

Cuestionario sobre Requerimientos de Software

1. (Introducción)
¿Con qué inicia realmente la ingeniería de software?
a) Con la programación inmediata.
b) Con el diseño de interfaces.
c) Con el entendimiento de las necesidades del cliente.
d) Con la selección del lenguaje de programación.
2. (Introducción)
¿Cuál es el objetivo principal del tutorial mencionado?
a) Manual de usuario
b) Requerimientos funcionales y no funcionales
c) Diagramas de red
d) Contrato con el cliente
3. (Introducción)
¿Cuál de los siguientes elementos forma parte del documento de especificación de requerimientos?
a) Diseñar la interfaz del programa
b) Comprender lo que el cliente necesita
c) Crear la base de datos
d) Probar el software
4. (Definición de requerimientos)
¿Qué son los requerimientos de software?
a) Descripciones de las funciones, características y restricciones que un sistema debe cumplir.
b) Líneas de código que hacen funcionar el sistema.
c) Diagramas de flujo del programa.
d) Manuales de usuario.
5. (Klaus Pohl, 2010)
¿Con qué comparó Klaus Pohl los requerimientos de software?
a) Con una receta de cocina.
b) Con los planos de una construcción.
c) Con un manual de usuario.
d) Con un algoritmo.
6. (Tipos de requerimientos)
¿Cuántos tipos principales de requerimientos existen?
a) Tres.
b) Dos.
c) Cuatro.
d) Cinco.
7. (Requerimientos funcionales)
¿Qué describen los requerimientos funcionales?

- a) El rendimiento del sistema.
- b) Los colores y estilos de la interfaz.
- c) Las restricciones del hardware.
- d) Las acciones o servicios que el sistema debe realizar.**

8. (Ejemplo de requerimiento funcional)

¿Cuál de los siguientes es un ejemplo de requerimiento funcional?

- a) El sistema debe tener disponibilidad del 99.9%.
- b) La página debe cargar en menos de dos segundos.
- c) El usuario debe poder iniciar sesión con su correo y contraseña.**
- d) Las contraseñas deben cifrarse con SHA-256.

9. (Requerimientos no funcionales)

¿A qué se refieren los requerimientos no funcionales?

- a) A las cualidades o restricciones del sistema.**
- b) A las funciones principales del sistema.
- c) A los errores comunes de programación.
- d) A los costos del proyecto.

10. (Clasificación de requerimientos funcionales)

¿Qué tipo de requerimiento describe la rapidez con la que el sistema responde a una acción del usuario?

- a) Requerimiento funcional
- b) Requerimiento de interfaz
- c) Requerimiento de desempeño**
- d) Requerimiento de diseño

11. (Herramientas de modelado de requerimientos)

¿Qué herramienta ayuda a modelar visualmente los procesos del sistema y su relación con los usuarios?

- a) Lista de verificación
- b) Diagrama de casos de uso**
- c) Diagrama de flujo de datos
- d) Matriz de trazabilidad

12. (Validación de requerimientos)

¿Cuál es el principal objetivo de validar los requerimientos?

- a) Confirmar que los requerimientos reflejan las necesidades reales del usuario**
- b) Reducir el tiempo de desarrollo
- c) Comprobar la compatibilidad del hardware
- d) Calcular el costo del sistema

13. (Cualidades de un buen requerimiento)

¿Qué característica debe tener un buen requerimiento?

- a) Ser complejo
- b) Ser verificable y medible**
- c) Ser general
- d) Ser ambiguo

14. (Fases del ciclo de vida del software)
¿En qué fase del ciclo de vida del software se documentan y aprueban los requerimientos?
a) Implementación
b) Análisis
c) Pruebas
d) Mantenimiento
15. (Gestión y resolución de conflictos de requerimientos)
Cuando se detecta un conflicto entre dos requerimientos, ¿qué debe hacerse primero?
a) Ignorar el conflicto y avanzar
b) Aceptar ambos requerimientos
c) Analizar con el cliente cuál tiene mayor prioridad
d) Modificar uno de los requerimientos sin consultar
16. (Ejemplo de requerimiento no funcional)
¿Cuál de los siguientes es un ejemplo de requerimiento no funcional?
a) El usuario debe de poder modificar su perfil
b) La página principal debe de cargarse en menos de dos segundos.
c) El sistema debe permitir generar reportes
d) El administrador puede eliminar usuarios
17. (Elizabeth Hull, 2017)
Según Elizabeth Hull, ¿por qué es importante clasificar los requerimientos?
a) Para dividir el trabajo en partes iguales.
b) Para que todos comprendan las expectativas y criterios de calidad del proyecto.
c) Para ahorrar tiempo en la programación.
d) Para facilitar la instalación del software.
18. (James Crowder, 2022)
¿Qué señala James Crowder sobre los errores en esta etapa?
a) Que son los más costosos de corregir.
b) Que se pueden ignorar sin consecuencias.
c) Que se corrigen fácilmente al final.
d) Que no afectan al proceso posterior.
19. (Beneficios de una buena gestión)
¿Qué ocurre cuando los requerimientos están completos, claros y documentados?
a) Disminuye la comunicación del equipo.
b) Se reduce el tiempo de prueba.
c) El equipo trabaja con una visión común.
d) Aumenta la cantidad de código.
20. (Beneficios)
¿Cuál es uno de los beneficios principales de una correcta gestión de requerimientos?

