



• ¿Qué son las pruebas y por qué deberíamos hacerlas?

Las pruebas son un proceso de evaluar un producto, aprendiendo a través de la exploración. Esto incluye cuestionar, estudiar, modelar, observar e inferir, etc.

• Si se hacen pruebas tendremos software sin errores? **NO**

Razones para hacer pruebas:

\* Resultado NO esperado

\* Costos altos fuera del presupuesto

\* Implicaciones legales o de estándares tecnológicos

# Procesos de pruebas de software

Metodología: Viene a ser la parte dónde se establece el criterio o estrategia de cómo se llevarán a cabo las pruebas.

Herramientas: Contar con herramientas adecuadas permitirá acelerar el testing

Recursos: Hay que estar preparado para hacer pruebas, sino con el tiempo las carencias se irán manifestando como defectos

---

---

Una certificación NO asegura  
que las pruebas se hagan  
correctamente

A lo largo del ciclo de vida del software se realizan distintas pruebas para garantizar que este cumpla con los requerimientos para los que fue diseñado y de la misma forma se encuentren procesos de mejora y optimización a medida que se desarrolla el software.

- Es necesario hacer pruebas en todas las fases del desarrollo de software ya que un error encontrado en una etapa tardía puede generar costos muy elevados.
- Errores detectados lo antes posible reducen los costos y son mucho más fáciles de corregir.
- El ciclo de vida permite que los errores se detecten lo antes posible y por lo tanto, permite a los desarrolladores concentrarse en la calidad del software, en los plazos de implementación y en los costos asociados.



#### 1. Definición de objetivos

En esta fase se define el alcance general del software y su papel dentro de una estrategia global o dentro del ecosistema que va a funcionar.

#### 2. Análisis de los requisitos y su viabilidad

Se trata de recopilar la mayor cantidad de información posible para evaluar la viabilidad del producto, encontrar posibles restricciones y analizar todos los requisitos del cliente.

#### 3. Diseño

*Alto nivel:* Se trata de realizar un diseño básico que valide la arquitectura de la aplicación.

*Bajo nivel:* Es una definición detallada de la estructura de la aplicación basada en el diseño general.

#### 4. Programación

Es la implementación de un lenguaje de programación para crear las funciones definidas durante la etapa de diseño.

#### 5. Pruebas de verificación

Aunque en todas las fases anteriores se hacen pruebas en esta fase se cubren: pruebas de componentes, integrales y de sistema.

#### 6. Prueba beta (o validación)

Se hace para garantizar que el software cumple con las especificaciones originales o también se hacen las pruebas de aceptación.

#### 7. Implementación

Se realiza una prueba del sistema implementado para encontrar posibles fallas en la Implementación.

#### 8. Mantenimiento

Se hace para todos los procedimientos correctivos (mantenimiento correctivo) y a las actualizaciones secundarias del software (mantenimiento continuo), junto con la actualización de las pruebas.

Si hacemos las pruebas durante cada fase del ciclo de vida del software tendremos al final del ciclo un producto validado y robusto de acuerdo a las necesidades del cliente

## Calidad y defectos

\* La calidad la define el cliente

¿Qué es la calidad?

Es el grado con el que un sistema, componente o proceso cumple con los requisitos especificados y las necesidades o expectativas del cliente o usuario.

## Verificación y validación

La verificación es ir en cada una de las etapas revisando que se cumpla lo que el cliente pidió.

La validación es antes de entregar al cliente validamos que efectivamente en el conjunto de todos los requerimientos todo lo entregado se cumple

Anomalía:

Cualquier condición insatisfactoria  
No reproducible

Detecto:

Es reproducible

No desempeña funciones

Fallo:

Situaciones no asociadas

Incapacidad dentro de márgenes

Error:

Acción humana incorrecta

"El error humano cometido inyecta un defecto en el software que, ocasionalmente, se observa como una anomalía a causa de un comportamiento incorrecto, no acorde a lo especificado, que finalmente provoca el fallo del sistema software"

