El futuro digital es de todos | MinTIC

UNIVERSIDAD EL BOSQUE | MisiónTIC 2022

Ciclo 2

Semana 4

Data Access Object (DAO) y Data Transfer Object (DTO), manejo de excepciones y documentación de código

Lectura 3 - Documentación de código

Convenio de asociación no. 777 de 2021 suscrito entre el Fondo Único de Tecnologías de la Información y las Comunicaciones y la Universidad el Bosque.

Documentación de código

Importancia de la documentación

Independientemente del tipo de aplicación que se haya desarrollado, siempre es importante contar con una documentación clara y actualizada, para que no solo quien realiza la aplicación tenga todo claro, sino para que cualquier persona que interactúe con la misma desde el usuario final hasta algún programador que desee hacer mantenimiento a la misma, pueda entender de la mejor manera la lógica y desarrollo realizado, por tanto es importante establecer estándares de documentación para conseguir una documentación que cumpla su objetivo. Gracias a la definición de dichos estándares de documentación, todo el equipo de trabajo estará sincronizado en el cómo encontrar, desarrollar o distribuir la documentación.

Existe diferentes tipos de documentación para los cuales se podrían crear estándares diferentes que cambiaran según la función del tipo de documento, por tanto, no es lo mismo crear estándares para documentación de proyectos, documentar procedimientos o la documentación técnica y del software en donde se tendrá información de cómo funciona, cómo se usa, cómo se creó y cómo se compiló.

Los grupos de trabajo que trabajan con software pueden consultar documentación para revisar requisitos del producto, notas de publicación, especificaciones de diseño, o usar la misma para detallar el código, las API y registrar los procesos de desarrollo de software, y aunque esta pueda ser extensa y pormenorizada se puede hacer uso de herramientas adecuadas y así garantizar la exactitud y ahorrar un poco de tiempo.

Competencias de la documentación

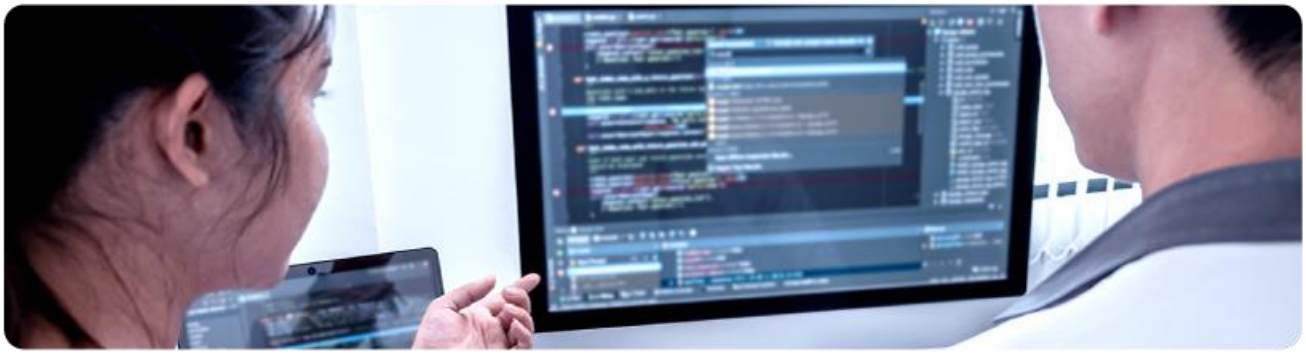
Como dijimos antes, existen diferentes tipos de documentación, pero existen unas competencias o características esenciales que toda documentación debería tener, para así garantizar que lo realizado cumpla con los estándares de documentación definidos y así sean útiles para los usuarios. Algunas de estas competencias son:

Organización: Aunque conozcamos bien la información que se quiere documentar, la forma en que se presenta debe ser correcta para así poderla seguir con facilidad en el momento de consultarla. No solo se debe organizar de manera lógica, sino también es importante el uso de encabezados de sección, listas con viñetas y lo que se requiera para que la guía para el usuario sea la mejor.

