TDD y BDD

Mayo 2023

KAIROSDS

KAIROS
DIGITAL SOLUTIONS

Quién soy

manu.fosela@kairosds.com

@manufosela

in/manufosela

KAIROSDS

Índice

TDD

KAIROSDS

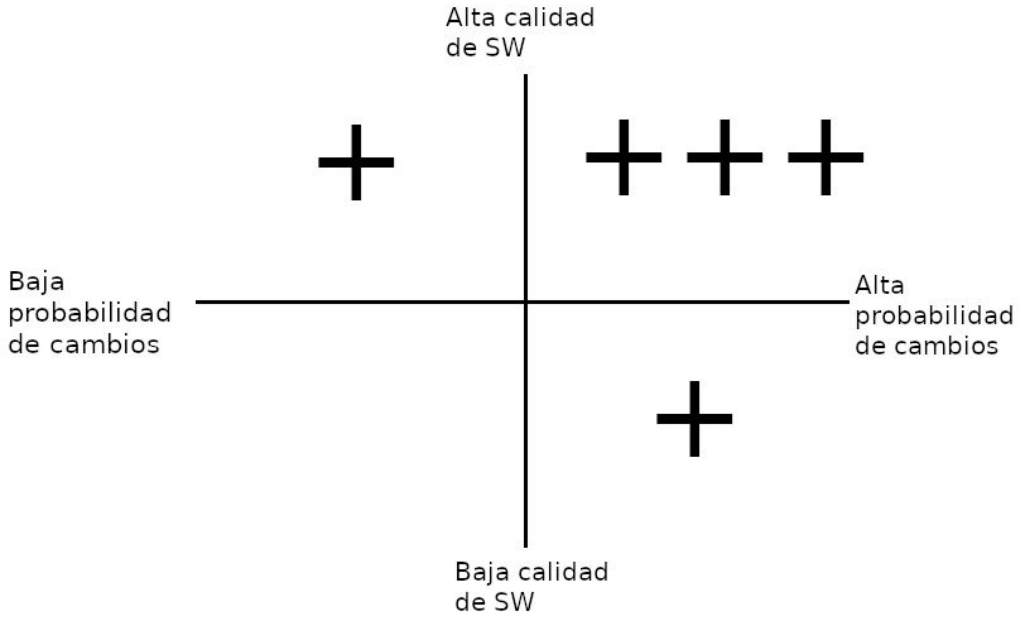
Es una práctica de programación que consiste en escribir primero las pruebas (unitarias) antes de escribir el código.

El objetivo de esta práctica es asegurarse de que el código esté bien diseñado, bien estructurado y sin errores, ya que se tiene la garantía de que todas las funcionalidades han sido probadas correctamente.

TDD: Flujo. Cómo aplicarlo

- **Escribir una prueba unitaria**: Se debe escribir una prueba unitaria que valide una pequeña parte del código que se desea implementar. Esta prueba debe fallar ya que el código aún no ha sido escrito.
- **Escribir el código mínimo necesario para pasar la prueba**: Se debe escribir el código mínimo necesario para que la prueba pase. Este código no tiene que ser perfecto ni completo, solo debe ser suficiente para que la prueba pase.
- **Refactorizar el código**: Una vez que la prueba pasa, se debe refactorizar el código para mejorarlo y hacerlo más legible, más mantenible y más escalable.
- **Repetir el proceso**: Se debe repetir este proceso para cada pequeña parte del código que se desee implementar.

TDD: Cuando aplicarlo



TDD: Impresiones cuando lo aplicas

Al principio hace ir más lento a los equipos.
Una vez se controla no, porque pones en producción código de gran calidad y preparado para ser modificado a corto o medio plazo.

El tiempo empleado en hacer los tests realmente es tiempo en diseñar.

El tiempo empleado en refactorizar asegura la calidad del código.

Los tests son una red de seguridad automática cuando se realice un cambio en el código.

TDD: Impresiones cuando lo aplicas

Si no necesitas calidad: PoC, prototipos, spikes, código pequeño, código no crítico... no es necesario aplicar TDD

Durante el proceso de aprendizaje de TDD no uses todo el rato TDD. Al principio los resultados no son los esperados y el equipo se frustará.
Es ideal que haya alguien en el equipo especializado en TDD para ayudar, guiar y apoyar.

TDD es una herramienta. Hay que aprender a sacarle provecho.

TDD: Conclusiones

Una vez que conocemos el concepto general de esta estrategia, hay que saber qué desafíos presenta utilizarla y cuáles son sus características:

- Necesitamos un gran conocimiento en desarrollo de pruebas unitarias.
- Desde una etapa muy temprana del proyecto debemos tener muy en claro qué se va a desarrollar, así como su alcance y sus limitaciones.
- Las pruebas están enfocadas desde la perspectiva del desarrollador.
- No se evalúa la calidad de la integración del producto, solo a nivel unitario.
- Además, conviene aclarar que esta estrategia eleva la calidad técnica del producto, es decir, de la salud del código, ya que provee de una gran cobertura de pruebas unitarias, incluso antes de comenzar el desarrollo. Y, ante cualquier cambio, se podría detectar mediante el fallo de estas pruebas.

