### **Guía de Pruebas de Automatización con Selenium para una Aplicación React**

#### **Requisitos Previos**

1. **Node.js** (que incluye npm) instalado.
2. **Aplicación web de React** ya en funcionamiento. Asegúrate de que tu aplicación React está corriendo localmente, en nuestro caso en http://localhost:5173.

### **Paso 1: Crear y Configurar el Proyecto de Pruebas**

1. **Instalar Selenium WebDriver**: Abre una terminal en la raíz de tu proyecto y ejecuta:  
    npm install selenium-webdriver

**Instalar el Driver del Navegador**: Dependiendo del navegador que desees usar, instala su controlador:

* **ChromeDriver** (para pruebas en Google Chrome):

npm install chromedriver

**Configurar Selenium**: Asegúrate de que chromedriver esté en el PATH del sistema (generalmente npm lo configura automáticamente). Esto permitirá que Selenium pueda controlarlo sin problemas.

**Configurar la estructura de carpetas**: Crea una carpeta tests en la raíz del proyecto para organizar los archivos de pruebas. Dentro de tests, crea un archivo pokemonViewer.test.js para escribir las pruebas.

### **Paso 2: Escribir el Script de Pruebas en Selenium**

En el archivo tests/pokemonViewer.test.js, copia y pega el siguiente código. Este script:

* Abre la aplicación web.
* Verifica si la lista de Pokémon se carga correctamente.
* Hace clic en un Pokémon para ver los detalles.
* Valida que los detalles del Pokémon se muestren correctamente.

// tests/pokemonViewer.test.js

const { Builder, By, until } = require('selenium-webdriver');

const chrome = require('selenium-webdriver/chrome');

(async function pokemonViewerTest() {

// Paso 1: Iniciar el navegador con Selenium y ChromeDriver

let driver = await new Builder().forBrowser('chrome').setChromeOptions(new chrome.Options()).build();

try {

// Paso 2: Navegar al sitio web de la aplicación React

await driver.get('http://localhost:3000');

// Paso 3: Verificar que la lista de Pokémon está cargada

const pokemonList = await driver.wait(until.elementLocated(By.css('ul')), 10000); // Espera hasta que aparezca la lista

const pokemonItems = await pokemonList.findElements(By.css('li'));

if (pokemonItems.length > 0) {

console.log(`Lista de Pokémon cargada con ${pokemonItems.length} elementos.`);

} else {

console.error('La lista de Pokémon no se cargó.');

return;

}

// Paso 4: Hacer clic en el primer Pokémon y verificar los detalles

await pokemonItems[0].click();

const pokemonName = await driver.wait(until.elementLocated(By.css('h2')), 5000);

console.log('Primer Pokémon seleccionado:', await pokemonName.getText());

// Verificar que los detalles del Pokémon se han cargado

const pokemonImage = await driver.findElement(By.css('img'));

if (pokemonImage) {

console.log('Detalles del Pokémon cargados correctamente.');

} else {

console.error('No se cargaron los detalles del Pokémon.');

}

} finally {

// Paso 5: Cerrar el navegador después de completar el test

await driver.quit();

}

})();

### **Explicación del Código**

1. **Inicio del Navegador**: El script crea una instancia de Chrome controlada por Selenium.
2. **Acceso a la Página**: Abre la URL http://localhost:3000 donde corre la aplicación React.
3. **Verificación de la Lista de Pokémon**: Espera hasta que aparezca una lista (<ul>) y verifica que contenga elementos (<li>).
4. **Selección de un Pokémon**: Hace clic en el primer Pokémon de la lista y valida que los detalles (nombre, imagen) se muestran correctamente.
5. **Cierre del Navegador**: Una vez completado el test, el navegador se cierra automáticamente.

### **Paso 3: Ejecutar el Test**

Asegúrate de que tu aplicación React está corriendo (puedes hacerlo con npm start en otra terminal). Luego, ejecuta el test con:

node tests/pokemonViewer.test.js

### **Análisis de Resultados**

Al ejecutar el script de prueba, deberías ver los siguientes resultados en la terminal:

* Si la lista de Pokémon se carga correctamente, verás un mensaje que indica la cantidad de Pokémon en la lista.
* Si los detalles del Pokémon seleccionado se muestran correctamente (nombre e imagen), verás mensajes de éxito en la terminal.

**Ejemplo de salida esperada**:

Lista de Pokémon cargada con 50 elementos.

Primer Pokémon seleccionado: bulbasaur

Detalles del Pokémon cargados correctamente.

**Posibles errores**:

* **"La lista de Pokémon no se cargó"**: Esto indica que el script no pudo encontrar elementos <li> en la lista de Pokémon. Podría ser un problema de conexión con la API o que el componente no se haya renderizado correctamente.
* **"No se cargaron los detalles del Pokémon"**: Esto significa que no se encontraron los detalles del Pokémon después de hacer clic. Asegúrate de que el componente de detalles esté correctamente configurado.

### **Opcional: Mejoras en el Test**

1. **Iterar sobre múltiples Pokémon**: Extiende el test para hacer clic en diferentes Pokémon y verificar que cada uno muestra sus propios detalles correctamente.
2. **Capturas de pantalla en caso de fallo**: Configura Selenium para tomar capturas de pantalla automáticamente si alguna prueba falla.
3. **Integración CI/CD**: Automatiza el test en un flujo de CI/CD, como GitHub Actions o Jenkins, para ejecutarlo cada vez que se hagan cambios en el proyecto.

### **Conclusión**

Este procedimiento te permite hacer pruebas de automatización básicas en tu aplicación React con Selenium, validando la carga de contenido y la navegación. Este flujo de trabajo es escalable y puede extenderse para cubrir más funcionalidades de tu aplicación a medida que la desarrolles.