Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Páginas e-amarillas

El ebook como revitalizador de viejos autores

Proyecto Final de Carrera

Licenciatura de Documentación

Autor: Miguel Ángel Ferrer Forment

Director: Juan Vicente Oltra Gutiérrez

Julio de 2013

Resumen

Proyecto en el que se diseña e implementa una aplicación web en PHP que permite almacenar y gestionar contenidos digitalizados en formato imagen procedentes de viejas revistas y/o libros. Esta aplicación permite crear de forma fácil y rápida para el usuario un producto final multiformato: epub, pdf, mobi, cbz y html. Este producto final podrá ser visualizado en cualquier dispositivo digital de lectura actual, como un ordenador personal, un lector electrónico de libros, una tableta digital o un móvil de última generación ayudando a preservar el documento original.

Palabras clave: aplicación web, contenidos digitales, imágenes, preservación de documentos, php, epub, html, cbz, pdf, mobi

Tabla de contenidos

1. Objeto y objetivos

El objeto del presente Proyecto de Fin de Carrera es la obtención del título de segundo ciclo de Licenciado en Documentación impartido en la Escuela Técnica Superior de Ingeniería Informática y expedido por la Universidad Politécnica de Valencia.

El objetivo principal de este Proyecto Fin de Carrera es el de crear una aplicación que permita almacenar y gestionar contenidos digitalizados en formato imagen procedentes de viejas revistas y/o libros, de forma que permita crear de forma fácil para el usuario un producto final multiformato, como por ejemplo, epub, pdf, mobi, cbz, etc. Este producto final podrá ser visualizado en cualquier dispositivo digital de lectura actual, como un ordenador personal, un lector electrónico de libros, una tableta digital o un móvil de última generación.

2. Introducción

Ya en pleno siglo XXI Internet se ha convertido en una de las principales fuentes de información para la consulta de documentos, este hecho ha obligado a la aparición de diversas iniciativas para digitalizar documentos físicos para que puedan ser accedidos en un mundo digital. Google books[1] o Internet Archive[2] son ejemplos del esfuerzo realizado para evitar que la obsolescencia del formato físico dé por perdidos por falta de accesibilidad a una gran cantidad de documentos que no estaban en formato digital.

Si queremos que un documento digitalizado, además de ser accesible por Internet, pueda ser consultado por la mayor cantidad de lectores debemos de poder generar formatos que proporcionen la mayor compatibilidad posible con los dispositivos de lectura actuales. Es una realidad que las plataformas móviles cada vez se utilizan más por los usuarios finales, por tanto, deberemos de proporcionar formatos que permitan acceder con facilidad a los documentos desde estas plataformas y también desde otras plataformas más tradicionales.

2.1 Antecedentes

El Proyecto Final de Carrera surge de una propuesta original de su director Juan Vicente Oltra Gutiérrez. No existen proyectos precedentes que hayan intentado dar una solución a la propuesta original, aunque es relativamente fácil encontrar software que proporcione soluciones parciales en el proceso de conversión no se ha encontrado ningún programa que proporcione una solución global al proceso de conversión de un lote de imágenes a un producto multiformato.

2.2 Propósito y objetivos

Este proyecto trata de dar solución a aquellos usuarios que pretenden generar a partir de imágenes formatos compatibles con las plataformas digitales de uso habitual en la actualidad.

El objetivo principal del Proyecto Final de Carrera es la creación de una aplicación que a partir de imágenes de documentos permita generar de forma sencilla diferentes formatos compatibles con las plataformas actuales, permitiendo así la preservación de los documentos originales en un formato nuevo.

1 http://books.google.es/

2 https://archive.org/

Como objetivos secundarios tenemos:

- Definir unos requisitos mínimos de aplicación para que permita guardar en alta resolución las imágenes que forman parte de los documentos originales y así como los metadatos básicos.

- Conocer el formato interno de los formatos habituales en los dispositivos actuales: epub, cbz, pdf, mobi y html.

- Crear una aplicación de acuerdo a los requisitos planteados pero que sea sencilla de utilizar para un usuario final y que quede oculto la complejidad de los formatos y también las múltiples opciones de exportación que podemos encontrar en otros programas.

- Ofrecer en la aplicación dos roles de usuario para satisfacer el perfil de un usuario básico y un usuario administrador.

- Uso exclusivo de herramientas de software libre para todo el proceso de desarrollo e implantación de la aplicación.

2.3 Motivaciones

La principal motivación para abordar un proyecto de esta temática ha sido que ésta aunaba por una parte la que es mi gran pasión por la informática y por otra parte la gran afinidad del contexto de la aplicación con los estudios de Documentación. Los formatos digitales para contener documentos así como el uso de metadatos para añadir información de éstos forma parte de los contenidos vistos en la Licenciatura en Documentación en asignaturas como Biblioteconomía, Formatos de intercambio de información bibliográfica y Catalogación.

En mi profesión de docente en informática no suelo tener que desarrollar productos software por lo que también me ha servido de reciclaje al tener que revisar y estudiar desarrollo web en PHP, javascript, librerías de desarrollo en PHP, el framework Codeigniter, etc.

2.4 Estructura

El presente Proyecto Final de Carrera está dividido en una serie de capítulos que abarcan todo el proceso de creación del producto final.

El Capítulo 1 y 2 son introductorios al proyecto y ya han sido planteados. Los capítulos 3, 4 y 5 tratan las fases de especificación de requisitos, análisis y diseño de la aplicación siendo estas unas fases habituales en desarrollos de productos software. El capítulo 7 muestra el sistema desarrollado y el capítulo 8 las conclusiones propias del Proyecto Final de Carrera.

Por último se incluye un anexo: un manual básico de estructura y creación de ficheros epub.

3. Especificación de requisitos

La especificación de requisitos es una tarea fundamental en el ciclo de desarrollo del software que pretende recoger las características que debe de satisfacer un programa.

3.1 Introducción

La descripción de los requisitos del programa nos permitirá conocer cuáles son las características fundamentales del interfaz de nuestro programa, saber los contenidos que va a almacenar y las funcionalidades generales del programa indicadas en los objetivos del proyecto.

3.1.1 Propósito

Para la especificación de requisitos del proyecto se ha utilizado como base la normativa IEEE 830-1998 y se sigue de forma general la estructura y el formato indicado en el estándar 830 incluyendo toda la información requerida por la norma. De esta forma obtendremos una especificación de requisitos consistente y sin ambigüedades que permitirá que el desarrollo del software final sea exitoso.

3.1.2 Ámbito

Este proyecto tiene como objetivo principal la creación de una aplicación web que permita generar diferentes formatos estándar a partir de imágenes escaneadas de documentos que previamente han sido almacenadas en una estructura lógica apoyada por un servidor de base de datos.

A la aplicación la hemos denominado eYellow y pretende que el usuario final pueda almacenar con una estructura organizada las imágenes escaneadas que forman parte de documentos digitalizados. Eyellow además proporcionará de forma sencilla generar formatos de fichero estándar para cada uno de los documentos almacenados por los usuarios. Tanto el proceso de creación de los documentos así como las opciones de exportación deben de ser sencillas y amigables, ocultando al usuario final las complejidades de alguno de los formatos finales utilizados.

La aplicación deberá de permitir también el acceso como administrador para realizar tareas más complejas que requieren un mayor nivel de permisos en el sistema.

3.1.3 Definiciones, acrónimos y abreviaturas

Por orden alfabético:

Ajax: Acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML). Técnica de desarrollo web que permite crear aplicaciones interactivas y puede realizar comunicación asíncrona con el servidor web en segundo plano de forma que se pueden realizar cambios en la página web visualizada si necesidad de recargar la página web completa.

Apache: Servidor web HTTP de código abierto disponible para las plataformas mayoritarias y que implementa el protocolo HTTP 1.1. A pesar de que en los últimos años ha sufrido un descenso en su cuota de mercado sigue siendo el servidor web más utilizado en la actualidad (Netcraft, 2013).

Autenticación: Proceso mediante el cual un usuario permite verificar su identidad, habitualmente mediante la introducción de un usuario y una contraseña.

Base de datos: Conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En nuestro caso la base de datos está en formato digital y se utiliza el gestor de base de datos Mysql.

CBZ: Acrónimo de Comic Book Zip. Formato muy habitual para visualizar cómics y consiste básicamente en varios archivos de imagen comprimidos en formato zip.

Codeigniter: Framework para aplicaciones web de código abierto para crear webs dinámicos con PHP. Codeigniter proporciona un conjunto de bibliotecas de desarrollo para tareas comunes, una interfaz simple y una estructura lógica para acceder a esas bibliotecas.

CSS: Acrónimo de Cascading Style Sheets (hojas de estilo en cascada). Lenguaje de hojas de estilos que se utiliza para describir la presentación de un documento escrito en lenguaje de marcas, habitualmente HTML.

Epub: Acrónimo de la expresión inglesa Electronic Publication (Publicación electrónica). Formato redimensionable de código abierto para archivos de libro electrónico. En el formato de libro epub se marca su contenido pero no se delimita su formato ya que éste se adapta en función del dispositivo utilizado. El formato epub se considera en la actualidad un estándar en la publicación de libros electrónicos.

eYellow: Nombre del producto software final generado como objetivo del presente proyecto final de carrera.

Framework para aplicaciones web: Conjunto de librerías software, estructuras y gestión de sesiones utilizados para apoyar el diseño de sitios web dinámicos, aplicaciones y servicios web.

GIF: Formato gráfico utilizando ampliamente en Internet para imágenes y animaciones. GIF es un formato sin pérdida de calidad para imágenes que permite hasta 256 colores.

HTML: Siglas de HyperText Markup Language (Lenguaje de marcado hipertextual). Es el lenguaje de marcado predominante para la elaboración de páginas web.

HTTP: Siglas de Hypertext Transfer Protocol (protocolo de transferencia de hipertexto). Protocolo utilizado en Internet para el acceso a la web. En la actualidad se utiliza la versión 1.1 siendo un protocolo orientado a transacciones entre un cliente y un servidor web.

Interfaz de usuario: Disposición de los distintos elementos de una aplicación web (texto, imágenes, botones, enlaces, menús, ...) que permiten la interacción entre el usuario y la aplicación.

Javascript: Lenguaje de programación interpretado que se utiliza habitualmente en el lado del cliente en una aplicación web y está implementado como parte del navegador web.

JPEG o JPG: Siglas de Joint Photographic Experts Group (Grupo Conjunto de Expertos en Fotografía). Es el formato de archivo de imagen más común utilizado por cámaras fotográficas digitales y otros dispositivos de captura, así como el formato más habitual para almacenar y transmitir imágenes digitales a través de Internet.

Mobipocket o Mobi: Formato para archivos de libro electrónico creado por Mobipocket S.A. y popularizado por Amazon al ser utilizados por los libros electrónicos Kindle. Aunque está basado en un formato abierto, el fichero final .mobi es un binario cuya estructura no es de libre acceso.

Mysql: Sistema de gestión de bases de datos relacional, multihilo y multiusuario. Licenciado como software libre para aquellos productos que sean compatibles con la licencia GNU GPL.

Navegador: Aplicación que permite la visualización de documentos de hipertexto con elementos multimedia obtenidos de un servidor web.

PDF: Siglas del inglés Portable Document Format (formato de documento portátil). Es un formato de almacenamiento de documentos digitales independiente de la plataforma de software o hardware desarrollado por Adobe Systems.

PHP: Lenguaje de programación del lado del servidor utilizado para el desarrollo web de contenido dinámico. En un proyecto de software libre muy utilizado en los servidores web y disponibles para las plataformas mayoritarias.

PNG: Siglas del inglés Portable Network Graphic (Gráficos de red portátiles). Formato gráfico basado en un algoritmo de compresión sin pérdida desarrollado para solventar las deficiencias del formato GIF como por ejemplo la escasa profundidad de color.

Sesión: Instancia que identifica de forma unívoca a un usuario que visita un sitio web desde que se autentifica hasta que abandona el sitio.

ZIP: Formato de compresión sin pérdida muy utilizado para la compresión de datos como documentos, imágenes o programas.

3.1.4 Referencias

Para la redacción de la especificación de requisitos se ha utilizado como fuente principal la norma IEEE 830-1998 “IEEE Recommended Practice for Software Requirements Specifications” (IEEE 830-1998).

3.1.5 Visión global

La especificación de requisitos consta de tres partes:

- Una introducción a la especificación que concluye este punto.
- Una segunda parte que incluye una descripción general donde se describen todos aquellos factores que afectan a producto y sus requisitos.
- Una parte final donde se especifican los requisitos específicos con un nivel de detalle suficiente que permita a los diseñadores diseñar un sistema que satisfaga por completo estos requisitos.

3.2 Descripción general

3.2.1 Perspectiva del producto

El producto a desarrollar no se incluye en un sistema más amplío, constituye por sí sólo en un producto autónomo que funcionará sobre un servidor web y al que se podrá acceder mediante cualquier navegador web actual.

Para su implementación deben de utilizarse las siguientes tecnologías:

- Lenguaje HTML como lenguaje de marcado para elaborar las páginas web.
- Hojas de estilo CSS para describir el aspecto y el formato de todo el sitio web. Las plantillas utilizadas deben de reutilizarse en cada una de las páginas del proyecto para que sea sencillo modificar la apariencia visual del sitio.
- Lenguaje Javascript para proporcionar cierta interactividad en el lado del cliente, por ejemplo, para mostrar un diálogo de confirmación de borrado de un documento.
- Lenguaje PHP para generar las páginas de forma dinámica desde el servidor web. El proyecto se ha desarrollado utilizando PHP 5.3 pero debe de funcionar en cualquier rama de la versión 5.x de PHP.
- Tecnología Ajax para añadir interactividad a la aplicación web sin necesidad de recargar la página de forma completa.
- Mysql como servidor de base de datos. La versión de Mysql utilizada en el desarrollo del proyecto ha sido la 5.5.31

Al tratarse de un proyecto web se debe de alojar todo el sitio en un servidor web que admita las tecnologías antes mencionadas. Se recomienda el servidor web apache utilizado en el desarrollo del producto ya que es un servidor mayoritario y es un estándar de facto.

3.2.2 Funciones del producto

Las funciones ofrecidas por el producto desarrollado deben de ser:

- Autenticación del usuario mediante usuario y contraseña.
- Mantenimiento de los documentos propios:
    - Consulta, altas, bajas y modificaciones.
    - Adición de portada del documento.
    - Adición de metadatos asociados al documento: tipología, autor, lengua, editor, ...
    - Mantenimiento de las diferentes imágenes escaneadas que conforman el documento: Subir ficheros, eliminar, reordenar, ...
- Exportación de los documentos a formatos estándar:
    - PDF: Con posibilidad de elegir el formato de página y la orientación.
    - HTML: A elegir por el usuario diferentes plantillas para generar el HTML.
    - CBZ: Dar la opción al usuario de elegir la calidad de las imágenes contenidas en el fichero generado.
    - Epub: El usuario debe de poder elegir un modelo de ebook o una resolución estándar y se debe de generar de forma automática el epub.
    - Mobi: Con opciones similares a la exportación de wpub.
- Mantenimiento de autores.
- Mantenimiento de editores.
- Mantenimiento de revistas.
- Mantenimiento de etiquetas o tags.
- Posibilidad de categorización de los documentos mediante diferentes tags o etiquetas.
- Mantenimiento de usuario del sistema. Accesible sólo mediante privilegios de administración.
- Mantenimiento de tipologías de documentos. Accesible sólo mediante privilegios de administración.

- Mantenimiento de lenguas del documento. Accesible sólo mediante privilegios de administración.

3.2.3 Características del usuario

Se establece como requisito de la aplicación que el usuario que haga uso de ella debe de haberse autentificado con anterioridad y, por lo tanto, se prohíbe el acceso anónimo a la aplicación.

Una vez el usuario ya se ha autentificado se distinguen dos categorías:

- Usuario normal: Pueden cargar y editar sus propios documentos.
- Usuario administrador: Tiene acceso a todos los documentos de todos los usuarios así como a otras opciones restringidas como el mantenimiento de usuarios o el mantenimiento de tipologías de documentos.

3.2.4 Restricciones generales

Este producto no depende de un sistema mayor así que sólo necesitamos desde el punto de vista del servidor un servidor web con soporte PHP y Mysql y desde el punto de vista del cliente un navegador web actualizado.

Debido a que las imágenes que conforman los documentos ocupan mucho espacio del alojamiento del servidor web es necesario tener en cuenta que el espacio web no debe de ser un elemento crítico y se debe de disponer de él sin problemas.

Sería conveniente para garantizar la seguridad de los datos utilizar acceso encriptado al servidor mediante https pero no es un requisito imprescindible.

3.2.5 Supuestos y dependencias

El proyecto debe de poder ser alojado en cualquier servidor web con soporte PHP y Mysql, debería de ser, por tanto, independiente del sistema operativo que utilice el servidor web. Para el desarrollo y pruebas del programa siempre se ha utilizado un servidor web apache 2.2 bajo una plataforma Linux con kernel 3.2. La versión de PHP utilizada ha sido la 5.3 y la versión 5.5 del servidor Mysql.

El navegador web necesario para acceder a la aplicación del servidor web es conveniente que esté actualizado a una versión reciente y con esto debería de ser suficiente para poder acceder a la aplicación eYellow. El navegador debe de ser

compatible con el protocolo HTTP 1.1, javascript, HTML 4.0 y CSS. Estas características las cumplen todos los navegadores más utilizados en la actualidad.

3.3 Requisitos específicos

En esta sección detallaremos los requisitos con un nivel de detalle suficiente para permitir a los diseñadores que el sistema diseñado satisfaga estos requisitos indicados. Además debe de permitir al equipo de pruebas planificar y realizar las pruebas sobre la aplicación que confirmen si el sistema satisface, o no, los requisitos.

3.3.1 Requisitos de interfaces externos

Se describirán los requisitos que afecten a la interfaz de usuario, interfaz con otros sistemas (hardware y software) e interfaces de comunicaciones.

3.3.1.1 Interfaz de usuario

La aplicación debe de permitir dos tipologías de usuario y consecuentemente dos interfaz de usuario:

- Usuario normal:
    - En la zona de cabecera de la página web debe de aparecer el nombre del usuario con el que se ha abierto sesión y un enlace que posibilite el cierre de sesión.
    - Debajo de la zona de cabecera debe de aparecer una barra de menú horizontal que permita al usuario acceder a las opciones más habituales: Ver mis documentos, crear nuevo documento, mantenimiento de autores, mantenimiento de editores, mantenimiento de revistas y mantenimiento de etiquetas.
    - En la zona central aparecerá por defecto la visualización de los documentos del usuario.
    - Se deben de poder crear nuevos documentos así como visualizar, borrar y modificar los ya existentes.
    - El proceso de exportación debe de estar accesible en la columna derecha cuando se esté visualizando un documento. Debe de aparecer un botón o enlace para cada uno de los formatos de exportación.
    - El proceso de exportación de un formato concreto debe de permitir modificar los parámetros básicos del formato.

- Para evitar errores el proceso de borrado de un documento solicitará confirmación del usuario.

- Usuario administrador:
  - En la zona de cabecera de la página web debe de aparecer el nombre del usuario con el que se ha abierto sesión indicando que es administrador y un enlace que posibilite el cierre de sesión.
  - En la zona central de la aplicación para un administrador debe de aparecer un menú que permita acceder a todas las opciones que se puedan realizar desde el programa. Todas las opciones estarán agrupadas en cuatro áreas: Administrador, documentos, auxiliares documentos y auxiliares generales.
  - El mantenimiento de documento con rol de administrador da acceso a todos los documentos de todos los usuarios que hayan cargado en el sistema.

3.3.1.2 Interfaz de Hardware

El servidor debe de tener el hardware adecuado para que pueda ejecutar los requisitos indicados en el punto 3.2.5. Teniendo en cuenta que los requisitos hardware del servidor no son demasiado exigentes habrá que tener en cuenta la carga que vaya a soportar el sistema para elegir el hardware concreto a utilizar, esta carga estará condicionada a cuatro recursos hardware básicos: CPU, memoria RAM, espacio de almacenamiento secundario y ancho de banda de la conexión de red.

El cliente debe de ser capaz de ejecutar con agilidad un navegador actual tal cual se ha indicado en el punto 3.2.5. Es recomendable para el acceso a la aplicación hacerlo desde un dispositivo que soporte al menos una resolución de 1024x768 píxeles.

3.3.1.3 Interfaz de Software

En el lado del servidor debemos de tener un sistema ejecutando un servidor web con soporte PHP y Mysql. En el lado del cliente un navegador actualizado es suficiente para acceder a la aplicación: Firefox, Opera, Chrome, Safari o Internet explorer son ejemplos de navegadores válidos. En el punto 3.2.5 se indica con más detalle las dependencias que debe de cumplir el software tanto en el servidor como en el cliente.

3.3.1.4 Interfaz de Comunicación

La comunicación se realizará mediante el protocolo HTTP bajo la arquitectura de comunicaciones de Internet o TCP/IP. La aplicación podrá ser accedida en local o en una red local, sólo necesitaremos conocer la IP y el puerto en el que está atendiendo las peticiones el servidor web. Si la aplicación va a ser accedida a través de Internet sería conveniente utilizar para facilitar el acceso un dominio o un subdominio de Internet.

Es importante destacar que la interfaz de comunicación puede ser un cuello de botella de la aplicación por lo que se deberá de calcular la carga de usuarios simultánea sobre el servidor web y la aplicación para contratar el ancho de banda adecuado para que los usuarios finales no aprecien retardo en el uso de la aplicación.

3.3.2 Requisitos funcionales

Los requisitos funcionales que debe de cumplir la aplicación por tipos de usuario son:

- Usuario normal:

N1	Autenticación en la aplicación
Introducción	En el caso de que un usuario no esté autenticado debe de aparecer por defecto siempre la pantalla de acceso a la aplicación. No debe de poderse consultar ninguna información en el sistema si el usuario no está autenticado.
Entradas	El usuario introduce su usuario y contraseña.
Proceso	El sistema comprueba si el usuario está registrado en el sistema consultando la tabla de la base de datos que almacena la información de los usuarios. Si el usuario está registrado en el sistema se inicializa una sesión, en el caso de ser administrador se muestra el menú de administrador. Si la contraseña es incorrecta o el usuario no existe no se permite el inicio de sesión.
Salidas	Si el proceso de autenticación es exitoso se muestra al usuario la pantalla de inicio de la aplicación y se indica en la barra de mensajes superior el nombre del usuario con el que se ha autenticado en el sistema. Si el proceso de autenticación es erróneo se indica al usuario que el usuario o la contraseña es incorrecta y se invita a introducir de nuevo sus credenciales.

N2	Consultar mis documentos
Introducción	Un usuario normal debe de poder consultar todos los documentos propios.
Entradas	Usuario.
Proceso	El sistema consulta la tabla de la base de datos que almacena la información de los documentos filtrando por

	el usuario activo.
Salidas	Si la consulta devuelve información se muestra al usuario en la zona central una lista de sus documentos con información básica: portada, título y año. Si no hay documentos asociados al usuario simplemente se muestra en la zona central una lista vacía.

N3	**Consultar un documento**
Introducción	Un usuario normal debe de poder consultar en detalle un documento incluido en sus documentos propios.
Entradas	Usuario y documento.
Proceso	El sistema consulta la tabla de la base de datos que almacena la información de los documentos filtrando por el usuario activo y el código de documento.
Salidas	Si la consulta devuelve información se muestra al usuario con detalle toda la información del documento: portada, título, tipología, autor, lengua, editor, año edición, observaciones, etiquetas y total de páginas. Se debe de poder paginar y visualizar las diferentes imágenes que componen el documento. Debe de haber un acceso fácil a la edición del documento, borrado del documento y edición de las páginas que componen el documento. También desde esta pantalla se debe de poder acceder a las opciones de exportación del documento: pdf, html, epub, cbz y mobi. Si el documento es inexistente o es un documento que pertenece a otro usuario la salida debe de ser nula y no debe de aparecer nada en la zona central de la aplicación.

N4	**Edición de un documento**
Introducción	Un usuario normal debe de poder editar el contenido de un documento incluido en sus documentos propios.
Entradas	Usuario y documento.
Proceso	El sistema consulta la tabla de la base de datos que

	almacena la información de los documentos filtrando por el usuario activo y el código de documento.
Salidas	Si la consulta devuelve información se muestra al usuario un formulario que permite modificar toda la información del documento: portada, título, tipología, autor, lengua, editor, número de edición, año edición, revista (sólo en la tipología de artículo de revista), observaciones y etiquetas. Tras pulsar el botón "Actualizar cambios" se comprueban los datos y se almacenan en base de datos. Si el documento es inexistente o es un documento que pertenece a otro usuario la salida debe de ser nula y no debe de aparecer nada en la zona central de la aplicación.

N5	**Alta de un documento**
Introducción	Un usuario normal debe de poder dar de alta documentos propios.
Entradas	Usuario y datos del documento.
Proceso	El sistema muestra un formulario de alta de documentos donde el usuario puede cumplimentar todos los datos solicitados a excepción de las imágenes o páginas que componen el documento. Los datos que debe de solicitar el formulario son: portada, título, tipología, autor, lengua, editor, número de edición, año edición, revista (sólo en la tipología de artículo de revista), observaciones y etiquetas.
Salidas	Tras pulsar el botón "Guardar" se comprueban los datos y se almacenan en base de datos toda la información necesaria para crear un nuevo documento. Si en el proceso de alta surge algún error el sistema debe de mostrar en el mismo formulario el error para que el usuario lo corrija.

N6	**Borrado de un documento**
Introducción	Un usuario normal debe de poder borrar sus documentos propios.
Entradas	Usuario y código del documento.

Proceso	Desde la consulta de documentos debe de aparecer un enlace donde el usuario pueda eliminar de forma fácil el documento y la imágenes asociadas al documento. Al tratarse de un proceso no reversible se le debe de pedir al usuario confirmación del borrado para evitar errores. El proceso de borrado debe de comprobar que no se extá intentando borrar un documento del que el usuario final no es el propietario.
Salidas	Tras pulsar el botón “Borrar” y confirmar el borrado se eliminan todos los datos del documento en la base de datos. Si en el proceso de borrado surge algún error el sistema debe de mostrar un mensaje al usuario final.

N7	**Edición de las páginas de un documento**
Introducción	Las páginas que componen un documento son imágenes y se le debe de proporcionar al usuario final una herramienta sencilla para subir las imágenes al servidor web así como borrar y reordenar aquellas imágenes que ya se han subido.
Entradas	Usuario, código del documento e imágenes que componen el documento.
Proceso	Desde la consulta de documentos debe de aparecer un enlace donde el usuario pueda acceder al mantenimiento de las imágenes que componen el documento. Debemos de poder subir una o varias imágenes desde nuestro equipo local al servidor web. Los formatos que debe de admitir la aplicación son JPG, PNG y GIF. Debe de ser sencillo eliminar uno a varias imágenes ya subidas así como reordenarlas.
Salidas	Tras pulsar el botón “Subir ficheros” y confirmar la subida al servidor las imágenes se incorporan al documento activo. Si en el proceso de subida de ficheros surge algún error el sistema debe de mostrar un mensaje al usuario final.

N8	Cerrar sesión
Introducción	Cualquier usuario debe de poder cerrar la sesión activa con un enlace desde cualquier pantalla de la aplicación. El enlace debe de estar en la barra de herramientas superior junto al nombre del usuario ya autentificado.
Entradas	Usuario y el enlace de cierre de sesión.
Proceso	Una vez se ha pulsado el enlace de cierre de sesión se anulan todas las variables asociadas a la sesión así como la misma sesión.
Salidas	Si el cierre de sesión es satisfactorio la aplicación se redirigirá a la pantalla de autenticación donde se solicita de nuevo un usuario y contraseña.

N9	Mantenimiento de autores
Introducción	Todos los usuarios pueden hacer un mantenimiento básico de los autores del sistema: altas, bajas, consultas y modificaciones.
Entradas	Listado y formulario de mantenimiento de autores.
Proceso	Inicialmente se muestra un listado de todos los autores almacenados en base de datos. El usuario debe de tener acceso a enlaces que permitan las siguientes acciones: añadir autor, exportar, imprimir y buscar. Si se añade un autor el sistema muestra un formulario que solicita nombre y apellidos del nuevo autor. La opción de exportar permite la descarga de un fichero de hoja de cálculo con el contenido de la tabla autores. La opción de imprimir muestra un diálogo de selección de impresora y tras confirmación imprime un reporte con el contenido de la tabla autores. La opción de búsqueda permita realizar un filtrado de los autores permitiendo poner condiciones sobre cada uno de los campos existentes. Si hay demasiados autores y no se pueden visualizar en una página la aplicación mostrará una barra de navegación de los registros y permitirá cambiar la página visualizada.

Salidas	Si todas las acciones realizadas son satisfactorias la aplicación va mostrando los cambios aplicados en los mismos resultados del mantenimiento. Si ocurre algún error la aplicación debe de mostrar un mensaje al usuario final.

N10	**Mantenimiento de editores**
Introducción	Todos los usuarios pueden hacer un mantenimiento básico de los editores del sistema: altas, bajas, consultas y modificaciones.
Entradas	Listado y formulario de mantenimiento de editores.
Proceso	Inicialmente se muestra un listado de todos los editores almacenados en base de datos. El usuario debe de tener acceso a enlaces que permitan las siguientes acciones: añadir editor, exportar, imprimir y buscar. Si se añade un editor el sistema muestra un formulario que solicita nombre del nuevo editor. La opción de exportar permite la descarga de un fichero de hoja de cálculo con el contenido de la tabla editores. La opción de imprimir muestra un diálogo de selección de impresora y tras confirmación imprime un reporte con el contenido de la tabla editores. La opción de búsqueda permita realizar un filtrado de los editores permitiendo poner condiciones sobre cada uno de los campos existentes. Si hay demasiados editores y no se pueden visualizar en una página la aplicación mostrará una barra de navegación de los registros y permitirá cambiar la página visualizada.
Salidas	Si todas las acciones realizadas son satisfactorias la aplicación va mostrando los cambios aplicados en los mismos resultados del mantenimiento. Si ocurre algún error la aplicación debe de mostrar un mensaje al usuario final.