BDD

KAIROSDS

Es una práctica de programación dirigida por el comportamiento

El objetivo de esta práctica es centrarse en una perspectiva del usuario y el comportamiento del sistema.
Estas pruebas son escritas en un lenguaje más coloquial y entendible.

BDD: Características

La característica principal es que se pueden escribir pruebas en un lenguaje común o de negocio (aquí entra en valor Gherkin) describiendo el flujo de un usuario por la aplicación, y cuál es el comportamiento del sistema como respuesta.

A estos flujos los llamamos Escenarios de pruebas.

No requiere que la parte directiva del proyecto tenga una alta formación técnica, ni que tenga conocimiento sobre el desarrollo de pruebas unitarias. Las **pruebas son de integración**, de forma que no tiene como objetivo principal las pruebas unitarias.

BDD: GIVEN, WHEN, THEN

Aparece la fórmula de GIVEN, WHEN, THEN (Dado…, Cuando…, Entonces…) que nos da una guía de cómo escribir correctamente cada escenario de pruebas.

Given. Aquí describimos las precondiciones necesarias para la ejecución del escenario.

When. Luego narraremos las acciones que hará el usuario en el sistema.

Then. Son todas las respuestas esperadas del sistema o validaciones.

BDD: Gherkin

Ejemplo de descripción de un escenario de pruebas utilizando Gherkin

```
Feature: Búsqueda en Google
  Como usuario web, quiero buscar en Google para poder responder mis dudas.

  Scenario: Búsqueda simple en Google
    Given un navegador web en la página de Google
    When se introduce la palabra de búsqueda "pingüino"
    Then se muestra el resultado de "pingüino"
```

BDD: Gherkin

Gherkin está diseñado en concreto para resolver un problema muy específico, que es un problema de comunicación entre los perfiles de negocio y los perfiles técnicos a la hora de trabajar bajo un enfoque BDD.

Esto es solo la forma en la que se describe la prueba; luego cada uno de los pasos o STEPS estarán asociados al código fuente que realizará las acciones que se describen en este lenguaje de negocio.

La estrategia de BDD surge desde TDD, con lo cual tiene sentido que sus flujos se entrelacen en algún punto.

BDD: Flujo

1. En esta primera instancia se identifican las funcionalidades de nuestro sistema, escritas en la documentación por todo el equipo. Por ese motivo, decimos que las pruebas en BDD no tienen solo la perspectiva del desarrollador. La intención es lograr tener una lista de requisitos priorizada para poder trabajar más libremente en la segunda etapa, una tarea propia del Product Owner.

BDD: Flujo

2. Tras identificar las funcionalidades a probar, a las que llamaremos Features, se comienzan a escribir los escenarios de pruebas. Pueden existir muchos escenarios relacionados con la misma funcionalidad. Nuevamente, estos escenarios de prueba siempre van a fallar porque aún no hay código para respaldar la funcionalidad descrita en lenguaje de negocio. Pero esta prueba nos sirve para acordar con los stakeholder las pruebas que se realizarán, al contrario de lo que ocurre con las pruebas unitarias, que solo pueden ser verificadas por el personal técnico.

BDD: Flujo

3. En esta etapa se escribe el código que respalda la prueba que se ha escrito en Gherkin. Como regla general, se escribe una función por cada STEP del escenario que será ejecutada en el orden en que esté escrito en Gherkin. Es decir, primero se ejecutará la función relacionada con el Given, luego la del When y, por último, la del Then.

BDD: Flujo

4. En esta etapa verificamos que las pruebas estén pasando por completo, ya que en BDD pueden pasar los STEPS individuales, pero fallar la prueba. Debemos asegurarnos de que la prueba completa es exitosa.

