

Repaso Integrador

DigitalHouse >
Coding School



**Certified Tech
Developer**
The Ultimate Degree

Agenda

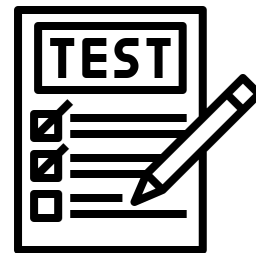
1. ¿Qué es Testing?
2. Los 7 principios de Testing
3. Ciclo de vida
4. Niveles de prueba
5. Tipos de prueba
6. Errores, defectos y fallas
7. Casos de prueba
8. Técnicas de prueba
9. Pruebas Estáticas y Dinámicas
10. Debugging
11. Tests Unitarios
12. Backend Testing
13. Automation Testing

1 | ¿Qué es Testing?

¿Qué es Testing?

Probar un software es un proceso que incluye diferentes actividades:

- Ejecución de prueba y comprobación de resultados.
- Planificación las pruebas.
- Análisis, diseño e implementación de las pruebas.
- Notificación del avance y resultado de la ejecución de pruebas.
- Evaluación de la calidad de un objeto de prueba.



2 | Los 7 principios de Testing

7 Principios del testing

La prueba muestra la presencia de defectos, no su ausencia

1



2

La prueba exhaustiva es imposible

La prueba temprana ahorra tiempo y dinero

3



4

Los defectos se agrupan

Cuidado con la prueba del pesticida

5



6

La prueba se realiza de manera diferente según el contexto

La ausencia de errores es una falacia

7



3 | Ciclo de Vida

Ciclo de Vida de Pruebas

El ciclo de vida de las pruebas de software consiste en las siguientes actividades principales:

Planificación

Implementación

Seguimiento y control

Ejecución

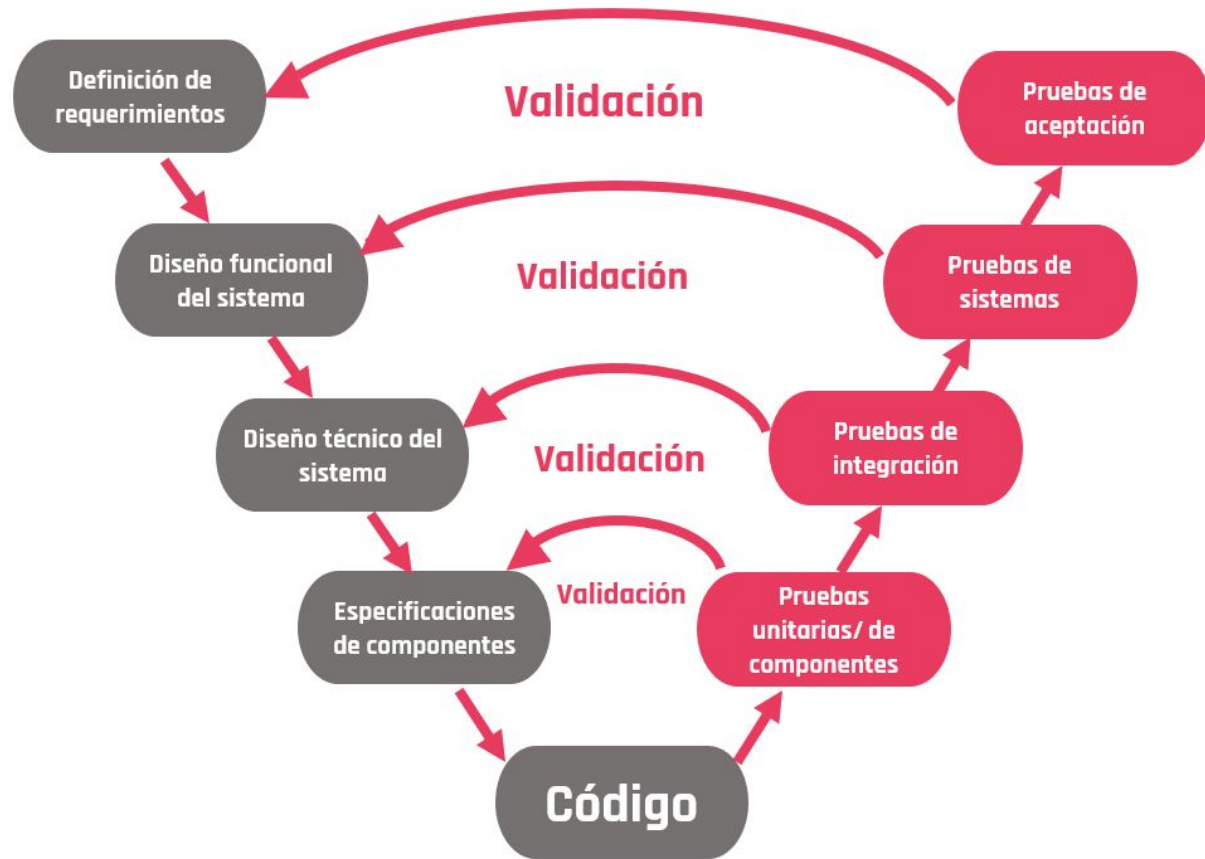
Análisis

Conclusión

Diseño

4 | Niveles de Prueba

Niveles de prueba



5 | Tipos de prueba

Prueba Funcional

- Incluye pruebas que evalúan las funciones que el sistema debe realizar. Las funciones describen qué hace el sistema.
- La prueba funcional observa el comportamiento del software.



Prueba No Funcional

- La prueba no funcional prueba “cómo de bien” se comporta el sistema
- El diseño y la ejecución de la prueba no funcional pueden implicar competencias o conocimientos especiales, como el conocimiento de las debilidades inherentes a un diseño o tecnología

