Documentación

La elaboración de software genera mucha documentación en cada una de las fases del proyecto. Es importante que esta documentación sea de calidad y comprensible para la persona a la que va dirigida. En casos extremos puede que sea mejor no escribir documentación a que sea inservible o poco útil. Muchas veces la mala calidad de la documentación viene de la persona que debe escribir, normalmente el programador, puestos que en muchas ocasiones carece del tiempo suficiente o no le gusta escribir documentación y lo hace de mala gana.

Cómo escribir con una cierta calidad

Algunos pautas o consejos para escribir documentación de calidad son:

- Hacer esquemas antes de ponerse a escribir. Incluso pueden añadirse los esquemas a la documentación.
- Clasificar la información según la importancia. Es mejor tener un buen documento principal y muchos anexos que un documento principal excesivamente grande.
- Realizar resúmenes, esquemas, documentos maestros, etc. Siempre es bueno hacer un trabajo de síntesis en el que se explique el conjunto de la documentación.
- Muchos lenguajes de programación tienen sus propios estándares y herramientas para generar documentación. Si es el caso es mejor usarlo. Un ejemplo sería Javadoc para Java.
- Al escribir la documentación hay que anticiparse y responder a las posibles preguntas que se pueda hacer el lector.
- A la hora de escribir debe primarse la claridad. Por eso, a muchas empresas les gusta usar la segunda persona, tú o usted, en vez de la tercera persona genérica, el usuario.
- Hay que escoger una herramienta de documentación adecuada. En ciertos casos puede no ser adecuado un procesador de texto y sea mejor una herramienta colaborativa como una wiki.
- Muchas veces se generan dos documentos de usuario:
  - ✗ **Guía de usuario**. Explica cómo realizar ciertas tareas del software de manera sencilla, normalmente en formato tutorial.
  - ✗ **Manual de referencia**. Explica el software más profundamente y muchas veces añade archivos de ayuda.
- Hay que tener en cuenta el nivel del lector o usuario al que va dirigida la documentación. En estos casos hay que dejar claro qué no puede o no debe hacer el usuario y sus posibles consecuencias.

Tipos de documentación

Un software pude verse desde dos puntos de vista distintos y la documentación debe cubrir ambos aspectos:

- **Aspecto técnico**. Son los componentes, clases, ficheros, interfaces, bases de datos, etc. Orientado a desarrolladores.
- **Aspecto funcional**. Cómo funciona el sistema, qué acciones realizan cada uno de los componentes, cómo se comunican, etc. Destinado a los usuarios del programa.

Debemos escribir documentación específica dependiendo de dónde nos encontremos en el ciclo de vida del software:

- **Fase inicial**. Se establecen las bases de cómo va a desarrollarse el resto del proyecto. Esta fase deben acometerla gente experimentada. Entre otras cosas se planifica el proyecto, se hacen estimaciones y se decide si es o no rentable. Se deben escribir varios documentos tanto de planificación, tanto general como detallada y de estimaciones de datos económicos, posibles soluciones al problema con sus costes, etc.
- **Análisis**. Se analiza el problema. Hay que recopilar información, examinar y formular los requisitos del cliente y analizar cualquier restricción que pueda aplicarse. Las entrevistas con el cliente deberían registrarse y tiene que quedar muy claro qué se espera que se haga, el tiempo estimado, el costo, etc. Estos documentos son muy importantes y evitan que se vayan modificando cosas sobre la marcha. Estos documentos, normalmente, tienen carácter contractual y generan obligaciones a ambas partes.
- **Diseño**. Consiste en determinar los requisitos generales de la arquitectura de la aplicación y dar una definición precisa de cada subconjunto de la aplicación. Se suelen escribir dos documentos técnicos, uno más genérico en el que se tiene una visión más general, y otro detallado en el que se tratan los detalles técnicos de cada módulo concreto del programa por separado. Estos documentos los suelen escribir los analistas con la supervisión del jefe del proyecto.
- **Codificación o implementación**. Consiste en la implementación del software en un lenguaje de programación para crear las funciones definidas durante la etapa de diseño. Entre otras cosas se explica las entradas, salida o el propósito de cada clase o método, quién lo ha escrito o cuando se revisó. Se escriben documentos muy detallados con el código ya que en un futuro se usará para el mantenimiento, posiblemente otra persona.
- **Pruebas**. Se realizan pruebas para asegurar que la aplicación funciona como se espera de ella, no hay errores y se comunica perfectamente con otros elementos necesarios. Cada tipo de pruebas puede describir en un documento separado. Los tipos de pruebas son:
  - ✗ Las **pruebas funcionales** comprueban que el programa hace lo que debe hacer. Muchas veces se hacen delante del cliente.

