SESIÓN 16: Introducción a las pruebas, verificación vs. validación

LA CALIDAD EN SOFTWARE: UNA IDEA MAL ENTENDIDA

En programación, solemos confundir **calidad** con **ausencia de errores**.
Eso es un error conceptual.

Un software de calidad no es el que "no falla", sino el que:

- Cumple su objetivo
- Se comporta como se espera
- Puede mantenerse y evolucionar sin romperse

La calidad no es un estado, es un **grado de confianza**.
Por eso, en ingeniería de software no se habla de "software perfecto", sino de **software suficientemente fiable para su contexto**.
Aquí nace el testing.

QUÉ ES PROBAR SOFTWARE

Probar software es un **proceso sistemático** que tiene como objetivo:

- Detectar defectos
- Reducir el riesgo
- Aumentar la confianza en el sistema

No se trata de ejecutar el programa "a ver qué pasa".
Se trata de **formular hipótesis sobre su comportamiento** y comprobarlas.

En testing no preguntamos:

- ¿Funciona?

Preguntamos:

- ¿Qué puede fallar?
- ¿En qué condiciones?
- ¿Qué impacto tendría?

EL ERROR COMO CONCEPTO TÉCNICO

Antes de hablar de pruebas, hay que aclarar algo fundamental:

En software existen **tres niveles de problema**:

1. **Error**: equivocación humana (mal diseño, mal razonamiento)
2. **Defecto (bug)**: error materializado en el código
3. **Fallo**: comportamiento incorrecto observable en ejecución

Las pruebas no eliminan errores humanos.

Las pruebas **detectan defectos antes de que provoquen fallos**.

El objetivo del testing es **romper la cadena** antes de que llegue al usuario.

VERIFICACIÓN: CONTROL DE CALIDAD INTERNO

La **verificación** es el conjunto de actividades que comprueban que el software:

- Está correctamente implementado
- Cumple las especificaciones técnicas
- Sigue el diseño previsto
- **¿Estamos construyendo el producto correctamente?**

Es un proceso **interno**, técnico y objetivo.

Qué se verifica realmente
Cuando verificamos, comprobamos:

- Que los algoritmos son correctos
- Que las estructuras de control funcionan
- Que los datos se procesan bien
- Que los módulos cumplen su contrato

Importa si **el software hace lo que se dijo que haría, técnicamente**.

Naturaleza de la verificación

- Se realiza durante el desarrollo
- Se apoya en:
  - Pruebas unitarias
  - Pruebas de integración
  - Revisiones de código
- Es repetible y automatizable

La verificación busca **corrección técnica**, no utilidad.

VALIDACIÓN: CONTROL DE CALIDAD EXTERNO

La **validación** es el conjunto de actividades que comprueban que el software:

- Resuelve el problema real
- Cumple las necesidades del usuario
- Tiene sentido en su contexto de uso
- **¿Estamos construyendo el producto correcto?**

Es un proceso **externo**, funcional y contextual.

Qué se valida realmente

Cuando validamos, comprobamos:

- Que los flujos de uso son adecuados
- Que las funcionalidades aportan valor
- Que el comportamiento es comprensible
- Que el producto cumple expectativas reales

Aquí **no importa cómo está hecho el código**.
Importa **qué experiencia genera**.

Naturaleza de la validación

- Se realiza cuando el sistema es usable
- Involucra:
  - Usuarios
  - Clientes
  - Testers funcionales
- No siempre es automatizable
- Tiene componente subjetivo

RELACIÓN ENTRE VERIFICACIÓN Y VALIDACIÓN

- La verificación es **condición necesaria**
- La validación es **condición suficiente**

Un software:
- No puede validarse sin estar verificado
- Puede estar verificado y no validado

Verificación sin validación = software inútil

Validación sin verificación = software inestable

Por eso **no son alternativas**, son complementarias.

LOS NIVELES DE PRUEBA COMO CONSECUENCIA LÓGICA

Los distintos tipos de pruebas no existen por capricho.

Existen porque **el software tiene distintos niveles de abstracción**.

- Nivel código: pruebas unitarias
- Nivel módulos: pruebas de integración
- Nivel sistema: pruebas de sistema
- Nivel usuario: pruebas de aceptación

Cada nivel responde a **una pregunta distinta**:
- ¿Funciona esta pieza?
- ¿Encajan las piezas?
- ¿Funciona el todo?
- ¿Sirve para quien lo usa?

Cada nivel reduce un tipo de riesgo diferente.

LA CULTURA DE LA CALIDAD

Un error muy común es pensar que, el testing es cosa del QA.

En realidad:

- El desarrollador es el **primer responsable de la calidad**
- El tester no “arregla” software
- El tester **evalúa riesgos**

La calidad no se añade al final.

Se construye desde:

- El diseño
- El código
- Las decisiones técnicas

El software es una hipótesis.
Las pruebas son el experimento que decide si esa hipótesis es fiable.