

JULY 11, 2021 - 11H

JUNE 15, 2021 - 15H

AUGUST 8, 2021 - 16H

// FEBRUARY 2021

MARIL 15, 2

ting with C

LITTLE POP-UP WINDOWS MEETING

Here is where your presentation begins

MEETING DATE & TIME

Main subject to discuss

JUNE 15, 2021 - 15H

MARCH 22, 2021 - 15H



¿QUÉ ES EL TESTING?

- El **Testing de Software** es una disciplina en la ingeniería de software permite tener procesos, métodos de trabajo y herramientas para identificar defectos en el software alcanzando un proceso de estabilidad.
- El **Testing** no es una actividad que se realiza al final del desarrollo del software, va paralelo a este. Se realiza de manera correcta de acuerdo a lo que necesita un usuario final. De ahí viene su importancia, ya que es una forma de corregir los posibles errores del software antes de que sea operable.





¿CUÁNDΟ Y CÓMO SURGIÓ EL TESTING?

- El **Testing de Software** nace aproximadamente en el año 1960 a partir de la crisis del desarrollo del software, cuando empiezan a desarrollarse los primeros softwares para el Departamento de Defensa de los Estados Unidos.
- A esta época se le llamó así porque el software era muy complicado para elaborar, no se entregaba a tiempo, era muy costoso, y difícil identificar su avance porque no es tangible. Y es por ello que se generaron múltiples soluciones, y una de ellas fue el Testing.



TIPOS DE PRUEBAS

Según si ejecutamos o no el código, tenemos pruebas estáticas y dinámicas.

Las estáticas son aquellas que se hacen sin necesidad de ejecutar el código, mientras que las dinámicas son aquellas en las cuales hay que ejecutar el software para poder probarlo.

Según el uso de herramientas, tenemos las pruebas manuales o automáticas.

Las pruebas manuales son aquellas en las que se prueba una navegación normal, mientras que en las automáticas se usa una herramienta para realizar estas pruebas.

Según lo que verifican, tenemos las funcionales y las no funcionales.

Las pruebas funcionales, basadas en las funcionalidades del software.

Existen varios tipos:

- Las unitarias, que son pruebas para trozos de código concretos.
- Las de integración, que son pruebas a todos los componentes juntos, para ver cómo interactúan entre ellos.
- Las de aceptación, que las suele hacer el usuario. Puede que un software funcione bien, pero tal vez no hace lo que el usuario pide.
- Las pruebas de regresión, que se realizan cuando hemos introducido cambios, pero no sabemos si al realizarlos hemos introducido algún defecto en alguna parte que ya funcionaba. Este tipo de pruebas suele ser muy común que se automaticen.

Las pruebas no funcionales, que por el contrario son aquellas que no se basan en aspecto funcional. Hay varios tipos:

- De seguridad, en las que se buscan vulnerabilidades de seguridad.
- De rendimiento, que permiten conocer el comportamiento del software ante una carga determinada.
- De usabilidad, que se emplean para saber cómo de usable es la aplicación, pero sin entrar en aspectos funcionales.



PROCEDIMIENTOS Y CASOS DE USO



Un caso de prueba o test case es un conjunto de condiciones o variables bajo las cuales un analista determinará si una aplicación, sistema o característica de éstos las cumple totalmente o solo una parte de ellas.

Debe haber al menos un caso de prueba para cada requisito a menos que un requisito tenga requisitos secundarios. En ese caso, cada requisito secundario deberá tener por lo menos un caso de prueba.



// Procedimientos y casos de prueba



→ **Introducción/visión general:** Contiene información general acerca de los Casos de Prueba.

- ◆ Identificador
- ◆ Caso de prueba dueño/creador
- ◆ Versión
- ◆ Nombre
- ◆ Identificador de requerimientos
- ◆ Propósito
- ◆ Dependencias

→ **Actividades de los casos de prueba**

- ◆ Ambiente de prueba/configuración
- ◆ Inicialización
- ◆ Finalización
- ◆ Acciones
- ◆ Descripción de los datos de entrada

→ **Resultados**

- ◆ Salida esperada
- ◆ Salida obtenida
- ◆ Resultado
- ◆ Severidad
- ◆ Evidencia
- ◆ Seguimiento
- ◆ Estado



PRUEBAS DE CÓDIGO



Para confirmar que
funciona todo
correctamente

```
probar.py ×
11
12 class TestOperaciones(unittest.TestCase):
13     def setUp(self):
14         # Aquí, opcionalmente, ejecuta lo que de-
15         # de comenzar cada test.
16         pass
17
18     def test_suma(self):
19         esperado = 3
20         actual = suma(1, 2)
21         # Pásalo en el orden: actual, esperado
22         self.assertEqual(actual, esperado)
23
24     def test_resta(self):
25         esperado = 5
26         actual = resta(10, 5)
27         # Pásalo en el orden: actual, esperado
28         self.assertEqual(actual, esperado)
```

PRUEBAS DE CÓDIGO

*



Manuales

- Son llevadas a cabo por personas que navegan e interactúan con el software.
- Pruebas costosas y expuestas a errores humanos.

Automatizadas

- Realizadas por máquinas que ejecutan un "test script".
- Más rápidas y confiables que las que se llevan a cabo manualmente

CUBRIMIENTO

*

- Determina cómo de buenas son nuestras pruebas unitarias.
- Nos dice la cantidad de código que está sometido a nuestras pruebas.



