El Momento de las Pruebas

Ayer, me propuse crear una herramienta de configuración automática para Shadowsocks Outline, con el objetivo de convertirla en un proyecto de Python para que otros puedan usarla. Desarrollé un script que actualiza un archivo `config.yaml` con las configuraciones del proxy Shadowsocks, decodificando las URLs de Shadowsocks desde un archivo `ssconfig`. Además, creé otro script que utiliza `gsutil` para subir el archivo de suscripción para los clientes a Google Cloud Storage.

Utilicé Windsurf, un editor de código con IA, para obtener asistencia. Sin embargo, tuvo dificultades para manejar dependencias simuladas en las pruebas unitarias de Python.

Reflexionando sobre las lecciones de pruebas compartidas por Yin Wang, recordé sus experiencias en Google, donde trabajó en un intérprete de Python e indexó el código de la empresa para la funcionalidad de búsqueda. Sus colegas insistían en escribir pruebas, lo que él encontraba molesto. Creía que escribir código elegante era más importante que las pruebas, y que sus colegas solo entendían aspectos superficiales sin captar la esencia.

Me di cuenta de mi error; la IA no lo señaló. Debería asegurarme de que el código principal de una biblioteca sea sólido antes de centrarme en las pruebas. El mismo principio se aplica a un proyecto de prueba de concepto. En trabajos anteriores, como iniciar un microservicio, las pruebas deberían escribirse después de que el microservicio tenga algunas API o funciones.

Si Windsurf hubiera manejado bien la parte de las pruebas, no tendría esta queja. Sin embargo, hay dos problemas distintos en juego: el momento de implementar las pruebas y la forma correcta de escribirlas. Actualmente, nos estamos enfocando en lo primero. Estos problemas están interrelacionados hasta cierto punto. Si un editor de código con IA o un humano encuentran fácil escribir código de prueba, el momento de las pruebas podría parecer trivial. Sin embargo, el esfuerzo requerido para escribir pruebas es comparable al de escribir el código principal, lo que hace que el momento sea una consideración importante.

Desde una perspectiva de colaboración, el enfoque hacia las pruebas puede variar. En un proyecto personal, podría escribir una cantidad significativa de código antes de crear pruebas. Sin embargo, cuando se trabaja en equipo, generalmente es mejor escribir pruebas para cada fragmento o característica. Pero esto no siempre es así; depende de cómo colabore el equipo. Una forma más precisa de decirlo es que las pruebas deben escribirse para el código que los miembros del equipo comparten entre sí. El objetivo es garantizar la calidad del código, por lo que antes de entregar el código, cada miembro del equipo es libre de elegir el momento en que realiza las pruebas.

En una experiencia laboral anterior, colaboré con otros tres ingenieros de backend en una funcionalidad que tomó medio año en entregarse. Desde una perspectiva de pruebas, los puntos discutidos en este artículo pueden explicar por qué el desarrollo fue lento durante ese tiempo.

Desde el punto de vista de la colaboración, aquellos responsables del código principal también deberían ser responsables de las pruebas relacionadas. Las tareas deben entrelazarse lo menos posible, con responsabilidades claras y separadas para cada miembro del equipo.

Volviendo al tema de las pruebas, los editores de código de IA también carecen de este tipo de optimización, lo que resalta un área de mejora. Este principio no se limita a la ingeniería de software; también es relevante para el hardware y otros campos. Las pruebas son una forma de optimización, y como dice el refrán: “La optimización prematura es la raíz de todos los males”.

Es crucial recordar el objetivo principal del trabajo. Aunque los procesos y procedimientos son inevitables, debemos tener en cuenta lo que realmente es importante.

Referencias:

- Desarrollo Guiado por Pruebas, Yin Wang
- La Lógica de las Pruebas, Yin Wang