JalaUniversity

Programación IV

Lab Week2

Name: Aldo Ochoa Condori

Tutor: Luis Morales Ledezma

Topic: Pruebas en Programación Funcional y Conceptos Relacionado

28/07/2024

Materia: Algorítmica I
Estudiante: Aldo Ochoa Condori

Actividad #3

Descripción de la actividad:

¿Describe la diferencia entre pruebas unitarias, de integración y e2e?

Pruebas unitarias: Se centran en componentes individuales o funciones de forma aislada. Verifican que cada parte del código funcione correctamente por sí sola. Son rápidas de ejecutar y ayudan a identificar problemas específicos en el código.

Pruebas de integración: Comprueban cómo diferentes partes del sistema trabajan juntas. Aseguran que varios componentes interactúen correctamente cuando se combinan. Son útiles para detectar problemas que surgen de la interacción entre componentes.

Pruebas e2e (end-to-end): Evalúan el sistema completo de principio a fin, simulando escenarios de usuario reales. Garantizan que la aplicación funcione como se espera en un entorno similar al de producción. Son las más complejas y lentas, pero ofrecen una visión global del funcionamiento del sistema.

¿Qué tipos de pruebas deberíamos priorizar al desarrollar un sistema y por qué?

Priorización de pruebas al desarrollar un sistema:

Se recomienda generalmente priorizar las pruebas en este orden:

1. Pruebas unitarias
2. Pruebas de integración
3. Pruebas e2e

Razones:

- Las pruebas unitarias son rápidas de escribir y ejecutar, proporcionando retroalimentación inmediata sobre pequeños cambios en el código.
- Las pruebas de integración detectan problemas que surgen cuando los componentes interactúan, que las pruebas unitarias podrían pasar por alto.
- Las pruebas e2e son valiosas, pero más complejas y requieren más tiempo para mantenerse, por lo que suelen ser menos numerosas.

Este enfoque forma una "pirámide de pruebas" con muchas pruebas unitarias en la base, menos pruebas de integración en el medio y un pequeño número de pruebas e2e en la cima.

¿Cuál es la cobertura de código ideal que debería tener un sistema?

Aunque a menudo se cita el 100% de cobertura de código como objetivo, no siempre es práctico o necesario. Un objetivo generalmente aceptado es:

70-80% de cobertura de código

Tutor: Luis Morales

Este nivel típicamente indica una base de código bien probada sin ser excesivamente difícil de mantener.

¿Qué biblioteca es el estándar para la generación de cobertura de código en Node.js?

La biblioteca estándar para generar cobertura de código en Node.js es:

Istanbul (nyc)

Istanbul es ampliamente utilizada y está integrada con la mayoría de los frameworks de pruebas de JavaScript, proporcionando informes detallados sobre la cobertura de código

Jest

Jest, además de ser un framework de pruebas, también incluye capacidades integradas para la generación de informes de cobertura de código. Es particularmente popular en proyectos de React y otros frameworks modernos de JavaScript, pero se usa ampliamente en todo tipo de proyectos Node.js