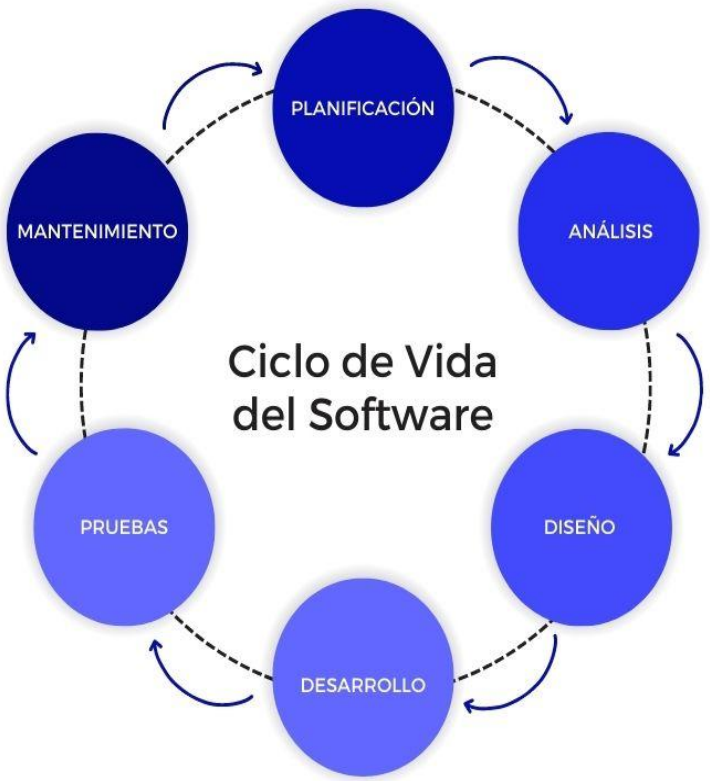
Tarea 2

Ciclo de vida del software (Relación 1)

1.- Define "Ciclo de vida del software".

El **ciclo de desarrollo** del software nos brinda la capacidad de identificar, gestionar y planificar el uso de recursos con el fin de alcanzar un objetivo predefinido. El número de etapas que se encuentran en cada proyecto puede variar según las necesidades de la empresa en cuestión.

El conocimiento del ciclo de vida de un proyecto nos permite una mejor supervisión de sus etapas y evita situaciones innecesarias. A pesar de que las etapas son consistentes en cualquier proyecto, cada uno requiere su propio conjunto único de procesos.



2.- Nombra las fases principales del desarrollo de software y explica brevemente qué se hace en cada una de ellas.

Análisis: En la etapa de análisis en un proyecto, se define qué funciones ejecutará el software y cuáles son sus características específicas. Este aspecto es esencial para optimizar la gestión de los recursos financieros y determinar el alcance del proyecto.

La asignación de costos es una de las labores más desafiantes en el desarrollo de software, ya que debe realizarse al principio, cuando tenemos menos información sobre el proyecto y existe un margen de error mayor. Afortunadamente, la experiencia en proyectos similares y la descomposición del proyecto en tareas individuales nos simplifican en gran medida la tarea de estimar los presupuestos de manera adecuada.

En esta fase se especifica los requisitos que debe cumplir el software a desarrollar y la especificación de requisitos debe:

- Ser completa y sin omisiones
- Ser concisa y sin trivialidades
- Evitar ambigüedades. Utilizar lenguaje formal.
- Evitar detalles de diseño o implementación
- Ser entendible por el cliente
- Separar requisitos funcionales y no funcionales
- Dividir y jerarquizar el modelo
- Fijar criterios de validación

Diseño: En esta etapa, se exploran diversas alternativas para la implementación del software a construir, al mismo tiempo que se determina su estructura general. El diseño representa una fase de alta complejidad y se lleva a cabo mediante un proceso iterativo.

Es factible que la primera solución propuesta no sea la más apropiada, por lo que en dicho caso, es necesario perfeccionarla. Sin embargo, existen valiosos catálogos de patrones de diseño que recopilan los errores cometidos por otros, lo que nos permite evitar caer en las mismas trampas.

