TDD

Indice

- ¿Que es TDD?
- ¿Por que se usa TDD?
- Reglas de TDD
- Diagrama explicado
  - Write a falling test
  - Make the pass test
  - Refactor
- ¿Por que es difícil incorporar TDD?
- Ventajas y Desventajas

¿Que es TDD?

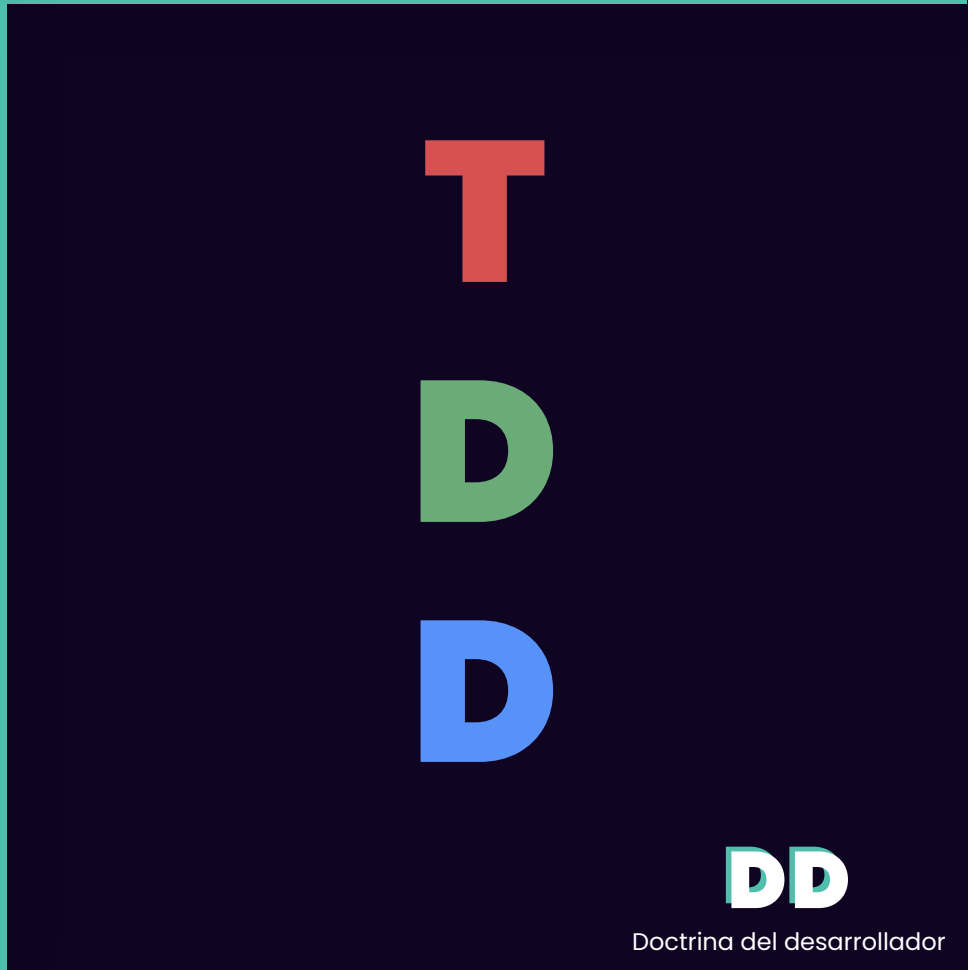
¿Que es TDD?

¿Qué es?

Es una metodología de desarrollo cuyo objetivo es crear primero las pruebas y luego escribir el software. Sus siglas en Inglés son: Test Driven Development y en español significa: Desarrollo guiado por pruebas

Objetivo

El objetivo de esta técnica es conseguir un código más robusto y un desarrollo mucho más rápido. Gracias a esto se pueden escribir las pruebas e ir consiguiendo que se mantengan fáciles de escribir



¿Por que se usa TDD?

¿Por que se usa TDD?

¿Por que?

- Construir el software correcto
- Mantener un diseño simple
- Generar código fácil de cambiar
- Asegurar el funcionamiento esperado
- Sostener el ritmo de desarrollo

Reglas de TDD

Reglas de TDD

- No escribir código hasta que el test lo requiera
- Referenciar nuevas clases, métodos, variables, etc, en el test primero para que no compile el código y luego crear las entidades o variables necesarias para que empiece a compilar y poder correr el test
- Apuntar a tener solo una funcionalidad “sin que funcione” por vez si es posible, esto permite atomizar los cambios

