}

td.reply {  max-width: 40%; }

.sep {
 padding: 1%;
 font-weight: bold;
 font-style: italic;
 font-size: 120%;
 text-indent: 5%;
}

div.help {
 font-size: 80%;
 background-color: lightyellow;
}

div.error {
 color: red;
 font-weight: bold;
}
```

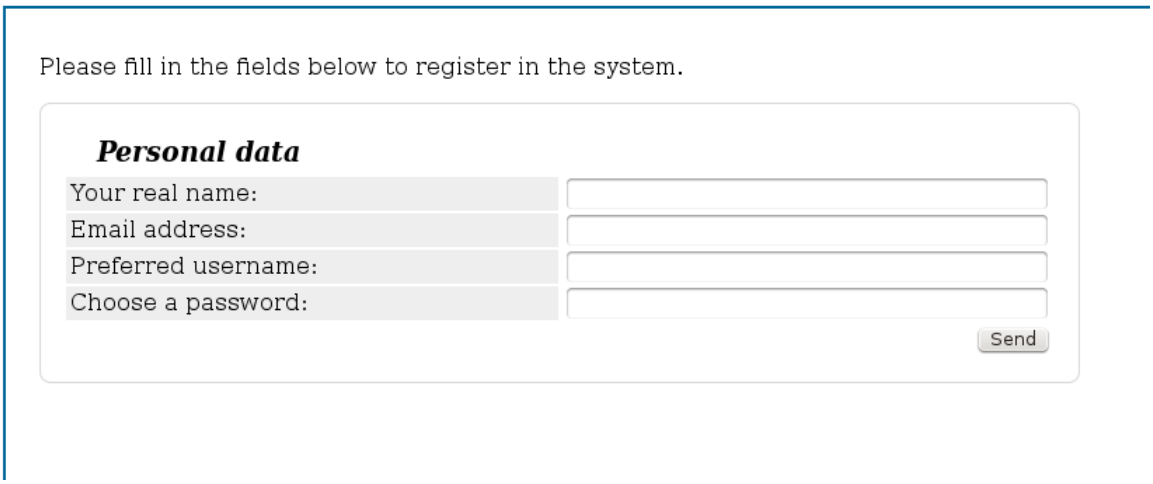

Figura 4.4: Formulario de registro

Algoritmo 4.55 Estructura de la página registro completado

```
{include file="header.tpl"}

Thank you for registering. An email message confirming your registration has been
    sent to the email address you specified.

<p> You can now <a href="login.php">login</a> and submit a proposal for technical
    sponsorship.
<p> <div style="margin-left: 20%">The IES VP for Small Conferences and Workshops</div
    >

{include file="end.tpl"}
```

Algoritmo 4.56 Estructura de la página de login

```
{include file="header.tpl"}

<p> <div class="error">{$global_error}</div>
<p>
<form method="post" action="login.php">
    <table class="maintable">
        <tr> <td class="question">User: <td class="reply"><div class="error">{
            $user_error}</div><input type="text" name="user" size=20 value="{
            $user_value}">
        <tr> <td class="question">Password: <td class="reply"><div class="error">{
            $pass_error}</div><input type="password" name="pass" size=20 value="{
            $pass_value}">
        <tr><td><td align="right"> <input type="submit" name="send" value="Send">
        </table>
</form>

<span><br> <a href="forget.php">Forgot your password?</a>
<br> Don't have username and password? <a href="register.php">Register to get them</a
    >.<br> </span>

{include file="end.tpl"}
```

de recuperar el nombre de usuario y contraseña, y otro a la página de registro (véase el algoritmo 4.56).

El estilo de los elementos que aparecen ya se ha definido en apartados anteriores. El resultado puede verse en la figura 4.5.

El logout se realiza haciendo click en la opción correspondiente del menú desplegable, tras el cual se mostrará un simple mensaje de confirmación contenido en la plantila *logout.tpl*. como se estructura en el algoritmo 4.57.

4.4.5. Recuperación de contraseña

La página de recuperación de contraseña no es más que un nuevo formulario, definido en la plantilla *forget.tpl*. tal y como se detalla en el algoritmo 4.58.

Cuando se complete la recuperación se mostrará un mensaje de confirmación contenido en la plantilla *recovered.tpl*, se omitirá por poseer una estructura similar al del

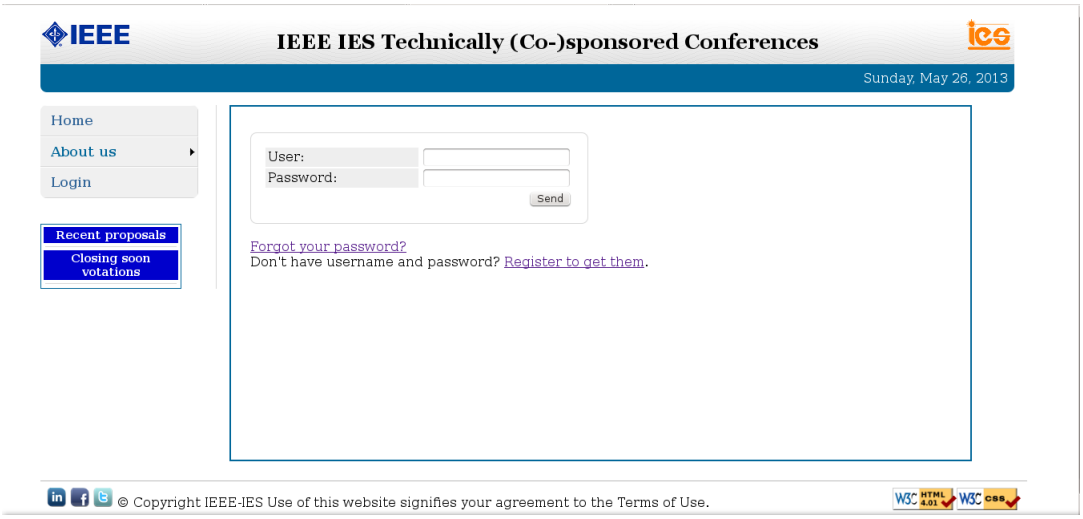

Figura 4.5: Formulario de login

Algoritmo 4.57 Estructura de la página de logout

```
{include file="header.tpl"}
<p>
{if $global_error}
    <div class="error">{$global_error}</div>
{else}
    You have been succesfully logged out.
{/if}

{include file="end.tpl"}
```

Algoritmo 4.58 Estructura de la página recuperación de contraseña