Claridad. Es necesario elegir un adecuado nivel de detalle que permita que la documentación termine siendo clara y fácil de entender. Se debe buscar un equilibrio en la cantidad de información a colocar pues aunque tener cantidad insuficiente de información puede hacer el documento un tanto confuso, el colocar demasiada también. Se debe ser concreto y elegir bien el uso de la gramática a usar.

Homogeneidad. Esta característica es muy útil en especial cuando se va a hacer uso de la documentación ya que si todos los documentos tienen homogeneidad resulta más fácil el seguimiento de la misma.

Documentación de programas



Los programas de software no solo se hacen para sean usados por otras personas sino también para que sean leídos por otras personas.

El estilo de programación está relacionado con la legibilidad del programa, por tanto, se deberían usar técnicas de programación propuestos por programadores experimentados aprovechando toda posibilidad para asegurar que el código sea claro y fácil. Tengamos en cuenta que tener un código claro no es algo que le importe al compilador pues independiente del estilo o versión de código colocado, este lo ejecutará; es siempre y solamente el programador quien se beneficia de la claridad del código.

Hay algunas recomendaciones que nos pueden servir en el momento de codificar programas, que contribuirán a que el código sea claro.

Existen comentarios dentro del programa que nos ayudaran, como lo son:

- **Prólogo:** Comentarios que acompañan a subprogramas o rutinas del programa, donde normalmente se coloca información de finalidad de la rutina, significado de variables importantes, en fin, indicaciones generales de cómo trabaja el subprograma con aspectos relevantes de uso.

- **Directorio:** Comentarios que colocan al inicio de los ficheros que componen un programa y muestran los subprogramas contenidos en ese fichero.

- **Explicatorios**: Comentarios que se ponen en cualquier parte del código y sirven para explicar algo puntual. Pueden ser de una sola línea o tipo bloque.

Tengamos en cuenta que se debe medir la cantidad de comentarios que se incluyan en el código, ya que tener pocos, pero también muchos, podría ser perjudicial en un momento dado, así que realmente la cantidad de comentarios dependerá de la complejidad de las rutinas.

- **Legibilidad:** hacer todo lo posible para que el código sea legible, por ejemplo, con el uso de espacios o líneas en blanco.

- **Declaraciones:** se pueden hacer declaraciones a variables importantes para conocer su propósito. Lo ideal es que cada variable tenga un propósito único y que no se reutilicen variables. Es preferible declarar variables adicionales que reutilizar alguna. El identificador o nombre debería indicar su significado, suficiente para comprender su significado, pero corto para que sea fácil de escribir.

- Otras reglas de tipo general:

    - No comentar lo obvio.
    - Claro y sencillo es siempre mejor a complejo y maravilloso.
    - Lo ideal es colocar una única sentencia en cada línea, así se facilita la búsqueda de errores.
    - Hacer la interfaz de usuario simple y consistente, teniendo en cuenta que es el usuario quien ejecutará el programa y no el programador.
    - Suministre toda la ayuda posible. Si es posible identificar todos los mensajes de error documéntelos con su posible solución.

