Laboratorio 4: Pruebas Unitarias y CI/CD

Introducción

El estudiante recibirá un proyecto de ejemplo en Python que simula un sistema de combate para videojuegos. Este laboratorio es individual, ya que se busca que todos se familiaricen con el proceso de inicialización de testing y pipelines.

Se les presentará una guía para crear una base de un proyecto de pruebas. Posteriormente, se le solicitará hacer tests y se les guiará en el proceso de realizar uno aprovechando del uso de interfaces, siguiendo lo que nos propone SOLID, especialmente con SOLI**D** - **Dependency Injection**.

Contexto del Proyecto

Este proyecto es una simulación modular de un sistema de combate para videojuegos, orientado a buenas prácticas de diseño de software, en particular **los principios SOLID**.

El objetivo es demostrar cómo diseñar sistemas desacoplados mediante **interfaces e inyección de dependencias**, lo cual permite construir servicios fácilmente testeables y mantenibles.

Pruebas Incluidas

Tipo de prueba	Descripción
`DummyWeapon`	Simula un arma simple, útil para probar la clase `CombatSystem` sin lógica real.
`MockDamageCalculator`	Simula el cálculo de daño para verificar cómo se comportan las armas ante diferentes escenarios.
`MagicMock`	Permite validar si se llamó a los métodos correctos con los parámetros adecuados.
Integración con Dummies	Valida que `CombatSystem` funciona correctamente cuando se le inyectan implementaciones simuladas de `Weapon`.

Estas pruebas demuestran que podemos cambiar las dependencias por versiones de prueba sin modificar el código de producción.

Prerequisitos para instalación

1. Consigue los archivos base

En la versión más reciente del repositorio del curso podrá encontrar los archivos base para realizar los tests. Presentan lógica de negocio y algunos tests ya existentes. Copie el código y paselo a un repositorio personal.

Desarrollo

Extensión del sistema y pruebas unitarias aplicando inyección de dependencias

Visualice el código y como se pueden realizar distintos tipos de tests. Identifique features que pueda agregar (de cualquier tipo, no hay restricciones ni en la lógica de negocio ni en los tests). Debe agregar al menos un archivo nuevo que permita realizar un comportamiento distinto del programa. Y agregue tests que sean interesantes y de alto valor para el programa para esta nueva funcionalidad.

Cómo ejecutar los tests?

Una manera de ejecutar todos los tests es desde el path de `lab4_tests_example/`, corriendo el comando: `python -m unittest discover -v`.

Creación de un CI/CD pipeline

Investigue cómo hacer un pipeline CI/CD en github para que pueda ejecutar estas pruebas automatizadas cada vez que se realice un commit.

Entregable

Suba a Mediación Virtual el link del repositorio donde se puede apreciar el código extendido y el pipeline ejecutando los tests.