```
{include file="header.tpl"}

<p><div class="error">{$global_error}</div>
<p>Please, enter your email address to recover your password.
<p>
<form method="post" action="forget.php">
    <table class="maintable">
        <tr> <td class="question">Email: <td class="reply"><div class="error">{
            $email_error}</div><input type="text" name="email" size=20 value="{
            $password_value}">
        <tr><td><td align="right"> <input type="submit" name="send" value="Send">
    </table>
</form>

<p><span> Don't have username and password? <a href="register.php">Register to get
    them</a>.<br> </span>

{include file="end.tpl"}
```

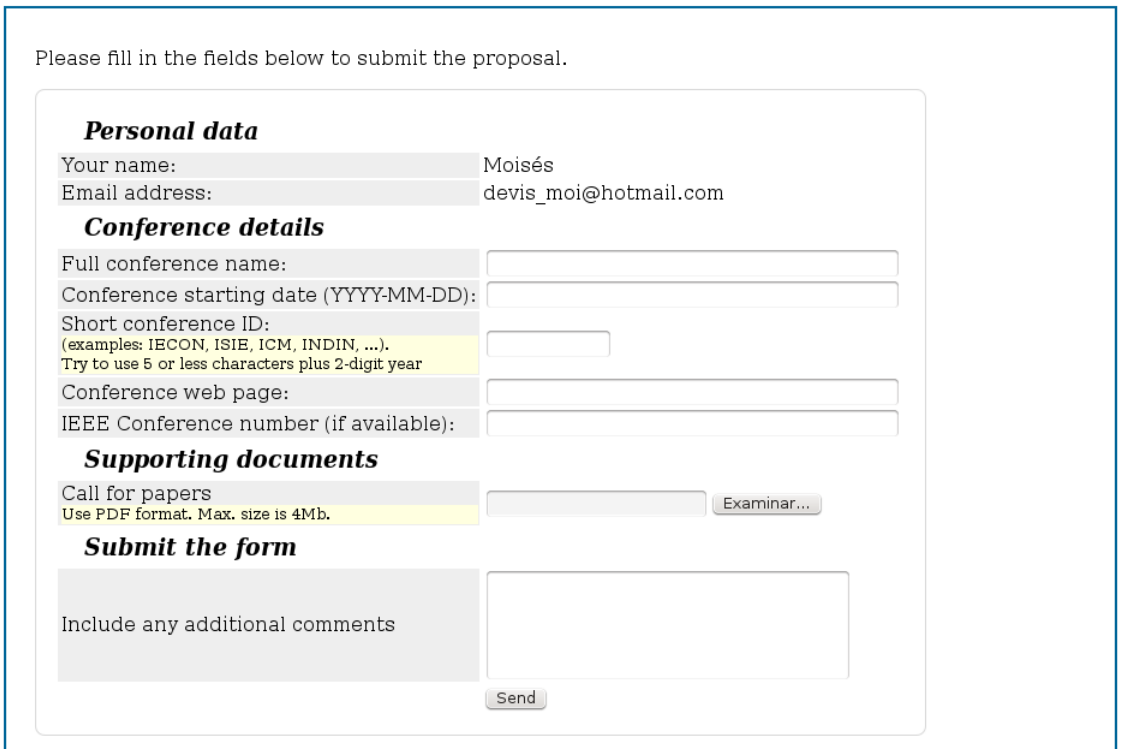

Figura 4.6: Formulario de realización de propuesta

apartado anterior.

4.4.6. Realización de propuesta

Aunque no deja de ser otro formulario, la plantilla *proposalform.tpl* incluye algunos campos nuevos, como la subida de un fichero con tamaño máximo y la posibilidad de añadir comentarios, siguiendo la estructura del algoritmo 4.59.

Además, en la plantilla *revisionform.tpl* se recoge el formulario para subir directamente una nueva revisión, conteniendo sólo los campos propios de una nueva revisión de forma idéntica al formulario anterior. Siguiendo la metodología de los procesos anteriores, una vez subida la propuesta se mostrará un mensaje de confirmación contenido en la plantilla *proposalsubmitted.tpl*.

El resultado del formulario puede verse en la figura 4.6.

4.4.7. Acciones sobre las propuestas

La apertura y cierre de votaciones no implica ningún intercambio de datos con el usuario, por lo tanto la web se limitará a mostrar un mensaje de confirmación

Algoritmo 4.59 Estructura de realización de una propuesta

```
{include file="header.tpl"}

<p> Please fill in the fields below to submit the proposal.
<div class="error">{$global_error}</div>

<p> <form enctype="multipart/form-data" method="post" action="submit.php">
    <table class="maintable">

        <tr> <td><div class="sep">Personal data</div>
        <tr> <td class="question">Your name: <td class="reply">{$name_value}
        <tr> <td class="question">Email address: <td class="reply">{$email_value}

        <tr> <td><div class="sep">Conference details</div>
        <tr> <td class="question">Full conference name: <td class="reply"><div
            class="error">{$conf_error}</div><input type="text" name="conference"
            size=40 value="{$conf_value}">
        <tr> <td class="question">Conference starting date (YYYY-MM-DD): <td class="
            reply"><div class="error">{$confdates_error}</div><input type="text" name
            ="confdates" size=40 value="{$confdates_value}">
        <tr> <td class="question">Short conference ID: <div class="help">(examples:
            IECON, ISIE, ICM, INDIN, ...).<br> Try to use 5 or less characters plus
            2-digit year</div> <td class="reply"><div class="error">{$confid_error}</
            div><input type="text" name="confid" size=10 value="{$confid_value}">
        <tr> <td class="question">Conference web page: <td class="reply"> <div class=
            "error">{$confurl_error}</div><input type="text" name="confurl" size=40
            value="{$confurl_value}">
        <tr> <td class="question">IEEE Conference number (if available): <td class="
            reply"> <div class="error">{$confnumber_error}</div><input type="text"
            name="confnumber" size=40 value="{$confnumber_value}">

        <tr> <td><div class="sep">Supporting documents</div>
        <tr> <td class="question">Call for papers <div class="help">Use PDF format.
            Max. size is 4Mb.</div> <input type="hidden" name="MAX_FILE_SIZE" value="
            4000000"> <td class="reply"><div class="error">{$cfp_error}</div><input
            type="file" name="cfp" value="{$cfp_value}">

