Desarrolle una investigación sobre dos temas, el primero es la calidad de programación, ¿Qué es? ¿Por qué es importante? y buenas prácticas para implementarlas en nuestros códigos. El segundo tema puede ser elegido de la siguiente lista:

* Abstracción procedimental, técnicas y buenas prácticas.
* Programación orientada a objetos. Historia, buenas prácticas, constructores, concepto de público y privado, Static, Constant.
* Sobre carga de métodos y herencia.

**La calidad de programación**, tambien conocida como”Calidad de software”, se basa en un conjunto de actividades de evaluación de las distintas etapas del proceso de desarrollo para garantizar que el resultado final sea óptimo. Es un campo de studio y práctiva que describe los atributos deseables de los productos de software, de acuerdo a su eficacia, eficiencia, satisfacción, seguridad y usabilidad.

Asegurar la calidad implica varias actividades como:

Certificación del software: Al documentar y determinar que el software se construyó con dentro de los lineamientos establecidos por entes auditores para los procesos, que estos son los correctos y que el producto se encuentra en función de la normalización. Algunos de estos certificados son ISO 9000 **BUSCAR QUE HACE**, CMMI **BUSCAR QUE HACE**, ISO?IEC 25010:2011, entre otros

Medición del software: se determinan los atributos del mismo, dividiéndolos hasta llegar a los más simples de medir estadísticamente. Esto se logra comenzando el análisis desde los atributos más generals hasta los más específicos y concretos, que son los que se miden. Al valor de la medición se le denomina **métrica.**

Las métricas pueden ser de diferentes tipos:

Cantidad de errores en un lapso determinado.

Fallo en el código del sistema.

Fallo en el diseño del sistema.

Estimación del costo.

Estimación de los esfuerzos y recursos requeridos.

Pruebas funcionales:

Pruebas unitarias: Conocidos como Unit Tests. Permiten detector fallos en el código, además de ser una herramienta de diseño, pues requieren que el código esté escrito de manera modular, hacienda que cada modulo manipule la menor cantidad de datos posible y que estos estén disponibles para diferentes clases (desde el punto de vista de Programación Orienta a Objetos, u “OOP” por sus iniciales en inglés).

Pruebas de integración: Comprueban el funcionamiento del sistema, uniendo sus partes y lograr que funcionen para lo que fueron diseñadas, sin errores. Permiten comprobar el comportamiento de las partes del sistema, lo que sale del enfoque de las pruebas unitarias, pues tienen acceso a bases de datos y peticiones de red, entre otros.

Pruebas funcionales: Diseñados por tartar de probar el sistema, tal y como el usuario lo haría. Su desarrollo requiere más tiempo y recursos, lo cual es una buena inversión pues protegen contra interfaces fallidas.

El enfoque en manejo de defectos de software se basa en determinar y contar los defectos, que se categorizan por su severidad, y se deben tomar en cuenta en el planeamiento. De acuerdo a su desarrollo, se pueden utilizar herramientas que permiten determinar errores en matrices, que se controlan mediante cuadros y mediciones, logrando un desarrollo de la capacidad de proceso.

La calidad del software se mide de acuerdo a 8 características básicas:

Idoneidad funcional.

Confiabilidad.

Operabilidad.

Eficiencia de ejecución.

Seguridad.

Compatibilidad.

Mantenimiento.

Transferabilidad.

Buenas prácticas de calidad de software a nivel de diseño y codificación:

Utilizar la modularidad hasta donde sea posible. Así, si algo llegara a fallar en el future o se requiriera escalar la funcionalidad del sistema, se podría hacer con menor riesgo de afectarlo todo.

Evitar duplicar código.

Utilizar los tipos de datos que mejor se adapten a las características de los datos.

Un switch con muchas cláusulas puede crear confusion.

Excesivo uso de ifs, fors o whiles anidados pueden causar problemas en el código, además que su mantenimiento será dificil.

Se recomienda utilizar nombres descriptivos para las clases, métodos y variables. Esto permite que la persona encargada de su mantenimiento lo pueda hacer sin comprometer todo el sistema.

El usuario debe ser capaz de interactuar con el sistema mediante la interfaz específica (front end). No se debe hacer análisis o procesamiento de datos en esta interfaz, sino en el backend, exclusivo para desarrolladores.

La información confidencial (cualquiera que pueda afectar al usuario o a la empresa), debe encriptarse en el backend. Si fuera requerido guardarla (como en el caso de contraseñas, números de identificación, cuentas bancarias y similares), se prefiere que, además de encriptarla, se almacene en una base de datos que cuente con niveles de acceso y su encriptación propia.

<https://es.wikipedia.org/wiki/Calidad_de_software>

<https://asq.org/quality-resources/software-quality>

<https://www.javiergarzas.com/calidad-software>