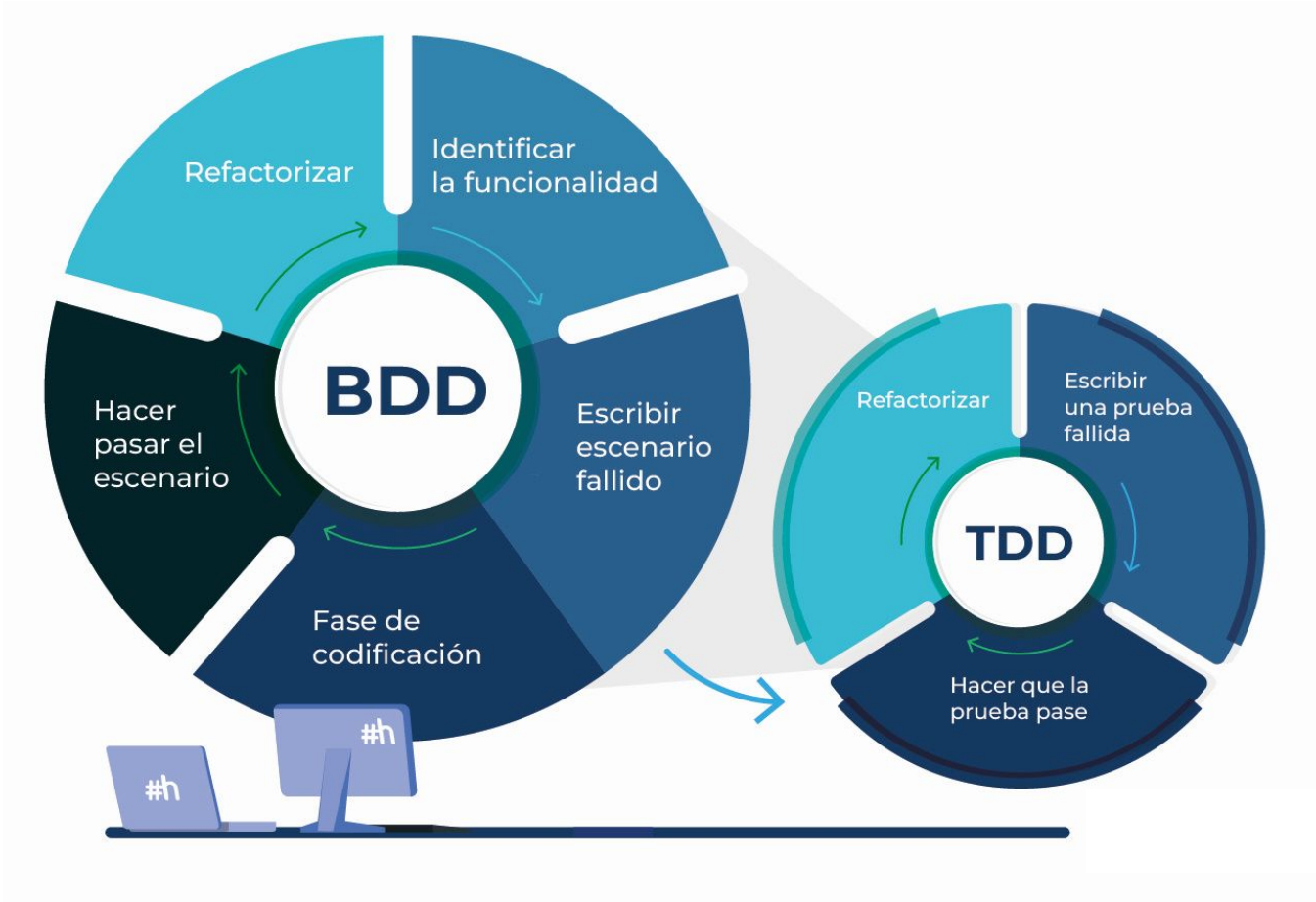
BDD: Flujo

5. Por último, daremos el salto a la optimización de código, además de velar por la calidad de código. En BDD es importante pensar en la "reutilización" de los STEPS porque pueden servir para otros escenarios. De esta manera, no será necesario volver a codificar la función relacionada con este step. Imagina un step que sea "Given el usuario inicia sesión en el sistema". Probablemente, esa precondición se repita en muchísimos escenarios y, si lo escribo de la forma correcta, puedo reutilizarlo tanto como lo necesites en lugar de duplicar código de forma innecesaria.

BDD: Documentación oficial

https://cucumber.io/docs/gherkin/

Diferencias entre TDD y BDD


Refactorizar
Identificar la funcionalidad
BDD
Hacer pasar el escenario
Escribir escenario fallido
Fase de codificación
Refactorizar
Escribir una prueba fallida
TDD
Hacer que la prueba pase

KAIROSDS

TDD vs BDD

Ambas son estrategias de desarrollo.

La diferencia principal es quién o qué dirige el desarrollo.

En el caso de TDD (Test Driven Development) las pruebas son las que van a marcar el camino a seguir.

En el caso de BDD (Behavior Driven Development) las pruebas se centran en la perspectiva del usuario y el comportamiento del sistema, usando un lenguaje coloquial: Gherkin.

TDD vs BDD

Ambas estrategias son complementarias

TDD se centra en pruebas unitarias, para testear los cambios en el código.
BDD se centra en probar el comportamiento, pruebas de integración, para testear que la aplicación se comporta como se espera.

GRANCIAS

Que es como **gracias** pero a lo **GRANDE**

@manufosela in/manufosela

KAIROSDS

KAIROSDS

KAIROSDS

KAIROSDS