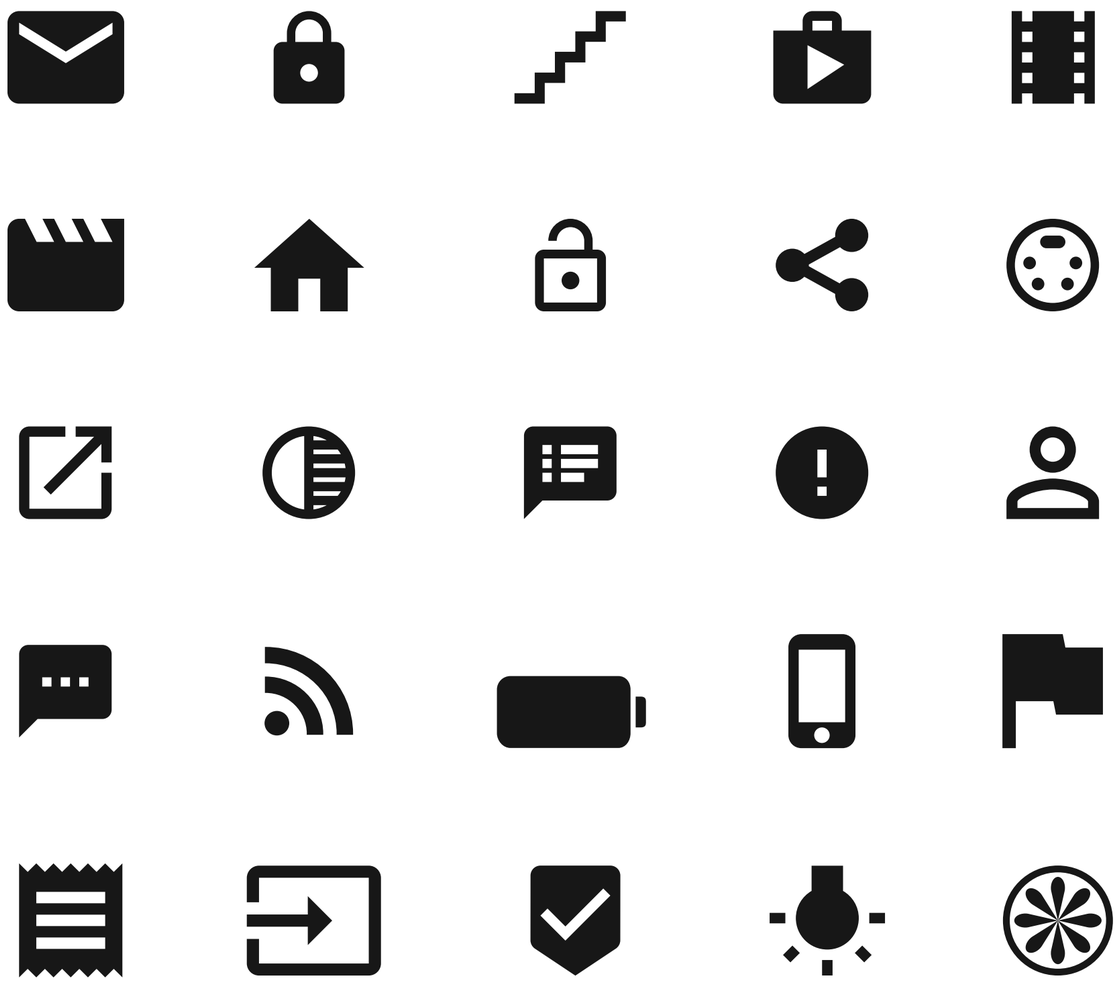
BDD

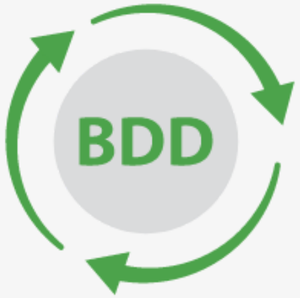
Eduardo Acevedo Barajas
Leslie Pedrero Lezama
Roberto Alfonso Ruiz Garay



BDD

BEHAVIOR DRIVEN DEVELOPMENT

BDD refiere a, desarrollo dirigido por comportamiento. Como bien lo indica su nombre, no se trata de una técnica de testing, sino que es una estrategia de desarrollo (así como TDD, que es test driven development). Lo que plantea es definir un lenguaje común para el negocio y para los técnicos, y utilizar eso como parte inicial del desarrollo y el testing.



