Preguntas Generales con respuesta

1. ¿Cuál es tu experiencia previa en pruebas de software?
Posible respuesta: "He trabajado como tester de software durante los últimos cinco años en una empresa de desarrollo de software líder. Mi experiencia incluye la realización de pruebas manuales y automatizadas en una variedad de proyectos, desde aplicaciones web hasta móviles, y he trabajado en equipos ágiles y de desarrollo tradicional."

2. ¿Qué te motivó a buscar un puesto en pruebas de software?
Posible respuesta: "Me apasiona la resolución de problemas y la búsqueda de defectos en el software. Creo que las pruebas de software desempeñan un papel crucial en la creación de productos de calidad que satisfacen las necesidades del usuario final, y quería ser parte de ese proceso."

3. ¿Cuáles son tus habilidades y fortalezas como tester de software?
Posible respuesta: "Mis habilidades incluyen una sólida comprensión de los principios de pruebas de software, tanto manuales como automatizadas. Soy detallista, analítico y tengo una habilidad innata para encontrar defectos. Además, tengo una excelente capacidad para trabajar en equipo y comunicarme eficazmente con diferentes partes interesadas."

4. ¿Cómo te mantienes al tanto de las últimas tendencias y prácticas en pruebas de software?
Posible respuesta: "Me mantengo al tanto de las últimas tendencias y prácticas en pruebas de software mediante la lectura de libros, blogs y artículos en línea, así como la participación en conferencias y grupos de discusión. También me gusta experimentar con nuevas herramientas y técnicas en proyectos de prueba reales."

5. ¿Has trabajado en equipos ágiles o metodologías de desarrollo similares?
Posible respuesta: "Sí, he trabajado en varios equipos ágiles utilizando metodologías como Scrum y Kanban. He participado en reuniones diarias, planificación de sprints y retrospectivas, y he colaborado estrechamente con los equipos de desarrollo para integrar las pruebas en el proceso de desarrollo ágil."

Preguntas Técnicas con respuestas

1. Explícame qué son las pruebas unitarias y cuál es su importancia en el desarrollo de software.
Posible respuesta: "Las pruebas unitarias son pruebas que verifican el comportamiento de unidades individuales de código, como métodos o funciones. Son importantes porque ayudan a garantizar que cada parte del código funcione correctamente de forma independiente, lo que facilita la detección y corrección temprana de errores."

2. ¿Qué es la automatización de pruebas y cuándo es apropiado automatizar pruebas en lugar de realizarlas manualmente?
Posible respuesta: "La automatización de pruebas es el proceso de utilizar herramientas y scripts para ejecutar pruebas de forma automática. Es apropiado automatizar pruebas cuando se realizan tareas repetitivas o de regresión, cuando se necesitan resultados rápidos y consistentes, o cuando se quiere aumentar la cobertura de pruebas."

3. ¿Cuál es la diferencia entre un error y un defecto en el contexto de pruebas de software?
Posible respuesta: "Un error es una discrepancia entre el resultado esperado y el resultado real durante una prueba. Un defecto, por otro lado, es una anomalía en el software que causa un comportamiento no deseado o incorrecto. No todos los errores se convierten en defectos, pero todos los defectos son causados por errores."

4. Describe cómo identificarías y documentarías un defecto durante el proceso de pruebas.
Posible respuesta: "Para identificar un defecto, primero reproduciría el problema siguiendo los pasos detallados en el caso de prueba. Luego, documentaría el defecto proporcionando una descripción clara del problema, pasos para reproducirlo, resultados esperados y reales, y cualquier información adicional relevante, como capturas de pantalla o registros de errores."

5. ¿Qué es la matriz de trazabilidad de requisitos y cómo se utiliza en el proceso de pruebas?
Posible respuesta: "La matriz de trazabilidad de requisitos es una herramienta que rastrea la relación entre los requisitos del software y las pruebas que los validan. Se utiliza para garantizar que todas las funcionalidades especificadas en los requisitos se prueben adecuadamente y para identificar cualquier brecha de cobertura de pruebas en el proceso de pruebas."

Preguntas de Escenarios y Casos de Uso con respuestas

1. ¿Imagina que estás probando una aplicación de comercio electrónico. ¿Qué casos de prueba probarías para garantizar su funcionalidad?
Posible respuesta: "Para garantizar la funcionalidad de una aplicación de comercio electrónico, probaría casos que cubran aspectos como la navegación del sitio, búsqueda de productos, proceso de compra, opciones de pago, gestión de cuentas de usuario y la integración con servicios de envío."

2. ¿Cómo abordarías la prueba de una función de búsqueda compleja en una aplicación web?
Posible respuesta: "Para probar una función de búsqueda compleja, desarrollaría casos de prueba que abarquen diferentes escenarios de búsqueda, como búsqueda por palabra clave, búsqueda avanzada con filtros, búsqueda por categorías y búsqueda de texto completo. También verificaría la precisión y relevancia de los resultados de búsqueda en diferentes contextos."

3. ¿Qué estrategias utilizarías para probar la compatibilidad de una aplicación móvil con diferentes dispositivos y sistemas operativos?
Posible respuesta: "Para probar la compatibilidad de una aplicación móvil, emplearía una combinación de pruebas en dispositivos físicos y simuladores o emuladores para cubrir una amplia variedad de dispositivos y sistemas operativos. Crearía una matriz de compatibilidad y ejecutaría pruebas en diversas combinaciones de dispositivos, versiones de sistema operativo y configuraciones de red."

4. ¿Cómo probarías una función de autenticación y autorización en una aplicación web?
Posible respuesta: "Para probar una función de autenticación y autorización, diseñaría casos de prueba que cubran distintos escenarios, como inicio de sesión exitoso, intentos de inicio de sesión con credenciales incorrectas, recuperación de contraseña y acceso a funciones restringidas. Además, evaluaría la seguridad de las contraseñas y la gestión de sesiones para garantizar la protección de los datos del usuario."

Preguntas de Resolución de Problemas y Escenarios Hipotéticos con respuestas

1. ¿Qué harías si encuentras un defecto crítico en la aplicación justo antes de su lanzamiento?
Posible respuesta: "Si encuentro un defecto crítico justo antes del lanzamiento, primero lo documentaría detalladamente, incluyendo la descripción del problema y los pasos para reproducirlo. Luego, me comunicaría de inmediato con el equipo de desarrollo y los stakeholders relevantes para discutir posibles soluciones y determinar el mejor curso de acción para corregir el defecto lo antes posible."

2. Imagina que recibes una solicitud de cambios de última hora en una característica importante de la aplicación. ¿Cómo manejarías esta situación?
Posible respuesta: "Ante una solicitud de cambios de última hora en una característica importante, primero evaluaría el impacto de los cambios en el proyecto, incluyendo el cronograma y los recursos disponibles. Luego, me comunicaría con el equipo de desarrollo y otros stakeholders para discutir la viabilidad de los cambios y ajustar las estrategias de prueba según sea necesario para garantizar la calidad del software."

3. ¿Qué harías si te enfrentas a una falta de recursos (tiempo, personal, herramientas) para completar todas las pruebas planificadas?
Posible respuesta: "Ante una falta de recursos para completar todas las pruebas planificadas, priorizaría las pruebas según la criticidad y el riesgo, centrándome en las áreas más críticas o susceptibles a errores.

Además, buscaría soluciones alternativas, como la automatización de pruebas o la reasignación de recursos, para maximizar la cobertura de pruebas con los recursos disponibles."

4. ¿Cómo priorizarías las pruebas si te pidieran realizar pruebas exhaustivas con un plazo ajustado?
Posible respuesta: "En caso de un plazo ajustado para realizar pruebas exhaustivas, priorizaría las pruebas basándome en la criticidad y el riesgo de las diferentes funcionalidades del software. Me enfocaría en las áreas críticas o de alto impacto para el negocio, así como en aquellas que tienen mayor probabilidad de contener defectos. También buscaría identificar y ejecutar casos de prueba de regresión automatizados para maximizar la cobertura con el tiempo limitado disponible."

5. ¿Qué acciones tomarías si descubres que una nueva versión de software no es compatible con versiones anteriores del sistema?
Posible respuesta: "Si descubro que una nueva versión de software no es compatible con versiones anteriores del sistema, primero documentaría detalladamente el problema y sus implicaciones. Luego, me comunicaría con el equipo de desarrollo y otros stakeholders para discutir posibles soluciones, como la actualización de las versiones anteriores del sistema o la implementación de parches de compatibilidad, y colaboraría en la resolución del problema para garantizar una transición suave a la nueva versión del software."

6. ¿Qué estrategias utilizarías para resolver conflictos entre los equipos de desarrollo y pruebas durante el ciclo de vida del proyecto?
Posible respuesta: "Para resolver conflictos entre los equipos de desarrollo y pruebas, primero buscaría comprender las preocupaciones y perspectivas de ambas partes. Luego, fomentaría la comunicación abierta y la colaboración entre los equipos, buscando puntos en común y compromisos mutuos. Además, me aseguraría de que todos los equipos tengan un claro entendimiento de los objetivos del proyecto y el impacto de sus decisiones en la calidad del software."