# Testing moderno y especialidades de testing

## 7 principios del testing moderno

"Los testers podemos comenzar a pasar de ser los dueños de las pruebas o la calidad, a ser los embajadores de la calidad del producto"

---

- ① Nuestra prioridad es mejorar el negocio
- ② Nosotros aceleramos al equipo, usando modelos como **Lean thinking** y **teoría de las restricciones** para ayudar a identificar, priorizar y mitigar cuellos de botella en el sistema
- ③ Somos la fuerza para la mejora continua, ayudando al equipo a **adaptarse** y **optimizar** para tener éxito en lugar de proporcionar una red de seguridad para detectar fallas



- ④ NOS preocupamos profundamente acerca de la cultura de calidad en el equipo, y afororamos, lideramos y nutrimos el equipo para llevarlos a una cultura de calidad más madura
- ⑤ Nosotros creamos que el cliente es el único capaz de juzgar y evaluar la calidad de nuestro producto
- ⑥ Nosotros usamos datos de manera extensa y profunda para entender las causas de uso del cliente y entonces cerrar huecos entre hipótesis del producto e impacto del negocio
- ⑦ Expandimos las habilidades de testing y el conocimiento en todo el equipo; entendemos que esto reduce o elimina la necesidad de un especialista dedicado al testing.

# Especialidades de testing

\* Automation tester

\* Security tester

\* Data science tester

\* SDET

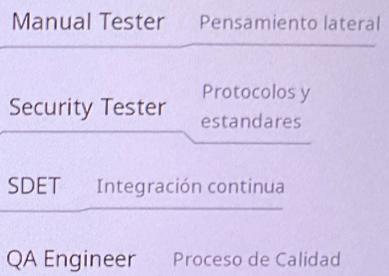
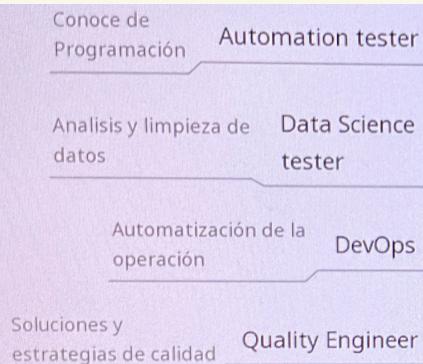
\* Devops

\* QA Engineer

\* QE

\* Manual tester

## ESPECIALIDADES DEL TESTING



# Testing durante el ciclo de vida del desarrollo de software

Construcción del software:

- \* Presupuesto
- \* Recursos
- \* Tiempo

Actividades clave de pruebas

\* Definición de necesidades

\* Análisis

\* Diseño

\* Codificación

\* pruebas

\* Validación

\* Mantenimiento y evolución

# Estrategia de pruebas

- \* ¿Qué problemas tenemos actualmente?
- \* ¿Qué problemas debemos evitar?

## Escenarios y Contextos

- \* Seguridad
- \* Arquitectura
- \* Performance
- \* Usabilidad
- \* Escalabilidad

# Testing VS Checking

## Estrategias de checking:

- \* Solo se ejecutan si ..
- \* Se ejecutan cada que ..
- \* Se ejecutan de manera programada

## Errores comunes durante la ejecución

- \* Pruebas duplicadas
- \* ✓ similares
- \* ✓ sin valor agregado
- \* ✓ caducadas

La automatización de pruebas consiste en el uso de software especial para controlar la ejecución de pruebas y la comparación entre los resultados esperados. Sin embargo, se trata de un checking repetitivo y automatizado.

