**Requisitos**: se refiere a la fase donde se identifican y documentan las necesidades y expectativas del usuario final y otras partes interesadas para un sistema de software. Además que estos requisitos describen qué debe de hacer el sistema y las restricciones bajo cuales debe de operar.

Por otro lado existen dos tipos de requisitos los funcionales y no funcionales. Los funcionales son aquellos que especifican las funciones que el sistema debe realizar, como tareas, servicios o comportamientos. Y los no funcionales en cambio son los que definen criterios que pueden usarse para juzgar la operación de un sistema, en lugar de funciones especificas, esto incluye apectos como rendimiento, seguridad, fiabilidad y usabilidad.

**Diseño**: es una fase que sigue a la definición de requisitos y se centra en como se va a construir el sistema de software para cumplir con esos requisitos.

Esto se divide en dos niveles:

Alto nivel (nivel arquitectónico): Este nivel define la estructura general del sistema, incluyendo la organización de los principales componentes o modulos, las interacciones entre ellos, y la arquitectura del sistema.

Bajo nivel (diseño detallado): Este nivel especifica el diseño interno de los modulos individuales o componentes del sistema. Esto incluye detalles sobre estructuras de datos, algoritmos y flujos de control.

**Implementación**: Esta fase es donde se lleva a acabo el desarrollo real del sistema de software, basada en el diseño previamente definido. Ademas en esta fase se incluyen algunas actividades como lo podrían ser codificación, integración y documentación.

Codificacion: Los desarrolladores traducen las especificaciones de diseño en código fuente. Este código debe ser claro, eficiente y fácil de mantener. Es común seguir buenas prácticas de programación y utilizar herramientas como control de versiones (por ejemplo, Git) para gestionar el código.

Integración: A medida que se desarrollan los diferentes módulos y componentes, se integran para formar un sistema funcional. La integración puede hacerse de manera incremental, donde se añaden y prueban módulos uno a uno, o como una integración más grande al final del proceso.

Documentación: Se documenta el código y el proceso de implementación, incluyendo comentarios en el código, manuales técnicos y guías de usuario, para facilitar la comprensión y mantenimiento futuros.

**Pruebas**: esta fase es crucial ya que se garantiza que el sistema desarrollado funcione según lo esperado y cumpla con los requisitos especificados. Ademas de que durante esta fase se evalua el software para identificar y corregir errores, asegurando que el producto final sea de alta calidad y confiable.

Estas pruebas se dividen en varios tipos, esto dependiendo el nivel y el enfoque:

Pruebas unitarias, pruebas de integración, pruebas de sistema, de aceptación y de rendimiento.

Además de que esta fase nos ayuda a identificar defectos o problemas en el software, que luego pueden ser corregidos antes del lanzamiento final.

 **Pruebas unitarias:** Se centran en probar componentes o módulos individuales del software en aislamiento para verificar que cada uno funcione correctamente por sí mismo. Estas pruebas son generalmente realizadas por los desarrolladores.

 **Pruebas de integración:** Verifican que los diferentes módulos o componentes del sistema interactúen correctamente entre sí una vez integrados. El objetivo es identificar problemas en las interfaces entre los módulos.

 **Pruebas de sistema:** Evaluan el sistema completo como un todo para asegurarse de que cumple con los requisitos funcionales y no funcionales. Estas pruebas simulan el entorno en el que se espera que el software opere.

 **Pruebas de aceptación:** Son realizadas por el usuario final o el cliente para confirmar que el sistema cumple con sus expectativas y necesidades. Este tipo de prueba es el último paso antes de que el software se entregue oficialmente para su uso.

 **Pruebas de rendimiento:** Miden cómo se comporta el software bajo diferentes condiciones de carga, como la cantidad de usuarios simultáneos o la velocidad de procesamiento. Esto es crucial para garantizar que el sistema sea eficiente y escalable.

**Mantenimiento**: es el proceso continuo de actualización, mejora y corrección del software después de haber sido desplegado y puesto en producción. Esta fase es crucial para asegurar que el software siga siendo útil y eficiente a lo largo del tiempo. Ademas de que existen cuatro tipos de mantenimientos, esto según la naturaleza del trabajo que realiza: mantenimiento correctivo, adaptativo, perfectivo y preventivo.

 **Mantenimiento correctivo:** Implica la corrección de errores o defectos que no fueron detectados durante las pruebas, pero que han surgido durante el uso del software. Estos pueden ser fallos en el funcionamiento o en el rendimiento del sistema.

 **Mantenimiento adaptativo:** Se refiere a las modificaciones realizadas al software para que funcione en un entorno nuevo o cambiado. Esto puede incluir la actualización del software para que sea compatible con nuevas versiones de sistemas operativos, hardware, o software de terceros.

 **Mantenimiento perfectivo:** Consiste en la mejora o optimización del software para hacerlo más eficiente, mejorar su rendimiento o añadir nuevas características que no fueron originalmente especificadas. Esto también puede incluir mejoras en la usabilidad y la experiencia del usuario.

 **Mantenimiento preventivo:** Se enfoca en la detección y corrección de posibles problemas antes de que ocurran, con el objetivo de evitar fallos futuros. Este tipo de mantenimiento puede incluir la reestructuración del código, la limpieza de redundancias o la actualización de la documentación.