SESIÓN 18
Pruebas de integración, regresión y no funcionales

Hasta ahora hemos trabajado con:

- **Pruebas unitarias:** pequeñas partes del código.
- **Verificación:** comprobar que el software está bien construido.
- **Validación**: comprobar que el software cumple lo que el usuario necesita.
- **Caja blanca:** revisar la lógica interna.
- **Caja negra:** revisar entradas y salidas.

En esta sesión damos un paso más.

Cuando un sistema crece, deja de ser un conjunto de métodos aislados y pasa a ser un conjunto de módulos que interactúan.

Ahí aparecen los problemas más importantes.

Por eso surgen tres tipos de pruebas avanzadas:

1. **Pruebas de integración**
2. **Pruebas de regresión**
3. **Pruebas no funcionales**

PRUEBAS DE INTEGRACIÓN

Las pruebas de integración verifican que los diferentes módulos o componentes del sistema funcionan correctamente cuando trabajan juntos.
No evalúan piezas aisladas, sino su interacción.

Por qué son necesarias
Muchos errores no aparecen dentro de un método, sino cuando:

- Una clase llama a otra.
- Se intercambian datos.
- Se combinan resultados.
- Se accede a recursos compartidos.

Un módulo puede estar perfectamente programado, pero si recibe datos incorrectos de otro módulo, el sistema fallará.

Qué comprueban

- Que los datos se transmiten correctamente entre clases.
- Que las llamadas entre métodos funcionan.
- Que no existen incompatibilidades.
- Que el flujo completo del sistema es coherente.

Enfoques teóricos de integración

Integración ascendente (Bottom-Up)

Se prueban primero los módulos más pequeños y luego se van combinando progresivamente hasta llegar al sistema completo.
Ventaja:

- Permite validar bien las capas internas.

Integración descendente (Top-Down)

Se empieza probando los módulos superiores (los que controlan el flujo principal) y se simulan los inferiores.

Ventaja:

- Permite validar pronto la lógica general del sistema.

Integración continua

En entornos modernos, cada vez que se modifica el código, se ejecutan automáticamente las pruebas de integración.

- Esto evita que se incorporen errores al sistema principal.

Diferencias

Prueba unitaria	**Prueba de integración**
Método aislado	Comunicación entre módulos
Usa datos simulados	Usa objetos reales
Más rápida	Más compleja

PRUEBAS DE REGRESIÓN

Definición formal

Las pruebas de regresión son un conjunto de pruebas que se ejecutan después de cada modificación del sistema para garantizar que las funcionalidades que ya funcionaban siguen funcionando correctamente.

Problema que resuelven

Cada vez que se modifica el código:

- Se añade una funcionalidad.
- Se corrige un error.
- Se mejora una parte del sistema.

Existe el riesgo de romper algo anterior. Eso se llama regresión.

Objetivo principal

Garantizar la estabilidad del sistema tras cambios.

Características

- Se basan en pruebas ya existentes.
- Se ejecutan de forma repetitiva.
- Deben estar automatizadas.
- Se ejecutan en cada nueva versión.

Relación con CI/CD

- **CI (Integración Continua):** Cada vez que se sube código, se compila y se ejecutan automáticamente las pruebas.
- **CD (Despliegue/Entrega Continua):** Si todo pasa correctamente, el sistema puede desplegarse automáticamente.

Las pruebas que hemos visto se integran dentro del proceso automático:

1. El desarrollador hace un cambio.
2. Se sube al repositorio (Git).
3. El sistema ejecuta automáticamente:
    - Pruebas unitarias
    - Pruebas de integración
    - Pruebas de regresión
    - (En entornos avanzados) pruebas de rendimiento o seguridad
4. Si algo falla: el despliegue se bloquea.

- Sin CI/CD: Las pruebas dependen de que alguien las ejecute.
- Con CI/CD: Las pruebas se ejecutan solas cada vez que hay un cambio.
- CI/CD convierte las pruebas en un control automático de calidad.
  No se despliega nada que esté roto.

En entornos profesionales:

1. Se hace un cambio.
2. Se ejecutan todos los tests.
3. Si alguno falla: no se despliega.

Esto permite mantener el sistema estable en el tiempo.

Diferencia importante

- **Integración**: verifica interacción.
- **Regresión**: verifica estabilidad tras cambios.

PRUEBAS NO FUNCIONALES

Hasta ahora las pruebas respondían a: ¿Hace lo que debe hacer?
Las pruebas no funcionales responden a: ¿Cómo lo hace?

- No evalúan funciones concretas, sino atributos globales de calidad.

Tipos principales

1. Pruebas de rendimiento

Evalúan:

- Tiempo de respuesta.
- Capacidad bajo carga.
- Consumo de recursos.
- Estabilidad prolongada.

Subtipos:

- Load test: comportamiento con muchos usuarios.
- Stress test: comportamiento en situaciones extremas.
- Soak test: comportamiento tras muchas horas de uso.

2. Pruebas de seguridad

Detectan vulnerabilidades como:

- Inyección SQL.
- Errores de autenticación.
- Accesos no autorizados.
- Fugas de datos.

Un sistema puede funcionar correctamente y aun así ser inseguro.

3. Pruebas de usabilidad y compatibilidad

Evalúan:

- Facilidad de uso.

- Claridad de mensajes.
- Comportamiento en distintos entornos.
- Experiencia del usuario.

Diferencia clave con las funcionales

Funcionales	**No funcionales**
Verifican qué hace el sistema	Evalúan cómo lo hace
Basadas en requisitos	Basadas en calidad
Resultado correcto	Resultado eficiente y seguro

RELACIÓN ENTRE LOS TRES TIPOS

Podemos entenderlo como capas:

1. **Integración:** asegura que las piezas encajan.
2. **Regresión:** asegura que no se rompen al cambiar.
3. **No funcionales:** asegura que el sistema es profesional.

Juntas forman la capa final antes del despliegue en producción.

IMPORTANCIA PROFESIONAL

En proyectos pequeños puede parecer que no son necesarias.
Pero en sistemas reales:

- Un error de integración puede bloquear un sistema completo.
- Una regresión puede afectar a miles de usuarios.
- Un problema de rendimiento puede tumbar un servidor.
- Una vulnerabilidad puede provocar una brecha de seguridad.

Por eso estas pruebas marcan la diferencia entre:
Software que funciona y Software profesional

RESUMEN

- Pruebas de integración:

    - Verifican que los módulos trabajan correctamente juntos.
- Pruebas de regresión:
    - Aseguran que los cambios no rompen lo anterior.
- Pruebas no funcionales:
    - Evalúan rendimiento, seguridad y experiencia global.
- La calidad no se comprueba al final.
    - Se construye y se mantiene constantemente.