TABLE OF CONTENTS



Classes in this File	Line Coverage	Branch Coverage	Complexity
Autentia	50% 4/8	25% 1/4	4
1 package com.autentia.tutorial; 2 3 1 public class Autentia { 4 5 public void tellMeSometing(int i) { 6 1 if (i < 5) { 7 1 System.out.println("Soy menor que cinco"); 8 1 return; 9 } 10 11 0 if (i % 2 == 0) { 12 0 System.out.println("Soy un número par"); 13 0 return; 14 } 15 0 } 16 }			

VALORES LÍMITES



Se trata de tomar los valores que están tanto en el límite inferior como en el superior de un determinado rango. El rango no tiene porque ser numérico, pero si tiene que estar definido, es decir, saber dónde empieza y dónde acaba.

Por ejemplo:

Tenemos un programa que permite valores entre 0 y 1000.

Límite inferior



-1, 0, 1

Límite superior



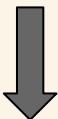
999, 1000,
1001

CLASES DE EQUIVALENCIA



El objetivo es ver comportamientos comunes en el rango de valores de nuestro programa, para luego seleccionar un valor representativo de cada conjunto y hacer las pruebas con estos valores

Valores no válidos menores a 0



Elijo -3

Valores válidos (entre 0 y 1000)



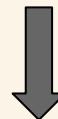
Elijo 5

Valores no válidos mayores a 1000



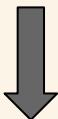
Elijo 1050

Formato numérico no admitido



Elijo 200.4

Valores inválidos no numéricos



Elijo "R"

Depurador



- ¿Qué es un depurador?

Un depurador es un programa que permite detectar y diagnosticar fallos en programas informáticos. El objetivo de estas herramientas es garantizar, a largo plazo, que el software funcione en todos los dispositivos y plataformas para los que está pensado. Por este motivo, muchos depuradores no solo analizan el código fuente del programa, sino también su interacción con el sistema operativo que lo ejecuta y con los elementos de hardware. El proceso de depuración o debugging ocurre mientras el programa se ejecuta, de forma que no es necesario cerrarlo para llevar a cabo el análisis.

Ejemplos de Depuradores



01

GNU Debugger

GDB o GNU Debugger es el depurador estándar para el compilador GNU.

03

OllyDbg

OllyDbg es un depurador de código ensamblador de 32 bits para sistemas operativos Microsoft Windows.

02

SoftICE

SoftICE es un depurador en modo kernel propietario y de pago para Microsoft Windows.

04

Cheat Engine

Cheat Engine, abreviado CE, es un escáner de memoria desarrollado como software libre, además de editor hexadecimal y depurador

JULY 11, 2021 - 11H

JUNE 15, 2021 - 15H

AUGUST 8, 2021 - 16H

// APRIL 2021

MAY 15, 2021

ing with C

PLANIFICACIÓN DE PRUEBAS

TESTING

Introducción al diseño y realización de pruebas

JUNE 15, 2021 - 15H

MARCH 22, 2021 - 15H

ÍNDICE



01

UNITARIAS

Herramientas

02

INTEGRACIÓN

03

SISTEMA

04

ACEPTACIÓN

05

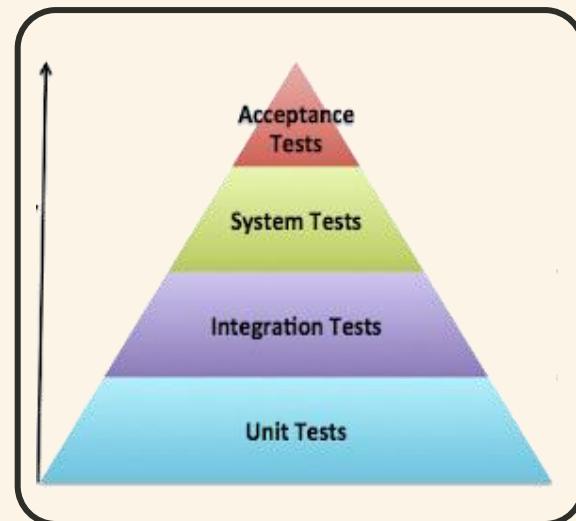
AUTOMATIZACIÓN



¿QUÉ SON LOS NIVELES DE PRUEBA? *

Son grupos de actividades de prueba que se organizan y gestionan conjuntamente.

Cada nivel de prueba es una instancia del proceso de prueba, realizadas en relación con el software en un nivel de desarrollo determinado.



OBJETIVOS GLOBALES



Generar confianza en la calidad.



Encontrar defectos.