        <tr> <td><div class="sep">Submit the form</div>
        <tr> <td class="question">Include any additional comments <td class="reply"><
            div class="error">{$comments_error}</div><textarea name="comments" rows="
            5" cols="40">{$comments_value}</textarea>
        <tr><td><td> <input type="submit" name="send" value="Send">
    </table>
</form>

{include file="end.tpl"}
```

Algoritmo 4.60 Estructura de decisión de una propuesta

```
{include file="header.tpl"}

{if $error} <div class="error">{$msg}</div>

{else}
<div class="title">Decide on proposal for {$conftitle}</div>
{include file="supportingdocs.tpl"}

<form action="decide.php" method="post">
    <table>
        {section name=i loop=$decision_types}
        <tr> <td><input type="radio" name="decision" value="{$decision_types[i].id}">
            {$decision_types[i].content}<br>
        {/section}
        <tr> <td> Comments:<br> <textarea rows="10" cols="40" name="comments"> </
            textarea> <input type="hidden" name="id" value="{$id}"><br> <input type="
            submit" name="submit" value="Submit decision">
    </table>
</form>

{/if}

{include file="end.tpl"}
```

pasado como parámetro Smarty a la plantilla *openclose.tpl* cuando se realice la acción correspondiente.

Para decidir una propuesta, se mostrará un formulario con los distintos tipos de opciones de decisión (pasados como parámetro desde el código PHP y mostrados mediante una sentencia de control de Smarty) entre las que el usuario deberá elegir una, además de un campo destinado a posibles comentarios (ver algoritmo 4.60).

Cuando se reciba la decisión, se mostrará el mensaje de confirmación estructurado en la plantilla *decidecast.tpl*.

La votación de una propuesta seguirá una estructura indéntica a la de la decisión. En la plantilla *vote.tpl* se estructura la presentación del formulario con los tipos de votos, tal y como se ha hecho con las decisiones, y cuando sea recibido se mostrará el mensaje de confirmación con la plantilla *votecast.tpl*.

En la figura 4.7 puede verse el formulario de votación.

4.4.8. Visionado de las propuestas

La lista general de propuestas se mostrará en una tabla con tres columnas que contendrán el nombre de la propuesta, el estado y botones de acción según los permisos del usuario. En la última fila se añadirá un indicador con la página que se está visionando y selectores para avanzar o retroceder. Dicha estructura está recogida en el algoritmo 4.63.

Algoritmo 4.61 Estructura del visionado de propuestas

```
{include file="header.tpl"}

<table class="information">
    <tr><th>Conference proposal</th><th>Status</th><th class="title">Actions</th>

    {section name=i loop=$proposals}
    <tr>
    {if $proposals[i].canviewsubm}
    <td><a href="proposal.php?id={$proposals[i].id}">    {$proposals[i].confid}, {
        $proposals[i].name}</a>, {$proposals[i].date}

    {if $proposals[i].canviewstatus}
    <td>    {$proposals[i].status}
    {/if}
    {/if}

    <td>
    {if $proposals[i].canvote}
    <div class="vote"><a href="vote.php?id={$proposals[i].votationid}">VOTE</a></div>
    {/if}

    {if $proposals[i].canopen}
    <div class="open"><a href="open.php?id={$proposals[i].id}">OPEN FOR VOTING</a></
        div>
    {/if}

    {if $proposals[i].canclose}
    <div class="close"><a href="close.php?id={$proposals[i].votationid}">CLOSE VOTING
        </a></div>
    {/if}

    {if $proposals[i].candecide}
    <div class="decide"><a href="decide.php?id={$proposals[i].id}">DECIDE</a></div>
    {/if}

   {if $proposals[i].cansubmit}
    <div class="decide"><a href="submitrevision.php?id={$proposals[i].id}">SUBMIT
        REVISION</a></div>
    {/if}
    {/section}

   <tr>    <tr><td>
    <td style="background-color: #CCFFFF; text-align:center;">Page
    <td>
    {if isset($prevpage)}    <a href="proposals.php?page={$prevpage}">&#60;</a>    {/
        if}    {$page}/{$pages}
    {if isset($nextpage)}    <a href="proposals.php?page={$nextpage}">></a>    {/if}
</table>

{include file="end.tpl"}
```

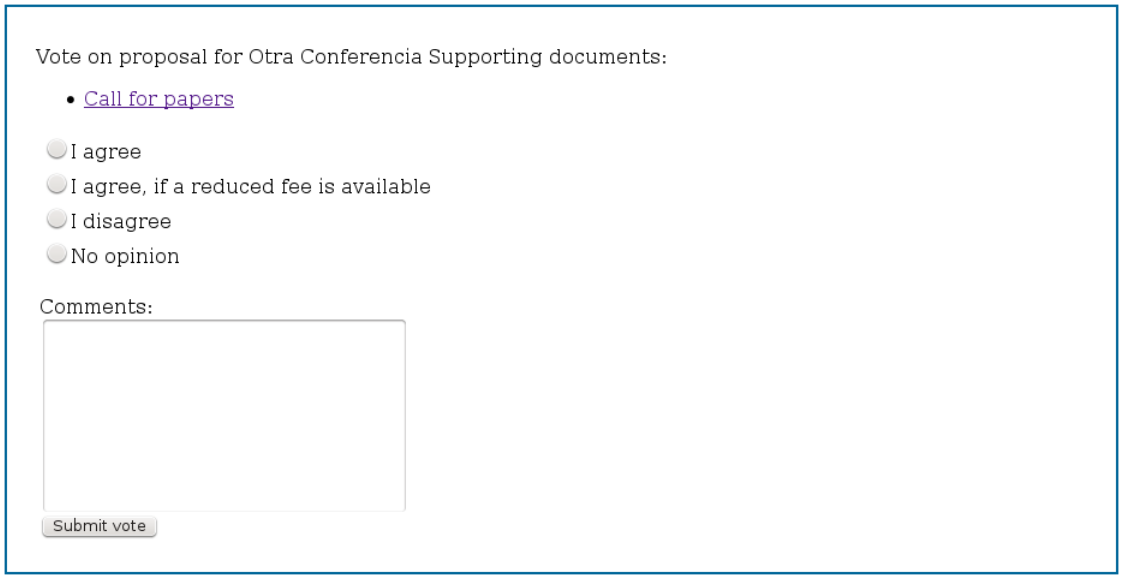

Figura 4.7: Formulario de votación

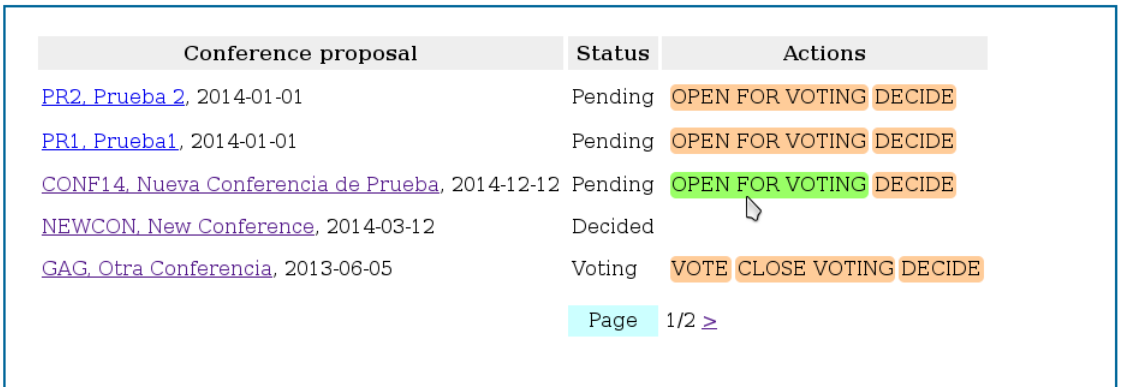

Figura 4.8: Tabla de visionado de propuestas

El estilo de los botones de acción queda definido en el archivo *style.css*, aprovechando la opción *hover* de CSS para cambiar el color del botón cuando se pase el cursor por encima, como puede apreciarse en el algoritmo 4.62.

Un ejemplo de cómo sería la tabla resultante puede verse en la figura 4.8.

4.4.9. Visionado detallado de una propuesta

La información detallada de una propuesta estará estructurada en la plantilla *proposal.tpl*. En primer lugar, se mostrará una tabla con los atributos fijos de una propuesta, como se comienza en la estructura del algoritmo 4.63.

Seguidamente se mostrará una fila en la que se indica la versión de la propuesta

Algoritmo 4.62 Estilo de los botones de acción

