

TDD

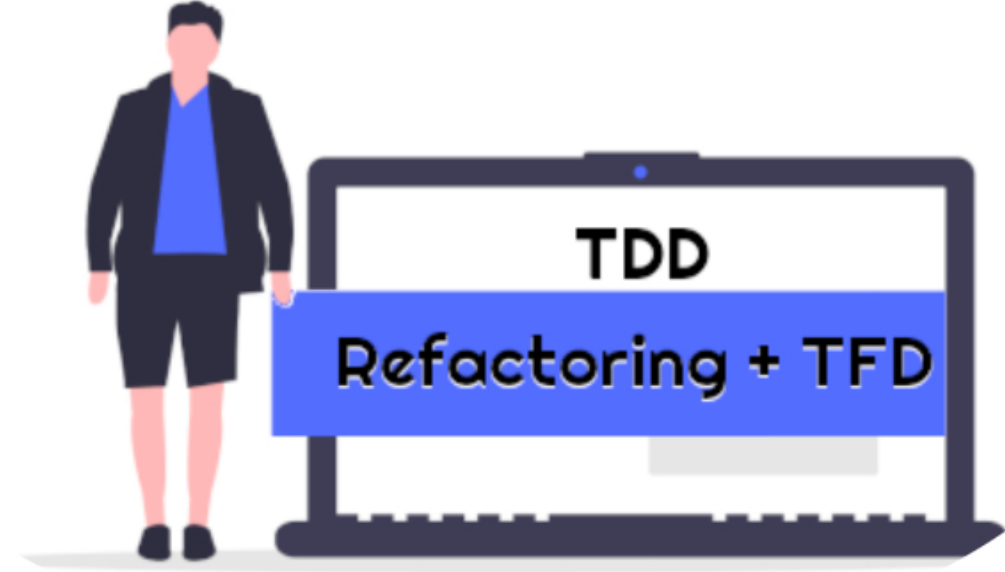
Test Driven Development

Escribe código como un profesional

Qué es TDD? (Test Driven Development)

TDD o Desarrollo guiado por pruebas en español, es un enfoque evolutivo en la ingeniería de software que combina 2 prácticas que permiten crear código de calidad:

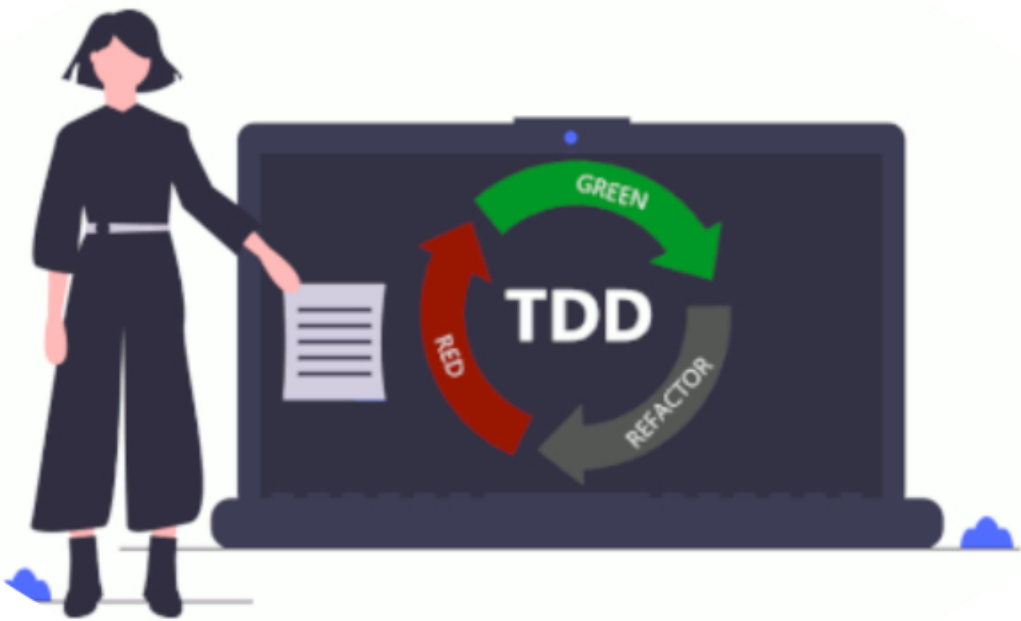
- **Escribir la prueba primero** (Test First Development - TFD): escribir la prueba antes del código de producción.
- **Refactorización** (Refactoring): mejorar la estructura del código.



Ciclo de TDD (Rojo - Verde - Refactor)

Para hacer TDD, debes cumplir un ciclo que consta de 3 partes:

- La prueba debe fallar. (**Red**: Muchas herramientas muestran los fallos de las pruebas en rojo)
- La prueba debe pasar. (**Green**: Al igual que lo anterior, las herramientas muestran las pruebas que pasan en verde)
- Se debe mejorar el código. (**Refactoring**)



Leyes de TDD (3 simples reglas)

1. No se permite escribir cualquier código de producción, a menos que sea para hacer pasar un test unitario fallido.
2. No se permite escribir más que un test unitario que sea lo suficiente para que éste falle; los errores de compilación se consideran fallos.
3. No se permite escribir cualquier código de producción mas que el que sea suficiente para hacer pasar un test unitario.



Principio F.I.R.S.T.

Al escribir una prueba, considerar siempre el principio F.I.R.S.T.:

- **Fast**: La prueba ejecuta de forma rápida.
- **Independent**: La prueba no depende de otras pruebas.
- **Repeatable**: La prueba ejecuta en cualquier entorno.
- **Self-validating**: La prueba se valida a través de un parámetro de salida verdadero o falso.
- **Timely**: La prueba es creada antes que el código de producción.



Ventajas de TDD

- Confianza
- Minimiza la cantidad de código a escribir
- Mejora el diseño del código
- Desarrollo más rápido cuando se ha dominado la práctica



Desventajas de TDD

- No es garantía de buen código
- Las pruebas requieren mantenimiento
- Es difícil de dominar, pruebas difíciles de escribir
- La curva de aprendizaje puede hacer lento el desarrollo (al principio)
- Difícil de aplicar en código legacy
- Requiere compromiso de todo el equipo



Herramientas para aplicar TDD

Actualmente existen muchas herramientas que permiten crear pruebas de forma sencilla, su uso dependerá del lenguaje de programación que se esté utilizando.

- **.Net**: [xUnit](#) and [NCrunch](#)
- **JS/TS**: [Jest](#) and [VS Code Jest extension](#)
- **Python**: [pytest](#)
- **Java**: [IntelliJ](#) and [Infinittest](#)