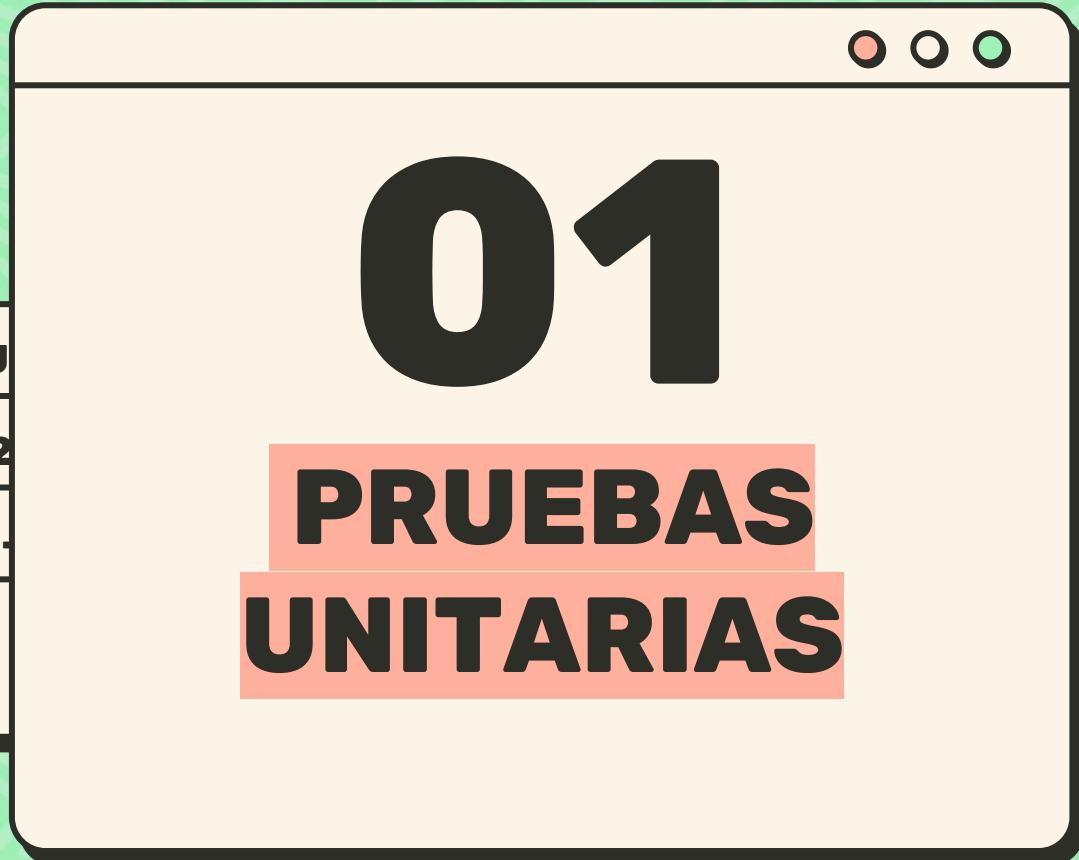
Prevenir la propagación de defectos a niveles de pruebas superiores.

Validar el comportamiento, la fiabilidad y estabilidad.

Verificar comportamientos funcionales y no funcionales del sistema.

01

PRUEBAS UNITARIAS



APRIL 15, 2021 - 18H

1 - 15H

X

X

X



Se centran en los componentes del software que se pueden probar por separado.

Las realiza el desarrollador que escribió el código.

OBJETIVO



**Comprobar que cada componente
aislado del software:**



- 1. Funciona como debe funcionar.**
- 2. Responde como debe de responder.**
- 3. Falla cómo y cuándo debe de fallar.**
- 4. Acepta lo que tiene que aceptar.**



PUEDEN CUBRIR:

**Características funcionales, por ejemplo,
exactitud de los cálculos.**

**Características no funcionales, por ejemplo,
búsqueda de fugas de memoria.**

**Propiedades estructurales, por ejemplo,
pruebas de decisión.**

OBJETOS DE PRUEBA:



**Componentes,
unidades o módulos.**



**Código y estructuras
de datos.**



Clases.



**Módulos de bases de
datos.**

UNIDAD MÍNIMA DE TRABAJO



Método o una función.



**A veces, conjunto de
métodos o de
operaciones que
producen un cambio
en el sistema o una
respuesta por parte
de este.**

ELEMENTOS NECESARIOS Y ETAPAS



ELEMENTOS

Conjunto de entradas

Unidad mínima de trabajo

Conjunto de salidas

1. Preparación o arrange:

- Se establecen los valores de entrada y el valor esperado de salida, así como cualquier otra preparación necesaria para ejecutar la prueba.

2. Ejecución o act:

- Donde se ejecuta la prueba

3. Comprobación o assert:

- Se revisa que las condiciones esperadas se hayan producido

CARACTERÍSTICAS QUE DEBEN CUMPLIR



1. Deben ser independientes del entorno.
2. Una es independiente de la ejecución de otra.
3. Es independiente de fuentes externas de datos (por ejemplo, de una base de datos).
4. Deben ser automatizables, sin requerir de supervisión.
5. Deben ejecutarse rápido (etapas de preparación y comprobación).
6. Deben ser fácilmente mantenibles.



DEFECTOS Y FALLOS CARACTERÍSTICOS

*



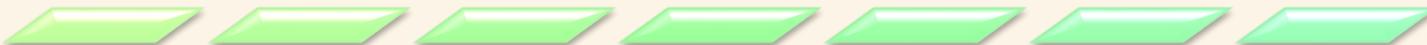
Funcionamiento incorrecto.



Problemas de flujo de datos.



Código y lógica incorrectos.



02

PRUEBAS DE INTEGRACIÓN



SEPTEMBER 15, 2021 - 18H

Meeting with Company B

*

Meeting with Company C

*

Meeting with Company D

*

QUÉ SON LAS PRUEBAS DE INTEGRACIÓN



Comprueban el funcionamiento entre dos o más sistemas a través de sus interfaces.

Verifican que los componentes de la aplicación funcionan correctamente en conjunto.

Se deben realizar al integrar un nuevo fragmento de código con algún otro componente para asegurarse de que no hay conflictos y trabajan juntos correctamente.



