Pruebas de desarrollo

El proceso de creación de pruebas para el desarrollo, consta de tres tipos o procesos de pruebas distintos, los cuales se pueden definir en una pirámide, en la base de la pirámide, se tendrá a las pruebas unitarias, arriba de estas se tendrá a las pruebas de integración, y en la cima de la pirámide están las llamadas pruebas *End To End,* a medida que se va subiendo en la pirámide, se va aumentando el tiempo que cuesta desarrollar dicho tipo de pruebas.

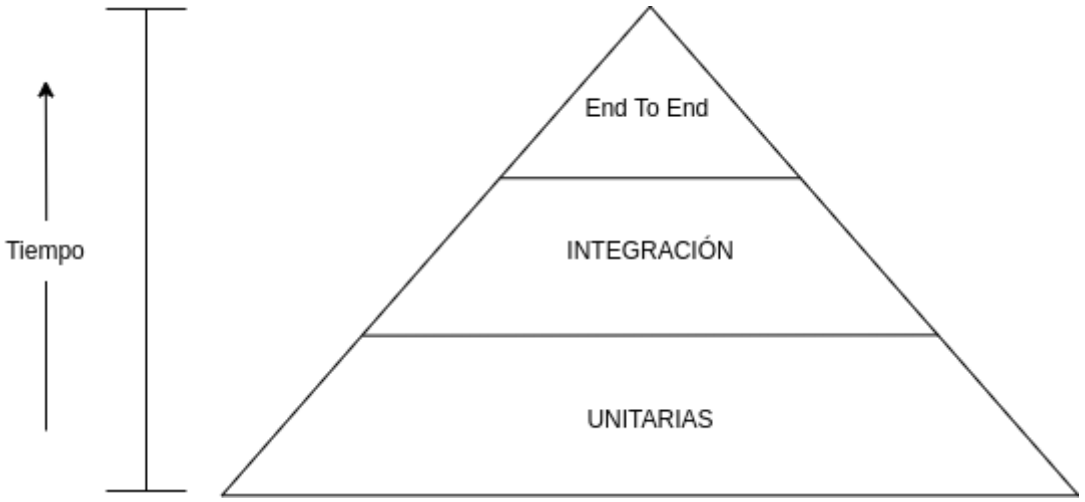

Figura 1. Pirámide de pruebas

Las pruebas *End To End,* son pruebas que se encargan de validar **todo** el comportamiento de la aplicación, por ello buscan simular el comportamiento real de un usuario, validar la integridad de los sistemas externos (como las bases de datos).

En las Pruebas de integración, se verifica que cada uno de los componentes que se han desarrollado por aparte funcionen en conjunto.

Las pruebas unitarias, por su parte, buscan verificar cada una de las partes más pequeñas de la aplicación, como por ejemplo clases o métodos, es decir, buscan probar componentes individuales antes de ser puestos en conjunto con los otros componentes que forman el desarrollo, las pruebas unitarias se caracterizan por ser:

- Rápidamente ejecutables
- Rápidamente programables (deben ser ágiles)
- Independientes de elementos externos (No dependen de la interfaz gráfica, la clase que se está probando debería cumplir con los principios SOLID)
- Independientes de otras pruebas unitarias

TDD: Test-Driven Development

En la metodología de **desarrollo guiado por las pruebas,** se busca que lo primero que se haga en el proceso de desarrollo (después del diseño) sean el desarrollo de las pruebas, esta metodología parte de la idea de que al desarrollar primero las pruebas, ya se tiene una conciencia de cuáles serán los casos que debe cumplir la funcionalidad que se desea implementar, y por ello, el desarrollo de esta es más efectivo y preciso.

El proceso de desarrollo basado en TDD, se puede simplificar en tres estados, un estado inicial donde las pruebas están siendo desarrolladas y la funcionalidad aún no lo ha sido, un segundo estado donde la funcionalidad empieza a ser desarrollada y probada con las pruebas unitarias antes concebidas, y un tercer estado donde se hacen modificaciones de las funcionalidades, bien sea porque no paso la prueba, o porque se debe agregar más funcionalidades a la clase que se prueba.

Este proceso puede ser representado en un esquema llamado RED, GREEN, YELLOW.

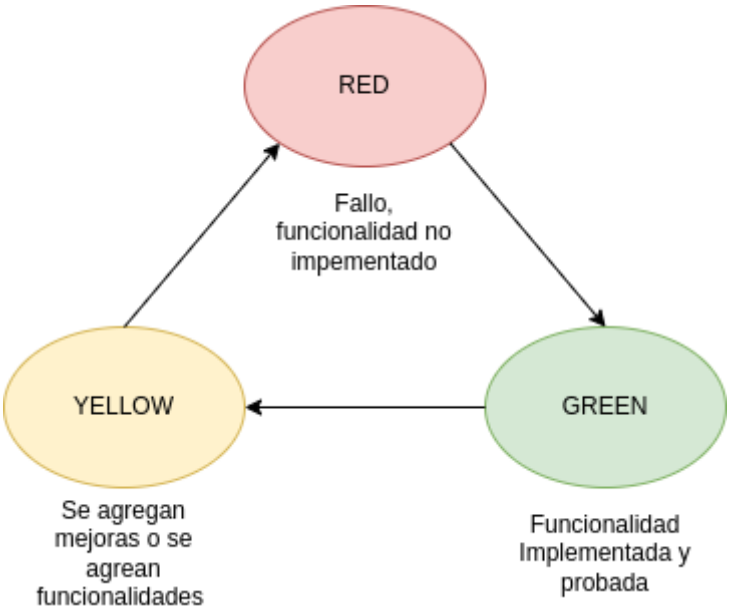

Figura 2. Esquema, RED, GREEN, YELLOW

En este esquema mostrado en la Figura 2. Partimos del estado RED, donde la prueba se está desarrollando y las funcionalidades aún no están implementadas y por ende la prueba falla, está en rojo; en el estado GREEN se hace una primera implementación de la funcionalidad, de tal forma que pasa una o varias de las pruebas unitarias planteadas, la prueba pasa a estar en verde; en el estado YELLOW, se pueden hacer cambios en el código, para agregar funcionalidades, o para corregir errores que hacen que el código no esté en verde.

Métricas

No basta solo con desarrollar un conjunto de pruebas unitarias, también en necesario validar si el código cumple con ciertas métricas que garantizan su calidad

Densidad de errores-fallos = total de fallos / total de pruebas
Confiabilidad = 1 - densidad de fallos
Completitud = casos de prueba / total funcionalidades

* Este último indicador se espera que sea un número mayor que 5 al final de la implementación del producto.

Estructura de las pruebas

La creación de las pruebas unitarias puede resumirse en tres componentes, *init, act* y *assert,* estas tres acciones consisten en:

- *init:* consiste en inicializar el escenario de prueba, el cual es el estado en que se encuentran las entidades del programa (objetos inicializados, listas con elementos, objetos que se encuentran nulos y serán creados)
- *act:* El llamado al método o funcionalidad que será probado
- *assert:* la validación del resultado esperado del punto anterior, la cual define en que estado se encontrará la prueba y cuál será el siguiente paso, recuerde:
    - GREEN: La prueba ha pasado correctamente y se puede seguir en el proceso de desarrollo de las otras pruebas o funcionalidades
    - RED: La prueba no ha pasado correctamente y es necesario corregir la funcionalidad del método pasando al estado YELLOW
    - YELLOW: se hace la modificación del método para que cumpla con las funcionalidades pedidas y para que pase las pruebas planteadas, esto puede implicar agregar validaciones, y/o extender o separar funcionalidades.