Las actividades habituales son las siguientes:

- Diseño arquitectónico. Aplica tus conocimientos en programación para establecer una estructura inicial del software que permita la incorporación de detalles en etapas posteriores.
- Diseño detallado. Los aspectos técnicos y la realización del diseño conceptual se centran en la ejecución de ideas. El diseñador aborda desafíos relacionados con la materialización de conceptos.
- Diseño de datos. Consiste en descubrir y definir completamente los procesos y características de los datos de la aplicación. ¿Qué datos requiere la aplicación? ¿Cómo se accederán a estos datos?
- Diseño de interfaz de usuario. Implica identificar y detallar por completo los procedimientos y atributos de los datos utilizados en la aplicación. ¿Qué información necesita la aplicación? ¿Cuál será el método de acceso a estos datos?

Codificación: Una vez que hemos establecido la arquitectura de nuestro software, es hora de iniciar la fase de programación. La elección del lenguaje de programación apropiado para nuestro proyecto y contar con un equipo de programadores con experiencia resulta esencial. El seguimiento de las mejores prácticas de codificación garantizará que nuestro proyecto sea escalable de manera eficiente.

Al programar, hay que intentar que el código no sea indescifrable siguiendo distintas pautas como las siguientes:

- Evitar bloques de control no estructurados.
- Identificar correctamente las variables y su alcance.
- Elegir algoritmos y estructuras de datos adecuadas para el problema.
- Mantener la lógica de la aplicación lo más sencilla posible.
- Documentar y comentar adecuadamente el código de los programas.
- Facilitar la interpretación visual del código utilizando reglas de formato de código previamente consensuadas en el equipo de desarrollo.

También es importante considerar la obtención de los recursos necesarios para que el software opere correctamente, además de elaborar escenarios de prueba para verificar su funcionamiento a medida que se va programando.

Pruebas: La etapa de pruebas en el ciclo de vida del software tiene como objetivo identificar y rectificar los errores que pudieron haber ocurrido en las etapas anteriores, preferiblemente antes de que el usuario final los descubra. Se considera un éxito en las pruebas cuando se detecta algún tipo de error.

Esta fase implica la corrección, eliminación y mejora de posibles fallos que no se habían previsto en las etapas anteriores. Es esencial repetir esta etapa del ciclo de vida del software tantas veces como sea necesario, ya que la calidad y estabilidad final del software dependen en gran medida de esta fase. Esto implica someter el programa a diversas situaciones para garantizar su óptimo funcionamiento.

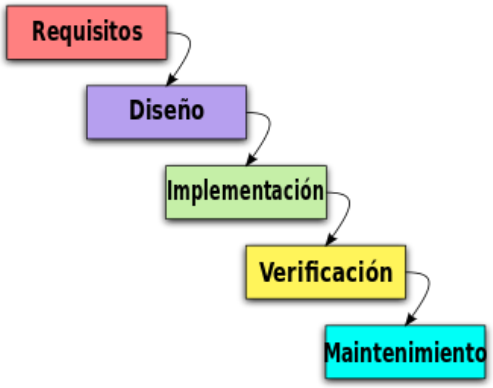
Mantenimiento: En la etapa de mantenimiento, el equipo se encarga, entre otras responsabilidades, de solucionar defectos, abordar las inquietudes de los usuarios y gestionar las modificaciones realizadas en el software. Además, monitorean el rendimiento integral del sistema, su seguridad y la experiencia del usuario, con el fin de descubrir nuevas oportunidades para mejorar el software existente.

Hay 3 tipos de mantenimiento:

- **Correctivo**: se refiere a un conjunto de actividades técnicas que tienen como objetivo solucionar problemas en los activos y dispositivos cada vez que surja la necesidad de repararlos o incluso reemplazarlos.
- **Perfectivo**: su propósito es adaptar el software mediante la incorporación de nuevas funcionalidades según sea requerido, y suprimiendo aquellas características que carecen de relevancia o resultan poco efectivas en el contexto del software en cuestión.
- **Evolutivo**: Se esfuerza por perfeccionar el sistema con el propósito de prevenir dificultades o inconvenientes en el futuro.
- **Adaptativo**: Implica realizar ajustes en el software sin perturbar su operatividad, con la finalidad de adecuarlo a las transformaciones en el entorno de la aplicación.
- **Preventivo**: Se enfoca en llevar a cabo una evaluación metódica y siguiendo ciertos estándares en dispositivos de diversas naturalezas, ya sean mecánicos, eléctricos, informáticos, entre otros, con el objetivo de prevenir fallos causados por el uso, desgaste o el paso del tiempo.

