Sesión 1 – Realizar Actividades de Mejora de la Calidad

Aprendiz

Jorge Alejandro Gómez Quecan

Servicio Nacional de Aprendizaje

SENA

Ficha

2848938

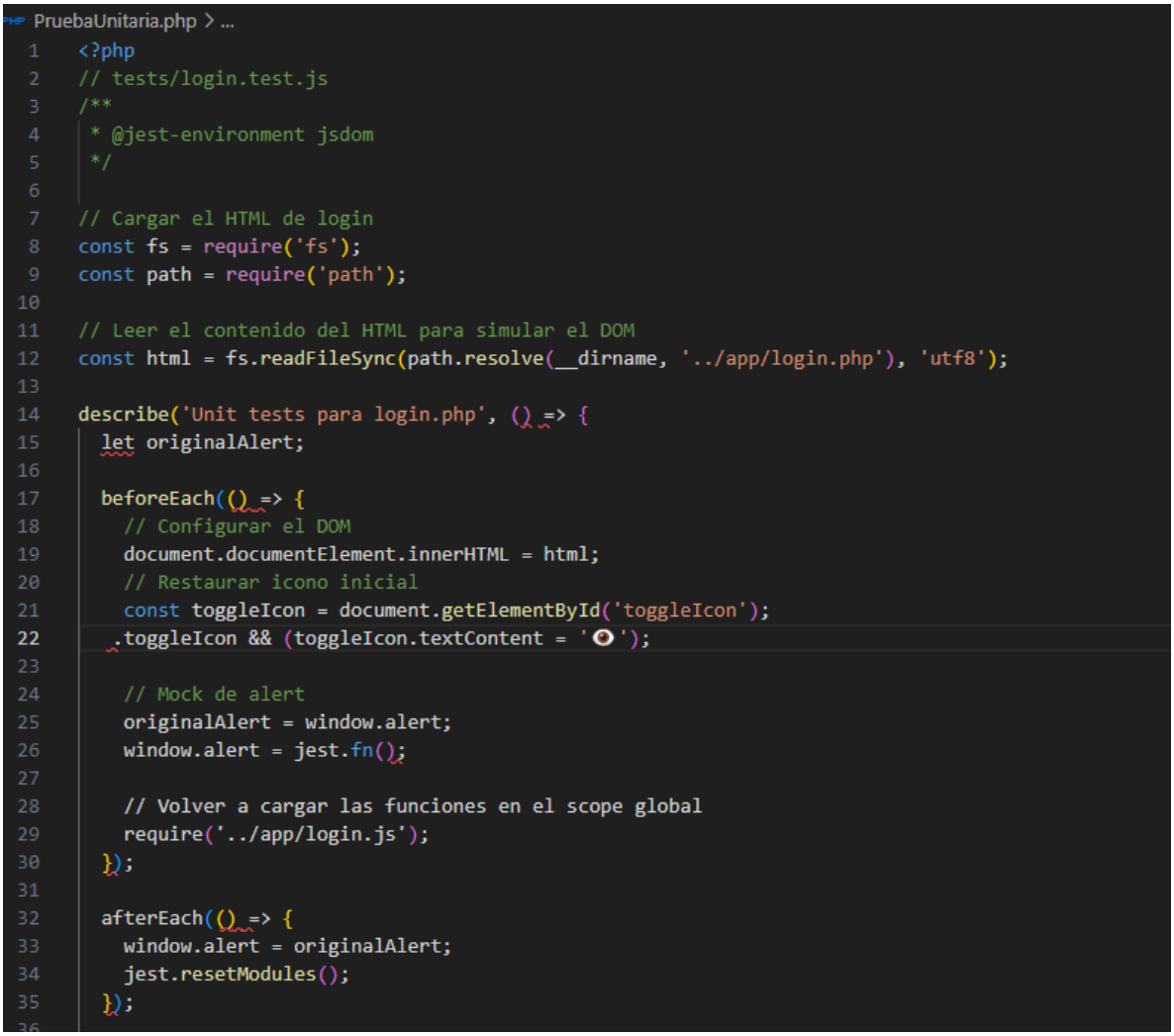
Instructor

William Mayorca

19 de Mayo del 2025

Prueba Unitaria:

Para las pruebas unitarias se ha creado login.test.js, que utiliza el entorno jsdom para simular el DOM de la página. Antes de cada prueba, se lee todo el HTML de login.php y se inyecta en document.documentElement.innerHTML, restaurando además el estado inicial del icono de mostrar/ocultar contraseña. Se mockea window.alert para verificar llamadas sin interrumpir la ejecución, y se recarga el módulo login.js que contiene las funciones validarFormulario, togglePassword y el código de focus/blur.



El primer test verifica “validarFormulario” con una contraseña de solo 5 caracteres: al invocar la función se espera que “window.alert” sea llamada con el mensaje correcto y que el retorno sea false. El segundo test cubre el caso válido, asegurando que no se dispara ninguna alerta y que la función retorna true. Así se comprueba la lógica de validación de longitud mínima de la contraseña del lado del cliente.