- ✗ Las **pruebas técnicas** son pruebas simulando situaciones reales, con mucha carga,...

- **Explotación**. La instalación y uso en un entorno real es normalmente es la fase más larga del proyecto, aunque surgen incidencias y nuevas necesidades. Se debe escribir un documento con todos los fallos o bugs. Hay que ser lo más detallado posible para intentar hallar la mejor solución. Las nuevas necesidades hay que detallarlas en otro documento que sirve de base para las labores de mantenimiento.
- **Mantenimiento**. Se realizan todo tipo de procedimientos correctivos o corrección de fallos y actualizaciones secundarias de software, lo que sería el mantenimiento de la aplicación. El mantenimiento debe documentarse para saber quién, por qué, cómo y cuándo ha hecho algo. Las labores de mantenimiento son mucho más fáciles con una buena documentación anterior.

Ninguna aplicación debería entregarse si no cuenta con, como mínimo, los siguientes documentos:

- **Manual de usuario**. Explicar cómo funciona la aplicación, que puede o no hacer y cómo hacerlo. Debe ser claro y fácilmente entendible para el usuario que utilice la apliación.
- **Manual técnico**. Dirigido a los responsables del mantenimiento. Alguien que tenga conocimientos del lenguaje del programa debería conocer perfectamente la aplicación tras su lectura.
- **Manual de instalación**. Se explica la instalación paso a paso, los requisitos del sistema y cómo se pone en funcionamiento la aplicación.

Plantillas

Una plantilla es un documento base con una estructura o diseño predefinido y en el que solo hace falta insertar los datos. Las plantillas agilizan el trabajo ya que normalmente solo hace falta añadir los datos concretos en el sitio adecuado. No es necesario que sea un documento completo. Se pueden usar plantillas en partes concretas de un documento como casos de uso con descripción, actores, precondiciones, curso normal, postcondiciones, alternativa, etc.

Comentarios

Un **comentario** es, según la RAE, una explicación de un texto para su mejor intelección (acción y efecto de entender). En un entorno de programación podemos entender que es una anotación que hace referencia a algo del código. Los comentarios son ignorados por el compilador y es como si no hubiese nada escrito. Usos posibles de los comentarios son:

- **Describir el código**. El uso más concreto de los comentarios tiene que ver con documentar las partes del programa por medio de descripciones

básicas.

- **Mejorar el código**. Se suele utilizar de cara a mejorar el código fuente, pensando en la las siguientes revisiones del código.
- **Control de versiones**. Los comentarios también se suelen utilizar para indicar las funcionalidades añadidas en cada versión, las correcciones, etc.
- **Informar de errores**. Informar que una parte del código no hace lo que debería o lo hace, pero no de la manera en qué debería hacerlo.

La sintaxis de los comentarios depende del lenguaje de programación que se esté usando aunque siempre pueden ser:

- **De bloque o multilínea**. Permite comentar varias líneas, párrafos incluso. Todo el texto encerrado entre la apertura y el cierre se considera comentario. En java se usa **/*** para abrir y ***/** para cerrar.
- **De línea**. Solo afecta a una línea desde el lugar en el que se pone el delimitador de comentario. En Java es **//**.