3.- Explica brevemente en qué consiste el modelo en cascada cuando hablamos de desarrollo de software.

El **modelo en cascada** dispone todas las etapas de manera secuencial, de manera que cada nueva fase depende de los resultados de la fase anterior. Desde un punto de vista conceptual, el diseño fluye de arriba hacia abajo, como una cascada. Aunque ha sido objeto de críticas por su rigidez, sigue siendo el enfoque más predominante en la actualidad.



4.- Ventajas e inconvenientes del modelo en cascada.

Ventajas.

- Administración estricta del proyecto: El modelo de cascada impone una estructura y secuencia clara en el desarrollo del software, lo que facilita una gestión rigurosa del proyecto.
- Resultados tangibles por fase: Proporciona resultados tangibles al final de cada fase, lo que permite una evaluación continua del progreso y la calidad del software.

Inconvenientes.

- Poco margen de cambio: Una vez que una fase se considera completa, hay limitado margen para realizar cambios significativos, ya que esto puede afectar al tiempo de entrega, costo y calidad del software.
- Adecuado para proyectos pequeños: Este enfoque es más adecuado para proyectos de desarrollo de software pequeños, donde los requisitos pueden definirse con precisión y las tareas se pueden gestionar fácilmente. En proyectos más grandes y complejos, la falta de flexibilidad puede ser problemática.

5.- ¿Qué se entiende por verificación? ¿Y por validación?

La **verificación** es un procedimiento sencillo que se lleva a cabo durante el proceso de desarrollo del software. Este proceso abarca actividades como reuniones, inspecciones, análisis exhaustivos, revisiones y otros métodos para examinar planos, códigos, documentos, especificaciones y requisitos.

También se describe como el proceso de evaluar la aplicación para determinar si cumple con los requisitos y puede satisfacer a los clientes o usuarios finales.

En consecuencia, el propósito principal de la verificación es asegurar la calidad de la aplicación de software, incluyendo aspectos como la arquitectura y el diseño, entre otros.

El proceso de verificación consta de tres fases:

- Verificación de los requisitos: Se trata de un procedimiento destinado a comprobar y corroborar que las solicitudes o requisitos están íntegros, exactos y detallados de manera precisa.
- Verificación del diseño: Se refiere a un procedimiento para verificar si la aplicación de software está en conformidad con las especificaciones de diseño mencionadas en el documento, a través de la realización de pruebas.
- Verificación del código: Se trata de un procedimiento para verificar que el código es preciso, coherente y abarca todas las partes necesarias.

La **validación** es un procedimiento empleado para valorar el software en concordancia con las necesidades del negocio o del cliente, ya sea durante el desarrollo del software o al concluirlo. Su finalidad es examinar la aplicación terminada para asegurarse de que cumple con las expectativas y los requisitos del cliente.

Se denomina como el método dinámico de confirmación del proyecto real en conjunto con las pruebas. La validación se enfoca en el resultado y no está relacionada con los procedimientos internos. Es un proceso independiente que comienza únicamente después de la etapa de verificación.

Los procesos de **validación** abarcan los siguientes pasos:

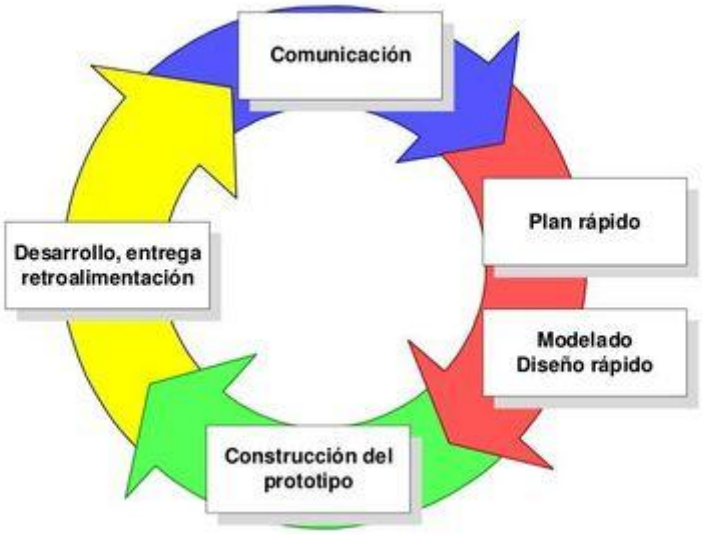
- Revisión del diseño: En esta fase, el equipo de pruebas de software define los requisitos establecidos por los clientes y elabora un plan de pruebas para verificar cada elemento del software antes de avanzar hacia la producción.
- Revisión de la instalación: A continuación, el equipo de pruebas de software procede a instalar la aplicación de software de acuerdo con el plan de pruebas establecido.
- Revisión operativa: Los evaluadores de software someten la aplicación a diversos escenarios de prueba para verificar su funcionamiento y asegurar su integridad.
- Revisión del rendimiento: Se lleva a cabo para confirmar que la aplicación de software puede operar de acuerdo a las necesidades de la empresa en situaciones reales.
- Revisión de la preparación para la producción: Una vez que se completan todas estas evaluaciones, el proceso de validación se considera concluido, y el producto se encuentra listo para su implementación en producción.