TDD

- Se enfoca a la prueba unitaria.
- Las pruebas se enfocan en funcionalidades.
- o se escribe en un lenguaje común.

BDD

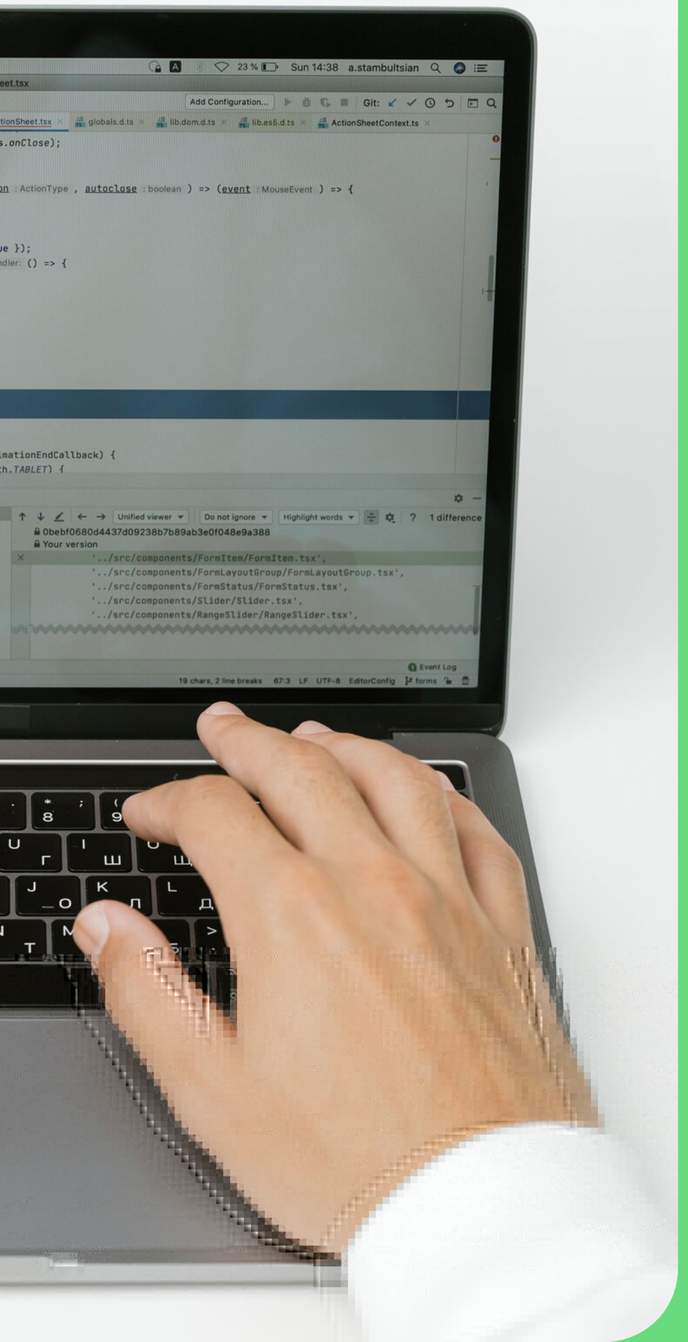
- Se enfoca en la prueba funcional, el foco está en cumplir con el negocio y no solo con el código.
- Las pruebas se centran en el usuario
- Se escribe en un lenguaje común utilizado por todas las partes interesadas.

¿Qué debo tener en cuenta antes de implementar BDD?

- Cada ejemplo debe ser un escenario de un usuario en el sistema.
- Ser consciente de la necesidad de definir "la especificación del comportamiento de un usuario" en lugar de "la prueba unitaria de una clase".
- Comprender la fórmula ‘Given-When-Then’ u otras como las historias de usuario ‘Role-Feature-Reason'.