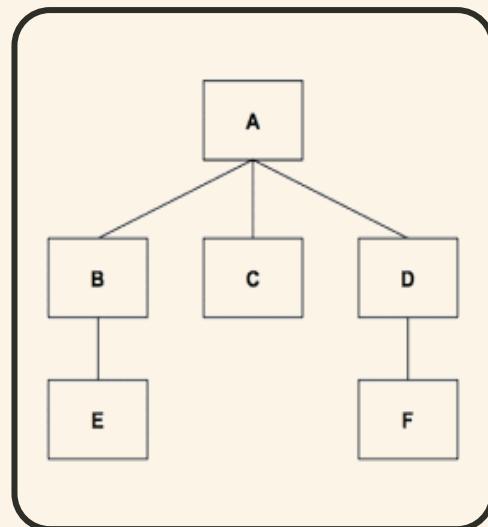


INCREMENTAL

- Añadir uno por uno los componentes y probar su funcionamiento → se prueban dos componentes y una vez probados se añade otro componente hasta integrar el conjunto

NO INCREMENTAL

- Probar cada componente por separado y luego al final unirlo con la prueba adecuada



EJEMPLO: APLICACIÓN DE TAREAS

×

OBJETIVO

PRUEBA

RESULTADOS ESPERADOS

×

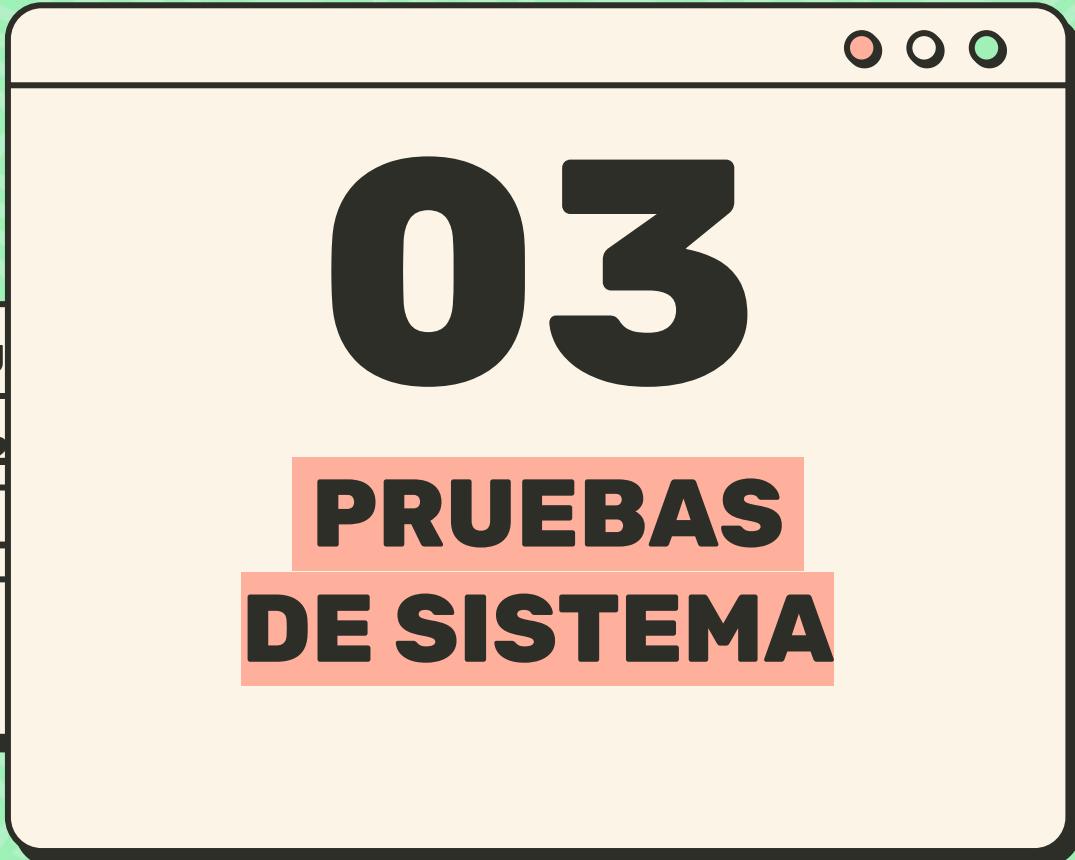
Validar la comunicación entre componente de modificación de tareas y el de la lista

Pulsar botón modificar

La aplicación dirige a la interfaz que muestra la lista de tareas abiertas

03

PRUEBAS DE SISTEMA



SEPTEMBER 15, 2021 - 18H

SEPTEMBER 15, 2021 - 15H

X

X

X

JULY 11, 2021

AUGUST 8, 2021

SEPTEMBER 15, 2021 - 15H

Meeting with Company A

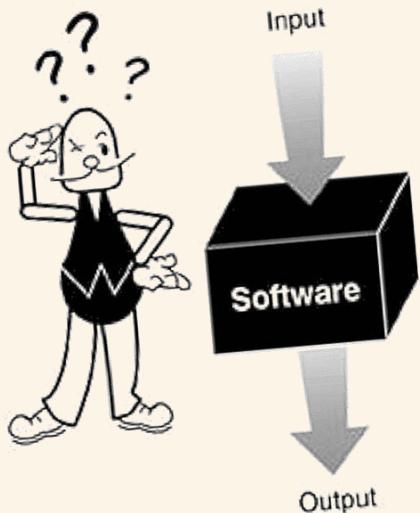


Permiten probar el sistema en su conjunto y con otros sistemas con los que se relaciona para verificar que las especificaciones funcionales y técnicas se cumplen.

Dan una visión muy similar a su comportamiento en el entorno de producción.