7. Imagina que te asignan probar una nueva característica de la aplicación que no está bien documentada. ¿Cómo abordarías esta tarea?
Posible respuesta: "Ante una nueva característica de la aplicación que no está bien documentada, comenzaría por recopilar toda la información disponible sobre la funcionalidad, incluyendo especificaciones, requisitos y cualquier otra documentación relevante. Luego, trabajaría estrechamente con el equipo de desarrollo y otros stakeholders para aclarar cualquier ambigüedad y elaborar casos de prueba detallados basados en los escenarios de uso esperados. Además, buscaría oportunidades para explorar y probar la funcionalidad en un entorno controlado para identificar posibles defectos y validar su comportamiento."

8. ¿Qué harías si un stakeholder informa de un problema en la aplicación que no puedes replicar durante las pruebas?
Posible respuesta: "Si un stakeholder informa de un problema en la aplicación que no puedo replicar durante las pruebas, primero investigaría detalladamente el problema, incluyendo recopilar información adicional sobre el entorno, la configuración y los pasos específicos que llevaron al problema. Luego, intentaría replicar el problema utilizando diferentes enfoques y recursos, como registros de errores, registros de actividad y entornos de prueba alternativos. Además, me comunicaría con el stakeholder para recopilar más información y colaborar en la identificación y resolución del problema."

9. ¿Cómo comunicarías los resultados de las pruebas y los hallazgos de los defectos al equipo de desarrollo y otros stakeholders?
Posible respuesta: "Para comunicar los resultados de las pruebas y los hallazgos de los defectos, utilizaría un enfoque estructurado y claro, proporcionando informes detallados que incluyan una descripción del problema, pasos para reproducirlo, resultados esperados y reales, y cualquier información adicional relevante, como capturas de pantalla o registros de errores. Además, me aseguraría de que la comunicación sea oportuna y efectiva, utilizando herramientas como reuniones, correos electrónicos y sistemas de seguimiento de problemas para mantener a todos los stakeholders informados sobre el estado y la calidad del software."

Batería Completa de Preguntas y Respuestas

P1. ¿Cuál es la diferencia entre requisito funcional y requisito no funcional?
Respuesta:

- **Requisito funcional:** especifica cómo DEBE HACER el sistema o la aplicación en qué lugar
- **Requisito no funcional:** especifica cómo DEBE SER el sistema o aplicación.

Algunos requisitos funcionales son

- Autenticación
- Reglas del negocio
- Información histórica
- Requisitos legales y reglamentarios
- Interfaces externas

Algunos requisitos no funcionales son

- Actuación
- Fiabilidad
- Seguridad
- Recuperación
- Integridad de los datos
- Usabilidad

P2. ¿Cómo se relacionan la gravedad y la prioridad entre sí?
Respuesta:

- **Gravedad**: indica la gravedad/profundidad del error donde, como
- **Prioridad**: indica qué error debe corregirse primero.
- **Gravedad** - Punto de vista de la aplicación
- **Prioridad** - Punto de vista del usuario

P3. ¿Explica los diferentes tipos de Severidad?
Respuesta:

- Defecto de la interfaz de usuario: bajo
- Defectos relacionados con los límites: medio
- Error en el manejo de defectos-Medio
- Defectos de cálculo: alto
- Defectos en la interpretación de datos: alto
- Fallos y problemas de hardware: alto
- Compatibilidad y defectos entre sistemas-Alta
- Defectos de flujo de control: alto
- Condiciones de carga (pérdidas de memoria durante pruebas de carga) -Alta

P4. ¿Cuál es la diferencia entre prioridad y gravedad?
Respuesta: Los términos Prioridad y Gravedad se utilizan en el seguimiento de errores para compartir la importancia de un error entre el equipo y solucionarlo.

- **Gravedad**: se encuentra en el punto de vista de la aplicación. Prioridad: se encuentra en el punto de vista del usuario. Gravedad: (indica la gravedad/profundidad del error).

El estado de Gravedad se utiliza para explicar qué tan grave está afectando la desviación a la compilación.
El tipo de gravedad lo define el evaluador en función de los casos de prueba escritos y la funcionalidad.

Ejemplo
Si una aplicación o una página web falla cuando se hace clic en un enlace remoto, en este caso es raro que un usuario haga clic en el enlace remoto, pero el impacto del bloqueo de la aplicación es severo, por lo que la gravedad es alta y la prioridad es baja.

- **Prioridad**: indica qué error debe corregirse primero

El estado de prioridad lo establece el evaluador al desarrollador mencionando el plazo para corregir un defecto. Si se menciona Alta prioridad, el desarrollador debe solucionarlo lo antes posible.
El estado de prioridad se establece en función de los requisitos del cliente.

Ejemplo
Si el nombre de la empresa está mal escrito en la página de inicio de un sitio web, entonces la prioridad es alta y la gravedad para solucionarlo es baja.

- **Gravedad**: describe el error en términos de funcionalidad.
- **Prioridad**: describe el error en términos de cliente.

Algunos ejemplos:

- Alta gravedad y baja prioridad -> La aplicación no permite la configuración esperada por el cliente.
- Alta gravedad y alta prioridad -> La aplicación no permite múltiples usuarios.
- Gravedad baja y prioridad alta -> Sin mensaje de error para evitar un funcionamiento incorrecto.
- Gravedad baja y prioridad baja -> El mensaje de error tiene un significado complejo.

Gravedad alta - Prioridad baja
Supongamos que intenta la operación más alocada o extraña en un software (por ejemplo, que se lanzará al día siguiente) que un usuario normal no haría y suponiendo que esto genere un error de tiempo de ejecución en la aplicación, la gravedad sería alta. La prioridad sería baja ya que las operaciones o los pasos que generaron este error con mayor probabilidad no serán realizados por un usuario.

Gravedad baja - Prioridad alta
Un ejemplo sería: encuentra un error de ortografía en el nombre del sitio web que está probando. Digamos que se supone que el nombre es Google y allí se escribe como "Gaogle". Sin embargo, no afecta la funcionalidad básica del software, debe corregirse antes del lanzamiento. Por tanto, la prioridad es alta.

Gravedad alta: prioridad alta
Un error que impide el espectáculo. es decir, un error debido al cual no podemos continuar con nuestras pruebas. Un ejemplo sería un error de tiempo de ejecución durante el funcionamiento normal del software, lo que provocaría que la aplicación se cerrara abruptamente.

Gravedad baja - prioridad baja
Errores cosméticos

P5. ¿Qué es la gravedad del defecto?
Respuesta: Un defecto es una anomalía o defecto del producto, que es una variación de las especificaciones deseadas del producto. La clasificación de un defecto en función de su impacto en el funcionamiento del producto se denomina gravedad del defecto.

P6. ¿Qué es la prueba A/B?
Respuesta: La prueba A/B se utiliza principalmente para estudiar el impacto de varios diseños de productos en las métricas del sitio web; se ejecutaron dos versiones simultáneas en una sola página web o en un conjunto de páginas web para medir la diferencia en las tasas de clics, la interfaz y el tráfico. .

P7. ¿Qué son los criterios de entrada y salida en las pruebas de software?
Respuesta:
Los criterios de entrada son el proceso que debe estar presente cuando un sistema comienza, como:

- SRS (Especificación de requisitos de software)
- FRS (Especificación de requisitos funcionales)
- Caso de uso
- Caso de prueba
- Plan de prueba

Los criterios de salida garantizan si se completan las pruebas y si la aplicación está lista para su lanzamiento, como:

- Informe resumido de la prueba
- Métrica
- Informe de análisis de defectos

P8. ¿Qué son las pruebas de concurrencia?
Respuesta: Las pruebas de concurrencia (también conocidas comúnmente como pruebas multiusuario) se utilizan para conocer los efectos del acceso a la aplicación, módulo de código o base de datos por parte de diferentes usuarios al mismo tiempo. Ayuda a identificar y medir los problemas en el tiempo de respuesta, los niveles de bloqueo y estancamiento en la aplicación.

Ejemplo
El corredor de carga se usa ampliamente para este tipo de pruebas, Vugen (generador de usuarios virtuales) se usa para agregar la cantidad de usuarios simultáneos y cómo se deben agregar los usuarios, comoAumento gradual o aumento gradual.

P9. ¿Explicar la cobertura del extracto/cobertura del código/cobertura de la línea?
Respuesta: La Cobertura de Declaración o Cobertura de Código o Cobertura de Línea es una métrica utilizada en White Box Testing donde podemos identificar las declaraciones ejecutadas y donde el código no se ejecuta debido al bloqueo. En este proceso, todas y cada una de las líneas del código deben sercomprobado y ejecutado.

Algunas ventajas de la Cobertura de Estado de Cuenta / Cobertura de Código / Cobertura de Línea son

- Verifica lo que se espera que haga y no haga el código escrito.
- Mide la calidad del código escrito.
- Comprueba el flujo de diferentes rutas en el programa y también garantiza si esas rutas se prueban o no.

Para calcular la cobertura del estado de cuenta,
Cobertura de estados de cuenta = Declaraciones probadas / Número total de declaraciones.

P10. ¿Explicar la cobertura de sucursales/cobertura de decisiones?
Respuesta: La métrica de cobertura de sucursales o cobertura de decisiones se utiliza para verificar el volumen de pruebas realizadas en todos los componentes. Este proceso se utiliza para garantizar si todo el código se ejecuta verificando cada rama o resultado de decisión (declaraciones if y while) ejecutando al menos una vez, de modo que ninguna rama provoque el fallo de la aplicación.
Para calcular la cobertura de sucursales,
Cobertura de sucursales = Resultados de decisiones probados / Resultados de decisiones totales.