```
table.information td div {
 float:left;
 margin-left: 0.2em;
 padding: 0.1em;
 background-color: #ffcc99;
 border-top-right-radius: 0.3em;
 border-top-left-radius: 0.3em;
 border-bottom-right-radius: 0.3em;
 border-bottom-left-radius: 0.3em;
}

table.information td div a:link {color: #000; text-decoration:none;}
table.information td div a:visited {color: #000; text-decoration:none;}
table.information td div a:hover {color: #000; text-decoration:none;}

div.open:hover {background-color: #99ff66;}
div.vote:hover {background-color: #99ff66;}
div.decide:hover {background-color: #66ffff;}
div.close:hover {background-color: #ff3333;}
```

Algoritmo 4.63 Estructura del visionado detallado de una propuesta I

```
{include file="header.tpl"}

{if $canviewsubm}
<table>
    <tr> <td class="data">Conference full name: <td class="info">{$fullname}
    <tr> <td class="data">Abbreviation: <td class="info">{$conf}
    <tr> <td class="data">Start date: <td class="info">{$date_start}
    <tr> <td class="data">Web page: <td class="info"><a href="{$url}" target="_blank"
        >{$url}</a>    <tr> <td class="data">Submitted by: <td class="info">{
        $proposer} (<a href="mailto:{$proposer_email}">{$proposer_email}</a>)
</table>

<p>
```

Algoritmo 4.64 Estructura del visionado detallado de una propuesta II

```
<table class="maintable" style="padding-top:0em;">
    <tr> <td>
    <table>
        <tr>
            <td class="data"> {if isset($prev_version)} <a href="proposal.php?id={$id
                }&version={$prev_version}"> {/if} &#60;&#60;View previous version {if
                 isset($prev_version)} </a> {/if}
            <td style="background-color: #CCFFFF; width:10em; text-align:center; font
                -weight:bold;">Revision {$version}
            <td class="data"> {if isset($next_version)} <a href="proposal.php?id={$id
                }&version={$next_version}"> {/if} View next version>> {if isset(
                $next_version)} </a> {/if}
    </table>

    <tr><td> <h3>Information</h3> Submitted on {$date_subm} <p> Comments: {$comments}
         <p> {include file="supportingdocs.tpl"}
```

(por defecto se muestra la última), y dos enlaces para avanzar o retroceder hacia otras revisiones. La tabla principal contendrá la información detallada de la versión que se esté mostrando (ver algoritmo 4.64).

Por último, se mostrará el estado de la revisión (con enlaces para cambiarlo si se tienen los permisos adecuados), dos tablas con los votos detallados de cada miembro y el recuento total, y el enlace para votar, si la propuestá está abierta a votación y el usuario tiene los permisos correspondientes, concluyendo así la estructura según el algoritmo 4.65.

El resultado final puede verse en la figura 4.9.

4.4.10. Resumen: listado y función de plantillas HTML

En la tabla 4.3 se recoge un listado con el nombre y la función de cada una de las plantillas HTML que estructuran la página web.

Algoritmo 4.65 Estructura del visionado detallado de una propuesta III

```
    {if $canviewstatus}
    <tr><td> <h3>Status</h3> Status of the proposal: {$status}
    {if $canopen}(<a href="open.php?id={$id}">Open for voting</a>){/if}
    {if $canclose}(<a href="close.php?id={$id}">Close voting</a>){/if}
    {/if}

    {if $canviewvotes}
    <tr><td> <h3>Votes</h3>
        <table class="subtable">
            <tr><th>Person<th>Vote<th>Comments {section name=i loop=$v}
            <tr> <td> {$v[i].person}  <td> {$v[i].vote} <td> {$v[i].comments} {/
                section} </table>
            <tr><td> <h3>Vote count</h3> <table class="subtable">
            <tr> {foreach from=$namevotes item=j} <th>{$j} {/foreach}
            <tr> {foreach from=$totalvotes item=j} <td>{$j} {/foreach}
        </table>
    {/if}

    {if $canvote}
    <tr><td> <h3>Voting</h3> <a href="vote.php?id={$id}">Vote this proposal</a> {/if}

