Buenas prácticas de desarrollo de software: Principios SOLID

Esta sección presenta una introducción a los 5 principios SOLID los cuales ayudan a desarrollar sotfware de calidad.

Cuando se trata del diseño y desarrollo de aplicaciones, los 'Principios SOLID' son un conjunto de conceptos esenciales que debes tener en tu repertorio como fundamentos clave en la arquitectura y creación de software.

SOLID es un acrónimo creado por Michael Feathers, que se basa en los principios de la programación orientada a objetos compilados por Robert C. Martin en su artículo de 2000, titulado 'Design Principles and Design Patterns'. Estos principios sientan las bases para el desarrollo de software de alta calidad y mantenible en el mundo de la programación orientada a objetos[1)].

Los 5 principios SOLID de diseño de aplicaciones de software son:

- S – Single Responsibility Principle (SRP)
- O – Open/Closed Principle (OCP)
- L – Liskov Substitution Principle (LSP)
- I – Interface Segregation Principle (ISP)
- D – Dependency Inversion Principle (DIP)

¿Que es Clean Code?

El código limpio no se basa en reglas rígidas, sino en una serie de principios que promueven la creación de un código intuitivo y fácil de modificar. En este contexto, la intuición implica que cualquier profesional de desarrollo pueda comprenderlo de inmediato. Un código que es fácilmente adaptable presenta las siguientes características:

- El flujo de ejecución del programa sigue una lógica clara y tiene una estructura sencilla.
- Las relaciones entre las distintas partes del código son transparentes.
- La función de cada clase, función, método y variable es evidente a simple vista.

Un código se considera fácil de modificar cuando es flexible y extensible, lo que facilita la corrección de posibles errores. Por estas razones, el código limpio resulta sencillo de mantener y exhibe las siguientes cualidades:

- Las clases y los métodos son concisos y, siempre que sea posible, tienen una única función bien definida.
- Las clases y los métodos son predecibles, operan de acuerdo a las expectativas y se acceden a través de API (interfaces) claramente documentadas.
- El código ha sido sometido a pruebas unitarias.

Las ventajas de este enfoque de programación son evidentes: el código limpio se vuelve independiente del desarrollador que lo creó. En esencia, cualquier programador puede trabajar con él, lo que evita los problemas asociados con el código heredado. Además, el mantenimiento del software se simplifica, ya que los errores son más fáciles de identificar y corregir[2)].

7 Reglas principales del Clean Code

1. La importancia de los nombres

La elección de nombres desempeña un papel fundamental en la comprensión de un código, ya sea para variables, funciones, parámetros, clases o métodos. Al seleccionar un nombre, es esencial considerar dos aspectos clave:

Debe ser preciso y expresar la idea central de manera directa. No temas usar nombres largos si ello es necesario para representar con claridad la función o propósito.

2. Regla del boy scout

Esta regla se basa en la idea de que, al salir de un área de acampada, debes dejarla más ordenada de lo que la encontraste. En el ámbito de la programación, esto se traduce en dejar el código más limpio de lo que estaba antes de editarlo.

3. Sé el autor auténtico del código

El código es una narrativa, y los programadores son los autores de esta historia. Para estructurar un código limpio, es fundamental crear funciones simples, claras y concisas. Dos reglas orientadoras son:

- Mantén las funciones pequeñas.
- Y, si es posible, aún más pequeñas.

Es importante no confundir "**nombre**" con "**función**". Como se mencionó en el primer principio, los nombres largos no son un problema, pero las funciones deben mantenerse breves.

4. DRY (No te repitas)

Este principio, acuñado en el libro "***The Pragmatic Programmer***," se aplica a diversas áreas de desarrollo, como:

- Bases de datos
- Pruebas
- Documentación
- Codificación

DRY defiende que cada elemento del conocimiento del sistema debe ser único y exento de ambigüedades, evitando así la duplicación de funcionalidades.

5. Comentar con moderación

Los comentarios en el código deben ser utilizados con moderación y solo cuando sean realmente

necesarios. Según la perspectiva de **Uncle Bob**, los comentarios pueden inducir a error, ya que suelen quedar obsoletos al modificarse el código. Por lo tanto, si se opta por comentar, debe ser de manera esencial y revisada conjuntamente con la versión del código.

6. Manejo de errores

El autor **Michael Feathers** destacó la importancia de tratar adecuadamente las excepciones en el desarrollo web. Los programadores son responsables de garantizar que el código siga funcionando incluso cuando surgen problemas. Tratar las excepciones de manera correcta es un aspecto clave en este proceso.

7. Pruebas limpias

La realización de pruebas es una etapa crucial en la programación, y solo un código que ha pasado pruebas limpias puede considerarse verdaderamente limpio. Para ello, se deben cumplir ciertas reglas:

- **Rápido**: Las pruebas deben ejecutarse rápidamente y en cualquier momento.
- **Independiente**: Las pruebas deben ser independientes para evitar efectos en cascada en caso de fallo.
- **Repetible**: Deben ser repetibles en diferentes entornos.
- **Autovalidación**: Las pruebas bien escritas devuelven respuestas claras (verdadero o falso) para evitar subjetividades en los errores.
- **Puntual**: Las pruebas deben ser escritas antes del código mismo, siguiendo estrictamente el criterio de puntualidad.

El **Clean Code** es un concepto arraigado que resuelve de manera eficiente uno de los principales desafíos que enfrentan muchos proyectos de desarrollo de sistemas: el mantenimiento[3].

←Volver atras

1) https://profile.es/blog/principios-solid-desarrollo-software-calidad/

2) https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/clean-code-que-es-el-codigo-limpio/

3) https://www.hostgator.mx/blog/clean-code-codigo-limpio/