BDD para automatizar pruebas

El lenguaje Gherkin es simplemente texto con algunas palabras clave y algo de estructura, que luego hay herramientas como Cucumber o JBehave, que permiten implementar una capa de conexión entre esa especificación de prueba y la interfaz del sistema que se quiere probar, pudiendo así utilizar eso como los pasos de una prueba automatizada.



‘Given-When-Then’ como lenguaje común con BDD

Para definir los casos BDD para una historia de usuario se deben definir bajo el patrón ‘Given-When-Then’, que se define como:

- Given ‘dado’: Se especifica el escenario, las precondiciones.
- When ‘cuando’: Las condiciones de las acciones que se van a ejecutar.
- Then ‘entonces’: El resultado esperado, las validaciones a realizar.

Un ejemplo práctico sería:

- Given: Dado que el usuario no ha introducido ningún dato en el formulario.
- When: Cuando hace clic en el botón Enviar.
- Then: Se deben mostrar los mensajes de validación apropiados.

‘Role-Feature-Reason’ como lenguaje común con BDD

Es un patron que también se utiliza en BDD para ayudar a la creación de historias de usuarios. Este se define como:

- As a ‘Como’: Se especifica el tipo de usuario.
- I want ‘deseo’: Las necesidades que tiene.
- So that ‘para que’: Las características para cumplir el objetivo.

Herramientas de definición BDD

La herramienta más destacada, basado en el patrón ‘Given-When-Then’ es Cucumber, un framework de test con soporte BDD. En Cucumber, las especificaciones de BDD están escritas en lenguaje Gherkin, basado en ‘Given-When-Then’. En otras palabras, Gherkin presenta el comportamiento de la aplicación, a partir de la cual Cucumber puede generar los casos de prueba de la aplicación.

