UNIVERSIDAD DE EXTREMADURA

ESCUELA POLITÉCNICA

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería Informática en Ingeniería del Software

Trabajo Fin de Grado

"Modernización de la gestión y visión de una revista digital"

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería Informática en Ingeniería del Software

Trabajo Fin de Grado

"Modernización de la gestión y visión de una revista digital"

Autor: David Morales Chaparro

Tutora: Elena Jurado Málaga

ÍNDICE GENERAL DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

Este trabajo comprende un nuevo enfoque de una revista digital. Para ejemplificar esta propuesta, se ha usado como modelo RevIbe.

"De acuerdo con los objetivos establecidos para el desarrollo del Campus Virtual Latinoamericano (CAVILA), la Revista Virtual Iberoamericana de Ciencias Sociales (REVIBE) se constituye como plataforma académica digital que aspira a crear un espacio científico propio sobre la identidad cultural educativa iberoamericana. Con este fin, sus contenidos se sustentan en la apertura de nuevas perspectivas y líneas de investigación interdisciplinar, de acuerdo con los presupuestos renovadores que potencien nuevas visiones sobre el mundo americano. Los trabajos deberán ser originales e inéditos. La periodicidad es anual." — *RevIbe*

Se propone un nuevo diseño con unas prestaciones diferentes y una implementación que las plasma. Esta lectura actualizada del sitio web, enfocado a la divulgación de números formados por artículos, presenta una transformación del antiguo modelo de la revista. La visión original consiste esencialmente en un repositorio de archivos que se pueden descargar. Este proyecto ofrece una vista que simula de manera sobria la estética real de una revista física como las que se pueden adquirir en un quiosco, con el fin de que el contenido literario de los números sea el del sitio web, en lugar de una colección de ficheros. La presencia gráfica de esta simulación consiste en una visualización de las páginas del número como si de un libro abierto se tratase. Al desplazarse a una página adyacente, se produce una animación que simula el paso de una página, tal y como se hace en papel. Además, se permite a los autores añadir audios y videos en los artículos para que el lector pueda reproducirlos.

La metamorfosis de los artículos, cuyo formato es pdf, en componentes textuales, gráficos... que encajan en componentes html y forman así la nueva estructura de la web es la clave central de esta propuesta.

1. INTRODUCCIÓN

Las revistas digitales son la versión electrónica de la revista impresa [103, 107]. Actualmente, esta nueva versión se presenta en internet, en forma de página web. Algunas revistas digitales son réplicas de su versión impresa, manteniendo el concepto de página en la misma, y otras se han desprendido de eso y despliegan sobre una página web un contenido diverso.

Las prestaciones del mundo digital permiten a las revistas digitales utilizar multimedia, a los usuarios descargar contenido o a suscribirse digitalmente… La revista impresa sí puede incluir fotografías, pero no video, ni audios, ni enlaces interactivos, por ejemplo.

RevIbe es una revista digital de ciencias sociales de la que aquí se propone una idea diferente que aproveche el potencial de la misma.

RevIbe tiene una estructura clásica. Entre las secciones se encuentran la hemeroteca de artículos, el número actual, el comité editorial, noticias... La periodicidad de la revista es anual y los números se ordenan en pdfs que se pueden descargar.

La idea es centrar la revista digital en que la presentación de los artículos sea atractiva con la animación de pasar la hoja, y que éstos puedan incluir videos y audios, además de que la revista pueda visualizarse en distintos navegadores y dispositivos.

Hoy en día la claridad y la brevedad son claves en el mundo informático. Cualquier medio publicado en internet, servicio o aplicación ha de ser muy rápido, muy fácil de entender y tener mucha accesibilidad a todos sus puntos clave desde casi cualquier lugar. En caso contrario, el usuario lo considera intolerable y lo abandona. Por eso mismo en este trabajo se va a proponer que los artículos sean lo primero que el usuario encuentre al entrar en la web.

Una motivación clara que mueve este trabajo es el de la importancia de la representación de los datos hoy en día: presentar un servicio con elegancia y con flexibilidad para el usuario.

1.1. Objetivos

Por una parte, los objetivos principales son:

- Ser una revista clara, sencilla de visualizar y animada.
- Ofrecer mayor expresividad a los artículos, permitiendo añadir audios y videos reproducibles y que las imágenes sean ampliables.
- La revista sea vista desde distintos navegadores y plataformas sin problemas.

Y por otra parte, los objetivos secundarios son:

- Que los artículos de un número estén encadenados.
- Que se pueda leer un artículo concreto sin necesidad de leer el número entero.

Esta web apuesta por una tendencia actual muy popular en internet [111, 112, 113, 114], aunque no tanto entre las revistas digitales [105], y es el llamado estilo minimalista. El minimalismo [109, 110] evita las formas complejas, y trata de explicar conceptos y representar formas con el mínimo número de detalles posibles. Las estructuras minimalistas tienen muy pocos elementos que son sencillos de entender. La clave es que la web sea representada a la perfección sin distracciones, sin que el estilo o la claridad se vean mermados.

2. ESTADO DEL ARTE

En este apartado se hace una división entre el estudio del estado del arte actual y una crítica del mismo. El estudio está relacionado con las revistas digitales, entre las que se encuentran diversas propuestas artísticas y estratégicas.

Por una parte, en este primer apartado se analizan revistas electrónicas de todo tipo, con el fin de examinar su estructura, sus ideas de presentación de contenido y aspectos relacionados con la experiencia del usuario; por otra parte, las que poseen contenido multimedia diverso y en las que predomine el texto y tengan enfoques culturales o divulgativos.

RevIbe es una revista digital de investigación, y las revistas de este tipo que se pueden encontrar en internet guardan mucho parecido con RevIbe: los números y sus artículos se disponen enlistados, con referencias bibliográficas y citando a sus autores. En algunos ejemplos, el texto de los artículos forma parte de la página web, y en otros los artículos son documentos descargables (en RevIbe son pdfs descargables).

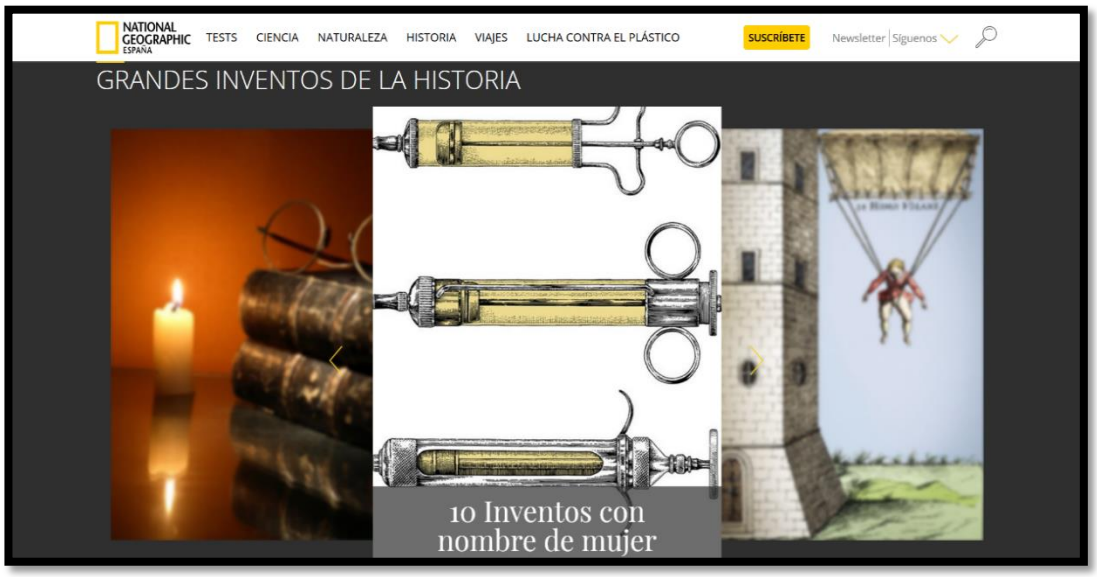
2.1. Estudio de unas revistas actuales

En los siguientes apartados se describen brevemente algunas revistas digitales relevantes.

2.1.1. National Geographic

National Geographic ya era una revista en papel antes de que existiese su versión electrónica. En ésta no sólo se aprecia una transformación del esqueleto tradicional físico a uno más frecuente en Internet hoy día, sino que además traspasa fronteras y consigue tener su propia personalidad frente a la versión original, con artículos diferentes, imágenes ampliables, enlaces a contenido, vídeos...

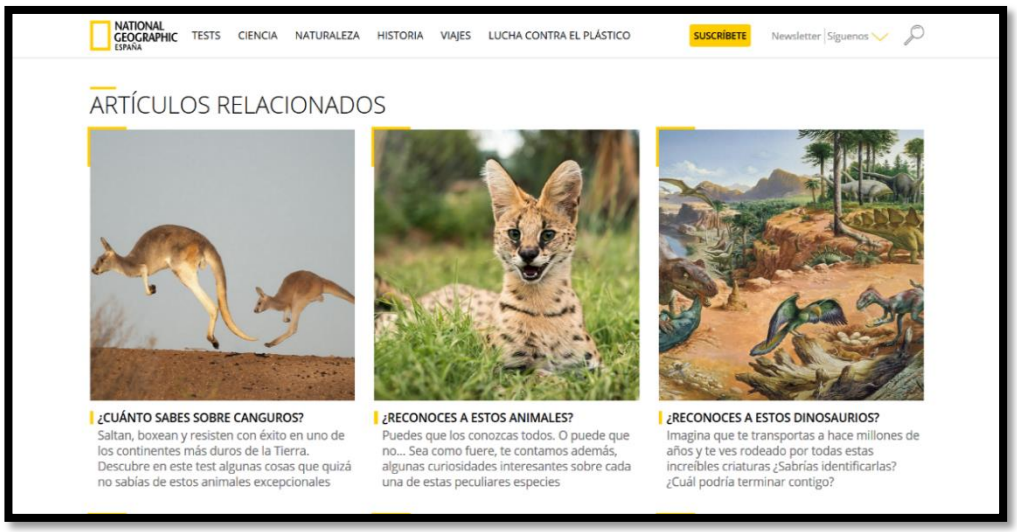
Figura 2.1. Vista de la revista de "National Geographic" (1)



En la figura 2.1. se puede ver la portada, que es ocupada principalmente por imágenes, y la cabecera.

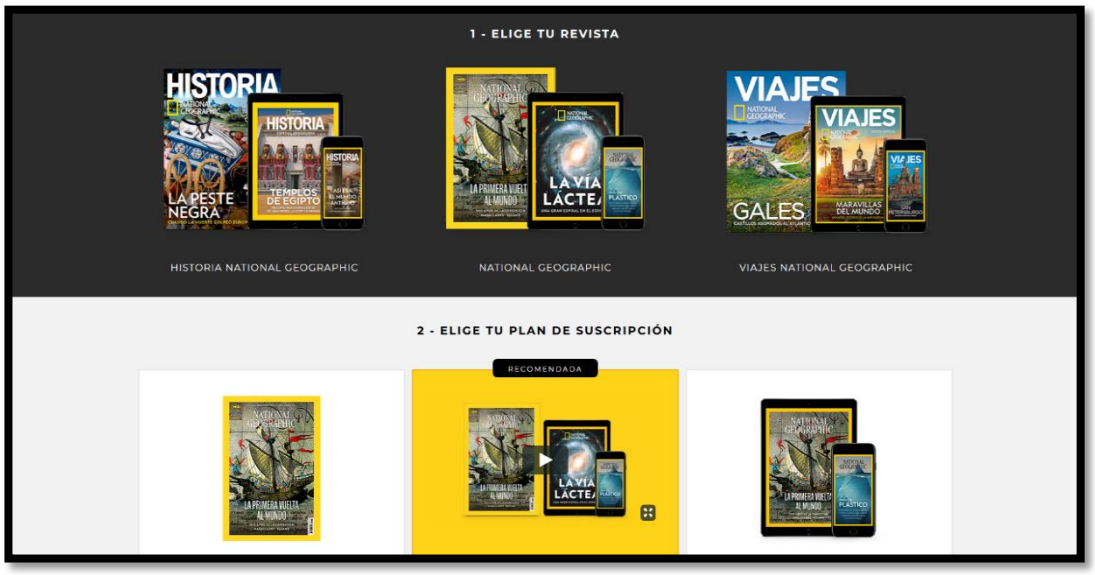
Este tipo de revistas digitales acaban por ser en muchos casos una extensión tan flexible y modificable que parecen la página web principal, y el medio físico la adaptación, cuando es lo contrario.

Figura 2.2. Vista de la revista de "National Geographic" (2)



La revista de National Geographic centra su estética narrativa en las fotos, como puede verse en la figura 2.2., que enfatiza el artículo con una portada. La forma más utilizada en esta web es la de artículo, que consiste en una columna central de texto con imágenes intercaladas. Su página principal despliega el contenido en secciones: actualidades, artículos destacados e invitación a suscribirse a la revista física.

Figura 2.3. Vista de la revista de "National Geographic" (3)



National Geographic invita al usuario a suscribirse a su revista física, pero no priva de contenido en la red a quien quiera verlo. Esto pone en perspectiva a National Geographic: su adaptación electrónica parece ser más un anticipo para captar a nuevos suscriptores. Como puede verse en la figura 2.3., puede elegirse entre diferentes versiones de la revista.

2.1.2. Espacio I+D

Figura 2.4. Vista de la revista de "Espacio I+D"

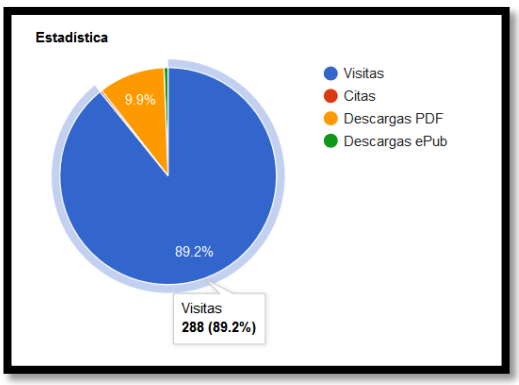
Por otro lado, está Espacio I+D, una revista de divulgación científica

multidisciplinaria. Como puede verse en la figura 2.4., la tónica general es presentar el contenido en forma de blog: cabecera para navegar entre las secciones, artículos mostrados de manera resumida en forma de lista, y una cadena lateral de widgets o enlaces de interés, sobre la web, documentos relacionados, un buscador o un sumario.

Figura 2.5. Vista de un artículo en "Espacio I+D"

La estrategia principal de esta revista es la de ser un repositorio de contenido etiquetado, organizado y explorable. En su apariencia, parece un tablón misceláneo que no sigue ninguna corriente específica en su tema, pero sí es riguroso en su forma: no presenta demasiadas secciones y su presentación es sencilla.

Figura 2.6. Gráfico de estadísticas del artículo en "Espacio I+D"



Las publicaciones se pueden descargar en EPUB y en PDF, se pueden citar y compartir por Twitter o por Facebook, y hay unas estadísticas ("figura 2.6.") que

indican las interacciones de los lectores con estas características.

2.1.3. Discover Magazine

Figura 2.7. Vista de la revista "Discover Magazine" (1)

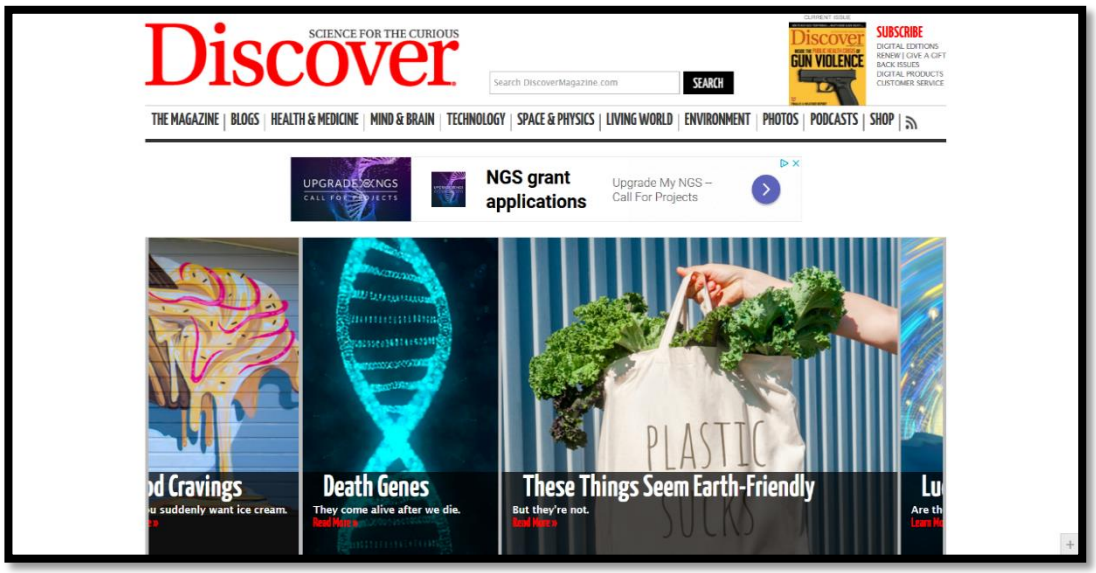
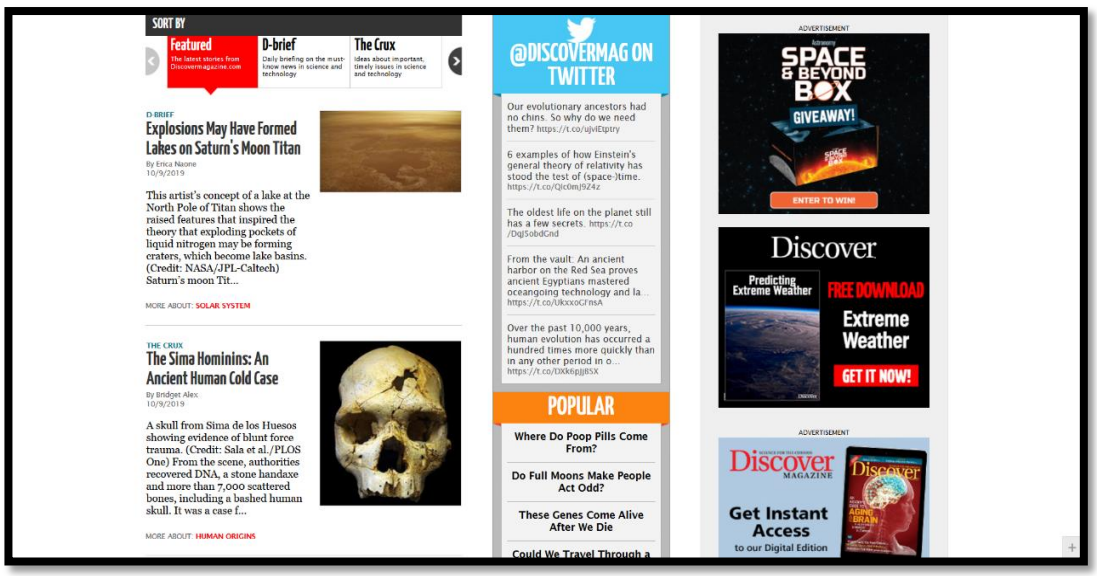


Figura 2.8. Vista de la revista "Discover Magazine" (2)



La revista Discover Magazine tiene versión digital e impresa. Lo primero que se observa de la web es una invitación a suscribirse a la revista (figura 2.7., arriba a la derecha).