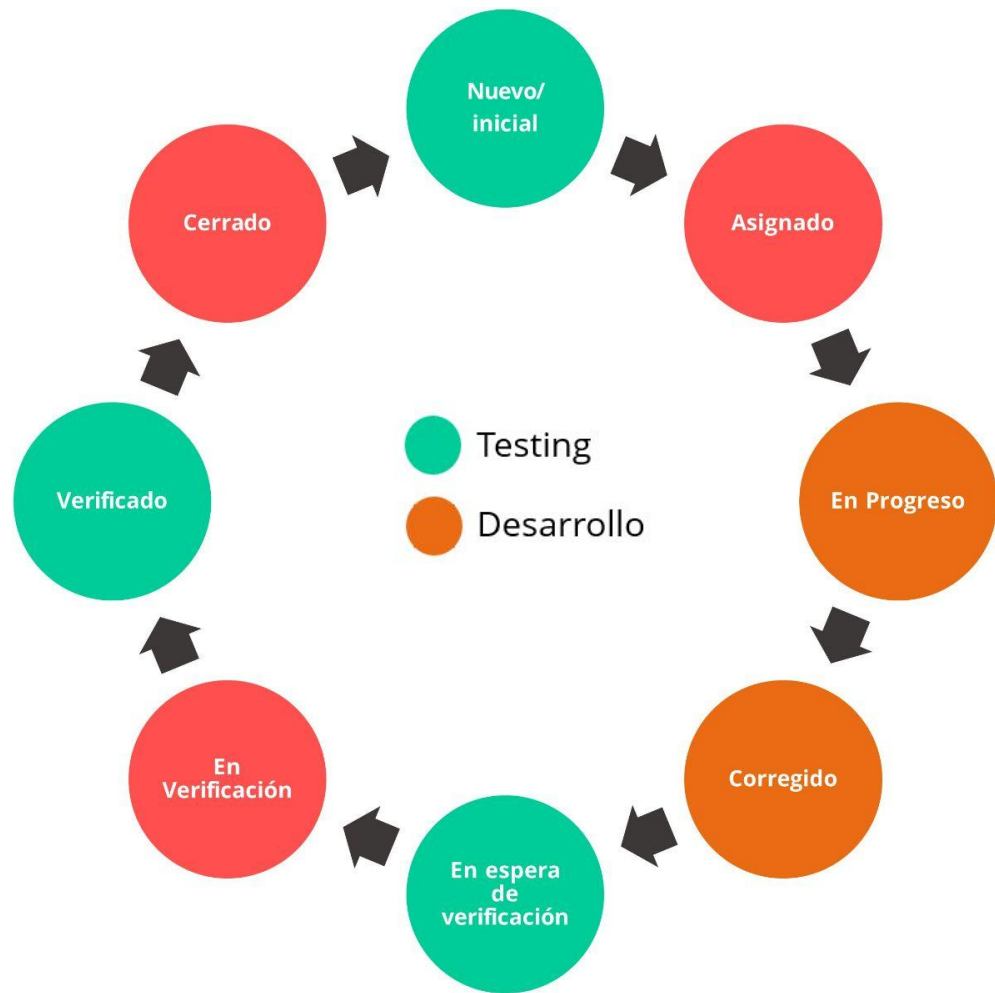


6 | Errores, defectos y fallas

Error, defecto y falla

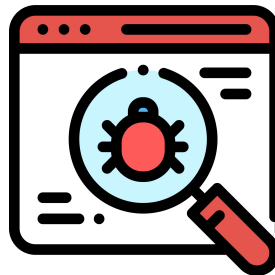


Ciclo de vida de un defecto



Partes de un informe de defectos

- ID
- Título
- Descripción
- Resultado actual
- Resultado esperado
- Pasos para reproducción
- Estado
- Prioridad
- Severidad
- Reportado por
- Asignado A

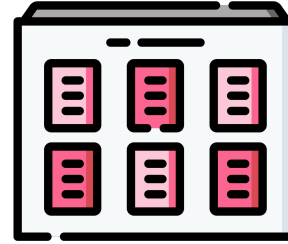


7

Casos de prueba

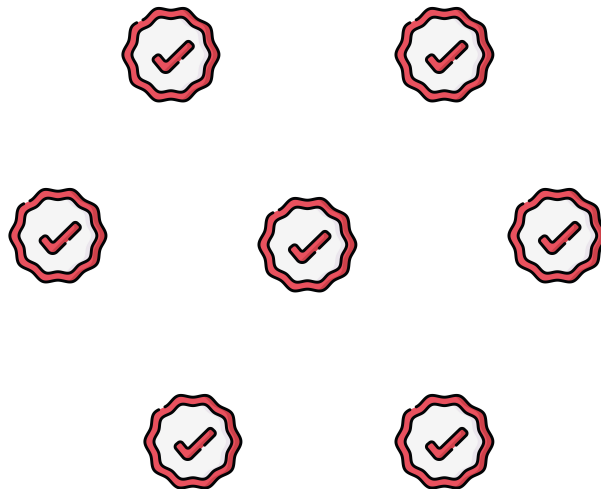
Partes de un caso de prueba

- Identificador
- Nombre del casos de prueba
- Descripción
- Precondición
- Pasos
- Resultado esperado
- Estado



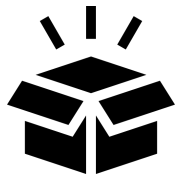
Un buen caso de prueba debe:

- Ser simple
- Tener un título fuerte
- Tener en cuenta al usuario final
- No asumir
- Asegurar la mayor cobertura posible
- Tener autonomía
- Ser único

