ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA DE SOFTWARE

VALIDACIÓN Y VERIFICACIÓN DE SOFTWARE

OCTAVO NIVEL

Ing. Natalia Patricia Layedra Larrea, Mg.

nlayedra@espoch.edu.ec

Periodo académico
Septiembre 2025 – Febrero 2026

UNIDAD II

PRUEBAS UNITARIAS E INTEGRACIÓN

GESTIÓN DE DEPENDENCIAS

- ¿Qué ocurre cuando un módulo depende de otro para funcionar?
- ¿Cómo probamos algo que aún no existe o no está terminado?

GESTIÓN DE DEPENDENCIAS

- **Módulo**
  - Un **módulo** en el contexto del desarrollo de software es una **unidad funcional independiente y reutilizable** que contiene una parte del código del sistema. Puede ser una **función, archivo, clase o componente** que realiza una tarea específica y puede integrarse con otros módulos para formar un sistema más complejo.

GESTIÓN DE DEPENDENCIAS

- **Prueba unitaria vs. prueba de integración**

Característica	Prueba Unitaria	Prueba de Integración
Nivel	Bajo (función, método)	Medio (módulos, componentes)
Alcance	Una sola unidad	Interacción entre unidades
Dependencias	Ninguna o simuladas (mocks)	Reales o parcialmente simuladas
Velocidad	Muy rápida	Más lenta
Objetivo	Validar lógica interna	Validar comunicación entre módulos

¿Cuándo usar cada una?

- **Pruebas unitarias**: desde el inicio del desarrollo, para verificar piezas de código.
- **Pruebas de integración**: cuando se empieza a conectar módulos o servicios entre sí.

GESTIÓN DE DEPENDENCIAS

¿Qué es una arquitectura modular?

- División del sistema en **módulos independientes pero cooperativos**.
- Módulos pueden ser funciones, clases, servicios, APIs, etc.

¿Qué son las dependencias?

- **Relaciones entre módulos**: cuando un módulo necesita que otro le proporcione datos o funcionalidad.
- Tipos de dependencias:
  - Internas (dentro del sistema)
  - Externas (librerías, APIs, BD, servicios de terceros)

GESTIÓN DE DEPENDENCIAS

- **Problemas comunes:**
  - Acoplamiento fuerte: cambios en un módulo afectan a otros.
  - Dificultad para probar módulos de forma aislada.
  - Dependencias no disponibles o no determinísticas.
- **Soluciones:**
  - Inversión de dependencias (Inversión de Control, IoC)
  - Mocking / Stubbing: crear reemplazos simulados de los módulos dependientes.
- **Herramientas útiles:**
  - Python: unittest.mock
  - Java: Mockito
  - C#: Moq