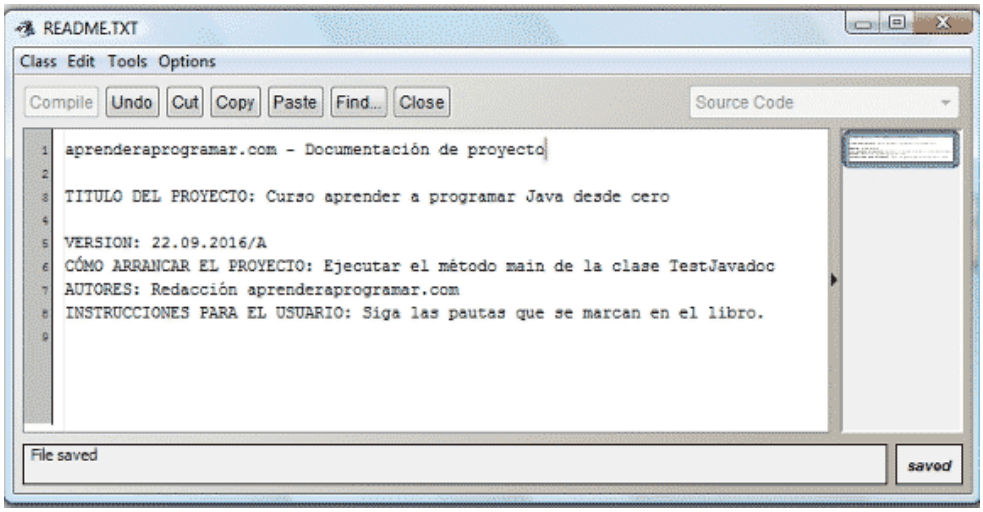
Documentar proyectos Java con Javadoc

Como hemos expresado anteriormente, una parte fundamental de la documentación es pensar en sus futuros mantenimientos, por eso al programar por ejemplo una clase se debe generar documentación suficientemente detallada de la clase como para que otros programadores puedan usarla.

Para la generación de documentos, Oracle provee una utilidad llamada Javadoc que genera APIs en formato HTML a partir del código fuente Java. Javadoc se convierte en un estándar que documenta clases de Java automáticamente.

Para que esa documentación pueda realizarse deben seguir unas normas como por ejemplo incluir el nombre de la clase, descripción general, número de versión, nombre de autores, además de documentación de cada método, en especial los públicos, con información como nombre del método, tipo de retorno, nombres y tipos de parámetros, descripción general, descripción de parámetros y descripción del valor que devuelve.

Adicional a la documentación de las clases, el proyecto debe tener un archivo Leeme o Readme, en donde se debe incluir información como el título del proyecto, descripción, versión, cómo iniciar el proyecto, autores e instrucciones para los usuarios.



Otras normas a seguir son:

- La documentación para javadoc se debe incluir entre símbolos de comentario que empiezan con una barra y doble asterisco, y terminan con un asterisco y barra simple.
- La ubicación le dice a javadoc qué representa el comentario, entonces si se incluye justo antes de la declaración de clase se tomará como un comentario de clase; si se coloca justo antes de la signatura de un método, se considera un comentario de ese método.
- Javadoc usa palabras reservadas (tags) precedidas por el carácter "@", dentro de los símbolos de comentario javadoc. Si no hay líneas que comiencen con @ no se toma el comentario para la documentación de la clase. Algunas de esas palabras reservadas se muestran en la siguiente imagen:

TAG	DESCRIPCIÓN	COMPRENDE
@author	Nombre del desarrollador.	Nombre autor o autores
@deprecated	Indica que el método o clase es obsoleto (propio de versiones anteriores) y que no se recomienda su uso.	Descripción
@param	Definición de un parámetro de un método, es requerido para todos los parámetros del método.	Nombre de parámetro y descripción
@return	Informa de lo que devuelve el método, no se aplica en constructores o métodos "void".	Descripción del valor de retorno
@see	Asocia con otro método o clase.	Referencia cruzada referencia (#método(); clase#método(); paquete.clase; paquete.clase#método()).
@version	Versión del método o clase.	Versión

- Los tags @author y @version son utilizados para documentar clases e interfaces y no son válidos en cabecera de constructores ni métodos.
- @param documenta constructores y métodos.
- @return solo es usada en métodos de tipo función.

El API de Java es muy útil pues está bien documentado y esto en relación al mantenimiento, ampliación y conexión del programa es importante, por eso la documentación de javadoc es necesaria en un proyecto profesional de Java. La documentación se puede complementar con todos los comentarios que se requieran si con esto se interpretará bien el código, solo se debe tener en cuenta que aquellos comentarios que no sigan las normas de javadoc no aparecerán en la documentación, aunque pueden ser útiles al consultar el código en un futuro.