**MANUAL READER SECUNDARIA EDELVIVES**

1. Introducción
2. Libros
3. CAT
4. Reader
   1. Fuentes de datos del reader
   2. Interfaz del reader

**Introducción**

Elementos de la aplicación

* LIBROS – Recursos + Manifiesto
* CAT
* READER

**LIBROS**

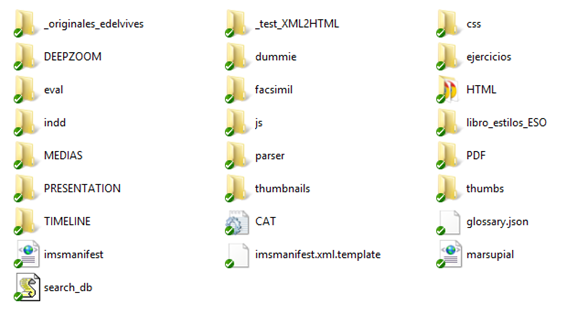
Los libros son el formato desarrollado para almacenar el contenido y que se encuentran alojados en distintos repositorios del servidor del cliente. Se trata de un conjunto de recursos web (html, imágenes, vídeo), acompañados de un manifiesto.

Cada recurso se almacenará en una carpeta diferente, por ejemplo

* HTML – Páginas
* MEDIAS – Vídeos, etc
* DEEPZOOM
* PRESENTATION
* TIMELINE
* etc

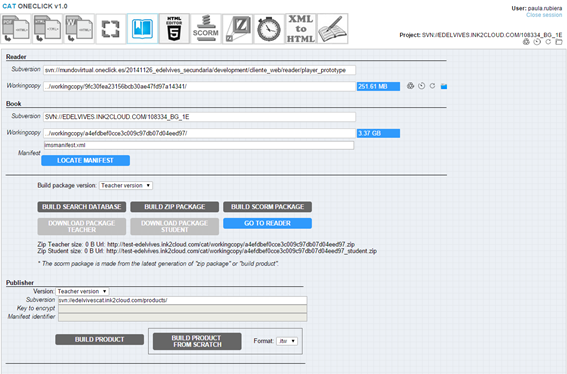
Los archivos más importantes que deben aparecer son:

* imsmanifest.xml
* CAT.ini (fichero de inicialización de la herramienta de autor)
* Glossary.json (fichero con el glosario del libro)
* Search\_db.js (fichero con la información del libro indexada para las búsquedas)



**CAT**

El CAT es un conjunto de herramientas que sirven para generar los distintos recursos web que se incorporan al libro (aunque esta tarea también se puede hacer de forma externa) y para crear el archivo con el producto final, uniendo una copia del repositorio del reader y una copia del repositorio del libro. El paquete completo se almacenará en el mismo servidor donde está el CAT.



**READER** - ***Alejandro Michelaud***

Provee de una interfaz de navegación al libro, siendo el contenedor en el que cargamos el resto de elementos con el contenido. Se trata de una aplicación web  basada en Javascript, HTML y CSS que consta de una sola página web en la que vamos mostrando y ocultando los diferentes elementos  que la componen. Al funcionar mediante javascript, toda la información se carga en local.

Se encuentra alojado en el servidor de oneclick

REPOSITORIO: svn://edelvives.ink2cloud.com/prototype\_secundaria/

El javascript del reader se deshace de la cabecera de los archivos HTML con el contenido del libro y los inserta dentro del mismo html. Los estilos CSS se comparten en la visualización final y esto puede afectar a la maquetación de las páginas del libro, por lo que dichos estilos tendrán que estar siempre referenciados a un div contenedor (con la clase xcontainer).

Para las distintas páginas obtendremos una dirección virtual que apunta a la parte del libro que queremos visualizar.

Si una página no carga, puede que lo que está leyendo esté mal cargado en el xml del manifiesto. Hay que comprobar que el ítem (el identificador al elemento del manifiesto) y enlace correspondiente sean correctos.

**FUENTES DE DATOS DEL READER**

El reader obtiene información de varias fuentes:

**Manifiesto**

Archivo xml que contiene la información del libro y que se inserta mediante jquery en el reader.

De forma simplificada se trata de una lista de enlaces ordenada según el estándar IMS Content Packaging.