6.- Explica cómo funciona el modelo de desarrollo mediante creación de prototipos.

Un **modelo de desarrollo evolutivo**, también conocido como enfoque prototipo, se utiliza principalmente en el desarrollo de software para proporcionar a los usuarios una vista previa del programa o sistema en desarrollo. Este enfoque evolutivo implica la creación gradual y mejora de un prototipo hasta que se convierte en el producto final.

El modelo de prototipos se centra en la creación rápida de un diseño que represente las características clave del programa, lo que permite a los usuarios probarlo y brindar sus opiniones sobre aspectos como la usabilidad y el rendimiento. Se pueden realizar modificaciones en el prototipo según sea necesario, y es esencial registrar los resultados de las pruebas para orientar el

desarrollo del producto final. En resumen, el enfoque prototipo implica un desarrollo gradual basado en la retroalimentación de los usuarios para asegurar que el producto final cumpla con sus expectativas.



7.- Explica cómo funciona el modelo espiral cuando se aplica al desarrollo orientado a objetos.

El **modelo en espiral**, también conocido como desarrollo o modelo incremental, es una estrategia de desarrollo de software que se ha concebido en respuesta a las limitaciones del enfoque en cascada. En este modelo, el ciclo de vida del software se representa mediante espirales que se repiten hasta que se completa el producto final.

Una característica fundamental de este enfoque es la mitigación de riesgos en el desarrollo de software, lo que puede resultar en un aumento de los costos totales, un mayor esfuerzo y un retraso en la entrega. Estos riesgos se contrarrestan mediante un enfoque incremental, que implica la creación de prototipos y su posterior paso por las diferentes fases del desarrollo de software. El desarrollo en espiral es un enfoque versátil que puede combinarse con otros métodos de desarrollo, tanto tradicionales como ágiles, lo que lo convierte en un modelo de "segundo orden".

El modelo de desarrollo en espiral se caracteriza por ciclos o cuadrantes que incluyen los siguientes pasos:

- Objetivo y determinación alternativa: En esta etapa, se establecen los objetivos del proyecto en colaboración con el cliente. También se discuten posibles alternativas y se especifican las condiciones generales, como los sistemas operativos y lenguajes de programación.
- Análisis y evaluación de riesgos: Se identifican y evalúan los posibles riesgos asociados al proyecto. Además, se evalúan las alternativas disponibles. Los riesgos se registran, evalúan y se reducen mediante el uso de prototipos, simulaciones y herramientas de análisis de software.
- Desarrollo y prueba: Los prototipos se expanden y se les añaden funcionalidades. Se escribe el código real, se prueba y se migra a un entorno de prueba varias veces hasta que el software esté listo para ser implementado en un entorno productivo.
- Planificación del siguiente ciclo: Al final de cada etapa, se planifica el siguiente ciclo. Si surgen errores, se buscan soluciones, y si una alternativa resulta ser una mejor solución, se prefiere en el próximo ciclo.

La fuerza impulsora clave en este modelo es el análisis y la evaluación de riesgos. La identificación temprana y la gestión de riesgos son esenciales para el éxito del proyecto. El proyecto se considera exitoso cuando los riesgos han sido eliminados. El objetivo es producir un producto en constante mejora, donde el software se perfecciona continuamente. El modelo en espiral es incremental, pero no necesariamente repetitivo. Las repeticiones ocurren sólo cuando los riesgos o errores amenazan el proyecto, lo que se conoce como una iteración o repetición.