Similar a las pruebas de integración pero con un alcance mucho más amplio.

TÉCNICA DE CAJA NEGRA



Especialmente si quien prueba no tiene acceso al código fuente del producto a probar.

Ven el software como un recuadro negro con entradas y salidas, pero no tienen conocimiento de cómo funciona el sistema.

El probador se concentra en lo que hace el software, no en cómo lo hace.



PERMITEN ENCONTRAR:



Funciones incorrectas o ausentes



Errores de rendimiento



Errores de interfaz



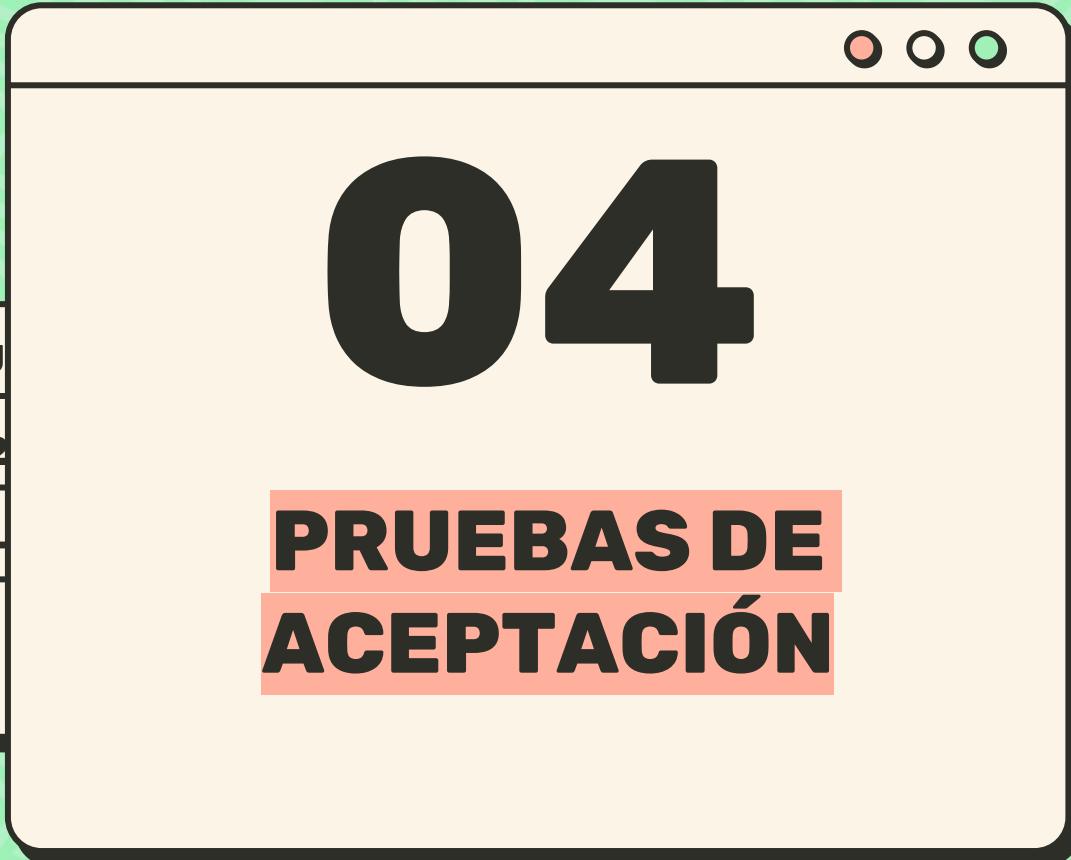
Errores de inicialización y terminación



Errores de estructuras de datos o en acceso a las BBDD externas

04

PRUEBAS DE ACEPTACIÓN



APRIL 15, 2021 - 18H

1 - 15H

X

X

X

// ¿QUÉ SON LAS PRUEBAS DE ACEPTACIÓN?



Se centra en el comportamiento y las capacidades de todo un sistema o producto...

Sus objetivos son:

Establecer confianza en la calidad del sistema

Validar que está compuesto y que funcionará como se espera.

Verificar que los comportamientos sean los especificados.



Definir las pruebas de aceptación



Pruebas de aceptación



Aplicar las pruebas de aceptación



Las formas más comunes son:

Prueba de aceptación de usuario.



Prueba de aceptación operativa.



Prueba de aceptación contractual y de regulación.



Prueba alfa y beta.





**Validación de la adecuación para el uso
del sistema por parte de los usuarios
previstos en un entorno operativo real
o simulado...**



OBJETIVO

**Crear confianza en que los usuarios pueden
usar el sistema para satisfacer sus
necesidades...**

Se centra en los aspectos operativos...



Incluye:

 **Prueba de copia de seguridad y restauración.**

 **Instalación, desinstalación y actualización.**

 **Recuperación ante desastres.**

 **Gestión de usuarios.**

 **Tareas de mantenimiento.**

 **Carga de datos y tareas de migración.**

 **Comprobación de vulnerabilidades de seguridad.**

 **Prueba de rendimiento.**



Se realiza en función de los criterios de aceptación del contrato para el desarrollo de software a medida...

OBJETIVO

Crear confianza en que se ha logrado la conformidad contractual o normativa.



Obtener retroalimentación de los usuarios, clientes y operadores potenciales o existentes antes de que el producto de software se ponga en el mercado...