## Desventajas del checking mal empleado

- \* Pobre cobertura de pruebas
- \* Falta de actualización
- \* Mal manejo de versiones

## Ventajas del checking bien empleado

- \* Correr pruebas en paralelo o en múltiples plataformas
- \* Reducción de error humano
- \* Probar grandes cantidades de datos

# Testing ágil

Involucra a todos del equipo para garantizar el valor deseado al cliente a un ritmo sostenible y continuo

---

---

## Niveles de pruebas de software

- \* Pruebas de componentes
- \* Pruebas de integración
- \* Pruebas de sistema
- \* Pruebas de aceptación

# Tipos de pruebas de software

- ① Pruebas funcionales: Se entiende como las funcionalidades del sistema como "lo que el sistema hace".
- ② Pruebas No-funcionales: El objetivo es probar "cómo funciona el sistema"
- ③ Pruebas estructurales: El tester debe de tener conocimientos acerca de la tecnología y el stack que se está empleando. Mejor conocidas como "Pruebas de caja blanca"
- ④ Pruebas de manejo de cambios: Es probar nuevamente un componente ya probado para verificar que no ha sido impactado por actualizaciones

# Pruebas estáticas y dinámicas

Las pruebas dinámicas se enfocan principalmente en comportamientos externos visibles durante la ejecución del software.

Las pruebas estáticas se basan en la examinación manual de los elementos que conforman la construcción del software.

# Definición y diseño de pruebas

¿Qué hace un tester?

① Encontrar problemas

② Documentar problemas

③ Comunicar problemas

# Pruebas de Caja

- \* Negra
- \* Blanca
- \* Gris

## TÉCNICAS DE PRUEBAS

### CAJA NEGRA

- Partición de equivalencia
- Valores límite
- Tabla de decisiones
- Transición de estados
- Casos de uso

### CAJA GRIS

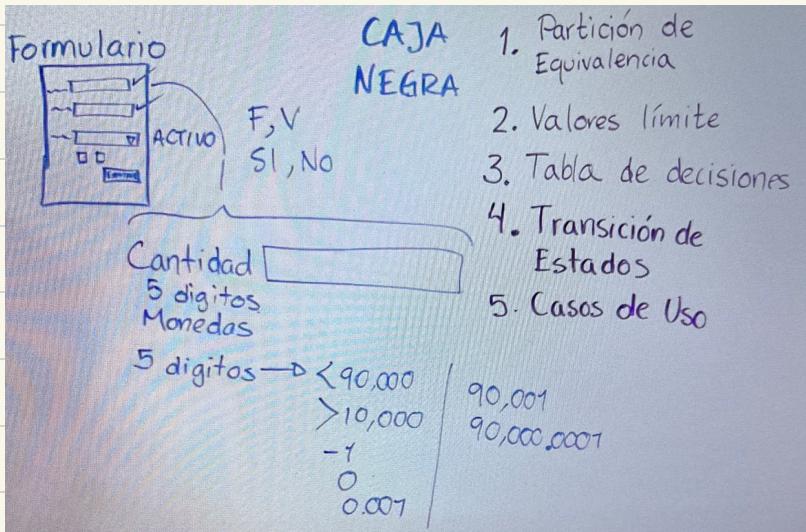
- Casos de negocio
- Pruebas End to End
- Pruebas de integración

### CAJA BLANCA

- Cobertura de declaración
- Cobertura de decisiones

# Caja negra

- ① Partición de equivalencia
- ② Valores límite
- ③ Tabla de decisiones
- ④ Transición de estados
- ⑤ Casos de uso



Cubre los requerimientos del Frontend

# Caja blanca

① Cobertura de declaraciones

② Cobertura de código

- Sentencias
- Decisiones
- Condiciones

Cubre los reavivamientos del  
Backend

# Caja gris

- ① Casos de negocio
- ② Pruebas End to End
- ③ Pruebas de integración



# Gestión de monitoreo y control

- Planeación de pruebas
  - Monitoreo y control de pruebas
  - Análisis de pruebas
  - Diseño de pruebas
  - Implementación de pruebas
  - Ejecución de pruebas
  - Finalización de las pruebas
- 

## Planeación de pruebas

Definición de objetivos de las pruebas, alcance de las mismas, las técnicas de pruebas que se llevarán a cabo junto con la estimación y definición de fechas de entrega, así como los criterios de salida.

## Monitoreo y control de Pruebas

Durante el Monitoreo se va midiendo y comparando los resultados de las métricas, y entonces durante el control se toman acciones para alcanzar el objetivo del plan y las criterios de salida.