Diagrama explicado

Diagrama

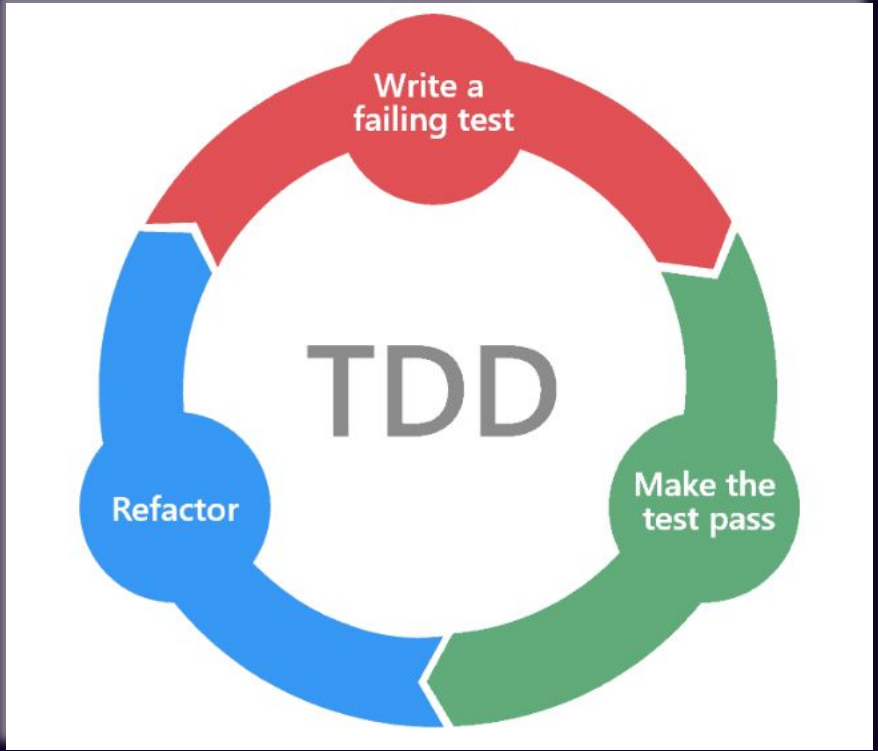


Diagrama – Write a falling test

Escribir test que fallen

Todo ciclo TDD comienza con la creación de una prueba de validación. En esta parte del proceso, el desarrollador tiene que conocer de forma clara los requerimientos, condicionales y casos bordes.
Se comienza creando los test (en lo posible unitarios) donde los mismos van a fallar al correrlos por que no se tiene código contra el cual validar.
Es fundamental para que el desarrollador se enfoque primero en los requerimientos previo a crear código. Esto también es otra forma de orientarse al negocio buscando crear valor desde la tecnología

Enterprise Business Rules
(Entities)

Diagrama – Make the pass test

Hacer que le test pase

Una vez que tenemos los test creados, debemos pasar a la segunda etapa que es crear el código necesario para que le test pueda contrastar con ellos.
Cuando los test pasan todas sus validaciones contra el código que creamos podemos dar como finalizada la etapa y ya podemos decir que el software es válido, pasa los requerimientos establecidos

Diagrama – Refactor

Refactorizar el código

En la etapa 2 el código no necesariamente se encuentre desarrollado de la manera más óptima (hay que recordar que los procesos de desarrollo deben ser ágiles). A medida que el software crece, comienzan a surgir necesidades de abstracción, modularización, eliminar casos de duplicación, entre otras.
Este proceso lo podemos llevar a cabo a lo largo de ciertos ciclos o inmediatamente después de que las pruebas pasaron correctamente. El tiempo que tengamos nos puede definir esta última etapa.

*En base a esto ultimo, siempre debemos intentar no dejar tareas para después, podemos incurrir en la deuda técnica (lo cual no es estrictamente malo), todo depende del tiempo que tengamos y cuan necesaria es la funcionalidad que debemos entrar

¿Por que no se usa TDD?

¿Por que es difícil incorporar TDD?

¿Por que?

La principal desventaja que enfrentan quienes se están iniciando en la implementación de TDD es que, inevitablemente, puede llevar más tiempo de desarrollo. La curva de aprendizaje puede resultar confusa, y a menudo la prioridad es cumplir con los plazos establecidos.
Además, otra razón por la cual puede resultar complicado implementar TDD es que el proceso en sí ofrece poca orientación concreta para su aplicación. Cuando uno está aprendiendo algo nuevo, es común seguir un conjunto de reglas preestablecidas para comprender el contexto y adquirir la experiencia necesaria para cuestionar o adaptar esas reglas. Ahí es donde surge la famosa pregunta sobre cuál es la mejor manera de llevar a cabo una acción: "Depende del contexto"

Ventajas y Desventajas

Ventajas y Desventajas

Ventajas

- Mayor calidad
- Diseño enfocado en las necesidades
- Mayor simplicidad en el diseño
- El diseño se va adaptando al entendimiento del problema
- Mayor productividad
- Menos tiempo invertido en debugging de errores

Desventajas

- Pronunciada curva de aprendizaje
- Errores no identificados (cabe destacar que se recomienda para test unitarios, para hacer cambios atómicos). No debemos olvidar los test de integración o end to end

Referencias

- **Medium:**
  https://medium.com/nursoft/implementar-tdd-facilmente-4d2cffaa9172
- **EdTeam:**
  https://ed.team/blog/que-es-el-tdd
- **Inesdi**
  https://www.inesdi.com/blog/que-es-TDD-test-driven-development/