Algunos consejos a la hora de escribir comentarios son:

- Usar nombres descriptivos. Clases, métodos y variables deben tener nombres descriptivos. En su defecto habría que seguir un patrón de definición de nombres de los elementos puesto que un código más claro necesita menos comentarios.
- Seguir siempre el mismo estilo. Si en cada sitio tenemos un estilo distinto, el lector solo puede acabar confundido. Para esto lo mejor es seguir una nomenclatura clara.
- Los comentarios deben ser útiles. No se debería comentar lo obvio, lo que se debe comentar son cosas que aporten algo.
- Pensar si un comentario es necesario antes de añadirlo.
- Evitar comentarios dentro de métodos o funciones. Discutible. Los comentarios fuera del método deberían explicar qué se hace y los de dentro del método cómo se hace. Según la complejidad de lo que se haga puede ser necesario comentar qué y cómo se hace.
- No es necesario comentar cada línea de código, comentar un bloque suele ser suficiente para entender las cosas varios meses después.
- Mantener los comentarios. Cuando se actualiza el programa también hay que actualizar los comentarios en la parte que les toque. No es útil un comentario de algo que se eliminó tres versiones atrás.
- No dejarlos para el final. Si parece aburrido comentar al tiempo que se hace el programa, comentar luego todo es proporcionalmente más aburrido y, al hacerlo con menos ganas, la calidad del texto se resiente.
- Ser educado. Nunca se sabe quien va a leer los comentarios.

Javadoc

Javadoc es una herramienta que genera páginas html a partir de los comentarios de documentación de archivo una archivo de código fuente de Java (.java). La inmensa mayoría de los IDE de Java permite la generación automática de Javadoc.
Javadoc utiliza etiquetas y comentarios. La documentación está encerrada en comentarios de bloque que empieza por /** y acaba por */.
Las etiquetas o parámetros definen las diversas partes de la documentación. Se pueden usar en clases, en métodos o en ambos. Las etiquetas empiezan por @. Si no se usa @ dará un error al intentar generar la documentación. Algunas etiquetas pueden usarse varias veces en un mismo bloque.
Las etiquetas que se pueden usar son:

- **@author**. Solo clases e interfaces. Nombre del autor de la clase que se documenta.
- **@deprecated**. La clase ha sido reprobada y no debería seguir usándose. Puede haber varias razones como problemas de seguridad o uso de la clase, haberla escrito con malas prácticas o que haya una nueva versión más actualizada. Normalmente se usa @see para indicar la nueva opción válida.
- **@exception**. Igual que @throws aunque definió antes y se considera que es mejor o más exacto usar @throws.
- **@param**. Solo métodos y constructores. Define un parámetro de entrada.
- **@return**. Solo métodos. Describe el valor de salida del método.
- **@see**. Enlace o referencia a otro sitio, ya sea clase, método, guía,…
- **@serial**. Describe el motivo del campo y sus posibles valores. También puede usarse **@serialField** o **@serialData**.
- **@since**. Indica a partir de qué versión se ha añadido el componente.
- **@throws**. Excepción que puede lanzar el método y hay que tener en cuenta.
- **@version**. Solo clases e interfaces. Número de versión de la clase. Es decisión de los programadores establecer un sistema de versiones, pero una vez establecido no deberían modificarlo.

El orden recomendado para poner las etiquetas es @author (solo clases y requerido), @version (solo clases y requerido), @param (solo métodos), @return (solo métodos), @exception (o su sinónimo @throws), @see, @since, @serial, @deprecated.

