**Guía de uso Hugo-zDoc**

Hugo es un generador de sitios estáticos. Hugo provee de utilidades, pero necesita de un tema con el que generar el aspecto visual y sus características.

Para instalar Hugo podemos seguir el tutorial de su página web: <https://gohugo.io/getting-started/quick-start/> una vez tengamos Hugo instalado podemos instalarle un tema.

En nuestro caso, tras comparar los temas existentes hemos elegido zDoc, ya que nos parece el más potente.

Para instalar zDoc podemos seguir el tutorial: <https://zzodocs.netlify.com/docs/gettingstarted/installation/>

La elección de zDoc se debe a que es un tema muy maduro y con muchas de caja. Tiene pequeños detalles que hacen de él un tema muy agradable, como: “dark theme”, integración con idiomas, auto generado de índice tanto global como del documento que estamos visualizando, o el favorecer mostrar código de lenguajes de programación de forma amigable y colorida.

Además es un tema que favorece el uso de un buscador (utiliza lunr y fuse para ello).

Dada la naturaleza estática de nuestro sitio web no es posible realizar búsquedas en el servidor. Para saltar esta limitación, el buscador utiliza un archivo “index.json” generado en tiempo de construcción. Este archivo aglutina la información de todo el sitio web, de forma que para realizar búsquedas el cliente se descarga este archivo y reliza las búsquedas sobre él de forma local.

Teniendo en cuenta que en versiones futuras se contempla la posibilidad de tener una zona privada se ha modificado este comportamiento para que se ajuste a nuestras necesidades.

En la actualidad la búsqueda se realiza únicamente en la carpeta seleccionada (“servidor/idioma/carpeta”, por ejemplo: “pagina/en/docs” tendría su índice “index.json” y “pagina/en/blog” el suyo propio) de esta forma podemos mantener los archivos “index.json” locales a cada una de las subcarpetas y en caso de bloquear el acceso a una de las rutas automáticamente se restringiría el uso de su índice evitando así los problemas de seguridad.

El archivo “index.json” contiene toda la información indexada, con la finalidad de realizar búsquedas sobre la mayor cantidad de contenido posible. Este archivo puede crecer y volverse demasiado pesado si la cantidad de páginas a indexar es muy grande, en caso de que eso pase, se podría eliminar el cuerpo de los artículos. Para ello deberíamos acceder a: “proyecto\layouts\\_default\list.searchindex.json” y modificar la línea en la que se genera el índice de cada archivo eliminando su contenido de los campos a indexar (línea 9 hay que borrar ("content" $element.Plain “) )

{{- $.Scratch.Add "searchindex" (dict "id" $index "title" $element.Title "uri" $element.Permalink "tags" $element.Params.tags "section" $element.Section "content" $element.Plain "description" $element.Description) -}}

A a partir de ese momento el índice será muchísimo más liviano y rápido, pero a su vez las búsquedas utilizarán menos información sobre la que buscar.

Una vez tenemos hugo y el tema seleccionado instalados podemos utilizar el comando “hugo server” en la terminal (hay que lanzarlo en la carpeta del proyecto). Esta funcionalidad nos permite ver los cambios en caliente, es decir, si servimos nuestra página web con “hugo server” podremos ver los cambios en vivo según trabajamos sobre el proyecto.

Esta funcionalidad es muy útil a la hora del desarrollo y también puede ser utilizada en producción si se elige hugo como servidor para el despliegue.

Hugo nos provee con una funcionalidad llamada “hugo” (hay que lanzarlo en la carpeta del proyecto) para generar los archivos que formarán el sitio estático. El comando genera una carpeta llamada “public” dentro del proyecto. Esta carpeta contiene el resultado de generar el sitio estático y puede ser servida con cualquier servidor web de estáticos. En mi caso para desacoplar la generación del servidor utilizo el comando “http-server” (dentro de la carpeta “public”), un servidor web que corre en nodejs: <https://www.npmjs.com/package/http-server>

El contenido a servir se organizará en carpetas dentro de la carpeta “content”, el primer nivel corresponde al idioma, de esta forma cada idioma tendrá a su vez su estructura de carpetas.

Cada carpeta contenedora ha de tener un archivo llamado “\_index.md”, este archivo contiene información sobre la carpeta en sí, no como nodo final.

Las carpetas pueden a su vez tener subcarpetas siempre que contengan su propio archivo “\_index.md” serán visualizadas como nodos contenedores.

También pueden contener archivos “\*.md” que serán visualizados como elementos finales y se renderizarán al pulsar sobre ellos.

Los archivos “\*.md” son de tipo “mark down”, se puede ver un ejemplo de qué tipo de cosas se puede hacer con ellos aquí: <https://guides.github.com/features/mastering-markdown/> y un ejemplo de editor aquí: <https://markdown-it.github.io/> este tipo tiene una gran aceptación debido a su sencillez y admite inserción de html, lo que les hace idóneos para la generación de páginas web.

Dentro de los archivos “\*.md” podemos seleccionar su nivel de precedencia añadiéndoles un atributo con un peso en su cabecera ej:

weight: -3

De esa forma podemos ordenarlos dentro de las carpetas. Las carpetas a su vez pueden tener ser ordenadas de esta forma añadiendo el atributo en el archivo “\_index.md”.

Un artículo puede ser “invisible” y no aparecer en la generación final si está marcado como draft:

draft: true

Uno de los puntos fuertes de hugo es que, una vez seleccionado un tema está listo para funcionar prácticamente de inmediato, las funcionalidades son en muchos casos más que suficientes para lo que podemos necesitar, incluso superiores en algunos aspectos.

El punto débil es la falta de documentación. Aunque existe una documentación bastante amplia sobre las variables que usa hugo no existe un lugar en el que se expliquen sus funcionalidades avanzadas, la estructura de carpetas puede llegar a ser un quebradero de cabeza. La forma en la que los archivos son inyectados o llamados en la construcción no está del todo clara. Tampoco hay buena documentación sobre cómo funcionan los plugins usados por la comunidad y en muchos casos su documentación está dispersa en diferentes blogs no enlazados.

Lo mismo pasa con los temas, en su mayoría la documentación es muy pobre o inexistente.

En resumen: Tanto Hugo como zDoc son herramientas muy útiles y con una gran funcionalidad, aunque la falta de documentación hace que los desarrollos sean más lentos y en ocasiones necesiten de ingeniería inversa o prueba y error para extender sus funcionalidades.