N11	Mantenimiento de revistas
Introducción	Todos los usuarios pueden hacer un mantenimiento básico de las revistas del sistema: altas, bajas, consultas y modificaciones.
Entradas	Listado y formulario de mantenimiento de revistas.
Proceso	Inicialmente se muestra un listado de todos las revistas almacenadas en base de datos. El usuario debe de tener acceso a enlaces que permitan las siguientes acciones: añadir revista, exportar, imprimir y buscar. Si se añade una revista el sistema muestra un formulario que solicita nombre de la nueva revista. La opción de exportar permite la descarga de un fichero de hoja de cálculo con el contenido de la tabla revistas. La opción de imprimir muestra un diálogo de selección de impresora y tras confirmación imprime un reporte con el contenido de la tabla revistas. La opción de búsqueda permita realizar un filtrado de las revistas permitiendo poner condiciones sobre cada uno de los campos existentes. Si hay demasiadas revistas y no se pueden visualizar en una página la aplicación mostrará una barra de navegación de los registros y permitirá cambiar la página visualizada.
Salidas	Si todas las acciones realizadas son satisfactorias la aplicación va mostrando los cambios aplicados en los mismos resultados del mantenimiento. Si ocurre algún error la aplicación debe de mostrar un mensaje al usuario final.

N12	Mantenimiento de etiquetas
Introducción	Todos los usuarios pueden hacer un mantenimiento básico de las etiquetas del sistema: altas, bajas, consultas y modificaciones.
Entradas	Listado y formulario de mantenimiento de etiquetas.
Proceso	Inicialmente se muestra un listado de todos las etiquetas almacenadas en base de datos. El usuario debe de tener

	acceso a enlaces que permitan las siguientes acciones: añadir etiqueta, exportar, imprimir y buscar. Si se añade una revista el sistema muestra un formulario que solicita nombre de la nueva etiqueta. La opción de exportar permite la descarga de un fichero de hoja de cálculo con el contenido de la tabla etiquetas. La opción de imprimir muestra un diálogo de selección de impresora y tras confirmación imprime un reporte con el contenido de la tabla etiquetas. La opción de búsqueda permita realizar un filtrado de las etiquetas permitiendo poner condiciones sobre cada uno de los campos existentes. Si hay demasiadas etiquetas y no se pueden visualizar en una página la aplicación mostrará una barra de navegación de los registros y permitirá cambiar la página visualizada.
Salidas	Si todas las acciones realizadas son satisfactorias la aplicación va mostrando los cambios aplicados en los mismos resultados del mantenimiento. Si ocurre algún error la aplicación debe de mostrar un mensaje al usuario final.

N13	**Exportar documento a PDF**
Introducción	El objetivo principal de la aplicación es poder exportar los documentos a un formato estándar como es el PDF. Todos los usuarios pueden exportar sus propios documentos al formato indicado.
Entradas	Usuario activo y documento a exportar. Parámetros exportación que debe de mostrar la aplicación: Formato de página (A4, A5, ...) y orientación de la página (normal o apaisada)
Proceso	Una vez indicados los parámetros de exportación se pulsa el botón “Generar PDF”. El proceso de generación del PDF comprueba que el documento pertenece al usuario activo y a partir de los

	datos del documento y las opciones de exportación indicadas por el usuario genera un fichero PDF que contiene el documento. Aparece un diálogo de descarga con el documento generado en el proceso.
Salidas	Si todo el proceso realizado es satisfactorio la aplicación genera y oferta para descargar un fichero PDF. Si ocurre algún error la aplicación debe de mostrar un mensaje al usuario final.

N14	**Exportar documento a HTML**
Introducción	El objetivo principal de la aplicación es poder exportar los documentos a un formato estándar como es HTML. Todos los usuarios pueden exportar sus propios documentos al formato indicado.
Entradas	Usuario activo y documento a exportar. Parámetros exportación que debe de mostrar la aplicación: Hoja de estilos (básica, CSS, ...)
Proceso	Una vez indicados los parámetros de exportación se pulsa el botón “Generar HTML”. El proceso de generación de HTML comprueba que el documento pertenece al usuario activo y a partir de los datos del documento y las opciones de exportación indicadas por el usuario genera un fichero comprimido zip que contiene el documento en formato HTML multipágina. Aparece un diálogo de descarga con el documento comprimido generado en el proceso.
Salidas	Si todo el proceso realizado es satisfactorio la aplicación genera y oferta para descargar un fichero zip comprimido con el contenido del documento. Si ocurre algún error la aplicación debe de mostrar un mensaje al usuario final.

N15	Exportar documento a CBZ
Introducción	El objetivo principal de la aplicación es poder exportar los documentos a un formato estándar como el CBZ. Todos los usuarios pueden exportar sus propios documentos al formato indicado.
Entradas	Usuario activo y documento a exportar. Parámetros exportación que debe de mostrar la aplicación: Resolución de las imágenes (baja, media, máxima resolución, ...)
Proceso	Una vez indicados los parámetros de exportación se pulsa el botón “Generar CBZ”. El proceso de generación del CBZ comprueba que el documento pertenece al usuario activo y a partir de los datos del documento y las opciones de exportación indicadas por el usuario genera un fichero CBZ que contiene el documento. Aparece un diálogo de descarga con el documento CBZ generado en el proceso.
Salidas	Si todo el proceso realizado es satisfactorio la aplicación genera y oferta para descargar un fichero CBZ con el contenido del documento. Si ocurre algún error la aplicación debe de mostrar un mensaje al usuario final.

N16	Exportar documento a EPUB
Introducción	El objetivo principal de la aplicación es poder exportar los documentos a un formato estándar como el EPUB. Todos los usuarios pueden exportar sus propios documentos al formato indicado.
Entradas	Usuario activo y documento a exportar. Parámetros exportación que debe de mostrar la aplicación: Calidad del epub donde se indica la resolución de las imágenes contenidas en el epub. La aplicación ofertará muchos modelos de ebooks actuales para que se

	asigne la calidad requerida de forma automática.
Proceso	Una vez indicados los parámetros de exportación se pulsa el botón “Generar EPUB”. El proceso de generación del EPUB comprueba que el documento pertenece al usuario activo y a partir de los datos del documento y las opciones de exportación indicadas por el usuario genera un fichero EPUB que contiene el documento. Aparece un diálogo de descarga con el documento EPUB generado en el proceso.
Salidas	Si todo el proceso realizado es satisfactorio la aplicación genera y oferta para descargar un fichero EPUB con el contenido del documento. Si ocurre algún error la aplicación debe de mostrar un mensaje al usuario final.

N17	**Exportar documento a MOBI**
Introducción	El objetivo principal de la aplicación es poder exportar los documentos a un formato estándar como el MOBI. Todos los usuarios pueden exportar sus propios documentos al formato indicado.
Entradas	Usuario activo y documento a exportar. Parámetros exportación que debe de mostrar la aplicación: Calidad del epub donde se indica la resolución de las imágenes contenidas en el epub. La aplicación ofertará muchos modelos de ebooks actuales para que se asigne la calidad requerida de forma automática.
Proceso	Una vez indicados los parámetros de exportación se pulsa el botón “Generar MOBI”. El proceso de generación del MOBI comprueba que el documento pertenece al usuario activo y a partir de los datos del documento y las opciones de exportación indicadas por el usuario genera un fichero MOBI que contiene el documento. Aparece un diálogo de descarga con el documento MOBI generado en el proceso.

Salidas	Si todo el proceso realizado es satisfactorio la aplicación genera y oferta para descargar un fichero MOBI con el contenido del documento. Si ocurre algún error la aplicación debe de mostrar un mensaje al usuario final.

- Usuario administrador:

  Un usuario con rol de administrador debe de poder realizar todas las acciones que realiza un usuario normal además de otras acciones que están asociadas a su rol.

A1	**Consultar todos los documentos**
Introducción	Un administrador debe de poder consultar todos los documentos propios y de otros usuarios.
Entradas	Usuario.
Proceso	El sistema consulta la tabla de la base de datos que almacena la información de todos los documentos.
Salidas	Si la consulta devuelve información se muestra al usuario en la zona central una lista de sus documentos con información básica: portada, título, año y propietario. Si no hay documentos asociados al usuario simplemente se muestra en la zona central una lista vacía.

A2	**Edición de un documento**
Introducción	Un administrador debe de poder editar el contenido de cualquier documento del sistema.
Entradas	Usuario y documento.
Proceso	El sistema consulta la tabla de la base de datos que almacena la información de los documentos.
Salidas	Si la consulta devuelve información se muestra al usuario un formulario que permite modificar toda la información del documento: portada, título, tipología, autor, lengua, editor, número de edición, año edición, revista (sólo en la

	tipología de artículo de revista), observaciones y etiquetas. Tras pulsar el botón “Actualizar cambios” se comprueban los datos y se almacenan en base de datos. Si el documento es inexistente la salida debe de ser nula y no debe de aparecer nada en la zona central de la aplicación.

A3	**Mantenimiento de usuarios**
Introducción	Los administradores pueden hacer un mantenimiento básico de los usuarios que pueden acceder al sistema: altas, bajas, consultas y modificaciones.
Entradas	Listado y formulario de mantenimiento de usuarios.
Proceso	Inicialmente se muestra un listado de todos los usuarios almacenados en base de datos. El administrador debe de tener acceso a enlaces que permitan las siguientes acciones: añadir usuario, exportar, imprimir y buscar. Si se añade un usuario el sistema muestra un formulario que solicita los siguientes datos del nuevo usuario: usuario, contraseña, email, nombre, apellidos, fecha de nacimiento y si es administrador o no. La opción de exportar permite la descarga de un fichero de hoja de cálculo con el contenido de la tabla de usuarios. La opción de imprimir muestra un diálogo de selección de impresora y tras confirmación imprime un reporte con el contenido de la tabla usuarios. La opción de búsqueda permita realizar un filtrado de los usuarios permitiendo poner condiciones sobre cada uno de los campos existentes. Si hay demasiados usuarios y no se pueden visualizar en una página la aplicación mostrará una barra de navegación de los registros y permitirá cambiar la página visualizada.
Salidas	Si todas las acciones realizadas son satisfactorias la aplicación va mostrando los cambios aplicados en los mismos resultados del mantenimiento. Si ocurre algún error la aplicación debe de mostrar un mensaje al usuario final.

A4	Mantenimiento de tipos de documento
Introducción	Los administradores pueden hacer un mantenimiento básico de los tipos de documento que pueden asignarse: altas, bajas, consultas y modificaciones.
Entradas	Listado y formulario de mantenimiento de tipos de documento. Deben de incluirse al menos las tipologías: monografía, cómic y artículo de revista.
Proceso	Inicialmente se muestra un listado de todos los tipos de documento almacenados en base de datos. El administrador debe de tener acceso a enlaces que permitan las siguientes acciones: añadir tipo de documento, exportar, imprimir y buscar. Si se añade un tipo de documento el sistema muestra un formulario que solicita el nombre del nuevo tipo de documento. La opción de exportar permite la descarga de un fichero de hoja de cálculo con el contenido de la tabla de tipos de documento. La opción de imprimir muestra un diálogo de selección de impresora y tras confirmación imprime un reporte con el contenido de la tabla tipos de documento. La opción de búsqueda permita realizar un filtrado de los tipos de documento permitiendo poner condiciones sobre cada uno de los campos existentes. Si hay demasiados tipos de documento y no se pueden visualizar en una página la aplicación mostrará una barra de navegación de los registros y permitirá cambiar la página visualizada.
Salidas	Si todas las acciones realizadas son satisfactorias la aplicación va mostrando los cambios aplicados en los mismos resultados del mantenimiento. Si ocurre algún error la aplicación debe de mostrar un mensaje al usuario final.

A5	Mantenimiento de lenguas
Introducción	Los administradores pueden hacer un mantenimiento básico de las lenguas que pueden asignarse a los documentos: altas, bajas, consultas y modificaciones.
Entradas	Listado y formulario de mantenimiento de lenguas.
Proceso	Inicialmente se muestra un listado de todas las lenguas almacenadas en base de datos. El administrador debe de tener acceso a enlaces que permitan las siguientes acciones: añadir lengua, exportar, imprimir y buscar. Si se añade una lengua el sistema muestra un formulario que solicita el nombre del nuevo tipo de documento. La opción de exportar permite la descarga de un fichero de hoja de cálculo con el contenido de la tabla de lenguas. La opción de imprimir muestra un diálogo de selección de impresora y tras confirmación imprime un reporte con el contenido de la tabla tipos de lenguas. La opción de búsqueda permita realizar un filtrado de las lenguas permitiendo poner condiciones sobre cada uno de los campos existentes. Si hay demasiadas lenguas y no se pueden visualizar en una página la aplicación mostrará una barra de navegación de los registros y permitirá cambiar la página visualizada.
Salidas	Si todas las acciones realizadas son satisfactorias la aplicación va mostrando los cambios aplicados en los mismos resultados del mantenimiento. Si ocurre algún error la aplicación debe de mostrar un mensaje al usuario final.

3.3.3 Requisitos de rendimiento

Los requisitos de rendimiento de la aplicación son dependientes de la carga de usuarios que va a soportar el servidor web, a mayor cantidad de usuarios se llevarán a cabo mayor cantidad de transacciones y obligará a tener un servidor con mayor capacidad de proceso, mayor cantidad de memoria y un mayor ancho de banda en su conexión a la red que utilicen los usuarios, ya sea intranet o internet. Debido a que los datos que se almacenan para cada uno de los documentos son principalmente imágenes se requiere

que el servidor web tenga una gran cantidad de espacio de almacenamiento secundario disponible para la subida de imágenes de los usuarios.

3.3.4 Atributos del sistema

En cuanto a otros atributos de calidad que deben de tenerse en cuenta en el sistema:

- **Fiabilidad**: La aplicación debe de ser fiable y estar preparada ante fallos no esperados, dando información concreta y no ambigua al usuario final del error ocurrido.
- **Seguridad**: Este es un aspecto fundamental en aquellas aplicaciones que están accesibles en redes como Internet. Hay que intentar evitar accesos no autorizados o ataques a la aplicación por lo que en el programa deben de tenerse en cuenta las siguientes técnicas de seguridad:
    - Sólo es posible acceder al sistema si hay una autenticación previa mediante un usuario y contraseña. Esta información estará almacenada en base de datos.
    - Los usuarios se agruparán por tipologías y cada tipología tendrá previstas unas acciones propias. Cada vez que se ejecute una acción se debe de comprobar si el usuario activo tiene permisos para realizar esa acción. Además en los usuarios normales no será posible acceder y/o modificar datos de otro usuario por lo que habrá que comprobar cada vez que se realicen estas acciones básicas si el documento le pertenece al usuario activo.
    - Se utilizará el método POST para el paso de valores al servidor web minimizando el riesgo de ataques por modificaciones de los valores en la URL al utilizar el método GET.
- **Disponibilidad**: Si la aplicación va a estar accesible por Internet debemos de alojarla en un servidor web que tenga un porcentaje de disponibilidad cercano al 99% 24 horas al día 7 días a la semana.
- **Mantenibilidad**: El servidor web requiere de tareas de mantenimiento para garantizar el funcionamiento óptimo de la aplicación, para ello el administrador del servidor deberá de realizar tareas o comprobaciones periódicas como:
    - Espacio libre en disco utilizado para almacenamiento de las imágenes.
    - Análisis de logs para comprobar accesos no autorizados o ataques sobre el servidor.
    - Espacio libre en disco utilizado por el servidor de la base de datos.
    - Análisis de logs de errores del servidor web.

- Análisis de la carga de la CPU del servidor web para encontrar picos de carga y cuellos de botella.

- **Portabilidad**: La aplicación debe de ser portable a cualquier otro servidor web que cumpla las características indicadas en el punto 3.2.5

4. Análisis

Una vez descritos los requisitos de la aplicación se va a realizar en la fase de análisis una descripción formal de las características que debería tener la aplicación.

4.1 Introducción

Para hacer el modelado de la aplicación utilizaremos el Lenguaje Unificado de Modelado o UML (Unified Modeling Language) ya que es el lenguaje de modelado más conocido y utilizado en la actualidad, este lenguaje gráfico nos permite visualizar, especificar, construir y documentar un sistema.

UML se compone de diversos elementos gráficos que combinados permiten crear diagramas cuya finalidad es presentar diversas perspectivas de un sistema que denominaremos modelo (Schmuller, 2001: 27). Existen diferentes tipos de diagramas en UML: de clases, de casos de uso, de estados, de secuencias, de actividades, de componentes, de distribución. En nuestro análisis utilizaremos sólo los diagramas de casos de uso y de clases porque son suficientes para modelizar nuestro sistema.

4.2 Casos de uso

El diagrama de casos de uso ayuda al analista a entender cómo se va a comportar el sistema y así obtener los requerimientos desde el punto de vista del usuario. Esta cercanía al usuario final se manifiesta en la sencilla notación utilizada en UML para representar los casos de uso.

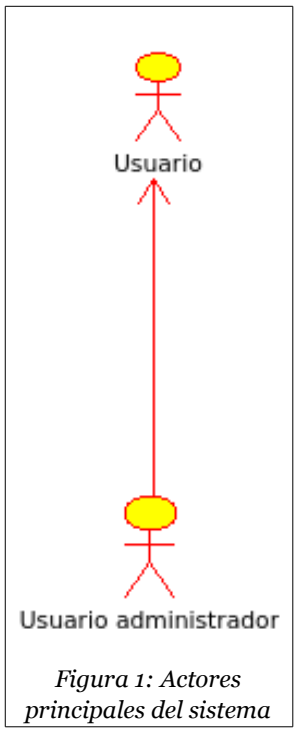


Figura 1: Actores principales del sistema

En la figura 1 se muestran los actores principales del sistema. El usuario administrador es una especialización del actor principal del sistema que es el usuario de la web. Tendremos casos de uso para ambos actores del sistema: un usuario normal y un usuario administrador.

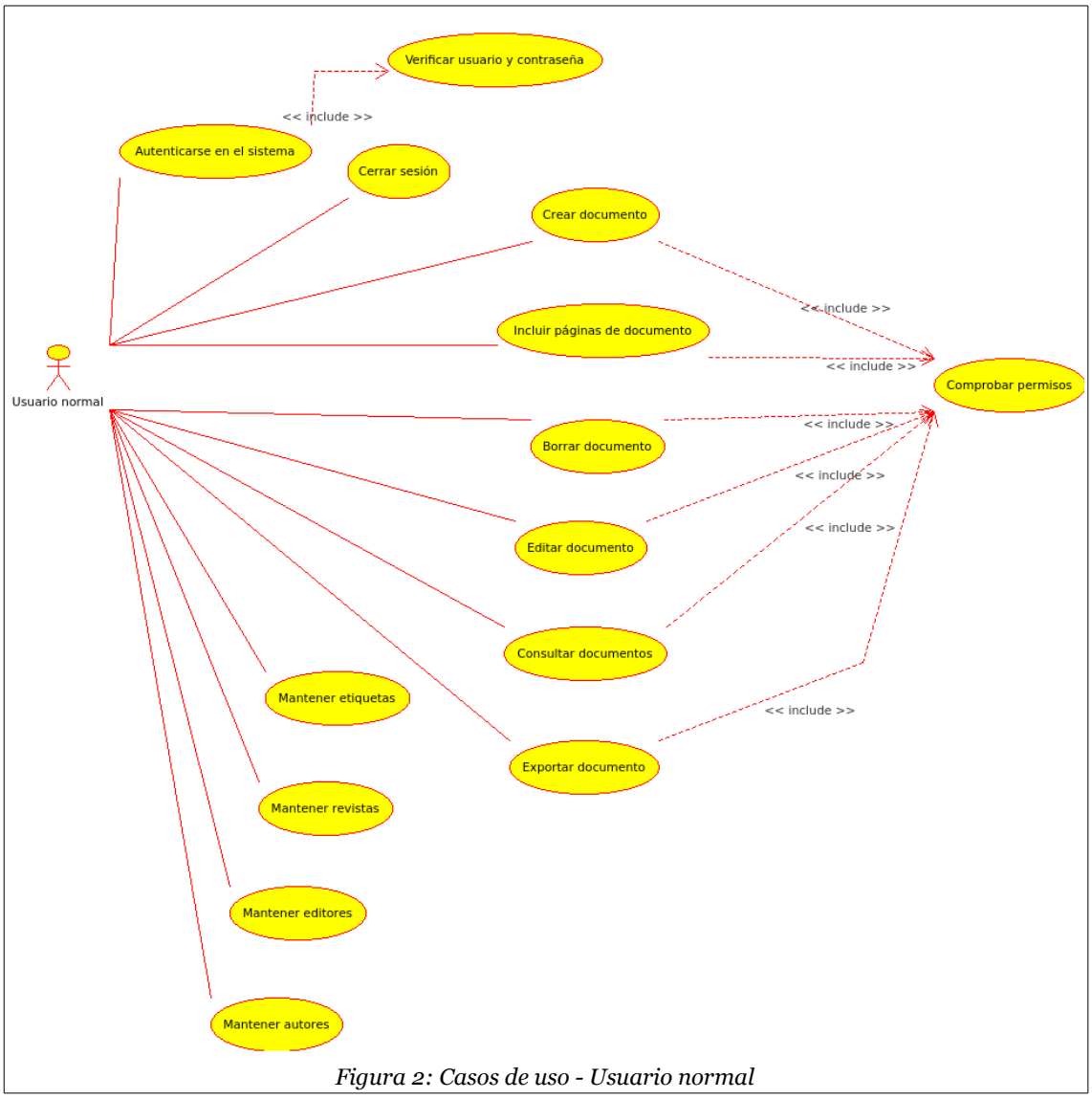


Figura 2: Casos de uso - Usuario normal

La figura 2 representa los casos de uso asociados a un usuario normal del sistema. Los casos de uso de un usuario normal pueden agruparse en 3 categorías: entrada y salida del sistema, mantenimiento de tablas auxiliares y gestión de documentos que incluye la opción de exportar el documento en los formatos ya indicados.

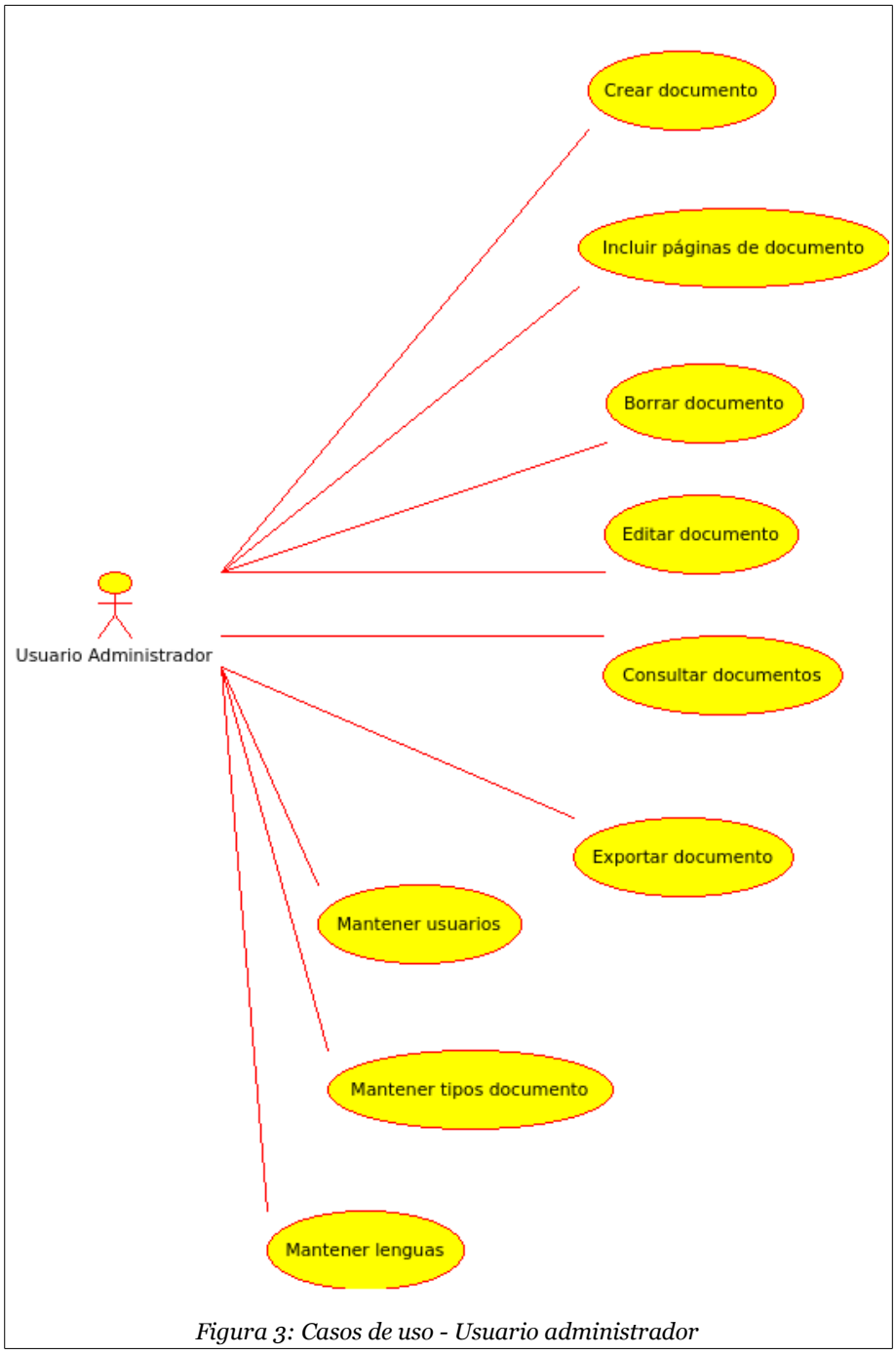


Figura 3: Casos de uso - Usuario administrador

En la figura 3 se han representado los casos de uso del usuario administrador. Este usuario también tiene los casos de uso del usuario normal pero los amplía y modifica debido a su rol. El cambio sustancial es que el usuario administrador puede mantener tablas a las que un usuario normal no puede acceder y todos los casos de uso relacionados con gestión de documento no requieren del "include" de la comprobación de permisos del documento ya que el administrador no tiene restricciones de acceso en ningún documento y, por tanto, no es necesario comprobarlo.

4.3 Diagramas de clases

El diagrama de clases en UML nos permite representar las relaciones existentes entre todas las clases existentes en el sistema.

La clase es la unidad básica que encapsula la información de un objeto, necesitaremos indicar para cada clase sus atributos y operaciones así como su visibilidad. Todos los objetos que pertenecen a una clase tienen el mismo comportamiento y los mismos atributos. En la aplicación eYellow un documento sería un objeto, así como un usuario o un autor.

Todas las clases que representan a un sistema a su vez se interrelacionan entre ellas generando diferentes tipos de relaciones como herencia, composición, agregación, asociación y uso.

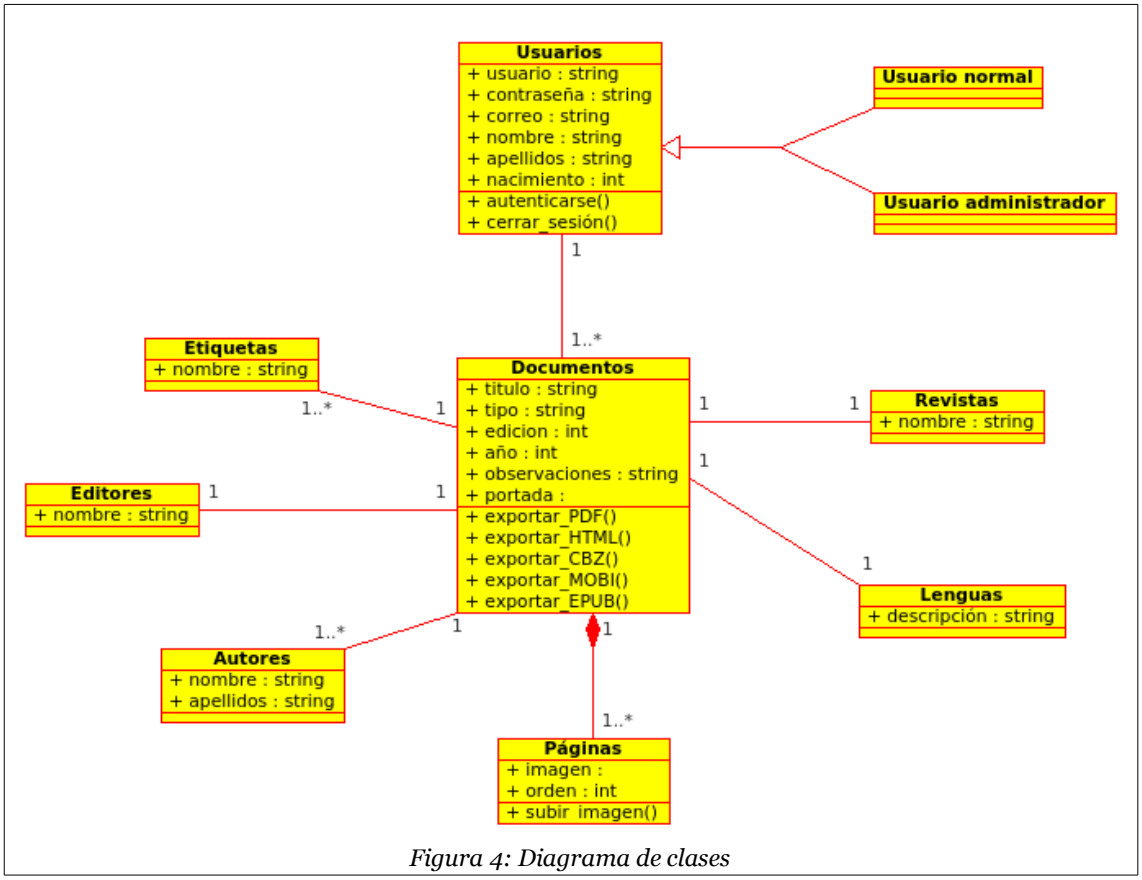


Figura 4: Diagrama de clases

La aplicación eyellow tiene como propósito principal almacenar documentos y exportarlos en un formato estándar por lo que como se puede apreciar en la figura 4 donde se representa el diagrama de clases todo gira en torno a la clase principal de Documentos. Un documento contiene páginas que son imágenes a las que se le ha asignado cierta ordenación. Las clases de etiquetas, revistas, editores, autores y lenguas son clases auxiliares que se relacionan y le añaden información a la clase Documentos. Los usuarios tienen todos los atributos y operaciones en común pero se especializan en dos subclases para distinguir aquellos usuarios que son administradores.