- a) Reducir la cantidad de reportes.
- b) Aumentar los costos del sistema.
- c) Evitar las pruebas finales.
- d) Alinear usuarios, clientes y desarrolladores.**

21. (Beneficios)

¿Para qué sirven los requerimientos dentro del proceso de desarrollo?

- a) Para elegir el lenguaje de programación.
- b) Como guía para el diseño, implementación y pruebas.**
- c) Para reemplazar la fase de documentación.
- d) Para eliminar errores de compilación.

22. (Beneficios)

¿Qué efecto tiene una buena gestión sobre los riesgos?

- a) Reduce riesgos innecesarios durante el desarrollo.**
- b) Los aumenta de forma controlada.
- c) Los ignora completamente.
- d) Los transfiere al cliente.

23. (Proceso de ingeniería de requerimientos)

¿Qué incluye el proceso de ingeniería de requerimientos?

- a) Planeación, codificación y prueba.
- b) Identificación, análisis, documentación, validación y gestión.**
- c) Diseño, mantenimiento y soporte.
- d) Instalación y actualización.

24. (Párrafo: Técnicas para obtener requerimientos)

¿Qué función tienen las técnicas de obtención de requerimientos?

- a) Escribir el código base del sistema
- b) Identificar, analizar, documentar y validar las necesidades del usuario**
- c) Sustituir la fase de pruebas
- d) Diseñar la interfaz de usuario

25. (Edgar Serna y Phillip Laplante)

¿Qué afirman Serna y Laplante sobre las técnicas de obtención de requerimientos?

- a) Que sólo existe una técnica efectiva.
- b) Que cada proyecto necesita una combinación de métodos.**
- c) Que todas las técnicas son iguales.
- d) Que no son necesarias en proyectos pequeños.

26. (Entrevistas)

¿Cuál es la principal ventaja de las entrevistas?

- a) Requieren poco tiempo.
- b) No necesitan la participación del usuario.
- c) Permiten profundizar y aclarar dudas.**
- d) Son completamente automáticas.

27. (Entrevistas)

¿Qué desventaja pueden tener las entrevistas?

- a) Requieren mucho tiempo si hay varios involucrados.**
- b) Son poco precisas.
- c) No generan resultados útiles.
- d) Dificultan la documentación.

28. (Cuestionarios/Encuestas)

¿Qué ventaja tienen los cuestionarios o encuestas?

- a) No requieren análisis posterior.
- d) Son útiles sólo en proyectos pequeños.
- c) Sustituyen las pruebas finales.
- d) Permiten recopilar información de muchos usuarios rápidamente.**

29. (Cuestionarios/Encuestas)

¿Cuál es una limitación de los cuestionarios o encuestas?

- a) Las respuestas pueden ser generales o imprecisas.**
- b) Requieren reuniones prolongadas.
- c) No pueden aplicarse en línea.
- d) Son muy costosos.

30. (Observación directa)

¿Qué busca la observación directa?

- a) Analizar cómo los usuarios realizan sus actividades diarias.**
- b) Evaluar el rendimiento del sistema.
- c) Medir el tiempo de desarrollo.
- d) Verificar el código fuente.

31. (Observación directa)

¿Qué posible inconveniente tiene esta técnica?

- a) No proporciona información útil.
- b) Puede alterar el comportamiento natural del usuario.**
- c) No requiere observadores.
- d) Siempre toma demasiado tiempo.

32. (Análisis de documentación)

¿Qué permite el análisis de documentación?

- a) Eliminar información obsoleta.
- b) Crear nuevos manuales automáticamente.
- c) Conocer procesos actuales revisando manuales o reportes antiguos.**
- d) Reducir el número de entrevistas.

