Nombre: Ángel David Gómez Pastrana

Tarea 1 - Principios Solid

> ¿Que Son?

> Los principios SOLID son 5 recomendaciones que sirven para poder hacer un código limpio, mas legible, mantenible y que sirven para aumentar la escalabilidad y facilitar la refactorización del código

> ¿Cuales Son?

> 1) Single Responsability: Cada parte del código debe tener una única función o razón de estar.

> ¿Que genera tenerlo?: Facilidad de testing, facilidad de entender, cambiar, usar y y reutilizar el código.

> ¿Como detectar que no se cumple?:

- Al requerir hacer cambios pequeños usualmente necesitamos hacer una refactorización de algún modulo entero.
- La función, componente, clase, etc. tiene muchos objetivos.
- Para cambiar alguna cosa de alguna capa, necesito modificar una segunda capa.
- Para testear se requiere más de un mock.

> 2) Open/closed: El software debe estar abierto para extensión y cerrado para modificación. (Se pueden agregar funcionalidades sin editar código funcional).

> ¿Que genera tenerlo?: Facilidad de escalabilidad del código

> ¿Como detectar que no se cumple?:

- Al agregar nuevas funcionalidades alguna parte del codigo ya funcional debe ser refactorizada o modificada

- Agregar nuevas validaciones (switch, if).

> 3) Liskov's Substitution: Las subclases (o implementaciones) deben poder reemplazar a sus clases bases (o a sus interfaces) sin alterar el comportamiento esperado

> ¿Que genera tenerlo?: Menor probabilidad de errores, mayor seguridad en el manejo de los tipos de datos.

> ¿Como detectar que no se cumple?:

- No hay validación de tipos de datos o propiedades que requiere un objeto.

- Errores de tipos de datos.

- Al sustituir una implementación se rompe el código.

> 4) Interface Segregation: Las piezas no deben obligarse a depender de interfaces que no usan. (Varias interfaces pequeñas es mejor que una grande con opcional)

> ¿Que genera tenerlo?: Liberación de dependencias inutiles, ayuda al principio de responsabilidad unica, por ende, mejora la legibilidad, uso y reuso del código.

> ¿Como detectar que no se cumple?:

- Se usa muchas cosas opcionales. (Parametros, variables, etc.)
- Las implementaciones no usan las interfaces completas.
- Muchos NULL o Undefined.

> 5) Dependency Inversion: Depende de las abstracciones, no de las implementaciones concretas.

> ¿Que genera tenerlo?: Facilidad de modificaciones del código y mayor escalabilidad.

> ¿Como detectar que no se cumple?:

- Hace una instancia directa de una implementación.
- Necesitamos reconfigurar el modulo para probar.
- Importar y crear dependencias en vez de crear y recibirlas

Nota: A veces no se tienen estos principios (Ej: Código Legacy.), hacer la regla del Boy Scout, dejar el código mejor que antes.