5. Diseño de la aplicación

5.1 Introducción

En el diseño actual de un sistema informático en un entorno web se utiliza la programación por capas o la arquitectura multinivel. La ventaja de utilizar una arquitectura basada en diversos niveles es que a cada nivel se encarga de un conjunto de tareas específicas, de forma que fácilmente se puede reemplazar o modificar el código de uno de los niveles sin necesidad de alterar el resto de niveles.

Cuando se trabaja en proyectos web la división por capas facilita el trabajo por grupos ya que cada equipo se centra en el nivel o capa que tiene asignado y no se preocupa del código existente en el resto de niveles, sólo de la comunicación entre ellos.

La figura 5 sería un ejemplo de la arquitectura web basada en un modelo de 3 niveles:

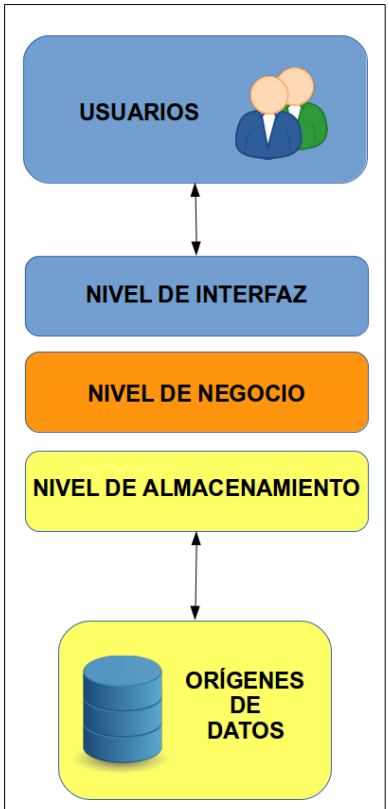


Figura 5: Arquitectura 3 niveles

Las ventajas de realizar un diseño por capas son:

- El desarrollo se divide en varios niveles y pueden ser llevados en paralelo.
- Las aplicaciones son más robustas debido al encapsulamiento de cada capa.

- El mantenimiento y soporte de la aplicación es más sencillo ya que un error o modificación no tiene porque implicar a más de una capa debido a la separación de los roles en capas.

5.2 Nivel de interfaz

El nivel de interfaz o capa de presentación está formado por aquellos elementos con los que el usuario interactúa con la aplicación o sistema: formularios, listados, mensajes de error, ventanas de confirmación, etc.

De este nivel también forma parte todo el diseño visual y usabilidad de la aplicación: formato, colores, disposición de los elementos de la página, columnas, etc.

El nivel de interfaz de la aplicación debe de adaptarse a los estándares de presentación en aplicaciones web haciendo que el usuario se sienta cómodo en la aplicación a pesar de no haber trabajado nunca con ella por lo que la disposición de las diferentes áreas es la habitual de otras aplicaciones web.

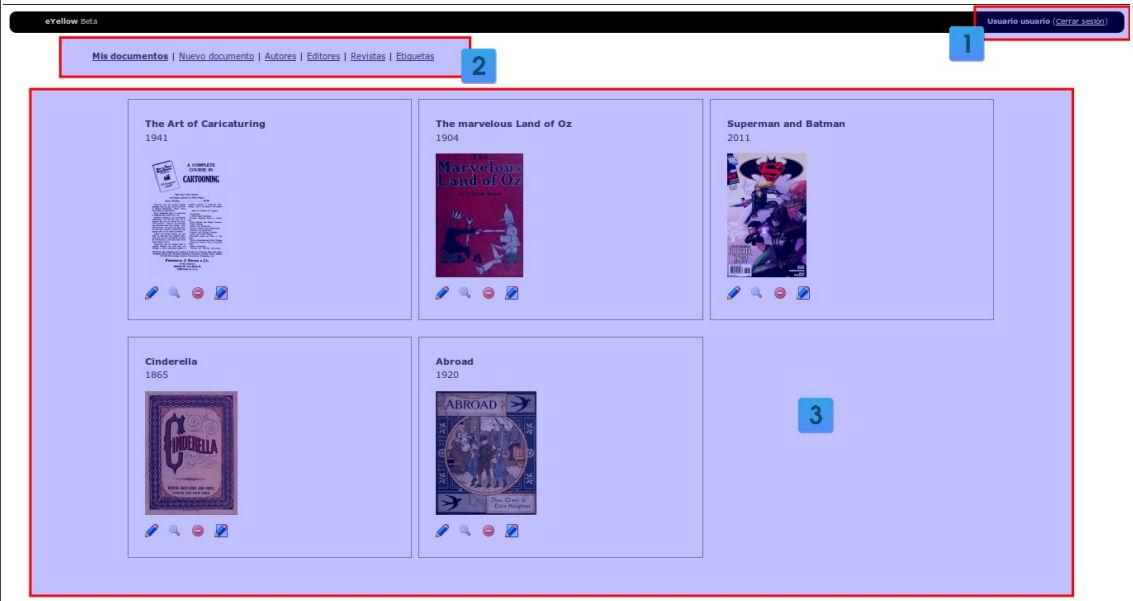


Figura 6: Disposición pantalla visualización documentos

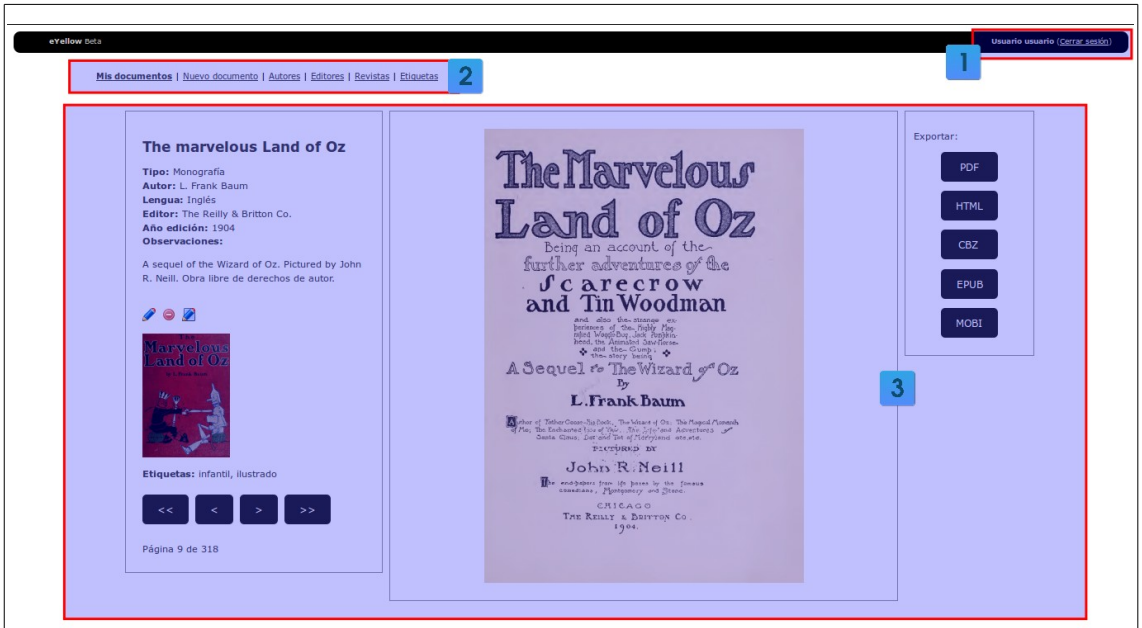


Figura 7: Disposición pantalla consulta documentos

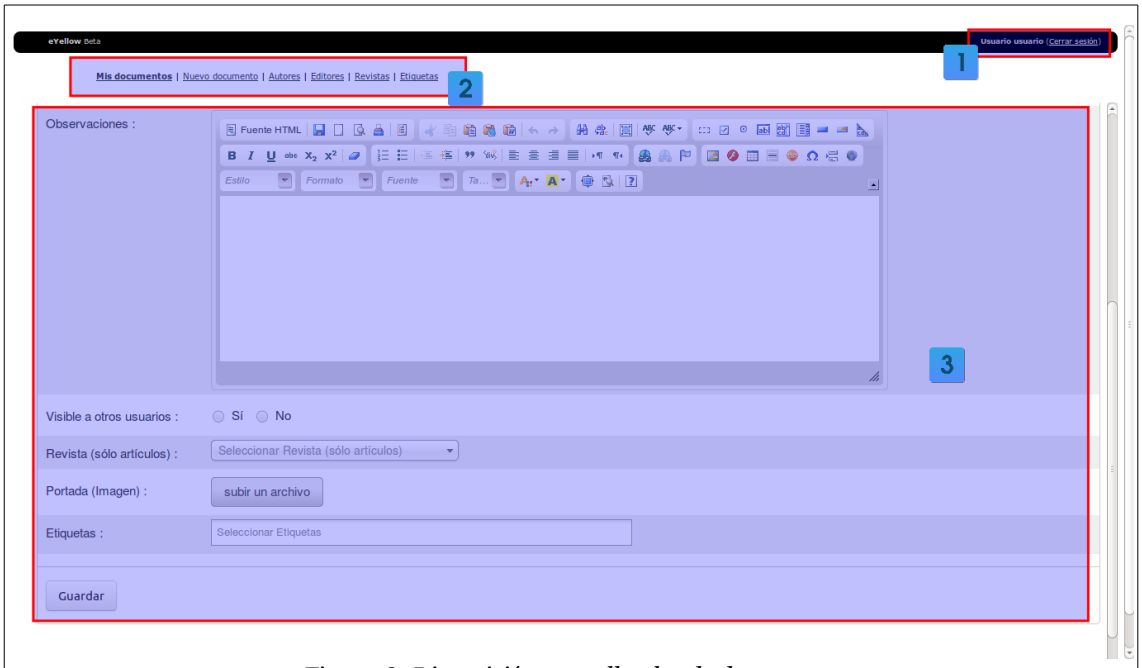


Figura 8: Disposición pantalla alta de documento

En las figuras 6, 7 y 8 podemos observar la disposición de los diferente elementos o bloques en la pantalla de la aplicación:

- El bloque 1 siempre está accesible y muestra el usuario activo en la aplicación así como la posibilidad del cierre de sesión.

- El bloque 2 es una barra de navegación que nos muestra las acciones básicas que puede realizar el usuario. Esté menú horizontal modifica sus elementos en función de la pantalla a la que está accediendo el usuario.

- El bloque 3 es el cuerpo o área de trabajo principal donde se visualizan los diferentes listados, formularios, elementos gráficos, etc. y conforma el bloque principal de la aplicación.

5.3 Nivel de negocio

El nivel de negocio o capa de lógica del negocio es donde se establecen todos los procesos que han de realizarse en la aplicación. Es la capa intermedia y recibe y envía información, peticiones y órdenes al resto de capas.

La capa de negocio incorporará todas las funciones indicadas en el diagrama de clases de la fase de análisis y es, por tanto, la capa que se va a encargar de dirigir todas las operaciones, gestionar el control de la aplicación, recibir peticiones del usuario, solicitar datos al nivel de almacenamiento, procesar y gestionar toda la información.

La comunicación con los otros niveles es fundamental ya que se encarga de realizar las peticiones al nivel de almacenamiento para obtener los datos almacenados y enviarlos al nivel de presentación para mostrarlos de forma adecuada. También se obtienen las peticiones realizadas desde el nivel de presentación por el usuario y las convierte en las órdenes adecuadas para transformar las peticiones en respuestas que pueden incluir accesos al nivel de almacenamiento.

5.4 Nivel de almacenamiento

En el nivel de almacenamiento se guardan los objetos utilizados por el nivel de negocio y se garantiza su persistencia siendo habitual en este nivel recurrir a gestores de base de datos. Cambiar el gestor de base de datos utilizado en una aplicación debería sólo afectar al nivel de almacenamiento y no al resto de niveles.

5.4.1 Diagrama entidad-relación

Para mostrar el diseño conceptual de la base de datos utilizada por eYellow se va a utilizar un diagrama entidad-relación, este diagrama se utiliza para modelado de datos y nos va a permitir representar las entidades relevantes de nuestro sistema de información, además de las relaciones y atributos de las entidades.

El diagrama entidad-relación se basa en 3 elementos gráficos que nos sirven para modelar el sistema de información en el mundo real:

- **Entidades**: Representan un “objeto” o “cosa” del mundo real, también puede ser un objeto con existencia física como una persona. En nuestro sistema de información encontramos diferentes entidades como documentos, usuarios, editores, etc.

  En el diagrama las entidades se representan mediante rectángulos.

- **Atributos**: Son las características que definen o identifican a una entidad. Dado un conjunto de entidades cada una de ellas tiene asignados unos atributos que la diferencian del resto. A partir del conjunto de atributos de una entidad podemos identificar unívocamente a esa entidad. Por ejemplo, los atributos que definen a la entidad usuario en nuestro sistema de información son: nombre de usuario, contraseña, correo electrónico, nombre, apellidos y su fecha de nacimiento.

  En el diagrama los atributos se representan mediante elipses y están asociados a una entidad.

- **Relaciones**: Describen la dependencia entre entidades y, por tanto, para describir una relación es necesario indicar las entidades que están involucradas así como la cardinalidad de la relación.

  En el diagrama las relaciones se representan como rombos.

En la figura 9 tenemos representado el modelo conceptual de nuestro sistema de información en un diagrama entidad-relación. Para aumentar la legibilidad de éste hemos optado por colorear las entidades y los atributos en un color diferente aunque esto no lo indica la norma.

Analizando este diagrama podemos observar:

- La entidad principal que ocupa la posición central es el documento. Todas las entidades que están directamente relacionadas sirven como soporte para almacenar el documento en base de datos.

- La entidad doc_páginas se utiliza para almacenar las imágenes que componen un documento. Un documento contiene varias páginas y la entidad doc_páginas tiene dependencia por existencia de la entidad documentos.

- A partir del análisis se ha establecido que un documento puede tener asignadas varias etiquetas y varios autores por lo que la relación entre estas entidades se ha establecido con una cardinalidad muchos a muchos.

- Un usuario puede tener varios documentos creados pero un documento sólo pertenece a un usuario.

- Las relaciones en la parte inferior del diagrama pdf_d, html_d, cbz_d y epub_d van a ser utilizadas para almacenar valores por defecto de las opciones de exportación que utilicen los usuarios.

  Por ejemplo:

  La entidad pdf_formato contendrá información de todos los posibles formatos a los que puede ser exportado un documento cuando se exporta en PDF: A4, A5, A6, etc. La relación pdf_d con una cardinalidad uno a uno nos guardará la última opción utilizada por un usuario en el proceso de exportación para que aparezcan por defecto esos valores en un nuevo proceso de exportación.

Si un usuario exporta habitualmente en formato epub para un determinado modelo de ebook deberá de seleccionarlo de la lista la primera vez. El resto de veces ya aparecerá automáticamente seleccionado.

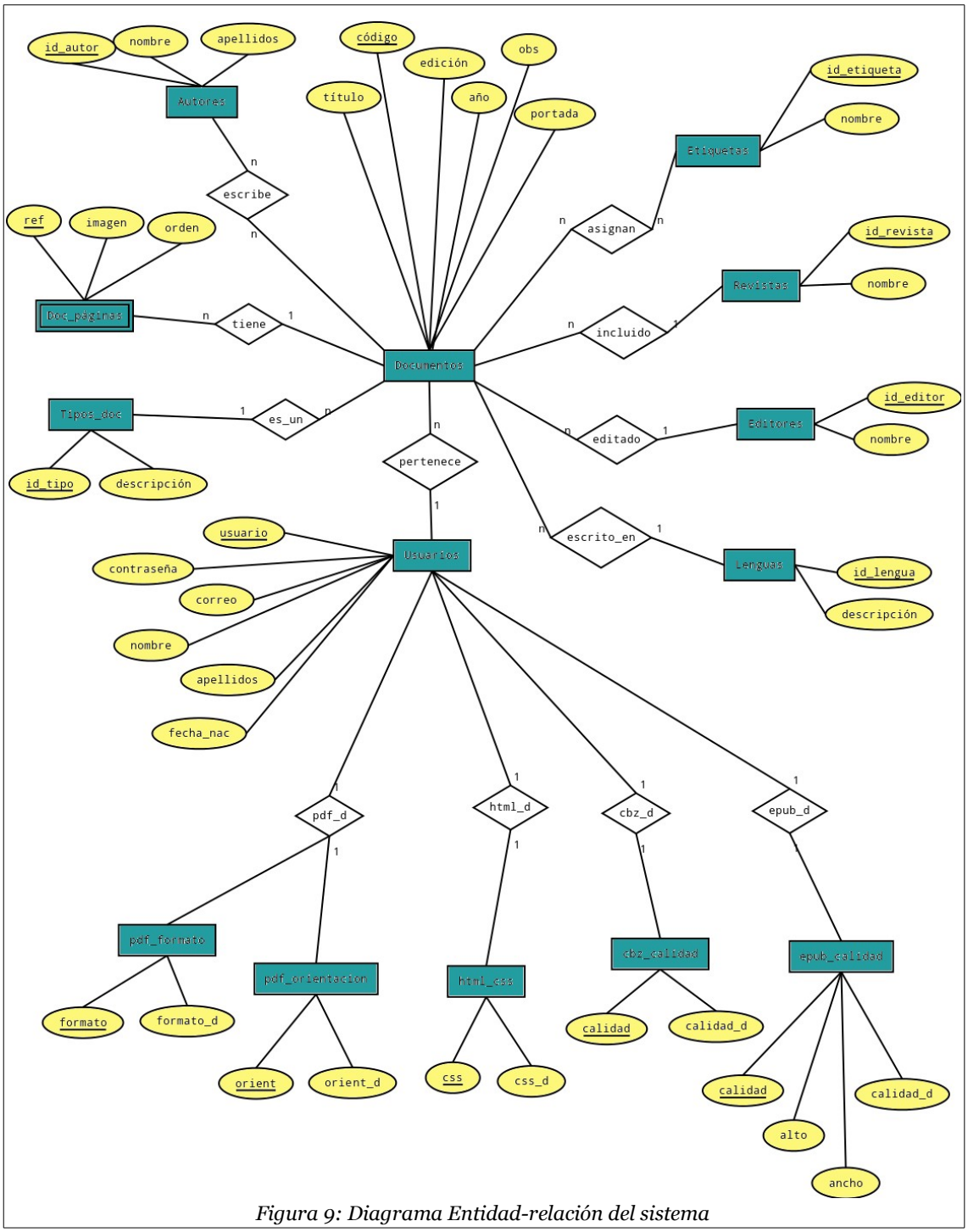


Figura 9: Diagrama Entidad-relación del sistema

5.4.2 Diseño lógico

El diseño lógico en una base de datos relacional consiste en la conversión del esquema conceptual en un esquema lógico.

El esquema lógico que se genera a partir del modelo entidad-relación y tras haber realizado el proceso de normalización da como resultado:

AUTORES (id_autor:entero_largo, nombre:varchar(50), apellido: varchar(80))
CP: {id_autor}
VNN: {nombre,apellido}

CBZ_CALIDAD (calidad: entero_largo, calidad_d: varchar(60))
CPP: {calidad}
VNN: {calidad_d}

CBZ_DEFAULTS (usuario: varchar(20), calidad: entero_largo)
CPP: {usuario}
VNN: {calidad}
CAj: {usuario} → USUARIOS

DOCUMENTOS (id_documento:entero_largo, usuario:varchar(20), tipo:entero_largo, titulo:varchar(120), lengua:varchar(3), editor:entero_largo, edicion:varchar(30), anyo:entero_corto, observaciones:texto, visible:logico, revista:entero_largo, portada:varchar(80))
CPP: {id_documento}
VNN: {usuario, tipo, titulo, lengua, editor, edicion, anyo, observaciones, visible, revista, portada }
CAj: {editor} → EDITORES
CAj: {usuario} → USUARIOS
CAj: {tipo} → TIPOS_DOC
CAj: {revista} → REVISTAS
CAj: {lengua} → LENGUAS

DOC_AUTORES (documento:entero_largo, autor:entero_largo)
CPP: {documento,autor}
CAj: {documento} → DOCUMENTOS
CAj: {autor} → AUTORES

DOC_ETIQUETAS (documento:entero_largo, etiqueta:entero_largo)
CPP: {documento,etiqueta}
CAj: {documento} → DOCUMENTOS
CAj: {etiqueta} → ETIQUETAS

DOC_PAGINAS (ref:entero_largo, documento:entero_largo, orden:entero_largo, imagen:varchar(80))
CPP: {ref}
VNN: {documento}

CAj: {documento} → DOCUMENTOS

EDITORES (id_editor:entero_largo, nombre:varchar(80))
CPP: {id_editor}
VNN: {nombre}

EPUB_CALIDAD (calidad:entero_largo, calidad_d:varchar(60), alto_imagen:entero_largo, ancho_imagen:entero_largo)
CPP: {calidad}
VNN: {calidad_d,alto_imagen,ancho_imagen}

EPUB_DEFAULTS (usuario:varchar(20), calidad:entero_largo)
CPP: {usuario}
VNN: {calidad}
CAj: {usuario} → USUARIOS

ETIQUETAS (id_etiqueta:entero_largo, nombre:varchar(80))
CPP: {id_etiqueta}
VNN: {nombre}

HTML_CSS (css:entero_largo, css_d:varchar(40), fichero:varchar(40))
CPP: {css}
VNN: {css_d,fichero}

HTML_DEFAULTS (usuario:varchar(20), css:entero_largo)
CPP: {usuario}
VNN: {css}
CAj: {css} → HTML_CSS

LENGUAS (id_lengua:varchar(3), descripcion:varchar(80))
CPP: {id_lengua}
VNN: {descripcion}

PDF_DEFAULTS (usuario:varchar(20), formato_pagina:varchar(2), orientacion_pagina:varchar(1))
CPP: {usuario}
VNN: {formato_pagina,orientacion_pagina}
CAj: {formato_pagina} → PDF_FORMATO
CAj: {orientacion_pagina} → PDF_ORIENTACION

PDF_FORMATO (formato_pagina:varchar(2), formato_pagina_d:varchar(40))
CPP: {formato_pagina}
VNN: {formato_pagina_d}

PDF_ORIENTACION (orientacion_pagina:varchar(1), orientacion_pagina_d: varchar(40))
CPP: {orientacion_pagina}
VNN: {orientacion_pagina_d}

REVISTAS (id_revista:entero_largo, nombre:varchar(80))
CPP: {id_revista}
VNN: {nombre}

TIPOS_DOC (id_tipo:entero_largo, descripcion:varchar(80))
CPP: {id_tipo}
VNN: {descripcion}

USUARIOS (usuario:varchar(20), pass:varchar(20), email:varchar(50), nombre:varchar(50), apellidos:varchar(80), fecha_nac:fecha, admin:logico)
CPP: {usuario}
VNN: {pass,email,nombre,apellidos,fecha_nac,admin}

6. Implementación

En este apartado detallaremos las herramientas y tecnologías utilizadas en la implementación de eYellow.

6.1 Introducción

En el proceso de desarrollo de eYellow se han empleado diversas herramientas y tecnologías como son:

- El entorno de desarrollo ha sido el servidor web apache 2.2 con soporte para el lenguaje de programación 5.3 PHP y extensión mysqli bajo Ubuntu Server 12.04.
- Servidor de base de datos Mysql 5.5.31
- Phpmyadmin 3.4.10
- Lenguaje de programación PHP 5.3
- El framework PHP de desarrollo rápido de aplicaciones web Codeigniter.
- Librería Grocery crud para la creación de formularios de mantenimiento de tablas con Codeigniter y PHP.
- Librería Image crud para la creación de mantenimiento de series de imágenes con Codeigniter y PHP.
- Librería TCPDF para la creación de PDFs desde PHP.
- Nivo Slider para la creación de pases de fotografías utilizando Javascript y las librerías jquery.

A continuación detallaremos con más profundidad las tecnologías ya mencionadas y otras menores.

6.2 Tecnologías

eYellow es una aplicación web desarrollada en PHP utilizando el framework Codeigniter 2.1.4, es imprescindible para entender el proceso de desarrollo conocer en profundidad el funcionamiento de Codeigniter y en las tecnologías en que se basa.

Codeigniter es un framework de desarrollo de aplicaciones en PHP. El objetivo de Codeigniter es permitir desarrollar aplicaciones web en PHP de forma más rápida que

si se programa código PHP desde cero, para ello proporciona una serie de librerías necesarias para las tareas habituales en programación web, además de un interface simple y una estructura sencilla para acceder a estas librerías. Codeigniter permite centrarse en el proyecto desarrollado minimizando las líneas de código utilizadas para tareas habituales.

Codeigniter está basado en el patrón de desarrollo Modelo-Vista-Controlador (MVC). MVC es una implementación concreta de un modelo de 3 capas que separa la lógica de la aplicación de la presentación. En la práctica permite que las páginas web contengan el mínimo código PHP al estar separada la presentación de todo el código PHP.

En el patrón de arquitectura de software MVC:

- El **modelo** representa las estructuras de datos. Las clases del modelo contendrán las funciones que ayudan a obtener, insertar y actualizar la información en la base de datos.

- La **vista** es la información que se presenta al usuario final. Una vista será habitualmente una página web, pero en Codeigniter, una vista puede ser también un fragmento de una página como una cabecera o un menú de acciones.

- El **controlador** se utiliza como intermediario entre el modelo y la vista, además tienes otros recursos necesarios para procesar la petición HTTP y generar una página web como resultado.

Codeigniter permite trabajar de una forma cómoda utilizando este patrón de arquitectura en 3 niveles aunque es suficientemente flexible para que en determinados casos se pueda obviar esta estructura. Se utilizan, por tanto, tres clases principales de Codeigniter que implementan el patron MVC denominadas estas clases Models, Views y Controllers.

El objetivo de Codeigniter es obtener el máximo rendimiento, capacidad y flexibilidad utilizando unas librerías ligeras y lo más pequeñas posibles.

Desde un punto de vista vista técnico Codeigniter fue creado con los siguientes objetivos:

- Instanciación dinámica. Los componentes en Codeigniter son cargados y ejecutados sólo cuando son requeridos. El sistema por defecto es muy ligero.

- Acoplamiento ligero. El acoplamiento es el grado en que los componentes de un sistema dependen unos de otros. A menor dependencia de los componentes entre ellos más reutilizable y flexible es el sistema.

- Singularidad de los componentes. La singularidad es el grado en que los componentes tienen una mayor amplitud y propósito. En Codeigniter cada clase y sus funciones son altamente autónomas para conseguir el mayor grado de utilidad.

6.2.1 Capa de presentación

Las tecnologías utilizadas en la capa de presentación son:

- **HTML**: Son las siglas de Hypertext Markup Language (lenguaje de marcado hipertexto) y es el lenguaje utilizando para la elaboración de páginas web de forma predominante. El HTML se utiliza para describir y traducir la estructura e información de una página web en forma de texto. El lenguaje HTML se escribe en forma de etiquetas rodeadas por los signos menor que y mayor que.

  La primera descripción del lenguaje HTML es del año 1991. En la actualidad se utiliza la versión 4 de HTML con incorporaciones y modificaciones de la versión 5 aunque esta versión non esté aún concluida.

  Para editar código HTML puede utilizarse cualquier editor de texto básico aunque existen algunas herramientas especializadas que permiten realizar sitios web mediante WYSIWYG.

  Los elementos de HTML normalmente tiene una etiqueta de inicio (<elemento>) y otra de final (</elemento>). Entre las dos etiquetas se indica el contenido y podemos añadir atributos y valores en la etiqueta de inicio.

```
56      </div>
57              <div id='form_acceso'>
58          <form class='form' action='http://localhost/eyellow/index.php/login
59              <h3>Acceso</h3>
60
61
62              <p>
63              <input type='text' name='usuario' id='usuario' size='25' />
64              <label for='usuario'>Usuario</label>
65              </p>
66
67              <p>
68              <input type='password' name='pass' id='pass' size='25' />
69              <label for='pass'>Contraseña</label>
70              </p>
71
72              <p class="submit">
73              <input type='Submit' value='Entrar' />
74              </p>
75          </form>
76      </div>
77      </div>
```

Figura 10: HTML generado con eYellow

- **CSS**: Las hojas de estilo en cascada son un lenguaje de estilos usado para describir la presentación y el formato de documentos escritos en lenguaje de marcas y utilizando habitualmente en para aplicar el estilo a las páginas web escritas en HTML.

  La información de estilo de una página web suele ir definida en un fichero .css y además suele reutilizarse desde diferentes páginas del sitio web para permitir cambios a nivel global más rápidos y sencillos. Modificar la apariencia completa de un sitio web puede ser tan sencillo como proporcionar un nuevo fichero .css.

