Testing

26 de Abril de 2018 a las 02:17

Seguimos con el monográfico sobre programación con un nuevo audio del podcast **Tomando Un Café** y un artículo en el blog. En este caso, hablaremos sobre *testing*, desde un punto de vista mas teórico y sin enfocarlo a un lenguaje de programación. El objetivo es dar conocer las cualidades del *testing* y las ventajas que presenta, tanto para aquellos programadores que no lo utilizan en su flujo de trabajo como para aquellos que estan empezando a programar y lo puedan incorporar en los futuros desarrollos de aplicaciones que comienzen.

Este artículo se puede considerar como una introducción para adquirir unas primeras nociones sobre *testing*, si le resulta interesante puede profundizar y empezar a utilizarlo en sus desarrollos.

Introducción

Cuando desarrollamos una aplicación, cualquier funcionalidad debe pasar un proceso de pruebas donde comprobamos que esa funcionalidad funciona, realiza aquello para lo que fue diseñada. Si el proceso de prueba es exitoso, el desarrollo de la funcionadad ha finalizado y podemos empezar con el desarrollo de otra funcionalidad, si el proceso falla debemos solucionarlo hasta que conseguir el exito de la prueba.

El proceso de pruebas es el que indica cuando una aplicación está lista para ejecutarse en un entorno de produción, eso significa que todas sus funcionalidades deben ser probadas y el resultado debe ser satisfactorio.

Hay que destacar la importancia del proceso de pruebas, porque el posible exito o fracaso de una aplicación puede ser por lo estricto que se ha realizado el proceso de prueba. El desarrollador que no ha ejecutado las pruebas de una forma estrictas puede que no haya detectado determinados errores en su aplicación y el funcionamiento no sea óptimo.

Hay que tener en cuenta que el proceso de prueba detecta un determinado tipo de errores, comprueba que la aplicación hace lo que desarrollador habia diseñado. Pero no detecta otro tipo de errores como pueden ser de sintaxis, compilación o rendimiento.

Pero, ¿Cómo realizar ese proceso de pruebas?, existen dos tipos de enfoques a la hora de afrontar este proceso.

Tipos de pruebas

Como he comentado anteriormente, todas las partes de un desarrollo debe ser probadas para saber si funcionan correctamente. Para realizar las pruebas que necesita un desarrollo, existen dos posibles enfoques que se diferencian en como se ejecutan las pruebas.

Pruebas manuales

El primer enfoque para probar una aplicación es la mas lógica y simple, el desarrollador ejecuta la aplicación y compueba que esta parte funciona correctamente de una forma manual, realizando determinadas opciones como si fuera un usuario de la aplicación. Realizando todas las acciones necesarias para probarlo: introducir datos de prueba, ejecutar acciones, click con el ratón, usar el teclado y cualquier acción requerida.

Este tipo de pruebas de forma manual tiene una serie de ventajas y desventajas.

Ventajas

- No requiere tener un conocimiento extra, como saber programación.
- Fácil de realizar, lo puede realizar el desarrollador o el cliente, solo es necesario conocer el funcionamiento de la aplicación.

Desventajas

- Es un proceso tedioso y requiere mucho tiempo.
- Solo se suele probar la funcionalidad mas reciente.
- Requiere intervención humana.
- No son pruebas muy estrictas, al tener intervención humana.

Las pruebas manuales la usan muchos desarrolladores porque son la primera opción que se piensa y son fáciles de realizar. Pero como se puede ver, tiene desventajas y bastante graves, en desarrollos de tamaño medio y grande serán un factor a resolver.

Pruebas automáticas

El segundo enfoque, se concentra en resolver las desventajas mediante un proceso de automatización. Este proceso también se suele llamar *Testing*, que se puede definir como una serie de técnicas y herramientas que permiten automatizar todo el proceso de pruebas.

El *Testing* y la automatización implica una serie ventajas como son:

- Reduce el tiempo de prueba.
- Todas las pruebas son ejecutadas en todo el ciclo de desarrollo.
- No requiere intervención humana.
- Permite integrarse en el ciclo desarrallo de una aplicación.

Los lenguajes de programación suelen incluir librerías para desarrollar test(pruebas), tanto como parte del lenguaje como utilizando librerías externas. Se pueden integrar fácilmente en el flujo de trabajo del desarrollador, utilizandolas en conjunción con otras

herramientas como *servidores de integración continua*.

Para desarrrollar test debe tenerse en cuenta una serie factores:

- Es necesario tener conocimientos en programación, ya que las pruebas son desarrolladas en un lenguaje de programación.
- Cuando se desarrolla un test se requiere un tiempo de desarrollo, aunque solo de forma inicial. Después esa test será ejecutado todas las veces que sea necesario de forma automática.
- Los test no puede dejar rastros en la aplicación. Esto significa que cualquier dato que requiera un test deben ser cargados al comienzo del test y borrados cuando finalize el test, las librerías de *testing* proporcionan diferentes métodos para realizar esta tarea.