<http://www.imsglobal.org/content/packaging/cpv1p1p2/imscp_infov1p1p2.html>

**Archivo lang.js**

Variable javascript con los literales estáticos de la aplicación en los distintos idiomas soportados. Cargaría la información de elementos como el nombre de las tiles, los menús de la aplicación etc

**Servicios Web**

Los servicios web son programas externos a la aplicación que nos devolverán un json con información para mostrar en pantalla como pueden ser palabras subrayadas, notas etc. Esta información se cargará en el html en tiempo de ejecución mediante jQuery.

Al estar almacenados en la nube, necesitaremos conexión a internet para poder utilizarlos.

Se les pasan al menos los siguientes parámetros:

* El Item identificador de la página html, que le dice dónde nos encontramos del libro.
* El usuario desde el que se está accediendo.
* El package identifier del manifiesto.

Requieren autenticación obligatoria por parte del servidor (la que se obtiene en el quiosco). Si no existe, esto puede hacer que se produzcan errores en cascada. Accediendo desde el CAT, de momento no tenemos dicha autenticación por lo que saltará un error.

Podemos ver estos elementos cuando se cargan en el inspector de elementos, en la pestaña NETWORK (xmls, jsons etc)

*¿Existe el manifiesto?*

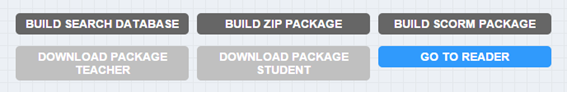
*¿Está en la ruta correcta?*

*¿Está la información correctamente incluida?*

*->Ver en el reader si se trata de un bug*

**Archivo search\_db.js**

Indexa todas las palabras que hay en el libro y las relaciona con su ítem para que al utilizar el buscador pueda mostrar la referencia adecuada y enlazar al contenido. Este archivo se genera a través del CAT mediante php



**Archivo glosary.json**

Listado de palabras y sus definiciones. No se trata de un webservice, sólo de un json que carga el javascript

**INTERFAZ DEL READER**

**https://lh4.googleusercontent.com/Gb7zTpG5lUNa2jP0F9a3JiVFlKasKRt8okkdVJwe68zjxlO4J1gaYNIxEf-1uifHyv4v4SJ6dcUVGeDHwr3xLfklUJEwxtaMJBkxe5FUjW7SCFNtThCpPj7sNkSb1HoHK28wSjg**

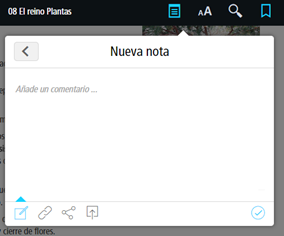
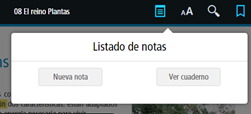
**TOC (table of contents)**

Índice del libro construido en base al manifiesto. Muestra también los distintos elementos multimedia asociados a cada capítulo.



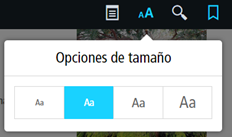
**Listado de notas**

Webservice que permite almacenar notas relativas a la página en la que nos encontramos. Al estar almacenada la información en un json, permite a un usuario acceder a los mismos datos desde distintos dispositivos.



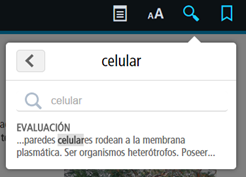
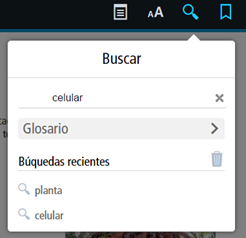
**Cambio de tamaño de letra**

Inserta un estilo css inline que permite modificar el tamaño de la fuente de texto mediante javascript.

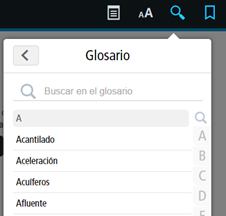


**Buscador y glosario**

No basa las búsquedas en consultas a un servidor como puede ser el caso de Google, sino que las llamadas a las búsquedas son internas y relativas al archivo search\_db.js que ya está indexado, por lo que se realiza de forma casi inmediata.



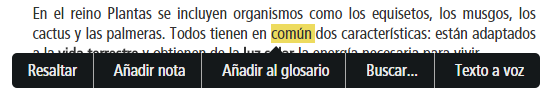
El glosario funciona de forma parecida, cargando un json con la información.



**Botón de marcadores**

Webservice que almacena los favoritos del usuario.

**Menú contextual**

****

Proporciona acceso a otras funcionalidades basadas también en webservices.