CSS tiene un sintaxis sencilla en formato de regla, podemos crear tantas reglas como queramos que serán aplicadas en cascada sobre los elementos de una página web. Cada regla incluye un conjunto de propiedades y valores que definen el formato a aplicar.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<title>eYellow | Acceso usuario</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="shorcut icon" href="http://localhost/eyellow/images/favicon.png" />
<link rel="stylesheet" media="screen" type="text/css" href="http://localhost/eyellow/css/estilo.css" />
<link rel="stylesheet" media="screen" type="text/css" href="http://localhost/eyellow/css/bloques.css" />
<link rel="stylesheet" media="screen" type="text/css" href="http://localhost/eyellow/css/forms.css" />
<script src="http://localhost/eyellow/css/jquery.min.js" type="text/javascript"></script>
```

Figura 11: Referencia al CSS desde código HTML de eYellow

```
body {
font-family: Verdana;
background-color: #fff;
color: #444;
text-decoration: none;
word-spacing: normal;
text-align: left;
letter-spacing: 0;
line-height: 1.5em;
font-size: 0.7em;
}

a:link {
color: #444444;
text-decoration: underline;
}

a:visited {
color: #444444;
text-decoration: underline;
}

a:hover {
color: #000000;
text-decoration: underline;
background-color: #cccccc;
}
```

Figura 12: CSS utilizado en eYellow

- **Javascript**: Es un lenguaje de programación interpretado utilizado en aplicaciones web en el lado del cliente permitiendo mejoras en la interfaz del usuario, la interactividad y las páginas web dinámicas. El soporte de Javascript viene siempre integrado e implementado en el navegador web. La mayoría de los navegadores web actuales son capaces de interpretar el lenguaje Javascript embebido en el lenguaje HTML que descargan del servidor web.

```
<script type="text/javascript">
$(function() {
        $(".confirm").easyconfirm({locale: {
            title: '¿Quiere borrar el documento?',
            text: '¿Está seguro que quiere anular el documento?',
            button: ['No','Sí'],
            closeText: 'Cerrar'
        }});

    });
</script>
</head>
<html>
<body>
```

Figura 13: Código Javascript embebido en HTML en la aplicación eYellow

- **Jquery**: Es un conjunta de librerías javascript que simplifican la forma de interactuar con los documentos HTML, permiten manipular el árbol DOM, controlar eventos, desarrollar animaciones y agregar interacción.

  Jquery es software libre y de código abierto permitiéndose su uso en cualquier proyecto web. La ventaja principal de jquery es ofrecer una gran cantidad de funcionalidades basadas en el lenguaje javascript que de otra forma requeriría mucho más código para implementarlas.

```
<link rel="stylesheet" media="screen" type="text/css" href="http://localhost/eyellow/css/forms.css" />
<script src="http://localhost/eyellow/css/jquery.min.js" type="text/javascript"></script>
<script src="http://localhost/eyellow/css/jquery-ui.min.js"></script>
<link rel="stylesheet" type="text/css" href="http://localhost/eyellow/css/jquery-ui.css" />
<script src="http://localhost/eyellow/css/jquery.easy-confirm-dialog.js"></script>
<script type="text/javascript">
$(function() {
        $(".confirm").easyconfirm({locale: {
            title: '¿Quiere borrar el documento?',
```

Figura 14: Referencia a las librerías Jquery en el código HTML de eYellow

6.2.2 Capa de negocio

Las tecnologías utilizadas en la capa de negocio son:

- **PHP**: Es un lenguaje de programación de uso general utilizado para generar contenido dinámico en el desarrollo web del lado del servidor. Es un proyecto de software libre y podemos encontrar versiones para la mayoría de servidores web y plataformas a coste cero.

  PHP fue creado por Rasmus Lerdorf en 1995 y sus siglas son un acrónimo recursivo que significan PHP Hypertext Pre-processor. En la actualidad la implementación principal del lenguaje es producida por The PHP Group y se considera el estándar de facto.

  PHP tiene gran parecido con otros lenguajes comunes de programación estructurada como C y Perl por lo que permite a un programador crear aplicaciones relativamente complejas con una curva de aprendizaje corta.

El código fuente de un programa escrito en PHP es invisible al navegador web y al cliente ya que es en el servidor donde se realiza la ejecución del código y la generación del código HTML que recibe el cliente en el navegador.

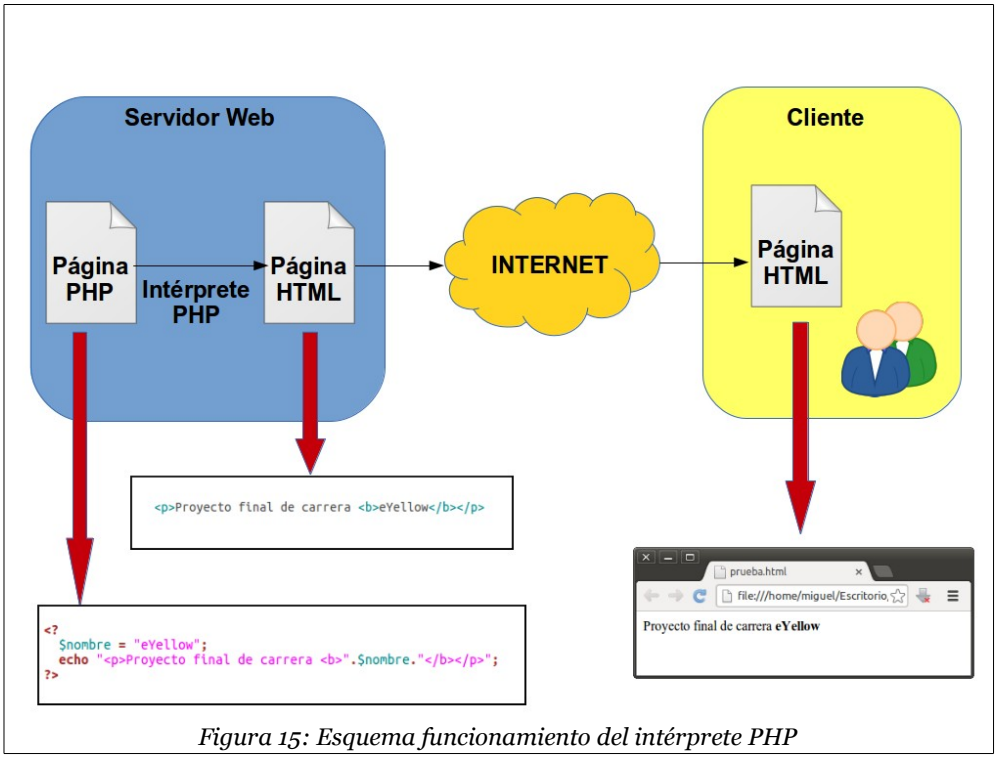


Figura 15: Esquema funcionamiento del intérprete PHP

Al ser uno de los lenguajes de servidor de desarrollo web más utilizados existe mucha documentación, gran cantidad de funciones y librerías, que permiten, por ejemplo, conectarse a un motor de base de datos como Mysql o generar ficheros PDF.

- **Codeigniter**: El núcleo de la aplicación ha sido escrito por completo utilizando este framework ya descrito en el punto 6.2. Aunque el uso de un framework incrementa el tiempo de desarrollo inicial debido a que hay que documentarse en la estructura del framework y las librerías a utilizar, se ve compensado en fases más tardías del desarrollo porque la ayuda para generar aplicaciones web de forma rápida es muy significativa.

  El desarrollo de la aplicación sin utilizar el framework Codeigniter habría generado un código PHP menos estructurado, más ilegible y con un funcionamiento poco claro para desarrolladores noveles en la aplicación. Mantener una aplicación generada con Codeigniter permite tener muy claro dónde se encuentran el código fuente de cada uno de los bloques del programa ya que así lo determina la arquitectura MVC que utiliza Codeigniter.

6.2.3 Capa de datos

Las tecnologías utilizadas en la capa de datos son:

- **Mysql**: Es una sistema de gestión de bases de datos relacional, multihilo y multiusuario. Pertenece a la empresa Oracle y se ofrece como software libre pero con un licenciamiento dual.

  Existen más de 6 millones de instalaciones de Mysql y es uno de los sistemas de gestión de bases de datos más utilizados en entornos web para aplicaciones dinámicas. Por ejemplo, Wikipedia, Flickr y Twitter utilizan mysql como motor de base de datos. La popularidad de Mysql en el entorno web está muy ligado a PHP ya que frecuentemente aparece en combinación con Mysql.

  Existen versiones de Mysql para gran cantidad de plataformas que van desde Linux, OS X, Windows o múltiples versiones de Unix. La instalación de la triada Apache+PHP+Mysql es en la actualidad muy sencilla en cualquier plataforma existiendo incluso instaladores de los 3 programas como Xampp o Wamp.

  La versión actual descargable desde la página oficial de Mysql es la 5.6 aunque el proyecto ha utilizado la versión 5.5 y no debería de existir problemas de compatibilidad entre ambas. Existe una versión compatible del motor de base de datos conocido como MariaDB que es totalmente libre.

6.3 Herramientas

En este punto detallamos algunas de las herramientas o aplicaciones utilizadas para llevar a cabo el desarrollo de eYellow.

6.3.1 Apache

El servidor apache es un servidor web HTTP de código abierto para plataformas Windows, OS X y Unix (linux entre otras). Es el servidor web más utilizado en Internet desde el año 1996, en la actualidad diversas estadísticas le asignan un uso de un 52% de los servidores web en Internet seguido por Internet Information Server de Microsoft con un 20% (Netcraft, 2013).

La arquitectura del servidor apache es modular, el servidor consta de un núcleo y todas las funcionalidades añadidas necesarias para hacer funcionar el servidor web se agregan como módulos. Por ejemplo, el módulo necesario para interpretar código PHP se llama mod_php y lo lleva incorporado la mayoría de las instalaciones de apache.

Apache se usa principalmente para generar páginas web estáticas y dinámicas en Internet.

La versión actual de apache descargable de la web oficial es la 2.4 aunque la versión utilizada en el desarrollo de eYellow ha sido la 2.2 tanto en la versión de desarrollo local como la remota de pruebas.

6.3.2 phpMyadmin

PphMyadmin es una herramienta escrita en PHP que permite administrar en un entorno web un servidor de base de datos Mysql. Es una herramienta muy completa que permite crear, modificar y eliminar bases de datos, tablas y campos, también permite tratar con los datos insertados así como ejecutar sentencias SQL. Como ya lo indica su nombre es una herramienta que requiere ser instalada en un servidor web que soporte PHP y Mysql.

PhpMyadmin es un proyecto de software libre multiplataforma que está traducido a más de 60 lenguas.

La última versión disponible en la web oficial es la 3.5.8, la versión utilizada en la versión local de desarrollo es la 3.4.10 y en la versión remota de pruebas la 3.4.2.

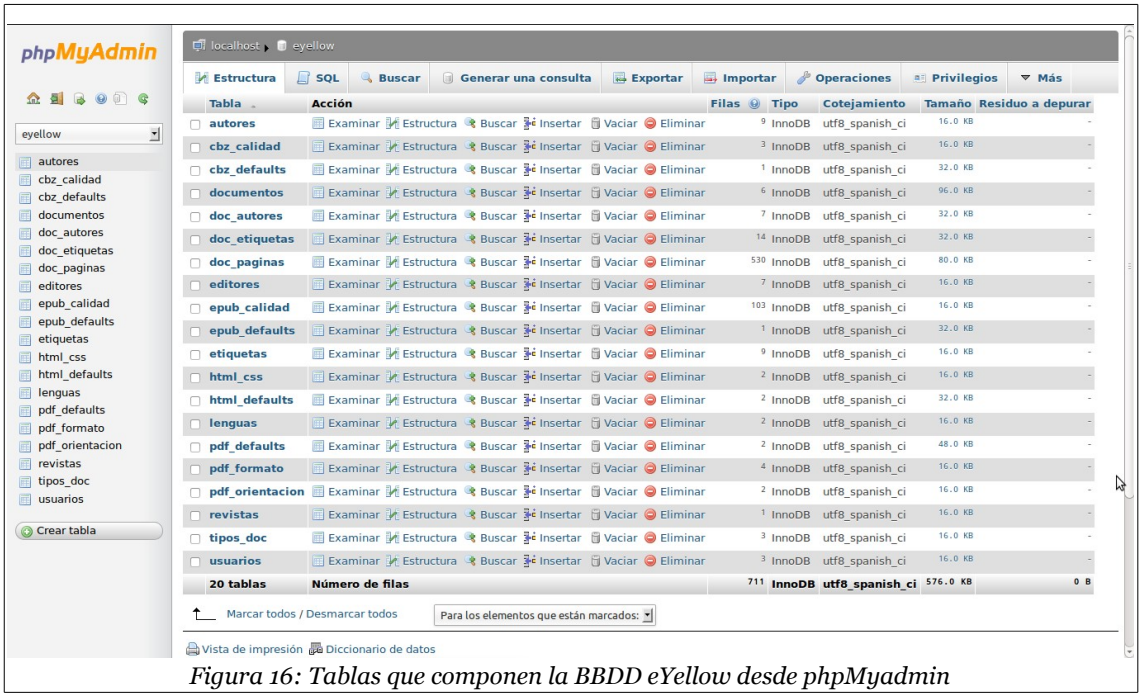


Figura 16: Tablas que componen la BBDD eYellow desde phpMyadmin

Mysql es accesible en su instalación por defecto mediante órdenes que se lanzan desde un entorno de ejecución en modo texto. Phpmyadmin oculta la complejidad de las ordenes del entorno de texto en una aplicación web que tiene todas las opciones accesibles mediante el click del ratón.

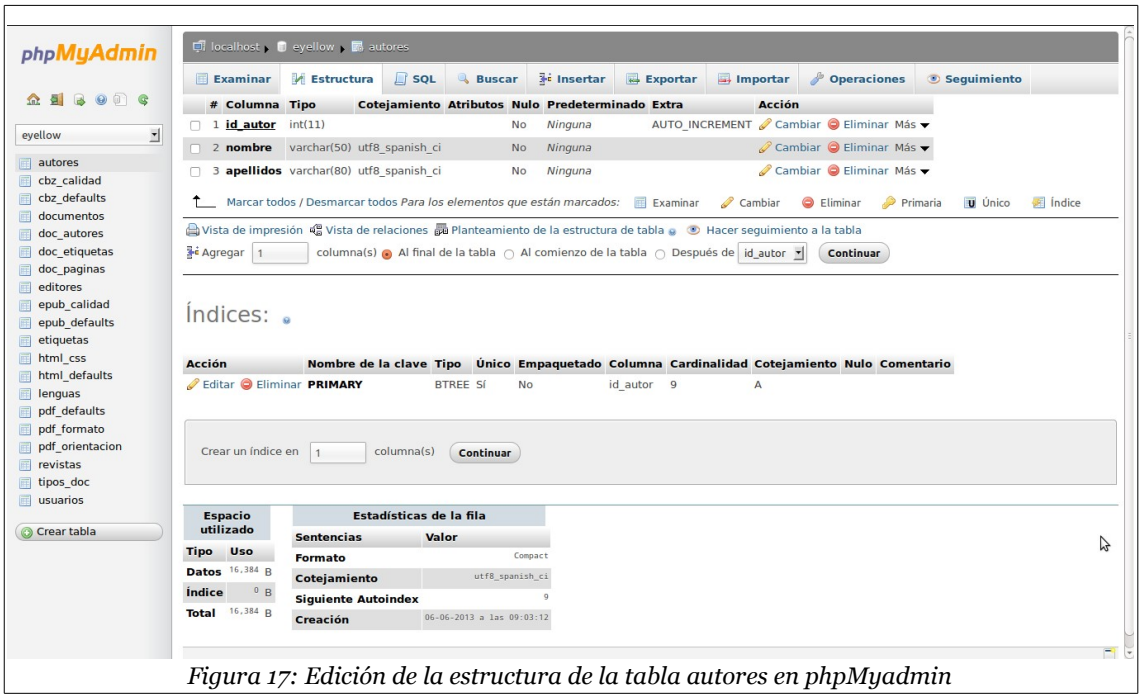


Figura 17: Edición de la estructura de la tabla autores en phpMyadmin

En la figura 16 y 17 podemos ver cómo es muy sencillo a hacer modificaciones de estructura de la base de datos de forma muy sencilla.

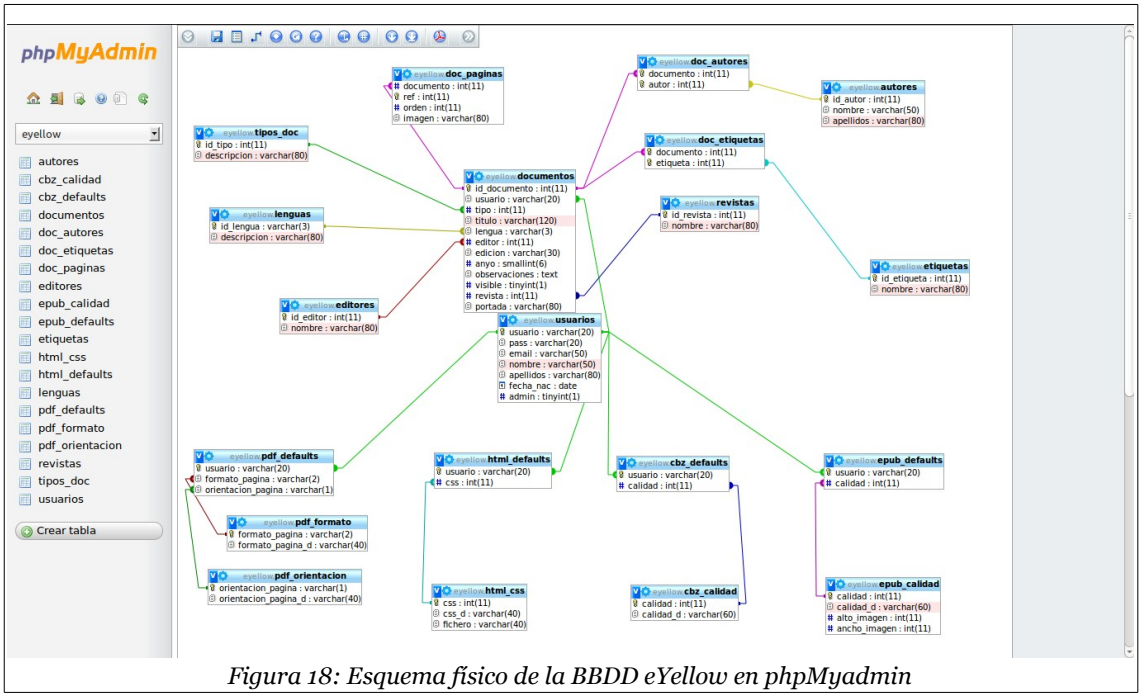


Figura 18: Esquema físico de la BBDD eYellow en phpMyadmin

En la figura 18 tenemos el esquema físico generado en mysql generado a partir del esquema lógico obtenido en la fase de análisis y que vimos en el punto 5.4.2.

6.3.3 Kompozer

Kompozer es una sencilla aplicación de diseño web que permite editar una página web de forma visual así como editando directamente las etiquetas HTML. Es una herramienta fácil de utilizar que permite crear a cualquier usuario crear páginas web sin necesidad de ser un experto en HTML. Kompozer tiene un buen soporte para CSS que evita tener que utilizar otras herramientas para dar el aspecto visual a un sitio web.

Kompozer es un proyecto de software libre con soporte para más de 20 lenguas y que está disponible para las plataformas Windows, OS X y Linux. La versión actual es la 0.8b3 y es la que ha sido utilizada en el entorno de desarrollo.

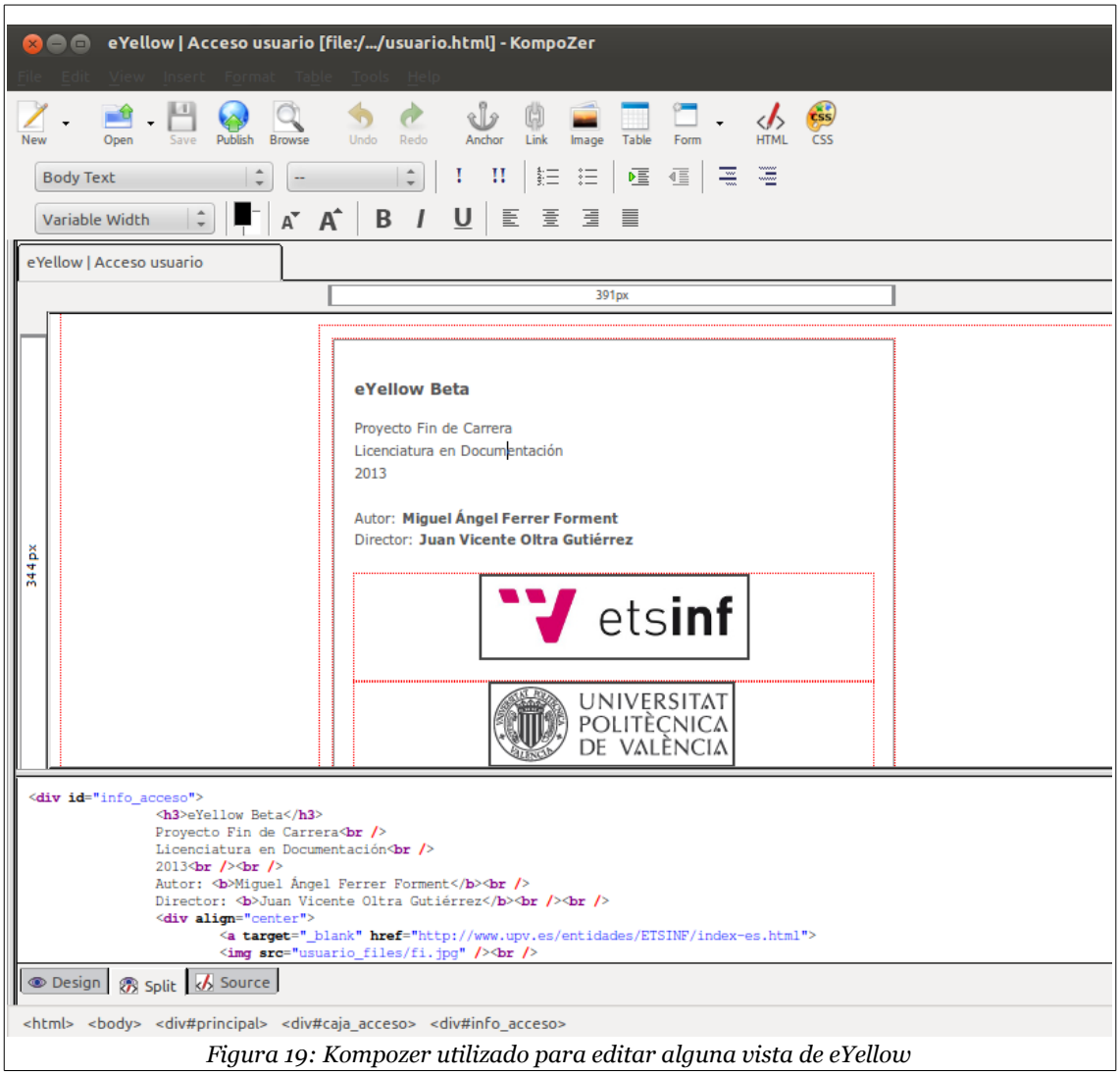


Figura 19: Kompozer utilizado para editar alguna vista de eYellow

En la figura 19 se aprecia como Kompozer permite la edición en modo WYSIWYG (What you see is what you get) y directamente en el código fuente HTML.

6.3.4 Filezilla

Filezilla es un cliente FTP multiplataforma de código abierto y software libre que se ha utilizado para subir los ficheros de la aplicación web al sitio remoto de pruebas.

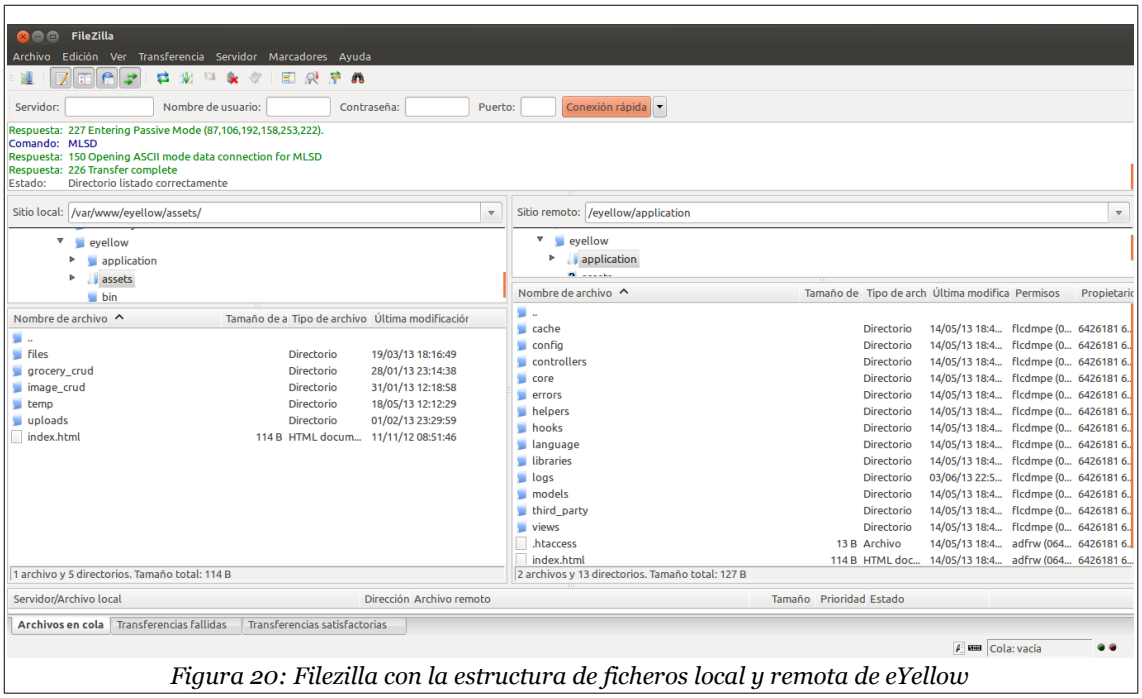


Figura 20: Filezilla con la estructura de ficheros local y remota de eYellow

6.3.5 Gimp

Gimp es un programa libre de edición de imágenes digitales en formato bitmap disponible para Windows, OS X y Linux. Disponible en más de 100 lenguas es el programa más utilizado en entornos libres para editar imágenes. Soporta multitud de formatos gráficos entre ellos aquellos que son más habituales en el desarrollo web: jpg, png y gif.

Para el desarrollo de eYellow se ha utilizado tanto para la creación de diversos elementos gráficos de la aplicación web como botones o logos, así como para la gestión de algunas de las imágenes que componían algunos documentos y que han sido utilizadas de pruebas para generar los diferentes formatos que soportaba la aplicación eYellow.

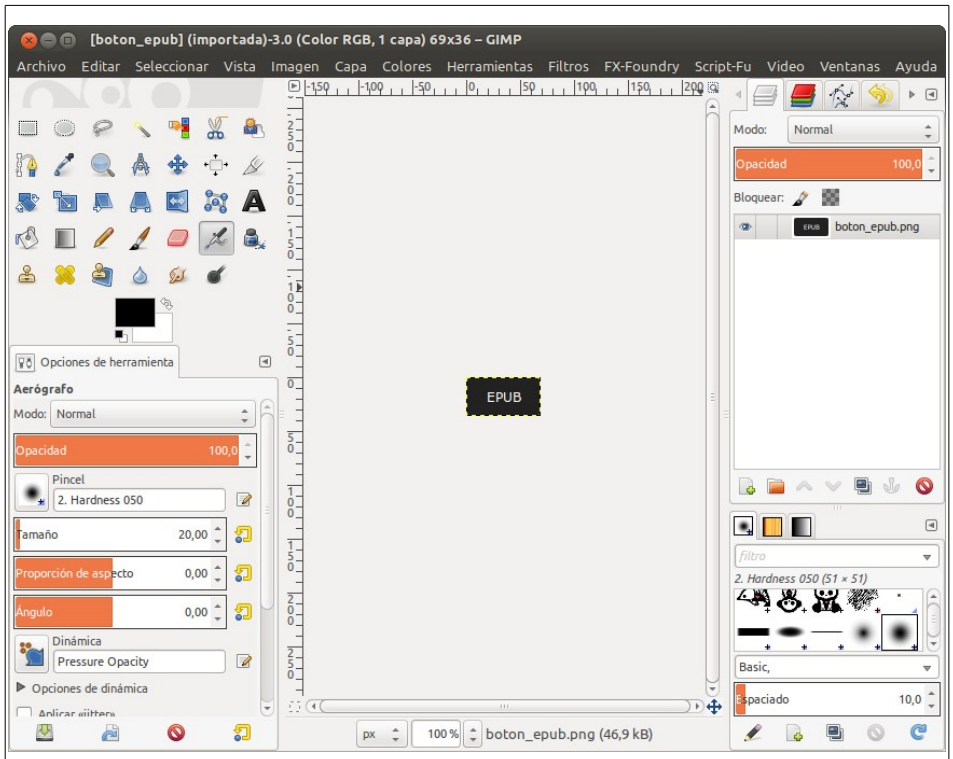


Figura 21: Editando el botón de exportar a epub con Gimp

6.4 Detalles de implementación

En este apartado se detalla minuciosamente la implementación de la aplicación eYellow en las 3 capas indicadas en apartados anteriores. Como ya hemos indicado con anterioridad eYellow ha sido desarrollado utilizando el framework Codeigniter por lo que este framework va a ser el núcleo principal de la aplicación y el objeto de detalle de cómo se ha implementando cada uno de los niveles utilizados en la aplicación web.

Las URLs en codeigniter están diseñadas para ser amigables y a su vez puedan ser entendidas por un motor de búsqueda con el objetivo de indexar de forma eficiente los contenidos de una web creada con codeigniter. La estructura de las URLs generadas con Codeigniter siguen el siguiente esquema:

Figura 22: Ejemplo url eYellow generada con Codeigniter

Cada uno de los segmentos que componen la URL con la arquitectura MVC habitualmente representan:

- **localhost** : Nombre del host o servidor donde está alojado el servidor web.
- **eyellow** : Carpeta del servidor web de la aplicación creada con codeigniter.
- **index.php** : Fichero php que genera todo el procesamiento del resto de segmentos de la URL. Este fichero puede ser omitido si se configura de forma adecuada el servidor web apache.
- **documento** : clase controladora a la que estamos haciendo la llamada.
- **view** : función o método de clase anterior que vamos a utilizar en la llamada.
- **4/1** : Parámetros que se le van a pasar a la función anterior.

De esta forma en la url de ejemplo estamos indicando que queremos utilizar la clase documento con su método view y pasándole los parámetros 4 y 1, en este caso significa visualizar el documento 4 empezando en la página 1.

6.4.1 Implementación de la funcionalidad

El control de la funcionalidad de una aplicación utilizando Codeigniter se implementa mediante clases simples cuyo nombre es utilizado para asociar la URL con la clase a utilizar. Estas clases se denominan en Codeigniter controladores y forman lo que se conoce como la capa de control en la arquitectura MVC.

Las clases controladoras del proyecto de Codeigniter están almacenadas en la carpeta /application/controllers/ de la carpeta del proyecto principal. Analizando las clases existentes en la carpeta y los métodos implementados en cada una de las clases podemos conocer todos las funcionalidades permitidas por la aplicación.

