**INTRODUCCION**

La presente monografía se refiere al tema de las buenas prácticas de la programación, como llevar un mejor manejo y desarrollo de nuestros proyectos y porque estas son vitales para el desempeño de todo PROGRAMADOR,

Pero ¿Qué son las buenas prácticas de la programación? ¿Cuáles son sus beneficios? Las buenas prácticas en programación son el conjunto de técnicas, principios metodologías que debemos implementar en nuestro software para que nuestros requerimientos no funcionales sean cada vez más óptimos. Los beneficios que estos aportan a nuestro código son, un código limpio, reutilizable, escalable y con mayor rango de cambios. También facilita la tarea a la hora de trabajar en forma colectiva con otro desarrollador. Nos ayudara a reducir o eliminar errores de compatibilidad entre las diferentes etapas de desarrollo y diseño, con estas practicas estaremos eliminando de raíz la mayoría de problemas de desarrollo de software

**DESARROLLO**

1. **Principios de Buena Programación**

* DRY – Don’t repeat yourself.
* Abstraction Principe.
* KISS – Keep it simple stupid
* Avoid Creating a YANGNI – You aren’t going to need it
* Do the simples thing that could possibly work
* Open/Close P

1. **DRY**

**Don’t repeat yorselft**

No repitas lo mismo evitar la repetición de código: funciones métodos clases, etc

No repetir librerías

No repetir documentación

Mejor Reutiliza

1. **ABSTRAC PRINCIPE**

Cada pieza de funcionalidad significativa en un programa debe ser implementada en un solo lugar del código fuente

1. **KISS**

**Keep it simple , stupid!**

Manténgalo simplemente estupidito La mejor solución a un problema es la mas simple

Menos código 🡪 menos bugs, Comentar código, Hacer test de usuarios

1. **YAGNI**

**You aren’t going to need it**

Evitar algo que no vamos a necesitar, es común tratar de ver el futuro y comenzar a crear abstracciones que todavía no estamos usando

Se pierde tiempo al añadir probar y documentar la característica no necesaria

Si la nueva funcionalidad no esta bien definida puede que no funcione correctamente, aunque con el tiempo se necesaria

El software se vuelve más grande y más complicado innecesariamente

1. **OPEN/CLOSE PRINCIPLE**

**Principio abierto/cerrado Clases, módulos funciones, etc. deben de estar abiertas a la extensión Escribir clases , no para que otros las modifique sino para que se usen y extiendan, Los beneficios del principio son:**

* **Robustez**: no se modifican las clases ya probadas
* **Flexibilidad:**  Se facilita la implementación de nuevos requisitos o cambios en ellos
* **Facilita las pruebas:** menos propenso a errores

1. **WRITE CODE FOR THE MAINTAINER**

Escribe código Pensando en el que va a mantenerlo

1. **PRINCIPLE OF LEAST ASTONISHMENT**

Principio del Menor Asombro, El nombre de cada función debería corresponder con lo que hace.

1. **SINGLE RESPONSIBILITY**

Principio de responsabilidad única, un componente de código debe ejecutar una única y bien definida tarea

1. **MINIMIZE COUPLING**

Minimizar el acoplamiento, cada componente debe minimizar las dependencias de otros componentes

1. **MAXIMIZE COHESION**

Evitar implementar En un componente dos funcionalidades que no estén relacionadas, cumpliendo tareas que no tienen relación

1. **HIDE IMPLEMENTATION DETAILS**

Ocultar detalles de implementación, Usar correctamente modificadores de acceso: publicx, private y protected

1. **LAW OF DEMETER**

Evitar la optimización prematura, evitar pensar optimizar código, si apenas lo estamos armando ir armando el código de tal forma que podamos refactorizarlo

1. **Consejos generales**

* Seguir los estándares
* Seguir los patrones de diseño
* Reutilizar código
* Usar librerías “reconocidas”, documentadas
* Test unitarios
* Conocer bien nuestro entorno de desarrollo shortcuts, plugins, etc
* Refactorización alterar su estructura interna sin cambiar su comportamiento
* Define una sintaxis de programación común para tu equipo de trabajo
* Define una estructura de directorios basada en la separación de conceptos
* Usar un sistema de control de versiones para compartir el trabajo con el equipo