P11. ¿Cuál es la diferencia entre el caso de prueba de alto nivel y el de bajo nivel?
Respuesta: Los casos de prueba de alto nivel son aquellos que cubren la funcionalidad principal de la aplicación (es decir, recuperar, actualizar la visualización, cancelar (casos de prueba relacionados con la funcionalidad), casos de prueba de la base de datos). Los casos de prueba de bajo nivel son aquellos relacionados con la interfaz de usuario (UI) de la aplicación.

P12. ¿Explicar las pruebas de localización con un ejemplo?
Respuesta: La localización es el proceso de cambiar o modificar una aplicación a una cultura o ubicación particular. Esto incluye cambios en la interfaz de usuario, diseños gráficos o incluso la configuración inicial de acuerdo con su cultura y requisitos.
En términos de pruebas de localización, verifica qué tan correctamente se cambia o modifica la aplicación en esa cultura e idioma de destino.
En caso de que se requiera traducción de la solicitud a ese idioma local, se deben realizar pruebas en cada campo para verificar la traducción correcta. Otros formatos como la conversión de fecha, el uso de hardware y software como el sistema operativo también deben considerarse enpruebas de localización.

P13. ¿Explicar el análisis de riesgos en las pruebas de software?
Respuesta: En pruebas de software, el análisis de riesgos es el proceso de identificar riesgos en aplicaciones y priorizándolos para realizar la prueba.
En las pruebas de software, pueden producirse algunos riesgos inevitables, como

- Cambio de requisitos o requisitos incompletos
- Asignación de tiempo para las pruebas.
- Los desarrolladores se retrasan en entregar la compilación para realizar pruebas.
- Urgencia del cliente para la entrega.
- Fuga de defectos debido al tamaño o la complejidad de la aplicación.

Para superar estos riesgos, se pueden realizar las siguientes actividades

- Realización de una reunión de revisión de evaluación de riesgos con el equipo de desarrollo.
- El Perfil de Cobertura de Riesgos se crea mencionando la importancia de cada área.
- Usar el máximo de recursos para trabajar en áreas de alto riesgo, como asignar más probadores para áreas de alto riesgo y recursos mínimos para áreas de riesgo medio y bajo.
- Creación de base de datos de evaluación de riesgos para futuros mantenimientos y revisión de gestión.

P14. ¿Cuál es la diferencia entre pruebas estáticas y pruebas dinámicas?
Respuesta:

- **Pruebas estáticas (realizadas en la etapa de verificación):** La prueba estática es una técnica de prueba de caja blanca en la que los desarrolladores verifican o prueban su código con la ayuda de una lista de verificación para encontrar errores en él; este tipo de prueba se realiza sin ejecutar la aplicación o programa realmente desarrollado. Las revisiones de código, inspecciones y recorridos se realizan principalmente en esta etapa de prueba.
- **Pruebas Dinámicas (realizadas en etapa de Validación):** Las pruebas dinámicas se realizan ejecutando la aplicación real con entradas válidas para verificar el resultado esperado. Ejemplos de metodologías de pruebas dinámicas son pruebas unitarias, pruebas de integración, pruebas de sistemas y pruebas de aceptación.

Algunas diferencias entre las pruebas estáticas y las pruebas dinámicas son:

- Las pruebas estáticas son más rentables que las pruebas dinámicas porque las pruebas estáticas se realizan en la etapa inicial.
- En términos de cobertura de declaración, las pruebas estáticas cubren más áreas que las pruebas dinámicas en menos tiempo.
- Las pruebas estáticas se realizan antes de la implementación del código, mientras que las pruebas dinámicas se realizan después de la implementación del código.
- Las pruebas estáticas se realizan en la etapa de verificación, mientras que las pruebas dinámicas se realizan en la etapa de validación.

P15. Explicar el diagrama de casos de uso. ¿Cuáles son los atributos de los casos de uso?
Respuesta: Los diagramas de casos de uso son una representación gráfica general de la funcionalidad de un sistema. Se utiliza en la fase de análisis de un proyecto para especificar el sistema a desarrollar. En los diagramas de casos de uso, todo el sistema se define como ACTORES, CASOS DE USO y ASOCIACIONES, los ACTORES son la parte externa del sistema como usuarios, software y hardware de computadora, USECASES es el comportamiento o funcionalidad del sistema cuando estos ACTORES realizan una acción. las ASOCIACIONES son la línea trazada para mostrar la conexión entre ACTORES y CASES DE USO. Un ACTOR puede vincular demasiados CASOS DE USO y un ACTOR puede vincular demasiados ACTORES.

P16. ¿Qué son las pruebas de aplicaciones web? Explica las diferentes fases
Respuesta: Las pruebas de aplicaciones web se realizan en un sitio web para verificar su carga, rendimiento, seguridad, funcionalidad, interfaz, compatibilidad y otras cuestiones relacionadas con la usabilidad. En las pruebas de aplicaciones web, se realizan tres fases de prueba, que son:

- Pruebas de nivel web: En las pruebas de nivel web, se probará la compatibilidad del navegador de la aplicación para IE, FireFox y otros navegadores web.
- Pruebas de nivel medio: En las pruebas de nivel medio, se probaron los problemas de funcionalidad y seguridad.
- Pruebas de nivel de base de datos: En las pruebas de nivel de base de datos, se probaron y verificaron la integridad y el contenido de la base de datos.

P17. Explique las pruebas unitarias, las pruebas de interfaz y las pruebas de integración.
Respuesta:

- **Pruebas unitarias:** Las pruebas unitarias se realizan para comprobar si los módulos individuales del código fuente funcionan correctamente. es decir, probar todas y cada una de las unidades de la aplicación por separado por parte del desarrollador en el entorno del desarrollador.
- **Pruebas de interfaz**: La prueba de interfaz se realiza para comprobar si los módulos individuales se comunican correctamente entre sí según las especificaciones. Las pruebas de interfaz se utilizan principalmente para probar la interfaz de usuario de la aplicación GUI.

- **Pruebas de integración**: Las pruebas de integración se realizan para verificar la conectividad combinando todos los módulos individuales y probar la funcionalidad.

P18. ¿Explica las pruebas Alfa, Beta y Gamma?
Respuesta:

- **Prueba alfa**: Alpha Testing es principalmente como realizar pruebas de usabilidad realizadas por los desarrolladores internos que desarrollaron el software o los evaluadores. A veces esta prueba alfa la realiza el cliente o un tercero con la presencia de desarrollador y tester. La versión lanzada después de la prueba alfa se llama Lanzamiento Alfa.
- **Pruebas beta**: La prueba Beta la realizan un número limitado de usuarios finales antes de la entrega; la solicitud de cambio se solucionará si el usuario brinda comentarios o informa un defecto. La versión lanzada después de la prueba beta se llama versión beta.
- **Pruebas gamma**: Las pruebas gamma se realizan cuando el software está listo para su lanzamiento con requisitos específicos; estas pruebas se realizan directamente omitiendo todas las actividades de prueba internas.

P19. ¿Explicar los estándares IEEE 829 y otros estándares de pruebas de software?
Respuesta:Se utiliza un estándar IEEE 829 para la documentación de prueba de software, donde especifica el formato para el conjunto de documentos que se utilizarán en las diferentes etapas de prueba de software.

Los documentos son, Plan de prueba-El plan de pruebas es un documento de planificación que contiene información sobre el alcance, los recursos, la duración, la cobertura de las pruebas y otros detalles.
Diseño de prueba- El documento de diseño de prueba tiene información de los criterios de aprobación de la prueba con las condiciones de la prueba y los resultados esperados.
Caso de prueba-El documento de caso de prueba contiene información sobre los datos de prueba que se utilizarán.

P20. ¿Cuál es la diferencia entre el registro de errores y el seguimiento de defectos?
Respuesta:

- **Registro de errores**: Documento de registro de errores que muestra el número de defectos, como abiertos, cerrados, reabiertos o aplazados, de un módulo en particular.
- **Seguimiento de defectos**: El proceso de seguimiento de un defecto, como síntoma, si es reproducible o no, prioridad, gravedad y estado.

P21. ¿Qué son las pruebas de integración y las pruebas de regresión?
Respuesta:
Pruebas de integración:

- Combinar los módulos y construir una arquitectura de software.
- Para probar el flujo de comunicación y datos
- Se utilizan técnicas de prueba de caja blanca y negra.
- Lo realiza el desarrollador y el evaluador.

Pruebas de regresión

- Se trata de volver a ejecutar nuestras pruebas después de corregir el error para garantizar que la compilación esté libre de errores.
- Hecho después de que se solucione el error

- Lo hace el probador.

P22. ¿Explicar la revisión por pares en las pruebas de software?
Respuesta: Es una forma alternativa de prueba, donde se invitaba a algunos colegas a examinar los productos de su trabajo en busca de defectos y oportunidades de mejora.

P23. ¿Explica las pruebas de compatibilidad con un ejemplo?
Respuesta: Las pruebas de compatibilidad sirven para evaluar la compatibilidad de la aplicación con el entorno informático, como el sistema operativo, la base de datos, la compatibilidad del navegador, la compatibilidad con versiones anteriores, la capacidad informática de la plataforma de hardware y la compatibilidad de los periféricos.

Ejemplo
Si se realizan pruebas de compatibilidad en una aplicación de juego, antes de instalar un juego en una computadora, se verifica su compatibilidad con las especificaciones de la computadora para determinar si es compatible con la computadora que tiene tantas especificaciones o no.