```
public function __construct()
public function do_logout()
private function check_isvalidated()
public function index()
public function delete($id_documento=0)
public function view($id_documento,$id_pagina=1)
public function crud_mant($id_act='add',$id_key='')
public function crud()
function documento_before_insert($post_array)
function documento_before_update($post_array)
function documento_after_insert($post_array,$primary_key)
public function ordena_asc($id_documento)
public function ordena_desc($id_documento)
public function borra_pag($id_documento)
public function export_pdf($id_key='')
public function crud_pdf()
public function gen_pdf($id_documento)
public function export_html($id_key='')
public function crud_html()
public function gen_html($id_documento)
public function export_cbz($id_key='')
public function crud_cbz()
public function gen_cbz($id_documento)
public function export_epub($id_key='')
public function crud_epub()
public function gen_epub($id_documento)
public function export_mobi($id_key='')
```

Figura 23: Métodos de la clase documento

En la figura 23 vemos un listado de todos los métodos de la que es la clase más compleja de la aplicación eYellow encargada de gestionar los documentos. Los nombres de los métodos son bastante descriptivos y se puede averiguar sin problemas cuál es la función de cada uno de esos métodos, por ejemplo, la función delete se encarga de borrar un documento y la función gen_pdf de generar un fichero PDF del documento indicado.

Esta conjunto de funciones de una clase se ve sensiblemente reducido cuando se trata de una clase donde sólo podemos hacer un mantenimiento simple, por ejemplo, la clase lenguas que aparece en la figura 24.

```
public function __construct()
private function check_isvalidated(){
public function crud_mant($id_act='list',$id_key='')
public function crud()
```

Figura 24: Métodos de la clase lenguas

El conjunto de clases que se encargan de gestionar la funcionalidad de la aplicación eYellow son:

- **admin.php** : Clase utilizada para gestionar la distribución de la pantalla del administrador.
- **autores.php** : Mantenimiento de los autores de un documento.

- **doc_paginas.php** : Mantenimiento de las imágenes que componen las páginas de un documento.
- **documento.php** : Clase principal de la aplicación utilizada para el mantenimiento, listado, visualización y listado de los documentos.
- **editores.php** : Mantenimiento del editor de un documento.
- **etiquetas.php** : Mantenimiento de las etiquetas que pueden asignarse a un documento.
- **lenguas.php** : Mantenimiento de las lenguas de un documento.
- **login.php** : Gestión de acceso y de las sesiones de los usuarios.
- **revistas.php** : Mantenimiento de las revistas de un documento.
- **tipos_doc.php** : Mantenimiento de las tipologías de un documento.
- **usuarios.php** : Mantenimiento de los usuarios de la aplicación.

En la implementación de eYellow se han utilizado unas librerías añadidas al núcleo de Codeigniter para generar de forma rápida determinados aspectos de la aplicación:

Grocery crud es una librería que permite crear un mantenimiento básico de una tabla de forma rápida y automática utilizando unas pocas líneas de código, para mantenimientos más complejos Grocery crud incluye una API completa donde podemos modificar cualquier aspecto que se nos ocurra en un mantenimiento de altas, bajas y modificaciones de una tablas.

Entre las características de Grocery Crud tenemos:

- Mantenimiento en tabla con paginación, ordenación y búsqueda de forma automática.
- Creación automática y configurable de los campos de input de un formulario de alta de un nuevo registro.
- Gestión de las relaciones 1:1, 1:n y n:n en las tablas de una base de datos.
- Fácil cambio de apariencia modificando unos pocos ficheros CSS.
- Posibilidad de generar reglas propias de validación.
- Posibilidad de desactivar operaciones del mantenimiento desde el código, por ejemplo, un usuario que pueda consultar un documento pero no editarlo o borrarlo.
- Impresión y exportación del contenido de la tabla en un click.
- Soporte multilingüe en más de 25 lenguas.

Figura 25: Mantenimiento en tabla de autores creado con Grocery Crud

Figura 26: Mantenimiento básico en formulario de autores creado con Grocery Crud

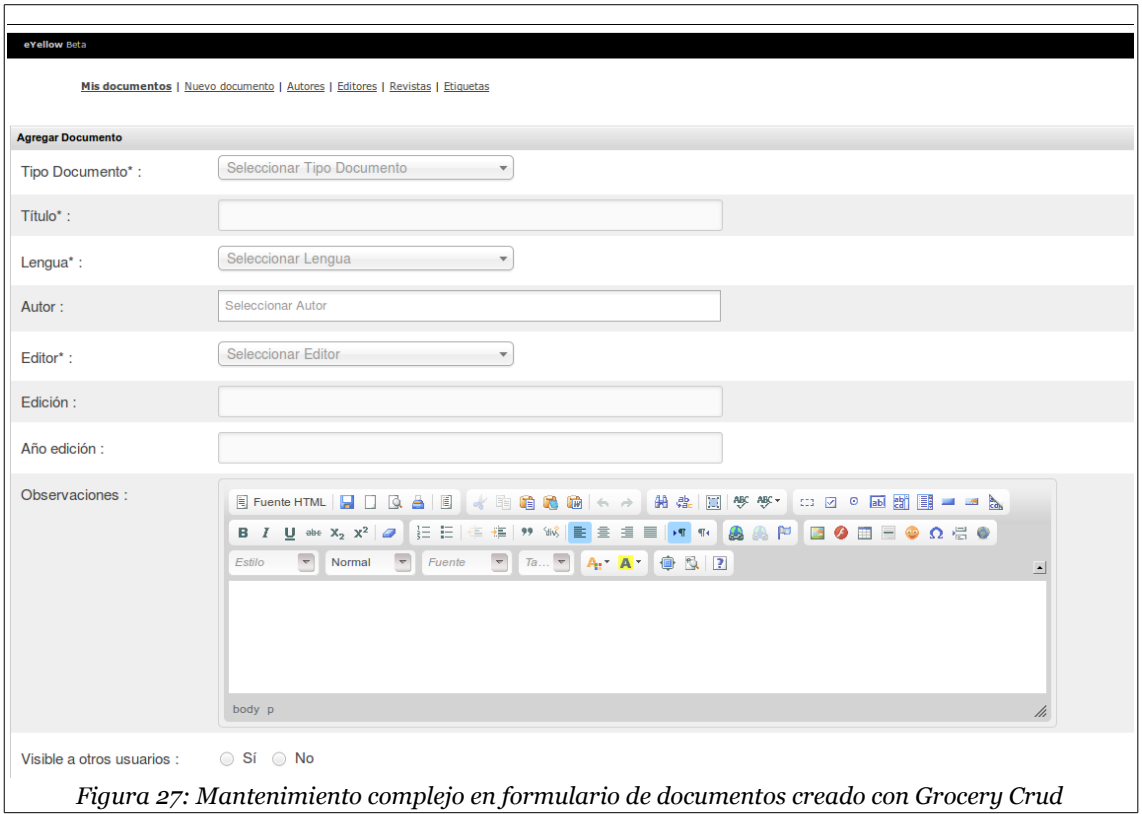


Figura 27: Mantenimiento complejo en formulario de documentos creado con Grocery Crud

Image crud es una librería inspirada en Grocery crud y que utiliza la misma filosofía que se utiliza para el mantenimiento de una galería de imágenes de forma automática. Image crud genera un mantenimiento múltiple de imágenes mediante las librerías jquery de javascript.

Figura 28: Mantenimiento de las páginas de un documento generado con las librerías Image Crud

Al ser unas librerías en PHP y disponer el código fuente de éstas ha habido que hacer ciertos cambios para adaptarlos al uso de los formatos de imagen para la aplicación eYellow, por una parte el mantenimiento de las imágenes se hacía en formato apaisado que no encajaba bien con el formato habitual de las páginas de un documento. Además se habían limitado las librerías para trabajar en un formato máximo de imagen de 1024x768 siendo éstas unas dimensiones insuficientes para el tratamiento de documentos escaneados a alta calidad.

Tcpdf es una librería PHP que incluye una clase general que permite crear ficheros PDF desde PHP y que pueden integrarse con relativa facilidad en Codeigniter.

Tcpdf incluye entres otras características:

- No requiere de otras librerías externas para generar el fichero PDF.
- Soporta diferentes tipos de página, codificaciones y tipos de fuente.
- Admite la inclusión de imágenes en diferentes formatos y códigos de barras.
- Admite la inclusión de encabezados y pies de página.

```
$query_doc=$this->documento_model->get_documentos($id_documento);
$query_pdf_def=$this->pdf_defaults->get_pdf_defaults();

// Crear PDF
$pdf = new Pdf($query_pdf_def['orientacion_pagina'], 'mm', $query_pdf_def['formato_pagina'], true, 'UTF-8', false);

// Metadatos
$pdf->SetTitle($query_doc['titulo']);
$pdf->SetAuthor($query_doc['autor_d']);
$pdf->SetCreator('eYellow');
$pdf->SetKeywords($query_doc['etiqueta_d']);

// Margenes
$pdf->setPrintHeader(false);
$pdf->setPrintFooter(false);
$pdf->SetMargins(0, 0, 0);
$pdf->SetAutoPageBreak(false, 0);

$pdf->setImageScale(PDF_IMAGE_SCALE_RATIO);

$query=$this->pagina_model->get_paginas($id_documento);

foreach ($query->result() as $pagina):

        $pdf->AddPage($query_pdf_def['orientacion_pagina'],$query_pdf_def['formato_pagina']);
```

Figura 29: Ejemplo de llamadas a la librería TCPDF desde la clase documento para generar un PDF

6.4.2 Implementación del interfaz

La capa de presentación se gestiona en Codeigniter mediante el uso de vistas. Una vista es simplemente una página web o un bloque de una página web como podría ser un menú, un encabezado o un pie de página. Por tanto, las vistas en Codeigniter tienen la flexibilidad de poder contener otras vistas si fuera necesario utilizar esta jerarquía.

Las vistas nunca son llamadas directamente sino que son cargadas desde los métodos de las clases controladoras, recordar la arquitectura MVC donde el controlador es el responsable de la llamada a una vista u a otra.

Las clases de las vistas del proyecto de Codeigniter están almacenadas en la carpeta /application/views/ de la carpeta del proyecto principal. Podemos organizar las vistas en subcarpetas para tenerlas mejor clasificadas.

```
public function index($msg = NULL){
        $data['msg']=$msg;
        $data['title']='Acceso usuario';
        $this->load->view('common/header',$data);
        $this->load->view('common/menu_usuario',$data);

        $data['caja_info']=$this->load->view('login_info',$data,true);
        $data['caja_login']=$this->load->view('login_view',$data,true);
        $this->load->view('login_caja',$data);

        $this->load->view('common/footer',$data);
}
```

Figura 30: Ejemplo de carga de vistas en Codeigniter desde un controlador

La figura 30 es un ejemplo del uso de las vistas desde un controlador. La página web que se genera es la suma de los diferentes bloques de las vistas que se llaman desde el controlador. La ventaja de utilizar esta división de una página web en diferentes vistas es que los bloques pueden ser reutilizados desde la generación de otra página web. Por lo que, por ejemplo, sólo hemos generado una vista que incluye el bloque para cerrar

sesión pero esta vista se llama desde prácticamente todos los controladores que generan una página web en la aplicación eYellow.

```
<div id="menu_usuario">
        <div id="musuario1">
        <b>eYellow</b> Beta
        </div>
        <div id="musuario2">
<? if ($this->session->userdata('validated')) {
        echo "<b>".$this->session->userdata('nombre')." ".$this->session->userdata('apellidos')."    </b>";
        if ($this->session->userdata('admin')==1) {
                echo " [Administrador] ";
        };
        echo '(<a href="'.base_url().'index.php/documento/do_logout/">Cerrar sesión</a>)';
} else {
        echo "<b>No estás en el sistema     </b>";
        echo '(<a href="'.base_url().'index.php/login/">Entrar</a>)';
}
?>
        </div>
</div>
```

Figura 31: Vista del bloque del cierre de sesión

6.4.3 Implementación de la gestión de datos

La gestión del acceso a los datos en Codeigniter que se realizar en las clases denominadas Modelos. Estas clases están pensadas para trabajar con un motor de base de datos como mySql aunque pueden utilizar muchos de los gestores de bases de datos más conocidos.

Las clases de los modelos del proyecto de Codeigniter están almacenadas en la carpeta /application/models/ de la carpeta del proyecto principal, al igual que las vistas pueden organizarse en subcarpetas si es necesario.

```
<?php
class Documento_model extends CI_Model {

public function __construct()
        {
                $this->load->database();
        }

public function get_documentos($id_documento = FALSE)
{
        //Acceso a BBD
}

public function ordena_paginas($id_documento = FALSE,$ascendente = TRUE)
{
        //Acceso a BBD
}

public function borra_paginas($id_documento = FALSE)
{
        //Acceso a BBD
}

public function chequea_paginas($id_documento = FALSE)
{
        //Acceso a BBD
}

private function _get_paginas0($id_documento)
{
        //Acceso a BBD

private function _get_num_paginas($id_documento)
{
        //Acceso a BBD
}
```

Figura 32: Estructura de la clase Documento_model en eYellow

La estructura de una clase típica de un modelo en Codeigniter podemos verla esquematizada en la figura 32: el constructor carga la configuración del acceso a la base de datos y los métodos públicos contienen código para cargar, consultar, editar y eliminar datos utilizando SQL o utilizando las funciones que proporciona la clase Database.

En la figura 33 tenemos un ejemplo de código asociado a un método de un modelo. El código utiliza SQL para acceder a la base de datos y es válido para cualquier motor de base de datos relacional, en nuestra aplicación se ha utilizado Mysql pero sería relativamente fácil, por ejemplo, cambiar el motor de base de datos a Postgresql u Oracle sin tocar prácticamente el código de la aplicación.

```
private function _get_autor_d($id_documento)
{
        $sql='select concat_ws(" ",a.nombre,a.apellidos) nombre_completo from doc_autores d,'.
             'autores a where d.autor=a.id_autor and d.documento='.$id_documento;

        $query = $this->db->query($sql);

        if ($query->num_rows() > 0)
        {
                $nombres='';
                foreach ($query->result() as $row)
                {
                        if ($nombres=='') {
                                $nombres=$row->nombre_completo;
                        } else {
                                $nombres=$nombres.', '.$row->nombre_completo;
                        };
                }
                return $nombres;
        } else {
                return 'Sin especificar';
        }

}
```

Figura 33: Ejemplo de un método de una clase modelo en eYellow

Para utilizar los modelos desde las clases controladores debemos de realizar primero una carga de éste como se indica en la figura 34.

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class Documento extends CI_Controller {

        public function __construct()
        {
                parent::__construct();
                $this->check_isvalidated();
                $this->load->library('grocery_CRUD');
                $this->load->model('documento_model');
                $this->load->model('pagina_model');
                $this->load->helper('file');
        }

        public function do_logout()
        {
                $this->session->sess_destroy();
                redirect('login');
        }

        private function check_isvalidated(){
```

Figura 34: Carga de un modelo desde una clase controladora

Los métodos de la clase controladora pueden utilizar directamente los métodos públicos de la clase modelo para acceder a los datos almacenados en la base de datos. Ver en la figura 35 un ejemplo de código de una clase de control realizando llamadas a la clase modelo.

```
public function view($id_documento,$id_pagina=1)
{
        $data['documento'] = $this->documento_model->get_documentos($id_documento);
        if (empty($data['documento']))
        {
                show_404();
        }

        $this->documento_model->chequea_paginas($id_documento);

        $datos_pagina['doc_paginas'] = $this->pagina_model->get_pagina($id_documento,$id_pagina);

        $data['title'] = $data['documento']['titulo'];
        $data['pagina'] = $id_pagina;

        $this->load->view('common/header',$data);
        $this->load->view('common/menu_usuario',$data);

        $this->load->view('common/menu_aplicacion',$data);

        $this->load->view('documentos/view', $data);

        if (!empty($datos_pagina['doc_paginas']))
        {
                $datos_pagina['ultima']=$data['documento']['ultima'];
```

Figura 35: Llamada a métodos del modelo desde métodos del controlador

Los modelos utilizados en la aplicación eYellow son:

- **login_model.php** : Comprueba la validación del usuario y contraseña introducidos en el inicio de sesión.
- **documento_model.php** : Utilizado para la gestionar los documentos en base de datos.
- **pagina_model.php** : Utilizado para la gestionar las páginas de los documentos en base de datos.
- **grocery_crud_model.php** : Modelo que forma parte de la librería grocery_crud y se utiliza para el acceso a base de datos de la tablas mantenidas por la librería.
- **cbz_defaults.php** : Utilizado para gestionar la selección y los valores por defecto en el diálogo de exportación de un documento a cbz.
- **pdf_defaults.php** : Utilizado para gestionar la selección y los valores por defecto en el diálogo de exportación de un documento a pdf.
- **epub_defaults.php** : Utilizado para gestionar la selección y los valores por defecto en el diálogo de exportación de un documento a epub.
- **html_defaults.php** : Utilizado para gestionar la selección y los valores por defecto en el diálogo de exportación de un documento a html.

Por último una aclaración acerca de la implementación del formato físico de la base de datos de eYellow para el almacenamiento de imágenes. Los documentos almacenados con eYellow contienen como elemento fundamental una serie de imágenes que contienen las páginas que conforman el documento. Para almacenar las imágenes de forma persistente en un sistema gestor de base de datos y una aplicación web tenemos dos opciones:

1. Almacenar la imagen con un campo de tipo binario o blob en la tabla que corresponda.
2. Almacenar la imagen en una carpeta del sistema de ficheros y guardar en un campo de tipo texto el nombre o ruta relativa al fichero.

La opción elegida para la aplicación eYellow ha sido la segunda que aunque obliga a gestionar de forma manual la anulación y o modificación de los ficheros editados tiene cierta ventajas para una aplicación que gestiona un volumen de datos como eYellow:

- El acceso a los ficheros es rápido y sencillo utilizando una estructura de carpetas para guardar las portadas, páginas y miniaturas que utiliza la página web.
- No se sobrecarga la base de datos tratando campos binarios que pueden ser de gran tamaño y hacen que el tamaño de las tablas crezcan rápidamente.
- Podemos obtener directamente las imágenes almacenadas desde el servidor web sin necesidad de recurrir a código PHP.
- Desde PHP es más lento el acceso a una imagen en base de datos que a un fichero en disco.
- La mayoría de las librerías que gestionan imágenes almacenan la imagen en disco utilizando el sistema de ficheros y la referencia en un campo de la tabla. Por ejemplo, Grocery crud con su librería image-crud ha optado por esta opción.

7. Evaluación y pruebas

7.1 Introducción

Una vez finalizada la aplicación se han llevado a cabo pruebas y comprobaciones sobre ésta. Por una parte se debe verificar que la aplicación tiene un funcionamiento correcto y libre de errores, que utiliza los estándares de generación de páginas web y va a poder visualizarse y funcionar en los navegadores actuales de Internet con diferentes resoluciones de pantalla. También se debe de comprobar que no existen enlaces rotos o que llevan a páginas que generan errores.

Por último y una vez generado un juego de pruebas con varios documentos, también debemos de evaluar el producto multiformato que genera la aplicación, por lo que deberemos de generar utilizando diferentes parametrizaciones los diferentes formatos a los que permite convertir los documentos y comprobar en plataformas reales que son válidos y pueden visualizarse sin problemas.

7.2 Pruebas aplicación web

Se han realizado diversas pruebas sobre la aplicación web para comprobar su corrección y adaptación a los estándares.

7.2.1 Validación CSS

Para validad el CSS utilizado en la aplicación eYellow se ha utilizado una herramienta online del W3C. El World Wide Web Consortium (W3C) es una comunidad internacional donde las organizaciones miembros, sus trabajadores y el público en general trabajan juntos para desarrollar los estándares de la web. Además de las especificaciones completas de las tecnologías relacionadas con la web: HTML, CSS, XML, SVG, etc. proporcionan una serie de herramientas de validación para estas tecnologías.

El servicio de validación de CSS de W3C es un software libre creado por W3C que ayuda a los diseñadores y desarrolladores web a validar hojas de estilo en cascada (CSS). W3C proporciona una herramienta online gratuita[1] para realizar la validación y es la que hemos utilizado. Esta herramienta permite comparar las hojas de estilo utilizadas en la aplicación con las especificaciones CSS, ayuda a encontrar errores comunes, errores de

1 http://jigsaw.w3.org/css-validator/

sintaxis o usos no válidos de CSS y errores que pueden afectar a la usabilidad de la página web.

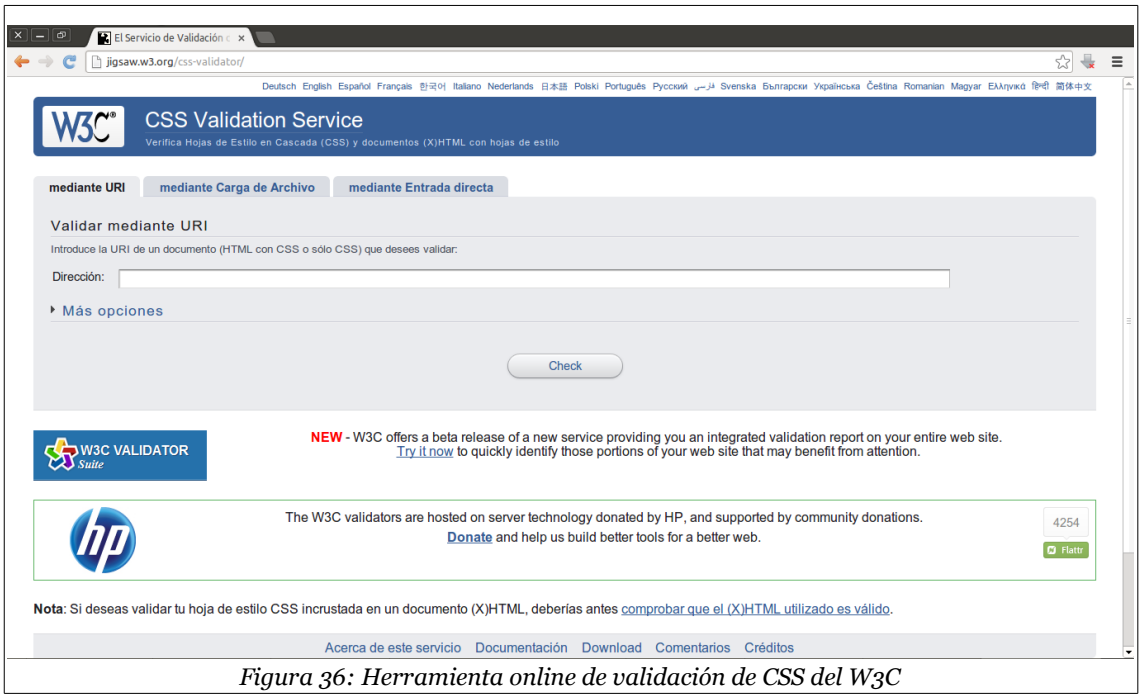


Figura 36: Herramienta online de validación de CSS del W3C

Tras realizar la validación del CSS de la aplicación eYellow se obtiene como resultado que la aplicación web eYellow se considera que contiene CSS versión 3 válido como podemos observar en la figura 37.

Figura 37: Validación CSS de eYellow

7.2.2 Validación en diferentes navegadores

Durante el desarrollo de eYellow se han utilizado de forma habitual los navegadores Firefox y Google Chrome para realizar las pruebas de funcionamiento de la aplicación dando resultados satisfactorios.

Firefox y Chrome se encuentran entre los navegadores más utilizados pero para completar las pruebas deberíamos realizarlas también con los navegadores Internet Explorer, Safari y Opera.

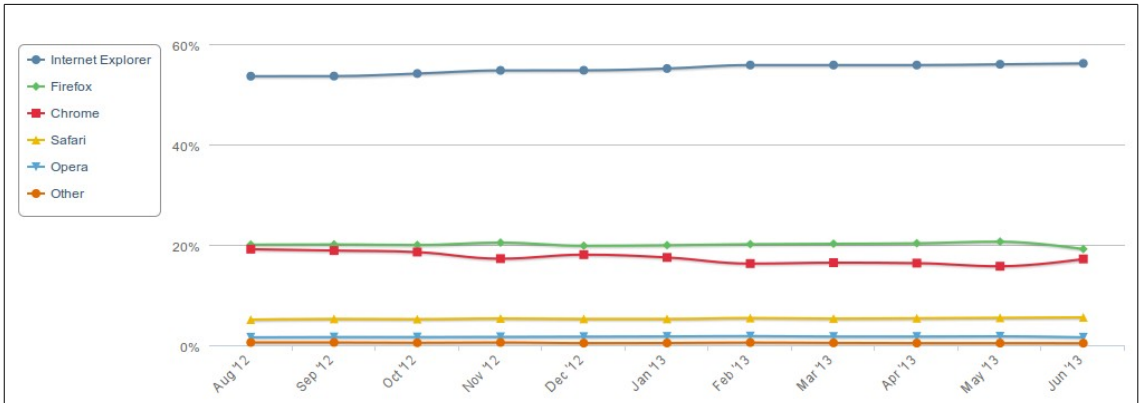


Figura 38: Navegadores de Internet más utilizados

A junio de 2013 las cuotas de uso de navegadores de Internet son (Netmarketshare, 2013):

- Internet explorer : 56,15%
- Firefox : 19,15%
- Chrome : 17,17%
- Safari : 5,55%
- Opera : 1,58%

Entre estos cinco navegadores suman más del 99% de uso por lo que si hacemos pruebas con estos navegadores es una prueba suficiente para comprobar la funcionalidad de la aplicación.

Para hacer las pruebas hemos utilizado el sitio web Browser Sandbox de spoon.net[1] que permite ejecutar cualquier navegador actual sin necesidad de descargarlo. Esta aplicación permite testear varios navegadores y diferentes versiones de forma muy sencilla, además no son necesarios instalarlos porque son ejecutados en un entorno virtualizado y sirven para testear tanto aplicaciones locales como aplicaciones en la red.

1 https://spoon.net/browsers/

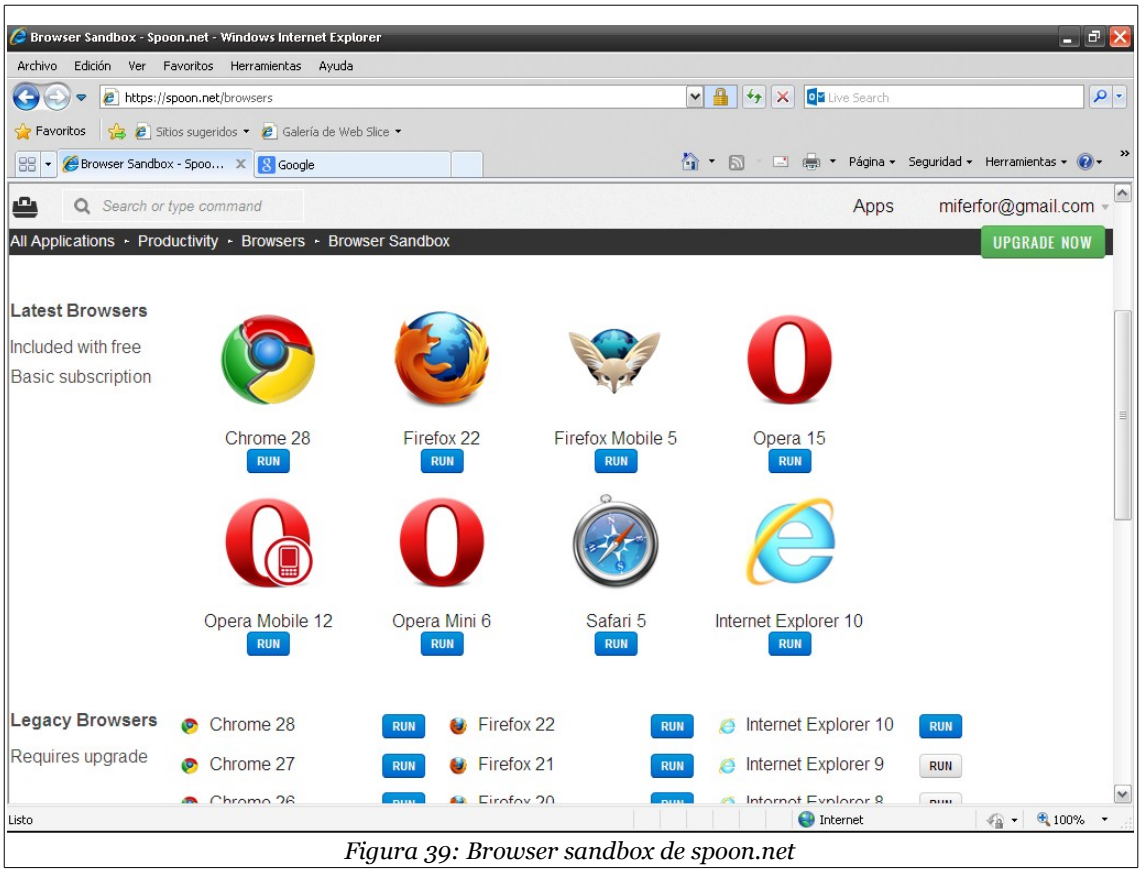


Figura 39: Browser sandbox de spoon.net

Las versiones de los navegadores utilizadas para testear eYellow han sido:

- Chrome 28
- Firefox 22
- Opera 15
- Safari 5
- Internet Explorer 10

En las siguientes figuras vemos ejemplos de ejecución de eYellow con los diferentes navegadores. La prueba que se realizó para comprobar el funcionamiento consistió en la creación de un documento nuevo desde cero con 10 páginas de imágenes y la generación de un par de formatos en cada navegador.