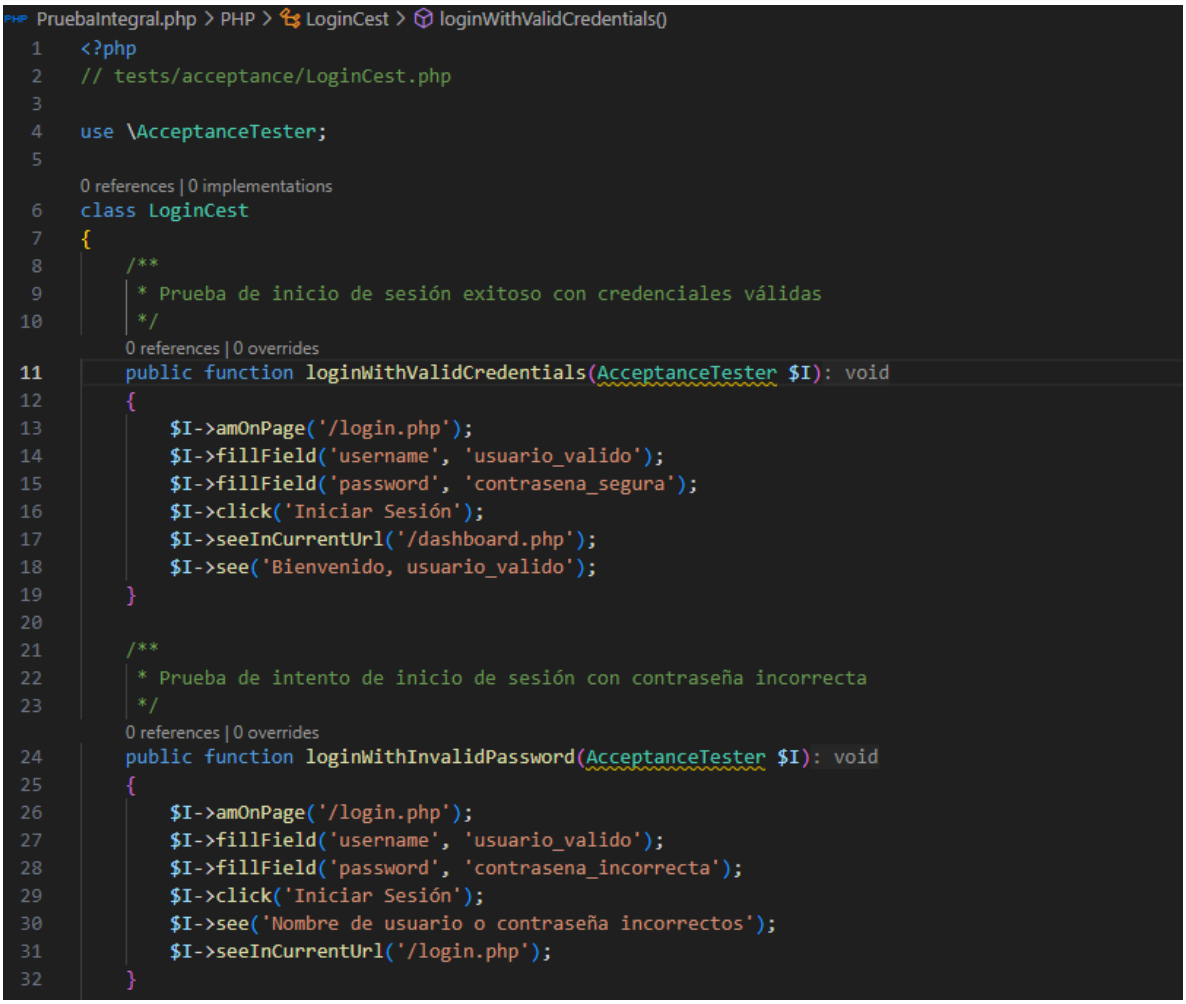
```
37  test('validarFormulario: contraseña corta desencadena alerta y retorna false', () => {
38    const passwordInput = document.getElementById('password');
39    passwordInput.value = '12345'; // 5 caracteres
40    const result = validarFormulario();
41    expect(window.alert).toHaveBeenCalledWith('La contraseña debe tener al menos 6 caracter
42    expect(result).toBe(false);
43  });
44
45  test('validarFormulario: contraseña válida retorna true sin alerta', () => {
46    const passwordInput = document.getElementById('password');
47    passwordInput.value = 'abcdef'; // 6 caracteres
48    const result = validarFormulario();
49    expect(window.alert).not.toHaveBeenCalled();
50    expect(result).toBe(true);
51  });
52
53  test('togglePassword: alterna tipo de input y contenido de icono', () => {
54    const passwordField = document.getElementById('password');
55    const icon = document.getElementById('toggleIcon');
56
57    // Estado inicial
58    expect(passwordField.type).toBe('password');
59    expect(icon.textContent).toBe('👁');
60
61    togglePassword();
62    expect(passwordField.type).toBe('text');
63    expect(icon.textContent).toBe('🙈');
64
65    togglePassword();
66    expect(passwordField.type).toBe('password');
67    expect(icon.textContent).toBe('👁');
68  });
69
```