Markdown

Markdown es una herramienta de conversión de texto plano a html. Surge para hacer texto con formato en texto plano, ya que los procesadores de texto, aunque tienen muchas más opciones, usan formatos que no son comprensibles por las personas.
Markdown pretende crear documentos que sean totalmente comprensibles para humanos y máquinas. Muchos editores permiten escribir en html, por ejemplo, pero no mucha gente domina html.
Markdown no pretende ser un sustituto de html ya que es consciente que no tiene tantas opciones de formato como CSS. Sin embargo hay pluggins en CMS que permiten editar el contenido en Markdown.
Sintaxis de Markdown:

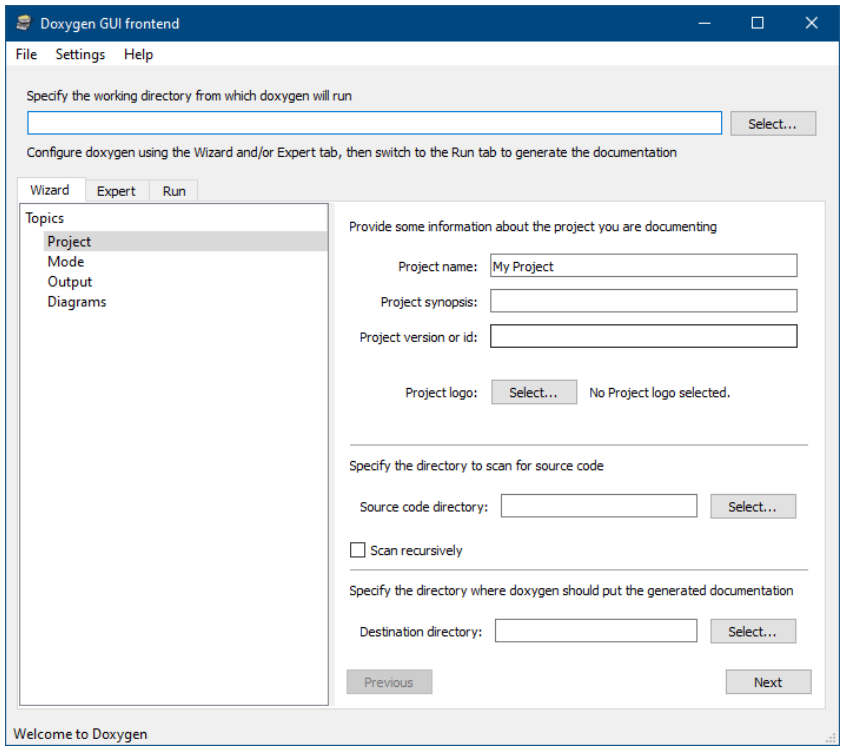
- Elementos de bloque:
  - ✗ **Párrafos**. **Dos veces intro**. Pone una línea en blanco.
  - ✗ **Saltos de línea**. No termina el párrafo pero continúa escribiendo en la siguiente línea. **Dos espacios más intro**.
  - ✗ **Encabezados**. Usar una almohadilla, **#**, por número de encabezado hasta 6. No hace falta cerrar el encabezado.
    - Otra forma de hacer encabezados 1 y2 es poniendo debajo de la línea del encabezado un **=**, igual, para h1 y **–**, guion, para h2.
  - ✗ **Citas**. Se añade un **>**, mayor que, al inicio del bloque.
    - Si la cita consta de varios párrafos hay que poner un > antes de cada párrafo.
    - Se pueden concatenar varios **>>** para hacer citas anidadas.
  - ✗ **Listas**.
    - **Desordenadas**. Se usa delante de cada elemento de la lista un *****, asterisco, **-**, guion o **+**, suma.
      - Se pueden intercambiar los símbolos.
      - Para anidar una lista hay que añadir **cuatro espacios** en blanco antes del símbolo de lista.
    - **Ordenadas**. Se usa **número + punto + espacio** antes del elemento de la lista. Se puede usar siempre el mismo número.
      - Se pueden anidar listas igual que desordenadas, **cuatro espacios**.
  - ✗ **Bloque**. Para crear un bloque o remarcar una zona hay que usar tres virgulillas (alt + 126), **~~~**, antes y después del texto del texto del bloque.
  - ✗ **Regla horizontal**. Tres asteriscos, guiones o guiones bajos. (*******, **---**, **___**). Pueden estar juntos o separados. (*** * *, - - -, _ _ _**).
- Elementos de línea:
  - ✗ **Énfasis**. Se puede, antes y después de las palabras a remarcar, usar asteriscos, *****, o guiones bajos, **_**.
  - ✗ **Cursiva.** Un único asterisco, *****, o guion bajo, **_**.

