iacc instituto profesional

EVALUACIÓN

EVALUACIÓN DE SISTEMAS QA
Semana 2

Michel Brevis
28-10-2024
Técnico en Análisis y Programación Computacional

DESARROLLO:

Caso Práctico: Automatización de Pruebas para una Aplicación de Cálculo de Fuerza

En el contexto de una empresa de desarrollo de software, se ha desarrollado una aplicación web que calcula la fuerza de acuerdo con la fórmula de la Ley de Newton: Fuerza = masa x aceleración. Se le ha asignado al equipo responsable de pruebas la tarea de diseñar pruebas automatizadas para garantizar el correcto funcionamiento de esta aplicación.

Estas pruebas deben demostrar si la funcionalidad de cálculo de fuerza produce resultados precisos y coherentes.

El equipo de pruebas ha recibido el código de la aplicación, implementada en Python con Flask, y se les ha proporcionado un ejemplo de cómo interactuar con la interfaz de usuario utilizando Selenium para realizar pruebas automatizadas. Se espera que el equipo utilice este marco de trabajo para diseñar casos de prueba que evalúen el resultado obtenido frente al resultado esperado, determinando así si la prueba fue exitosa o fallida.

La aplicación web cuenta con una interfaz de usuario simple que permite al usuario ingresar la masa y la aceleración, y luego muestra el resultado del cálculo de fuerza. El equipo de pruebas debe asegurarse de que la aplicación maneje correctamente diferentes combinaciones de valores de masa y aceleración, y que genere el resultado esperado de acuerdo con la fórmula establecida.

El desafío radica en diseñar casos de prueba efectivos que cubran una amplia gama de escenarios posibles y que detecten posibles errores o anomalías en el cálculo de la fuerza. Además, el equipo debe establecer una estrategia para la gestión de datos de prueba, agregando tantas pruebas y datos como sean necesarios para validar la robustez y la precisión de la aplicación.

No olviden consultar la guía de ejercicios, donde encontrarán material práctico fundamental para abordar el ejercicio.

A continuación, se muestran los códigos fuentes entregados al equipo de prueba y algunos aspectos de interés:

1.Código fuente completo de calculadora_fuerza.py

2. Código fuente completo de la platilla HTML (index.html)

3.Código Python – Selenium

A continuación, responde las siguientes preguntas:

1. **¿Cuáles son los componentes principales que debemos considerar al examinar los frameworks de automatización de pruebas, especialmente en el contexto de la aplicación de cálculo de fuerza?**

Considerando la aplicación que se le dará al framework, los componentes principales según lo averiguado serian:

- **Lenguaje de programación**:

Es importante asegurarnos de que el framework de pruebas esté en un lenguaje compatible con el de la aplicación y con el equipo de desarrollo. En este caso, si nuestra aplicación usa Python, frameworks como unittest o pytest son buenas opciones porque se integran bien con el código y facilitan el trabajo en equipo.

- **Compatibilidad con herramientas de automatización de pruebas**:

Como estamos usando Selenium para automatizar pruebas de la interfaz web, necesitamos un framework que sea compatible con Selenium. Esto permitirá que podamos ejecutar pruebas en el navegador simulando las acciones del usuario, como ingresar la masa y la aceleración en los campos, y verificar el resultado.

- **Facilidad para escribir y mantener pruebas**:

Idealmente, queremos un framework que permita escribir pruebas de manera sencilla y que se puedan actualizar fácilmente en caso de cambios en la aplicación. Esto es importante porque si la interfaz o las funciones de la aplicación cambian, también tendremos que actualizar las pruebas.

- **Manejo de datos de prueba**:

El framework debería facilitarnos el uso de diferentes datos de entrada, como varios valores de masa y aceleración, para asegurarnos de que la aplicación funcione correctamente con diferentes combinaciones de números. Esto se logra usando “datos parametrizados” o “archivos de datos” en las pruebas, lo cual es ideal para simular diferentes escenarios.