El tercer test, sobre togglePassword, confirma que al alternar el campo de contraseña cambia su atributo type entre 'password' y 'text', y que el icono asociado también cambia. Esto garantiza la interactividad de la vista para mostrar u ocultar la contraseña. Los últimos dos tests se centran en los eventos focus y blur: despachan manualmente cada evento sobre un input y comprueban que la propiedad “style.borderColor” cambia a #007bff al enfocar y vuelve a #ccc al perder el foco, validando así la mejora visual de la experiencia de usuario.

```
69
70    test('evento focus cambia color del borde al enfocar input', () => {
71      const input = document.querySelector('input[name=username]');
72      const event = new Event('focus');
73      input.dispatchEvent(event);
74      expect(input.style.borderColor).toBe('#007bff');
75    });
76
77    test('evento blur restablece color del borde al perder foco', () => {
78      const input = document.querySelector('input[name=password]');
79      const event = new Event('blur');
80      input.dispatchEvent(event);
81      expect(input.style.borderColor).toBe('#ccc');
82    });
83  });
84
```

Prueba Integral:

En el archivo LoginCest.php se define una clase LoginCest que agrupa cinco escenarios de aceptación. Cada método comienza por navegar a la página de login ($I->amOnPage('/login.php')), luego interactúa con los campos de usuario y contraseña usando $I->fillField(), y finalmente comprueba tanto la navegación como los mensajes que aparecen en pantalla. Por ejemplo, en loginWithValidCredentials tras enviar un formulario válido se espera llegar a /dashboard.php y ver un saludo personalizado. Este patrón “Arrange–Act–Assert” garantiza que el flujo completo de inicio de sesión funciona correctamente desde la perspectiva del usuario.



El segundo método, loginWithInvalidPassword, sirve para verificar que cuando el usuario introduce una contraseña equivocada, el sistema no permite el acceso y muestra el mensaje “Nombre de usuario o contraseña incorrectos”, sin cambiar de URL. Esto valida la lógica de autenticación en el back-end y la presentación del error. La tercera prueba, loginWithMissingFields, comprueba la validación del front-end: al dejar los campos vacíos, debería mostrarse un mensaje de “campo obligatorio” junto a cada input, lo cual hace uso de los atributos required en el HTML y de posibles validaciones JavaScript o HTML5.

```
34      /**
35       * Prueba de validación de campos obligatorios cuando faltan campos
36       */
        0 references | 0 overrides
37      public function loginWithMissingFields(AcceptanceTester $I): void
38      {
39          $I->amOnPage('/login.php');
40          // Dejar ambos campos vacíos
41          $I->click('Iniciar Sesión');
42          $I->see("Este campo es obligatorio", 'input[name=username]');
43          $I->see("Este campo es obligatorio", 'input[name=password]');
44      }
45
46      /**
47       * Prueba de prevención de inyección SQL
48       */
        0 references | 0 overrides
49      public function sqlInjectionPrevention(AcceptanceTester $I): void
50      {
51          $I->amOnPage('/login.php');
52          $I->fillField('username', "' OR 1=1 -- ");
53          $I->fillField('password', 'cualquier');
54          $I->click('Iniciar Sesión');
55          $I->see('Nombre de usuario o contraseña incorrectos');
56      }
57
58      /**
59       * Prueba de enlaces de navegación para registro y recuperación de contraseña
60       */
        0 references | 0 overrides
61      public function navigationLinks(AcceptanceTester $I): void
62      {
63          $I->amOnPage('/login.php');
64          $I->seeLink('Registrarse', 'registro.php');
65          $I->click('Registrarse');
66          $I->seeInCurrentUrl('/registro.php');
67          $I->amOnPage('/login.php');
68          $I->seeLink('¿Olvidaste tu contraseña?', 'recuperar.php');
69          $I->click('¿Olvidaste tu contraseña?');
70          $I->seeInCurrentUrl('/recuperar.php');
71      }
72  }
73
```

En sqlInjectionPrevention se simula un intento de inyección SQL introduciendo un payload malicioso en el campo de usuario (' OR 1=1 --). La prueba asegura que, pese a este intento, el sistema responde con el mismo mensaje de credenciales incorrectas, demostrando que las consultas están correctamente parametrizadas o escapadas para impedir inyección de código. Finalmente, navigationLinks recorre los enlaces “Registrarse” y “¿Olvidaste tu contraseña?”, haciendo clic y comprobando que cada uno redirige al archivo correspondiente (registro.php y recuperar.php), lo que garantiza la navegabilidad del formulario de login.