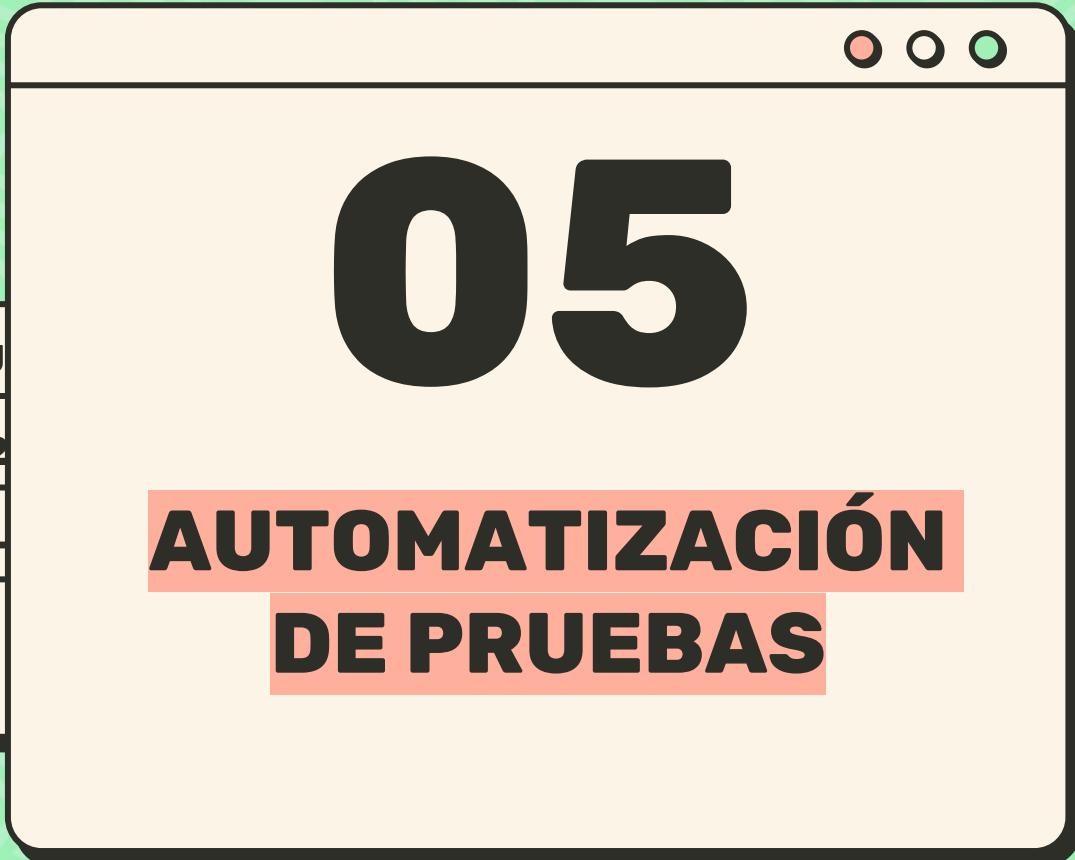
OBJETIVO

Generar confianza entre los clientes potenciales o existente y operadores de que pueden utilizar el sistema en condiciones normales y cotidianas.



05

AUTOMATIZACIÓN DE PRUEBAS

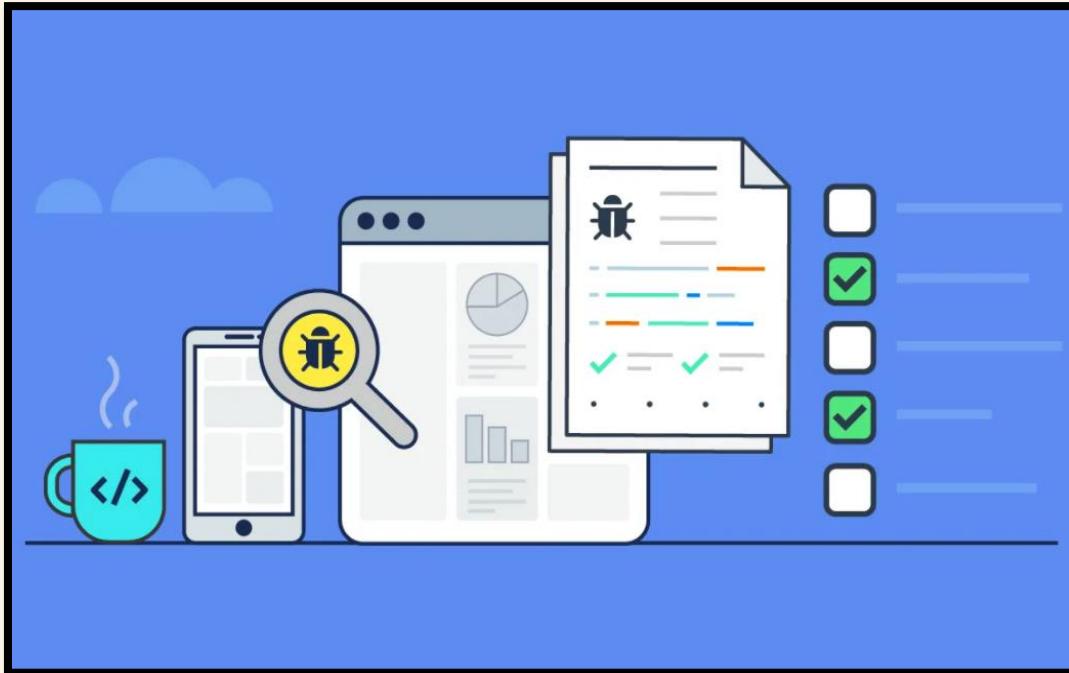


APRIL 15, 2021 - 18H



¿QUÉ ES LA AUTOMATIZACIÓN?