- **Generación de reportes**:

Un buen framework debe permitirnos ver los resultados de las pruebas de manera clara. Con reportes de prueba, podremos revisar qué pruebas pasaron, cuáles fallaron y, si hubo fallas, entender dónde ocurrieron. Esto es útil para mejorar la aplicación y corregir errores.

Estos componentes nos permiten evaluar si un framework es adecuado para automatizar las pruebas de una aplicación sencilla como esta, que calcula la fuerza aplicando la fórmula de Newton.

2. **¿Cuáles son las características clave de Selenium IDE que la convierten en una herramienta eficaz para la detección de errores del software en el caso práctico de pruebas automatizadas?**

Selenium IDE es una herramienta de automatización de pruebas fácil de usar. Aquí explico algunas de sus características clave y cómo ayudan a detectar errores en el caso de nuestra aplicación de cálculo de fuerza:

- **Grabación y reproducción de pruebas**:

Selenium IDE permite grabar las interacciones que un usuario hace en la interfaz (por ejemplo, ingresar valores de masa y aceleración y hacer clic en el botón de cálculo). Esto facilita la creación de pruebas sin necesidad de escribir código desde cero, ya que las acciones se registran automáticamente. Luego, estas pruebas se pueden reproducir para verificar que la aplicación siga funcionando correctamente con los mismos pasos.

- **Ejecución paso a paso**:

Con Selenium IDE, es posible ejecutar las pruebas paso a paso, lo cual es útil para entender en qué momento exacto falla la aplicación si algo no funciona como se espera. Esto ayuda a identificar problemas específicos en el flujo de trabajo, como errores al calcular la fuerza o al mostrar el resultado.

- **Asistencia visual para depuración**:

Selenium IDE proporciona un sistema visual para revisar las pruebas y verificar si los elementos en la interfaz (como los campos de entrada de masa y aceleración) están correctamente identificados y operando según lo previsto. Esto ayuda a detectar errores en la interfaz, como campos que no responden a los valores ingresados.

- **Facilidad de edición y mantenimiento**:

Las pruebas grabadas pueden modificarse fácilmente si se necesita cambiar algún detalle, como un valor específico o la ubicación de un botón. Esto es especialmente útil en nuestro caso, ya que, si la aplicación de cálculo de fuerza cambia, podemos actualizar las pruebas rápidamente sin necesidad de reescribirlas desde cero.

- **Ejecución de pruebas en diferentes navegadores**:

Selenium IDE permite correr las pruebas en diferentes navegadores (Chrome, Firefox, etc.), lo que es importante para asegurarnos de que la aplicación funcione bien en distintas plataformas. En nuestro caso, esto garantiza que el cálculo de fuerza funcione igual, sin importar el navegador que esté usando el usuario.

- **Exportación de scripts de prueba**:

Si en algún momento queremos llevar las pruebas grabadas a un código más avanzado, Selenium IDE permite exportarlas en diferentes lenguajes (como Python o JavaScript). Esto da flexibilidad para integrar las pruebas en un entorno de desarrollo más completo y con otras herramientas de pruebas automatizadas.

3. ¿Cómo podemos aplicar los principios de desarrollo de scripts y pruebas automatizadas para mejorar la calidad del sistema de cálculo de fuerza, teniendo en cuenta los requisitos específicos y los escenarios de prueba identificados en nuestro caso práctico?

Para mejorar la calidad del sistema de cálculo de fuerza mediante el desarrollo de scripts y pruebas automatizadas, podemos aplicar algunos principios básicos que ayudarán a validar correctamente los escenarios de prueba identificados en nuestro caso práctico.

1. Validación de Entradas

Asegurar que los valores ingresados sean válidos antes de realizar el cálculo es fundamental.