Entre las secciones hay una que habla sobre la revista en sí, otra que tiene blogs que

agrupan artículos, algunas que tratan de ciencia, tecnología… En general, es una revista científica, enfocada a las ciencias tecnológicas y las de la salud.

La web gestiona el contenido con orden, a pesar de tener muchos bloques de contenido en su estructura de lectura. La parte principal de la estructura agrupa por un lado una columna de artículos o noticias, y en otra el acceso a redes sociales ("Figura 2.8."). Los artículos se pueden leer fácilmente y en algunos se observan videos o imágenes.

2.1.4. Music Connection Digital

Figura 2.9. Vista de la revista "Music Connection Digital"

Music Connection Digital es una web que integra literalmente una revista digital. No es un concepto literario en sí mismo, ni la web se centra en ser solo la revista, sino que es una especie de blog sobre música que trata de hablar indirectamente de la revista, como con "novedades de la música" y "reviews". En otro apartado, puede leerse la revista como si fuera un lector de pdf, con botones y elementos que permiten navegar hasta cualquier página ("Figura 2.9.").

Sin embargo, no se puede seleccionar el texto, porque posiblemente este lector animado sea de imágenes de las páginas del pdf. Se puede descargar ese pdf y visualizarlo en un lector de pdf personal, donde si puede seleccionarse el texto. Las páginas además están animadas y se mueven al pasar sobre ellas, como se hace en papel.

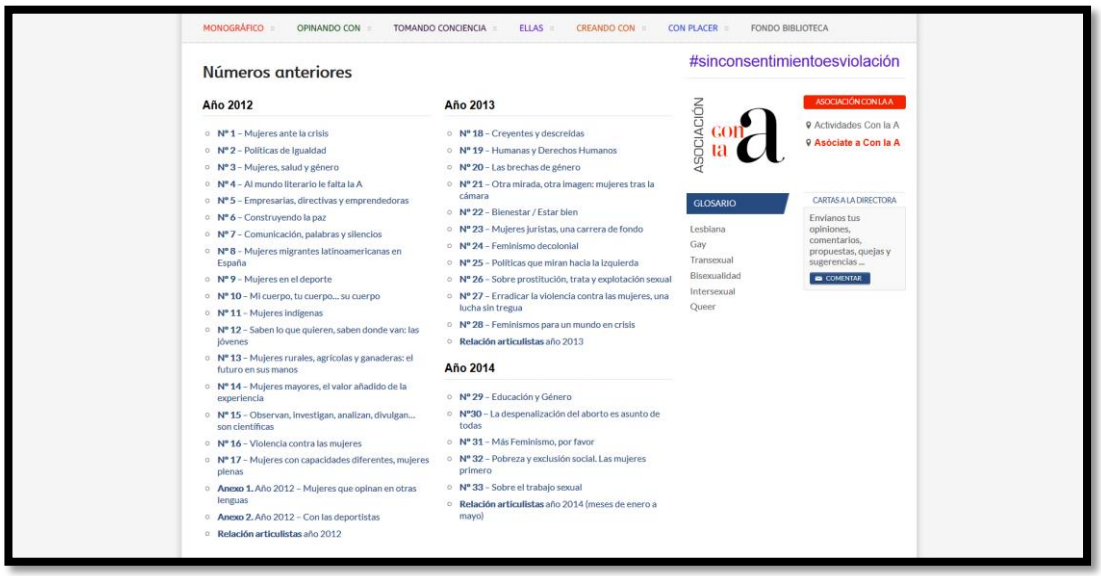
2.1.5. Con la A

Figura 2.10. Vista de la revista "Con la A"

Con la A es una revista digital en forma de blog con monográfico, artículos de opinión y artículos divulgativos y activistas, con carácter feminista y gestionado únicamente por mujeres (figura 2.10.). La temática por tanto es moderna y multidisciplinar, ya que al ser una motivación social, todos los campos tienen cabida.

El número del momento de la revista es el que viste la portada, es decir, los artículos y publicaciones visibles son las pertenecientes al número actual. La forma de presentar el contenido de la página es por tanto el de hacer que todos los elementos pertenezcan al número que se está "publicando". Se observa en la cabecera de la web cuál es el número del momento, y se pueden examinar cuáles fueron los números anteriores en una sección que se llama "Números anteriores".

Figura 2.11. Vista de números anteriores de la revista "Con la A"



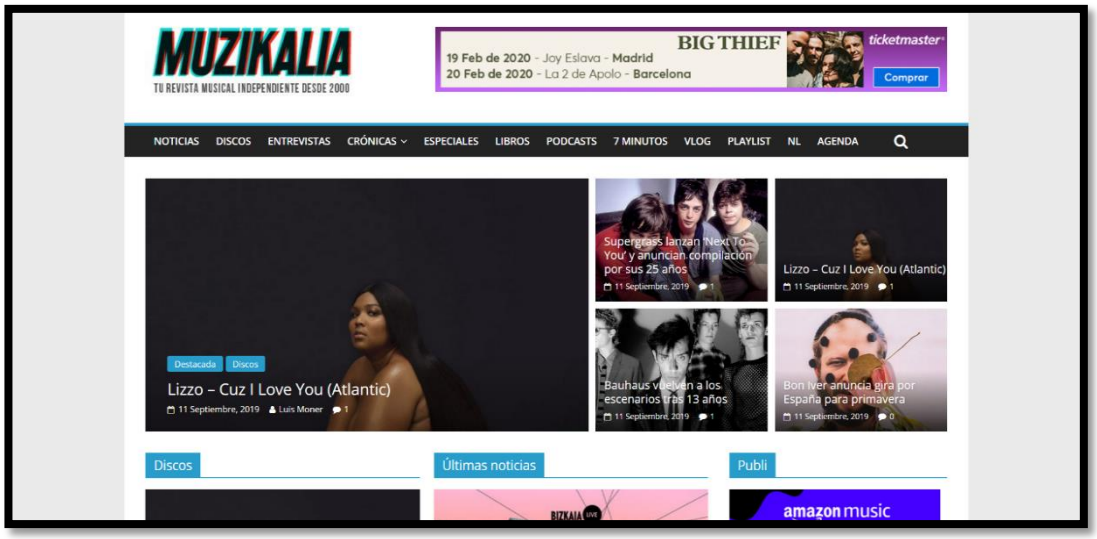
Normalmente, en otras revistas digitales lo que se puede encontrar es una serie de artículos diversos que se van publicando con el tiempo. Esta guarda un parecido con RevIbe, al tratarse de una revista que publica números en un período de tiempo, como pueden verse listados en la figura 2.11.

El aspecto de la revista es el de periódico online, que centra sus artículos por fechas y los presenta como un blog y los anuncia como si fuesen noticias. Lo que da a este blog la apariencia final de periódico es la multiculturalidad en las secciones y la mezcla entre actualidad y pensamiento, expresado en forma de críticas, opinión y prensa. Tiene sentido, pues la revista física es similar al periódico, aun sin dirigirse al mismo público. Este concepto se difumina en la revista electrónica, que tiene un concepto más amplio y ya no solo imita a la revista tradicional en medios electrónicos, con rigurosidad de publicación de números o temáticas centrales. Las revistas electrónicas en la actualidad también pueden ser blogs, en la que las entradas o los períodos son el número.

Sin embargo, el concepto claro de revista digital, o al menos el más orgánico, se ve reflejado perfectamente en esta web.

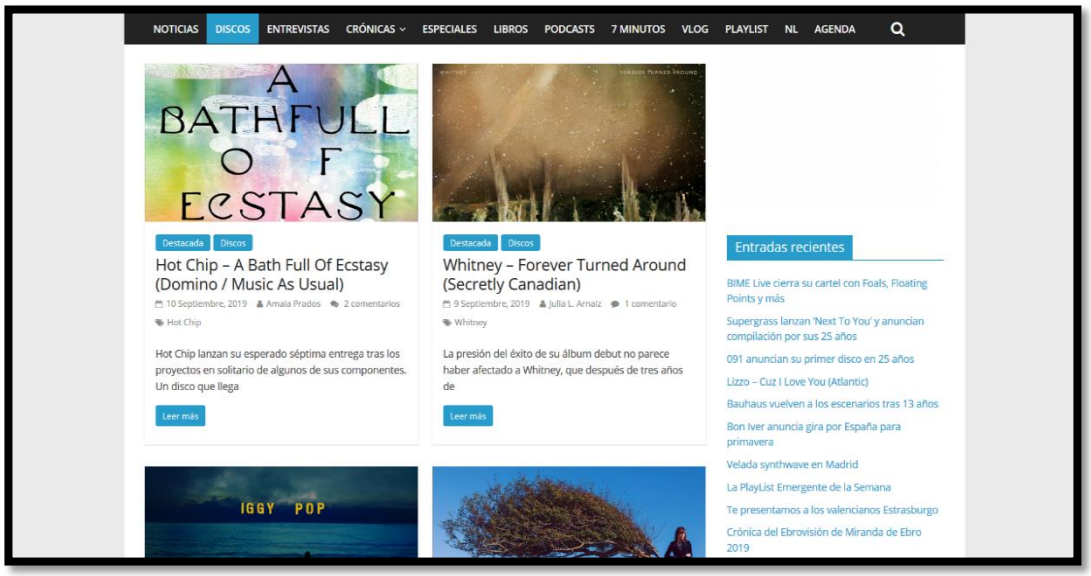
2.1.6. Muzikalia

Figura 2.12. Vista de la revista "Muzikalia"



Muzikalia es otro buen ejemplo de revista digital que, bien por desvirtuación del término o por una reinterpretación del término, no se parece tanto a una revista digital por definición, como la previa de "Con la A", y más bien parece una web más grande, porque tiene secciones que tratan sobre discos, video blogs, podcasts, playlists (figura 2.12.)… Va sobre música, actualidad, conciertos y prensa indie musical, underground o de personas que siguen este estilo de música o en concreto tratan de ser independientes, en sus producciones o en los procesos de éstas.

Figura 2.13. Vista de la sección de Discos de la revista "Muzikalia"



En la figura 2.13 puede observarse la sección de discos, que en cuadricula muestra publicaciones sobre discos de los que hablar.

La revista parece más una agrupación de blogs, con una agenda de conciertos, una lista de podcasts... y resulta compleja. Lo que más hace que recuerde a una revista es su carácter divulgativo y que esté enfocado a un público concreto. No encaja tanto en el modelo tradicional de revista, pero puede considerarse una lectura contemporánea de la misma, que empieza a ser habitual.

Estos ejemplos han sido escogidos por ser considerados cada uno de ellos representaciones de las revistas digitales comunes. El análisis general demuestra carencias y aciertos que a juicio de autor se exponen en forma de crítica, que también sirve de conclusión.

2.2. Crítica al estado del arte

En general podemos encontrar webs muy recargadas, pero el estilo de moda es el minimalismo, la claridad y la elegancia. El minimalismo no impide que la web incorpore diversas funcionalidades, pero siempre es preferible que éstas estén integradas de manera que no perturben el objetivo de la web, y que sean sencillas y no la hagan menos comprensible.

Contextualizando, la revista tradicional y los fancines han sido medios de mayor creatividad y concreción que cualquier otro medio de prensa. Sin embargo, su adaptación, la revista electrónica, presenta una diferencia clara: en muchos casos estas páginas web son esencialmente blogs de opinión e investigación, pero no revistas. Aunque claro, hay que aceptar que las revistas digitales puedan incluir nuevas características y secciones porque se sirven de la informática, que es principalmente el objetivo de hacer este formato en su versión digital. En cualquier caso, el comentario de este apartado se enfoca en cómo ha podido desvirtuarse o reinventarse el concepto de revista, y qué puede aportar.
En algunos casos, las revistas digitales se parecen demasiado a un blog y desaprovechan la naturaleza de la revista, en otros se consigue que la revista desaparezca entre demasiados detalles, que son añadidos posiblemente desde la

buena intención de dar a la revista más diversidad de funciones, mayor utilidad y mayor contenido, pero en otros casos tienen una forma interesante y bien dispuesta, como en el caso de Con la A, o la revista Discover.
Hay que asumir, por otra parte, la fusión de estilos, y que quizá la revista digital deje de ser tan diferenciada en su forma con el paso del tiempo. Se respetan las propuestas estéticas observadas en pro de que se pueda argumentar cuál podría ser una buena postura estilística para RevIbe.

Aun así, hay revistas que abandonan la complejidad para ser claras y directas.
Por ejemplo, en Muzikalia, a primera vista, la estética guarda un cierto orden. Después, como concepto global, el hecho de que esté dividida en secciones muy diferentes entre sí, que incluso abarcan nuevamente estilos agradables pero también diferentes, hace que la personalidad de la web se pierda. Es un buen intento de hacer una revista que enfoque varios temas relacionados, pero que en realidad son lejanos en cómo se tratan, como el uso de una sección de noticias de música y otra de agenda de conciertos, que no son tan parecidos.
Esto no quiere decir que la revista no pueda ser variada, o tener varias perspectivas, siempre que no acabe pareciendo un entramado de blogs. En el caso de MusicConnectionDigital, una revista que parece la impresa está insertada dentro de la revista electrónica. Es decir, la web es una revista electrónica… y luego presenta un lector de un pdf. Podría ser interesante que este lector de pdf dejase adelantar lo que ofrece la revista en físico, para invitar a la suscripción, ser un medio publicitario… en lugar de repetir un poco lo que ya se encuentra en las secciones de la web.
En general, puede resultar una complicación que el usuario que lee la revista se encuentre con demasiada información. La estrategia que a veces se abarca para evitar este escenario es la de desplegar menos información, y otras veces es la de presentar la información con orden, mesura y estética. Una revista que parece poner en práctica este pensamiento es la de National Geographic, que maneja mucha información, pero está plasmada con sencillez, y ocupa un espacio grande en la pantalla y no hay elementos que interrumpan su visualización. En la lectura de un artículo en esta revista, se encuentra casi siempre un texto sencillo dividido por fotografías relacionadas con la cuestión. La página principal presenta diversos contenidos, como artículos destacados y anuncios, pero para relajar esta presentación, se muestra

menos contenido. En el enlistado de los artículos de la revista, National Geographic sólo muestra un resumen. Con esto y más, hace que la página principal exprese el contenido de la web sin saturar su visualización.
El internauta moderno es muy impaciente, porque quiere claridad y poder controlar todo desde el principio, que se pueda ir casi a cualquier sitio de la web en todo momento, y que se pueda hacer en poco tiempo. En ese sentido, tanto esta revista como otras, en general, tratan de lograr esta difícil tarea de ingeniería social e informática con resultados tibios. Algunas logran ser bastante plásticas, pero casi todas caen en el exceso. Una visión agradecida y extendida popularmente es la sobriedad tanto en estilo como en estructura. La revista de Con la A no es compleja en su esqueleto y es sencilla en su forma. Sin embargo, una buena idea que no se plantea tanto es la de presentar estructuras similares entre las propias páginas de la web, para que aunque discutan temas diferentes, el usuario pueda conocer la forma de la web desde el principio. Es decir, que las secciones de la web se parezcan en la forma, aunque varíen el contenido. La creatividad a veces es un arma de doble filo, y la inquietud por abarcar múltiples herramientas puede resultar un peligro si no se usan con cuidado.

Al navegar a través de una revista, puede ser interesante que la lista de sus secciones o artículos sea de tal manera que sus elementos posean un tamaño similar sobre la pantalla. Esto puede recordar a las estanterías, a las cajas de exposición, a las galerías de arte... Es la temática del blog o del foro, una forma muy común de representar la información. La web se compone tanto de estos pequeños elementos que hacen que la vista sea casi la de una cuadrícula o un panal, cuando luego abrir cualquiera de estos enlaces despliega una página o varias de texto, fotografías, etc. Al ser metáforas sencillas y familiares, el lector de la revista puede sentirse cómodo con su uso.
La informática abre una puerta tecnológica que permite al autor diseñar su propia experiencia. Esto genera nuevos estilos, y se vuelven a interpretar los conceptos conocidos, en este caso, de la prensa y de las webs. A nivel estético o artístico, todas las propuestas tienen su sitio por definición. La crítica a esta propuesta vendrá desde el punto de vista social, y en cierta medida, desde el técnico, tanto en cuanto afecte a su parte lógica. Tolerar el arte exige crítica. A nivel de ingenio, técnica, estructura y lógica, el arte tiene que convivir con las formas y diseños que propone el "departamento" de desarrollo. La tecnología ha de permitir que el arte existe, y el

arte ha de evitar interponerse en la tecnología.
Por otra parte, si tanta es la diversidad que permite esta modernidad de auge tecnológico, ¿por qué las webs son parecidas? ¿Por qué todas las revistas digitales son tan similares? La revista digital que pretende salirse de la norma, o al menos dar un enfoque un poco más personal, normalmente guarda parecido con una web típica. Las revistas, tradicionalmente, son muy concretas. A veces tanto, que son amplias, porque profundizan, aunque sin salirse de la temática. La parte negativa de esto se refleja en el momento en el que la revista digital no amplía sus niveles conceptuales, sino que llena la web, en algunas ocasiones, de recursos de menor utilidad, que pueden alejarse de la temática, o que pretenden convertir a la revista en una navaja suiza virtual.

Podría destacarse Con la A, por ser concreta, expresiva y ordenada. Así como National Geographic, por tener ventajas similares, además de ser elegante.

Sobre las herramientas que la web ofrece y el usuario puede utilizar durante su experiencia, todas las revistas presentan algunas parecidas, por ejemplo: los buscadores dentro de la web, las bibliografías, las hemerotecas, las bibliotecas de contenido...

Es cierto que es tan subjetivo el estilo estético como el tipo de contenido, pero no dejan de ser partes que han de existir en el mismo sistema. Una revista que habla de música no necesita tener música en sus páginas sonando para crear un entorno, pero siempre será una cuestión artística que se habrá de plantear, y que tendrá mayor o menor sentido dependiendo del enfoque global.

Se recalca la importancia de entender al autor o autores del contenido, de percibir los posibles intereses del lector al que se dirige y, tras ello, enfocar un diseño acorde.
Lo más importante debería ser respetar al autor y hacer la experiencia del usuario lo más sencilla posible.