×



El uso de una suite de pruebas que no requiere de la constante presencia de personal QA.

×

// DIFERENCIA ENTRE LA AUTOMATIZACIÓN Y LAS PRUEBAS AUTOMÁTICAS

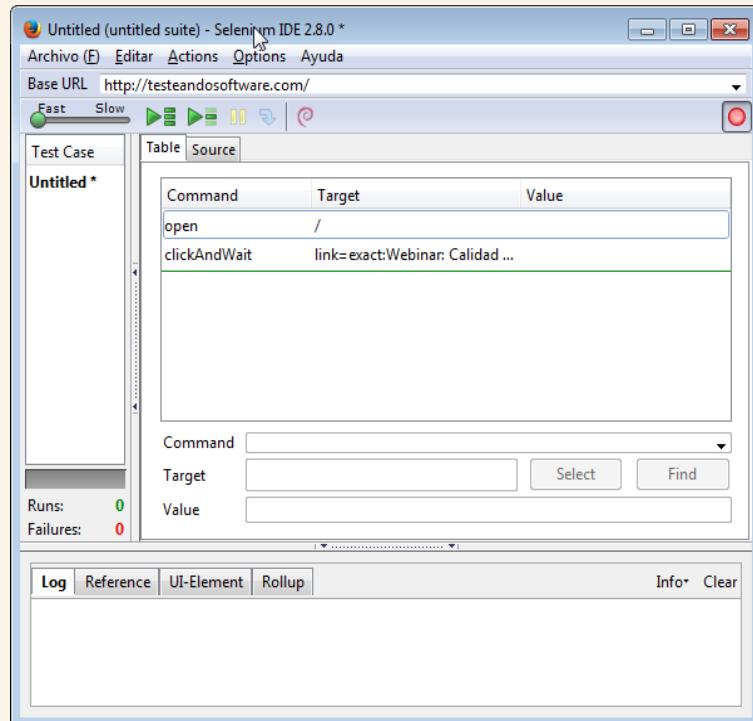


Las pruebas automatizadas no dejan de ser un check manual de codificado. Esto termina consistiendo en repetir unos pasos para verificar que la salida es correcta



La automatización de pruebas consiste en el uso de una herramienta de software para controlar y configurar las condiciones previas a las pruebas, la ejecución de las pruebas y la comprobación de los resultados reales contra los resultados esperados.

Aunque siempre hay que tener en cuenta que para que la automatización sea más efectiva y esto no sustituye las pruebas manuales.



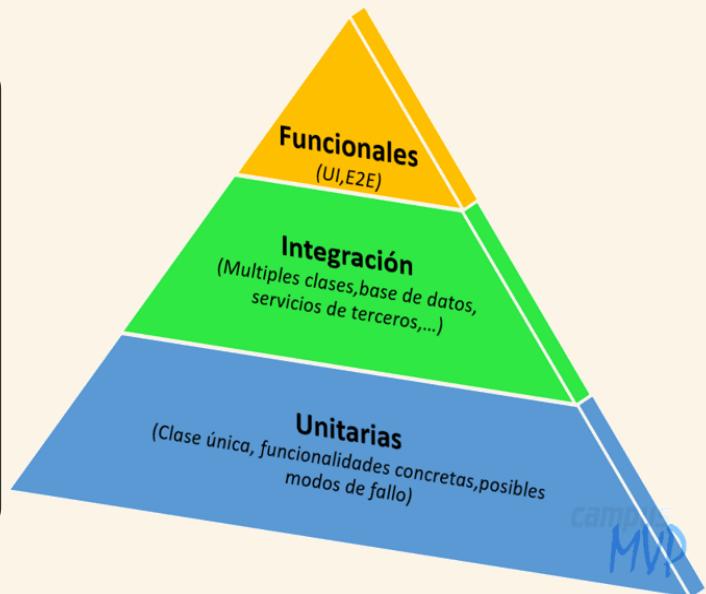
DÓNDE SE DEBE AUTOMATIZAR



La siguiente pirámide representa el ideal de cómo deberían estructurarse los distintos niveles de test.



Cuanto más subimos por la pirámide, más tiempo y esfuerzo será necesario para realizar las pruebas; por ello, en la medida de lo posible, se debería empezar a automatizar pruebas y continuar con su adaptación a lo largo del proyecto lo antes posible y concentrarlas en las pruebas unitarias o de componente.



VENTAJAS E INCONVENIENTES



VENTAJAS



Reducción de trabajo manual repetitivo.



Las pruebas siguen unos requisitos marcados.



La herramienta que se utilice suele ofrecer gráficos.

INCONVENIENTES



Se puede llegar a confiar demasiado en la herramienta



El coste inicial puede ser muy elevado



Los resultados son más visibles a largo plazo.



JULY 11, 2021 - 11H

JUNE 15, 2021 - 15H

AUGUST 8, 2021 - 16H

APRIL 15, 2

ting with C

Calidad del software

JUNE 15, 2021 - 15H

MARCH 22, 2021 - 15H

Calidad del software



01

¿Qué son las certificaciones?

02

Normas y certificaciones

03

Medidas de calidad del software

04

Herramientas para medir la calidad del software



¿Qué son las normas y las certificaciones?



La certificación es un procedimiento mediante el cual un tercero otorga una garantía escrita de que un producto, elaboración o servicio está en conformidad con ciertas normas. El *certificado* (por terceros) le demuestra al comprador que el proveedor cumple con ciertas normas, lo cual puede ser más convincente que una garantía del proveedor.