01 **Java**

JBehave, JDave, Instinct, beanSpec

02 **C**

CSpec

03 **C#**

Spec, Behave

04 **.Net**

- Spec, Behave, SpecFlow

05 **PHP**

PHPSpec, Behat

06 **Ruby**

- RSpec, Cucumber

07 **Python**

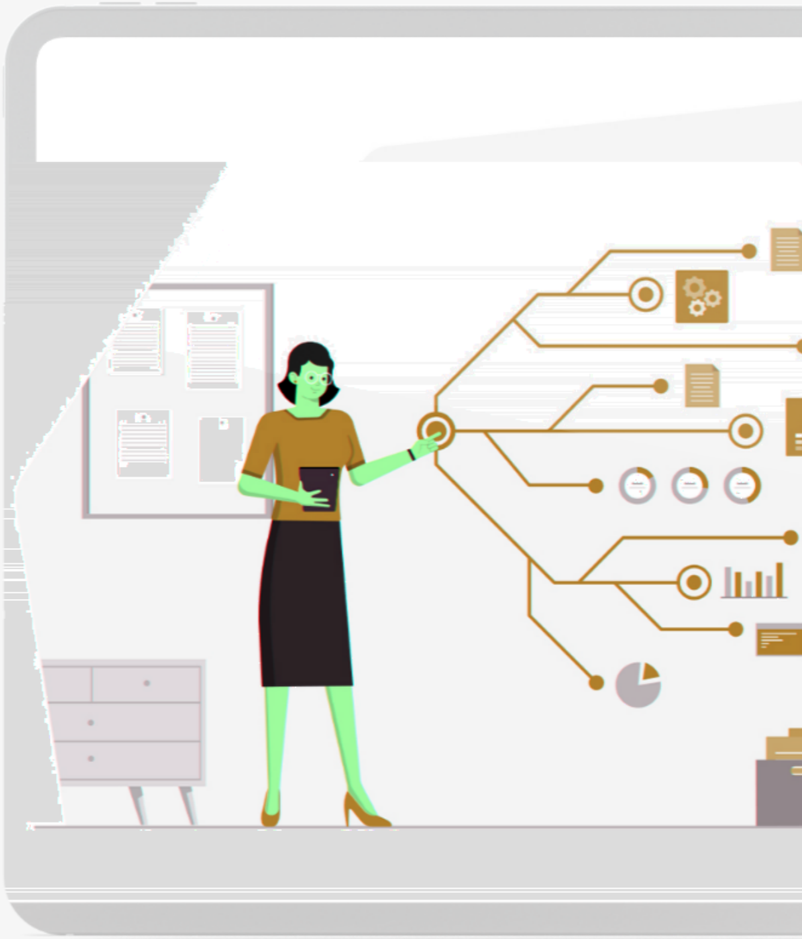
Freshen

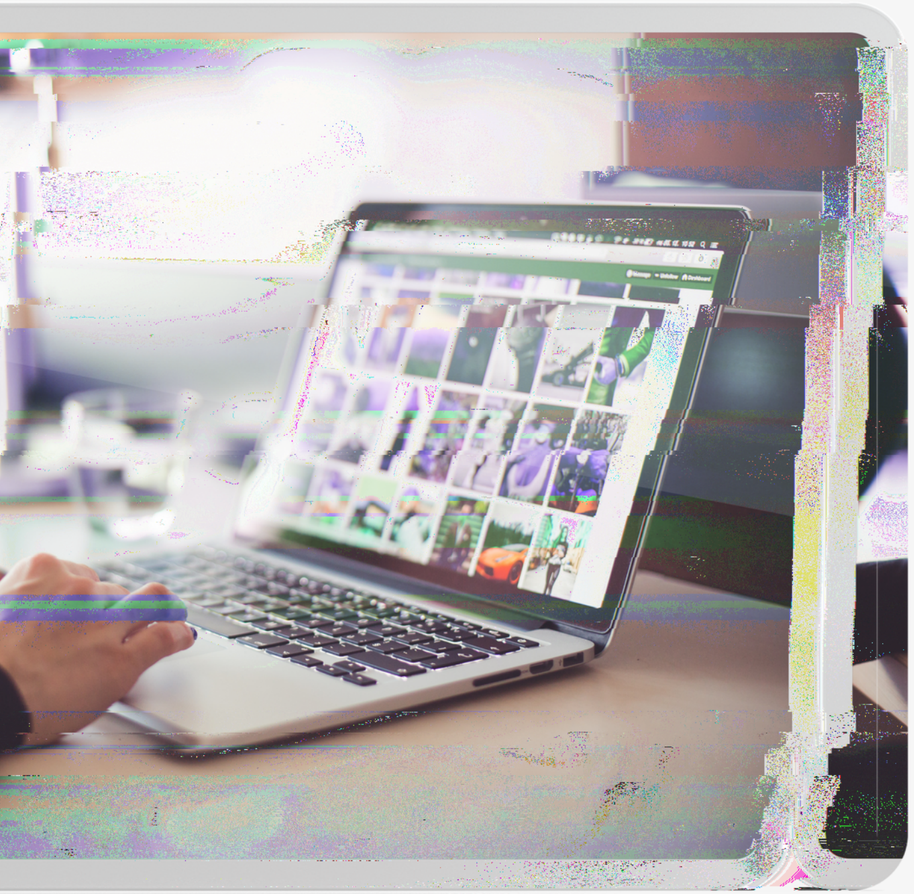
85 **JavaScript**

JSSpec, jspec

¿Por qué usar BDD?

- No se estan definiendo 'pruebas', sino que está definiendo 'comportamientos' y 'escenarios'.
- Mejora la comunicación entre desarrolladores, testers, usuarios y la dirección.
- Debido a que BDD se específíca utilizando un lenguaje simplificado y común, la curva de aprendizaje es mucho más corta que TDD.
- Como su naturaleza no es técnica, puede llegar a un público más amplio.
- El enfoque de definición ayuda a una aceptación común de las funcionalidades previamente al desarrollo.





Conclusión

BEHAVIOR DRIVEN DEVELOPMENT

Esta estrategia de desarrollo te permite desarrollar, probar y pensar el código desde la perspectiva del usuario.
Se debe tener la mentalidad de implementar ejemplos del mundo real en lugar de implementar solo funcionalidades.
Las ventajas son muy considerable para integrar a todo el equipo con un objetivo común, además de empatizar con el usuario final de tus desarrollos.