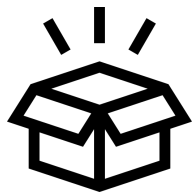


8 | Técnicas de prueba

Clasificación



Técnicas de caja negra



Técnicas de caja blanca

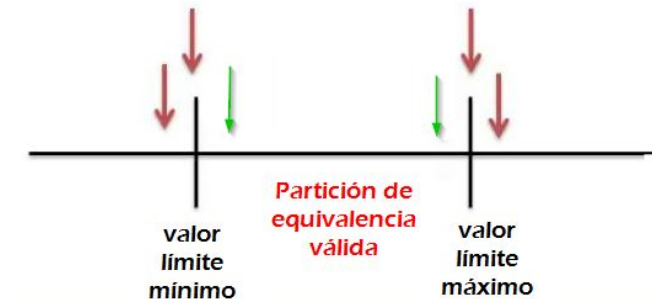


Técnicas basadas en la experiencia

Caja Negra



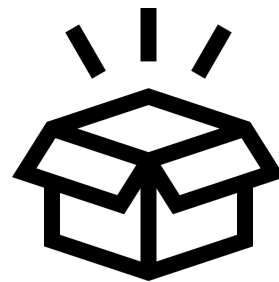
- Partición de equivalencia
- Análisis de valores límites
- Tabla de Decisión
- Transición de Estados



| Condiciones | Reglas | | | | | | | |
|-------------|--------|---|---|---|---|---|---|---|
| Condición 1 | S | S | S | S | N | N | N | N |
| Condición 2 | S | S | N | N | S | S | N | N |
| Condición 3 | S | N | S | N | S | N | S | N |
| Acción 1 | X | X | | | | | | |
| Acción 2 | | | | X | | X | | X |
| Acción 3 | | | X | | | | X | |
| Acción 4 | | | | | X | | | |

Caja Blanca

- Prueba y cobertura de sentencia
- Prueba y cobertura de decisión



Basadas en la experiencia

- Predicción de errores
- Prueba Exploratoria
- Prueba basada en listas de comprobación



9 | Pruebas estáticas y dinámicas

Comparaciones

| Prueba estática | Prueba dinámica |
|--------------------------------------------------------------------------------|--------------------------------------------------------------|
| Detecta los defectos en productos de trabajo, sin ejecutar el software. | Detecta los defectos y fallas cuando se ejecuta el software. |
| Se centra en mejorar la consistencia y la calidad de los productos de trabajo. | Se centra en los comportamientos visibles desde el exterior. |
| El costo de solucionar un defecto es menor | El costo de solucionar un defecto es mayor |

10 | Debugging

Debugging

Debuggear o depurar al proceso de encontrar, analizar y remover las causas de fallos en el software.

Un **breakpoint** es una pausa en nuestro código para entender en qué estado están nuestras variables.

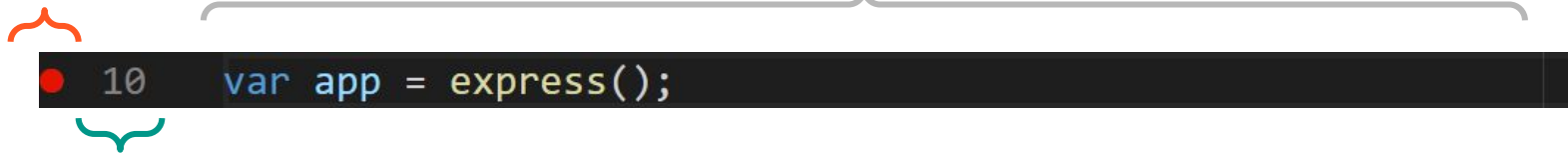
Breakpoints

Breakpoint

Al hacer clic a la izquierda del número de línea, establecemos un breakpoint.

Contenido

El contenido de nuestro código.



Número de línea

Nos indica el número de línea en el que está el código.

11 | Unit Testing

Pruebas unitarias

El objetivo principal es aislar cada **unidad** del sistema para identificar, analizar y corregir los defectos.