- **Negrita**. Dos asteriscos, ******, o guiones bajos, **__**.
- **Ambos**. Tres asteriscos, *******, o guiones bajos, **____**.
- **Enlaces**. La palabra del enlace se pone entre corchetes, **[]**, mientras que el enlace se pone entre paréntesis, **()**.
  - Se puede hacer un enlace de referencia, similar a hover, usando palabras de referencia, que puede estar en otro sitio del texto, con formato **[referencia]: http://www.sitiodelenlace.com**.
  - Si se quiere ver la ruta del enlace, la url, en vez de la referencia, hay que usar ángulos, **<>**, en la ruta del enlace.
- **Código**, similar a la etiqueta <code> de html. Se encierra el texto entre comillas sencillas, **‘** (la de al lado de la p).
- **Texto preformateado**, similar a la etiqueta <pre>. Empezar cada línea del texto con **cuatro espacios**.
- **Imágenes**. Usar exclamación, **!**, antes de un enlace a una imagen. **![Texto alternativo](/ruta/a/la/imagen.extension)**.
  - **Texto alternativo**. Usar comillas, **"**, tras la ruta de la imagen.

Si necesitamos usar un carácter que Markdown usa para formatear el texto como asteriscos o guiones, debemos usar la barra invertida, ****.

Doxygen

Doxygen es una herramienta muy versátil que genera documentación en varios lenguajes de programación como Java, C o Fortran. El formato de salida de la documentación es muy variado y se puede obtener, entre otros, html, xml, rtf o pdf.
La instalación es muy sencilla, basta con descargar el archivo ejecutable desde la página oficial y usar el asistente de instalación.



En la pantalla principal del modo asistente tenemos el menú de la aplicación, la ruta donde se generará la documentación y varias pestañas con paneles de configuración.
En la pestaña **Wizard** se introduce la información mediante el asistente.
En el tema Project está la información básica del proyecto:

- **Project name**. Nombre del proyecto.
- **Project synopsis**. Resumen o breve descripción del proyecto.
- **Project version**. Versión de la aplicación.
- **Project Logo**. Imagen del logotipo que representa la aplicación.
- **Source Code directory**. El directorio de la aplicación que documentamos.
- **Destination directory**. Carpeta donde se guardará la documentación que se genere.

El botón next cambia de tema y pasa a Mode.

El tema **Mode** podremos establecer qué tipo de documentación queremos escribir.