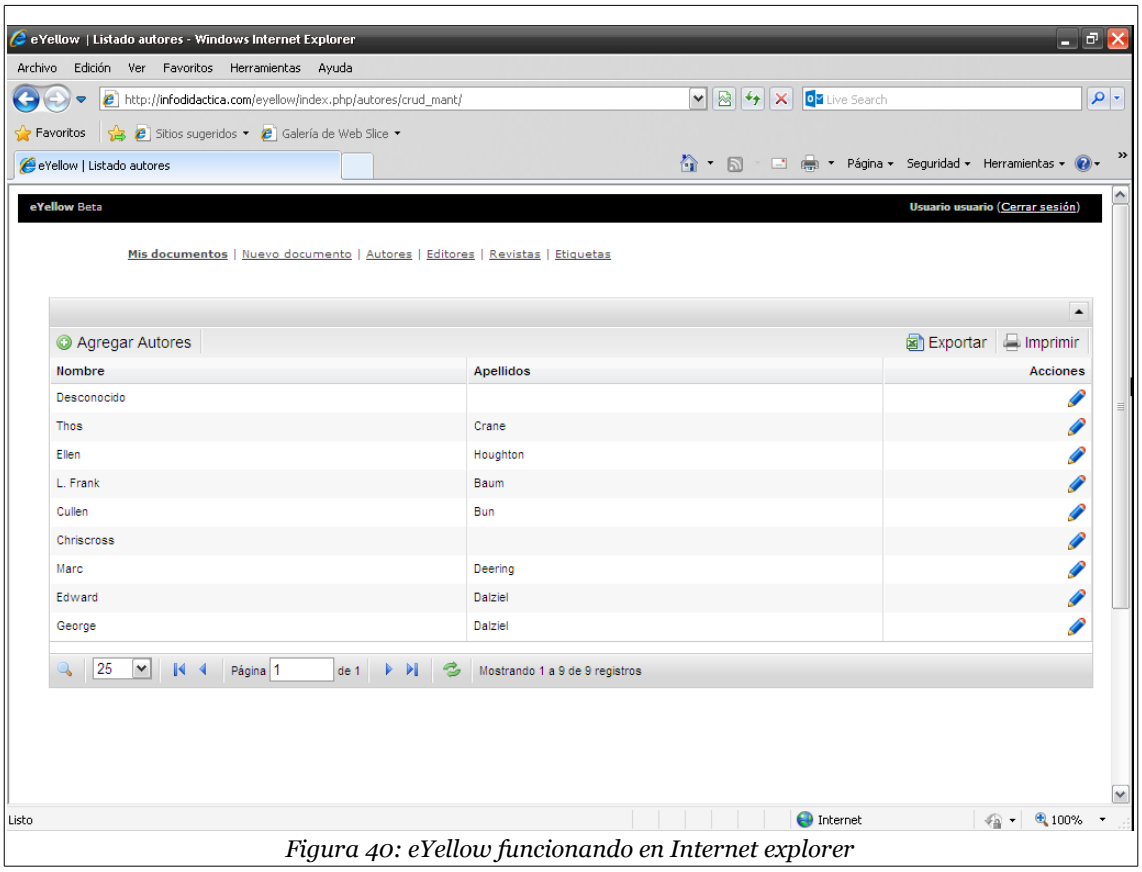


Figura 40: eYellow funcionando en Internet explorer

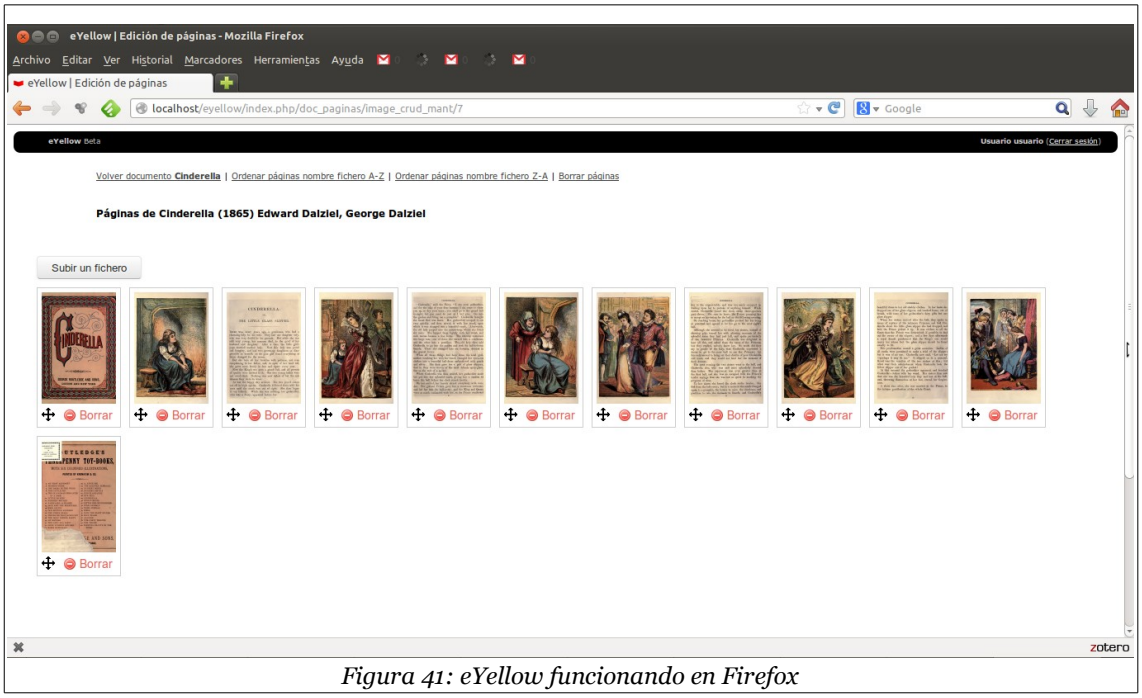


Figura 41: eYellow funcionando en Firefox

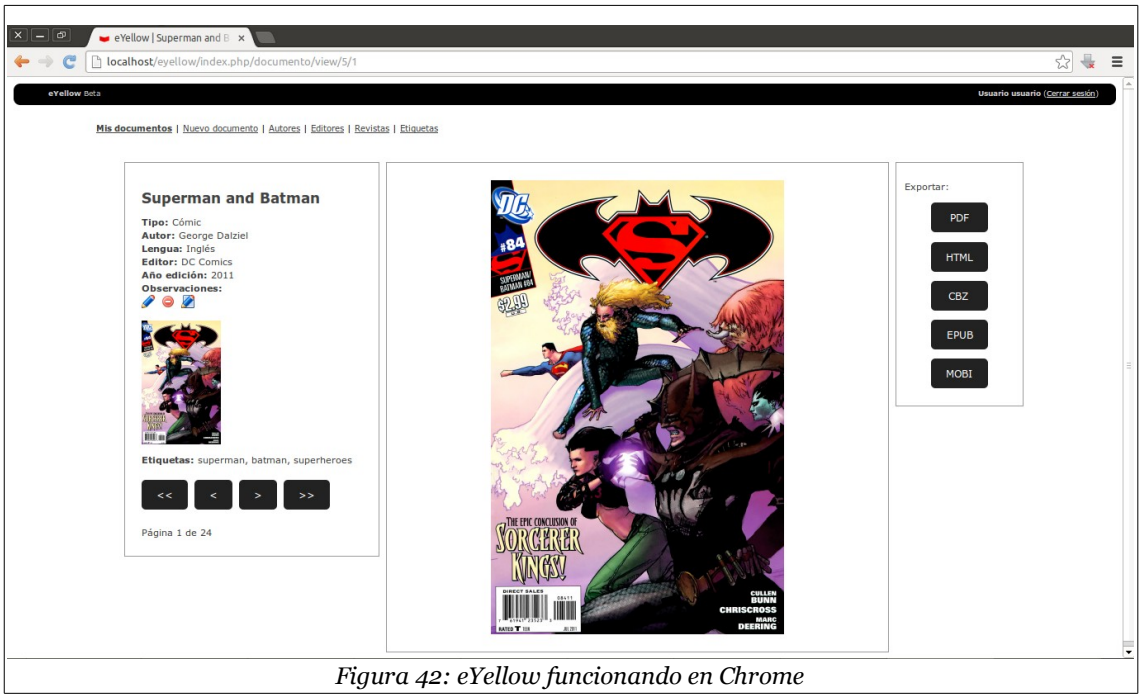


Figura 42: eYellow funcionando en Chrome

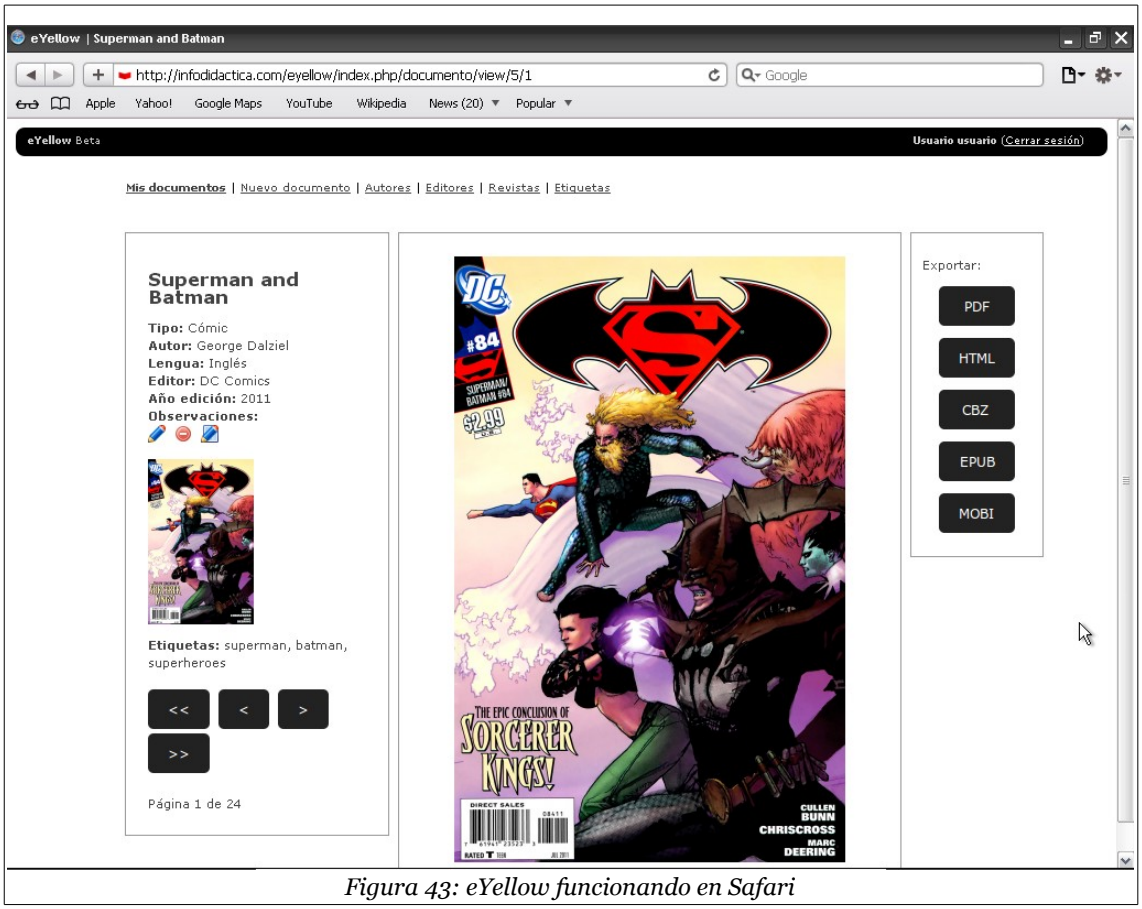


Figura 43: eYellow funcionando en Safari

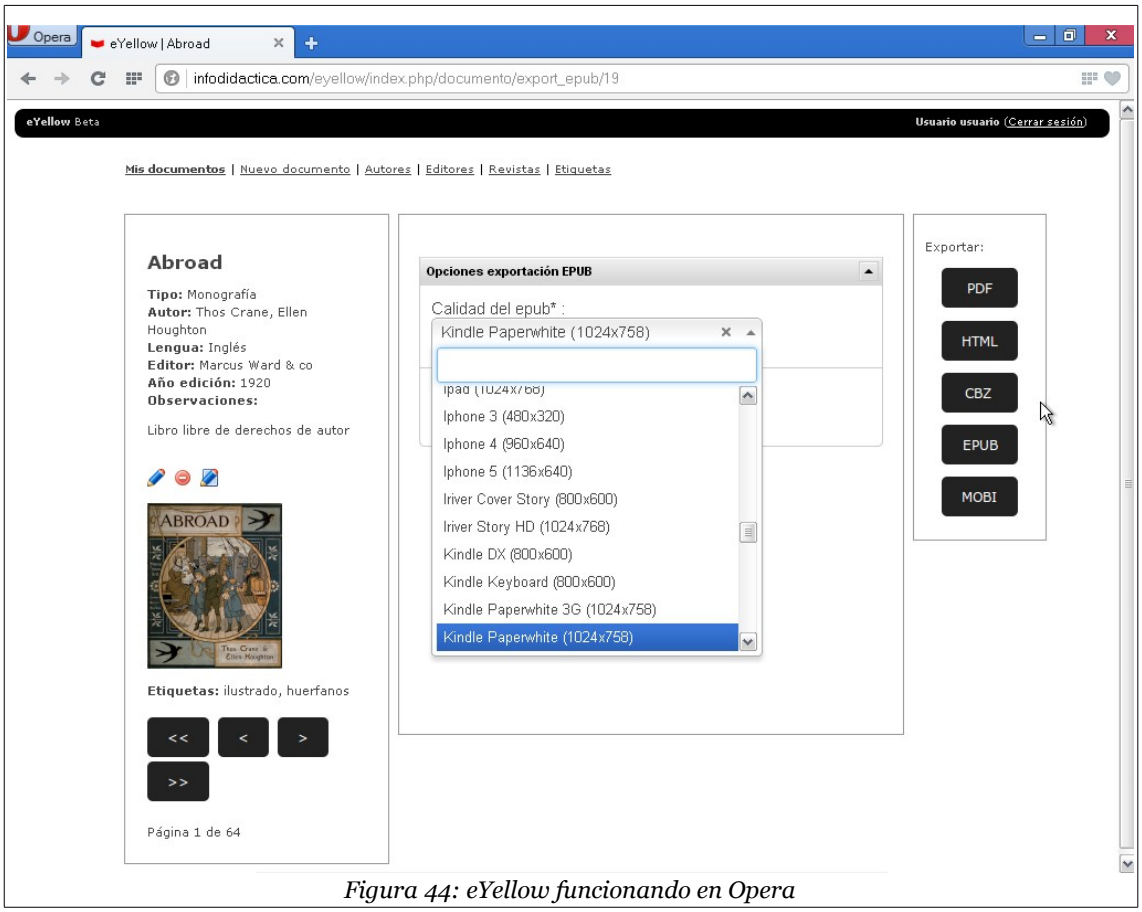


Figura 44: eYellow funcionando en Opera

Como resultado todas las pruebas realizadas en los diferentes navegadores fueron satisfactorias y la aplicación eYellow funcionó de la forma esperada y la información fue mostrada correctamente.

7.2.3 Validación de diferentes resoluciones de pantalla

eYellow es una aplicación pensada para su uso en un ordenador de escritorio por lo que las resolución necesaria para que la aplicación se visualice correctamente es, al menos, 800x600 píxeles aunque se recomienda utilizar 1024x768 píxeles.

En la figura 45 tenemos un gráfico con los porcentajes de las resoluciones más habituales en los navegadores utilizados para acceder a Internet a nivel mundial, son datos estadísticos extraídos por Netmarketshare de aproximadamente 40.000 sitios web repartidos por todo el mundo (Netmarketshare, 2013). Si observamos los datos de las resoluciones vemos que más del 60% de los usuarios de Internet utilizan una resolución suficiente para acceder correctamente a la aplicación eYellow, si bien es cierto que la tendencia al acceso a Internet desde dispositivos móviles con resoluciones inferiores cada vez se está incrementando por lo que sería interesante considerar adaptar la aplicación para poder visualizarse en resoluciones inferiores.

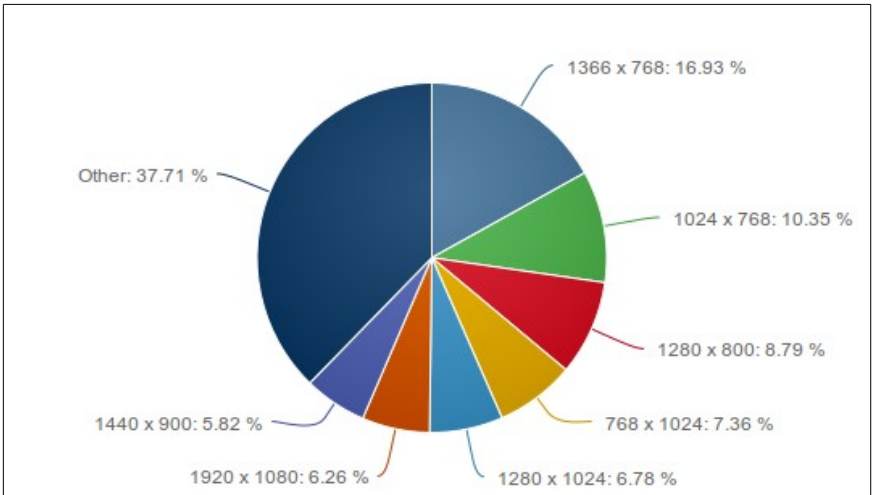


Figura 45: Resoluciones más habituales de los navegadores - Junio 2013

Quirktools ofrece una aplicación online llamada Screenfly[1] que permite testear una aplicación web en una gran variedad de dispositivos y resoluciones, una herramienta muy sencilla que sólo necesita la introducción de la URL del sitio a testear.

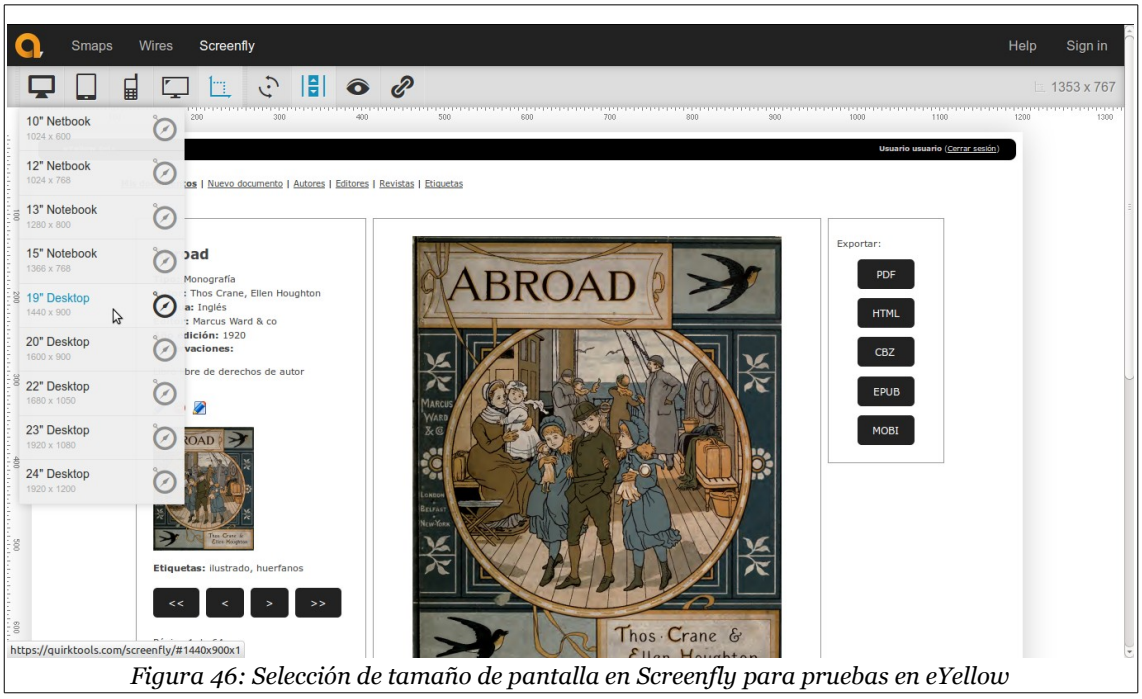


Figura 46: Selección de tamaño de pantalla en Screenfly para pruebas en eYellow

1 https://quirktools.com/screenfly/

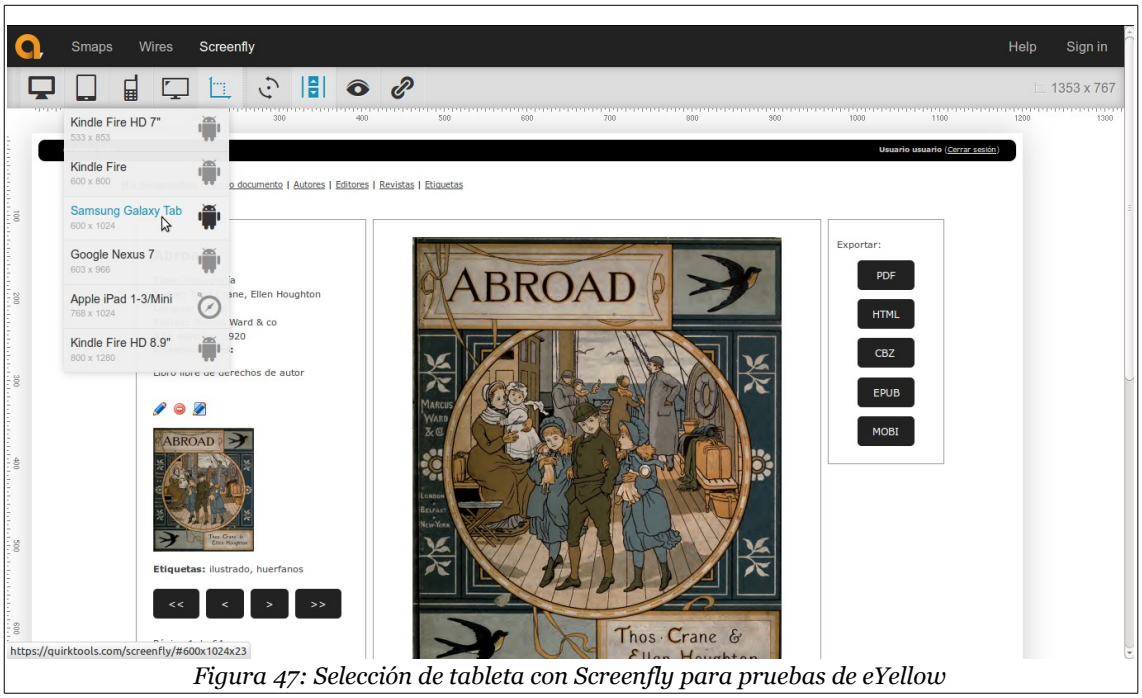


Figura 47: Selección de tableta con Screenfly para pruebas de eYellow

Después de las pruebas realizadas verificamos que la aplicación se comporta correctamente con resoluciones superiores a 800x600 píxeles, a menores resoluciones los bloques que componen la web se desordenan, solapan y no es posible trabajar correctamente con la aplicación como se observa en la figura 48.

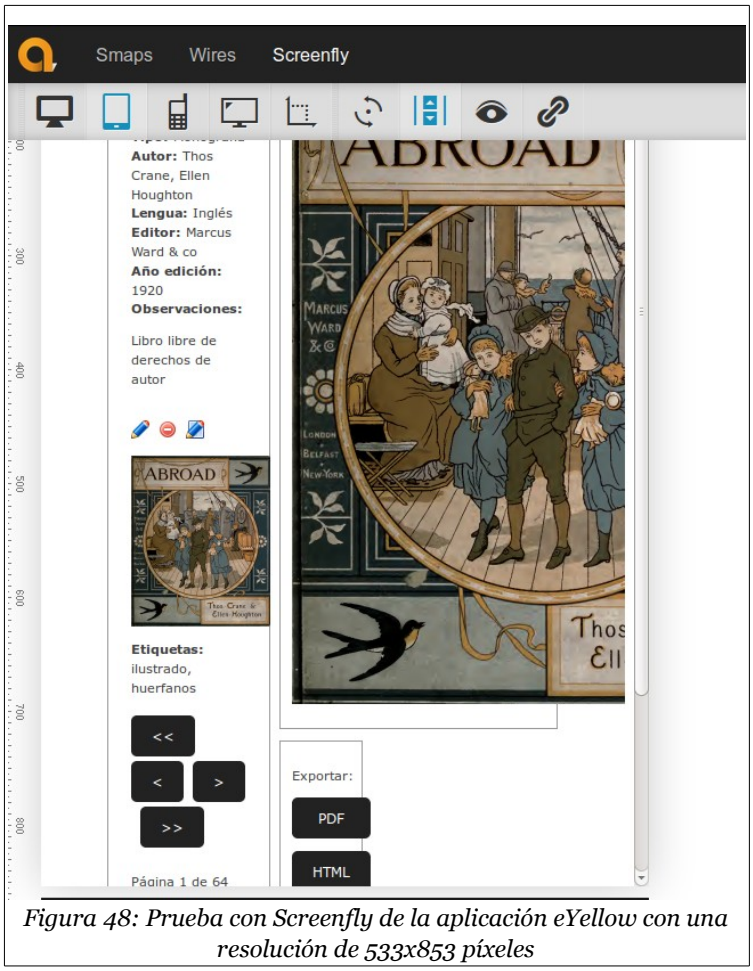


Figura 48: Prueba con Screenfly de la aplicación eYellow con una resolución de 533x853 píxeles

7.3 Pruebas producto multiformato

El objetivo principal de la aplicación eYellow era poder generar de forma sencilla diferentes formatos de documentos compatibles con las plataformas actuales que incluyeran las imágenes que habíamos incorporado al documento.

En este apartado de pruebas vamos a revisar uno a uno los formatos de documentos que genera la aplicación y comprobando que son correctos y se adecuan a lo esperado según los parámetros seleccionados en el proceso de generación.

7.3.1 Pruebas formato PDF

El formato PDF (Portable Document Format) es un formato para almacenar documentos digitales independiente de la plataforma. El PDF fue creado por la empresa Adobe Systems, pasó a ser un estándar abierto en 2008 y como estándar ISO 32000-1. Es un formato muy utilizado en Internet para el intercambio de documentos y puede contener texto, imágenes, enlaces y otros elementos multimedia.

La ventaja de que el documento PDF sea un estándar abierto es que se han generado muchos programas y librerías de software para poder crearlos con relativa facilidad. En nuestro caso y como ya indicamos en el apartado de implementación utilizamos las librerías TCPDF que permiten generar ficheros PDF desde PHP. Un documento PDF no es escalable por lo que va a ser impreso exactamente igual que se ve en la pantalla.

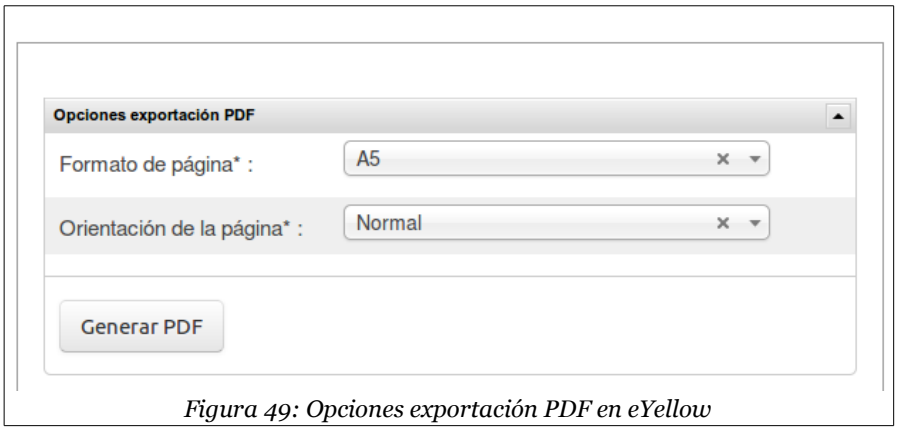


Figura 49: Opciones exportación PDF en eYellow

Como observamos en la figura 49 eYellow nos da a elegir dos parámetros cuando seleccionamos la generación del fichero PDF:

- **Formato de página** : Donde indicamos el tamaño de la página a generar y puede ser un A4, A5, A6, etc.
- **Orientación de la página** : Normal y apaisada.

Las pruebas realizadas con la exportación PDF en diferentes formatos y orientaciones han dado resultados satisfactorios y han podido ser visualizados en diferentes programas lectores de PDF.