Las normas de productos son especificaciones y criterios aplicables a características de los productos. *Las normas de elaboración* son criterios relativos a la manera en que éstos deben ser fabricados.

Normas ISO



ISO/IEC 12207

Es el estándar para los procesos de ciclo de vida del software de organización.

Es la base para ISO 15504-SPICE

ISO/IEC 15504

También conocido como SPICE es un conjunto de 7 normas para mejorar la capacidad y madurez de los procesos de las organizaciones.

ISO 25000

La familia de normas 25000 establecen un modelo de calidad para el producto software además de definir la evaluación de la calidad del producto.



CMMI y SCRUM



CMMI

Convirtiéndose mundialmente en un requisito para la exportación de servicios software, la norma provee una guía para implementar una estrategia de calidad y mejorar los procesos de organizaciones dedicadas al desarrollo y/o mantenimiento de software.

SCRUM

Un método sencillo y práctico para empezar a practicar calidad. Gestiona el desarrollo en tres fases: Una breve fase de planificación en la que se realizan labores básicas, otra de desarrollo y una fase final de entrega, donde se hacen balances de éxitos y fracasos logrados.



¿Qué es el control de calidad del software?

El Control de la Calidad del Software son las técnicas y pruebas de carácter operativo encargadas de mantener bajo control los procesos y eliminar defectos.



Estas pruebas a menudo son conocidas como:

- **Verificación:** conjunto de actividades que aseguran que el software implementa correctamente una función específica
- **Validación:** actividades que aseguran que el software construido se ajusta a los requisitos del cliente.



Las distintas actividades para la implantación del control de calidad en el desarrollo de software son:

1. Aplicación y metodología y técnicas de desarrollo.
2. Reutilización de procesos de revisión formales.
3. Prueba del software.
4. Ajustes a los estándares de desarrollo.
5. Control de cambios, mediciones y recopilación de información.
6. Gestión de informes sobre el control de calidad



Desde el punto de vista de la calidad, la Gestión de la Calidad del Software está formada por 4 partes, las cuales son:

1. Planificación de la Calidad del Software.
2. Control de la Calidad del Software.
3. Aseguramiento de la Calidad del Software.
4. Mejora de la Calidad del Software.

Los principios básicos de la pruebas de software son:



1. A todas las pruebas se les debería poder hacer un seguimiento hasta los requisitos del cliente.
2. Las pruebas deberían planificarse mucho antes de que empiecen.
3. Las pruebas deberían empezar por “lo pequeño” y progresar hacia “lo grande”.
4. Para ser más eficaces, las pruebas deberían ser realizadas por un equipo independiente.

PRUEBAS DE CALIDAD



ESTRATEGIA TRADICIONAL

- **Prueba de unidad:** asegura que cada unidad funcione correctamente y eficientemente por separado.
- **Prueba de integración:** una vez que se han aprobado las pruebas unitarias, se hace una comprobación grupal del código.
- **Prueba del sistema:** verifica que cada elemento se ajusta e forma adecuada y que se alcanza la funcionalidad y el rendimiento del sistema total

- **Prueba de regresión:** vuelve a ejecutar un subconjunto de pruebas que se han llevado a cabo anteriormente para asegurarse que los cambios no han propagado efectos colaterales no deseados
- **Prueba alfa:** el cliente junto al desarrollador prueban el programa registrado así los problemas que ocurran
- **Prueba beta:** el cliente comprueba la aplicación sin ser controlada por el desarrollador, registrado todos los problemas para comunicar al desarrollador.



PRUEBAS DE CALIDAD



ESTRATEGIA CLIENTE/SERVIDOR

- **Prueba de función de aplicación:** se aprueba la funcionalidad de las aplicaciones cliente utilizando métodos
- **Pruebas de servidor:** se prueban la coordinación y las funciones de gestión de datos servidor, considerándose el rendimiento del servidor.



- **Prueba de bases de datos:** se prueba la precisión e integridad de los datos almacenados en el servidor examinando las transacciones enviadas para comprobar que los datos se almacenen, actualicen y recuperen.
- **Pruebas de transacciones:** se hacen pruebas para comprobar que todas las clases de transacciones se procesen de acuerdo con los requisitos, haciendo hincapié en la corrección de procesamiento y en los temas de rendimiento.

Herramientas para medir calidad



PMD



Es un analizador estático de código que utiliza unos conjuntos de reglas para identificar problemas dentro del software. Detecta código duplicado, código muerto o complejidad de métodos. Trabaja principalmente con lenguaje Java.

Check Style

Herramienta de análisis estático de código que comprueba que el código analizado cumple con una serie de reglas de estilo según el estándar “Sun Code Conventions”. Trabaja para Java.

Herramientas para medir calidad



Sonar

Herramienta de software libre y gratuita que permite gestionar la calidad del código fuente. Permite recopilar, analizar, y visualizar métricas del código fuente. Sonar es la fusión de diferentes herramientas para medir la calidad del software, entre ellas está CheckStyle. Permite visualizar informes con resúmenes de las métricas.

Google CodePro

Herramienta gratuita que ofrece un entorno para evaluación de código, métricas, análisis de dependencias, cobertura de código, generación de Test unitarios, etc. Mira las excepciones, refactorizaciones potenciales, convenios de JavaDoc, métricas, etc. Trabaja principalmente con eclipse mediante un plugin.