Tabla 2.1. Características de las revistas digitales analizadas

Revista digital	Admite suscripción	Existe en papel	Presenta claridad	Monotema	Tiene más secciones	Tiene un lector animado
National Geographic	SÍ	SÍ	SÍ	NO	SÍ	NO
Espacio I+D	NO	NO	NO	NO	SÍ	NO
Science Magazine Digital	SÍ	NO	SÍ	NO	SÍ	NO
Music Connection Digital	SÍ	SÍ	SÍ	NO	SÍ	SÍ
Con la A	SÍ	NO	SÍ	NO	SÍ	NO
Muzikalia	NO	NO	NO	NO	SÍ	NO

Detalle sobre la tabla:

- Admite suscripción: se refiere a si el lector puede suscribirse al documento para poder estar pendiente de novedades. La importancia de esta característica puede ser mayor en el caso de que la revista digital presente versión en papel.
- Existe en papel: trata de aportar información sobre si es una adaptación o una revista enteramente de origen digital.
- Presenta claridad: se indica "sí" si no hay demasiados elementos en la visualización de la página, las secciones son accesibles y está claro en todo momento de qué va cada una. Es importante para resumir si la organización es ejemplar para este trabajo o no.
- Monotema: indica si la revista trata solo un tema o se centra en varios. La importancia de esto reside en conocer las dificultades que la revista sostiene, intuyéndose que pudiera ser más fácil reunir facetas diversas que una sola. Todas aportan varias temáticas, y agrupan los blogs y los artículos en secciones.
- Tiene más secciones: indica si, además de centrarse en los artículos, tienen más secciones, como pueden ser podcasts de radio, foros de opinión, galería de fotos, calendario, hemeroteca…

- Tiene un lector animado: indica si posee algún lector de pdf de artículos de la revista que se pueda animar como si fuera una revista impresa o parecida. Es importante pues es la estética central de este trabajo.

3. PROPUESTA DE SOLUCIÓN

En esta sección se comentan brevemente las motivaciones de la solución propuesta y se repasan las ideas principales, además de revisar el punto de vista del autor sobre el modelo de revista digital de Revibe. Después, se describen las estrategias contempladas y los análisis pertinentes. Seguidamente, se habla sobre las herramientas usadas, se hace un repaso sobre la arquitectura del sistema, y se detalla la especificación de las soluciones (implementaciones) más importantes que tiene el código.

Hasta el momento, Revibe es en esencia un repositorio de artículos en formato pdf organizados por número que el usuario puede descargar. Esto los convierte en elementos pasivos en el entorno web.

La solución propuesta se basa en transformar el contenido de los artículos de Revibe en el protagonista de la revista digital, en lugar de ser un elemento más de la web.

Se plantea un cambio de visión que propone que los artículos de los números de la revista estén desplegados en la web y se puedan leer, como si se tratase de un medio divulgativo. Además, se puede hacer click en las imágenes para que se amplíen en efecto *popup*, y se permite a los autores la inclusión de audios y videos, que pueden reproducirse en la revista.

Con esta sencillez y sin más detalle, podría decirse que la herramienta que aporta esta solución ya existe: el visor de pdf. Si no se plantea un cambio de formato y se mantiene el usado hasta ahora en la revista, este visor podría ser la herramienta perfecta, pero el objetivo está lejos de transformar la revista solamente en un visualizador, porque es limitado y carece de dinamismo. Como propuesta es pobre y no tiene tanto interés: se necesita un contexto, algo parecido a una web con secciones de contenido, publicaciones, un cierto orden…

Se propone en su lugar una web que sea una metáfora virtual de una revista física, en la que el lector puede leer las páginas como si se tratase de una revista impresa [105], pasando las páginas hacia atrás o hacia adelante, pudiendo seleccionar el texto y ver sus imágenes. Esto añadiría estética y un concepto con más personalidad.

En esencia se propone una transformación del visor de pdf a otro más responsivo y que permita generar esa metáfora visual, pues la idea se traza alrededor de la expresividad de la información. Como se observa en el estudio del estado del arte, las revistas digitales abarcan el "estilo periódico" (publicaciones enlistadas y cronológicas) y el divulgativo o científico (secciones web, agrupaciones por autores, temáticas, galerías de fotos…).

El resultado final consistiría en una mezcla de estilos y autores, no sólo a nivel documental sino también tipográfico. La variedad de contenido ya la contempla o asume la dirección de la revista al planificar números anuales y autores dispares. Si la revista global fuese una idea previa al desarrollo, que podría adoptarse en este trabajo o partir de él pero que no se hace, podría especificarse una serie de normas para las autores para conseguir este estándar estilístico. Sin embargo, es idea del desarrollador transformar los artículos en una revista conceptual. Es importante entender que la revista es artísticamente flexible, debido a que parte de este trabajo se centra en el manejo de las situaciones concretas que produce la gestión informática de los artículos de la revista, debido a las características que poseen, como el formato pdf. Esto significa que se decide asumir el reto de crear esta revista con el contenido que ya existía. En caso contrario, los autores tendrían que recrear artículos pasados para que cumpliesen nuevas reglas que el desarrollador pudiese especificar, situación que se prefiere evitar.

Se asume trabajar con pdf, que es un formato de documentos digitales que es independiente del software, hardware o sistema operativo. Además de las ventajas que presenta el formato pdf, se acepta utilizarlo porque permite un estándar en la presentación de los datos. La construcción depende del autor, y no importarán las formas: el formato pdf es cerrado y con eso se construirá la revista.

3.1. Línea estratégica

A la hora de abordar la idea, se traza un plan de estrategias que comienza por la búsqueda de herramientas que permitan la visualización de pdf como si fuesen una revista física de verdad.

Se encuentran herramientas de pago y algunas gratuitas, la mayoría en forma de aplicación o software de terceros. Se opta por buscar algo similar a una librería de

código y se encuentra: turn.js [94]. En resumen, turn.js es una solución software que permite animar páginas de contenido html como si fueran hojas de verdad. Incluye las animaciones y posibilidades de zoom.

Esto solo resuelve la problemática de la animación de revista buscada, pero no la del despliegue de contenido pdf en html.

Se opta por la transformación del formato pdf o la muestra del mismo de una manera usable en la web, para lo que se buscan diversas herramientas pero se encuentran pocas soluciones. Se encuentra: pdf.js [87]. Esta librería permite leer los pdfs y transformarlos a html, pero haciendo que la imagen final sea exactamente igual que la del pdf, en otras palabras, respeta el formato de pdf. Con estas dos herramientas se trabaja para la construcción de la revista.

3.2. Análisis de las herramientas

A continuación se muestras dos tablas, en este orden: la tabla 3.1., que enumera los recursos aprovechados para el desarrollo de la revista digital, y la tabla 3.2., que enumera los desarrollados por el autor.

Tabla 3.1. Características de herramientas y entornos utilizados

Recurso	Tipo	Nombres	Uso
Fichero	Javascript	-jquery-3.3.1.min.js -pdf.js -pdf.worker.js -text_layer_builder.js -turn.min.js -turnpage.js -ui_util.js -rotate.js	Estos ficheros son librerías de extensión de javascript (jquery), y útiles para la lectura del pdf (pdf.js, pdf.worker.js), y para las animaciones de lectura (turn.min.js, turnpage.js, ui_util.js)
Editor de texto para código		-Sublime Text	Desarrollo de ficheros jsp y css.
Entorno de desarrollo integrado		-Microsoft Visual Studio 2019	Desarrollo de ficheros Javascript.
Entorno de desarrollo integrado		Eclipse JEE	Desarrollo de servlets y despliegue de servidor Apache Tomcat.

Como se documenta en la tabla, se usan librerías javascript para este proyecto.

Javascript, al ser un lenguaje ligero que permite una mayor interactividad del usuario con la web, es una buena opción para el modelo propuesto de Revibe en este trabajo. Se ocupa de animaciones complejas y populares de múltiples páginas web a lo largo de todo Internet. Son bien conocidas las animaciones de texto, movimiento de elementos, efectos de movimiento de los recursos, cambios de formato de la presentación de los datos... El uso de este lenguaje traslada la carga de trabajo de estas funciones a la "parte delantera" de la web.

En este proyecto se hace uso de jQuery [9], una librería de Javascript que simplifica soluciones, incluyendo la manipulación del html desde el código, así como la del css, entre otras utilidades. La librería turn.js y la pdf.js, como sus formatos indican, están implementadas en Javascript.

Además, se anota en la tabla el uso de un editor y un entorno:

- *Sublime Text*: se escoge un editor de texto que permite coloreado de sintaxis y búsqueda de elementos, entre otros recursos interesantes, para la codificación rápida y eficiente de los ficheros jsp y css.
- *Microsoft Visual Studio 2019*: práctico y fácil de usar, y permite una buena gestión del código, aunque se barajó el uso del entorno Eclipse por su uso durante el curso académico. Este trabajo se centra en codificación de scripts independientes, por lo que servía cualquiera entorno que permitiese una codificación cómoda.
- *Eclipse JEE*: usado para la implementación y despliegue del servidor Apache Tomcat. En este IDE se desarrollaron los dos servlets de gestión de peticiones HTTP del servidor.

Tabla 3.2. Características de recursos desarrollados

Recurso	Tipo	Nombres	Uso
Fichero	Javascript	-revista.js -bookshelf.js	Definición de la lógica de la revista digital.
	Java	-InitServlet.java -ReadServlet.java	Servlets java que gestionan las peticiones al servidor.
	JSP	-lecturaArticulo.jsp -revistas.jsp	Elaboración de páginas web. Usado para la construcción del prototipo web de Revibe.
	CSS	-body.css -estante.css -flip.css -footer.css -STYLE_principal.css	Definición de la representación gráfica de Revibe, con especial importancia el formato de la revista.

Nuevamente, se habla de Javascript, pero en esta ocasión de los desarrollados para este proyecto, que son "revista.js" y "bookshelf.js". "revista.js" es un fichero bastante extenso que centra la mayor parte de la lógica de la web y toda la de la revista animada. Es de esta dimensión porque gestiona la creación de los elementos html que soportan la revista para ser leída, la carga de documentos y la respectiva carga de cada página del documento en la web… con el añadido de que todas las funciones auxiliares que se precisan para ello, como se detalla en la Guía del Desarrollador, anexada. Por su parte, "bookshelf.js" es un fichero que gestiona la carga de números de la revista en la página de inicio, y la correcta carga del artículo escogido a leer. Es decir, en la página de inicio ("revistas.jsp") se pueden observar en horizontal las portadas de varios números de la revista RevIbe. El lector puede pasar el cursor del ratón por encima de ellos, y seleccionar si quiere leerlo, o si sólo quiere leer algún artículo del número. Esta gestión está centralizada en "bookshelf.js", que en inglés significa "estante para libros". También se habla de los ficheros .java, los servlets.

Para diseñar una página web que muestre la revista se precisa de jsp, y para aportar diseño y forma, css. Se genera un fichero "lecturaArticulo.jsp" y uno llamado "revistas.jsp". "revistas.jsp" despliega un menú que permite seleccionar el número a leer, escogiendo si se quiere leer entero o si se prefiere leer un artículo concreto. Este menú se muestra con una metáfora visual parecida a la de un estante, en la que se ven en fila horizontal los números seleccionables. "lecturaArticulo.jsp" es la página que permite leer el artículo seleccionado, o el número concreto, si se ha seleccionado leer todo el número.

Los archivos css de este proyecto se encargan de dar una estructura y un estilo a la web. "STYLE_PRINCIPAL.css" aúna las reglas de diseño general, e importa en su propio texto el resto de ficheros css. El estilo está dividido en ficheros para que sea más sencillo de desarrollar y esté organizado. Por ejemplo, "footer.css" está reservado para el estilo del footer de la web, la parte inferior de la misma, y "flip.css" tiene las reglas de estilo de las páginas de la revista animada, así como la definición de la parte gráfica de la animación de las esquinas del papel que se doblan al girarlas…

3.3. Arquitectura del software

En la arquitectura de este software hay tres bloques diferenciables: el de jsp y css por un lado; el de javascript por otro, es decir, el que da estructura a la web y el que le da una lógica; y el de servlets, que gestiona las peticiones HTTP.

El único fichero del "bloque de lógica" a destacar arquitectónicamente es el de "revista.js". Este código centra su uso en el contenido devuelto por la función *getDocument*, de la librería "pdf.js", que es la librería que maneja el *parsing* del pdf. El bloque principal de este script es la construcción de la revista como elemento HTML, la carga de contenido y funciones de gestión, carga y lectura, con sus consecuentes subfunciones. El funcionamiento de este script se comenta en su documentación interna, y se describe en el Anexo I Guía del Desarrollador, para su mejor entendimiento.

Tabla 3.3. Flujo principal de datos de la revista.js en la siguiente tabla

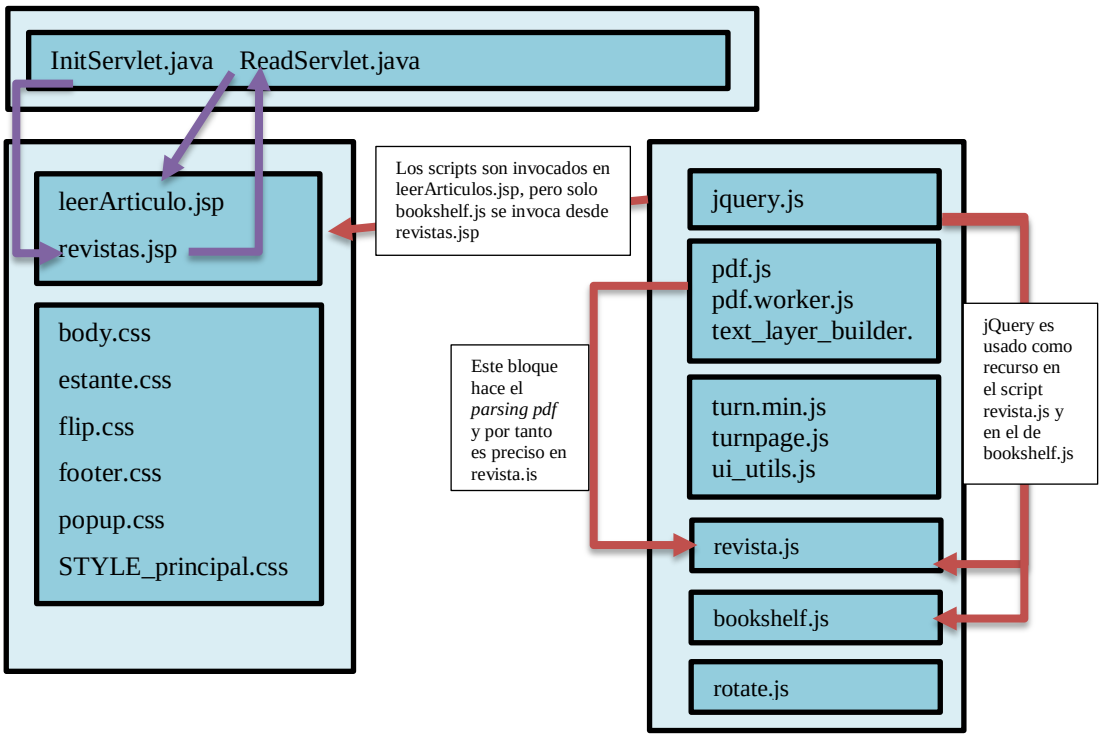
Paso 1	Declaración de estructuras de datos para almacenar los artículos. Carga de las direcciones de los artículos.
Paso 2	Obtención del primer artículo de la lista.
Paso 3	Generación de la revista.
Paso 4	Generación de la portada.
Paso 5	Carga del primer par de páginas del artículo.
Paso 6	El manejador del evento "Va a producirse un desplazamiento hacia otra página" define las cargas de las páginas del artículo a las que el lector se desplaza.
Paso 7	El manejador del evento "Se ha producido un desplazamiento hacia una página" comprueba si se ha llegado al final del número, caso en el que lo da por finalizado, y añade una estilística "tapa de cierre", como si fuese la contraportada de la revista.

Como se observa en la tabla 3.2., se generan las estructuras de datos y se cargan de contenido (Pasos 1 y 2), para después generar la revista animada, con su portada (Pasos 3 y 4), y después se produce la carga del texto en las páginas (Paso 5).

Los pasos 6 y 7 se refieren, respectivamente, a los eventos "turning" y "turned", que ayudan a desarrollar situaciones según si se desplaza hacia una página o si acaba de producirse ese desplazamiento. Un ejemplo de una de estas situaciones sería que sea obligado que haya texto sobre una página antes de dirigirse hacia ella.

Para mayor claridad, se expone a continuación un sencillo esquema que explica el orden lógico de los ficheros para el funcionamiento de la aplicación:

Figura 3.1. Diagrama de módulos



En la "figura 3.1." se observan dos cajas azules:

- La caja azul de la izquierda contiene:
    - Los jsp.
    - Los css.
- La caja azul de la derecha agrupa los ficheros javascript, agrupados por dentro por categorías, siguiendo el orden del esquema de arriba a abajo:
    - JQuery.
    - Archivos que gestionan el pdf.
    - Archivos que gestionan la animación.
    - "revista.js", que maneja la revista.
    - "bookshelf.js", que maneja la portada de la web.

- La caja de arriba agrupa los servlets. El InitServlet inicia la aplicación, redirecciona a la portada "revistas.jsp", y desde ahí, si se pincha un número, se va a ReadServlet, que redirecciona a "leerArticulo.jsp".

  Se indica (de arriba abajo) en la primera flecha que los scripts javascript se invocan desde "leerArticulos.jsp". Esto es así porque este fichero jsp es el que permite leer el artículo y necesita tanto del bloque de scripts que gestionan el Parser del pdf, como el que gestionan la animación. "bookshelf.js" solo es llamado desde "revistas.jsp". Este fichero jsp es el de la portada de la web. Obviamente, jQuery, al ser una librería utilitaria, es usada en todas partes. Las flechas a la derecha del bloque azul derecho indican esto último. Por su parte, la última flecha, a la izquierda del bloque azul derecho, habla de cómo el bloque del que nace dicha flecha, el de Parser de pdf, es usado en revista. El bloque bajo el de Parser pdf es el bloque de animación de la revista.

3.4. Soluciones softwares más importantes

Para tratar de manera breve y completa las partes más importantes de la implementación, se hace la división de la misma en bloques, con su correspondiente explicación.

3.4.1. Parser PDF: pdf.js

Este bloque transforma cada página del pdf en una en html. En la solución concreta del trabajo, se pueden leer la revista abierta de par en par, con una página a la izquierda y otra a la derecha. Sobre cada una de ellas se cargará cada vez una página del pdf que se estén leyendo.