## Análisis de pruebas

Se necesita determinar qué debemos probar basandonos en las propiedades de cobertura

# Diseño de pruebas

- Diseño de casos de alto nivel
- Diseñar y priorizar las pruebas
- Identificar los datos de pruebas
- Identificar el entorno de pruebas - infraestructura y herramientas
- Hacer trazabilidad entre pruebas y sus condiciones

## Implementación de pruebas

Para poder prepararnos para hacer las pruebas, primero tenemos que asegurarnos que tenemos todo lo necesario para ello.

## Ejecución de pruebas

Durante esta etapa las suites de pruebas se ejecutan de acuerdo con el programa de ejecución de las pruebas

## Finalización de pruebas

- \* Defectos con el estatus correcto
- \* Reporte de resultados de pruebas
- \* Finalizar y archivar ambiente de pruebas y datos
- \* Entrega del Testware al equipo de mantenimiento
- \* Analizar lecciones aprendidas
- \* Recopilar información para mejorar la madurez del proceso de prueba

# Roles y responsabilidades

- Especialista en pruebas Manuales
- Especialista en pruebas técnicas
- Líder del equipo de pruebas
- Ingeniero de calidad

Independientemente del rol, un tester participa de todas las etapas del proceso de desarrollo de software, colaborando para asegurar la máxima calidad del producto. Su perfil conjuga un conjunto de habilidades con el conocimiento del negocio, de la aplicación bajo prueba y de cómo planificar, diseñar, ejecutar y administrar las pruebas.

\*Un tester Investiga un producto de software con el objetivo de obtener información acerca de su calidad y del valor que representa para quienes lo utilizan\*

## HABILIDADES DE UN TESTER

- ◎ Capacidad de abstracción y modelado para entender y simular el comportamiento del sistema bajo prueba.
- ◎ Comunicación para interactuar con todos los miembros del equipo
- ◎ Pensamiento lateral para generar ideas e imaginar los problemas que podrían existir.
- ◎ Pensamiento crítico para observar y evaluar, hacer deducciones y vincular lo observado con los requerimientos de calidad de la empresa.
- ◎ Pragmatismo ser lo suficientemente abierto para poner en práctica las ideas de los demás o propias, adecuar las técnicas y el esfuerzo al alcance del proyecto.
- ◎ Trabaja en equipo, sabe interactuar para lograr el mayor beneficio entre todos los involucrados.



Cuando el esfuerzo en la calidad se enfoca y se distribuye en roles y responsabilidades podemos encontrar que la mayoría de las empresas manejan los siguientes roles:

El tester manual, se enfoca en la estrategia, definición, ejecución y cobertura de pruebas para cumplir los requerimientos, echando mano de cualquier técnica para obtener información suficiente y así cumplir con las asignaciones correspondientes.

El tester técnico, trabaja muy de cerca con el tester manual, mientras que el tester manual define las pruebas, el tester técnico acelera la capacidad de ejecución de las pruebas. Esto lo hace implementando herramientas que permitan la automatización de pruebas, o la correcta selección de datos de pruebas, o el monitoreo de la ejecución de las pruebas.

El líder de pruebas, generalmente dentro de sus responsabilidades es volverse un facilitador de servicios, información y herramientas para el equipo de pruebas, para poder estimar presupuestos, recursos y tiempos respecto al plan de desarrollo de software.

El Ingeniero de calidad, ya no solamente está al pendiente del producto y los procesos, comienza a involucrarse más con el negocio, ayudando tanto a testers como cualquier otro miembro del equipo a llevar cabo pruebas que reduzcan, en todas las etapas del ciclo de vida del software, el error humano.

