**React**

**Introducción a las pruebas unitarias y de integración.**

Documentación: <https://es.reactjs.org/>

<https://es.reactjs.org/docs/getting-started.html>

**¿Qué son y para qué sirven las pruebas unitarias?**

Primero que nada, las pruebas unitarias no son una perdida de tiempo como muchas personas piensan que así lo son. Por ejemplo: ¿Tu manejarías un auto que jamás a sido probado por su fabricante? Lo más seguro es que no, inclusive si el fabricante te regala el auto, tu sospecharías que algo anda mal no? Entonces mas o menos por ahí va el asunto de las pruebas unitarias en nuestra aplicación.

Existen 2 tipos principales de pruebas:

1. **Las unitarias:** Estas pruebas están enfocadas en pequeñas funcionalidades de nuestra aplicación, en pequeñas piezas que hay que probar.
2. **Las de integración:** Estas pruebas están enfocadas en como reaccionan varias piezas en conjunto de nuestra aplicación.

Un ejemplo visual de las pruebas, volviendo al caso del auto, sería que: Podríamos pensar que una **prueba unitaria** sería la llanta del auto como si fuera una pieza sencilla, pensemos en esta llanta como un componente aislado que será nuestra prueba unitaria, nosotros probaríamos que la llanta ande bien, que gire correctamente, no tenga deformaciones. Al realizar estas pruebas podríamos detectar si la llanta no tiene la resistencia suficiente, que existan valores que no funcionen en esa llanta, se destruye muy rápido, etc. Esto sería una prueba unitaria.

Y una **prueba de integración** sería que nosotros tomemos 4 llantas y probaríamos las llantas en conjunto con la carrocería del auto, motor, etc y ver si se integran correctamente y trabajan como nosotros esperamos. Esto sería una prueba de integración.

**¿Cuáles son sus características de las pruebas unitarias?**

Cuando hablamos de las pruebas en programación, sus características son las siguiente:

* Fáciles de escribir.
* Fáciles de leer.
* Confiables.
* Rápidas.
* Principalmente unitarias.

**¿Cuáles son los mitos de las pruebas unitarias?**

Existen ciertos mitos que rodean a las pruebas, los cuales son los siguientes:

1. **Hacen que nuestra aplicación no tenga errores.** No es cierto: La aplicación y las pruebas son escritas por humanos y por ende somos susceptibles a cometer errores y equivocarnos.
2. **Las pruebas no pueden fallar.** No es cierto: Las pruebas pueden dar falsos positivos o falsos negativos. Puede que algo funcione pero en la prueba no o puede que la prueba funcione y que la aplicación no funcione.
3. **Hacen más lenta mi aplicación.** No es cierto: Las pruebas solamente son locales y corren en la parte del desarrollo y no llegan a la parte de producción.
4. **Son una pérdida de tiempo**, esto en parte si lo es, solo si no se hacen las pruebas correctamente o con sentido. Por ejemplo: Si nosotros importamos una librería X, y queremos probarla, se supone que el que creo esa librería X ya debió hacer las respectivas pruebas y es el quien tiene que hacer esas pruebas, nosotros no. Nosotros lo que probaríamos en nuestra aplicación es que las interacciones con esa librería X sean las que queremos.
5. **Debemos de probar todo.** Esto no es del todo correcto, ya que en la vida real a veces el tiempo es limitado y no podemos tomar tanto tiempo para realizar las pruebas, si nuestro tiempo es corto pues solo haríamos las pruebas críticas de la aplicación, pero si tenemos todo el tiempo del mundo, entonces sí podríamos probar todo en nuestra aplicación, cabe recalcar que las pruebas se programan también, entonces nos podría llevar más tiempo que la que nos tomó desarrollar nuestra aplicación.

**¿Cuáles son los lineamientos que debemos seguir de las pruebas unitarias?**

Estos pasos se aplican y son conocidos como AAA:

* Arrange (Arreglar): Es el paso donde nosotros preparamos el estado inicial y que usualmente se conoce como el sujeto a probar. Normalmente aquí se realiza lo siguiente:
  + Se inicializan variables
  + Se hacen las importaciones necesarias
  + Se prepara el ambiente
* Act (Actuar): Este paso es en donde vamos aplicar acciones o estímulos al sujeto de prueba. Normalmente aquí se realiza lo siguiente:
  + Se llaman métodos
  + Se hacen simulaciones de click’s
  + Se realizan las acciones sobre el paso anterior

La idea es que aplicamos el estímulo al sujeto de pruebas para preparar el siguiente paso que es el Assert.

* Assert (Afirmar): Este paso se resume a observar el comportamiento resultante.
  + Son los resultados esperados

Lo que es la parte del AAA son características que nosotros deberías aplicar en todas nuestras pruebas.