Se toma la decisión de que las páginas del pdf se compongan de elementos svg [31, 32], que es un elemento html basado en un formato de gráficos vectoriales bidimensionales.

La tarea principal de pdf.js es transformar el documento pdf en los elementos html que el desarrollador prefiera, para lo que tendrá que indicarlo específicamente, como se indica en la Guía del Desarrollador.

Se decide por tanto usar el elemento SVG por hacer una imitación precisa del documento pdf. La metodología empleada consiste, con ayuda de la lectura de la documentación que existe por la red y que se anota en la bibliografía, en utilizar un método especializado de pdf.js llamado SVGGraphics, que genera una colección de elementos svg. Este método en particular devuelve esa colección en base a una página de pdf que se le otorga por parámetro. Según las indicaciones encontradas, lo que se ha de hacer es desplegar esta colección de manera anidada, es decir: se genera un "nodo padre", que es un elemento svg principal, y que representa a la página del pdf; seguidamente, se itera sobre la colección de elementos de tipo svg que otorga SVGGraphics, y se añaden "hijos" al "nodo padre". Estos elementos svg ya vienen con un subtipo específico gracias a SVGGraphics, siendo por ejemplo un svg de tipo imagen, o un svg:image, en caso de haber leído una imagen, o svg:text en caso de ser un segmento de texto. De esta manera se imita toda la estructura de la tipografía del pdf, y se agrupan jerárquicamente en svg de tipo g o svg:g. Estos últimos son elementos utilizados para agrupar elementos svg.

La principal complicación que esto presenta es que el formato svg es de difícil selección desde javascript, siendo en algunas ocasiones menos intuitivas algunas operaciones, como por ejemplo, la búsqueda de elementos en el documento html según un tipo concreto de elemento html. Esto complica algunas tareas que quizá serían más sencillas si fueran otros elementos más comunes, como por ejemplo uno de tipo <p>, que sirve para definir texto de párrafo en html, en lugar de usar svg:text.

3.4.2. Efecto revista: revista.js

Este bloque transforma un elemento html en un elemento animado con efectos de pasar página. También encierra en sí mismo una estructura que agrupa las páginas que se están pasando. En concreto, agrupa alrededor de 6 páginas a la vez y no más: las 2 que se están leyendo, las 2 previas y las 2 posteriores. Esto es así para evitar cargar todo el documento a la vez y que el rendimiento no se vea afectado.

Esta librería permite la llamada a una serie de funciones, con el uso de jQuery para indicar el elemento que invoca dichas funciones, de las cuales solo se nombran algunas (las que se usan).

La metodología de adición de páginas al documento responsivo o revista (en este caso), es el siguiente: el elemento hace una llamada al método *turn* e indica entre comillas "addPage", para indicar que se va a añadir una página, y el elemento concreto que va a categorizar dicha página. Cabe decir que el elemento anidado es un elemento html de tipo div que está vacío. Es después de que esta página, a la que (por referencia al arte) se le llama *canvas*, sea cargada cuando se decide el contenido de la misma. Para ello se hace referencia a una función del script revista.js llamada cargarPagina, que indica qué página del pdf y de qué pdf se va a cargar en este recién creado y vacío canvas.

Aquí entra el uso mezclado de los dos grupos grandes de scripts: interviene nuevamente el Parser pdf para poder hacer lectura de la página pdf solicitada y su transformación en una página usable en html. Como se ha comentado antes, este elemento será una larga cadena anidada de grupos de svg. Para la correcta adición, se obtiene el canvas html que se ha generado antes para que estuviese listo, y a él se van anidando los svg que genera la librería de Parser pdf.

Cabe destacar el uso de dos eventos concretos que Turn.js puede gestionar: "turned" y "turning".

"Turned" simboliza el estado *girado*, es decir, el momento en el que se ha girado a una página concreta. "Turning", por el contrario, simboliza el estado *girando*, que es el momento en el que se va a girar a una página. El estado "Turned" es activado cuando la nueva página a la que se ha girado está cargada. El estado "Turning" es activado cuando el ratón se posiciona sobre alguna de las esquinas de una página que pueda girarse, momento en el que se despierta la animación de javascript que simula como dicha esquina se dobla hacia adentro, como si se tratase de una de papel de verdad. Este último estado pretende controlar los escenarios que han de darse cuando se intuye que el lector está a punto de moverse de página. Para ello, tras invocar al elemento que hace referencia al elemento que representa a la revista en el documento html, se llama a bind (de igual manera que antes se llamaba a turn), para después indicar el evento "turned" o "turning", según corresponda.

3.4.3. Procesos varios de gestión de lectura correcta

Para cambiar de artículo: se detecta si la página solicitada se sale del artículo que se estaba leyendo en ese momento. De ser así y de haber algún pdf que corresponda con la solicitud, se calcula cuál es el pdf que corresponde y se procede a la carga de la información.

Para reconstruir un canvas: si el lector ha avanzado deprisa por la revista y alguna página que se estima cargada, no ha sido realmente provista, se analiza justo antes de que el lector regrese a dicho punto para arreglar este fallo, momento en el que activará la recuperación y dicho canvas contendrá la página del artículo que le corresponde.

3.4.4. Carga de contenido multimedia

Si se trata de una imagen, se detectan los posibles elementos svg:image del par concreto de páginas que se estén leyendo. En repetidas pruebas, se dieron escenarios inesperados en los que se detectaban imágenes de páginas pasadas en momentos en los que éstas no estaban siendo mostradas por pantalla. Estos escenarios tenían lugar porque, en versiones previas de la implementación, se calculaban los posibles elementos svg:image de manera general, sin especificar cuáles. Para evitar el despliegue de imágenes incorrectas, se le da un tipo a los elementos grandes (los canvas) a través de identificadores que marcan a qué par pertenecen (cada par de páginas mostradas simultáneamente durante la lectura son identificadas con el mismo código numérico en su atributo "par"). Esto es así porque es más práctico que mezclar listas: habría que obtener los elementos en la página de la izquierda del par, y luego los de la derecha. Darían como resultado dos HTMLCollection. Para evitar muchas acciones repetidas y mezclas, se opta por nombrar a los pares y así obtener los elementos de ambas páginas preguntando por el par.

Esto mismo se hace en la carga de audio y video para no desplegar contenido de otras páginas. Antes de comenzar el proceso de montaje multimedia, se obtienen sólo los elementos que estén agrupados por dicho par.

Con respecto al efecto *popup*, en el caso de la imagen, se hace por medio de las clases modales. El concepto consiste en que en el jsp donde se despliega hay un div,

o un elemento general html, que tiene nombre, identificador… y en su interior, un elemento html del recurso a mostrar, en este caso, un elemento de tipo img. La filosofía consiste en, con ayuda del css, detectar cuándo se pincha sobre la imagen, para después, con la ayuda de un controlador de este evento, desplegar el div que se mencionaba al principio, que con la ayuda del css se muestra en el centro de la pantalla, como una imagen flotante. Además, incluye un botón para cerrar la vista y regresar al documento. Para cargar en este elemento flotante la misma imagen que se ha pinchado, se transmite a través del código javascript la dirección de dicha imagen y se asigna como atributo.

Como se pueden tratar de muchas imágenes, dentro de un bucle, se define para cada una de las imágenes (enlistadas gracias al selector de imágenes por par de folios leídos de la revista) un controlador onclick, que permite especificar el escenario cuando una imagen se pincha. Lo definido aquí es que la ventana modal, o emergente, se muestre, con el atributo de fuente de recurso correspondiente.

En el caso de audio y video, la metodología es la misma. Sin embargo, para la carga de imágenes hay otra técnica.

Al tratarse de un pdf, añadir en éste audio o video es más difícil. Los recursos de audio y video están en las carpetas "audios" y "videos" respectivamente, fuera del documento pdf, así que es necesario marcar con una señal en el pdf dónde y cuál de estos recursos hay que cargar. Para esto, se planteó el uso de imágenes de tipo plantilla, donde cargar video o audio según correspondiese, pero se descartó porque la información que caracteriza al recurso tendría que estar anidada a esa imagen plantilla. Esto último es un proceso que no podría realizarse desde la escritura del pdf, de modo que obligaría al autor a indicar qué clase de recurso hay que cargar sobre qué imagen plantilla, y al desarrollador a añadirlos dinámicamente desde el código, situación que se decide rechazar.

Optando por una solución quizá demasiado descubierta pero suficientemente resolutiva, sobre todo para el planteamiento de solución-prototipo de este trabajo, se opta por establecer una regla que permite al autor del pdf saber cómo escribir una señal en su pdf que indique dónde hay que cargar qué recurso. En el texto, allí donde se quiera reproducir el recurso se ha de escribir @Audio o @Video según el tipo de recurso, seguido de "_n=Titulo#", sin comillas, donde n será un número y título una

cadena de texto, incluyendo espacios en blanco. Este número estará haciendo referencia a un recurso concreto, y el título es el del recurso. Por ejemplo, si el usuario quiere poner dos audios en su pdf, tendrá que guardar dos audios que se llamen audio_1 y audio_2, para que las instrucciones del código indiquen correctamente los recursos a obtener. Un ejemplo: *@Video_1=Las leyes sociales#*

En la ejecución de la aplicación, esta sentencia sirve para cargar el recurso concreto, y dicha sentencia desaparece. En su lugar, aparece una frase que consiste en un símbolo que representa un "play" típico de la reproducción multimedia, y después la frase "Reproducir video" o "Reproducir audio" según corresponda, y el título concreto del recurso.

Existen dos carpetas llamadas respectivamente "audios" y "videos" dentro de cada carpeta de número, y en éstas serán buscados los recursos. Los recursos se llaman "audio_1_2" o "video_2_3", por ejemplo, siendo el primer número una alusión al artículo al que pertenece el recurso, y el segundo el número de recurso a cargar. Al pulsar en el símbolo de "play", el audio o el vídeo se reproducirá en forma *popup*, con controles para desplazarse a lo largo del recurso, y de ampliar hasta pantalla completa en el caso de que sea un vídeo. Al igual que con la imagen, posee un botón para cerrar.

Se especifica técnica y detalladamente al completo el funcionamiento de estos procesos en la Guía del Desarrollador.

4. VALIDACIÓN

A la hora de hablar de cómo se ha validado este trabajo se han de abordar dos secciones: cómo se ha validado el correcto funcionamiento del diseño web y cómo se ha validado el funcionamiento de los scripts, los propios y los adquiridos.

4.1. Funcionamiento web

Para el desarrollo html simplemente se han seguido los estándares HTML que obligan a respetar la existencia de cabecera de contenido de hipervínculo, un cuerpo de contenido con <body>, así como el resto de etiquetas pertinentes. Se han utilizado estructuras de div's lo más sencillas posibles, siempre orientándose a basar el prototipo de la revista como la principal materia de estudio del trabajo. Por tanto, salvo un footer que aporta pie y algo de estilo, el resto es principalmente un div para la revista, que responde por el nombre otorgado al elemento de control de *turn.js*. Es decir, desde turn.js hay que indicar cuál es el div de la web que va a formar la revista.

Además, hay otro div especial para generar los *popup*. De esto se habla en Propuesta de Solución y en Guía de Desarrollo.

Utilizando W3C Validator se ha comprobado que los jsp y los css son válidos.

4.2. Funcionamiento de los scripts

En primer lugar había que probar que las librerías importadas funcionaban correctamente. Se probó la librería de efecto revista con un texto lorep ipsum. Para que el funcionamiento fuese correcto únicamente había que seguir las instrucciones: había que importar cualquier librería que tuviera el código de turn.js (para este trabajo se opta por importar turnmin.js, una versión comprimida), y luego crear un nuevo script que realizase la orden de tipo *main*, el script turnpage.js. Este script inicia el funcionamiento de la revista y añade características básicas a gusto del usuario.

Tras esto, se validó el uso de pdf.js, el Parser pdf, con un pdf de prueba. Se realizaron operaciones de transformación que convirtieron ese pdf en una página html, demostrando que la librería funcionaba. Esas operaciones son mencionadas en el Anexo I, Guía del Desarrollador.

A la hora de hablar de cómo se ha probado este trabajo, en concreto los scripts, hay que hablar de los mensajes de control y de error del navegador.

Con mensajes de salida por pantalla de tipo javascript, se mostraron durante el proceso estructuras de datos, mensajes de punto de acceso, de control, de error y de ruptura, para observar trazas en ejecución. Si se estaba probando que las herramientas de reconstrucción de contenido que no ha cargado correctamente en la revista funcionaban bien, se establecían mensajes que indicaban el número de página leído en todo momento, si tenían contenido o no, si se había solicitado la carga de contenido, si el contenido existía y si este era vacía o no… Otros mensajes que se lanzaban a través del navegador para el control consistían en ligeros cálculos, para observar por qué el contenido se cargaba en páginas incorrectas, o por qué algunas páginas (div's concretamente) se superponían a otras hasta que el usuario pasase el ratón por determinados espacios de la página web… Para detectar la causa de escenarios no previstos se mostraban las estructuras de datos y se analizaban en muchas ocasiones el interior de las mismas con el navegador de estructuras JSON o del estilo que poseen los navegadores en sus perspectivas de Herramienta de Desarrollador Web, o parecidos. También se usan mensajes de control para indicar si se estaban pasando páginas hacia adelante o hacia atrás, si se ha cambiado de pdf, si el pdf tiene algún fallo… Casi todo el proceso interno se ha mostrado a través de mensajes de control o de error para luego poder verlo en el apartado Consola de la Herramienta para el Desarrollador del navegador.

Para validar el código javascript, se atendieron a mensajes que informaban de operaciones que violaban accesos, eran incorrectas o no existían. Estos mensajes se visualizaron en forma de mensajes de error que se lanzaban en la propia consola del navegador, y que normalmente aportaron suficiente información para entender la forma del problema, de la que se podía deducir la causa o investigar sobre ella.

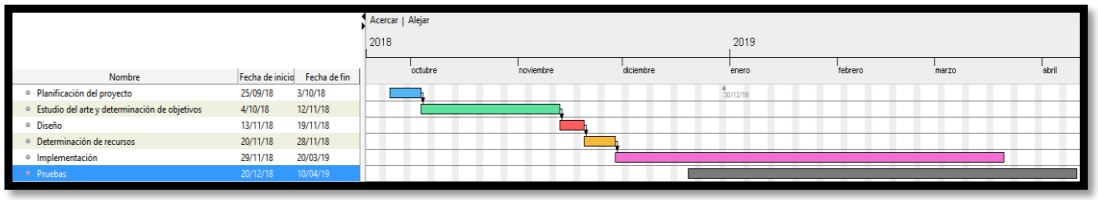
Para estudiar la eficiencia, variables sin usar, repetidas, errores de estilo o estándar, código sin usar, parámetros sin definir… se utilizó la herramienta JSHint, una herramienta que ayuda a detectar fallos, problemas importantes y todo lo

mencionado previamente, aunque esta versión no comprende ciertas cosas correctamente y las tilda de fallos, como el símbolo de $, propio de jQuery.

5. GESTIÓN DEL PROYECTO

La planificación del proyecto ha sido sencilla y ordenada, pero llevarlo a la realidad resultó ser más impredecible.

Figura 5.1. Diagrama de tiempo de la gestión del proyecto



En primer lugar, se estudió claramente la idea a abordar a través del análisis particular de la revista. La idea fue comunicada a través de la tutora del trabajo, durante una reunión.

Planificación del proyecto: las ideas artísticas que afloran de la falta de referencias es motivo de inspiración para muchos creadores de diferentes mundos, y de esta forma de actuar surgió la idea de que la revista debía pasar a ser sencilla, que fuera directa al asunto y que no hiciese olvidar al usuario que lo importante es lo que se cuenta en los artículos. Hay que tratar en su justa medida que las posibilidades que la informática nos ofrece no creen experiencias innecesarias en el usuario, sobre todo para que no ahoguemos el rumbo de la información entre los estímulos modernos.

Estudio del arte y determinación de objetivos: tras esto, el estudio del arte aportó una motivación de apostar por un estilo minimalista. Las revistas digitales como concepto se han dispersado. Las páginas web, ya no solo revistas digitales, que intentan imitar el funcionamiento de una revista física o bien privatizan su servicio o lo usan como un elemento más dentro de una nube de conceptos. En esta brecha se encuentra la posibilidad de que RevIbe se pudiese encontrar a medio camino entre la idea de revista y sencillez. Fue un momento de indagación.

Diseño, Determinación de recursos y posterior Implementación: se planeó primero diseñar un prototipo web sencillo con información de prueba, solo para comprobar que la estructura funcionaba. De ahí sólo se debía pasar al parsing.

La parte del parsing pdf, o el uso de las librerías pdf.js, ha ocupado mucha parte del tiempo de desarrollo. Esto enseña que, en todo plan de desarrollo, es preferible tener en cuenta que puede existir un tiempo indefinido que aportar al aprendizaje de herramientas nuevas, con lo imprevisible que esto puede resultar, en gestión de errores y en proceso de entendimiento. El paradigma central de pdf.js se basa en el uso de Promises, recursos asíncronos en javascript. La inexperiencia en ese terreno del mismo gastó tiempo de proyecto en su desarrollo, y supuso un bloqueo. Tan sólo en la transformación correcta de pdf a html se tardó un mes de manera muy aproximada, entre el planteamiento, la búsqueda de herramientas y su implementación. La gestión de esta herramienta según el proyecto crecía supuso nuevos problemas, que ocuparon un tiempo extra que hicieron que el resto de tareas ocupasen más tiempo.

A la hora de probar esta herramienta, se consiguió un resultado efectivo, pero todas las soluciones aportadas presentaban un mal diseño, que tuvo que reestructurarse. Este mal diseño reside en que ciertas funciones del código estaban construidas siguiendo una secuencialidad que no compaginaba con el asincronismo de las Promises que utiliza esta librería. Dichas operaciones secuenciales o eventos consistían por ejemplo en la espera de los turnos, en el orden de las instrucciones, en la pregunta de datos o su consulta… todo confiando en que el resultado esperado no fuera el obtenido. Al ser estas Promises un recurso más limitado y menos intuitivo, y al no poseer experiencia sobre el tema, esa parte de la gestión del proyecto ocupó muchos días no contabilizados, que se pueden resumir en que son el factor que ha hecho que la mayoría de tareas de desarrollo del código más largas.

Pruebas, paralelas a la Implementación: superada esta parte aparecieron nuevos problemas, como que la reconstrucción de contenido de los artículos no ha sido bien desplegada sobre la web. Si, a raíz del uso de Promises, algún comportamiento no esperado generaba una situación en la que alguna página del pdf no había sido cargada correctamente, había que enmendar dicho escenario. La técnica empleada reside en la detección del estado del contenido al que el usuario se dirige en todo momento, con un análisis que detecta si el contenido existe, se intentó cargar, si existe espacio para él, e incluso si corresponde contenido alguno. Este control asegura que el usuario no detecte que se dirige a una sección del documento que se

está reconstruyendo, pues dicho control se asegura de que el contenido esté siempre listo. Dentro de la carga habitual del contenido que corresponde en la página que le pertenece, existe este control, que se asegura de si ya se ha cargado antes, o si no está cargado y debería estarlo. Durante el resto del desarrollo, se encontraron comportamientos no bien controlados que tuvieron que corregirse.

