6.I. Testing en Spring

Acceso a Datos

Alejandro Roig Aguilar
a.roigaguilar@edu.gva.es

IES Álvaro Falomir
Curso 2025-2026

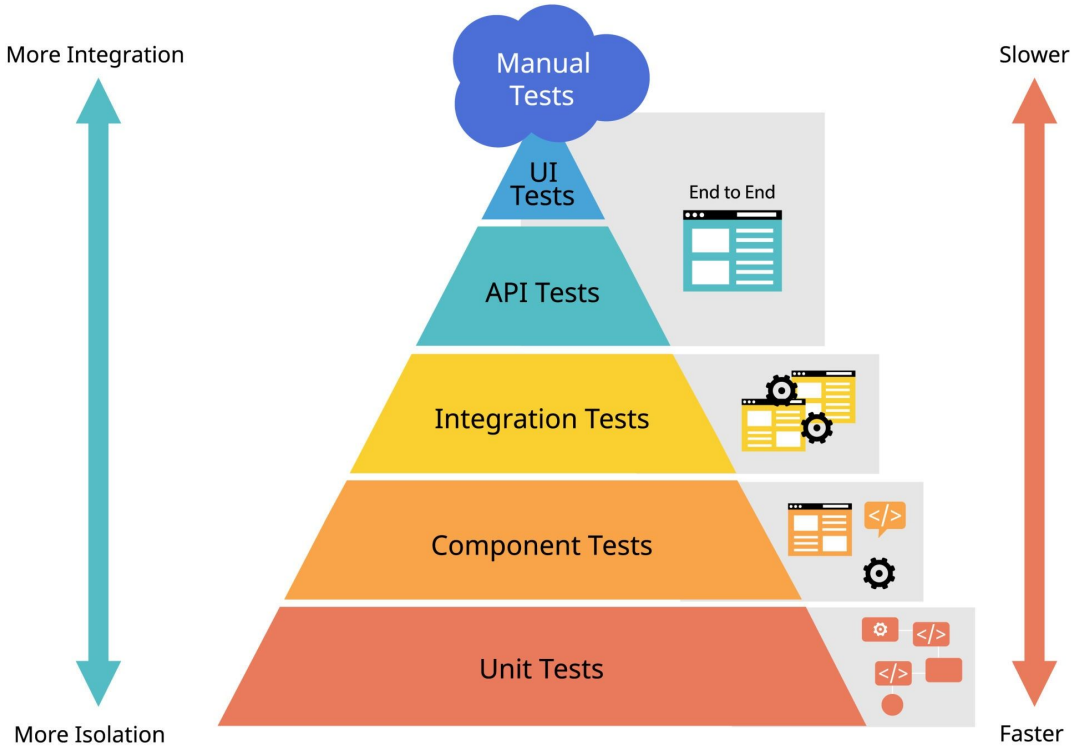
Calidad del software

Escribir software libre de defectos es prácticamente imposible de alcanzar.

Las pruebas nos permiten verificar que el código cumple con las especificaciones en ciertas circunstancias antes de llegar a producción.

Dijkstra: “Las pruebas de software prueban la presencia de errores, no su ausencia”.

Pirámide de pruebas



Pirámide de pruebas

La pirámide de pruebas propone la organizar las pruebas en capas según su cantidad y velocidad:

- Pruebas End-to-End (E2E). Evalúan el flujo completo desde la perspectiva del usuario (lentas)
- Pruebas de integración. Validan la interacción entre múltiples componentes.
- Pruebas unitarias. Pruebas de unidades pequeñas en aislamiento (rápidas).

Google recomienda una distribución de 70% pruebas unitarias, 20% pruebas de integración, y 10% de pruebas e2e.

Dobles de prueba (Mocks)

Los dobles de prueba (test doubles) sustituyen componentes reales del para simular sus comportamientos.

Un Mock es objeto preprogramado para devolver valores específicos o lanzar excepciones durante el test.

Es esencial para sustituir dependencias reales y poder así aislar el componente que estamos probando (SUT) de ellas.

Testing en el ecosistema Spring

s herramientas necesarias:

- JUnit 5 – El framework principal para ejecutar pruebas
- Mockito – El motor para crear y gestionar los mocks
- AssertJ – Librería para escribir aserciones legibles y fluidas

Estrategias de Testing por Capa

En Spring, no siempre queremos cargar toda la aplicación para probar una sola cosa.

- Unit Test (Service): Usamos @ExtendWith(MockitoExtension.class) + @Mock + @InjectMocks. Sin contexto de Spring (máxima velocidad).
- Persistence Test (Repository): Usamos @DataJpaTest. Solo carga la base de datos (H2) y los repositorios.
- Web Test (Controller)– Usamos @WebMvcTest + MockMvc. Prueba los endpoints sin levantar el servidor.

Ciclo de Vida y Metodologías

¿Cuándo aparece la fase de pruebas?

- Tradicionalmente, después de la implementación.
- TFD (Test First): Escribir las pruebas antes de la implementación.
- TDD (Test Driven): Las pruebas guían el diseño y desarrollo del código.
- BDD (Behaviour Driven): Pruebas de alto nivel basadas en el comportamiento y requisitos del usuario.