33. (Análisis de documentación)

¿Cuál es una posible desventaja del análisis documental?

- a) La información puede estar desactualizada.**
- b) Requiere la presencia de los usuarios.
- c) Es demasiado costoso.

d) No aporta información relevante.

34. (Talleres/Workshops)

¿Qué ventaja tienen los talleres de requerimientos?

a) No requieren planificación.

b) Eliminar la validación de requerimientos.

c) Reducir el tiempo de codificación.

d) Fomentan la colaboración entre usuarios y desarrolladores.

35. (Talleres/Workshops)

¿Qué problema puede presentarse en los talleres si no hay buena organización?

a) Se pierde la comunicación.

b) Se omiten requerimientos importantes.

c) Pueden generarse confusiones.

d) Se interrumpe el desarrollo.

36. (Prototipado)

¿Qué consiste el prototipado?

a) Diseñar diagramas de flujo.

b) Construir versiones preliminares del sistema para visualización.

c) Elaborar reportes finales.

d) Crear manuales de usuario.

37. (Prototipado)

¿Qué ventaja tiene el uso de prototipos?

a) Permiten validar requerimientos y detectar errores antes del desarrollo real.

b) Sustituyen la fase de pruebas.

c) Reducen la participación del usuario.

d) Disminuyen la necesidad de documentación.

38. (Prototipado)

¿Qué precaución debe tomarse al usar prototipos?

a) Evitar mostrarlo a los usuarios.

b) Eliminarlo al finalizar el proyecto.

c) No documentarlo.

d) Aclarar que es solo un modelo y no el sistema final.

39. (Tormenta de ideas)

¿Qué busca la técnica de tormenta de ideas?

a) Reemplazar la documentación técnica.

b) Crear reportes de errores.

c) Promover la creatividad del equipo y generar nuevas funciones.

d) Automatizar decisiones.

40. (Tormenta de ideas)

¿Qué debe hacerse después de una sesión de ideas?

a) Filtrar y conservar solo las ideas viables.

b) Implementarlas todas.

- c) Ignorarlas si no son técnicas.
- d) Enviarlas al cliente sin revisión.

41. (Historias de usuario)

¿Qué describen las historias de usuario?

- a) Tareas técnicas internas del sistema.
- b) Manuales de instalación.
- c) Funcionalidades desde el punto de vista del usuario final.**
- d) Diagramas UML.

42. (Historias de usuario)

¿En qué tipo de metodologías se usan las historias de usuario?

- a) En metodologías ágiles.**
- b) En metodologías en cascada.
- c) En proyectos industriales.
- d) En pruebas automáticas.

43. (Historias de usuario)

¿Qué ventaja tienen las historias de usuario?

- a) Evitan la participación del cliente.
- b) Son fáciles de entender**
- c) Sustituyen el código fuente.
- d) No requieren validación.

44. (Historias de usuario)

¿Qué limitación pueden tener las historias de usuario?

- a) Requieren mucho tiempo.
- b) Son difíciles de escribir.
- c) No pueden modificarse.
- d) Pueden omitir detalles técnicos.**

45. (Proceso completo)

¿Cuáles son las fases de la ingeniería de requerimientos?

- a) Identificación, análisis, documentación, validación y gestión.**
- b) Diseño, prueba, mantenimiento, soporte y control.
- c) Planeación, codificación, prueba y entrega.
- d) Modelado, instalación y evaluación.

46. (Diagrama final)

¿Qué muestra el diagrama de flujo del tutorial?

- a) Los errores comunes del proceso.
- b) Los costos de desarrollo.
- c) La relación de cada técnica con las etapas de la ingeniería de requerimientos.**
- d) Los resultados de las pruebas.

47. (Conclusión)

¿Cuál es el fundamento sobre el cual se construye cualquier sistema?

a) El código fuente.

b) Los requerimientos de software.

c) Las pruebas unitarias.

d) El manual de usuario.

48. (Conclusión)

¿Qué puede causar un mal levantamiento de requerimientos?

a) Fallas graves, sobrecostos y pérdida de tiempo.

b) Una ejecución más rápida.

c) Menor comunicación con el cliente.

d) Un producto más flexible.

49. (Conclusión)

¿Qué garantiza una buena gestión de requerimientos?

a) Que el proyecto termine antes de tiempo.

b) Que el producto final sea funcional, confiable y de alta calidad.

c) Que el diseño sea más complejo.

d) Que el código sea más extenso.