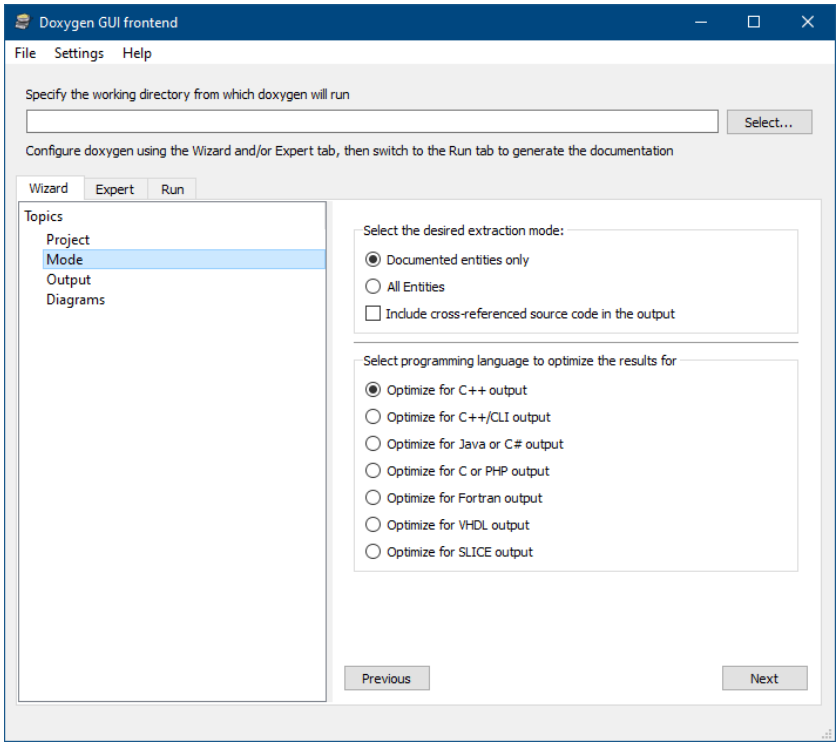
- **Extraction mode**. Especificamos qué queremos documentar.
- **Select programing languaje**. Lenguaje de programación que estamos documentando para optimizar los resultados.

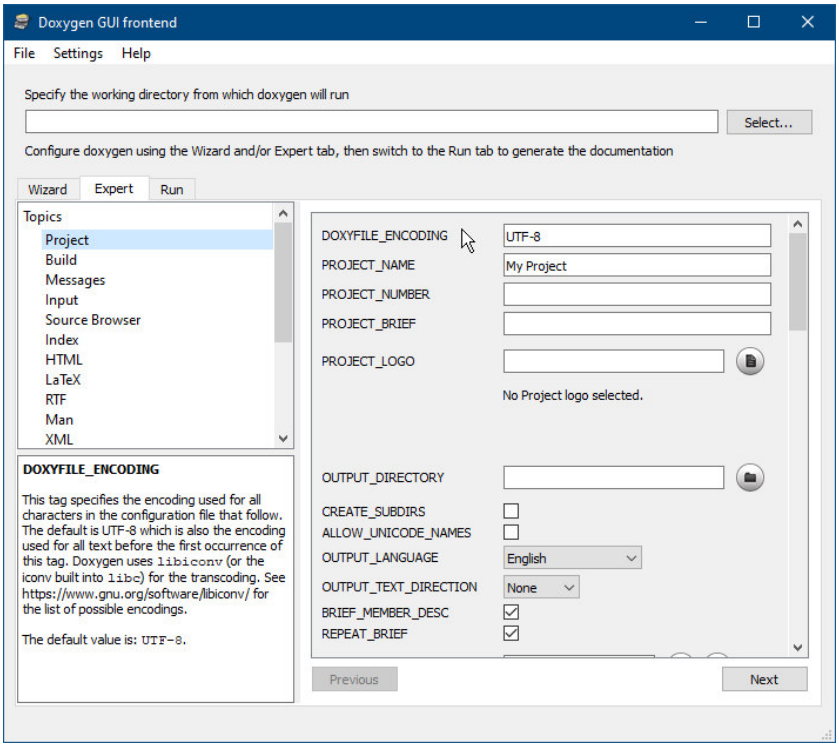
En el tema **Output** decidimos el formato de salida en el que queremos la documentación. Entre otros podemos escoger html, pdf, rtf o xml.