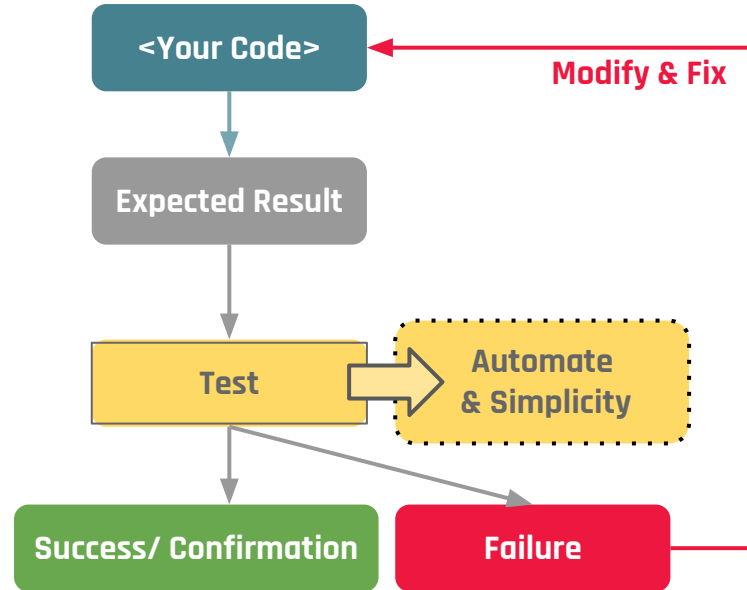
Ventajas:

- Reducen costo de pruebas
- Mejora el diseño y el código del software
- Reduce defectos en cambios recientes
- Pruebas de regresión de componentes construyen confianza en la calidad de los entregables

Proceso

En general, el proceso es el siguiente:

1. Se crea el código del software.
2. Se definen los resultados esperados.
3. Se ejecuta el test.
 - a) Si el test pasa, se confirma el resultado esperado.
 - b) Si el test falla, se modifica el código para solucionar el defecto encontrado



Lo ideal es automatizar los test para poder simplificar el proceso de prueba.

Automatización

- Test Runner
- Assertion Library

JEST es un framework que incluye tanto el test runner como la assertion library

Frameworks

- JUnit
- Nunit
- JMockit



- EMMA
- PHPUnit



12

BackEnd Testing

BackEnd testing

Nos garantiza que los datos contenidos en la base de datos de una aplicación y su estructura satisfagan los requisitos del proyecto.

API Testing

Estas pruebas consisten en hacer peticiones HTTP (GET,POST,PUT,DELETE) y luego verificar la respuesta.

- Se utiliza POSTMAN



Postman

Se observa que al momento de enviar la petición se ejecutan todos los test relacionados a la misma. De esta manera, podemos crear un set de test vinculados a cada petición y verificar de manera rápida el estado de los mismos.

The screenshot displays the Postman interface for a GET request to `https://jsonplaceholder.typicode.com/users`. The 'Tests' tab is selected, showing two test scripts:

```
1 pm.test("Status code is 200", function () {  
2   pm.response.to.have.status(200);  
3 });  
4  
5 pm.test("Verificar si Leanne Graham tiene el ID de usuario 1", function () {  
6   var jsonData = pm.response.json();  
7   pm.expect(jsonData[0].name).to.eql("Leanne Graham");  
8 });  
9  
10
```

On the right side, there is a section for 'Test scripts' with a link to 'Learn more about tests scripts' and a 'SNIPPETS' section listing various checks like 'Response body: JSON value check', 'Response body: is equal to a string', 'Response headers: Content-Type header check', 'Response time is less than 200ms', and 'Status code: Successful POST request'.

At the bottom, the 'Test Results' section shows two tests, both marked as 'PASS':

- PASS Status code is 200
- PASS Verificar si Leanne Graham tiene el ID de usuario 1

The status bar at the bottom indicates 'Status: 200 OK', 'Time: 291 ms', 'Size: 6.64 KB', and a 'Save Response' button.

13 | Automation Testing

Tipos de pruebas automatizadas

Se pueden automatizar los siguientes tipos de pruebas:

- Pruebas unitarias.
- Pruebas de API.
- Pruebas de interfaz gráfica.

Otras pruebas que también pueden automatizarse son las de rendimiento, de regresión, de integración, de seguridad, pruebas de compatibilidad en diferentes navegadores, casos repetitivos.

Selenium

- Es uno de los frameworks más importantes con los que se generan pruebas automatizadas.
- Permite a los usuarios simular interacciones realizadas por los usuarios finales.



Selenium Grid



Selenium WebDriver



Selenium IDE

DigitalHouse>
Coding School