</table>
{/if}

{include file="end.tpl"}
```

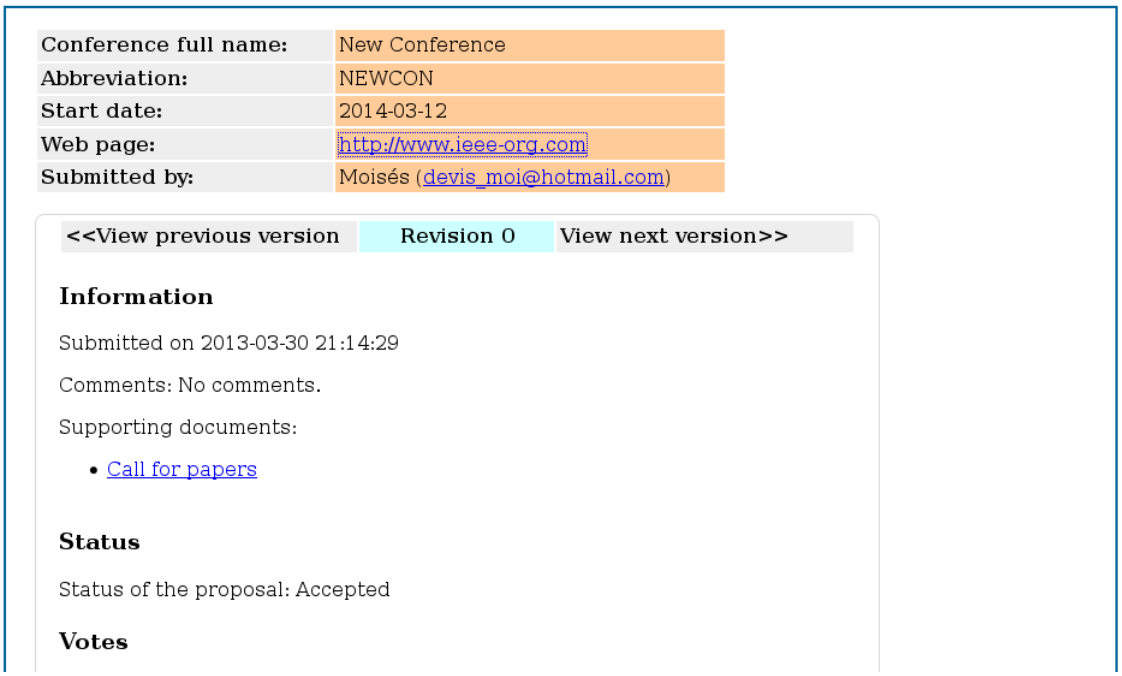

Figura 4.9: Visionado detallado de una propuesta

Nombre del archivo	Función
header.tpl	Estructura principal de la web (cabecera y menú)
end.tpl	Estructura del pie de página de la web
index.tpl	Estructura de la página principal
registerform.tpl	Estructura del formulario de registro
registersubmitted.tpl	Mensaje de confirmación de registro
forget.tpl	Formulario de recuperación de contraseña
recovered.tpl	Confirmación de recuperación de contraseña
login.tpl	Estructura del formulario de login
logout.tpl	Mensaje de confirmación de logout
proposalform.tpl	Formulario de realización de propuesta
revisionform.tpl	Formulario de nueva revisión de una propuesta
proposalsubmitted.tpl	Confirmación de propuesta realizada
openclose.tpl	Confirmación de apertura/cierre de votación
decide.tpl	Formulario de decisión de propuesta
decidecast.tpl	Mensaje de confirmación de decisión
vote.tpl	Formulario de votación
votecast.tpl	Mensaje de confirmación de votación
proposals.tpl	Estructura de la tabla de visionado de propuestas
proposal.tpl	Estructura del visionado detallado de una propuesta
supportingdocs.tpl	Enlace al PDF adjunto de la propuesta

Cuadro 4.3: Listado de plantillas HTML

4.5. Plantillas de notificación por correo

Para las notificaciones por correo electrónico se usarán plantillas genéricas en texto plano con algunos parámetros variables para adaptarlas a cada caso concreto. Por ejemplo, a continuación se muestra la plantilla de notificación al usuario cuando ha realizado una propuesta:

```
Dear {$name},

Your proposal of Technical (Co-)Sponsorship for the conference {$confid} has been
    correctly received and stored. It will now be handled by the IEEE IES
    Vicepresident for Workshops and Small Conferences and the IES Conferences Board.

During the process, it is possible that you will be requested to provide additional
    information.

When a decision is made, you will be informed. It is possible to see the status of
    the proposal at any time by going to

http://ieee-ies.org/conferences/login.php

and selecting "View proposals" in the left-hand menu.

Thank you for your interest in IEEE IES Conference activities!

Sincerely,

The IES VP for Small Conferences and Workshops
```

El resto de plantillas pueden encontrarse en el directorio *msg* del CD. La siguiente tabla recoge la lista de plantillas junto con la función de cada una:

Nombre del archivo	Uso
register.txt	Confirmación de registro
forget.txt	Recuperación de usuario y contraseña
submit.txt	Confirmación de propuesta realizada al usuario
submit-vpw.txt	Notificación de propuesta realizada al administrador
decided.txt	Confirmación de propuesta decidida
voted.txt	Confirmación de voto

Cuadro 4.4: Lista de plantillas de notificación

Capítulo 5

Implementación

El presente capítulo tiene por objetivo detallar la instalación y configuración de las tecnologías adoptadas a la hora de acometer la puesta en marcha del sistema web. Aunque dichas tecnologías son multiplataforma, por seguridad se aconseja su implementación sobre sistemas operativos tipo Unix o derivados, como GNU/Linux o FreeBSD. De ahora en adelante se supondrá que todo el proceso de instalación y configuración se realiza sobre GNU/Linux, el más utilizado en estos casos, lo que no supondrá pérdida de generalidad para la implementación en otros sistemas.

Ante cualquier duda, siempre se recomienda seguir el manual de instalación y configuración oficial de cada tecnología.

5.1. Apache y PHP

La mayoría de las distribuciones ya cuentan con paquetes precompilados para la instalación del servidor web Apache y el módulo PHP. Se recomienda esta opción con el fin de facilitar la administración y la actualización. En cualquier caso, tanto Apache como PHP pueden descargarse de la web oficial de los respectivos proyectos e instalarse siguiendo las instrucciones. Además, en el CD se incluye el código fuente de la versión usada para el actual proyecto, que incluye sus correspondientes instrucciones de instalación.

Una vez instalados, será necesario configurar Apache para que cargue el módulo PHP y procese ficheros con dicha extensión. Para ello no habrá más que añadir las siguientes líneas en el fichero de configuración *httpd.conf*, que normalmente se encuentra en algún subdirectorio de /etc:

```
LoadModule php5_module modules/libphp5.so

<FilesMatch \.php$>
    SetHandler application/x-httpd-php
</FilesMatch>
```

Para arrancar el servidor, simplemente habrá que iniciar el demonio *httpd* ejecutando el siguiente comando como superusuario:

```
#> /etc/init.d/httpd start
```

5.2. Sendmail

Para enviar los correos electrónicos, PHP usa la función del sistema sendmail. La mayoría de las distribuciones GNU/Linux ya vienen con la librería preinstalada o cuentan con paquetes precompilados para su instalación. Para ininiciar el servicio y que puedan enviarse las notificaciones por correo electrónico, tan sólo habrá que ejecutar el siguiente comando como usuario root:

```
#> /etc/init.d/sendmail start
```

5.3. PhpMyAdmin (Opcional)

PhpMyAdmin es una herramienta escrita en PHP creada para manejar la administración de MySQL desde un navegador web. Aunque no es necesaria para el funcionamiento del sistema web del presente proyecto, se recomienda su instalación con el fin de facilitar la administración de la base de datos.

Puede descargarse e instalarse siguiendo las instrucciones desde la página web de phpMyAdmin, o usando el código fuente incluído en el CD adjunto, pero de nuevo se recomienda la instalación mediante el sistema de gestión de paquetes propio del sistema operativo. Una vez instalado, basta acceder a la dirección *http://localhost/phpmyadmin* desde un navegador web corriendo en el propio servidor para comenzar a usarlo.

5.4. MySQL

Como en los casos anteriores, la mayoría de las distribuciones GNU/Linux disponen de paquetes precompilados para la instalación de MySQL, facilitando así la administración y actualización del programa. En cualquier caso, tanto de la web oficial

como del CD adjunto puede obtenerse el código fuente e instalarse siguiendo las instrucciones. Una vez completados los pasos de una u otra forma, se iniciará el demonio de MySQL tal desde la línea de comandos tal y como se hizo con Apache:

```
#> /etc/init.d/mysql start
```

El primer paso, muy importante, será definir una contraseña para el usuario root de MySQL mediante los siguientes comandos:

```
#> mysql -u root
mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('password');
```

A continuación puede comenzarse la implementación de la base de datos. Aunque también se explicará cómo hacerlo manualmente a través de la línea de comandos de MySQL, se recomienda usar la herramienta phpMyAdmin para facilitar las tareas de creación y administración de la base de datos.

5.4.1. Creación de la base de datos mediante phpMyAdmin

Tras acceder a la interfaz gráfica de phpMyAdmin, como se indica en el apartado anterior, habrá que hacer login como usuario root de MySQL y seguir los siguientes pasos:

1. Hacer click en la pestaña *Bases de datos* del menú superior. Introducir el nombre de la base de datos, *tech_conf*, en el formulario y pulsar *Crear*.

2. Seleccionar la base de datos creada en la lista de la izquierda.

3. Hacer click en la pestaña *Importar* del menú superior. Pulsar *Seleccionar archivo* y abrir el archivo *tech_conf.sql* contenido en el CD. Hacer click en *Continuar* para ejecutar las sentencias.

4. Hacer click en la pestaña *Privilegios* del menú superior. Rellenar los campos usuario (*tech_conf*) y contraseña del formulario, y seleccinar servidor local. Marcar los privilegios SELECT, INSERT y UPDATE. Pulsar *Continuar* para crear el usuario.

5.4.2. Creación de la base de datos manualmente

Para crear la base de datos, se establecerá conexión con el servidor MySQL como usuario root:

```
#> mysql -h localhost -u root -p
Enter password: ******
```

Una vez establecida la conexión, se creará y usará la base de datos con los siguientes comandos MySQL:

```
mysql> CREATE DATABASE tech_conf;
mysql> USE tech_conf;
```

Ahora ya pueden definirse las tablas una a una con los comandos especificados en el apartado de diseño de la base de datos, o directamente importando el archivo *tech_ conf.sql* contenido en el CD con el siguiente comando:

```
mysql> source /PATH/tech_conf.sql;
```

Siendo *PATH* la ruta donde se encuentra el archivo.

Finalmente, se creará el usuario local usado por el sistema web para acceder a la base de datos (especificado en el fichero *config.php*) y se le concederán los privilegios necesarios con los siguientes comandos:

```
mysql> CREATE USER 'tech_conf'@'localhost' IDENTIFIED BY 'password';
mysql> GRANT SELECT, INSERT, UPDATE ON tech_conf TO 'tech_conf'@'localhost';
```

5.5. Smarty

La instalación de Smarty es muy simple, tan sólo es necesario copiar las librerías que se pueden descargar en la web del proyecto a algún directorio del sistema, ya sea manualmente o mediante un paquete precompilado propio de la distribución en uso. Para que funcione, habrá que incluir la ruta del archivo *Smarty.class.php* tal como se indica en el apartado de diseño del contenido dinámico. Además, Smarty necesita que existan los directorios *templates_ c/*, *configs/*, *templates/* y *cache/*, con permiso de escritura en los dos últimos.

5.6. Creación de directorios y copia de ficheros

El último paso para la implementación del sistema web será crear los directorios necesarios y situar los ficheros en su lugar correspondiente. Por defecto, Apache sirve los directorios localizados en */var/www*, por lo que se usará dicha ruta en la implementación del sistema. Para crear los directorios, se ejecutarán los siguientes comandos como usuario root:

```
#> cd /var/www
#> mkdir -p /conferences/tech
#> cd /conferences/tech
#> mkdir attach cache configs templates_c
```

Ahora es necesario conceder a Apache permisos de escritura en los directorios requeridos a tal efecto por Smarty. Suponiendo que se ejecuta como usuario y grupo *apache*, se haría con los siguientes comandos:

```
#> chown apache:apache templates_c/ cache/
#> chmod 770 templates_c/ cache/
```

Finalmente, habrá que copiar los archivos php, las plantillas Smarty y el resto de archivos (directorio de imágenes y directorio de plantillas de notificación) al directorio del sistema web. Para ello habrá que situarse en el directorio donde se haya montado el contenido del CD y ejecutar los siguientes comandos:

```
#> cd Sistema/
#> cp *.php *.css /var/www/conferences/tech
#> cp -r images/ /var/www/conferences/tech
#> cp -r msg/ /var/www/conferences/tech
#> cp -r templates/ /var/www/conferences/
```

Si todo se ha realizado correctamente, la página web será accesible desde la ruta *http://IPServidor/conferences/tech*.

Capítulo 6

Seguridad

Dado que el sistema web estará expuesto a cualquier usuario con acceso a Internet, será extremadamente importante tomar medidas de seguridad que eviten dentro de lo posible que usuarios maliciosos accedan a información confidencial o alteneren malintencionadamente el proceso de funcionamiento del sistema.

En este capítulo se tomarán algunas consideraciones de seguridad atentiendo a la naturaleza de los posibles ataques.

6.1. Ataques de fuerza bruta

Los ataques de fuerza bruta intentan aprovechar que todos los recursos utilizados para dar servicio (ancho de banda, memoria...) son limitados, saturándolos con el fin de denegar el servicio a otros usuarios o forzar una vulnerabilidad del sistema.

En general, dichos ataques son imposibles de evitar si el usuario cuenta con los recursos suficiente, siendo la detección del ataque y la limitación del servicio (ya sea a nivel de IP mediante Firewall o mediante permisos del sistema) las únicas acciones posibles.

6.1.1. Registro IP

Con el fin de facilitar la detección de un ataque al sistema, se creará una tabla en la base de datos que contenga un registro de las acciones llevadas a cabo por los usuarios en la web, almacenando su IP y la marca de tiempo. La implementación en MySQL se llevará a cabo con la siguiente sentencia:

Algoritmo 6.1 Implementación de la tabla IP_registry

```
CREATE TABLE IF NOT EXISTS `IP_registry`(
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `user` int(11) NOT NULL,
    `action` varchar(20) NOT NULL,
    `IP` varchar(20) NOT NULL,
    PRIMARY KEY (`id`),
);
```

De esta forma, en cada acción del usuario bastará con insertar el siguiente código PHP para que quede registrada en la base de datos:

Algoritmo 6.2 Registro de la IP/acción del usuario en la base de datos

```
$sender_ip=$_SERVER['REMOTE_ADDR'];
$idx=db_query("insert into IP_registry (IP,user,action) values ('$sender_ip','$iduser
    ','Register');");
```

Sustituyendo *Register* por la acción que corresponda en cada caso.

6.1.2. Modificación de permisos

Cuando el ataque se salta el bloqueo por IP, no queda más remedio que limitar temporalmente los servicios ofrecidos. La modularidad del diseño del sistema web permite, mediante la modificación de permisos, jugar con estas limitaciones para que la web siga ofreciendo unos servicios mínimos.

Por ejemplo, en el caso de que un usuario malicioso se dedique a realizar numerosas propuestas en un corto período de tiempo con el único fin de llenar de información basura la base de datos, el administrador podría limitar temporalmente la opción de realizar propuestas. Gracias a los permisos binarios en forma de atributos en la tabla USER_TYPES, se haría de forma muy sencilla desde phpMyAdmin o manualmente con la siguiente sentencia MySQL:

```
mysql> UPDATE user_types SET CANSUBMIT=0 WHERE user_types.role=0;
```

O en otro caso, si se están produciendo registros de usuarios sospechosos, directamente puede deshabilitarse momentáneamente la página de registro sin más que quitar los permisos de lectura del archivo *register.php* ejecutando el siguiente comando como usuario root:

```
#> chmod -r /var/www/conferences/tech/register.php
```

6.2. Ataques lógicos

Este tipo de ataques intentan aprovechar vulnerabilidades en el diseño del sistema o de algunas de las tecnologías que lo implementan. Aunque dichas vulnerabilidades suelen ser fáciles de solucionar, su detección es mucho más complicada, y en algunos casos la solución puede depender de terceros.

6.2.1. Inyección de código

La inyección de código se produce cuando un usuario malicioso utiliza una de las variables de entrada al sistema web para ejecutar código que realice una acción no deseada. Por ejemplo, introduciendo una sentencia SQL en una de las variables para modificar a su antojo la base de datos, o código javascript en un comentario para que se ejecute cuando se muestre al resto de usuarios.

En general, la regla para evitar este tipo de ataques es no confiar nunca en las variables recibidas por el usuario, forzando siempre que sea posible la conversión de tipos y limpiando las cadenas de caracteres especiales.

En el caso del presente proyecto, sólo se reciben enteros y cadenas de caracteres por parte del usuario. Para el primer caso, se fuerza siempre la conversión de tipos con la función *intval* de PHP, y para el segundo se usa la función *escape_string* definida y explicada en el apartado de diseño del contenido dinámico.

6.2.2. Usuarios y permisos

Los permisos de los usuarios compenen una parte fundamental de la seguridad del sistema, tanto los relativos al diseño propio como los relativos a las tecnologías de terceros que implementan la página web. Un sistema de permisos bien diseñado permite reducir los daños producidos por las vulnerabilidades.

El sistema de permisos propio del sistema web ya se ha explicado detenidamente en los apartados de diseño de la base de datos y contenido dinámico. Si en el futuro se implementan nuevas acciones, será importante seguir el mismo esquema de comprobación para no comprometer la seguridad del sistema.

Respecto al usuario usado para conectarse a la base de datos, es aconsejable otorgarle únicamente los permisos de acciones que van a usarse en las sentencias SQL, tal y como se muestra en el apartado de implementación de MySQL. De esta forma se minimizan los daños si un usuario malicioso consigue inyectar sentencias SQL. En ningún caso debe usarse el usuario root de MySQL para la conexión a la base de datos.

Por último, también es importante que los usuarios que ejecutan los demonios del servidor Apache, MySQL y cualquier otro proceso implicado en el sistema web, tengan permisos limitados en el sistema operativo, de forma que de nuevo se minimicen los daños en caso de vulnerabilidades en dichas tecnologías. Por defecto, suelen ejecutarse en usuarios ya predefinidos como *apache* o *mysql*, por lo que se aconseja no modificar dicha configuración.

6.2.3. Robo de sesión

Cuando un usuario accede a la página web y se inicia una sesión, automáticamente PHP asigna un identificador aleatorio y temporal de sesión. Si otro usuario malicioso es capaz de averiguar dicho identificador, durante un período de tiempo podrá acceder a la página web usando dicha sesión, y por tanto interactuar con el sistema con los parámetros asociados a la misma (usuario, contraseña, permisos...).

A continuación se enumeran algunas recomendaciones para dificultar este ataque:

- Usar únicamente cookies para pasar el identificador de sesión, poniendo a uno la opción *session.use_only_cookies* de PHP. Desde la versión 5.3.0, está activada por defecto.
- Regenerar el identificador de sesión en cada nueva sesión, usando una marca en la variable global *$_SESSION* y la función de PHP *session_regenerate_id* (véase el archivo *congif.php* o el algoritmo 4.16).
- Regenerar el identificador de sesión siempre que el usuario cambie de estado en el sistema (véase el archivo *login.php* o el algoritmo 4.25).
- Activar HTTPS en Apache, para que toda la información HTTP intercambiada entre el usuario y el servidor viaje cifrada por la red.

6.2.4. Tecnologías de terceros

Las vulnerabilidades que afectan a tecnologías de terceros son las más difíciles de solucionar, ya que dependen de equipos de desarrollo externos. En general, se aconseja seguir las siguientes recomendaciones:

- Actualizar de forma periódica todas las tecnologías implicadas en el sistema.
- Elegir en lo posible tecnologías ampliamente utilizadas y con un equipo desarrollo activo detrás.

- Considerar, siempre que se ajusten a las necesidades del proyecto, soluciones de software libre. Cuando el código está disponible para el cliente, las vulnerabilidades son detectadas más rápidamente.
- Minimizar, en la medida de lo posible, el número y la complejidad de las tecnologías a utilizar. Un gran número de funcionalidades sin uso sólo aporta potenciales vulnerabilidades del sistema.

Capítulo 7

Conclusiones

7.1. Resultados

El presente proyecto ha dado como resultado un sistema que resuelve la necesidad de una gestión, adaptada a las nuevas tecnologías web, del proceso de propuestas y selección de conferencias copatrocinadas por el IEEE-IES.

La generalidad y la modularidad del diseño permiten, primero, que el sistema web sea fácilmente adaptable a cualquier proceso de propuestas y toma de decisiones basada en votaciones. Segundo, escalabilidad ante la necesidad de nuevas funcionalidades futuras, nuevos tipos de usuario, permisos y votos, o cambios en el formato de las propuestas. Y tercero, libertad total para modificar la estructura de la web conforme a las nuevas tecnologías y corrientes de estilo marcadas en un futuro, gracias a la separación del código dinámico del etiquetado de presentación.

Por último, la elección de tecnologías libres, ampliamente utilizadas y amparadas por grandes equipos de desarrollo, garantizan a medio plazo soporte y actualizaciones de los módulos que componen el sitema, y suponen que el coste del mismo se reduzca a la electrónica y los recursos humanos.

7.2. Líneas de trabajo futuras

Las líneas de trabajo futuras se presentarán mayoritariamente durante el desarrollo de la vida del sistema web. Los posibles problemas que surjan y la realimentación de los usuarios tras su experiencias con el sistema, determinarán sin duda la necesidad de implementar nuevas funcionalidades, así como de adaptar el diseño y la estructura de la página a futuros estándares en la presentación de la información, o simplemente hacerla más atractiva conforme a las corrientes de estilo aceptadas por los usuarios.

Respecto a nuevas funcionalidades, a continuación se presenta una lista de algunas que probablemente sea interesante implementar en un futuro:

- Interfaz web de administración: algunas de las tareas de administración, como el cambio de tipo de usuario o la definición de nuevos roles, se realizan modificando manualmente la base de datos o a través de herramientas como phpMyAdmin. Si el número de usuarios crece, podría ser últil añadir al sistema la posibilidad de realizar estas tareas directamente desde la página web haciendo login como administrador.

- Mejorar la interactividad de los usuarios: hay múltiples opciones que permitirían mejorar la interactividad del usuario en la página web, como la inclusión de un sistema de comentarios y respuestas para las propuestas o de un calendario de eventos en la página principal. Dichas necesidades se irán vislumbrando con la experiencia de uso de los propios usuarios.

- Sistema de avisos/noticias: la tabla del menú de navegación lateral permitiría insertar fácilmente en la página principal la visualización de avisos o noticias por parte del administrador (insertados por ejemplo desde la interfaz de administración que se comenta en el primer punto), en caso de que se hiciera necesario en un futuro.

- Historial/estadísticas de uso: para la organización puede resultar útil mantener un registro con las principales estadísticas de uso de la web, como los países desde los que se realizan las propuestas o se visita la página.

Respecto al diseño de la página, la tendencia actual de los usuarios a navegar por Internet cada vez más desde dispositivos móviles abre nuevas necesidades y posibilidades para el sistema web. Aunque la página actual es perfectamente visualizable y funcional desde un navegador web instalado en el móvil, el tamaño de las pantallas o las funcionalidades táctiles de las mismas pueden ser razones para acometer un nuevo proyecto:

- Crear un nuevo diseño de la página web que se adapte a la resolución de los terminales móviles. Es común encontrar páginas webs que disponen de una versión móvil, y CSS ya permite la creación de una hoja de estilo específica para lo mismo, de forma que la página se mostrará con un diseño u otro según el usuario navegue desde el móvil o un ordenador personal.

- Crear una aplicación específica para un sistema operativo móvil, como por ejemplo Android, que realizando las peticiones HTTP pertinentes al sistema web permita al usuario disfrutar de las funcionalidades sin necesidad de abrir un navegador.

Bibliografía

[1] Mehdi Achour, Friedhelm Betz, Antony Dovgal, Nuno Lopes, Hannes Magnusson, Georg Richter, Damien Seguy, and Jakub Vrana. *PHP Manual*, 2013.

[2] Bert Bos, Tantek Çelik, Ian Hickson, and Håkon Wium Lie. Cascading style sheets level 2 revision 1 (css 2.1) specification. World Wide Web Consortium, Candidate Recommendation CR-CSS21-20070719, July 2007.

[3] MySQL Documentation. *http://dev.mysql.com/doc*.

[4] Smarty Documentation. *http://www.smarty.net/docs/en*.

[5] Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems, Fourth Edition*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[6] H. Nielsen, P. Leach, and S. Lawrence. An HTTP Extension Framework. RFC 2774 (Experimental), February 2000.

[7] Dave Raggett, Arnaud Le Hors, and Ian Jacobs. Html 4.01 specification. W3C Recommendation, December 1999.

[8] T. Suzumura, S. Trent, M. Tatsubori, A. Tozawa, and T. Onodera. Performance comparison of web service engines in php, java and c. In *Web Services, 2008. ICWS '08. IEEE International Conference on*, pages 385–392, 2008.