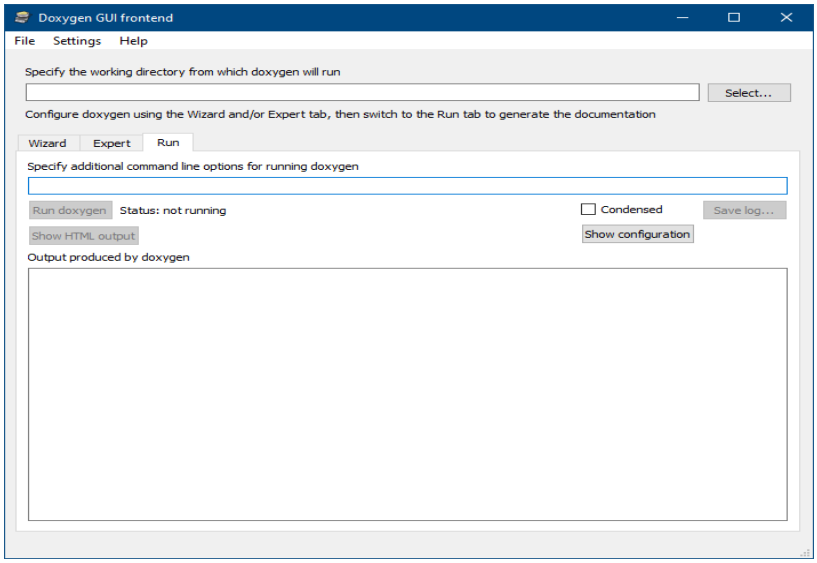
En el tema **Diagrams** podemos añadir diagramas si los tenemos preparados o usar alguna herramienta como GraphViz para generarlos.

En la pestaña **Expert** podemos fijar los detalles de los puntos que se han establecido en el asistente mucho más concretamente.

En la pestaña **Run** se inicia la generación de la documentación una vez esté todo configurado.







Además de crear documentación normal, Doxygen genera documentación html usando usando lo comentarios o las anotaciones contenidas en los archivos de código fuente. También soporta la sintaxis de Markdown.

Cada lenguaje de programación tiene su propio formato para los comentarios. Algunos ejemplos son:

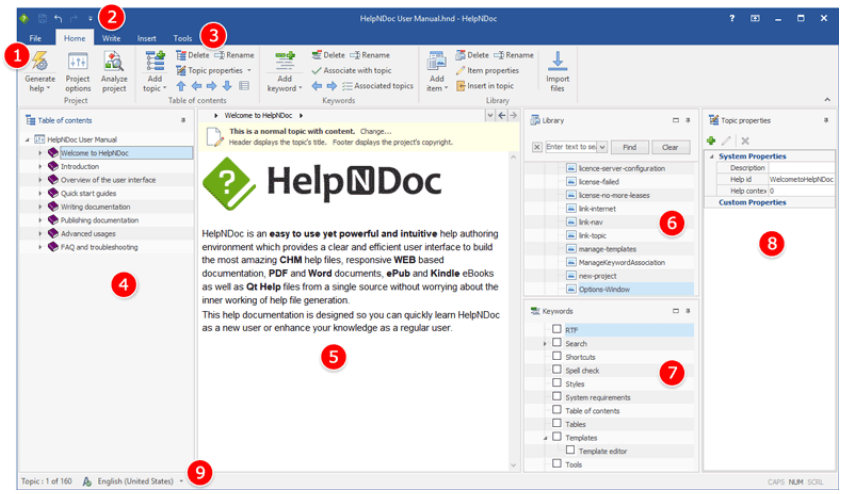
- **/**** en la línea inicial, ***/** en la final y * en cada línea restante.
- **///** en cada línea.
- Para que se vea mejor, hay a quien le gusta poner * o / toda la línea superior e inferior.

Se pueden utilizar los parámetros de javadoc. Además, Doxygen tiene muchos comandos propios. Se puede leer la descripción de éstos en https://www.doxygen.nl/manual/commands.html. Para usarlos hay que hay que escribir **** o **@** antes del comando.

HelpNDoc

HelpNDoc es un entorno de autoría de ayuda fácil de usar, pero potente e intuitivo, que proporciona una interfaz de usuario clara y eficiente para construir los archivos de ayuda CHM más sorprendentes, documentación basada en la web con capacidad de respuesta, documentos PDF y Word, ePub y Kindle eBooks. La carpeta de salida por defecto es Documentos > HelpNDoc. CHM (Archivo de ayuda HTML compilado) es un formato de ayuda online desarrollado por Microsoft.