P24. ¿Qué es la Matriz de Trazabilidad?
Respuesta: La Matriz de Trazabilidad es un documento que se utiliza para rastrear el requisito, los casos de prueba y el defecto. Este documento está preparado para que los clientes satisfagan que la cobertura realizada es completa de principio a fin. Este documento consta de Requisito/línea base, número de referencia del documento, caso de prueba/condición, defectos/identificación de error. Con este documento, la persona puede realizar un seguimiento del requisito en función de la identificación del defecto.

P25. ¿Explique las pruebas de valor límite y las pruebas de equivalencia con algunos ejemplos?
Respuesta:
La prueba de valores límite es una técnica para determinar si la aplicación acepta el rango esperado de valores y rechaza los valores que quedan fuera del rango.

Ejemplo
Un cuadro de texto de ID de usuario debe aceptar caracteres alfabéticos (az) con una longitud de 4 a 10 caracteres. BVA se hace así, valor máximo: 10 pases; max-1: 9 pasadas; max+1=11 falla; min=4 pasa; min+1=5 pasa; min-1=3 falla;
Del mismo modo, verificamos los valores de las esquinas y llegamos a una conclusión sobre si la aplicación acepta el rango de valores correcto.

Las pruebas de equivalencia normalmente se utilizan para comprobar el tipo de objeto.

Ejemplo
Un cuadro de texto de ID de usuario debe aceptar caracteres alfabéticos (a - z) con una longitud de 4 a 10 caracteres. En la condición +ve hemos probado el objeto dándole alfabetos. es decir, solo az char, después de eso debemos verificar si el objeto acepta el valor, pasará.
En la condición -ve, tenemos que realizar la prueba dando caracteres que no sean alfabetos (az), es decir, AZ, 0-9, espacios en blanco, etc., fallará.

P26. ¿Qué son las pruebas de seguridad?
Respuesta: Las pruebas de seguridad son el proceso que determina que los datos confidenciales permanezcan confidenciales o ¿Está probando qué tan bien protege el sistema contra el acceso interno o externo no autorizado, daños intencionales, etc.?
Este proceso implica pruebas funcionales, pruebas de penetración y verificación.

P27. ¿Qué son las pruebas de instalación?
Respuesta: Las pruebas de instalación se realizan para verificar si el hardware y el software están instalados y configurados correctamente. Esto garantizará que todos los componentes del sistema se hayan utilizado durante el proceso de prueba. Esta prueba de instalación buscará un gran volumen de datos, mensajes de error y pruebas de seguridad.

P28. ¿Qué es la fuga por defecto?
Respuesta: La fuga de defectos ocurre en el lado del Cliente o del usuario final después de la entrega de la aplicación. Después del lanzamiento de la aplicación al cliente, si el usuario final presenta algún tipo de defecto al usar esa aplicación, se denomina fuga de defectos. Esta fuga de defectos también se denomina fuga de errores.

P29. ¿Cuáles son los contenidos de un informe de error eficaz?
Respuesta:

- Proyecto
- Sujeto
- Descripción
- Resumen
- Detectado por (Nombre del probador)
- Asignado a (Nombre del desarrollador responsable del error)
- Cable de prueba (nombre)
- Detectado en la versión
- Cerrado en versión
- Fecha de detección
- Fecha prevista de cierre
- Fecha real de cierre
- Prioridad (Media, Baja, Alta, Urgente)
- Gravedad (varía de 1 a 5)
- Estado
- ID de error
- Adjunto
- Caso de prueba fallido (caso de prueba que falló debido al error)

P30. ¿Qué es la adivinación de errores y la siembra de errores?
Respuesta: Error Guessing es una técnica de diseño de casos de prueba en la que el evaluador tiene que adivinar qué fallas podrían ocurrir y diseñar las pruebas para representarlas.
La siembra de errores es el proceso de agregar fallas conocidas intencionalmente en un programa con el fin de monitorear la tasa de detección y eliminación y también para estimar la cantidad de fallas que quedan en el programa.

P31. ¿Qué son las pruebas ad hoc?
Respuesta: Las pruebas ad hoc se refieren a las pruebas de aplicaciones sin seguir ninguna regla o caso de prueba. Para las pruebas ad hoc se debe tener un conocimiento sólido sobre la aplicación.

P32. ¿Cuáles son las soluciones básicas para los problemas de desarrollo de software?
Respuesta:

- Se debe desarrollar un requisito claro, detallado, completo, alcanzable y comprobable. Utilice algunos prototipos para ayudar a precisar los requisitos. En entornos ágiles, se necesita una coordinación estrecha y continua con los clientes/usuarios finales.
- Los horarios deben ser realistas: tiempo suficiente para planificar, diseñar, probar, corregir errores, volver a probar, cambiar y documentar en el cronograma establecido.
- Las pruebas deben iniciarse temprano, deben volver a probarse después de que se corrija o cambie el error, se debe dedicar suficiente tiempo a las pruebas y la corrección de errores.
- Estudio adecuado sobre los requisitos iniciales. Esté preparado para ocuparse de más cambios después de que haya comenzado el desarrollo y esté preparado para explicar los cambios realizados a los demás. Trabajar en estrecha colaboración con los clientes y usuarios finales para gestionar las expectativas. Esto evita cambios excesivos en las etapas posteriores.
- Comunicación: realizar inspecciones y recorridos frecuentes en el período de tiempo adecuado. Garantizar que la información y la documentación estén disponibles en formato electrónico actualizado, si es posible. Más énfasis en promover el trabajo en equipo y la cooperación dentro del equipo; utilizar prototipos y una comunicación adecuada con los usuarios finales para aclarar sus dudas y expectativas.

P33. ¿Cuáles son los problemas comunes en el proceso de desarrollo de software?
Respuesta:

- Requisitos inadecuados del Cliente: si los requisitos dados por el cliente no son claros, inacabados y no comprobables, pueden surgir problemas.
- Horarios poco realistas: a veces se le da demasiado trabajo al desarrollador y se le pide que lo complete en un corto período de tiempo, entonces los problemas son inevitables.
- Pruebas insuficientes: los problemas pueden surgir cuando el software desarrollado no se prueba adecuadamente.
- Dado otro trabajo bajo el proceso existente: la solicitud de la alta dirección para trabajar en otro proyecto o tarea traerá algunos problemas cuando el proyecto se esté probando en equipo.
- Falta de comunicación: en algunos casos, el desarrollador no fue informado sobre los requisitos y expectativas del Cliente, por lo que puede haber desviaciones.

P34. ¿Cuál es la diferencia entre pruebas de software y control de calidad (QA)?
Respuesta:

- Las pruebas de software implican la operación de un sistema o aplicación en condiciones controladas y la evaluación del resultado. Está orientado a la 'detección'.
- El aseguramiento de la calidad (QA) involucra todo el PROCESO de desarrollo de software: monitorear y mejorar el proceso, asegurarse de que se sigan los estándares y procedimientos acordados y garantizar que se encuentren y resuelvan los problemas. Está orientado a la "prevención".

P35. ¿Cómo probar la botella de agua?
Respuesta:
Nota: Antes de generar alguna idea de prueba sobre cómo probar una botella de agua, me gustaría hacer algunas preguntas como:

- ¿Es una botella hecha de vidrio, plástico, caucho, algún metal, algún tipo de material desechable o cualquier otra cosa?

- ¿Está pensado sólo para agua caliente o podemos usarlo con otros fluidos como té, café, refrescos, chocolate caliente, sopas, vino, aceite de cocina, vinagre, gasolina, ácidos, lava fundida (!), etc.?
- ¿Quién va a utilizar esta botella? ¿Un niño que va a la escuela, un ama de casa, una empresa fabricante de bebidas, un empleado de oficina, un deportista, una turba que protesta en un mitin (que va a utilizar como misiles), un esquimal que vive en un iglú o un astronauta en una nave espacial?

Este tipo de preguntas pueden permitir que un evaluador conozca mejor un producto (que va a probar). En nuestro caso, supongo que la botella de agua tiene forma de botella para mascotas y en realidad está hecha de plástico o vidrio (hay 2 versiones del producto) y está diseñada para usarse principalmente con agua. Acerca del usuario objetivo, ¡ni siquiera la empresa fabricante está segura de él! (¡Suena familiar! ¡Cuando una empresa de software desarrolla un producto sin una idea clara sobre los usuarios que van a utilizar el software!)

