1. **En las fases del ciclo de vida del software existen dos fases con nombre similar: implantación e implementación. ¿Podrías decir de qué fases estamos hablando, qué fases tienen antes y después y contextualizarlas con un ejemplo (imagina que diseñas una app para iphone)?**

La **implementación** es la fase de Codificación, en la que se escribe el programa y todos sus componentes. Tiene lugar entre las fases de Diseño y de Prueba. La **implantación** es la fase de Explotación, en la que se entrega la solución final al cliente en el formato adecuado. Tiene lugar entre las fases de Documentación y la de Mantenimiento.

En el caso de estar diseñando una app para iphone, la fase de implementación consistiría en codificar el programa, en el lenguaje de programación elegido y basándonos en los requisitos del cliente. La fase de implantación sería cuando la app se entrega finalmente al cliente, en función de los requisitos y el formato acordado.

Fuente: apuntes ED.2. Ingeniaría del software.

2. **En el ciclo de vida incremental existe una fase llamada integración. ¿Podrías buscar en la Red y en los apuntes y decirnos, con tus propias palabras, en qué consiste?**

Consiste en unir (integrar) todos los incrementos aprobados y validados por la gestión del proyecto. Estos incrementos se componen de las tareas pendientes, que han sido llevadas a cabo y su resultado ha sido aprobado por los responsables del proyecto.

Fuente: https://blog.comparasoftware.com/modelo-incremental-fases/

3. **El inicio de las metodologías ágiles se data el 12 de febrero de 2001, cuando 17 expertos firman lo que se llamó Manifiesto Ágil.**
   a. **Primero: Lista 12 principios que contiene (MANIFIESTO ORIGINAL)**

   1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
   2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
   3. Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.

4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.

5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.

6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.

7. El software funcionando es la medida principal de progreso.

8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.

9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.

10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.

11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.

12. A intervalos regulares, el equipo reflexiona sobre cómo ser más eficaz y perfeccionar su comportamiento en consecuencia.

Fuente: https://sentrio.io/blog/valores-principios-agile-manifiesto-agil/

b. Segundo: Intenta comprimirlos en 5 (MANIFIESTO RESUMIDO)

1. Se aceptan cambios en los requisitos.
2. Entregas continuas de software funcional.
3. Simplicidad y revisión continua del trabajo y la eficiencia.
4. Colaboración estrecha y cara a cara con el cliente y todos los participantes involucrados.
5. Equipos motivados y auto-organizados para alcanzar la máxima eficiencia.

4. Enhorabuena, tu empresa ha firmado un contrato para realizar una aplicación de móvil por valor de 500.000 de euros, pero tu jefe sigue se empeña en usar el ciclo de vida en cascada. Ahora, en noviembre, cerrará los requisitos con el cliente y en julio tenéis prevista la entrega. Busca en la Red qué ocurrirá si el cliente cambia los requisitos iniciales en enero, es decir, si os escribe un email u os llama para deciros que hay ciertas funcionalidades que quiere modificarlas.

Si hay que cambiar los requisitos y hay que modificar el diseño, seguramente habrá que volver a empezar de cero. Esto supondrá un gran coste económico e importantes retrasos en la entrega del software final.

5. **Todas las fases del ciclo de vida del software son importantes pero queremos que valores y analices la importancia de eliminar cada una de ellas del ciclo de vida de tu proyecto. Imagina un escenario en el que tienes que desarrollar un proyecto software desde cero en un tiempo mínimo. El cliente asume que el producto no será "robusto" y tú te niegas a hacerlo pero tus jefes te presionan y accedes con la condición de que el cliente asuma ciertas consecuencias por escrito. Existe una etapa que no puedes eliminar bajo ningún concepto ¿Cuál es esa etapa? ¿Crees que hay más de una "intocable"? No hay una única respuesta si la argumentas correctamente. Repasa todas las etapas e indica en cada una de ellas las consecuencias de eliminarla de la planificación, es decir, qué le harías firmar al cliente si (para ahorrar tiempo) acordáis eliminar esa etapa.**

- Fase de **análisis**: esta etapa se puede pensar fácilmente que es suprimible si no tenemos tiempo. Sin embargo, para mi esta etapa es esencial, ya que si no entendemos bien el problema, no podremos darle solución y nunca podremos crear lo que el cliente nos está solicitando. Si hubiera que eliminar esta fase, habría que hacerle firmar al cliente un documento donde se especifique y acepte que se van a comenzar el resto de etapas con cero ERS y cero prototipos, por si más adelante cambia de opinión, o para que no pueda luego recriminarnos que la solución no es la que quería porque no hicimos un buen análisis.
- Fase de **diseño**: para mí la fase de diseño también es muy importante, para evitar futuros retrasos y tener que volver a empezar de cero. Podría ser suprimible en caso de necesidad, pero seguramente supondría muchos cambios luego en la fase de codificación, y retrasos en el proyecto, con un aumento del coste económico. En caso de suprimir esta fase, el cliente debería firmar un documento mostrando su conformidad con esta decisión, y debería estar abierto a posibles retrasos y aumentos del precio.
- Fase de **codificación**: esta fase no se puede suprimir en ningún caso. Si no codificamos, no tendremos ningún software. Se puede crear un software sin hacer un análisis y diseño

exhaustivos, pero nunca sin codificar las instrucciones que debe ejecutar nuestro software para que resuelva los problemas del cliente. Nunca habría que aceptar hacer un proyecto sin esta fase, ya que sería inviable.

- Fase de **pruebas**: esta fase podría suprimirse en caso de necesidad. No es recomendable, ya que no se detectarían posibles errores en el software desarrollado, pero se podrían ir buscando soluciones según se detectaran. También se incrementaría la duración del proyecto y el coste económico si luego hay que volver a hacer parte de la codificación para resolver problemas antes no detectados. Si se omite esta fase, el cliente debería firmar un documento en el que nos exima de cualquier responsabilidad si el software no funciona como él esperaba.
- Fase de **documentación**: para mí, esta sería la fase más fácilmente suprimible. Si el cliente quiere suprimir esta fase, debería dejar por escrito su conformidad con esta decisión, para que en un futuro no nos venga pidiendo explicaciones de cómo se hizo el software o cómo funciona. Sin embargo, suprimir esta fase implicaría que, en un futuro, no se tendrá ninguna información sobre cómo se desarrolló el software ni para que sirven ninguno de sus componentes. Esto complica su mantenimiento futuro, sobre todo si cambian los miembros del equipo.
- Fase de **explotación**: esta fase debería ser esencial, ya que el objetivo del proyecto que estamos realizando es entregar al cliente el software que nos ha pedido, con sus requisitos y el formato deseado. Podría ser que las necesidades del cliente cambiaran, o que se agotara su presupuesto, y en ese caso no se realizaría la fase de explotación. Se debería firmar un documento de cese de proyecto, aceptado por el cliente.
- Fase de **mantenimiento**: esta fase se podría suprimir. Sin embargo, en el momento que empiecen a haber fallos en el software y nadie se haga cargo de ellos, el software dejaría de funcionar correctamente y acabaría dejando de ser usado. También podría ocasionar otros problemas al cliente, si se producen errores en sus datos y éstos se modifican. Si se quiere prescindir de esta fase, el cliente debe firmar un documento donde quede constancia de esta decisión, con las posibles consecuencias que pueda tener para el cliente.