# Preguntas

- ① Es el encargado de hacer coaching de calidad en toda la empresa  
Ingeniero de calidad
- ② Es el responsable de emplear herramientas que automatizcen o aceleren las ejecuciones de las pruebas  
Tester técnico
- ③ Es el responsable de asegurarse de la cobertura de pruebas y nuevos escenarios  
Tester manual
- ④ Es el facilitador del equipo que se asegura de implementar lo necesario para que se ejecuten las pruebas  
Líder de pruebas

# Retrabajo

Acciones de control:

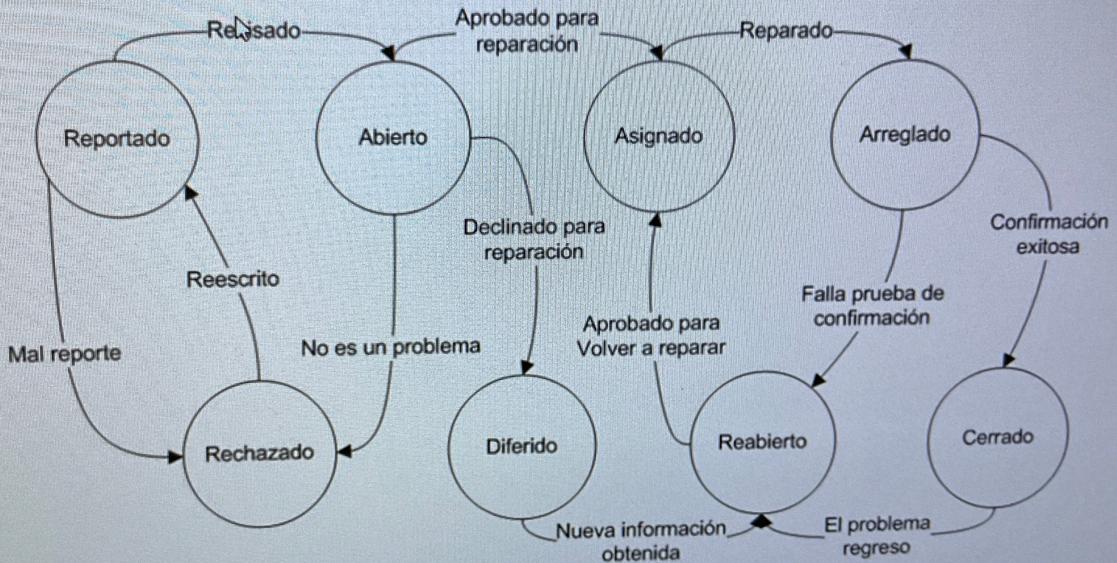
- Si identificamos un riesgo...
- ✓ ✓ falta de ambientes...
- Si el criterio de salida no se cumple...

El retrabajo es un esfuerzo adicional necesario para la corrección de una inconformidad en algún producto.

- \* Falta o mala documentación
- \* Falta de capacitación o dominio de herramientas
- \* ✓ ✓ ✓ ✓ ✓ del software a desarrollar
- \* Falta de comunicación

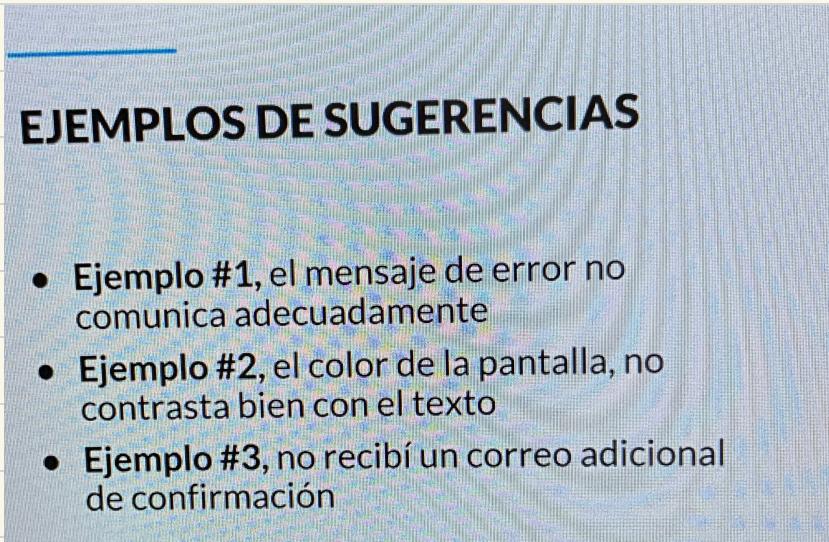
# Gestión de bug

## CICLO DE GESTIÓN



# Defectos y Sugerencias

## Defectos VS Sugerencia



# Depuración / Debugging

Beneficiados :