Una cosa a tener en cuenta, es que el objetivo del proceso de *testing* no es buscar errores en la aplicación sino comprobar que la aplicación funciona como se diseño. De hecho, un test puede fallar y la aplicación no tenga un error de ejecución.

Tipos de test

Cuando se realiza *testing* se debe desarrollar una *bateria de test* para comprobar el funcionamiento de una aplicación. Esta batería esta compuesta por un conjunto de test que podran ser de diferentes tipos, los cuales comprobaran diversas partes del la aplicación.

Cada tipo de test utilizará diferentes herramientas o librerías, cada tipo tipo tendrá un objetivo y funcionalidad diferente.

Test unitarios

Es el tipo de test que mas utilizarás y su objetivo es comprobar el funcionamiento de una unidad de código, que puede ser una función, método o una clase. Este tipo de pruebas nos permite comprobar el funcionamiento de de diferentes partes de código de una forma separada. Cada test unitario tiene un alcance limitado y acotado.

Para diseñar unos test unitarios de calidad debemos tener en cuenta los siguientes requisitos:

- No pueden solicitar datos por pantalla(automátización): se utilizarán datos de prueba, mas adelantes se tratara este tema, que se cargaran al comienzo de los test.
- Deben cubrir la mayor parte de código, aunque llegar al 100% del código no es algo estrictamente necesario. Para conocer el procentaje de código que esta cubierto por un test se utiliza el termino de *cobertura*.
- Deben ser reutilizables y que puedan ejecutar todas las veces que se consideren oportunos.

- Los test unitarios deben ser independientes , la ejecución de un test no puede afectar a otro test.

- Los test unitarios son código y se deben prestar la misma importancia que el código de la aplicación.

Este tipo de test suelen ser muy utilizados porque son relativamente fáciles de implementar y gestionar, también suelen ser bastante numerosos para abarcar la mayor cantidad de código de la aplicación.

El funcionamiento de un test unitario es muy sencillo, se realiza mediante una afirmación (assert) donde le proporcionamos una datos de entrada y un resultado correcto. Con los datos proporcionados, el test se ejecuta y comprueba que el resultado propocionado es igual al proporcionado por el código del aplicación, si es igual el test ha sido un exito y si el resultado no es igual el test ha fallado.

Si el código donde se ejecuta el test tiene dependencias o requiere una acceso exterior, entonces debe utilizarse otros componentes como son *stub* o *mocks* , porque los test unitarios no pueden interactuar con la aplicación directamente, como acceso a un BD o enviar un correo, sino que debe simular la acción.

Test de integración

Si ya tenemos una bateria de test unitarios que abarcan gran parte de nuestro código de forma independiente. Otro tipo de prueba sería comprobar la interacción de dos o mas partes de al aplicación (clases, métodos, funciones...) o comprobar el desarrollo completo. Este tipo de test nos permite conocer la aplicación por completo y comprobar que cada una de las partes de integran correctamente.

Los test de integración serán bastante menor que los test unitarios, pero son mas difíciles de crear y gestionar, aunque proporcionan información muy importante del funcionamiento de la aplicación.

Test funcionales

En este tipo de test comprueba el correcto funcionamiento de la aplicación, mediante la programación de una serie de acciones que se realizan de forma automática.

Para realizar este tipo de test necesita de alguna herramienta que permite definir un flujo de acciones para una aplicación. Por ejemplo, si realizamos un test funcional para una aplicación web , podemos programar una serie de acciones, como insertar una usuario, listar usuario y borrar usuarios. De esta forma vemos si esa acciones se han realizado de forma correcta.

Datos de prueba

Cualquier prueba que se realize a una aplicación necesitará una serie de datos, cuanto mas numeroso mejor pruebas se podrán realizar, porque podremos tener un amplio abanico de opciones que probar.

Estos datos de prueba pueden generarse de forma manual, accedemos a la aplicación

y creamos esos datos, pero este proceso es muy tedioso y lento. Existen librerías que automatizan este proceso, solo tenemos que indicar el número de datos que queremos generar , el idioma y el tipo de datos. A esto último se le llaman *proveedores*, que permite indicar a que ambito pertenecen los datos. Por ejemplo, podemos definir un ámbito para clientes, bancos, geográficos, libros...etc, incluso podemos definir un ámbito propio.

Una vez generados los datos deseados podemos utilizarlos en nuestra aplicación, con cualquier tipo de test.

Conclusión

El proceso de testing permite ahorrar mucho tiempo y ser mas eficiente a la hora de probar una aplicación,aunque requiere un tiempo de desarrollo y aprendizaje, este tiempo está bien invertido por el ahorro que conlleva en el ciclo de desarrollo.

Conviene empezar a realizar testing en el mismo desarrollo, incluso conforme se aprende a programar, para integrarlo de una forma mas fácil en nuestro flujo de trabajo.