**Test Driven Development**

**Prework**

* Analizar qué es esta técnica de programación, sus orígenes y la lógica detrás de su funcionamiento. -Aplicar el ciclo rojo, verde, refactorización.
* Explorar la escuela Londres de TDD (de afuera hacia adentro o también conocido como mockistas).
* Explorar la escuela Chicago de TDD (de adentro hacia afuera o también conocido como clasistas).

TDD o Test-Driven Development (desarrollo dirigido por tests) es una práctica de programación que consiste en escribir primero las pruebas (generalmente unitarias), después escribir el código fuente que pase la prueba satisfactoriamente y, por último, refactorizar el código escrito.

* Paso 1: Diseñar una prueba. Antes de realizar cualquier trabajo de implementación, se especifican las condiciones que deben cumplirse. ¿Cuál es la salida o el resultado esperado?
* Paso 2: Ejecutar la prueba. Dado que aún no se ha realizado ningún trabajo para satisfacer la prueba, es razonable esperar que la prueba falle si se ejecuta. Una prueba fallida suele mostrarse como una barra roja en un entorno de desarrollo integrado. Sin embargo, no hay que suponer que esa prueba vaya a fallar.
* Paso 3: Escribir el código y volver a probar. La característica que falta debe ser implementada para que la prueba pase (verde). Hay que escribir el código suficiente para satisfacer la prueba y no más. También es importante ejecutar todas las demás pruebas para asegurarse de que la implementación no ha hecho fallar la funcionalidad existente.
* Paso 4: Refactorizar el código. Ahora que la funcionalidad está en su sitio, puede optimizarse para mejorar la legibilidad y la capacidad de mantenimiento.

El ciclo que siguen los programadores en el test-driven development también recibe el nombre de red-green-refactor y describe cada una de las fases que deben cumplirse para alcanzar una mayor eficiencia:

* Fase roja (red phase): en esta fase hay que ponerse en los zapatos del usuario, que quiere poder usar el código de forma sencilla. Se redacta, por lo tanto, un test que contenga componentes que aún no hayan sido implementados, para luego decidir qué elementos son realmente necesarios para que el código funcione.
* Fase verde (green phase): suponiendo que el test falle y se marque en rojo, se adopta entonces el papel de programador y se intenta encontrar una solución simple. Es muy importante redactar únicamente la cantidad de código que sea necesaria. El código redactado se integra luego en el código productivo, de forma que el test quede marcado en verde.
* Refactoring: en este paso, el código productivo se pasa a limpio y se perfecciona su estructura. Podría decirse que ahora se completa y reestructura de manera que resulte elegante y comprensible para los desarrolladores. Entre otras cosas, se eliminan los duplicados en el código, y se vuelve más profesional.

**Work**

* Analizar qué es esta técnica de programación, sus orígenes y la lógica detrás de su funcionamiento.
* Aplicar el ciclo rojo, verde, refactorización.
* Explorar la escuela Londres de TDD (de afuera hacia adentro o también conocido como mockistas)
* Explorar la escuela Chicago de TDD (de adentro hacia afuera o también conocido como clasistas)

**Postwork**

* Desarrollar un programa siguiendo las reglas de TDD.
* Añadir pruebas unitarias adicionales para probar su funcionalidad.
* Identificar los principales elementos de TDD.
* Identificar el impacto de la premisa de transformación elegida en el resultado final.