En la parte final, se diseñó la solución que permite la inclusión de imagen, audio y videos con mayor expresividad en el documento (en concreto en el audio y en el video, con su simple aparición y reproducción en el documento). Para ello no hubo planificación exacta, tan sólo usar el tiempo restante para implementar una solución práctica que diese al prototipo un brillo final. Entre una semana y dos se usaron para generar el popup correcto de las imágenes, el diseño de la solución de reproducción de video y audio y su implementación. Se encontraron muchos problemas a la hora de generar una representación intuitiva para los segmentos de reproducción de recursos, como se anexa más adelante.

6. CONCLUSIONES

A pesar de ser un prototipo, la respuesta final podría haber sido más completa, y haber mostrado una web con algo más de acabado final, con una sección principal en la que se podía escoger el número de la revista a cargar para leer. Sin embargo, la clave del problema ha sido resuelta, es decir, la carga de un número formado por pdfs en una revista responsiva y expresiva.

6.1. Comparativa de objetivos

En la siguiente tabla se representan en la columna izquierda los objetivos planeados al principio del trabajo, y en la columna de la derecha los objetivos logrados y su comparación con los planificados.

Tabla 6.1. Comparativa de objetivos planeados y logrados

Objetivos planeados	Objetivos logrados
Ser una revista clara, sencilla de visualizar y animada.	Conseguido
Ofrecer mayor expresividad a los artículos, permitiendo añadir audios y videos reproducibles y que las imágenes sean ampliables.	Conseguido
La revista sea vista desde distintos navegadores y plataformas sin problemas.	Conseguido salvo en móviles, que en "Trabajos futuros" se comenta.
Que los artículos de un número conformen una revista, para que puedan leerse de principio a fin.	Conseguido
Que se pueda leer un artículo concreto sin necesidad de leer el número entero.	Conseguido

6.2. Aprendizaje

Con la realización de este trabajo he hecho un aprendizaje de:

- Javascript, especialmente el uso de Promises. Los conocimientos de javascript al comienzo de este trabajo eran escasos. Al terminar este trabajo se conocen con mayor profundidad los principios estilísticos del lenguaje, así como buenas prácticas, metodologías y claves principales.
- Uso de librerías de turn.js y pdf.js.
- Ampliación de conocimientos sobre html, jsp y css.

A nivel personal, he aprendido a tener más rigor y contemplar con más seriedad la realización de un proyecto. Quizás la formalidad del trabajo o la personalidad del mismo por ser el proyecto cumbre del Grado, es lo que me ha generado esta sensación de compromiso. Me ha permitido entender que en la gestión de un proyecto software a nivel profesional tengo que estar preparado a niveles de madurez e inteligencia emocional, para poder sobrellevar planificaciones, crestas de trabajo, combinaciones entre trabajo y aprendizaje constante, capacidad de autocrítica y de recepción de crítica externa, enfrentamiento a problemas con tecnologías, actualización de proyectos… También he entendido que he de estar receptivo a aprender tecnologías continuamente.

Ese cariz que toma la formación de un proyecto como este es lo que me ha hecho entender que he de estar preparado para todo en el ámbito profesional, pero me ha generado además mucha más pasión por esta disciplina, por el buen hacer y por el logro de las metas que se puedan establecer.

7. TRABAJOS FUTUROS

En el futuro, este proyecto aceptaría más soluciones software o herramientas, como la búsqueda de texto, y quizá el uso de más material multimedia, o la adaptación del contenido del pdf a personas con problemas de accesibilidad. También sería importante que los artículos pudieran descargarse, desde una página concreta de la web, y también mientras se leen. Una opción que permitiese la descarga de todo un número, con todos sus artículos, sería ideal.

En móviles no funciona correctamente, porque la lectura de artículos tal y como la plantea esta revista no es práctica, porque la pantalla de los móviles es mucho más pequeña. Un buen trabajo futuro podría ser la realización de una versión de la página web adaptada para móviles, en las que el texto se pudiera leer ocupando toda la pantalla, sin efecto de pasar página, parecido a como se puede leer una noticia o un artículo de internet desde el móvil.

REFERENCIAS BIBLIOGRÁFICAS

[1]"JavaScript.com". [Online]. Disponible en: https://www.javascript.com/

[2]"JavaScript Tutorial". [Online]. Disponible en: https://www.w3schools.com/Js/

[3]"How to enable JavaScript in Windows". [Online]. Disponible en: https://support.microsoft.com/en-us/help/3135465/how-to-enable-javascript-in-windows

[4]"JavaScript | MDN". [Online]. Disponible en: https://developer.mozilla.org/en-US/docs/Web/javascript

[5]"The Modern JavaScript Tutorial". [Online]. Disponible en: https://javascript.info/

[6]"The Dollar Sign ($) and Underscore (_) in JavaScript". [Online]. Disponible en: https://www.thoughtco.com/and-in-javascript-2037515

[7]"Arrow functions - JavaScript | MDN - MDN Web Docs". [Online]. Disponible en: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions

[8]"JavaScript comparison operators: Identity vs. Equality ...". [Online]. Disponible en: https://stackoverflow.com/questions/5447024/javascript-comparison-operators-identity-vs-equality

[9]"jQuery". [Online]. Disponible en: https://jquery.com/

[10]"jQuery Tutorial - w3schools.com". [Online]. Disponible en: https://www.w3schools.com/jquery/

[11]"jQuery API Documentation". [Online]. Disponible en: https://api.jquery.com/

[12]"jQuery UI". [Online]. Disponible en: https://jqueryui.com/

[13]"jQuery Introduction - W3Schools". [Online]. Disponible en: https://www.w3schools.com/jquery/jquery_intro.asp

[14]"$ vs $() | jQuery Learning Center". [Online]. Disponible en: https://learn.jquery.com/using-jquery-core/dollar-object-vs-function/

[15]"jQuery - Bug Tracker". [Online]. Disponible en: https://bugs.jquery.com/

[16]"javascript - document.getElementById vs jQuery $() - Stack ...". [Online]. Disponible en: https://stackoverflow.com/questions/4069982/document-getelementbyid-vs-jquery

[17]"javascript - How to get a DOM Element from a JQuery ...". [Online]. Disponible en: https://stackoverflow.com/questions/1677880/how-to-get-a-dom-element-from-a-jquery-selector

[18]"How do I pull a native DOM element from a jQuery object ...". [Online]. Disponible en: https://learn.jquery.com/using-jquery-core/faq/how-do-i-pull-a-native-dom-element-from-a-jquery-object/

[19]"jQuery Selectors - w3schools.com". [Online]. Disponible en: https://www.w3schools.com/jquery/jquery_selectors.asp

[20]"How To Detect Which Element Was Clicked, Using jQuery ...". [Online]. Disponible en: https://www.metaltoad.com/blog/how-detect-which-element-was-clicked-using-jquery

[21]"jQuery – How to get element with CSS class name and id ...". [Online]. Disponible en: https://www.mkyong.com/jquery/jquery-how-to-get-element-with-css-class-name-and-id/

[22]"You Might Not Need jQuery". [Online]. Disponible en: http://youmightnotneedjquery.com/

[23]"jQuery – How to get the tag name – Mkyong.com". [Online]. Disponible en: https://www.mkyong.com/jquery/jquery-how-to-get-the-tag-name/

[24]"Get parent element by jQuery/JavaScript :: Advance Sharp". [Online]. Disponible en: http://www.advancesharp.com/blog/1058/get-parent-element-by-jquery-javascript

[25]"jQuery .on(): Dynamically Added Elements | jQueryHouse". [Online]. Disponible en: https://jqueryhouse.com/jquery-on-method-the-issue-of-dynamically-added-elements/

[26]"JavaScript Arrays - w3schools.com". [Online]. Disponible en: https://www.w3schools.com/js/js_arrays.asp

[27]"jQuery foreach: Using jQuery $.each - RevillWeb". [Online]. Disponible en: https://blog.revillweb.com/jquery-foreach-using-jquery-each-99904fea31bf

[28]"Array.prototype.push() - JavaScript | MDN". [Online]. Disponible en: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/push

[29]"Remove Item from Array using jQuery | jQuery By Example". [Online]. Disponible en: http://www.jquerybyexample.net/2012/02/remove-item-from-array-using-jquery.jsp.

[30]"Promise - JavaScript | MDN". [Online]. Disponible en: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

[31]"SVG Tutorial - W3Schools". [Online]. Disponible en: https://www.w3schools.com/graphics/svg_intro.asp

[32]"SVG File (What It Is and How to Open & Convert One)". [Online]. Disponible en: https://www.lifewire.com/svg-file-4120603

[33]"SVG: Scalable Vector Graphics | MDN". [Online]. Disponible en: https://developer.mozilla.org/en-US/docs/Web/SVG

[34]"W3C SVG Working Group - w3.org". [Online]. Disponible en: https://www.w3.org/Graphics/SVG/

[35]"SVG in HTML - W3Schools". [Online]. Disponible en: https://www.w3schools.com/graphics/svg_inhtml.asp

[36]"JavaScript Promises: an Introduction | Web Fundamentals...". [Online]. Disponible en: https://developers.google.com/web/fundamentals/primers/promises

[37]"Using Promises - JavaScript | MDN". [Online]. Disponible en: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Using_promises

[38]"An Overview of JavaScript Promises - SitePoint". [Online]. Disponible en: https://www.sitepoint.com/overview-javascript-promises/

[39]"Promises chaining - JavaScript". [Online]. Disponible en: https://javascript.info/promise-chaining

[40]"Master the JavaScript Interview: What is a Promise? - Medium". [Online]. Disponible en: https://medium.com/javascript-scene/master-the-javascript-interview-what-is-a-promise-27fc71e77261

[41]"Understanding promises in JavaScript - By - hackernoon.com". [Online]. Disponible en: https://hackernoon.com/understanding-promises-in-javascript-13d99df067c1

[42]"JavaScript: Promises explained with simple real life analogies". [Online]. Disponible en: https://codeburst.io/javascript-promises-explained-with-simple-real-life-analogies-dd6908092138

[43]"Promises - Part 8 of Functional Programming in JavaScript ...". [Online]. Disponible en: https://www.youtube.com/watch?v=2d7s3spWAzo

[44]"Promesas de JavaScript: introducción | Web Fundamentals ...". [Online]. Disponible en: https://developers.google.com/web/fundamentals/primers/promises?hl=es

[45]"javascript - How do I wait for a promise to finish before...". [Online]. Disponible en: https://stackoverflow.com/questions/27759593/how-do-i-wait-for-a-promise-to-finish-before-returning-the-variable-of-a-functio

[46]"6 Reasons Why JavaScript Async/Await Blows Promises Away ...". [Online]. Disponible en: https://hackernoon.com/6-reasons-why-javascripts-async-await-blows-promises-away-tutorial-c7ec10518dd9

[47]"Error handling with promises - javascript.info". [Online]. Disponible en: https://javascript.info/promise-error-handling

[48]"javascript - Handling errors in Promise.all - Stack Overflow". [Online]. Disponible en: https://stackoverflow.com/questions/30362733/handling-errors-in-promise-all

[49]"Understanding JavaScript Promises - flaviocopes.com". [Online]. Disponible en: https://flaviocopes.com/javascript-promises/

[50]"JavaScript For Loop - w3schools.com". [Online]. Disponible en: https://www.w3schools.com/js/js_loop_for.asp

[51]"Loops and iteration - JavaScript | MDN". [Online]. Disponible en: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Loops_and_iteration

[52]"HTML Tutorial - W3Schools". [Online]. Disponible en: https://www.w3schools.com/html

[53]"HTML5 - Wikipedia". [Online]. Disponible en: https://en.wikipedia.org/wiki/HTML5

[54]"HTML - Basic Tags - Tutorialspoint". [Online]. Disponible en: https://www.tutorialspoint.com/html/html_basic_tags.htm

[55]".jsp() | jQuery API Documentation". [Online]. Disponible en: https://api.jquery.com/html/

[56]"CSS Tutorial - w3schools.com". [Online]. Disponible en: https://www.w3schools.com/Css/

[57]"CSS: Cascading Style Sheets | MDN". [Online]. Disponible en: https://developer.mozilla.org/en-US/docs/Web/CSS

[58]"CSS Syntax - w3schools.com". [Online]. Disponible en: https://www.w3schools.com/Css/css_syntax.asp

[59]"The W3C CSS Validation Service". [Online]. Disponible en: http://jigsaw.w3.org/css-validator

[60]"CSS '>' selector; what is it? - Stack Overflow". [Online]. Disponible en: https://stackoverflow.com/questions/4459821/css-selector-what-is-it

[61]".css() | jQuery API Documentation". [Online]. Disponible en: https://api.jquery.com/css/

[62]"CSS vertical-align Property - w3schools.com". [Online]. Disponible en: https://www.w3schools.com/cssref/pr_pos_vertical-align.asp

[63]"6 Methods For Vertical Centering With CSS - Vanseo Design". [Online]. Disponible en: https://vanseodesign.com/css/vertical-centering/

[64]"How to Vertically Center Text with CSS". [Online]. Disponible en: https://www.w3docs.com/snippets/css/how-to-vertically-center-text-with-css.jsp

[65]"How to Align Text Vertically Center in Div Using CSS". [Online]. Disponible en: https://tutorialdeep.com/knowhow/align-text-vertically-center-css/

[66]"How To Vertically Center an Image in a DIV (All Browsers ...". [Online]. Disponible en: https://www.designpieces.com/2012/12/vertical-centering-image-in-a-div/

[67]"CSS display Property - w3schools.com". [Online]. Disponible en: https://www.w3schools.com/CSSref/pr_class_display.asp

[68]"Working with CSS Display Property - Tutorial Republic". [Online]. Disponible en: https://www.tutorialrepublic.com/css-tutorial/css-display.php

[69]"Understanding 'display: none' and 'visibility: hidden' in CSS". [Online]. Disponible en: https://www.lifewire.com/display-none-vs-visibility-hidden-3466884

[70]"The Anti-hero of CSS Layout - "display:table" | Colin Toh". [Online]. Disponible en: https://colintoh.com/blog/display-table-anti-hero

[71]"Deference Between CSS Display and Visibility - Tutorial ...". [Online]. Disponible en: https://www.tutorialrepublic.com/css-tutorial/css-visibility.php

[72]"CSS display: inline vs inline-block - Stack Overflow". [Online]. Disponible en: https://stackoverflow.com/questions/9189810/css-display-inline-vs-inline-block

[73]"CSS Grid Layout - CSS: Cascading Style Sheets | MDN". [Online]. Disponible en: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout

[74]"css - Floating elements within a div, floats outside of ...". [Online]. Disponible en: https://stackoverflow.com/questions/2062258/floating-elements-within-a-div-floats-outside-of-div-why

[75]"CSS Outline Properties - W3Schools". [Online]. Disponible en: https://www.w3schools.com/css/css_outline.asp

[76]"html - CSS How to Make Image Display outside of div ...". [Online]. Disponible en: https://stackoverflow.com/questions/18467436/css-how-to-make-image-display-outside-of-div

[77]"CSS - Nested Div Extending Outside The Parent Div ...". [Online]. Disponible en: http://www.allwebdevhelp.com/css-styles/css-help-tutorials.php?i=35017

[78]"Why is my content going outside the div? - Quora". [Online]. Disponible en: https://www.quora.com/Why-is-my-content-going-outside-the-div

[79]"CSS list-style-position property - W3Schools". [Online]. Disponible en: https://www.w3schools.com/cssref/pr_list-style-position.asp

[80]"Position a child div relative to parent container in CSS". [Online]. Disponible en: https://tomelliott.com/html-css/css-position-child-div-parent

[81]"shape-outside - CSS: Cascading Style Sheets | MDN". [Online]. Disponible en: https://developer.mozilla.org/en-US/docs/Web/CSS/shape-outside

[82]"CSS Layout - The position Property". [Online]. Disponible en: https://www.w3schools.com/Css/css_positioning.asp

[83]"text-overflow | CSS-Tricks". [Online]. Disponible en: https://css-tricks.com/almanac/properties/t/text-overflow/

[84]"Floating causes child div goes outside parent div - HTML ...". [Online]. Disponible en: https://www.sitepoint.com/community/t/floating-causes-child-div-goes-outside-parent-div/40864

[85]"PDF.js - mozilla.github.io". [Online]. Disponible en: https://mozilla.github.io/pdf.js/

[86]"Getting Started - mozilla.github.io". [Online]. Disponible en: https://mozilla.github.io/pdf.js/getting_started/

[87]"Pdf.js - JavaScripting". [Online]. Disponible en: https://www.javascripting.com/view/pdf-js

[88]"Setup PDF.js in a website · mozilla/pdf.js Wiki · GitHub". [Online]. Disponible en: https://github.com/mozilla/pdf.js/wiki/Setup-PDF.js-in-a-website

[89]"pdf.js: Rendering PDF with HTML5 and JavaScript | Andreas Gal". [Online]. Disponible en: https://andreasgal.com/2011/06/15/pdf-js/

[90]"pdf-viewer - npm". [Online]. Disponible en: https://www.npmjs.com/package/pdf-viewer

[91]"Displaying PDF files with PDF.js library | Tizen Developers". [Online]. Disponible en: https://developer.tizen.org/community/tip-tech/displaying-pdf-files-pdf.js-library

[92]"PDF.js 'Hello, world!' example - JSFiddle". [Online]. Disponible en: https://jsfiddle.net/pdfjs/9engc9mw/

[93]"pdf.js: Interesting Project, Incorrect Rendering | PDFTron". [Online]. Disponible en: https://www.pdftron.com/blog/pdf-js/pdf-js-incorrect-rendering/

[94]"Turn.js: The page flip effect in HTML5". [Online]. Disponible en: http://turnjs.com/

[95]"Method: addPage - Turn.js Documentation". [Online]. Disponible en: http://www.turnjs.com/docs/Method:_addPage

[96]"Turn.js responsive book - JSFiddle". [Online]. Disponible en: http://jsfiddle.net/hashem/uz3wo7L8/

[97]"Looking for Flipbook frameworks. Any alternatives to Turn.js?". [Online]. Disponible en: https://stackoverflow.com/questions/17424280/looking-for-flipbook-frameworks-any-alternatives-to-turn-js

[98]"Turn.js Magazines | Drupal.org". [Online]. Disponible en: https://www.drupal.org/project/turnjs_magazines

[99]"Js Tutorial - Turn.js - Make a flip book with HTML5". [Online]. Disponible en: http://js-tutorial.com/turnjs-make-a-flip-book-with-html5-516

[100]"Responsive magazine using Turn.js - Creative Technology ...". [Online]. Disponible en: https://medium.com/creative-technology-concepts-code/responsive-magazine-using-turn-js-ea793e599e0c

[101]"4 Ways to Turn On Javascript in Mozilla Firefox - wikiHow". [Online]. Disponible en: https://www.wikihow.com/Turn-On-Javascript-in-Mozilla-Firefox

[102]"turnjs | Drupal.org". [Online]. Disponible en: https://www.drupal.org/project/turnjs

[103]"Definiciones de revista digital - Emma Llensa - Consultora ...". [Online]. Disponible en: https://emmallensa.com/definiciones-revista-digital/

[104]"10 pasos para diseñar tu propia revista digital". [Online]. Disponible en: http://www.entreperiodistas.com/10-pasos-disenar-revista-digital/

[105]"Ejemplos de revista digital y E-Book 3d Issue". [Online]. Disponible en: https://www.3dissue.com/es/ejemplos-revista-digital.jsp

[106]"Revista de Investigación Educativa". [Online]. Disponible en: http://revistas.um.es/rie

[107]"Definición de revista - Qué es, Significado y Concepto". [Online]. Disponible en: https://definicion.de/revista/

[108]"Revistas Digitales". [Online]. Disponible en: http://www.revistasdigitales.com/

[109]"Significado de Minimalista (Qué es, Concepto y Definición ...". [Online]. Disponible en: https://www.significados.com/minimalista/

[110]"Qué Es Minimalismo - Significado, Concepto, Definición". [Online]. Disponible en: https://significadoconcepto.com/minimalismo/

[111]"10 tendencias del diseño web actual - DevCode". [Online]. Disponible en: https://www.devcode.la/blog/10-tendencias-del-diseno-web-actual/

[112]"15 tendencias en Diseño Web 2018 | Futurvia". [Online]. Disponible en: https://www.futurvia.es/blog/15-tendencias-diseno-web-2018

[113]"Tendencias en diseño web 2019: lo que viene para quedarse". [Online]. Disponible en: https://www.40defiebre.com/tendencias-diseno-web

[114]"Tendencias en diseño web para este 2019". [Online]. Disponible en: https://www.webempresa.com/blog/tendencias-diseno-web-2019.jsp

ANEXOS

Anexo I: Guía del Desarrollador

1. Introducción

Para abordar la perspectiva del desarrollador, primero se detalla la estructura de directorios y ficheros usados en este proyecto.