- \* Programador
- \* Tester
- \* Analista / Investigador

→ Errores = Oportunidades para mejorar

Tipos de herramientas :

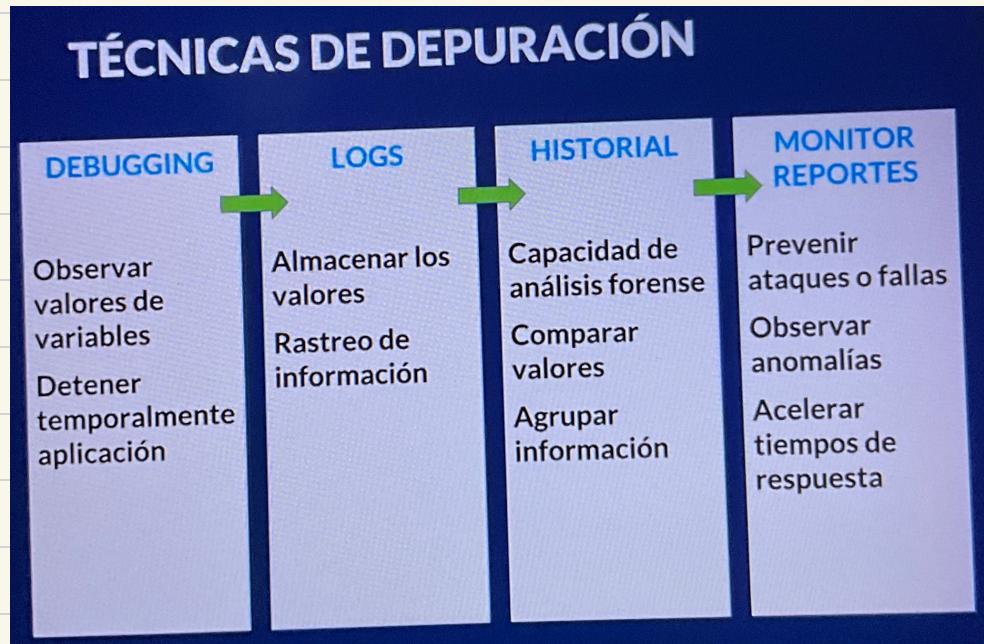
- Debugger
- Manual
- Local / Remota

# Pruebas de verificación

- Tratar de reproducir el escenario fallido con los datos nuevos
- Se buscan nuevos escenarios adicionales para hacer las pruebas

"Pruebas de regresión"

# Técnicas de depuración



Desventajas de NO usar logs :

- Visibilidad nula de errores
- Metodología de trabajo no es standarizada
- Acceso e información descentralizada
- Incremento del tiempo de respuesta