Ideas de prueba

- Verifique las dimensiones de la botella. ¡Mira si realmente parece una botella de agua o un cilindro, un cuenco, una taza, un florero, un portalápices o un cubo de basura! [¡Prueba de verificación de construcción!]
- Vea si la tapa encaja bien con la botella.[¡Pruebas de instalación!]
- Pruebe si la boca de la botella no es demasiado pequeña para verter agua. [¡Pruebas de usabilidad!]
- Llene la botella con agua y manténgala sobre una superficie lisa y seca. Mira si gotea. [¡Pruebas de usabilidad!]
- Llene la botella con agua, séllela con la tapa y observe si hay fugas de agua cuando la botella se inclina, se invierte o se aprieta (en el caso de una botella de plástico). [¡Pruebas de usabilidad!]
- Tome agua en la botella y guárdela en el refrigerador para que se enfríe. Mira qué pasa. [¡Pruebas de usabilidad!]
- Mantenga una botella llena de agua en el refrigerador durante mucho tiempo (digamos una semana). Vea qué pasa con el agua y/o la botella. [¡Pruebas de estrés!]
- Mantenga una botella llena de agua en condiciones de congelación. Observa si la botella se expande (si es de plástico) o se rompe (si es de vidrio). [¡Pruebas de estrés!]
- ¡Intenta calentar (¡hervir!) agua manteniendo la botella en un horno de microondas. [¡Pruebas de estrés!]Vierta un poco de agua caliente (¡hirviente!) en la botella y observe el efecto. [¡Pruebas de estrés!]
- Mantenga una botella seca durante mucho tiempo. Mira qué pasa. Vea si se produce alguna deformación física o química en la botella.
- Pruebe el agua después de guardarla en la botella y vea si hay algún cambio químico. Vea si es seguro consumirla como agua potable.
- Mantenga agua en la botella por algún tiempo. Y ver si cambia el olor del agua.
- Intente usar la botella con diferentes tipos de agua (como agua dura y blanda). [¡Pruebas de compatibilidad!]
- Intenta beber agua directamente de la botella y comprueba si te resulta cómodo de usar. O se derrama agua al hacerlo. [¡Pruebas de usabilidad!]
- Pruebe si el biberón tiene un diseño ergonómico y si es cómodo de sostener. Fíjate también si el centro de gravedad de la botella se mantiene bajo (tanto cuando está vacía como cuando está llena de agua) y no se cae fácilmente.
- Deje caer la botella desde una altura razonable (puede ser la altura de una mesa de comedor) y vea si se rompe (tanto en el modelo de plástico como en el de vidrio). Si se trata de una botella de vidrio, en la mayoría de los casos puede romperse. Fíjate si se rompe en pedacitos diminutos (que suelen ser difíciles de limpiar) o en bonitos trozos grandes (que podrían limpiarse sin mucha dificultad). [¡Pruebas de estrés!] [¡Pruebas de usabilidad!]
- Pruebe la idea de prueba anterior con botellas vacías y botellas llenas de agua. [¡Pruebas de estrés!]

- Pruebe si la botella está hecha de material que sea reciclable. En el caso de una botella de plástico, pruebe si se puede romper fácilmente.
- Pruebe si la botella también se puede usar para contener otros artículos domésticos comunes como miel, jugo de frutas, combustible, pintura, trementina, cera líquida, etc. [¡Prueba de capacidad!]

P36. ¿Qué es la partición de equivalencia?
Respuesta:
Conceptos: la partición de equivalencia es un método para derivar casos de prueba. En este método, las clases de condiciones de entrada llamadas clases de equivalencia son identificados de manera que cada miembro de la clase cause el mismo tipo de que se produzca el procesamiento y la salida. En este método, el evaluador identifica varias clases de equivalencia para la partición. Una clase es un conjunto de condiciones de entrada que probablemente el sistema maneje de la misma manera. Si el sistema manejara erróneamente un caso de la clase, manejaría todos los casos erróneamente.

P37. ¿Por qué aprender la partición por equivalencia?
Respuesta: La partición de equivalencia reduce drásticamente la cantidad de casos de prueba necesarios para probar un sistema de manera razonable. Es un intento de obtener una buena "tasa de aciertos", para encontrar la mayor cantidad de errores con el menor número de casos de prueba.

DISEÑO DE CASOS DE PRUEBA UTILIZANDO PARTICIÓN DE EQUIVALENCIA
Respuesta: Para utilizar la partición de equivalencia, deberá realizar dos pasos.

1. Identificar las clases de equivalencia.
2. Diseñar casos de prueba

PASO 1:
IDENTIFICAR CLASES DE EQUIVALENCIA Tome cada condición de entrada descrita en la especificación y obtenga al menos dos clases de equivalencia para ella. Una clase representa el conjunto de casos que satisfacen la condición (la clase válida) y otra representa los casos que no la cumplen (la clase inválida). A continuación se presentan algunas pautas generales para identificar clases de equivalencia: a) Si los requisitos establecen que se ingresa un valor numérico al sistema y debe estar dentro de un rango de valores, identifique una clase de entradas válidas que estén dentro del rango válido y dos clases de equivalencia no válidas, entradas que sean demasiado bajas y entradas que sean demasiado altas. Por ejemplo, si un artículo en el inventario puede tener una cantidad de -9999 a +9999, identifique las siguientes clases:

- Una clase válida: (QTY es mayor o igual a -9999 y es menor o igual a 9999). Esto se escribe como (- 9999 < = CANTIDAD < = 9999)
- La clase no válida (QTY es menor que -9999), también escrita como (QTY < -9999)
- La clase no válida (QTY es mayor que 9999), también escrita como (QTY >9999) b) Si los requisitos establecen que la cantidad de elementos ingresados por el sistema en algún momento debe estar dentro de un rango determinado, especifique una clase válida donde El número de entradas está dentro del rango válido, una clase no válida donde hay muy pocas entradas y una clase no válida donde hay demasiadas entradas.

P38. ¿Cuáles son dos tipos de métricas?
Respuesta:

- Métricas de proceso: las métricas primarias también se denominan métricas de proceso. Esta es la métrica que preocupa a los profesionales de Six Sigma y en la que pueden influir. Las métricas primarias son casi la característica de resultado directo de un proceso. Es una medida de un proceso y no una medida de un objetivo empresarial de alto nivel. Las métricas

principales del proceso suelen ser los defectos del proceso, el tiempo del ciclo del proceso y el consumo del proceso.

- Métricas del producto: Las métricas del producto caracterizan cuantitativamente algún aspecto de la estructura de un producto de software, como una especificación de requisitos, un diseño o un código fuente.

P39. ¿Cuál es el resultado de las pruebas?
Respuesta: Una aplicación estable, que realiza su tarea como se esperaba.

P40. ¿Por qué opta por las pruebas de caja blanca cuando las pruebas de caja negra están disponibles?
Respuesta: Un punto de referencia que certifica aspectos comerciales (negocios) y también aspectos funcionales (técnicos) son los objetivos de las pruebas de caja negra. Aquí los bucles, estructuras, matrices, condiciones, archivos, etc. son de nivel muy micro, pero son Basement para cualquier aplicación, por lo que White Box toma estas cosas en el nivel Macro y las prueba.

P41. ¿Qué es el documento de referencia? ¿Puede decir dos?
Respuesta: Un documento de referencia, que inicia la comprensión de la aplicación antes de que el evaluador comience las pruebas reales. Documento de especificaciones funcionales y requisitos comerciales

P42. ¿Diga nombres de algún tipo de prueba que haya aprendido o experimentado?
Respuesta: Es bueno decir en la entrevista 5 o 6 tipos relacionados con el perfil de la empresa.

- Pruebas ad hoc
- Prueba de cookies
- CET (Prueba de experiencia del cliente)
- Prueba de profundidad
- Evento conducido
- Pruebas de rendimiento
- Pruebas de recuperación
- Test de sanidad
- Pruebas de seguridad
- Prueba de humo
- Pruebas web

P43. ¿Qué es exactamente el enfoque de lista de verificación heurística para pruebas unitarias?
Respuesta: Es un método para lograr la solución más adecuada de varias encontradas mediante métodos alternativos que se seleccionan en sucesivas etapas de prueba. La lista de verificación Preparado para proceder se llama lista de verificación heurística

P44. ¿Qué es una guía de datos?
Respuesta: Las pautas de datos se utilizan para especificar los datos necesarios para completar el banco de pruebas y preparar los guiones de prueba. Incluye todos los parámetros de datos que se requieren para probar las condiciones derivadas del requisito/especificación. El documento, que respalda la preparación de los datos de prueba, se denomina Pautas de datos.

P45. ¿Por qué optas por Test Bed?
Respuesta: Cuando se ejecuta la condición de prueba, su resultado debe compararse con el resultado de la prueba (resultado esperado), ya que para esto se necesitan datos de prueba, aquí viene el papel de banco de pruebas donde se preparan los datos de prueba.

P46. ¿Por qué preparamos condiciones de prueba, casos de prueba y script de prueba (antes de comenzar la prueba)?
Respuesta: Estos son documentos de diseño de prueba que se utilizan para ejecutar las pruebas reales. Sin los cuales la ejecución de las pruebas es imposible. Finalmente, esta ejecución encontrará los errores que se corregirán, por lo que hemos preparado estos documentos.

P47. ¿No es una pérdida de tiempo preparar la condición de prueba, el caso de prueba y el script de prueba?
Respuesta: Ningún documento preparado en ningún proceso es una pérdida de tiempo. Tampoco se puede decir que los documentos de diseño de prueba que desempeñan un papel vital en la ejecución de la prueba sean una pérdida de tiempo, sin los cuales no se pueden realizar las pruebas adecuadas.

P48. ¿Cómo se realizan las pruebas de una aplicación web?
Respuesta: Para abordar la prueba de una aplicación web, el primer ataque a la aplicación debe ser su comportamiento de rendimiento, ya que eso es muy importante para una aplicación web y luego la transferencia de datos entre el servidor web y el servidor front-end, el servidor de seguridad y el servidor back-end.

P49. ¿Qué tipo de documento necesita para realizar una prueba funcional?
Respuesta: La especificación funcional es el documento definitivo, que expresa todas las funcionalidades de la aplicación y otros documentos como el manual del usuario y BRS también son necesarios para las pruebas funcionales. El documento de análisis de brechas agregará valor para comprender el sistema esperado y existente.

P50. ¿Se pueden realizar las pruebas del sistema en cualquier etapa?
Respuesta: No. El sistema en su totalidad se puede probar solo si todos los módulos están integrados y todos los módulos funcionan correctamente. Las pruebas del sistema deben realizarse antes de la UAT (prueba de aceptación del usuario) y antes de la prueba unitaria.

