PRUEBAS

Las pruebas de software son un conjunto de procesos con los que se pretende probar un sistema o aplicación en diferentes momentos para comprobar su correcto funcionamiento. Este tipo de pruebas abarca cualquier estadio del desarrollo del sistema, desde su creación hasta su puesta en producción. Lo interesante de las pruebas es que se puedan ejecutar de manera automática, para determinar en cualquier momento si tenemos una aplicación estable o si, por el contrario, un cambio en una parte ha afectado a otras partes sin que nos demos cuenta.

Tipos de pruebas de software

Imagina que estás desarrollando una aplicación web. Durante su proceso de creación utilizarías las **pruebas unitarias y de integración** para comprobar que todo funciona de la manera esperada internamente. Una vez que la web va cobrando forma, utilizarías las **pruebas funcionales** para poder comprobar que los requisitos de funcionamiento se están cumpliendo.

Llegados a este punto tenemos una web que cumple con el funcionamiento que se espera pero... ¿soportará el número de usuarios esperado?

Para resolver esa pregunta, recurrimos a las **pruebas de carga y de estrés** para poder asegurar que la web va a ser capaz de dar respuesta al número de usuarios esperado. De este modo es posible detectar problemas que, simplemente con el código, no se pueden ver con facilidad, y ayuda a encontrar los puntos calientes de nuestro sistema donde deberíamos mejorar.

En estos momentos ya tenemos una web que funciona y es capaz de dar soporte al número de usuarios que queremos. ¿Es usable la web? Por muy buen trabajo que se haya hecho, si la web no es usable no vamos a llegar a los usuarios de la misma. Para poder detectar esto, es el momento de hacer **pruebas de usabilidad**. Con ellas vamos a poder detectar problemas sobre la propia experiencia de usuario.

Todas estas pruebas, en los diferentes momentos del ciclo de vida de la aplicación, son pruebas de software que permiten asegurar la calidad del producto que estamos desarrollando. Pero aún podríamos comprobar más cosas: la interfaz gráfica, pruebas de extremo a extremo (End To End o E2E)...

¿Es rentable hacer pruebas de software?

Es evidente que crear pruebas **es un trabajo extra**, independientemente del momento del ciclo de vida del proyecto en el que nos encontremos. En primer lugar, hay que crear los tests y mantenerlos. A medida que el software crece y evoluciona, se producen cambios que obligan a modificar las pruebas, y eso es otro coste adicional, ya que una prueba que no funciona bien es totalmente inútil....

Pues bien, la respuesta es un rotundo... **depende**. Depende principalmente de qué tipo de prueba de software estemos hablando y de cuál sea el alcance del sistema.

Si nuestra aplicación es muy básica y vamos a tener 3 o 4 usuarios, hacer pruebas de carga o estrés seguramente no tenga ningún sentido. En cambio, ¿te imaginas que una web como por ejemplo Facebook o Twitter no las tengan? ¡Podría ser catastrófico! Un cambio en un punto clave del sistema que haga que una petición tarde un 20% más en responderse, y de repente errores de servidor saturado a miles de personas... Pero, incluso para esa aplicación básica, sería interesante que tuviese alguna prueba de integración que comprobará que sus puntos clave están funcionando en todo momento.

Las pruebas de software deben ser acordes al producto que se está desarrollando, pero **rara es la vez (por no decir nunca) donde no sean útiles**.

Hoy en día existen herramientas y *frameworks* para automatizar la gran mayoría de las pruebas de software en sus diferentes niveles, de modo que se ejecuten automáticamente, de manera periódica o con los cambios. Gracias a esto, es posible reducir al mínimo los errores cuando se realizan cambios durante el ciclo de vida de un software.

Vale, son útiles en proyectos grandes y grandes equipos como Facebook o Twitter, pero, **¿si trabajamos solos en proyectos pequeños no carecen un poco de sentido?** La realidad es que, por muy buenos que seamos, las personas nos equivocamos y no tenemos una memoria perfecta. Así que, incluso para trabajar una sola persona en proyectos pequeños, es interesante añadir pruebas que aseguren el funcionamiento. **Al cabo de unas pocas semanas, ¿eres capaz de recordar el 100% de código que has desarrollado y la utilidad de cada línea o método?** Lo que puede parecer un cambio menor que no va a afectar en nada al sistema, puede tener un alcance imprevisto al modificarlo.

POR ELLO NOS VAMOS A CENTRAR EN LAS TRES PRIMERAS, LAS MÁS IMPORTANTES.