# Fases del debugging ...

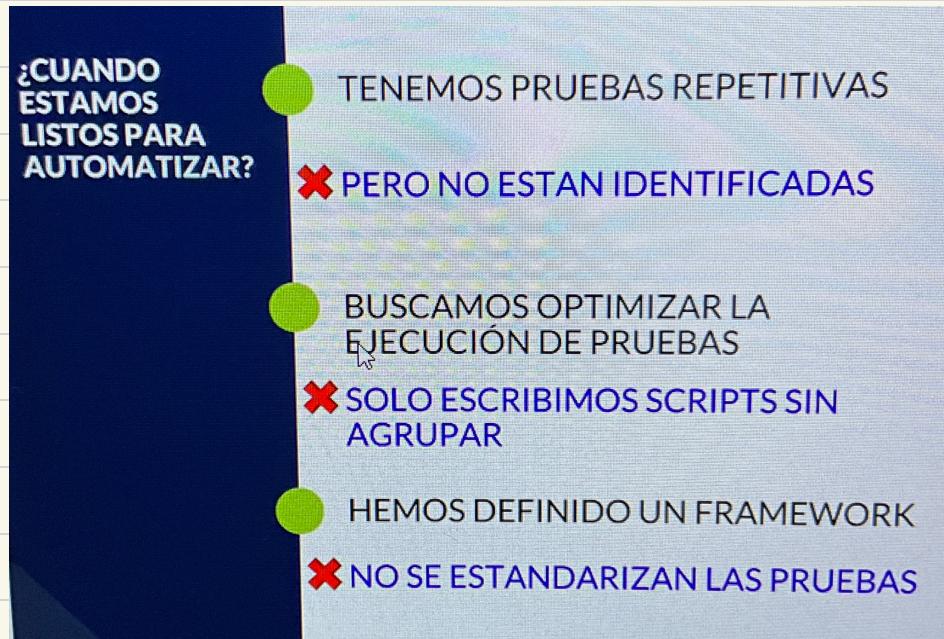
① Encontrar el error

② Corregir el error

## FASE 1: PASOS PARA DEPURAR

1. Ir al módulo que falla
2. Establecer breakpoints
  - a. En asignación de valores
  - b. Procesamiento de valores
  - c. Cambio de estados
3. Diseñar una mátrix de pruebas
4. Establecer los datos de prueba
5. Comenzar a depurar

# Bases de la automatización de pruebas



- \* Pruebas unitarias
- \* Pruebas de integración
- \* Pruebas funcionales o de aceptación

# Frameworks ...

## TDD (TEST DRIVEN DEVELOPMENT)

Es una técnica de diseño e implementación de software incluida dentro de la metodología XP (**Extreme Programming**).

Esta técnica nos permite obtener una **cobertura de pruebas muy alta**, aunque es importante destacar que este índice no indica que tengamos una buena calidad de tests. Por lo tanto, no debe ser un valor en el que fijarse únicamente.

Proceso :

- ① Escribimos una prueba
- ② Ejecutamos la prueba: Falla
- ③ Se escribe el código
- ④ Ejecutamos la prueba: Pasa

## **BDD (BEHAVIOR DRIVEN DEVELOPMENT)**

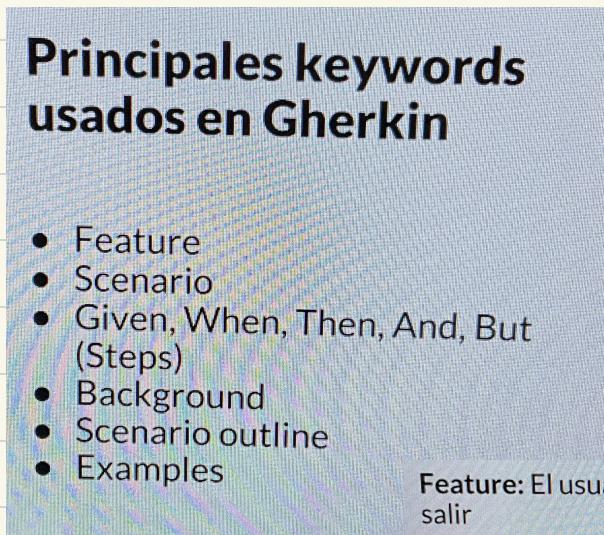
BDD es el desarrollo guiado por el comportamiento. Es un proceso que proviene de la evolución del TDD.

En BDD también se escriben pruebas antes del código, pero en vez de ser pruebas unitarias son pruebas que van a verificar que el comportamiento del código es correcto desde el punto de vista de negocio.

# Gherkin

Ventajas:

- \* Simple
- \* Palabras claves
- \* Estandariza los casos de uso
- \* Reduce el tiempo de diseño



**Feature:** El usuario abre 1 puerta de perilla para salir

#comentarios

**Scenario:** El usuario tiene una puerta cerrada

Given girando la perilla

And empujando la puerta

When la puerta abre hacia afuera

Then la puerta queda abierta

And el usuario puede salir