P51. ¿Qué son las pruebas de mutación y cuándo se pueden realizar?
Respuesta: La prueba de mutación es una poderosa técnica de prueba basada en fallas para pruebas a nivel unitario. Dado que es una técnica de prueba basada en fallas, su objetivo es probar y descubrir algunos tipos específicos de fallas, es decir, cambios sintácticos simples en un programa. Las pruebas de mutación se basan en dos supuestos: la hipótesis del programador competente y el efecto de acoplamiento. La hipótesis del programador competente supone que los programadores competentes escriben programas casi "correctos". El efecto de acoplamiento indicó que un conjunto de datos de prueba que puede descubrir todas las fallas simples en un programa también es capaz de detectar fallas más complejas. Las pruebas de mutación inyectan fallas en el código para determinar las entradas de prueba óptimas.

P52. ¿Por qué es imposible probar un programa por completo?
Respuesta: Con cualquier software que no sea el más pequeño y simple, hay demasiadas entradas, demasiadas salidas y demasiadas combinaciones de rutas para realizar una prueba completa. Además, las especificaciones del software pueden ser subjetivas y interpretarse de diferentes maneras.

P53. ¿Cómo revisará el caso de prueba y cuántos tipos hay?
Respuesta: Hay 2 tipos de revisión:

- Revisión informal:revisión de líderes técnicos.
- Revisión por pares:por un par de la misma organización (¿recorrido? técnico - inspección).

Reseñas:

- Revisión de gestión
- Revisión técnica
- Revisión de código
- Revisión formal (inspecciones y auditorías)
- Revisión informal (revisión por pares y revisión de código)

Objetivos de las revisiones:

- Encontrar defectos en los requisitos.
- Encontrar defectos en el Diseño.
- Identificar desviaciones en cualquier proceso y también proporcionar sugerencias valiosas para mejorar el proceso.

P54. ¿A qué te refieres con pruebas piloto?
Respuesta: La prueba piloto implica que un grupo de usuarios finales prueben el sistema antes de su implementación completa para brindar comentarios sobre las características y funciones de IIS 5.0.

La prueba piloto es una actividad de prueba que se asemeja al entorno de producción.

- Se hace exactamente entre UAT y caída de producción.
- Pocos usuarios que simulan el ambiente de Producción para continuar la Actividad Empresarial con el Sistema.
- Verificarán la funcionalidad principal del sistema antes de entrar en producción. Básicamente, esto se hace para evitar los desastres de alto nivel.
- La prioridad de las pruebas piloto es alta y los problemas que surjan en las pruebas piloto deben solucionarse lo antes posible.

P55. ¿Qué es SRS y BRS en las pruebas manuales?
Respuesta: BRS es Especificación de requisitos comerciales, lo que significa que el cliente que desea crear la aplicación proporciona la especificación a la organización de desarrollo de software y luego la organización la convierte a SRS (Especificación de requisitos de software) según la necesidad del software.

P56. ¿Qué es la prueba de humo y la prueba de cordura? ¿Cuándo se utilizarán las pruebas anteriores?
Respuesta:

- **Prueba de humo**:Se hace para asegurarnos de si la compilación que obtuvimos es comprobable o no, es decir, para verificar la capacidad de prueba de la compilación, también denominada verificación del "día 0". Hecho en el 'nivel de construcción'
- **Pruebas de cordura**:Se realiza durante la fase de lanzamiento para comprobar las funcionalidades principales sin profundizar más. A veces también se denomina subconjunto de pruebas de regresión. Cuando no se realizan pruebas de regresión rigurosas en la compilación,

la cordura hace esa parte al verificar las funcionalidades principales. Hecho en el 'nivel de lanzamiento'

P57. ¿Qué es la depuración?
Respuesta: La depuración consiste en encontrar y eliminar "errores" que hacen que el programa responda de una forma no prevista.

P58. ¿Qué es la determinación?
Respuesta: La determinación tiene diferentes significados en diferentes situaciones. Determinación significa una intención fuerte o una intención fija para lograr un propósito específico. La determinación, como valor fundamental, significa tener una gran fuerza de voluntad para lograr una tarea en la vida. Determinación significa un fuerte sentido de devoción y compromiso personal para lograr o realizar una tarea determinada. Se sabe que las personas que están decididas a lograr diversos objetivos en la vida tienen mucho éxito en diversos ámbitos de la vida.
De otra manera, también podría significar calcular, conocer o incluso realizar una determinada cantidad, límite, carácter, etc. También se refiere a un resultado determinado de tal determinación o incluso definición de un determinado concepto.
También puede significar tomar una decisión particular y lograr firmemente su propósito.

P59. ¿Cuál es la diferencia exacta entre depuración y prueba?
Respuesta: Las pruebas no son más que encontrar un error/problema y las realizan los evaluadores, mientras que la depuración no es más que encontrar la causa raíz del error/problema y los desarrolladores se encargan de eso.

- Depuración: está eliminando el error y lo realiza el desarrollador.
- Pruebas: está identificando el error y lo realiza el evaluador.

P60. ¿Qué son las pruebas de conformidad?
Respuesta: El proceso de probar que una implementación se ajusta a la especificación en la que se basa. Generalmente se aplica para probar la conformidad con una norma formal.

P61. ¿Qué son las pruebas basadas en el contexto?
Respuesta: La escuela de pruebas de software impulsada por el contexto es una versión de Agile Testing que aboga por la evaluación continua y creativa de las oportunidades de prueba a la luz de la información potencial revelada y el valor de esa información para la organización en este momento.

P62. ¿Qué son las pruebas de un extremo a otro?
Respuesta: De manera similar a las pruebas de sistemas, el extremo 'macro' de la escala de pruebas implica probar un entorno de aplicación completo en una situación que imita el uso del mundo real, como interactuar con una base de datos, usar comunicaciones de red o interactuar con otro hardware, aplicaciones, o sistemas si procede.

P63. ¿Cuándo deberían finalizar las pruebas?
Respuesta: Las pruebas son un proceso sin fin, debido a algunos factores, las pruebas pueden terminar. Los factores pueden ser que se ejecutan la mayoría de las pruebas, la fecha límite del proyecto, el agotamiento del presupuesto de las pruebas y la tasa de errores cae por debajo de los criterios.

P64. ¿Qué son las pruebas paralelas/de auditoría?

Respuesta: Prueba en la que el usuario concilia la salida del nuevo sistema con la salida del sistema actual para verificar que el nuevo sistema realiza las operaciones correctamente.

P65. ¿Cuáles son las funciones de las herramientas de prueba de caja de cristal y caja negra?
Respuesta:

- **Pruebas de caja negra**: No se basa en el conocimiento del diseño o código interno. Las pruebas se basan en requisitos y funcionalidad. La prueba de caja negra se utiliza para encontrar los errores siguientes:
    - Errores de interfaz
    - Errores de rendimiento
    - Errores de inicialización
    - Funcionalidad incorrecta o faltante
    - Errores al acceder a la base de datos externa
- **Pruebas en caja blanca**: Se basa en el diseño interno de un código de aplicación. Las pruebas se basan en la cobertura de rutas, la cobertura de sucursales y la cobertura de estados de cuenta. También se conoce como prueba de caja blanca. Los casos de prueba de caja blanca pueden verificar:
    - Todas las rutas independientes dentro de un módulo se ejecutan al menos una vez.
    - Ejecutar todos los bucles
    - Ejercita todas las decisiones lógicas.
    - Ejercer la estructura de datos interna para asegurar su validez.

P66. ¿Cuál es su experiencia con el control de cambios? Nuestro equipo de desarrollo tiene sólo 10 miembros. ¿Crees que gestionar el cambio es tan importante para nosotros?
Respuesta: Siempre que se produzcan modificaciones en el proyecto actual se adaptan todos los documentos correspondientes a la información. Para mantener los documentos siempre sincronizados con el producto en cualquier momento.

P67. ¿Qué es el análisis de brechas?
Respuesta: El análisis de brechas se puede realizar mediante una matriz de trazabilidad, lo que significa rastrear cada requisito individual en SRS hasta varios productos de trabajo.

P68. ¿Cómo saber si su código cumple con las especificaciones?
Respuesta: Con la ayuda de la matriz de trazabilidad. Todos los requisitos se rastrean hasta los casos de prueba. Cuando todos los casos de prueba se ejecutan y pasan es una indicación de que el código ha cumplido con los requisitos.

P69. En su opinión, ¿en qué etapa del ciclo de vida comienzan las pruebas?
Respuesta: Las pruebas son un proceso continuo y comienzan cuando los requisitos del proyecto /producto comienza a ser enmarcado. Fase de requisitos: se realizan pruebas para comprobar si los detalles del proyecto/producto reflejan las ideas de los clientes o dan una idea del proyecto completo desde la perspectiva del cliente (como él deseaba) o no.

P70. ¿Cuáles son las propiedades de un buen requisito?
Respuesta: Las especificaciones de requisitos son importantes y uno de los métodos más confiables para asegurar problemas en un proyecto de software complejo. Los requisitos son los detalles que describen la funcionalidad y las propiedades percibidas externamente de una aplicación. Los requisitos deben ser claros, completos, razonablemente detallados, cohesivos, alcanzables y comprobables.