Los desarrolladores, como mínimo, debemos desarrollar 3 tipos de pruebas sobre el código:

- Pruebas unitarias
- Pruebas de integración
- Pruebas funcionales

Pruebas unitarias

El primero de los grupos, las **pruebas unitarias**, es como se conoce a todas esas **pruebas que solo comprueban una funcionalidad específica de una única clase**. A este tipo de pruebas "les da igual" el comportamiento del resto, ya que la gran mayoría de las veces reciben un objeto que simula el comportamiento externo a esa clase, necesario para funcionar.

La finalidad última de las pruebas unitarias es comprobar funcionalidades muy concretas de cada clase.

Son pruebas que no deberían costarnos más de 5 minutos escribir, y de las cuales **vamos a tener tantas como sea necesario para probar el 100%** (o lo máximo posible al menos) de los posibles casos que estén contemplados en el código.

Por ejemplo, el hecho de comprobar que un método lanza una excepción en una condición concreta es un caso claro de prueba unitaria.

De este tipo de pruebas se espera que sean rapidísimas de ejecutar, ya que en un proyecto grande habitualmente habrá cientos (¡o incluso miles!) de éstas.

Pruebas de integración

El segundo de los grupos, las **pruebas de integración**, es como conocemos a todas esas pruebas que **verifican que las relaciones existentes entre las diferentes clases y servicios funcionan correctamente**.

Este tipo de pruebas lo que busca es encontrar todos esos problemas que surgen al mezclar las diferentes capas de nuestra aplicación.

Por ejemplo, si trabajamos con una base de datos, en una prueba de integración utilizaremos un motor de base de datos real y no uno en memoria o simulado. De este modo validaremos la correcta integración de nuestro sistema con esa base de datos.

En este tipo de pruebas es posible utilizar simulaciones para algunos niveles que todavía no se pueden probar, como por ejemplo un servicio de terceros.

Suelen ser pruebas más costosas de desarrollar y ejecutar que las pruebas unitarias ya que en cada una de ellas se deben integrar varios puntos del sistema para asegurar que funcionan correctamente en conjunto.

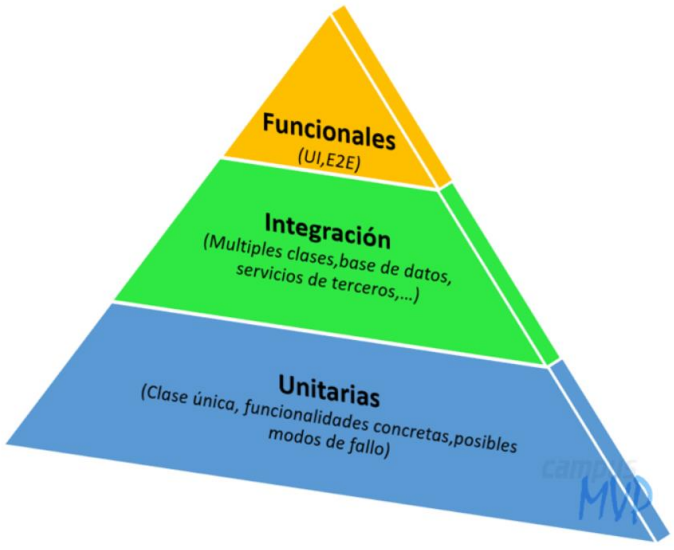
Pruebas funcionales

Las **pruebas funcionales** verifican el comportamiento del sistema para afirmar que **se cumple con la funcionalidad completa**. Se les suele denominar también **pruebas End-To-End** o E2E.

Este tipo de pruebas se realizan con los requisitos de funcionamiento en la mano, y **no se centran en los pormenores del sistema**, que ya deberían estar probados con los dos grupos anteriores.

Por lo general, son pruebas que requieren de más esfuerzo que las anteriores, ya que en ellas debemos probar el funcionamiento completo del sistema en base a los requisitos existentes. Por esto debemos utilizar el mismo tipo de sistemas que utilizará el código en producción.

Además, es habitual que se hagan pruebas específicas para evaluar el rendimiento y comprobar que está dentro de los parámetros deseados.



En general **siempre debe haber muchas más pruebas unitarias que de integración y muchas más que test funcionales**.

En resumen

Como conclusión de todo lo anterior podríamos sacar las siguientes enseñanzas:

- Las pruebas de software son un conjunto de técnicas que nos permiten asegurar la calidad del producto que estamos desarrollando en sus diferentes etapas del ciclo de vida.
- Aunque tienen un coste de desarrollo y mantenimiento extra, son especialmente útiles para facilitar la calidad del software a medio y largo plazo.
- El tipo y cantidad de pruebas tiene que ser acorde al producto que estamos desarrollando: no es lo mismo un producto grande que una aplicación web para anotar tareas.
- Todo software debería tener algún tipo de prueba para asegurar su calidad independientemente del tamaño del mismo o del número de personas involucradas en su desarrollo.