

Presentación de

REPORTE DE BUENAS PRÁCTICAS Y DOCUMENTACIÓN DE CÓDIGO

ALFONSO CRUZ MENESES

OBJETIVO DEL REPORTE

Conocer las caracteristicas y como se realizan las buenas practicas de documentacion para mejorar la experiencia en cuanto a la creacion de codigo

BUENAS PRÁCTICAS DE CODIFICACIÓN

Nombres de variables

- Poner nombres adecuados a las variables es uno de los pilares del Clean Code. Como dice el autor Robert C. Martin: "El código se lee mucho más de lo que se escribe". Un buen nombre debe explicar por sí mismo por qué existe, qué hace y cómo se usa.

INTENCIONALIDAD Y CLARIDAD

El nombre debe ser lo suficientemente descriptivo para que no necesites un comentario al lado. Evita nombres genéricos como data, info o temp.

Ejemplo:

- Mal: var d // días transcurridos
- Bien: var diasTranscurridos;



EVITAR LA DESINFORMACIÓN

No incluyas el tipo de dato en el nombre ni uses nombres que puedan confundir.

Ejemplo:

- Mal: listadoArray,
usuarioString.
- Bien: usuarios, nombreUsuario.



USAR NOMBRES PRONUNCIABLES Y BUSCARLES

Si no puedes pronunciar el nombre de una variable, no podrás discutir el código con un colega. Además, las variables de una sola letra (como e) son imposibles de encontrar en archivos grandes mediante una búsqueda (Ctrl+F)

Ejemplo:

- Mal: const mod_ymd_hms = "2023-10-25";
- Bien: const fechaModificacionActual = "2023-10-25"



CUÁNDO COMENTAR:

Los comentarios deben aportar un valor que el código por sí solo no puede expresar. Úsalos en los siguientes casos:

Explicación de la Intención: Cuando hay una decisión de diseño que no es obvia.

- Ejemplo: "Se utiliza este algoritmo de ordenamiento específico porque el dataset siempre viene casi ordenado"



Estructura del código

- La estructura del código es lo que separa a un programador aficionado de un profesional. Un código bien estructurado es como un libro bien organizado: tiene capítulos claros, un índice lógico y es fácil de navegar. Aquí profundizamos en los tres pilares que mencionas para que tus entregas en esta unidad sean impecables.

Evitar Duplicidad
(Principio DRY) El principio DRY (Don't Repeat Yourself) establece que cada pieza de conocimiento o lógica debe tener una representación única dentro del sistema.

- Solución: Si ves que estás copiando y pegando código, detente. Crea una función o una clase que centralice esa lógica.

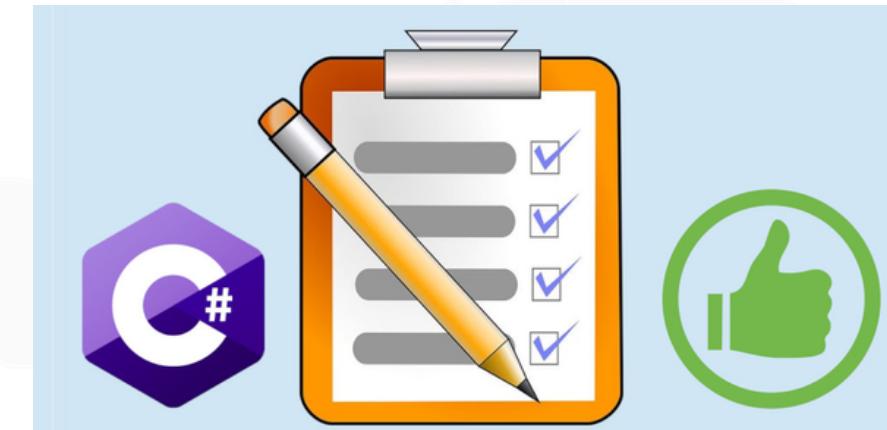
EVITAR LA DUPLICIDAD

- ¿Por qué evitarla? Si tienes la misma lógica de "cálculo de impuestos" en cinco lugares diferentes y la ley cambia, tendrás que buscar y editar los cinco lugares, aumentando el riesgo de errores.

DOCUMENTACIÓN DEL CÓDIGO

Estándares

JSDoc / Docstrings Estructurados Este estándar permite que el código sea "autodocumentado". No es solo texto para humanos; herramientas como VS Code lo usan para mostrarte ayuda mientras escribes, y herramientas como Doxygen o Swagger lo usan para crear manuales web automáticos



Elementos recomendados

Descripción N/A Una frase corta que explique qué hace el componente.
Parámetros @param Define el nombre, tipo de dato y descripción de las entradas.
Retorno @returns Indica qué devuelve la función y de qué tipo es. Excepciones
@throws Advierte sobre posibles errores que la función puede lanzar.

EL ARCHIVO README



Es la "tarjeta de presentación" de tu proyecto. Se escribe en formato Markdown y es el primer archivo que lee cualquier persona (o profesor) al abrir tu repositorio o carpeta.

Un buen README debe incluir:

- Título y Descripción: ¿Qué hace el proyecto?
- Instalación: ¿Cómo preparo el entorno?
- Uso: Ejemplos rápidos de ejecución.
- Tecnologías: Lista de lenguajes y librerías usadas.
- Créditos: Quién lo desarrolló

GENERADORES DE DOCUMENTACIÓN

Son herramientas que escanean tus comentarios (usando el estándar que elegimos, como JSDoc) y crean automáticamente un sitio web o un PDF profesional con toda la información técnica.

JSDoc / TypeDoc: Para JavaScript y TypeScript.

Sphinx: El estándar de oro para Python.

Swagger (OpenAPI): Especializado en documentar APIs y sus "endpoints".

Doxygen: El más potente para C++, Java y proyectos multi-lenguaje.



EXTENSIones DE VS CODE (TUS MEJORES ALIADOS)

No tienes que memorizar todas las reglas si usas las herramientas adecuadas en tu editor:

Extensión Función Prettier Formatea tu código automáticamente (indentación, comillas, espacios).

ESLint / Pylint Te avisa en tiempo real si estás rompiendo una buena práctica. Document This Genera automáticamente la estructura de JSDoc al presionar una tecla.

Better Comments Te permite categorizar comentarios por colores (alertas, tareas, dudas).

Markdown All in One Facilita la creación del archivo README con previsualización en vivo.

RECOMENDACIONES FINALES

**Para asegurar la máxima calidad, yo sugiero adoptar
estos hábitos desde el primer día:**

**Prioriza la Legibilidad sobre la Brevedad: Es mejor
una función de 10 líneas que sea fácil de entender,
que una de 3 líneas que parezca un jeroglífico. El
código se lee mucho más de lo que se escribe.**



GRACIAS

Por la atención