Se instala un servidor Apache Tomcat (versión 7.0) con el uso de la aplicación de Eclipse JEE. Esto quiere decir que todos los ficheros implicados en esta aplicación están dentro de un proyecto Java de Eclipse en el entorno Eclipse JEE, preparado para soportar servidores. En esta perspectiva del entorno, una de sus pestañas es "Servers", desde donde se puede añadir un servidor Apache Tomcat. En ese punto, Eclipse solicita al usuario que indique la ruta exacta de la carpeta de instalación de Apache Tomcat.

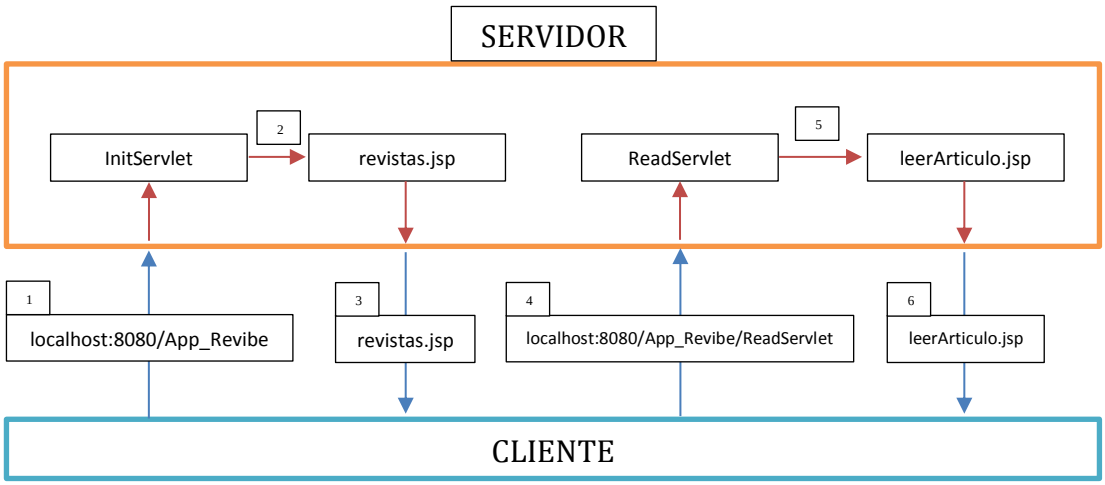


Figura 7.1. Arquitectura Cliente-Servidor

1. Donde pone localhost, habría que poner en su lugar la dirección IPv4 del dispositivo que está ejecutando el servidor, para así acceder desde otro dispositivo.
2. La llamada a "/App_Revibe" invoca a InitServlet, que va a enviar "revistas.jsp" al navegador del cliente.
3. El fichero "revistas.jsp" es desplegado en el navegador web del cliente.

4. Al seleccionar un número, “bookshelf.js” obliga al navegador a redirigirse a ReadServlet.
5. La llamada a “/App_Revibe/ReadServlet” invoca a ReadServlet, que va a enviar “leerArticulo.jsp” al navegador del cliente.
6. El fichero “leerArticulo.jsp” es desplegado en el navegador web del cliente.

Los ficheros de la web, como .jsp y .css, así como los scripts .js y las fuentes de letra están almacenados en directorios dentro del proyecto java de eclipse. En el directorio de recursos java, es decir, el código fuente java del proyecto, se encuentran dos ficheros .java, que corresponden a dos Servlets que gestionan las peticiones al servidor, “InitServlet” y “ReadServlet”. “InitServlet” lee los títulos de los artículos de los números desde sus directorios, y los envía a “revistas.jsp”, para que puedan ser mostrados en la portada, cuando se seleccionen los números. “bookshelf.js” es el script que gestiona en tiempo real la portada de la revista, y cuando detecta que un número ha sido seleccionado, recoge las claves que encuentra en el elemento activado, y que son necesarias para identificar el contenido solicitado por el usuario, y se redirige a “ReadServlet”, desde donde se navega automáticamente a “leerArticulo.jsp”. Es en “revista.js” donde se gestionan esas claves, y son pasadas a las funciones de “pdf.js” para que pueda realizar su transformación de pdf a html. El contenido se va mostrando sobre “leerArticulo.jsp”.

1.1. Estructura de directorios

Los directorios del proyecto que contienen código generado para este trabajo son:

- Java Resources
    - InitServlet.java: inicia la aplicación y carga la portada de la revista.
    - ReadServlet.java: carga la lectura del artículo o del número completo.
- WebContent
    - CSS: directorio con los css.
    - "fonts": tipos de fuente de letra, importados en CSS.
    - "numeros": números de la revista. Contiene un directorio interno por cada número de la revista, y cada uno contiene estos directorios/ficheros:
        - (Directorio) Audios: ficheros de audio del número.
        - (Directorio) Videos: ficheros de video del número.
        - Ficheros pdf (uno por cada artículo del número).
        - "#_titulos.txt", siendo la # la numeración del número de la revista, este fichero .txt contiene una lista ordenada de los títulos de los ficheros .pdf.
        - "portada#.png", siendo la # la numeración del número de la revista, esta es la imagen de portada del número.
    - "scripts": ficheros javascript.
- WEB-INF
    - "lib": librería de jstl, lenguaje de uso práctico en los ficheros .jsp
    - "revistas.jsp"
    - "leerArticulo.jsp"
    - Fichero web-xml que indica que "InitServlet" es el punto de inicio de la aplicación.

El **directorio audios (dentro del directorio de los números)** guarda las pistas de audio que se pueden reproducir en los artículos. Estos audios son de tipo mp3.

El **directorio css** guarda los scripts de estilo de tipo .css.

El **directorio números** guarda en su interior más directorios, uno por cada número de la revista. Dentro de cada uno de estos directorios, están los artículos de dicho número, en formato pdf.

El **directorio scripts** guarda los scripts en formato javascript de este proyecto, tanto los implementados por el autor como los utilizados de otros autores.

El **directorio videos (dentro del directorio de los números)** guarda los archivos de video que pueden reproducirse en los artículos. Estos vídeos son de tipo mp4.

El **fichero "leerArticulo.jsp"** es el encargado de mostrar una revista animada que imita a una impresa.

El **fichero "revistas.jsp"** es una muestra de los números seleccionables. Estos números seleccionables son cargados desde la carpeta "numeros" mencionada más arriba en este mismo texto. En esta página, se puede pinchar sobre estos números. Al pinchar sobre cualquiera de ellos, se despliega una lista de los artículos de dicho número, así como una opción dentro de esa lista que permite ver el artículo entero. Si se pincha sobre cualquier artículo de la lista, se navegará automáticamente hasta "leerArticulo.jsp" y se podrá leer dicho artículo. Si se desea leer el número entero y se pulsa esa opción, se carga una revista con todos los artículos en orden.

1.2. Estructura de directorios en detalle

Cada uno de estos directorios ha de tener una estructura interna concreta, que se detalla a continuación.

1.2.1. El directorio "audios" y el directorio "videos"

Tanto la carpeta "audios" como la carpeta "videos" organizan su contenido de manera que sea claro a qué número y artículo pertenece cada recurso. Los videos se titulan como "video_X_Y.mp4", y los audios como "audio_X_Y.mp3", siendo X la numeración del número de la revista, y la Y el número del recurso. Se ha de

acompañar una imagen con el audio, y deberá estar en la carpeta “audios”, nombrada “imagen_X_Y.jpg”, siendo también X la numeración del número de la revista, y la Y el número del recurso.

1.2.2. El directorio “numeros”

Este directorio aloja los artículos de cada número. Por precaución, el directorio se nombra “*numeros*” en lugar de “números”, sin tilde, para evitar el uso de caracteres especiales que pudieran producir fallos o complicaciones.

Esta carpeta tiene en su interior una carpeta por cada número. Estas carpetas guardan ficheros pdf: los artículos. Además, guardan una fotografía, que es la portada del número.

A las carpetas de número se las nombra como “*numero_nombreDeRevista”*: “numero” es un número natural distinto de 0, como se indica en la sección *“2.1.1. El directorio ‘audios’ y el directorio ‘videos’”* de este mismo anexo; y “nombreDeRevista” es el nombre de la revista.

Los artículos se nombran con “*numero_nombreDeArticulo*”, siendo “numero” un número natural distinto de cero, “nombreDeArticulo” una cadena de texto normal y corriente. Además, debería haber una foto en formato .jpeg que se ha de llamar “0”, que corresponde con la portada.

Con respecto al título de las carpetas que guardan el número de la revista, la convención de que el título de la revista esté ahí escrito no es estrictamente necesaria para la codificación, ya que implica mencionarlo en el código para que el directorio sea accedido correctamente, pero se añade en este proyecto como mero estándar para aportar claridad.

Al añadir el título de la revista en estas carpetas, hay que especificarlo además en la ruta de directorio para leer los recursos:

Figura 8.1. Línea 22 de fichero “revista.js”

Esta línea pertenece al fichero “revista.js” del directorio “scripts”. Pertenece a la primera parte del código, y es la definición y asignación de una variable *var* de javascript, cuyo propósito es ser la ruta del directorio del número a leer, como puede intuirse por su nombre, “direccionNumero”. Esta ruta estará formada por una cadena de texto “numeros/” que hace referencia a la carpeta “numeros”, la variable “numeroSolicitado” que es un número que hace referencia a un número concreto, y la cadena de texto “_revibe/” (subrayado en gris en la figura 8.1.), que hace referencia al título que tiene la revista que se está usando de ejemplo en otro proyecto. Hay varias opciones, en caso de que esto último quisiese modificarse:

- Se puede dejar de escribir “_nombreDeRevista” (como por ejemplo “_revibe”) al final del nombre de la carpeta, lo que implicaría que la sentencia en el código sería más corta:

Figura 8.2. Versión de la línea 22 de fichero “revista.js”

```
var direccionNumero = "numeros/" + numeroSolicitado;
```

- La otra opción es cambiar manualmente el nombre de la revista en las carpetas y en la sentencia de código. Por ejemplo, si la revista fuese la de National Geographic, como la expuesta en la sección del Estudio del Arte, el nombre podría ser algo como “_natGeographic”, y habría que ponerlo en las carpetas de los números (“1_natGeographic”, “2_natGeographic”…), y en la sentencia de código, expuesta en las figuras 8.1. y 8.2.

Figura 8.3. Ejemplo de modificación de la línea 22 de fichero “revista.js”

```
var direccionNumero = "numeros/" + numeroSolicitado + "_natGeographic/";
```

1.2.3. El directorio "scripts"

Guarda los ficheros javascript del proyecto. Son editados desde Microsoft Virtual Studio. Son los siguientes:

- jquery-3.3.1.min.js
- pdf.js
- pdf.worker.js
- text_layer_builder.js
- ui_utils.js
- bookshelf.js
- revista.js
- turn.min.js
- turnpage.js
- rotate.js

Se observan cuatro bloques en esta lista: el verde, el rojo, el naranja y el azul.

El **verde** representa una librería de javascript, la conocida jQuery, usada para facilitar la codificación. Es la versión 3.3.1 y versión min, o sea, comprimida.

El **rojo** representa las librerías de PDF.js y algunas auxiliares de esa misma, con el objetivo de hacer el Parser PDF, es decir, transformar el pdf en un html. "pdf.js" es la librería principal. "pdf.worker.js" es cargado por "pdf.js", y es auxiliar. "text_layer_builder.js" permite funciones de selección de texto en el html formado a raíz del pdf. "ui_utils.js" guarda algunas funciones especiales, y es preciso para que "text_layer_builder.js" funcione.

El **azul** es un bloque que representa la animación de la revista. "turn.min.js" es una librería que permite esa animación, que consiste en una hoja que se mueve como si fuera la de un libro o una revista. Es "min", comprimida. "turnpage.js" es un script de muy pocas líneas que pone en marcha la animación de la revista con algunos datos a elegir, opcionales, como la velocidad de la animación, o si la revista se va a visualizar en doble página o en una sola página.

El bloque **naranja** representa los dos scripts desarrollados para este proyecto. "bookshelf.js", que en inglés significa "estante", es el script que gestiona la portada o página principal de la web. Esta página es una simulación o metáfora de un estante en el que se colocan libros, y consiste en que los números de la revista están puestos en orden, se pueden seleccionar con el ratón y se pueden leer. "revista.js" es un script que gestiona la lectura del artículo o los artículos seleccionado/s.

El bloque **violeta** es el fichero "rotate.js", que obliga al usuario a girar el dispositivo en el que se esté ejecutando la aplicación para que lo coloquen apaisado. Hasta que no se coloca en esa posición, no desaparece un mensaje que pide al usuario que lo gire.

1.2.4. El directorio "css"

Los ficheros css están almacenados en este directorio.

- **body**: tiene reglas de estilo relacionadas con la etiqueta <*body*> de html y el gif de animación de "cargando…" de la portada.
- **estante**: tiene reglas de estilo sobre el estante de "revistas.jsp", incluyendo las listas de artículos que éste despliega.
- **flip**: tiene reglas de estilo sobre la animación de las hojas que pueden pasarse como si fueran las de una revista.
- **footer**: tiene unas reglas de estilo que forman el footer de la web, que es simplemente una barra horizontal oscura que esta al pie de la página.
- **popup**: reglas de estilo para los elementos que saltan en popup. Estos elementos son las imágenes, los audios y los vídeos.
- **STYLE_principal**: tiene reglas de estilo sobre el bloque contenedor de la revista y el bloque principal del cuerpo de la web, e "import" del resto de librerías css.

2. Requerimientos para la ejecución de la aplicación

Antes de explicar el código desarrollado, en esta sección se aborda cómo ejecutar esta aplicación, y dónde y cómo importar las librerías.

- Importar el proyecto de java descargable en Eclipse JEE.
- Hay que tener instalado Apache Tomcat versión 7.0 en el equipo.
- Se despliega un servidor Apache Tomcat versión 7.0 en la pestaña "Servers" de Eclipse 7.0 y se selecciona la aplicación "New > Server > Apache Tomcat 7 > Next > App_Revibe.
- Se ejecuta el servidor.
- Desde un navegador web desde el ordenador donde se esté ejecutando el proyecto de eclipse, escribir "localhost:8080/App_Revibe" para la ejecución de la aplicación. Para acceder desde otro dispositivo, éste ha de estar conectado a la red local donde está desplegado el servidor, y escribir la misma dirección url, intercambiando "localhost" por la dirección IPv4 pública del ordenador donde está lanzado el servidor.

Es preciso que existan todos los ficheros mencionados en la sección anterior, que se encuentran dentro del proyecto importado. La estructura propuesta de sus directorios es, nuevamente, la que sigue el código, y de no querer ser respetada, habría que introducir cambios en los scripts "bookshelf.js", "revista.js", y en los ficheros "leerArticulo.jsp" y "revistas. jsp". Estos dos últimos ficheros están en un directorio, y se encuentran junto al resto de directorios.

"leerArticulo.jsp" y "revistas.jsp" importan las librerías javascript que necesitan y las de css.

"leerArticulo.jsp" importa las librerías de: pdf.js, ui_utils.js, text_layer_builder.js, turn.min.js, revista.js y turnpage.js. Nótese que no importa pdf.worker.js porque esto ya lo hace automáticamente pdf.js al ser importado, pero incluirlo no cambia el resultado. Nótese además que no se importa directamente la librería de jQuery, pues en html no hace falta, ya que es una librería usada para codificar en javascript, así que la necesitan el resto de scripts para codificar, pero no el html. Esta página

también importa el fichero STYLE_principal.css. Sólo importa ese fichero, porque internamente importa todos los demás ficheros css.

“revistas.jsp” también importa el fichero STYLE_principal.css, pero sólo importa el fichero bookshelf.js, ya que es el que gestiona esta parte de la web, y no importa ningún fichero más porque no se hace parsing pdf ni se necesita la animación de la revista.

3. El código

En esta sección se va a explicar la arquitectura principal de los ficheros desarrollados para el proyecto y sus funciones. Para mayor detalle del código, éste se encuentra documentado internamente.

3.1. Los ficheros jsp

Como la naturaleza de estos ficheros está explicada en la sección anterior, en esta subsección se comenta lo más relevante para poder ser entendidos.