Figura 50: Visualización de un PDF generado con eYellow

7.3.2 Pruebas formato HTML

La posibilidad de visualizar en HTML un documento la tenemos prácticamente en cualquier ordenador, tableta o móvil actual. Se decidió, por tanto, generar un formato del documento en HTML para poder consultarlo de forma local en el dispositivo o bien tener la posibilidad de subirlo a un servidor web y que fuera consultado en Internet.

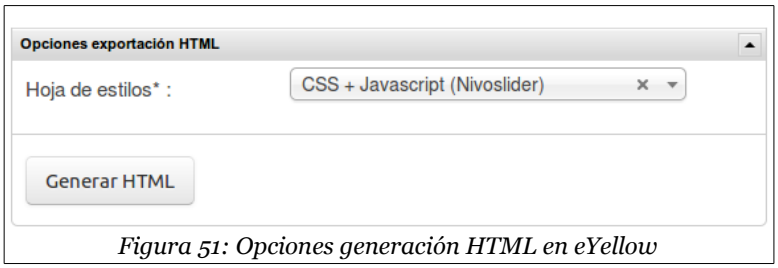
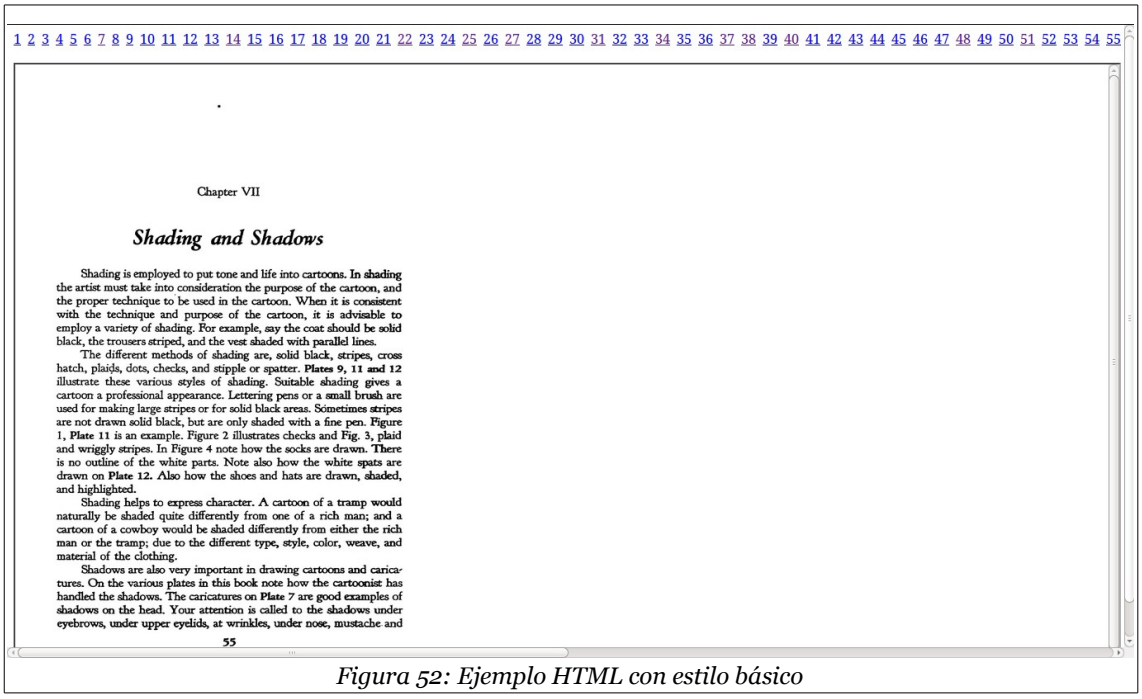


Figura 51: Opciones generación HTML en eYellow

eYellow sólo nos da a elegir un parámetro cuando seleccionamos la generación del formato HTML:

- **Hoja de estilos** : Nos da a elegir un estilo básico o un estilo más elaborado que utiliza CSS, javascript y una librería para realizar una galería de fotografías llamada Nivo-Slider[1].

Las pruebas realizadas con la generación de documento en HTML utilizando ambos estilos han sido correctas y se visualizan perfectamente en un navegador tanto en local como en un servidor web.


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55

Chapter VII

Shading and Shadows

Shading is employed to put tone and life into cartoons. In shading the artist must take into consideration the purpose of the cartoon, and the proper technique to be used in the cartoon. When it is consistent with the technique and purpose of the cartoon, it is advisable to employ a variety of shading. For example, say the coat should be solid black, the trousers striped, and the vest shaded with parallel lines.

The different methods of shading are, solid black, stripes, cross hatch, plaids, dots, checks, and stipple or spatter. **Plates 9, 11 and 12** illustrate these various styles of shading. Suitable shading gives a cartoon a professional appearance. Lettering pens or a small brush are used for making large stripes or for solid black areas. Sometimes stripes are not drawn solid black, but are only shaded with a fine pen. **Figure 1, Plate 11** is an example. Figure 2 illustrates checks and **Fig. 3,** plaid and wriggly stripes. In Figure 4 note how the socks are drawn. **There** is no outline of the white parts. Note also how the white spats are drawn on **Plate 12.** Also how the shoes and hats are drawn, shaded, and highlighted.

Shading helps to express character. A cartoon of a tramp would naturally be shaded quite differently from one of a rich man; and a cartoon of a cowboy would be shaded differently from either the rich man or the tramp; due to the different type, style, color, weave, and material of the clothing.

Shadows are also very important in drawing cartoons and caricatures. On the various plates in this book note how the cartoonist has handled the shadows. The caricatures on **Plate 7** are good examples of shadows on the head. Your attention is called to the shadows under eyebrows, under upper eyelids, at wrinkles, under nose, mustache and

55

Figura 52: Ejemplo HTML con estilo básico

La figura 52 muestra la salida generada de un documento utilizando el estilo básico, aparece un paginador en la parte superior y no se utiliza CSS ni ningún tipo de efecto con Javascript.

1 http://dev7studios.com/nivo-slider/

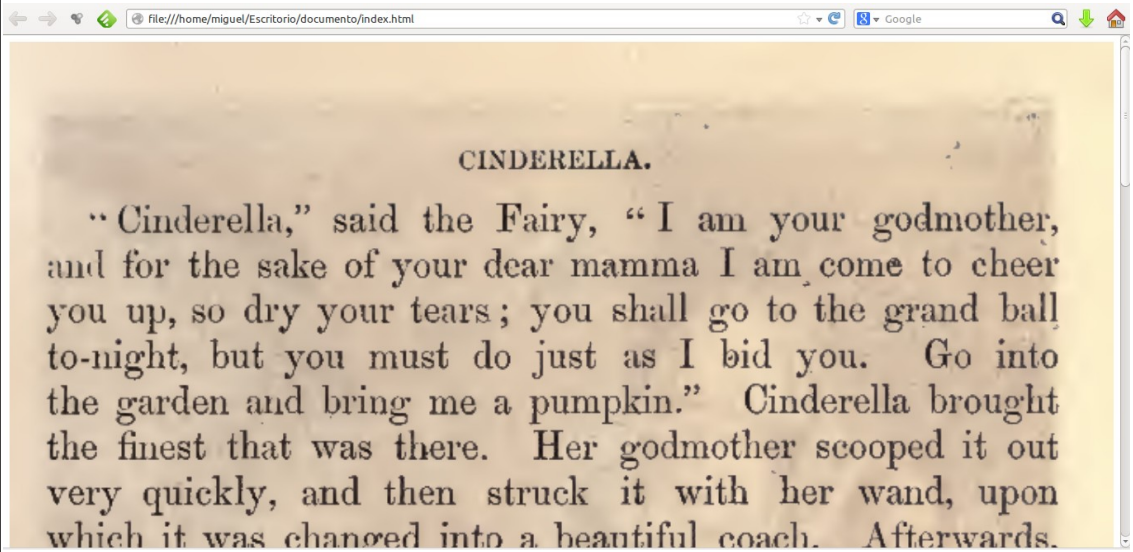


Figura 53: Visualización de formato HTML generado con el estilo con Nivo-Slider

En la figura 53 se muestra la salida HTML que genera la exportación CSS con javascript y Nivo-Slider. En este formato la imagen sale a tamaño real sin escalar, aparece en la parte inferior un paginador y en los laterales unos botones que permiten ir página adelante y atrás. La transición de páginas se hace con un efecto de desvanecimiento.

En esta opción de exportación de la aplicación es muy susceptible de ser ampliada e incluir más estilos y opciones en la generación.

7.3.3 Pruebas formato CBZ

CBZ es un formato de fichero cuyo objetivo es contener una secuencia de imágenes por lo que es un formato muy utilizado para contener cómics, por eso es también conocido como "Comic Book". Un "Comic book" consiste en una serie de imágenes, habitualmente JPG o PNG que se almacenan en un sólo fichero comprimido, la extensión del fichero comprimido nos indica el formato de compresión utilizado, por ejemplo, CBR es un "Comic Book" comprimido en RAR y CBZ en un "Comic Book" comprimido en ZIP.

Internamente los ficheros CBZ contienen todas las imágenes que forman parte del libro, estas imágenes se numeran de forma ascendente y así se genera la ordenación interna de las páginas.

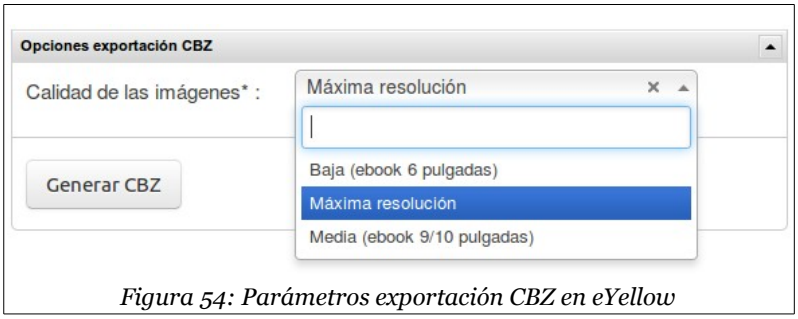


Figura 54: Parámetros exportación CBZ en eYellow

eYellow sólo nos da a elegir un parámetro cuando seleccionamos la generación del formato CBZ:

- **Calidad de las imágenes** : Permite seleccionar la resolución de las imágenes que van a ser incluidas en el CBZ. Podemos elegir baja resolución que sería óptima para un ebook de 5 o 6 pulgadas, resolución media que se adaptaría a un ebook de 9 o 10 pulgadas y, por último, máxima resolución donde la imagen mantiene la misma resolución con la que se ha almacenado en la aplicación. A mayor resolución mayor calidad de imagen pero el fichero CBZ ocupa más espacio.

Las pruebas realizadas con la generación de ficheros CBZ han sido correctas y se han podido visualizar tanto en un ordenador, un móvil y un ebook.

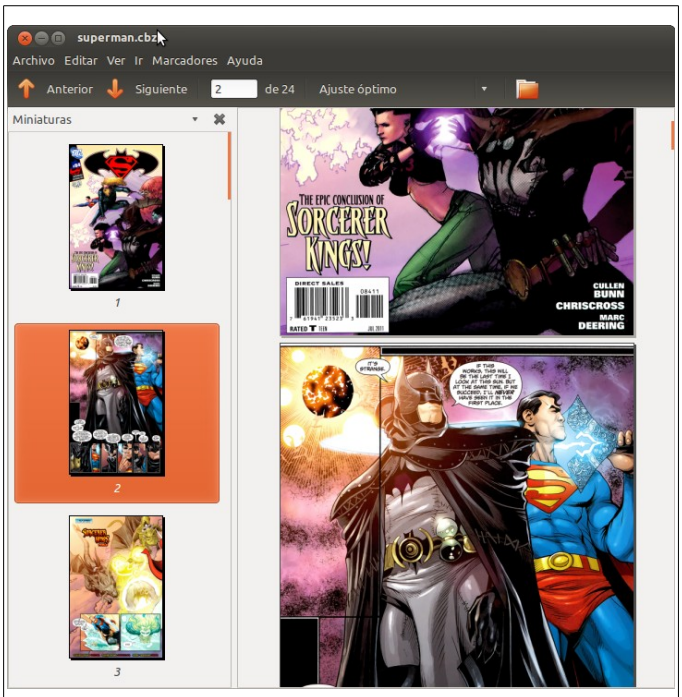


Figura 55: Fichero CBZ generado con eYellow visualizándose en una aplicación de escritorio

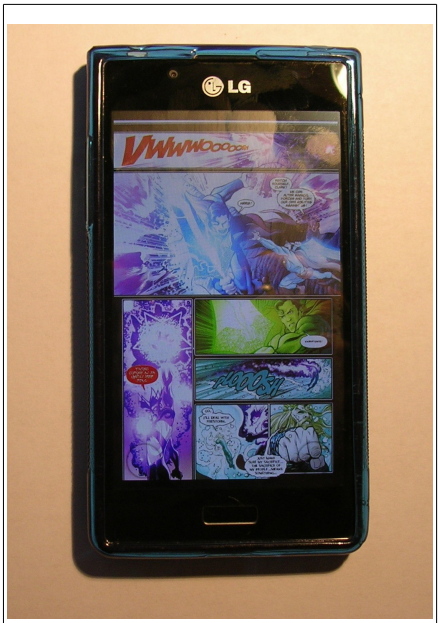


Figura 56: Fichero CBZ generado con eYellow en un móvil con Android 4

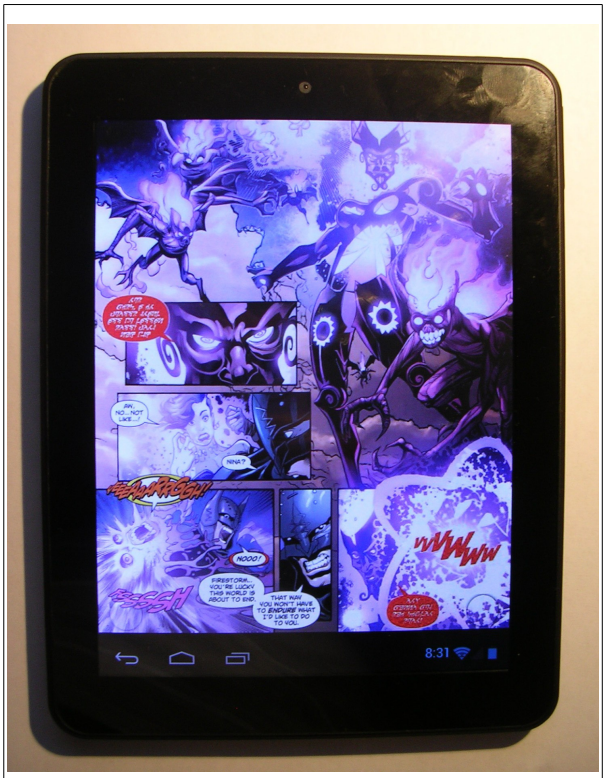


Figura 57: Fichero CBZ generado con eYellow en una tableta con Android 4

En las figuras 55, 56 y 57 observamos un fichero CBZ (Comic Book comprimido en zip) visualizándose sin problemas en varios dispositivos.

7.3.4 Pruebas formato EPUB

EPUB es un formato de código abierto para archivos de libro electrónico creado por Internacional Digital Publishing Forum (IDPF)[1]. El formato EPUB es redimensionable por lo que el fichero marca el contenido pero no se delimita el formato como si hace, por ejemplo, el formato PDF. Cuando el contenido del fichero epub es texto éste se adapta al tamaño de pantalla de los diversos libros electrónicos y dispositivos que existen en el mercado, por lo que no tiene sentido indicar el número de páginas que tiene un fichero epub porque en función de la capacidad de la pantalla del dispositivo y del tamaño del tipo de letra elegido la paginación del epub se modifica totalmente.

La versión actual de epub es la 3 aunque la mayoría de los libros electrónicos sólo soportan la versión 2 ya que la 3 incluye capacidades como audio incrustado y normas de pronunciación que no son soportadas por los dispositivos. Los ficheros epub que se generan con eYellow son compatibles con ambas versiones.

Cualquier usuario puede generar un epub de forma libre, al estar publicado el estándar de forma gratuita ha favorecido la aparición de muchos programas que dan soporte a la creación del epub. En el Anexo I se ha incluido un pequeño manual acerca de cómo crear un epub, se indica los ficheros necesarios y la estructura para que sea válido y pueda ser utilizado en un dispositivo que soporte el formato.

Una de las críticas al formato epub es que trabaja de forma excelente con texto pero cuando se deben de mostrar, por ejemplo, gráficos el formato está muy limitado. Los ficheros generados con epub son precisamente todo gráficos y no contienen texto, por lo que para adaptarlos correctamente al dispositivo que vamos a utilizar se deben de indicar las dimensiones de las imágenes para que puedan visualizarse a pantalla completa y no debamos de utilizar el scroll para leer el libro.

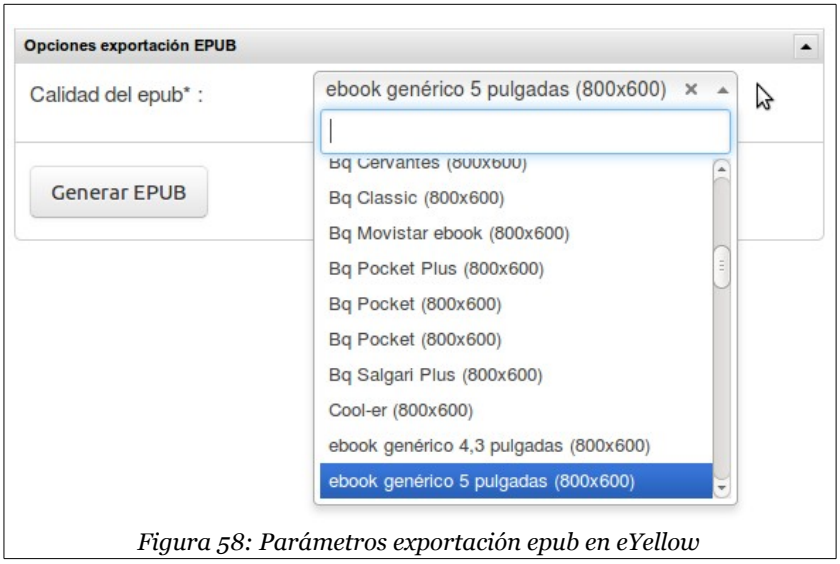


Figura 58: Parámetros exportación epub en eYellow

1 http://idpf.org/

eYellow sólo nos da a elegir un parámetro cuando seleccionamos la generación del formato EPUB:

- **Calidad del EPUB**: Permite seleccionar la resolución de las imágenes que van a ser incluidas en el EPUB. Para facilitar la selección en un dispositivo cuya resolución desconozca el usuario final se han incluido además de resoluciones estándar una lista de libros electrónicos más utilizados y la aplicación asigna la resolución de forma automática.

Las pruebas realizadas con la generación de ficheros EPUB han sido correctas y se han podido visualizar tanto en un ordenador, un móvil, una tableta y un ebook.

Internacional Digital Publishing Forum proporciona un validador online[1] de epubs para verificar que un epub generado por nosotros es válido de acuerdo al estándar propuesto. IDPF también proporciona un programa en java Epubcheck[2] con la misma funcionalidad y que está pensado para validar epubs versión 2 y 3 de forma más masiva.

En la figura 59 vemos el resultado mostrada por pantalla tras la validación online con la herramienta proporcionada por IDPF de un documento generado con eYellow.

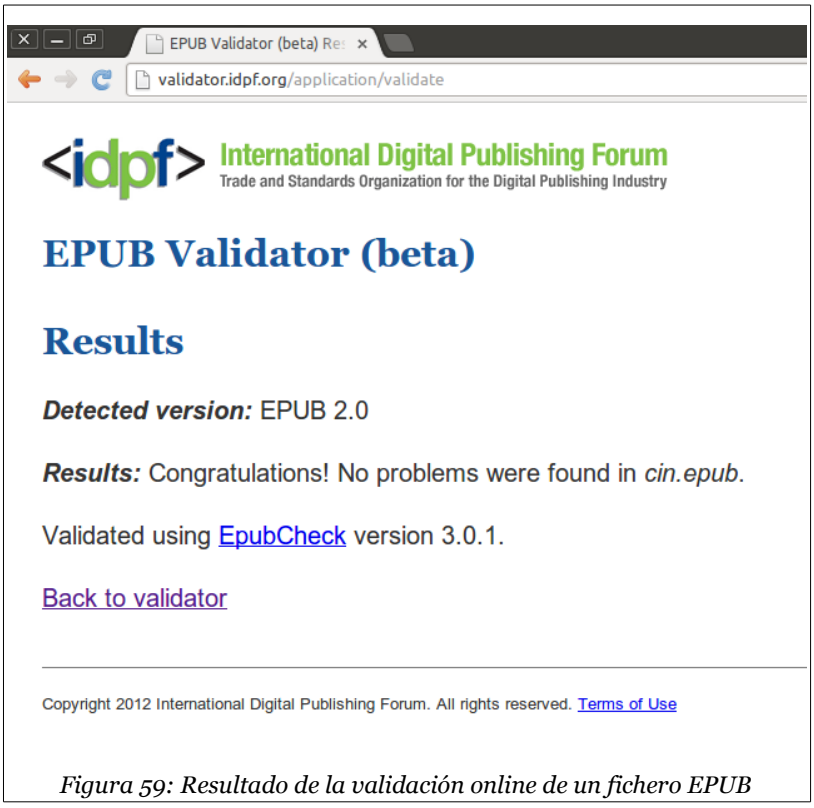


Figura 59: Resultado de la validación online de un fichero EPUB

1 http://validator.idpf.org/

2 http://code.google.com/p/epubcheck/

Calibre es un gestor de libros electrónicos libre muy utilizada para la catalogación y conversión de libros electrócnicos. Una de las pruebas realizadas con los ficheros epub generados con eYellow ha sido importarlos a Calibre y comprobar que los datos visualizados eran los esperados. En la figura 60 tenemos el visor de Calibre con un epub generado con eYellow y con una presentación correcta, la figura 61 es una pantalla de edición de metadatos asociados al libro y como podemos apreciar también han sido generados correctamente desde eYellow.

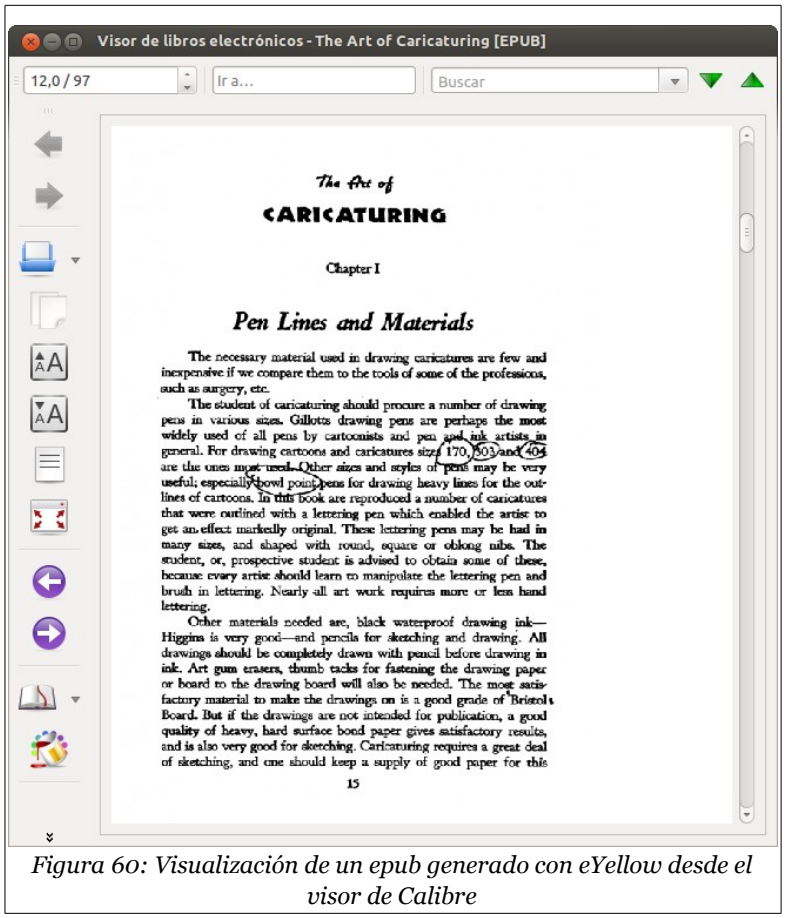


Figura 60: Visualización de un epub generado con eYellow desde el visor de Calibre

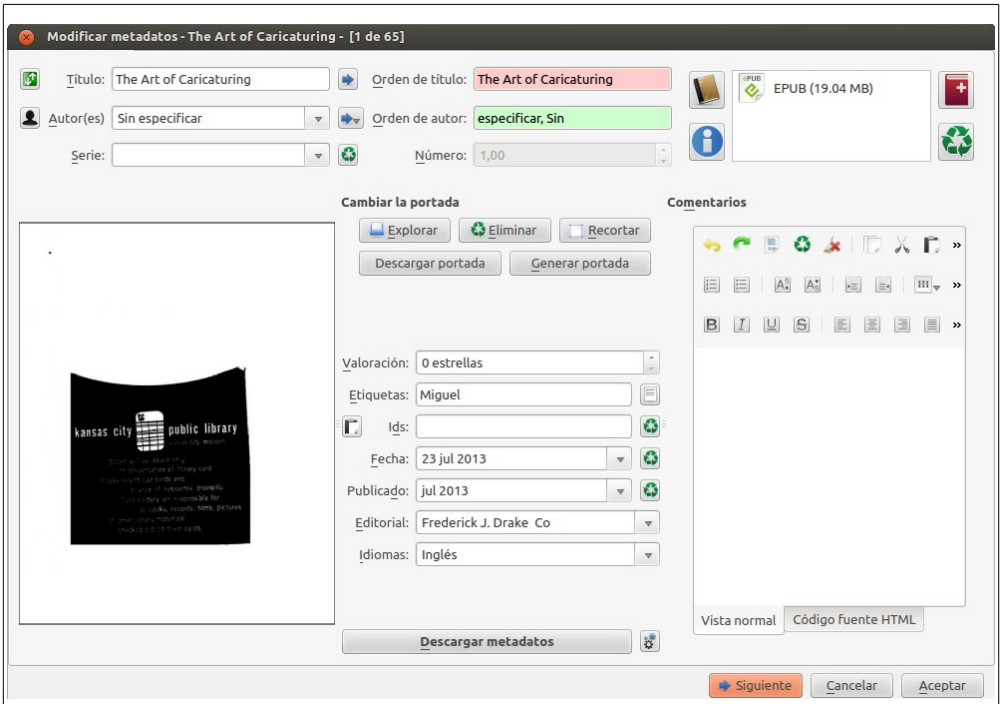


Figura 61: Visualización de los metadatos del epub en Calibre

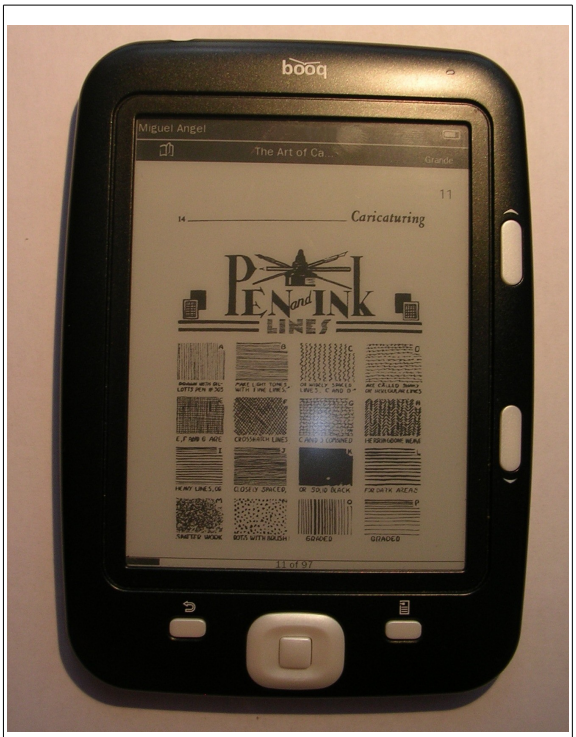


Figura 62: Visualización de un epub generado con eYellow en un ebook

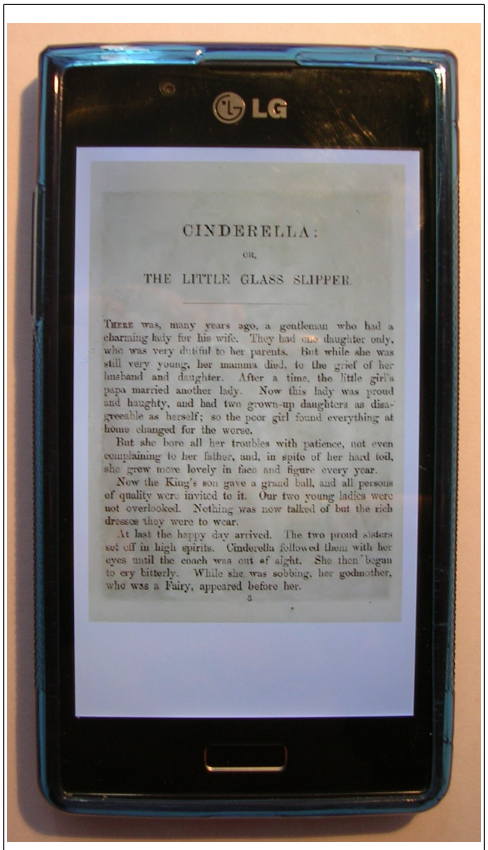


Figura 63: Visualización de un epub generado con eYellow en un móvil

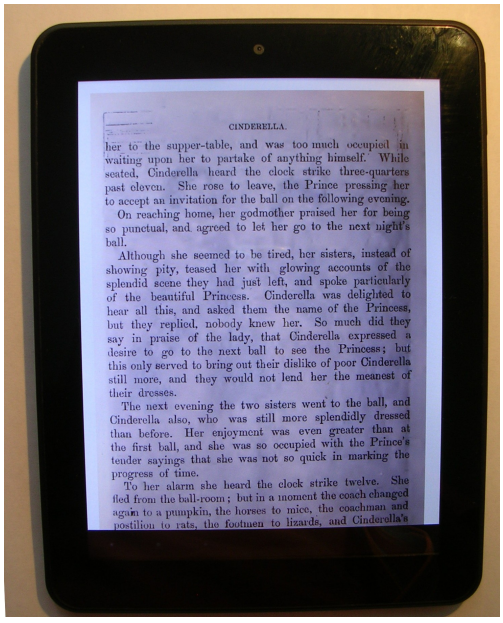


Figura 64: Visualización de un epub generado con eYellow en una tableta

En las figuras 62, 63 y 64 se muestra la visualización de un epub de prueba generado en eYellow y mostrado en un ebook, un móvil y una tableta. En el móvil y la tableta se ha utilizado el programa Aldiko[1] de Android para la visualización del ebook.

