# Unit Testing

## Analogía del Automóvil

Para entender mejor a que nos referimos cuando hablamos de unit testing. En vez de hablar por el momento de software, vamos a imaginarnos la construcción de un carro.

Un carro tiene un motor, una transmisión, un sistema de frenos, refrigeración, cinturones de seguridad, faros, etc. Un automóvil es la suma de todas esas partes o subsistemas. Cada una de estas partes tienes sus propias especificaciones y requisitos que definen la funcionalidad esperada, tolerancias e interfaces con el resto del sistema.

Imaginen el punto en el cuál todos esos subsistemas son ensamblados y atornillados en una línea de montaje sin que se realice ninguna prueba previa. El conductor que va a realizar la prueba del auto se sienta y este no arranca. ¿ De que subsistema fue la culpa? Si no se realizaron pruebas independientes a cada uno de esos subsistemas y sus partes será muy difícil encontrar el problema.

Suponiendo que se pone en marcha y avanza ¿el piloto podrá probar cada una de las partes del auto? Que el sistema de refrigeración funcione a altas revoluciones o que el motor funcione si el aceite está más abajo que los niveles permisibles. Para realizar todas estas pruebas el piloto se tomará una gran cantidad de tiempo y también serían mejor realizadas por las personas especializadas que construyeron el subsistema y de una manera que el usuario final sería incapaz de hacerlo.

Lo mismo ocurre con el software, de manera similar a los autos, el software es una colección de componentes y partes. Realizar pruebas independientes a cada una de estas partes nos ayudará a depurar el API antes de que alguien intente usarla. Si a través de alguna de estas pruebas falla será muchísimo más fácil de encontrar y corregir el error en comparación a si estuviéramos probando todo el sistema, asimismo se permite ahorrar mucho más tiempo y dinero ya que es más eficiente para cualquier corregir errores antes de entrar a una fase de ensamblaje y detener a todo el equipo ante cada fallo.

Piensen en esto, ustedes preferirían conducir un automóvil cuyos frenos han sido alta y consecutivamente probados por los ingenieros especializados o confiarían en las pruebas de sistema que ha realizado el piloto de pruebas. La misma lógica debemos seguir al desarrollar un software, las pruebas no es algo que se deba realizar luego de que todas las partes han sido combinadas sino se debe realizar de manera independiente a cada parte antes de ensamblarlas en un sistema mucho más grande. Realizar estas pruebas desde el inicio y ejecutarlas constantemente cada vez que se realice algún cambio nos ayudará a a garantizar la calidad del software que las usar y a identificar problemas antes de llegar a QA o a los usuarios finales.