En este caso, se deben diseñar pruebas automatizadas que verifiquen el comportamiento del sistema al recibir entradas válidas e inválidas. Por ejemplo:

- Pruebas con valores válidos de masa y aceleración (números positivos y decimales).
- Pruebas con entradas inválidas, como valores negativos, caracteres alfabéticos o campos vacíos, para verificar que el sistema maneje errores adecuadamente y no intente realizar el cálculo.

2. Pruebas Funcionales con Valores Límite

Evaluar cómo el sistema responde en los límites de sus condiciones operativas.

Podemos usar valores extremos en nuestras pruebas para verificar que el sistema responda de manera adecuada. Algunos ejemplos incluyen:

- Masa o aceleración en cero para confirmar que el resultado sea cero.
- Valores extremadamente altos para ver si el cálculo produce resultados esperados o si el sistema se bloquea o muestra un error.

3. Automatización de Escenarios de Prueba con Selenium

Utilizar pruebas automatizadas para validar diferentes flujos de trabajo en la interfaz de usuario.

Con Selenium, se pueden escribir scripts de prueba para automatizar la entrada de datos en los campos de "masa" y "aceleración", hacer clic en "Calcular" y verificar que el resultado sea correcto en cada caso. Esto permite simular varios escenarios de usuario y comprobar que la aplicación funcione correctamente de principio a fin.

4. Pruebas de Rendimiento

Evaluar el rendimiento de la aplicación bajo carga y con múltiples solicitudes.

pueden escribir scripts que envíen múltiples solicitudes simultáneas al endpoint `/calcular` con diferentes valores de entrada, observando si el sistema responde rápidamente y sin errores. Estas pruebas ayudan a asegurar que la aplicación puede manejar múltiples usuarios sin problemas de rendimiento.

5. Pruebas de Regresión

Asegurarse de que las nuevas versiones del sistema no introduzcan errores en funcionalidades que ya funcionaban bien.

Al crear una suite de pruebas automatizadas, cada vez que se realicen cambios en la calculadora de fuerza, estas pruebas podrán ejecutarse nuevamente para confirmar que los cambios no afectaron el funcionamiento correcto de las funciones anteriores.

6. Generación de Reportes

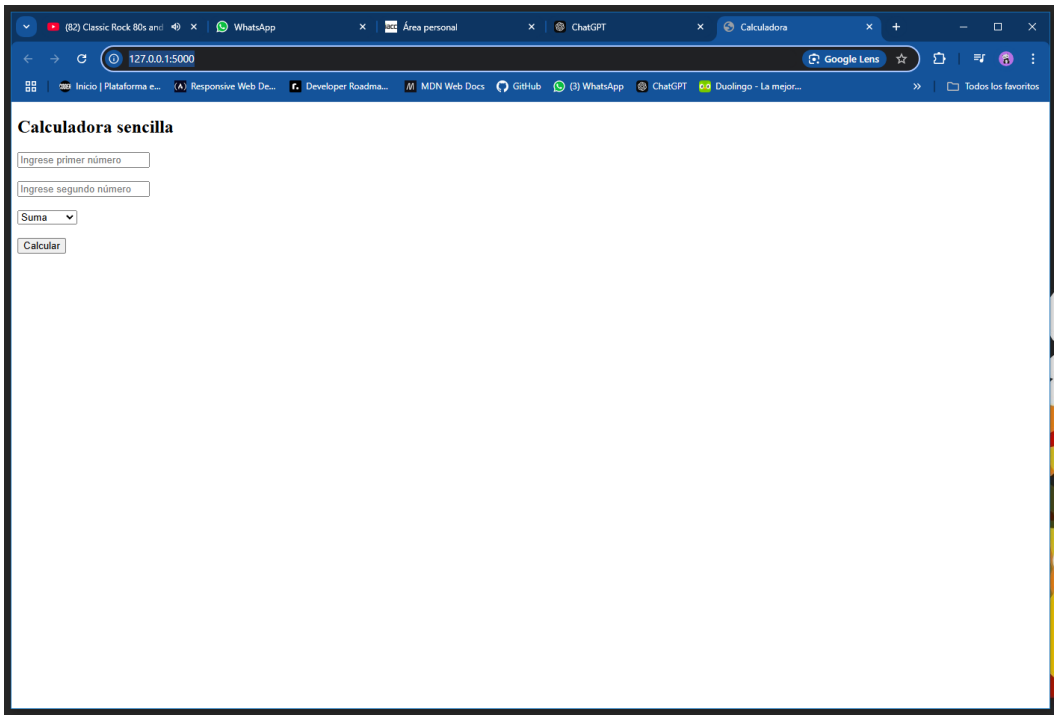
Proporcionar una retroalimentación clara y detallada sobre el estado de las pruebas.