7.3.5 Pruebas formato MOBI

El formato de fichero MOBI o MOBIPOCKET es un formato binario de distribución de libros electrónicos. La implementación del libro está basado en el estándar propuesto por IDPF por lo que la estructura es similar a un fichero electrónico EPUB por lo que al igual que ocurre con éste la presentación del contenido de un libro electrónico en un formato MOBI también es redimensionable.

Hasta aquí las semejanzas entre los dos formatos porque MOBI es un formato que se creó para incluir DRM (Digital Rights Management o Gestión de Derechos Digital) y el último paso para la creación del formato incluye un proceso de encriptación en un formato binario del que no existe ningún tipo de documentación clara debido a que está protegido por derechos de autor y no es libre (Mobipocket, 2013). ¿Cómo se ha realizado este último paso en eYellow? Pues después de investigar y conocer que las únicas aproximaciones al proceso se habían generado por ingeniería inversa y no quedaban nada claras se optó por utilizar la herramienta que proporciona Amazon denominada Kindlegen[2]. Kindlegen es una herramienta de línea de ordenes que permite convertir un fichero EPUB a MOBI, no se incluye el código fuente de la herramienta pero es gratuita, tiene versiones para Windows, OS X y Linux. Utilizando la versión Linux se ha incluido dentro de los ficheros de la aplicación eYellow en un servidor web bajo Linux y el proceso de conversión se realiza desde el mismo eYellow generando un fichero binario MOBI totalmente válido.

Las pruebas en este apartado se han limitado a la comprobación de que la herramienta Kindlegen generaba correctamente el fichero MOBI a partir del fichero EPUB que se había generado previamente desde eYellow dando resultados satisfactorios.

1 http://www.aldiko.com/

2 http://www.amazon.com/gp/feature.html?ie=UTF8&docId=1000765211

8. Conclusiones

Como conclusión al Proyecto Final de Carrera me gustaría destacar la experiencia personal satisfactoria que me ha supuesto la creación de la aplicación eYellow, una herramienta que aúna por una parte mi perfil de informático y por otra la de documentalista cuya titulación es el fin de este Proyecto Final de Carrera.

La programación web es un área interesante y muy demandada dentro de los perfiles existentes de un informático o documentalista pero si además se conjuga con la generación de contenidos y formatos compatibles con dispositivos móviles se crea una simbiosis perfecta para la realización de un proyecto que sea útil en la vida real debido a la cantidad de tecnologías que abarca.

Aunque quedan fuera de la extensión y objetivo inicial del proyecto me gustaría plantear una serie de posibles ampliaciones y mejoras a la aplicación eYellow que podrían ser interesantes incluso como continuación de otro proyecto final de carrera:

- El interface de la aplicación eYellow está pensada para su uso en equipos de escritorio o con una resolución de 800x600 píceles por lo que se podría adaptar la aplicación web eYellow para poder utilizarse en dispositivos móviles o tablets.

- Mejorar la gestión de los metadatos en todos los documentos generados.

- Añadir más parámetros de exportación en los diferentes formatos que genera la aplicación. Por ejemplo, diferentes plantillas en el formato HTML o más parámetros en la generación del PDF.

- Mejorar el rendimiento de la aplicación en el proceso de generación de PDFs. Ahora requiere bastante tiempo de CPU la generación de un PDF de muchas páginas, cuando la aplicación se ejecuta en local no es problema pero cuando se está utilizando una aplicación remota donde el tiempo de CPU del servidor web es un elemento crítico y puede generar timeouts al usuario final de eYellow. Este retardo se da sobretodo cuando la generación conlleva el escalado de las imágenes incluidas en el PDF.

- Añadir más dispositivos y parametrizaciones en el diálogo de selección de formato de la generación de epub.

Y para finalizar hacer la afirmación de que el proyecto realizado ha cumplido con mis expectativas, tengo la sensación de haber realizado algo útil, no para cubrir el expediente y espero obtener provecho del proyecto en mi vida laboral.

Anexo – Creación de un epub

1 Introducción

En este anexo del proyecto se incluye un tutorial completo para crear un epub desde cero mediante el uso de un editor de textos y un compresor. Aunque existen herramientas que permiten generar epubs, como por ejemplo Sigil[1], la mayoría de ellas ocultan al usuario final la estructura real de un fichero epub. Este tutorial no busca suplir ninguna de estas herramientas ya que sólo tiene como objetivo mostrar de forma sencilla el contenido de un epub, cómo se estructura éste y que sirva de ejemplo didáctico de su creación.

2 El formato EPUB

EPUB es el acrónimo de la expresión inglesa electronic publication (publicación electrónica) y es un formato redimensionable de código abierto creado por Internacional Digital Publishing Forum (IDPF). Epub define la forma de representar, empaquetar y codificar la estructura de contenido web distribuido en un formato de un solo fichero.

Como se va a ver en el tutorial en el formato epub no se delimita el formato con el que se va a mostrar el documento sino que se marca el contenido y éste se adapta a los diferentes tamaños de pantallas existentes en los distintos dispositivos que permiten visualizar epubs.

Existen en la actualidad 2 versiones de EPUB:

- **EPUB 2.0.1** : Inicialmente estandarizada en el año 2007 con la versión 2 se aprobó en en 2010 la versión 2.0.1 como versión de mantenimiento. Esta versión es el formato que soportan la mayoría de los lectores de libros electrónicos existentes en el mercado.

- **EPUB 3** : Es la versión actual de EPUB que se aprobó en octubre de 2011 y sustituía a la versión 2.0.1. Esta versión añade mejoras sustanciales al formato EPUB en cuanto al contenido multimedia que puede incluir un epub como imágenes, audio y vídeo, contenido en HTML 5, formato en CSS3, ficheros SVG, posibilidad de pronunciación de parte del texto, etc. La realidad es que los lectores de libros electrónicos existentes en la actualidad en el mercado no son capaces de mostrar toda esa riqueza de contenido que permite expresar la versión 3.

1 http://code.google.com/p/sigil/

Todas las versiones de epub propuestas por la IDPF son libres, cualquiera puede utilizarlas para crear sus documentos sin necesidad que tener que pagar ningún tipo de licencia así como crear programas que permitan generar documentos EPUB.

Como todos los documentos epub versión 2 son compatibles con la versión 3, este tutorial lo realizaremos creando un documento en la versión 2. Todos los conceptos vistos aquí son también aplicables a la versión 3 aunque sin tener en cuenta sus mejoras.

Un fichero epub se define mediante las especificaciones de tres estándares abiertos:

- **Open Publication Structure (OPS)**[1] : Es el estándar que describe cómo representar el contenido de una publicación electrónica mediante etiquetado con XHTML o HTML y para dar formato con hojas de estilos utilizando un subconjunto de CSS2.

- **Open Packaging Format (OPF)**[2] : Describe el mecanismo mediante el que varios componentes de una publicación OPS son enlazados juntos y proporciona una estructura adicional y semántica para la publicación electrónica. Permite almacenar: imágenes, metadatos, orden de lectura de la publicación y proporciona un mecanismo para especificar la estructura global de navegación mediante un fichero NCX.

- **Open Container Format (OCF)**[3] : Define como agrupar un conjunto de ficheros para generar un único fichero epub. Tener un único fichero contenedor facilita el transporte, gestión y acceso al contenido del epub.

Las herramientas básicas necesarias para crear un epub son un editor de textos y un compresor zip. Este tutorial se ha escrito utilizando Ubuntu 12.04 aunque puede ser fácilmente reproducido en cualquier sistema operativo que incluya las dos herramientas indicadas.

El tutorial reproduce paso a paso cómo crear un epub básico que incluye información de varias frutas.

3 Crear estructura de carpetas

Procedemos a crear la estructura de carpetas que alojará el contenido del epub de acuerdo al estándar indicado en OCF.

Primero creamos una carpeta raíz y dentro de esta carpeta el siguiente árbol de carpetas:

1 http://www.idpf.org/epub/20/spec/OPS_2.0.1_draft.htm

2 http://www.idpf.org/epub/20/spec/OPF_2.0.1_draft.htm

3 http://www.idpf.org/doc_library/epub/OCF_2.0.1_draft.doc

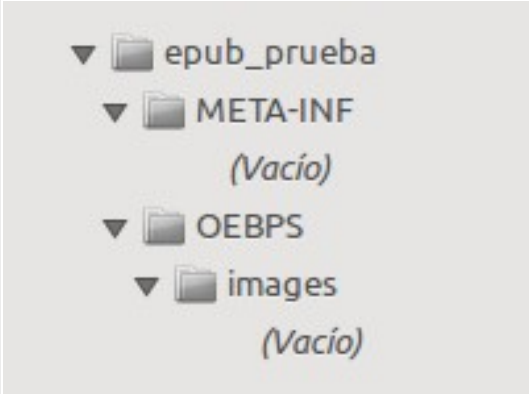


Figura 65: Estructura de carpetas de un epub

La carpeta epub_prueba puede tener el nombre que se quiera, el resto de carpetas deben de tener el mismo nombre respetando la capitalización.

4 Crear el contenido del epub

En este punto crearemos el contenido del epub y le daremos formato. Para el contenido utilizaremos HTML o XHTML y para el formato CSS2 tal y como indica el estándar OPS.

La especificación OPS soporta muchas de las etiquetas HTML pero no todas. La lista completa de etiquetas permitidas son:

Nombre módulo	**Etiquetas permitidas**
Estructura	body, head, html, title
Texto	abbr, acronym, address, blockquote, br, cite, code, dfn, div, em, h1, h2, h3, h4, h5, h6, kbd, p, pre, q, samp, span, strong, var
Hipertexto	a
Listas	dl, dt, dd, ol, ul, li
Objetos	object, param
Presentación	b, big, hr, i, small, sub, sup, tt
Edición	del, ins
Texto Bidireccional	bdo
Tablas	caption, col, colgroup, table, tbody, td, tfoot, th, thead, tr

Nombre módulo	Etiquetas permitidas
Imágenes	img
Mapa imagen	area, map
Metadatos	meta
Hoja de estilos	style
Enlaces	link
Base	base

Vamos a crear un fichero epub con 3 páginas que incluirá en cada una de las páginas una descripción de una fruta con una fotografía. También incluiremos una portada en el epub.

Descargamos las imágenes con licencia creative commons que utilizaremos en el epub:

- Cerezas en http://www.flickr.com/photos/calafellvalo/5732951597/
- Melocotones en http://www.flickr.com/photos/90315233@N00/205157538/
- Uva en http://www.flickr.com/photos/32179778@N00/8125206809/
- Portada en http://www.flickr.com/photos/8851458@N04/3411354782/

Copiamos las imágenes en la carpeta images dentro de OEBPS. Los formatos de imágenes admitidos según el estándar son gif, png, jpeg y svg, si tenemos problemas de visualización de las imágenes en algún dispositivo mejor utilizar el formato png.

Figura 66: Localización de las imágenes en la estructura de carpetas del epub

Ahora crearemos los ficheros XHTML con el contenido del epub, estos ficheros que crearemos deben de almacenarse en la carpeta OEBPS.

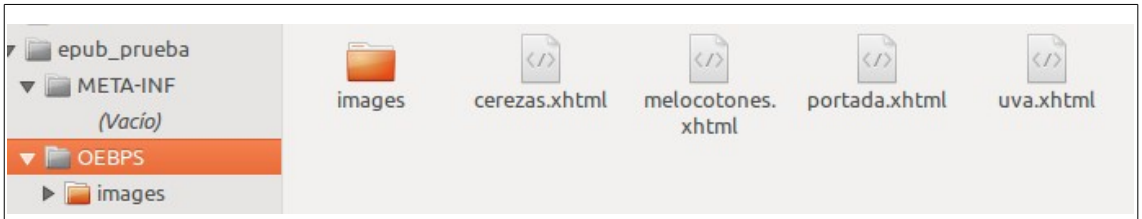


Figura 67: Localización de los ficheros HTML en la estructura de carpetas del epub

Un ejemplo del contenido de los ficheros utilizados en este tutorial lo podemos ver en las figuras 68, 69, 70 y 71.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
        <head>
                <title>Las frutas</title>
                <link href="estilo.css" rel="stylesheet" type="text/css" />
        </head>
<body>
        <h1>Las frutas</h1>
        <div>
                <img src="images/portada.jpg" alt="" />
        </div>
</body>
</html>
```

Figura 68: Contenido de portada.xhtml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
        <head>
                <title>Cerezas</title>
                <link href="estilo.css" rel="stylesheet" type="text/css" />
        </head>
<body>
        <div>
                <img src="images/cerezas.jpg" alt="" />
        </div>
        <h1>Las cerezas</h1>
        <p>La Cereza es una fruta rica en vitaminas A, B, C, E, K y PP, en hierro, calcio, magnesio, potasio y azufre.</p>
        <p>En España se distingue como guinda al fruto de Prunus cerasus y como cereza al de Prunus avium,
           siendo en América más común la denominación cereza ácida, para el primero y cereza dulce, para el segundo.</p>
</body>
</html>
```

Figura 69: Contenido de cerezas.xhtml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
        <head>
                <title>Melocotones</title>
                <link href="estilo.css" rel="stylesheet" type="text/css" />
        </head>
<body>
        <div>
                <img src="images/melocotones.jpg" alt="" />
        </div>
        <h1>Melocotón</h1>
        <p>Es originario de China, Afganistán e Irán. Fue traído a occidente por los romanos que lo
           tomaron como originario de Persia y así lo denominaron. Esta denominación «persica»,
           persiste en numerosos nombres populares ibéricos como, por ejemplo, alberchigo (el pérsico)
           o bresquilla.</p>
</body>
</html>
```

Figura 70: Contenido de melocotones.xhtml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
        <head>
                <title>Uva</title>
                <link href="estilo.css" rel="stylesheet" type="text/css" />
        </head>
<body>
        <div>
                <img src="images/uva.jpg" alt="" />
        </div>
        <h1>Las uva</h1>
        <p>La uva es una fruta obtenida de la vid. Las uvas, granos de uva, vienen en racimos y son pequeñas y dulces.
 Se comen frescas o se utilizan para producir agraz, mosto, vino y vinagre.
 Crecen agrupadas en racimos de entre 6 y 300 uvas.
 Pueden ser negras, moradas, amarillas, doradas, púrpura, rosadas, marrones, anaranjadas o blancas.</p>
</body>
</html>
```

Figura 71: Contenido de uva.xhtml

Las dos primeras líneas de los ficheros XHTML son comunes en todos los documentos así como el atributo xmlns de la etiqueta html. El resto del fichero es XHTML básico y no hay ninguna referencia al estilo excepto el link a la hoja de estilos estilo.css que tenemos en el encabezado del documento.

No todos los lectores electrónicos soportan todas las etiquetas especificadas en el estándar por lo que es recomendable utilizar pocas etiquetas como <p>, <h1>, <h2>, <a>, <li>, <img>, <ol>, etc. Cuanto más simple sea el documento más posibilidades que se visualice correctamente en cualquier dispositivo.

5 Dar formato al contenido del epub

En el punto 3 del estándar OPS se indican las reglas a seguir para crear el CSS que se utilizará para dar formato al documento EPUB. El estilo OPS sigue la sintaxis de CSS2 y hay que tener en cuenta de que las reglas de estilo creadas diferencias entre mayúsculas y minúsculas. De nuevo, y como ya ocurría con el XHTML, no todos los lectores electrónicos soportan todo el juego de selectores y valores CSS que indica el estándar por lo que siempre será mejor crear un CSS simple para garantizar que nuestro epub se visualice bien en mayor número de dispositivos electrónicos.

Para dar el formato del contenido del epub añadiremos las reglas CSS en el fichero que hayamos referenciado desde los XHTML, en nuestro casos creamos el fichero estilo.css en la carpeta OEBPS.

Figura 72: Localización del fichero CSS en la estructura de carpetas del epub

Este fichero contendrá las reglas necesarias para formatear el contenido creado.

```
h1 {
font-family: "Cochin", Palatino, serif;
font-size: 3em;
text-align: center;
color: #158f40;
}

p {
font-family: "Optima-Regular", Verdana, sans-serif;
line-height: 2em;
text-align: left;
margin:0;
}

div {
margin: 0 .5em 0 0;
float:left;
width:90%;
line-height:2em;
}

img {
width:100%;
vertical-align: text-top;
margin-bottom:.5em;
}
```

Figura 73: Contenido del fichero estilo.css

6 Preparar los ficheros XML

Ahora necesitamos indicar los elementos que componen el epub, el orden de lectura, metadatos y la estructura global y para ello utilizaremos el estándar Open Packaging Format (OPF).

El primer paso es crear el fichero container.xml en la carpeta META-INF, este fichero suele ser igual en todos los libros electrónicos y su única misión es indicar en la estructura final donde se encuentra localizado el fichero que lista los contenidos del ebook.

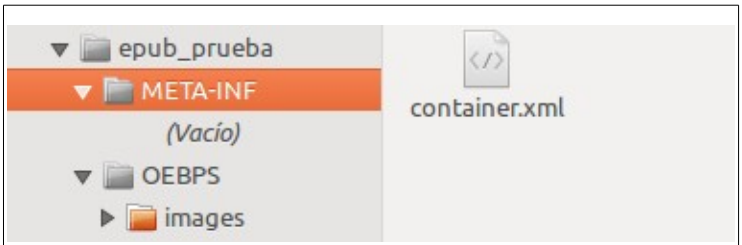


Figura 74: Localización del fichero container.xml en la estructura de carpetas del epub

La línea más importante del fichero container.xml es la ruta que indica donde está localizado el fichero content.opf.

```
<?xml version="1.0"?>
<container version="1.0" xmlns="urn:oasis:names:tc:opendocument:xmlns:container">
  <rootfiles>
    <rootfile full-path="OEBPS/content.opf" media-type="application/oebps-package+xml"/>
  </rootfiles>
</container>
```

Figura 75: Contenido del fichero container.xml

El fichero content.opf está situado en la carpeta OEBPS junto a los XHTML.

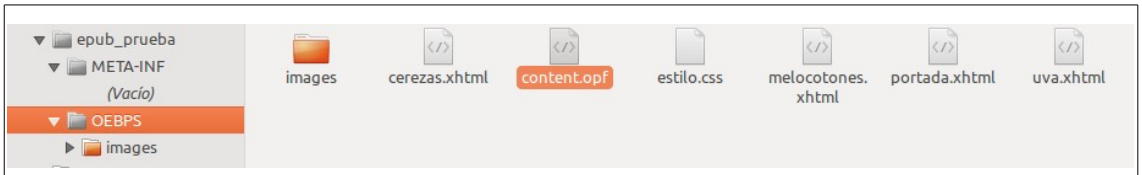


Figura 76: Localización del fichero content.opf en la estructura de carpetas del epub

Este fichero XML contiene una lista de todos los ficheros que incluirá el contenedor epub así como el orden de éstos, también se incluyen los metadatos que añaden información al contenido del libro electrónico.

La figura 77 muestra el contenido del fichero content.opf del epub que estamos creando.

```
<?xml version="1.0"?>
<package xmlns="http://www.idpf.org/2007/opf" xmlns:dc="http://purl.org/dc/elements/1.1/" unique-identifier="BookID" version="2.0">

<metadata xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:opf="http://www.idpf.org/2007/opf">
  <dc:title>Las frutas</dc:title>
  <dc:creator opf:role="aut">Miguel Ángel Ferrer</dc:creator>
  <dc:language>esp</dc:language>
  <dc:publisher>Miguel Ángel Ferrer</dc:publisher>
  <dc:identifier id="BookID" opf:scheme="UUID">fruta123456</dc:identifier>
</metadata>

<manifest>
<item id="ncx" href="toc.ncx" media-type="application/x-dtbncx+xml" />
<item id="css" href="estilo.css" media-type="text/css" />
<item id="titlepage" href="portada.xhtml" media-type="application/xhtml+xml" />
<item id="chapter001" href="cerezas.xhtml" media-type="application/xhtml+xml" />
<item id="chapter002" href="melocotones.xhtml" media-type="application/xhtml+xml" />
<item id="chapter003" href="uva.xhtml" media-type="application/xhtml+xml" />
<item id="image0001" href="images/portada.jpg" media-type="image/jpg" />
<item id="image0002" href="images/cerezas.jpg" media-type="image/jpg" />
<item id="image0003" href="images/melocotones.jpg" media-type="image/jpg" />
<item id="image0003" href="images/uva.jpg" media-type="image/jpg" />
</manifest>

<spine toc="ncx">
<itemref idref="titlepage" />
<itemref idref="chapter001" />
<itemref idref="chapter002" />
<itemref idref="chapter003" />
</spine>

</package>
```

Figura 77: Contenido del fichero content.opf

La estructura del fichero content.opf tiene 3 partes diferenciadas:

- Una zona de metadatos delimitada por la etiqueta <metadata> que incluye datos en el formato Dublin core[1] y al menos deben de aparecer los metadatos title, identifier y language.

- Una zona de manifiesto delimitada por la etiqueta <manifest> que debe de incluir una lista de todos los ficheros que forman parte de la publicación. Esta zona debe de contener al menos un elemento y el orden en el que aparecen los elementos no es importante. A cada uno de los elementos del manifiesto se le asigna un identificador con el atributo id.

- Una zone de spine que contiene las referencias a los identificadores de la zona de manifiesto que contienen datos del documento EPUB en XHTML. El orden de los elementos en esta zona si que es importante ya que se establece el orden lineal de lectura de la publicación.

Por último necesitamos crear un fichero toc.ncx en la carpeta OEBPS que está pensado para facilitar la navegación y proporcionar mayor accesibilidad al lector.

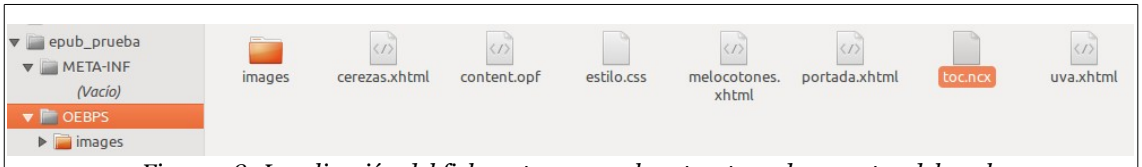


Figura 78: Localización del fichero toc.ncx en la estructura de carpetas del epub

Este es un fichero de control de navegación y muestra la estructura jerárquica de la publicación para que el usuario navegue por ella, permite posicionarse directamente en cualquiera de los elementos estructurales que estén indicados en el fichero de forma directa, pudiendo ser un capítulo, una sección o, incluso, una imagen.

1 http://dublincore.org/documents/2004/12/20/dces/

En la figura 79 vemos el contenido del fichero toc.ncx de la publicación que estamos generando en este tutorial. La parte importante de este fichero está delimitada por la etiqueta <navMap> y es donde indicamos la estructura del mapa de navegación de nuestra publicación.

```
<?xml version="1.0" encoding="UTF-8"?>
<ncx xmlns="http://www.daisy.org/z3986/2005/ncx/" version="2005-1">

<head>
    <meta name="dtb:uid" content="fruta123456"/>
    <meta name="dtb:depth" content="1"/>
    <meta name="dtb:totalPageCount" content="0"/>
    <meta name="dtb:maxPageNumber" content="0"/>
</head>

<docTitle>    <text>las frutas</text>     </docTitle>

<navMap>
    <navPoint id="titlepage" playOrder="1">
      <navLabel>  <text>Portada</text>  </navLabel>
      <content src="portada.xhtml"/>
    </navPoint>
    <navPoint id="chapter001" playOrder="2">
      <navLabel>  <text>Cerezas</text>  </navLabel>
      <content src="cerezas.xhtml"/>
    </navPoint>
    <navPoint id="chapter002" playOrder="3">
      <navLabel>  <text>Melocotones</text>  </navLabel>
      <content src="melocotones.xhtml"/>
    </navPoint>
    <navPoint id="chapter003" playOrder="4">
      <navLabel>  <text>Uvas</text>  </navLabel>
      <content src="uva.xhtml"/>
    </navPoint>
</navMap>

</ncx>
```

Figura 79: Contenido del fichero toc.ncx

7 Crear el contenedor epub

La última fase de la creación del epub es la de la creación del contenedor. Para ello necesitamos un nuevo fichero en la estructura con el nombre mimetype que utilizará el sistema operativo para averiguar el contenido del contenedor.

Este fichero mimetype debe de estar situado en la carpeta raíz de la estructura que hemos creado:

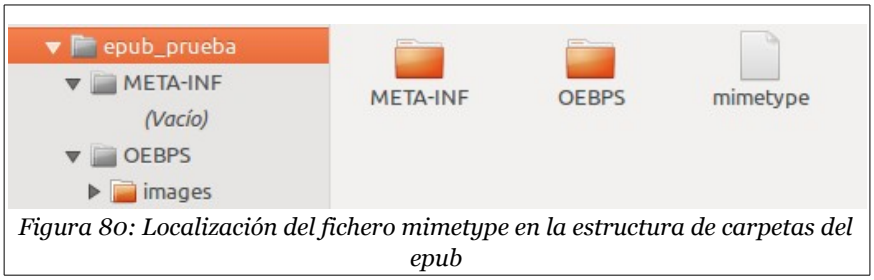


Figura 80: Localización del fichero mimetype en la estructura de carpetas del epub

El contenido de este fichero será texto en ASCII y contendrá una única línea con el contenido indicado en la figura 81.

Figura 81: Contenido del fichero mimetype

Ahora debemos de crear el contenedor de los ficheros que hemos generado, es buen momento para comprobar si están todos los que necesitamos y en la carpeta adecuada.

```
./mimetype
./META-INF
./META-INF/container.xml
./OEBPS
./OEBPS/estilo.css
./OEBPS/portada.xhtml
./OEBPS/uva.xhtml
./OEBPS/cerezas.xhtml
./OEBPS/toc.ncx
./OEBPS/content.opf
./OEBPS/images
./OEBPS/images/melocotones.jpg
./OEBPS/images/uva.jpg
./OEBPS/images/portada.jpg
./OEBPS/images/cerezas.jpg
./OEBPS/melocotones.xhtml
```

Figura 82: Relación de todos los ficheros contenidos en epub

Por último debemos crear el fichero epub que no se trata más que de un fichero zip al que se le ha cambiado la extensión por epub. Este fichero tiene una particularidad ya que todo su contenido está comprimido a excepción del fichero mimetype que no debe comprimirse.

Para hacer el proceso en linux ejecutaríamos las siguientes ordenes desde la carpeta que contiene la carpeta con toda la estructura de ficheros creada:

```
zip -v0X prueba epub_prueba/mimetype
zip -vr prueba epub_prueba/* -x prueba.zip mimetype
mv prueba.zip prueba.epub
```

Ahora ya podemos abrir el epub desde cualquier lector o desde un libro electrónico y comprobar que todo se ha generado correctamente.

Figura 83: Ejemplo visualización del epub creado

Figura 84: Ejemplo visualización del epub creado

Índice de figuras

Bibliografía


Codeigniter (2013). "Codeigniter User Guide version 2.1.4". http://ellislab.com/codeigniter/user-guide/toc.html 24/7/2013


Cosío Gutiérrez, Celia (2011). "Casos prácticos de UML". Editorial Complutense. Madrid.


CSS (2013). "CSS tutorial". http://www.w3schools.com/css/default.asp 24/7/2013


Epub 2.0.1 (2010). "Epub 2.0.1 specification". http://idpf.org/epub/201 24/7/2013


Grocerycrud (2013). "Documentation of Grocerycrud". http://www.grocerycrud.com/documentation 24/7/2013


HTML (2013). "HTML tutorial". http://www.w3schools.com/html/ 24/7/2013


IEEE 830 (1998).
"IEEE Recommended Practice for Software Requirements Specifications" Institute of Electrical and Electronics Engineers. 25 junio 1998.


javascript (2013). "Javascript tutorial". http://www.w3schools.com/js/default.asp 24/7/2013


Jquery (2013). "jquery tutorial". http://www.w3schools.com/jquery/default.asp 24/7/2013


Mobipocket (2013). "Mobipocket file format documentation page". http://www.mobipocket.com/dev/article.asp?BaseFolder=prcgen 24/7/2013


Netcraft (2013). "July 2013 Web Server Survey". http://news.netcraft.com/archives/2013/07/02/july-2013-web-server-survey.html 24/7/2013


Netmarketshare (2013). "Desktop Top Browser Share Trend and screen resolutions". http://www.netmarketshare.com/ 24/7/2013

PHP (2013). “PHP tutorial”. http://www.w3schools.com/php/default.asp 24/7/2013

PHPmyadmin (2013). “phpMyAdmin Official Documentation”. http://www.phpmyadmin.net/home_page/docs.php 24/7/2013

Schmuller, Joseph (2001). “Aprendiendo UML en 24 horas”. Prentice Hall. Madrid.

Tcpdf (2013). “Documentation of Tcpdf PHP library”. http://www.tcpdf.org/examples.php 24/7/2013

Wikipedia (2013). http://es.wikipedia.org/ 24/7/2013