3.1.1. "revistas.jsp"

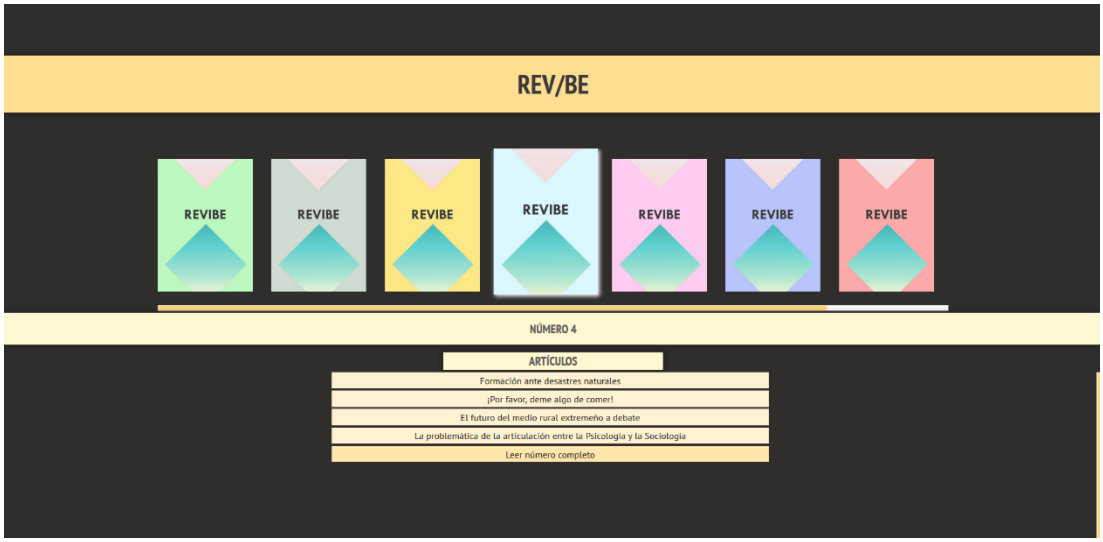


Figura 8.4. Portada, "revistas.jsp"

"revistas.jsp" importa las librerías en la etiqueta head. Como se comenta anteriormente, sólo importa el fichero principal de css y el script de "bookshelf.js". El cuerpo principal de este fichero se llama "bloque-principal" y está dentro de body. Este bloque principal una estructura sencilla en la que tiene por un lado un div llamado "seccion_titulo" que consiste en una barra horizontal que se coloca en lo más alto del navegador, y que sirve para mostrar una cadena de texto que indique el número que el usuario ha seleccionado con el ratón. Después, está el div "bookshelf", que es el que muestra los números. Se basa en una clase "shelf", que es el estante principal, "row-1" que va anidado, y "loc" que también va anidado. Que estos tres elementos vayan anidados comprenden tan solo cuestiones de estilo. Lo importante está dentro de "loc", el último div, que tiene un "foreach" del lenguaje

JSTL, que leerá la colección de números que "InitServlet" le manda. Dentro de este foreach existe un elemento "img", de modo que se generará uno para cada número que tenga la colección. Cada una de estos elementos "img" tienen un id del tipo "numero#", donde la almohadilla es realmente un número, que sirve para ser identificado, y una atributo "numero", con un valor número.

Figura 8.5. Atributos del elemento "img" que representa cada número

```
<img id="numero1" numero="1"… >
```

El objetivo de esto es utilizar el "id" para identificar el elemento, y "numero" para almacenar la numeración del número de la revista, y así evitar tener que obtener la parte de "numero1" que es un 1, y simplificar el código. Esto permite operaciones sencillas como la mostrada en la figura 8.5.

Figura 8.6. Invocación a la función "mostrarListaArticulos" de "bookshelf.js"

```
mostrarListaArticulos($(this).attr("numero"));
```

Esta llamada alude a un elemento "img" de los mostrados en la figura 8.4., y se accede a su atributo "numero", para usarlo como parámetro en la función, la cual mostrará los artículos de ese número. Es más claro hacer esto así que tener que obtener el "1" de "numero1" del id.

Este elemento "img" tiene un atributo "src" en blanco. "src" hace alusión a la dirección del recurso, en este caso, una imagen. Está en blanco para que pueda añadirse una dirección programáticamente. "bookshelf.js" añade a cada número su imagen correspondiente. Conviene recordar que esta imagen es la de la portada, como se puede intuir, pues es el estante de la portada de la web, donde se muestran las revistas. Por tanto, se están mostrando las portadas. Estas imágenes están dentro de las carpetas de los números, y es una imagen .jpeg que se ha de llamar "0".

Después de este "row-1", hay otros dos, pero son paralelos a este primero, no anidados. El segundo aporta solo un rectángulo estilístico sobre el que parece que los números se apoyan, como si fueran un estante. El tercero se llama "seccion-

articulos" y está pensado para ser una lista de artículos. Desde javascript se añaden estos artículos, cuando se detecta que un número ha sido seleccionado con el ratón, y se añaden elementos a esta lista de manera programática.

Fuera de "bookshelf" se encuentra un div llamado "loading". Este div solo se muestra para crear una pantalla de espera de carga, la típica de "cargando…" que aparece cuando aún no se ha cargado el contenido de la web. Cuando aún no se han cargado las imágenes en el estante, aparece esta pantalla, que consiste en un gif de espera, y cuando se han cargado, este div es ocultado programáticamente desde javascript.

3.1.2. "leerArticulo.jsp"

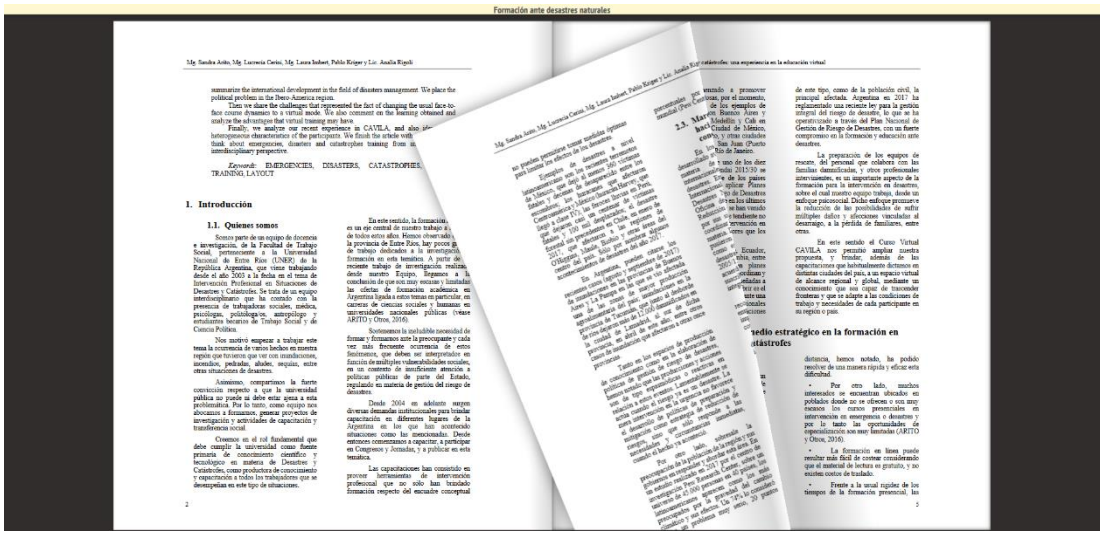


Figura 8.7. Lectura, "leerArticulo.jsp"

"leerArticulo.jsp" importa las librerías en la etiqueta head, y una en body. Como se comenta anteriormente, sólo importa el fichero principal de css. En head importa el script de pdf.js, el ui_utils.js y su text_layer_builder.js, turn.min.js y revista.js. En body se encuentra el script de turnpage.js, que es el constructor de la animación de la revista, y es donde debe estar, según las indicaciones de la librería de turn.js.

Dentro de <*body*>, hay dos grandes div, uno que se llama "popup_window", y otro que se llama bloque-principal, como ya se comentó en la subsección anterior.

“popup_window” es un elemento modal, es decir, un elemento que interrumpe la aplicación, y hasta que no se “cierra”, no se puede continuar. “popup” es un elemento emergente, que sale sobre la pantalla. Este div es por tanto un elemento que se mostrará en forma de popup. Este div pretende ser el popup para las imágenes cuando sean seleccionadas, o cuando un video o un audio tengan que ser reproducido. En cualquier de estos tres casos, este elemento se pondrá delante de todo lo demás.

Este elemento tiene un elemento “span” que permite cerrar la ventana, y un elemento html para imagen, video y audio. Según el recurso multimedia a mostrar, se mostrará uno y se ocultarán los demás. Por defecto, los tres elementos están ocultos. Programáticamente, desde javascript, se mostrará el que sea solicitado, y cuando sea cerrado, volverá a ocultarse.

Para poder controlar la visibilidad de estos elementos correctamente, tienen un id que los diferencia. Los tres pertenecen además a la clase modal-content, del archivo css “popup”.

“bloque-principal” guarda una fila de tipo “row-1” como también se usa en el archivo html explicado en la subsección anterior, que sirve para mostrar en todo momento el título del artículo que se está leyendo, y está sobre la revista.

También tiene un “contenedor_revista”, que lleva dentro “revista”, el div al que hará referencia el constructor de la animación de la revista. “turnpage.js” es el constructor, y necesita saber qué elemento html será el que porte la revista. En este constructor se indica que es “revista”, y durante el script de “revista.js” se hace referencia a este elemento “revista” para gestionar sus funciones.

3.2. Los ficheros javascript

En esta sección se habla sobre los ficheros "bookshelf.js" y "revista.js", desarrollados para el proyecto.

3.2.1. "bookshelf.js"

El objetivo de este script es controlar la lógica de "revistas.jsp". Sus funciones principales son:

- Carga una pantalla de "cargando…" hasta que las portadas han sido cargadas.
- Gestiona la lista de artículos que tiene cada número, y cuándo debe aparecer.
- Manda el artículo o número completo que el usuario quiere leer al script que gestiona la lectura en "leerArticulo.jsp", el script "revista.js".

3.2.1.1. Estructura del fichero

Lo fundamental es que maneja los eventos en "revistas.jsp". Para su correcta ejecución, se hace uso de funciones auxiliares que se detallan más adelante.

3.2.1.2. Evento: el ratón se sitúa en cualquiera de los números

Figura 8.8. Evento del ratón sobre un número de la revista en "revistas.jsp"

```
$('.numero').each(function (index, value) {
    $(this).mouseenter(function () {
       if(!hayNumeroClickado)
              $('#titulo_numeros').html('<div class="textoCentrado"> <p>
       N&UacuteMERO ' + $(this).attr("numero") + "</p> </div>");
});
```

Todos los elementos pertenecen a la clase "numero", de ahí la expresión *$('.numero')*. Dado que la definición del manejador del evento afecta a todos, sea cual sea, se define ".each" en elementos que sean de la clase "numero", que significa que afectará a todos los que cumplan esa condición. Dentro de este bloque "each" hay más eventos, como éste, que se detalla en este punto. "this" está haciendo referencia al elemento con clase "numero" que ha activado ese evento.

"mouseenter" hace referencia a lo que ha de pasar cuando el ratón se sitúa sobre el elemento. Lo que se define aquí es simplemente que, cuando se sitúe el ratón sobre un número, se podrá ver el título de dicho número en el div "titulo_numeros". Esto se consigue haciendo referencia al elemento, dentro de "mouseenter", con la función "html", que es la que permite asignar valor al interior de un elemento html. Lo que se introduce es "Número", escrito de una manera especial para que acepte la "u" con tilde, además de escribir el valor del atributo "numero" de dicho elemento, con "this" y accediendo al atributo número con la función "attr".

3.2.1.3. Clic sobre cualquiera de los números

Figura 8.9. Evento de hacer clic sobre un número de la revista en "revistas.jsp"

```
$('.numero').each(function (index, value) {
…
    $(this).on('click', function () {
        …
        $(this).toggleClass("clicked");
        if (!hayNumeroClickado) {

            numeroClickado = $(this).get(0);
            hayNumeroClickado = $(this).hasClass("clicked");
            mostrarListaArticulos($(this).attr("numero"));

        } else if (hayNumeroClickado &&
                    numeroClickado !== $(this).get(0)) {
            $('.articulo_item_hidden').fadeOut('slow');
            $('.articulo_item').remove();
            numeroClickado.classList.remove("clicked");
            numeroClickado = $(this).get(0);
            mostrarListaArticulos($(this).attr("numero"));

        } else if (hayNumeroClickado &&
                    numeroClickado == $(this).get(0)) {
            $('.articulo_item_hidden').fadeOut('slow');
            $('.articulo_item').remove();
            …
            numeroClickado = null;
            hayNumeroClickado = $(this).hasClass("clicked");
```

Cuando se pincha sobre un número, hay tres casos diferentes que se pueden desencadenar:

- Que no haya ningún número seleccionado, por lo que sería el primero.

  Pertenece al primer if: si "hayNumeroClickado" no es cierto, se "guarda" en "numeroClickado" el ítem que se acaba de pinchar; se actualiza la variable booleana "hayNumeroClickado"; y se muestran

los artículos del número que se acaba de pinchar, usando la función "mostrarListaArticulos".

- Que haya alguno seleccionado, y que el número sobre el que se pincha sea uno diferente.

  Pertenece al primer else-if: si "hayNumeroClickado" es cierto, y además el "numeroClickado" no es el que se está pinchando, se borra los elementos de la lista de artículos del número que esté seleccionado; a ese número previamente seleccionado (accediendo a través de "numeroClickado") se le borra la clase "clicked", que se le añade programáticamente al ser seleccionado y que le da un estilo css concreto, que lo hace más grande y destaca del resto; se actualiza el "numeroClickado" poniendo el que se está seleccionado en este caso; y después se muestra los artículos del número seleccionado.

- Que se pinche sobre un número ya seleccionado.

  Pertenece al segundo else-if: si "hayNumeroClickado" es cierto, y además el "numeroClickado" es el que se está pinchando, en este caso hay que deseleccionarlo; se borra la lista de artículos con "remove"; y se pone a null el "numeroClickado" pues ya no hay ninguno.

3.2.1.4. El ratón deja de estar sobre los artículos de un número

Figura 8.10. Evento del ratón que deja de estar sobre la lista de artículos

```
$('#seccion_articulos').mouseleave(function () {
    if (elementoCorrecto(numeroClickado) &&
            !numeroClickado[0].hasClass("clicked")) {
        $('.articulo_item').remove();
        $('.articulo_item_hidden').remove();
        $('#titulo_numeros').html("");
    }
});
```

"seccion_articulos" es precisamente la sección de la web sobre la que se despliegan los artículos, y "mouseleave" se dispara cuando el ratón deja de estar sobre esta

sección. Lo que ocurre dentro de este controlador es esta comprobación y consecuente acción: si el "numeroClickado" no es null ni indefinido (función *elementoCorrecto*), y el número no está seleccionado, se borra la lista de artículos. Esto quiere decir que, por mucho que el ratón salga de la lista de artículos, si el número sigue seleccionado, los artículos no desaparecerán. De esta manera, la lista de artículos es fija y visible mientras el número este seleccionado.

3.2.1.5. Clic sobre un ítem de la lista de artículos

Figura 8.11. Evento de hacer clic sobre un ítem de la lista de artículos

```
$("#seccion_articulos").on("click", ".articulo_item_hidden", function
(index, value) {
   localStorage.setItem("numeroArticulos", articulosMostrados.length);
   localStorage.setItem("numeroALeer", $(this).attr("numero"));

   if ($(this).attr("lectura") == "completa") {
       localStorage.setItem("leerNumeroCompleto", "true");
   …
   } else if ($(this).attr("lectura") == "parcial") {
       localStorage.setItem("leerNumeroCompleto", "false");
       localStorage.setItem("articuloALeer", $(this).attr("articulo"));
   }

   window.location.href = "/App_Revibe/ReadServlet";
});
```

Aquí se muestra una versión resumida del control de este evento.

Dentro de la seccion_articulos, se especifica el evento de hacer clic sobre "articulo_item", con ".on(click, .articulo_item_hidden…)". Dentro del cuerpo de este código, se marcan dos ítems en localStorage, que sirven para comunicarse con el otro script. "localStorage" es una propiedad del objeto Window, y sirve para el almacenamiento local. El objetivo del uso de esta propiedad es la de marcar varios datos que especifican la petición de lectura, para que en "leerArticulo.jsp" se cargue la información solicitada. Por ejemplo, en localStorage almacenaremos el número del artículo que se ha solicitado leer, para que se cargue en la otra página, o todos si se está solicitando una lectura completa de número.

En localStorage se almacena el número de artículos de la colección, y se almacena en "numeroArticulos", y el número a leer en "numeroALeer". Seguidamente, se presenta esta condicional: si el atributo "lectura" de ese ítem de la lista reza

"completa", se almacena un atributo en localStorage llamado "leerNumeroCompleto" con el valor "true"; pero si ese valor es "parcial", se guarda "false", además del artículo a leer. Esto se explica de esta manera: los ítems de la lista que corresponden a un artículo, son parciales, y el ítem de "Leer número completo", es total. Lo que quiere decir esto es que si se quiere leer el número completo, se pulsa sobre "Leer número completo", pero para leer un artículo concreto se pincha sobre él. Los ítems de la lista tienen un atributo que se llaman "parcial", y el de "Leer número completo" tiene el valor de "completa". Esto es así para que se sepa qué tipo de ítem se ha pinchado. No es lo mismo solicitar todos los artículos que uno solo.

La última línea provoca una navegación forzada a ReadServlet, que se encargará de redireccionar a "leerArticulo.jsp".

3.2.1.6. mostrarListaArticulo en "bookshelf.js"

Esta función tiene como objetivo mostrar los artículos del número que el usuario ha seleccionado desde la portada. Cuando el controlador del evento de un número pulsado se activa, el atributo "numero" del elemento seleccionado es extraído con jQuery, y usado como parámetro para la función mostrarListaArticulo. Es entonces cuando esa función muestra los artículos de ese número. Además, añade un elemento, con la función *append*, un div con el siguiente texto: "Leer número completo". Además, este elemento tiene el atributo *lectura* con el valor "completa". Si se pulsa este elemento, se leen todos los artículos del número.

3.2.2. "revista.js"

El objetivo de este script es controlar la lógica de "leerArticulo.jsp". Sus funciones principales son:

- Crear la revista sobre la que se va a desplegar el contenido que se va a leer.
- Leer si hay que leer un artículo o un número completo.
- Cargar una página del artículo sobre la revista.
- Cargar el contenido multimedia sobre el artículo.
- Cargar un nuevo artículo cuando, en una lectura completa del número, se avanza uno nuevo.
- Cargar una página de un artículo cuando se debería haber cargado, pero no se ha hecho por culpa de algún error.