P71. ¿Cómo se analiza, organiza y ejecuta un proyecto de prueba?
Respuesta: El Alcance se puede definir desde la BRS, SRS, FRS o desde puntos funcionales. Puede ser cualquier cosa proporcionada por el cliente. Y en cuanto a la organización, debemos analizar la funcionalidad que se cubrirá y quién probará los módulos y los pros y los contras de la aplicación. Identifique el número de casos de prueba, asignación de recursos, cuáles son los riesgos que necesitamos mitigar, todo esto entra en escena.
Una vez hecho esto, es muy fácil ejecutarlo según el plan que hemos trazado.

P72. ¿Cómo garantizaría una cobertura del 100% de las pruebas?
Respuesta: No podemos realizar pruebas al 100% en ninguna aplicación. pero los criterios para garantizar la finalización de la prueba en un proyecto son:

- Todos los casos de prueba se ejecutan con un cierto porcentaje de aprobación.
- El error cae por debajo de cierto nivel
- Presupuesto de prueba agotado
- Plazos alcanzados (proyecto o prueba)
- Cuando todas las funcionalidades están cubiertas en un caso de prueba.
- Todos los errores críticos y altos deben tener un estado CERRADO

P73. ¿Vas a probar una aplicación web?
Respuesta: Idealmente, para probar una aplicación web, se deben probar los componentes y la funcionalidad tanto del lado del cliente como del servidor. Pero es practicamente imposible.
El mejor enfoque para examinar los requisitos del proyecto, establecer prioridades basadas en el análisis de riesgos y luego determinar dónde centrar los esfuerzos de prueba dentro de las limitaciones de presupuesto y cronograma.
Para probar una aplicación web, necesitamos realizar pruebas tanto para la GUI como para la arquitectura cliente-servidor.
Con base en muchos factores, como los requisitos del proyecto, el análisis de riesgos, el presupuesto y el cronograma, podemos determinar qué tipo de prueba será apropiada para su proyecto. Podemos realizar pruebas de integración unitaria n, pruebas de funcionalidad, pruebas de GUI, pruebas de usabilidad, pruebas de compatibilidad, pruebas de seguridad, pruebas de rendimiento, pruebas de recuperación y pruebas de regresión.

P74. ¿Cuáles son sus puntos fuertes?
Respuesta: Estoy bien motivado, bien organizado, buen jugador de equipo, dedicado al trabajo y tengo un fuerte deseo de triunfar, y siempre estoy listo y dispuesto a aprender nueva información y habilidades.

P75. ¿Cuándo debería comenzar a realizar las pruebas?
Respuesta: Para cualquier proyecto, la actividad de prueba estará ahí desde el principio en adelante. Después de recopilar los requisitos, se preparará el documento de diseño (alto y bajo), que se probará, ya sea que confirmen los requisitos o no, se diseñará y luego se codificará el cuadro blanco. Después de que la compilación o el sistema esté listo, se realizará la integración seguida de pruebas funcionales, hasta que el producto o proyecto esté estable. Una vez que el producto o proyecto esté estable, se detendrán las pruebas.

P76. ¿Cuándo debería comenzar a planificar las pruebas?
Respuesta: La planificación de la prueba la realiza el líder de prueba. Como líder de prueba, la planificación de la prueba comienza cuando el gerente del proyecto finaliza el TRM y lo entrega al líder de prueba. Aquí el líder de prueba tiene algunas responsabilidades, esas son:

- Formación del equipo de prueba.
- Identificar riesgos tácticos
- Preparando el plan de prueba
- Reseñas sobre planes de prueba.

P77. ¿Te gustaría trabajar en equipo o solo, por qué?
Respuesta: Me gustaría trabajar en equipo. Porque el proceso de desarrollo de software
Es como una carrera de relevos donde muchos corredores tienen que aportar en sus respectivas vueltas. Es importante porque la complejidad del trabajo y el grado de esfuerzo requerido están más allá del nivel de un individuo.

P78. ¿Cuándo deberían comenzar las pruebas en un proyecto? ¿Por qué?
Respuesta: Pruebas en una actividad continua realizada en cada etapa del proyecto. Primero prueba todo lo que obtiene del cliente. Como tester (probador técnico), mi trabajo comenzará tan pronto como comience el proyecto.

P79. ¿Alguna vez ha creado un plan de prueba?
Respuesta: Esta es solo una respuesta de muestra: "Nunca he creado ningún plan de prueba. Desarrollé y ejecuté un caso de prueba. Pero estuve involucrado/participé activamente con mi líder de equipo mientras creaba los planes de prueba".

P80. Define calidad para mí como tú la entiendes.
Respuesta: Es un software que está razonablemente libre de errores y se entrega a tiempo y dentro del presupuesto, cumple con los requisitos y expectativas y es mantenible.

P81. ¿Cuál es el papel del control de calidad en un proyecto de desarrollo?
Respuesta: Quality Assurance Group asegura la Calidad y debe monitorear todo el proceso de desarrollo. se concentran más en la prevención de errores.
Debe establecer estándares, introducir procedimientos de revisión y educar a las personas sobre mejores formas de diseñar y desarrollar productos.

P82. ¿Qué tan involucrado estuvo con su líder de equipo en la redacción del plan de prueba?
Respuesta: Según tengo entendido, los miembros de prueba siempre están fuera del alcance mientras se prepara el plan de prueba, el plan de prueba es un documento de nivel superior para el equipo de prueba. El plan de prueba incluye propósito, alcance, alcance del cliente/cliente, cronograma, hardware, entregables y casos de prueba, etc. Plan de prueba derivado de PMP (Plan de gestión de proyectos). El alcance de los miembros del equipo es simplemente pasar por el PLAN DE PRUEBA y luego llegar a saber cuáles son todas sus responsabilidades y la entrega de los módulos.
El plan de prueba es solo para documentos de entrada para cada equipo de prueba, así como para el líder de prueba.

P83. ¿Qué procesos/metodologías conoce?
Respuesta:

- Metodología espiral
- Metodología de cascada.
- Procesamiento unificado racional (IBM)
- Desarrollo rápido de aplicacioness (Microsoft)

P84. ¿Qué son las pruebas de globalización?
Respuesta: El objetivo de las pruebas de globalización es detectar problemas potenciales en el diseño de aplicaciones que podrían inhibir la globalización. Garantiza que el código pueda manejar todo el soporte internacional sin interrumpir la funcionalidad que causaría pérdida de datos o problemas de visualización.

P85. ¿A quién debería contratar en un grupo de prueba y por qué?
Respuesta: Las pruebas son una parte interesante del ciclo del software. y es responsable de proporcionar un producto de calidad a un cliente. Implica encontrar errores, lo cual es más difícil y desafiante. Quiero ser parte del grupo de pruebas por esto.

P86. ¿Cuál cree que debería ser el papel del lead del equipo de prueba? ¿En relación con la alta dirección? ¿En relación con otros grupos técnicos de la empresa? ¿En relación con su personal?
Respuesta: Las funciones del manager del equipo de pruebas incluyen:

- Tasas de búsqueda y cierre de defectos por semana, normalizadas según el nivel de esfuerzo (¿estamos encontrando defectos y pueden los desarrolladores mantenerse al día con el número encontrado y los que es necesario corregir?)
- Número de pruebas planificadas, ejecutadas y aprobadas por semana (¿sabemos qué tenemos que probar y podemos hacerlo?)
- Defectos encontrados por actividad frente al total de defectos encontrados (¿qué actividades encuentran la mayor cantidad de defectos?)
- Estimaciones programadas versus reales (¿haremos las fechas y qué tan bien estimamos?)
- Personas en el proyecto, planificadas versus reales por semana o mes (¿tenemos las personas que necesitamos cuando las necesitamos?)
- Cambios de requisitos mayores y menores (¿sabemos lo que tenemos que hacer y cambia?)

P87. ¿Cómo analiza los resultados de sus pruebas? ¿Qué métricas intentas proporcionar?
Respuesta: Los resultados de las pruebas se analizan para identificar las principales causas del defecto y cuál es la fase que ha introducido la mayoría de los defectos. Esto se puede lograr mediante el análisis causa/efecto o el análisis de Pareto. El análisis de los resultados de las pruebas puede proporcionar varias matrices de prueba. Donde se miden matrices para cuantificar s/w, recursos de desarrollo de s/w y proceso de desarrollo de s/w. Algunas matrices que podemos proporcionar son:

- Densidad de defectos: número total de defectos reportados durante las pruebas/tamaño del proyecto Eficacia de la prueba'/(t+uat) donde t: número total de defectos registrados durante las pruebas y UAT: número total de defectos registrados durante las pruebas de aceptación de uso.
- Eficiencia de eliminación de defectos (DRE): (número total de defectos eliminados / número total de defectos inyectados)*100

P88. ¿Cómo se realizan las pruebas de regresión?
Respuesta: Las pruebas de regresión se realizan tanto de forma manual como automática. Las herramientas automáticas se utilizan principalmente para las pruebas de regresión, ya que se centran principalmente en probar repetidamente la misma aplicación para detectar los cambios por los que pasó la aplicación para la nueva funcionalidad, después de corregir los errores anteriores, cualquier nuevo cambio en el diseño, etc. ejecutando los casos de prueba, que ejecutamos para encontrar los defectos. Siempre que se produzca algún cambio en la Aplicación, debemos asegurarnos de que la funcionalidad anterior siga estando disponible sin interrupciones. Por esta razón, se deben realizar las pruebas de regresión en la aplicación ejecutando los casos de prueba escritos previamente.

