**AUTOEVALUACIÓN UT3.-PRUEBAS DE SOFTWARE**

1. Relaciona los siguientes conceptos con su definición:

|  |  |  |
| --- | --- | --- |
| Concepto |  | Definición |
| Pruebas |  | Proceso de evaluación del sistema o de uno de sus componentes, para determinar si satisface los requisitos especificados. |
| Verificación |  | Proceso que permite verificar y revelar la calidad de un producto software. |
| Validación |  | Comprobación que un sistema o parte de un sistema, cumple con las condiciones impuestas |

1. Relaciona el tipo de prueba con sus características:

|  |  |  |
| --- | --- | --- |
| Tipos de prueba |  | Características |
| De unidad |  | O de caja negra se centra en las funciones, entradas y salidas que recibe y produce un módulo o función concreta. |
| De carga |  | O de caja blanca se centra en la estructura interna del programa (analiza los caminos de ejecución). |
| Funcional |  | Se deben llevar a cabo cada vez que se hace un cambio en el sistema, tanto para corregir un error(correctivo), como para realizar una mejora (evolutivo). |
| Estructural |  | Probar el correcto funcionamiento de un módulo de código |
| De regresión |  | Número de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga. |
| De estrés |  | Se utiliza normalmente para romper la aplicación |

1. Relaciona los criterios de cobertura que se sigue en las **pruebas estructurales**:

|  |  |  |
| --- | --- | --- |
| Cobertura |  | Características |
| De sentencias |  | Establece que se debe ejecutar al menos una vez cada secuencia de sentencias encadenadas, desde la sentencia inicial del programa, hasta su sentencia final. |
| De decisiones |  | Se trata de crear los suficientes casos de prueba para que cada elemento de una condición, se evalúe al menos una vez a falso y otra a verdadero. |
| De condiciones |  | Se han de generar casos de pruebas suficientes para que cada instrucción del programa sea ejecutada, al menos, una vez. |
| De caminos |  | Se trata de crear los suficientes casos de prueba para que cada opción resultado de una prueba lógica del programa, se evalúe al menos una vez a cierto y otra a falso. |

1. ¿En qué momento se suelen realizar las pruebas Beta Test?
2. Antes de las pruebas de integración.
3. En el mismo entorno productivo de la aplicación, una vez instalada.
4. Después de realizar las pruebas unitarias.
5. A lo largo de la etapa de mantenimiento.
6. La realización de pruebas \_\_\_\_\_\_\_\_\_\_\_nos permite detectar errores de cada parte del programa por separado.
   1. Semánticas.
   2. De integración.
   3. Funcionales.
   4. Unitarias.
7. La prueba de software:
   1. Sólo sirve para verificar el sistema.
   2. Sirve para verificar y validar el sistema.
   3. Sólo sirve para validar el sistema.
   4. Su realización es opcional.
8. En la planificación de pruebas.
   1. Se depura el programa.
   2. Se realiza la documentación de las pruebas.
   3. Se establecen puntos de ruptura en el código.
   4. Se diseñan los tipos de prueba y los casos de prueba.

.

1. La herramienta de prueba unitaria más extendida en Java es:
   1. SimpleTest.
   2. JUnit.
   3. NUnit.
   4. MOQ
2. La regresión es:
   1. un tipo de prueba de validación.
   2. un proceso que se realiza cuando se produce un cambio en el código.
   3. un paso necesario en la depuración del programa.
   4. un proceso asociado a la fase de diseño del proyecto.
3. Un caso de prueba.
4. Es cada fase de la planificación de pruebas.
5. Es cada elemento que interviene en la depuración.
6. Es diseñado con la colaboración del cliente.
7. Se diseña intentando que la probabilidad de detección de errores sea máxima.
8. ¿Cómo se llama la prueba que comprueba el cumplimiento de los requisitos funcionales?
9. Regresión.
10. Validación.
11. Integración
12. Sistema.
13. El objetivo del cubrimiento
    1. es obtener casos de prueba representativos.
    2. comprobar que todos los caminos se pueden ejecutar.
    3. establecer casos de prueba con valores en el límite del rango.
    4. establecer clases de equivalencia que disminuyan el número prueba.
14. Las clases de equivalencia.
    1. Nos ayudan a diseñar casos de prueba con valores límite.
    2. Nos permiten validar el sistema.
    3. Son herramientas de depuración.
    4. Nos permite crear casos de prueba representativos de un conjunto de valores posibles.
15. ¿Qué herramienta de automatización de pruebas no es para Java?
    1. JUnit.
    2. FoxUnit.
    3. TestNG.
    4. JTiger.
16. ¿En qué momento se suelen realizar las pruebas de la unidad?
    1. Antes de las pruebas de integración.
    2. En el mismo entorno productivo de la aplicación, una vez instalada.
    3. Después de realizar las pruebas unitarias.
    4. A lo largo de la etapa de mantenimiento.
17. Las pruebas funcionales.
    1. son pruebas de caja negra.
    2. son pruebas de caja blanca
    3. analizan y prueban directamente el código de la aplicación.
    4. todas las respuestas anteriores son falsas.
18. A partir de la especificación funcional en el que se especifique que el código del banco sea blanco o un número de tres dígitos y que el primero de ellos tiene que ser mayor que 1. El código de banco <100, corresponde a :
    1. A una clase de equivalencia válida.
    2. A una clase de equivalencia no válida.
    3. A un valor límite.
    4. Ninguna de las anteriores.
19. Dado el siguiente diagrama de flujo

1

SI

2

NO

3

SI

NO

4

4

5

6

Indicar cuál es su complejidad ciclomática:

1. V(G)= 2.
2. V(G)=3.
3. V(G)=4.
4. Ninguna de las anteriores.
5. Si se desea hacer pruebas de carga de una aplicación web, ¿cuál de las siguientes herramientas utilizarías?
   1. Jtiger.
   2. JMeter.
   3. Las dos anteriores.
   4. Ninguna de las anteriores respuestas es correcta.
6. Si se desea depurar un código en NetBeans, utilizaremos la tecla F7 para:
   1. Ejecutar el código hasta el siguiente punto de ruptura.
   2. Ejecutar el código paso a paso y entrar en el siguiente método.
   3. Las dos anteriores son correctas.
   4. Ninguna de las anteriores es correcta.