Los elementos de la pantalla de helpndoc con sus opciones son:



1. Menú Archivo.
   - ✗ Gestiona los proyectos; crear, abrir, guardar,...
   - ✗ Acceso a proyectos y lugares recientes
   - ✗ Opciones de la aplicación
   - ✗ Acceso a la ayuda de HelpNDoc
   - ✗ Salir de la aplicación
2. Barra de herramientas de acceso rápido.
   - ✗ Acceso a acciones de uso frecuente como "Guardar proyecto", "Deshacer" y "Rehacer"
3. Barra de herramientas de la cinta de opciones
   - ✗ Contiene todas las acciones disponibles en HelpNDoc
   - ✗ Puede minimizarse para proporcionar un mayor espacio en la pantalla de edición de la documentación
4. Tabla de contenidos
   - ✗ Define y gestiona la jerarquía de temas para el proyecto actualmente abierto
   - ✗ El tema raíz es el tema del proyecto, utilizado para ver y modificar la configuración del proyecto
   - ✗ Al seleccionar un tema se mostrará su contenido asociado para su edición
5. Editor de temas
   - ✗ Se utiliza para editar el contenido del tema seleccionado
   - ✗ Configura el origen y el comportamiento del tema

6. Biblioteca
   - ✗ Define y gestiona los elementos multimedia y reutilizables como imágenes, películas, fragmentos, documentos incluidos…
   - ✗ Añade elementos a los temas
7. Editor de palabras clave
   - ✗ Define y gestiona la jerarquía de palabras clave para el proyecto actualmente abierto
   - ✗ Asocia las palabras clave a los temas individuales
8. Propiedades de los temas
   - ✗ Define las propiedades del sistema del tema
   - ✗ Gestiona las propiedades personalizadas del tema
9. Barra de estado
   - ✗ Obtiene estadísticas sobre la documentación
   - ✗ Gestiona los diccionarios y las opciones del corrector ortográfico
   - ✗ Obtiene información sobre el estado del teclado

Algunos temas básicos importantes son:

1. **Crear un nuevo proyecto**.
https://www.helpndoc.com/documentation/html/Createanewproject.html
2. **Generar la documentación**.
https://www.helpndoc.com/documentation/html/Generatingdocumentation.html
3. **Añadir temas**.
https://www.helpndoc.com/documentation/html/Addingtopics.html
4. **Editor de estilos**.
https://www.helpndoc.com/documentation/html/Styleseditor.html
5. **Asociar palabras clave con temas**. Indexa palabras a temas para encontrar más fácilmente la información que se está buscando.
https://www.helpndoc.com/documentation/html/Managekeywordassociation.html
6. Uso de la **biblioteca**, una zona central de almacenamiento para todo tu proyecto, en donde se guardan los archivos multimedia, incluyendo imágenes, fotografías, vídeos, documentos, código HTML, variables y fragmentos de código.
https://www.helpndoc.com/documentation/html/Usingthelibrary.html

Documentación

Javadoc

https://docs.oracle.com/javase/7/docs/technotes/tools/windows/javadoc.html
https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html

Markdown

https://markdown.es/ página no oficial en español.
https://daringfireball.net/projects/markdown/dingus. Editor online.
https://dillinger.io/. Editor online.
https://markdownlivepreview.com/. Editor online.

Doxygen

https://www.doxygen.nl/download.html
https://www.doxygen.nl/manual/index.html
https://www.doxygen.nl/manual/commands.html

HelpNDoc

https://www.helpndoc.com/documentation/html/index.html
https://www.helpndoc.com/es/ayuda-online/
https://www.helpndoc.com/es/guias-paso-a-paso/