P89. Descríbame cuándo consideraría emplear un análisis modal de fallo y efecto.
Respuesta: Método de calidad que permite la identificación y prevención de errores de proceso o producto antes de que ocurran. El análisis de modos y efectos de fallas (FMEA) es un enfoque disciplinado que se utiliza para identificar posibles fallas de un producto o servicio y luego determinar la frecuencia y el impacto de la falla.: FMEA (Análisis de modos y efectos de fallas) es una herramienta, técnica y calidad proactiva Método que permite la identificación y prevención de errores de proceso o producto antes de que ocurran. El análisis de modos y efectos de fallas (FMEA) es un enfoque disciplinado que se utiliza para identificar posibles fallas de un producto o servicio y luego determinar la frecuencia y el impacto de la falla.

P90. Definir Verificación y Validación. Explique las diferencias entre los dos.
Respuesta:

- **Verificación**: evaluación realizada al final de una fase para determinar que se han cumplido los requisitos establecidos durante la fase anterior. Generalmente, la verificación se refiere a la actividad general de evaluación de software, incluida la revisión, inspección, verificación y auditoría.
- **Validación**: - El proceso de evaluación del software al final del proceso de desarrollo para garantizar el cumplimiento de los requisitos. La validación generalmente implica pruebas reales y se lleva a cabo despuésla verificación está completa.

También se puede definir con las siguientes preguntas:

- **Verificación**: ¿Estamos construyendo el producto correctamente?
- **Validación**: ¿Estamos construyendo el producto/sistema correcto?

P91. ¿Qué criterios utiliza al determinar cuándo automatizar una prueba o no?
Respuesta: El tiempo y el presupuesto son factores clave para determinar si la prueba se realiza de forma manual o se puede automatizar. Aparte de eso, la automatización es necesaria para áreas como funcional, regresión, carga e interfaz de usuario para obtener resultados precisos.

P92. ¿Qué te gustaría hacer dentro de cinco años?
Respuesta: Me gustaría ocupar un puesto directivo, idealmente trabajando en estrecha colaboración con clientes externos. He trabajado en puestos de atención al cliente durante más de dos años y disfruto el desafío de mantener al cliente satisfecho. Creo que es algo en lo que soy bueno. También me gustaría asumir responsabilidades adicionales dentro de esta área y posiblemente de otras áreas como. Finalmente, me gustaría estar en el camino profesional correcto para convertirme eventualmente en Gerente Senior dentro de la empresa. Soy muy consciente de que son objetivos ambiciosos, sin embargo siento que con trabajo duro y dedicación son bastante alcanzables.

P93. ¿Qué es IEEE? ¿Por qué es importante?
Respuesta: "Instituto de Ingenieros Eléctricos y Electrónicos" Organización de ingenieros, científicos y estudiantes involucrados en campos eléctricos, electrónicos y afines. También funciona como editorial y organismo normativo.

P94. ¿Cuál es el papel del control de calidad en una empresa que produce software?
Respuesta: El papel del control de calidad en la empresa es producir un software de calidad y garantizar que cumpla con todos los requisitos de sus clientes antes de entregar el producto.

P95. ¿Cómo construirías un equipo de prueba?
Respuesta: Para formar un equipo de prueba se necesitan varios factores para juzgar. En primer lugar, hay que considerar la complejidad de la aplicación o proyecto que se va a probar. Próxima prueba, tiempo asignado para realizar los niveles de prueba. Con todos estos parámetros en mente, debe decidir las habilidades y el nivel de experiencia de sus evaluadores y cuántos evaluadores.

P96. En una aplicación actualmente en producción, se está modificando un módulo de código. ¿Es necesario volver a probar toda la aplicación o basta con probar la funcionalidad asociada con ese módulo?
Respuesta: Depende de la funcionalidad relacionada con ese módulo. Necesitamos verificar si ese módulo está interrelacionado con otros módulos. Si está relacionado con otros módulos, también debemos probar los módulos relacionados. De lo contrario, si se trata de un módulo independiente, no es necesario probar otros módulos.

P97. ¿Qué son las normas ISO? ¿Por qué son importantes?
Respuesta: ISO 9000 especifica los requisitos para un Sistema de Gestión de Calidad que supervisa la producción de un producto o servicio. No es un estándar para garantizar que un producto o servicio sea de calidad; más bien, da fe del proceso de producción y de cómo se gestionará y revisará.

P98. ¿Qué es el Método de Desarrollo en Cascada y estás de acuerdo con todos los pasos?
Respuesta: El enfoque en cascada es un enfoque tradicional para el desarrollo de software y agua. Esto funcionará porque el proyecto es pequeño (no complejo). Los proyectos en tiempo real necesitan una metodología en espiral como SDLC. Algunos desarrollos basados en productos pueden seguir la cascada, si no son complejos.
El costo de producción es menor si seguimos el método en cascada.

P99. ¿Qué diferencia existe entre error y fallo? ¿Qué es la confiabilidad en las pruebas?
Respuesta:

- **Error:** una acción humana que produce un resultado incorrecto.
- **Fallo:** una manifestación de un error en el software o una desviación del software de su entrega o servicio esperado.
- **Confiabilidad:** la probabilidad de que el software no cause el fallo del sistema durante un tiempo específico bajo condiciones específicas.

P100. ¿Por qué son necesarias las pruebas?
Respuesta: Las pruebas son necesarias porque es probable que el software tenga fallas y es mejor (más barato, más rápido y más conveniente) encontrar y eliminar estas fallas antes de ponerlo en funcionamiento. Las fallas que ocurren durante la operación en vivo son mucho más costosas de abordar que las fallas que ocurren durante las pruebas antes del lanzamiento del software. Por supuesto, otras consecuencias de que un sistema falle durante el funcionamiento en vivo incluyen la posibilidad de que los clientes demanden al proveedor de software.
Las pruebas también son necesarias para que podamos conocer la confiabilidad del software (es decir, la probabilidad de que falle dentro de un tiempo específico y en condiciones específicas).

P101. ¿Qué son las pruebas UAT? ¿Cuándo se debe hacer?
Respuesta: UAT significa 'Prueba de aceptación del usuario'. Esta prueba se lleva a cabo desde la perspectiva del usuario y generalmente se realiza antes de un lanzamiento.

UAT significa Prueba de aceptación del usuario. Lo realizan los usuarios finales junto con los evaluadores para validar la funcionalidad de la aplicación. También se denomina prueba de preproducción.

P102. ¿Cómo encontrar que las herramientas funcionan bien con su sistema existente?
Respuesta: Creo que necesitamos hacer una investigación de mercado sobre varias herramientas dependiendo del tipo de aplicación que estemos probando. Digamos que estamos probando una aplicación hecha en VB con una base de datos Oracle y luego Win runner dará buenos resultados. Pero en algunos casos puede que no sea así, digamos que su aplicación utiliza una gran cantidad de Grids y módulos de terceros que se han integrado en la aplicación. Entonces depende del tipo de aplicación que estés probando.
También necesitamos saber qué tipo de pruebas se realizarán. Si necesita probar el rendimiento, no puede utilizar una herramienta de grabación y reproducción, necesita una herramienta de prueba de rendimiento como Load Runner.

P103. ¿Cuál es la diferencia entre una estrategia de prueba y un plan de prueba?
Respuesta:

- **Plan de prueba:** es un plan para las pruebas. Define alcance, enfoque y entorno
- **Estrategia de prueba:** una estrategia de prueba no es un documento. Es un marco para tomar decisiones sobre el valor de las pruebas.

P104. ¿Qué son los escenarios en términos de pruebas?
Respuesta: Escenario significa desarrollo.
Definimos escenario mediante la siguiente definición: Conjunto de casos de prueba que garantizan que los flujos de procesos de negocio se prueben de un extremo a otro. Pueden ser pruebas independientes o una serie de pruebas que se suceden, cada una de las cuales depende del resultado de la anterior. Los términos escenario de prueba y caso de prueba suelen utilizarse como sinónimos.

P105. ¿Explique las diferencias entre las pruebas de caja blanca, caja gris y caja negra?
Respuesta:

- **Prueba de caja negra:** Las pruebas se basan en requisitos y funcionalidad. No se basa en ningún conocimiento de diseño o código interno.
- **Prueba de caja blanca:** Las pruebas se basan en la cobertura de declaraciones de código, ramas, rutas y condiciones. Basado en el conocimiento de la lógica interna del código de una aplicación.
- **Prueba de caja gris:** Una combinación de metodologías de prueba de caja blanca y negra, que prueba una pieza de software según sus especificaciones pero utilizando cierto conocimiento de su funcionamiento interno.

P106. ¿Cómo desempeñan las pruebas unitarias un papel en el ciclo de vida del desarrollo/software?
Respuesta: Podemos detectar errores simples como GUI y pequeños errores funcionales durante las pruebas unitarias. Esto reduce el tiempo de prueba. En general, esto ahorra tiempo en el proyecto. Si el desarrollador no detecta este tipo de errores, esto pasará a la parte de prueba de integración y si un evaluador lo detecta, esto debe pasar por un ciclo de vida de error y consume mucho tiempo.

P107. ¿Qué te hizo elegir las pruebas en lugar de otra carrera?
Respuesta: Las pruebas son un aspecto muy importante en el ciclo de vida del desarrollo de software. Me gusta ser parte del equipo responsable de la calidad de la aplicación que se entrega. Además, el

control de calidad tiene amplias oportunidades y un gran alcance para aprender diversas tecnologías. Y por supuesto tiene muchas más oportunidades que el Desarrollo.