3.2.2.1. Estructura del fichero

El fichero se fundamenta en dos partes principales: la carga inicial de los artículos, y un cuerpo de código que tiene lugar en el interior de la resolución de *PDFJS.getDocument*, una función muy importante de pdf.js.

PDFJS.getDocument es una función de pdf.js que transforma un pdf en un elemento utilizable desde javascript. Se le pasa por parámetro la dirección del recurso pdf, y se devuelve una variable usable que tiene varias propiedades, entre las que se pueden destacar: una en la que está todo el contenido en forma de texto, una que da el número de páginas y otra con el título.

Hay dos variables importantes en este fichero: "idArticuloLeido" y "artículos". "idArticuloLeido" es tan sólo un número, y representa la posición que tiene el artículo leído en el array "artículos". Cuando se habla de "artículo leído" se refiere al artículo que se está mostrando en la revista. Cuando se comienza la lectura de un número completo, el artículo leído al principio es el primero del número, y según se vaya avanzando, el id del artículo leído irá cambiando a los siguientes. Cuando se avance al siguiente artículo, el id también aumentará su valor, siendo así posible que se acceda a la siguiente posición del array.

"artículos" es ese array, y es importante saber que tiene estas propiedades: url, orden y pdfCargado. "url" es la ruta del directorio donde está el artículo, "orden" es un valor numérico que representa el orden del artículo dentro del número completo, y "pdfCargado" es una variable booleana que indica si ese artículo ya ha sido cargado antes, para evitar cargar contenido que ya fue cargado. Hay dos propiedades que se añaden más tarde, cuando son calculadas: "primerCanvas" y "ultimoCanvas". Para el resto del documento, "página" hará referencia a una página del pdf del artículo, y "canvas" al espacio en blanco que simula un folio de papel de la revista, sobre el que se despliega la información. En base al orden de cada artículo, y al número de páginas que ocupa cada uno, es trivial calcular el canvas en el que empieza y termina cada artículo. Sabiendo esto, es más sencillo controlar los errores y evitar que una página sea cargada incorrectamente.

3.2.2.2. Operaciones de turn.js

La librería turn.js permite animar la revista, pero también gestiona su creación. Al ser un elemento dinámico, la revista se crea en el momento de lectura, y las páginas a leer, también. Esto implica que exista una serie de operaciones, entre la que se distinguen: añadir una página, y los eventos de "turning" y "turned".

La operación de adición de página se llama "addPage". Con el uso de jQuery, se puede hacer uso de esta función haciendo referencia al elemento html que representa la revista, el elemento identificado como "#revista". El símbolo # representa que "revista" como término es un identificador html. Con la función "turn", se puede indicar la operación "addPage", y el elemento que se quiere añadir como página.

Figura 8.12. Ejemplo de adición de dos páginas

```
$("#revista").turn("addPage", $("<img />"));
$("#revista").turn("addPage", $("<div />", { "par": 1 }));
```

En la figura 8.11 se observa dos tipos de adición: la primera indica que se va a añadir una imagen a toda la página (esto se hace al principio para añadir la portada, que es una imagen que tapa toda la página); y el segundo ejemplo es un div, lo que permitirá que después pueda añadirse sobre ese elemento todo el contenido de la página pdf, y un atributo llamado "par", con valor 1. Se permite la adición de atributos al elemento principal de la página, como en este caso, de un atributo al que se bautiza como

"par". En partes más avanzadas del código, en la lectura de posibles elementos multimedia, se hace uso de ese atributo "par" para hacer una búsqueda más sencilla. El atributo "par" se añade a todas las páginas de la revista, y cada par de páginas mostradas en la lectura tienen el mismo valor en "par". De esta manera, las dos primeras páginas del primer artículo tendrán un atributo "par" con valor 1, y el siguiente par tendrá el valor 2. El objetivo de esto es evitar hacer una búsqueda de elementos multimedia en cada página, y mezclar los resultados. Los resultados de las búsquedas en estos elementos html es una HTMLCollection, con complicaciones en su manejo. De esta manera, al buscar por valor del atributo "par", buscará en ambas páginas y devolverá una sola colección.

Si en lugar de "turn", se usa la función "bind", y se pone después "turned" o "turning", se podrá manejar el evento de "página girada" o el de "página antes de ser girada", respectivamente.

Figura 8.13. Evento "turned"

```
$("#revista").bind("turned", function (event, pagRevistaVisible,
pageObject) {…}
```

Figura 8.14. Evento "turning"

```
$("#revista").bind("turning", function (event, pagRevistaVisible,
pageObject) {…}
```

En ambos manejadores de evento, se especifica una función de tres parámetros, que es la que se dispara cuando el evento se produce. Cualquiera de los tres parámetros pueden tener un nombre puesto por el desarrollador, pero sus valores hacen referencia a tres propiedades concretas. "event" es el primero de estos parámetros, y hace referencia al objeto Event que se acaba de producir. El segundo parámetro, que en este código se ha nombrado como "pagRevistaVisible" hace referencia a la página que dispara el evento. En este caso se ha llamado así haciendo referencia a "página visible de la revista". El tercer parámetro es llamado "pageObject", y es un array que guarda las dos páginas que se están visualizando cuando el evento se produce.

El evento "turned" es aquello que sucede cuando una página se ha girado. En este evento, los parámetros "event" y "pagRevistaVisible" son definidos pero no son

usados. Dentro de este manejador, se comprueba si se ha llegado al final del número antes. Si no ha se ha llegado aún al final, se comprueba si se acaba de llegar en el momento en el que se ha producido el evento, pues el objetivo de este manejador es el de colocar una "tapa" final, como si fuera la contraportada de una revista, y evitar la creación de más páginas a la revista. La forma de hacerlo es comprobando, en primer lugar, si se está leyendo el último artículo. Si esto es así, se comprueba si se está en la última página, caso en el que se crea una página más y se advierte que se ha llegado al final del número, para evitar que este manejador de evento vuelva a repetir este código.

Figura 8.15. Creación de tapa final en el evento "turned"

```
$("#revista").turn("addPage", $("<div />", { "id": "tapa-final" }));
```

Como se puede comprobar en la figura 8.14., se crea una página nueva con identificador "tapa-final", que es una regla css que ya describe estilísticamente una tapa gris, que simula el fin de la revista. Al indicar este "id" en la creación de la página, se añade esta propiedad de estilo. De lo contrario, tendría el aspecto de un folio blanco normal, como las del resto de la revista.

3.2.2.3. "cargarPagina" y "cargarPaginaSiguiente"

Puede resultar confusa la proximidad de los nombres de estas funciones. Para despejar las dudas: "cargarPagina" es una función que carga una página de un pdf en una canvas en blanco de la revista; y "cargarPaginaSiguiente" es una función que encapsula "cargarPagina", que gestiona esta operación para que sea correcta, hace algunas comprobaciones y hace más sencilla la operación.

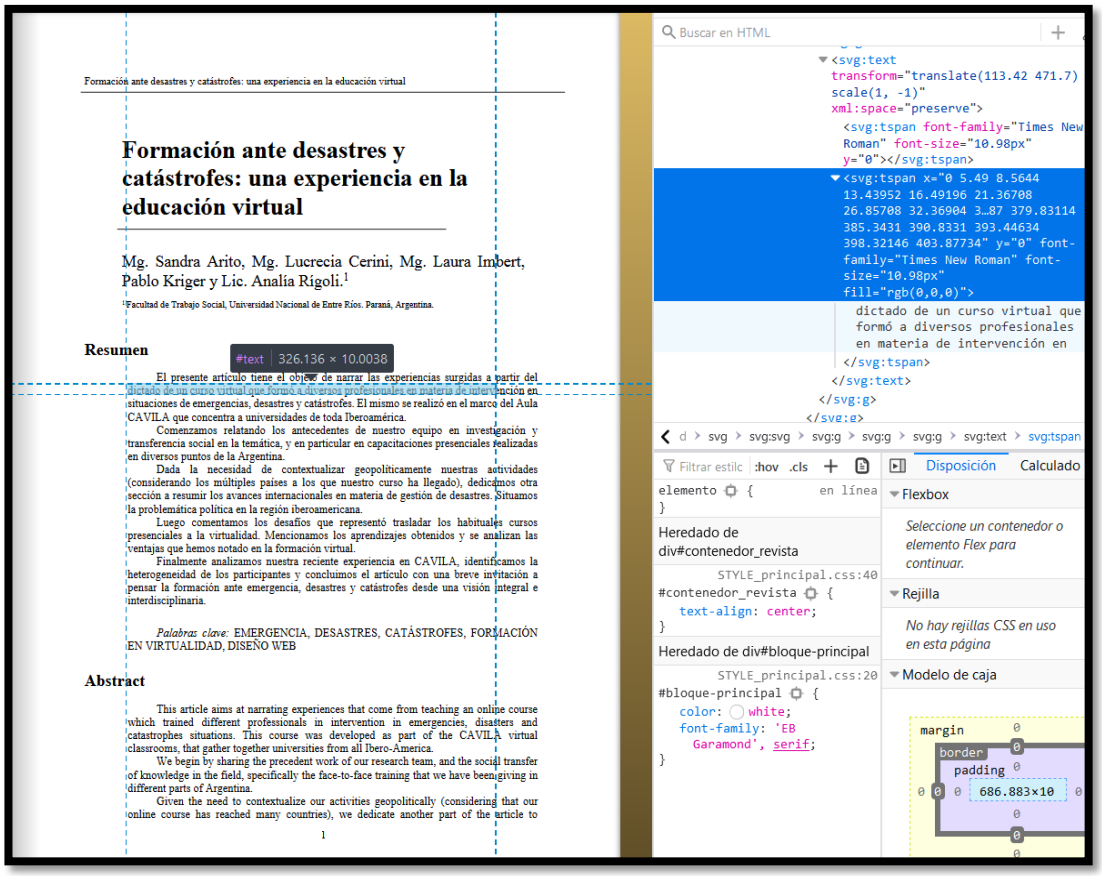
"cargarPagina" es una función que carga la página pedida, en el canvas solicitado, de un artículo concreto. "cargarPaginaSiguiente" se encarga de definir y devolver como resultado una Promise, la cual controla en su interior: que la página sobre la que se va a cargar no tenga ya contenido (para no sobrescribirlo); y comprobar que el canvas concreto sobre el que se quiere desplegar información no ha sido mal seleccionado (se comprueba por ejemplo que el canvas 11 pertenece al artículo 2, y que no pertenece al artículo 5, situación que podría generar contenido desordenado).

"cargarPagina" hace uso de la función *getPage* de pdf.js. Esta función es del objeto "pdf" obtenido al principio del código, un objeto especial del que se puede obtener una página concreta con esa función *getPage*.

Lo primero que se hace en este código es comprobar el canvas sobre el que cargar los datos, se comprueba si existe y si está vacío. En ese caso, se carga una página con la librería turn.js.

Seguidamente, se crea un elemento del tipo "svg". Después, la página del pdf devuelta por *getPage*, nombrada "paginaPDF", hace uso de la función de *getOperatorList*, que devuelve una serie de opciones estilísticas. Usando esa serie de opciones, se engarzan los svg que conforman la página.

Figura 8.16. Ejemplo de elemento svg en la revista



En la figura se ve como se selecciona, desde la herramienta del desarrollador, un segmento del texto de una página de la revista. A la derecha, se puede ver que consiste en una frase del documento, y que es un elemento svg. A esto es a lo que se

refiere esta guía cuando habla de "svg anidados". Todo el documento está formado de esta manera, por elementos svg que imitan al documento original.

En realidad, es la función "cargarPaginaSegunDireccion" quien se encarga de cargar una página en la revista. Es una función que va a llamar en su interior a la función "cargarPaginaSiguiente", pero tiene como parámetro la dirección en la que hay que cargar la página.

Esto es importante, porque el usuario puede avanzar hacia la derecha, o retroceder hacia la izquierda. No desembocará los mismos escenarios avanzar que retroceder, pues avanzar siempre supone la posibilidad de encontrarse con un artículo nuevo que aún no ha sido cargado, e ir hacia atrás siempre supone ir sobre artículos que ya han sido cargados. Si se va hacia atrás, hay que comprobar, antes de cargar una página, si por algún casual, un error o una carga mal efectuada dejaron una página en blanco, situación que el lector no debe notar. Además, cargar una página en una dirección o en otra implica un cálculo diferente. Si se avanza hacia la derecha, la carga es más obvia, pues el canvas a cargar irá aumentando progresivamente. En el momento en el que se detecte un cambio de artículo, el pdf correspondiente será cargado. Sin embargo, si se retrocede, hay que tener en cuenta que se acabará llegando a un artículo previo, salvo que en el momento del retroceso el lector se encuentre en el primero de todos los artículos, lo que supone que los contadores y variables auxiliares que se encargan de revisar la integridad del documento han de comenzar a contar desde la última página del artículo, y no desde el principio, como se hace cuando se avanza en orden y hacia la derecha.

Cuando se retrocede, hay que comprobar en primer lugar si la página previa pertenece al pdf anterior. En ese caso, los contadores toman nuevos valores. Es en ese momento en el que se comprueba si los canvas previos están bien construidos.

Hablando de que "cargarPaginaSegunDireccion" es realmente la función más visible de la adición de páginas, conviene señalar su naturaleza principal: la revista, al mostrarse siempre abierta de par en par, está mostrando dos páginas simultáneamente, por lo que esta función siempre carga dos páginas.

3.2.2.4. "actualizarPDF"

Esta función es importante porque se encarga de que la variable "pdf" tome el valor del pdf que toca leer en cada momento. Si se cambia de artículo, esta variable tiene que guardar el pdf de ese artículo, para que el resto de operaciones del script funcionen correctamente.

En primer lugar, es destacable que las comprobaciones de si hay que actualizar pdf o no se hacen dentro de esta función.

Esta función comienza por comprobar si la página del pdf que corresponde al canvas que se está comprobando excede el límite de páginas del artículo. Es decir, si el artículo llega hasta el canvas 11, y se comprueba la página 15, se comprueba que es un canvas que excede el límite, y que ya pertenece a otro pdf.

Con esa primera señal, se comprueba si el pdf al que el lector se dirige ha sido cargado, con el uso de la variable "pdfCargado". De no haber sido nunca cargado, se crean canvas de la revista para que pueda desplegarse la información. La importancia de comprobar esto es no crear canvas de más. Por mucho que se acceda a este nuevo artículo, esta sección del código solo reserva espacio para el contenido, por lo que si no es la primera vez que se accede, ya habrá espacio reservado desde la primera vez que se accedió a este artículo, y generar más canvas sólo provocará canvas que estorban.

Como filosofía estilística, se separan los artículos con páginas en blanco para que los artículo siempre empiecen en la página izquierda del par. El único artículo que rompe esta regla es el primero, que comienza en la página derecha, porque deja exactamente una página en blanco entre la portada del número y el artículo.

A la hora de crear este canvas, tiene en cuenta la regla de estilo z-index. Este atributo representa el valor de un elemento html en la pila z de visualización html. Esta pila hace referencia al nivel de profundidad de los elementos html. Un elemento que tiene un valor z-index mayor significa que está por delante de otros que tienen uno menor. Esto sirve en la revista para que unas páginas estén encima de otras. Las que se están visualizando, tienen un valor mayor que las que están detrás. Cuando se gira de página, los valores de z-index cambian, siendo los mayores valores siempre los de las páginas que se están visualizando.

A la hora de crear varias páginas seguidas, el valor de z-index no se corresponde con lo esperado. Cuando se detecta la actualización del pdf, se crean páginas en blanco, que aunque son creadas después de las últimas páginas del pdf que se está leyendo, acaban mostrándose encima. Para evitar esta situación, se asigna 0 al atributo z-index, para que quede cubierto.

3.2.2.5. "leerMultimedia"

Esta función lee el par de páginas que se están visualizando en busca de elementos multimedia a desplegar.

Para la visualización de audios y videos se utilizan marcas en el texto. Esta función se encarga de leer esas marcas, entender lo que significan, transformarlas en reproductores del contenido multimedia al que hacen referencia, y borrar la marca para que el lector no la vea. Los recursos se leen de las carpetas "audios" y "videos" que hay dentro de la carpeta del número. Cada carpeta de número tiene dos carpetas "audios" y "videos".

Figura 8.17. Marcas de texto para la carga posterior de videos y audios

@Video_1=Ejemplo de título de video#

@Audio_2=Ejemplo de título de audio#

Las marcas comienzan por un "@", seguido de "Audio" si es un audio, o "Video" si es un video, una "_" seguida de un número. Los ficheros a los que hacen referencia estas marcas han de llamarse igual, es decir, "@Video_1", hace referencia a un fichero que se tiene que llamar "video_#_1", siendo "#" el número del artículo al que pertenece el recurso. Si en este ejemplo se trata del artículo 1, el fichero debería llamarse "video_1_1", y lo mismo pasa con los audios. En la marca de texto del pdf, seguidamente, se usa "=" y después una cadena de texto. Esta cadena de texto es el título del video que el autor del artículo quiere ponerle. Para finalizar la marca, se pone un "#".

La función "leerMultimedia" obtiene todas las posibles marcas a través del atributo "par" que tengan las dos páginas, en concreto, aquellos elementos "svg:tspan", donde

estará el texto. Como no puede asegurarse en cuantos elementos svg estará dividida la marca, hay que ir coleccionando poco a poco los fragmentos de la marca hasta formarla por concreto. Es decir, puede que el principio de la marca "@Video" esté en un svg, y que en otro esté "_1=". Con la ayuda de la marca de inicio "@", y la marca de fin "#", se sabe cuándo se ha comenzado a leer la marca y cuando se ha terminado. Se inspeccionan los svg:tspan uno a uno hasta coleccionar todas las marcas. Cuando se tienen las marcas completas, se lee entre la "@" y el "=" para obtener el nombre del recurso, y entre "=" y "#" para obtener el título del recurso. Durante el proceso, se han almacenado los elementos svg que guardan el comienzo de cada marca en una lista. Esta lista, al final de esta lectura, servirá para tener guardados los puntos de la página jsp en los que están las marcas. De esta manera, podrá sustituirse el contenido de cada una de las marcas por una frase del estilo:

Figura 8.18. Marcas de texto de la figura 8.17 tras la adaptación

```
"► Reproducir Video: Ejemplo de título de video"

"► Reproducir Audio: Ejemplo de título de audio"
```

Al pulsar sobre estas marcas, se reproduce el recurso, en una ventana popup. Lo mismo sucede con las imágenes, ya que **en la función *generarImagenPopup* se analiza el par de páginas con el atributo "par", y se genera una imagen popup sobre la pantalla al pinchar una imagen del artículo, tal y como se explica en el apartado que habla sobre el fichero "leerArticulo.jsp", y el uso de "cajas modales"**.