Al usar frameworks de prueba como `pytest` junto con Selenium, se pueden generar reportes de prueba detallados que muestren el resultado de cada prueba, incluidas aquellas que fallaron y por qué. Estos reportes facilitan la detección y corrección de errores.

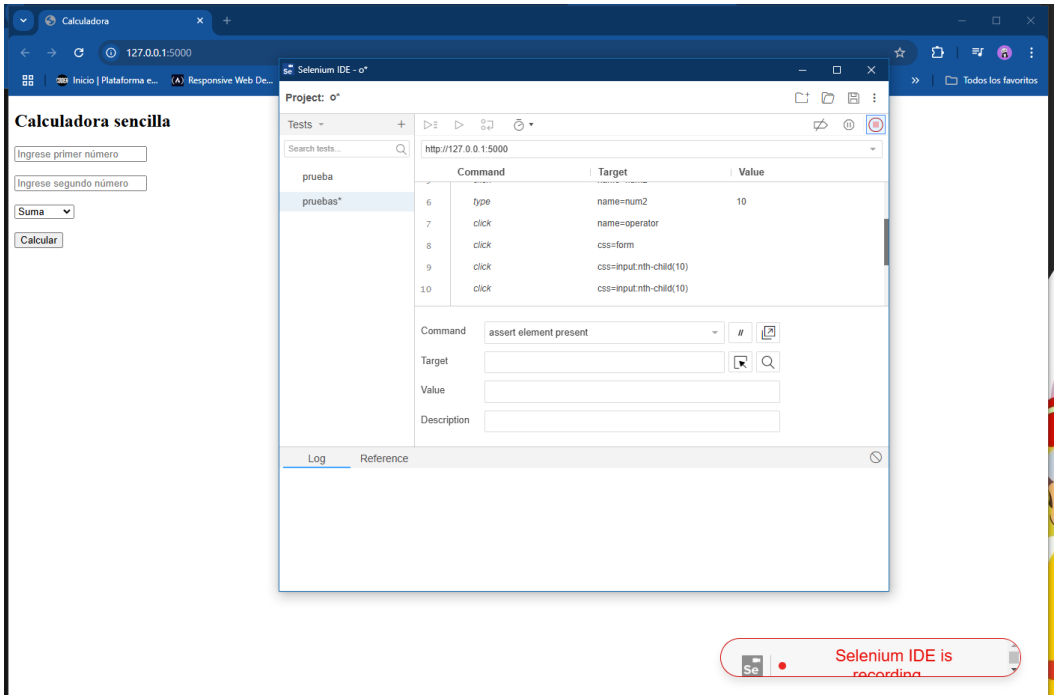
Finalmente, incluye en el documento una o varias capturas de pantalla que muestren la salida después de ejecutar el programa con los datos de prueba utilizados.

No pude correr el código proporcionado directamente desde visual code, pero pude hacer pruebas de la aplicación directamente desde la extensión de Selenium de Google Crome

Aplicación de calculadora en navegador.



Pruebas con Selenium desde navegador



REFERENCIAS BIBLIOGRÁFICAS:

- **Ejemplo texto de lectura de IACC:**

IACC. (2024). *Evaluación de sistemas QA* Semana 2