

## Informe de Pruebas Unitarias – Función login

### 1. Introducción

El presente informe documenta las pruebas unitarias realizadas sobre la función **login** perteneciente al controlador `usuarioController.js` del sistema **Tenis Master**.

El objetivo es validar el comportamiento del proceso de autenticación de usuarios, verificando tanto los casos correctos como los escenarios de error.

Las pruebas se ejecutaron utilizando **Jest**, con módulos de base de datos y `bcrypt` **mockeados**, permitiendo validar únicamente la lógica interna del controlador.

### 2. Función evaluada: login

La función login realiza:

- Validación de usuario y contraseña.
- Consulta a MongoDB mediante `Usuario.findOne()`.
- Comparación de contraseñas usando `bcrypt.compare()`.
- Creación de sesión (`req.session.usuario`).

Las pruebas buscan verificar que esta lógica responda adecuadamente en situaciones reales.

### 3. Casos de prueba realizados

#### 3.1. Caso 1 – Falta username o password

**Objetivo:** Validar que la función rechace solicitudes incompletas.

**Entrada simulada:**

```
{ "username": "", "password": "" }
```

**Resultado esperado:**

- Código 400
- Render de la vista login con el mensaje *"Completa usuario y contraseña."*

**Resultado obtenido:** ✓ Correcto.

#### 3.2. Caso 2 – Usuario inexistente

**Entrada simulada:**

```
{ "username": "test", "password": "123" }
```

Mock: `Usuario.findOne = () => null`

**Resultado esperado:**

- Código 401

- Mensaje: *"Usuario y/o contraseña inválidos."*

**Resultado obtenido:** ✓ Correcto.

### 3.3. Caso 3 – Contraseña incorrecta

Mock:

- Usuario.findOne devuelve un usuario válido.
- bcrypt.compare devuelve false.

**Resultado esperado:**

- Código 401
- Mensaje: *"Usuario y/o contraseña inválidos."*

**Resultado obtenido:** ✓ Correcto.

### 3.4. Caso 4 – Login exitoso

Mock:

- Usuario existente.
- bcrypt.compare = true.

**Resultado esperado:**

- Sesión creada:

```
req.session.usuario = {
  id, username, rol, nombre, apellido
}
```

- Redirección a /panel.

**Resultado obtenido:** ✓ Correcto.

```
> tenis_master@1.0.0 test
> node --experimental-vm-modules node_modules/jest/bin/jest.js

(node:1700) ExperimentalWarning: VM Modules is an experimental feature and might change at any time
(Use `node --trace-warnings ...` to show where the warning was created)
PASS Test/autorizar.test.js
PASS Test/login.test.js

Test Suites: 2 passed, 2 total
Tests:      8 passed, 8 total
Snapshots:  0 total
Time:       1.13 s
Ran all test suites.
```

#### 4. Código de la prueba unitaria utilizada

```
// Test/login.test.js

import { login } from '../controllers/usuarioController.js';
import Usuario from '../models/Usuario.js';
import bcrypt from 'bcrypt';

jest.mock('../models/Usuario.js');
jest.mock('bcrypt');

describe('Pruebas de login', () => {
  let req, res;

  beforeEach(() => {
    req = {
      body: {},
      session: {},
    };

    res = {
      status: jest.fn().mockReturnThis(),
      render: jest.fn(),
      redirect: jest.fn(),
    };
  });

  test('Debe rechazar cuando faltan campos', async () => {
    req.body = { username: '', password: '' };

    await login(req, res);

    expect(res.status).toHaveBeenCalledWith(400);
  });
});
```

```
    expect(res.render).toHaveBeenCalled('login', { mensaje: 'Completá usuario y contraseña.' });  
  });
```

```
test('Debe rechazar cuando el usuario no existe', async () => {
```

```
  req.body = { username: 'test', password: '123' };
```

```
  Usuario.findOne.mockResolvedValue(null);
```

```
  await login(req, res);
```

```
  expect(res.status).toHaveBeenCalled(401);
```

```
  expect(res.render).toHaveBeenCalled('login', { mensaje: 'Usuario y/o contraseña inválidos.' });  
});
```

```
test('Debe rechazar cuando la contraseña es incorrecta', async () => {
```

```
  req.body = { username: 'test', password: '123' };
```

```
  Usuario.findOne.mockResolvedValue({ username: 'test', password: 'hashed' });
```

```
  bcrypt.compare.mockResolvedValue(false);
```

```
  await login(req, res);
```

```
  expect(res.status).toHaveBeenCalled(401);
```

```
  expect(res.render).toHaveBeenCalled('login', { mensaje: 'Usuario y/o contraseña inválidos.' });  
});
```

```
test('Debe permitir login correcto', async () => {
```

```
  req.body = { username: 'test', password: '123' };
```

```
  const user = {
```

```
    _id: '1', username: 'test', password: 'hashed', rol: 'cliente', nombre: 'A', apellido: 'B'
```

```
  };
```

```
Usuario.findOne.mockResolvedValue(user);
bcrypt.compare.mockResolvedValue(true);

await login(req, res);

expect(req.session.usuario).toEqual({
  id: '1',
  username: 'test',
  rol: 'cliente',
  nombre: 'A',
  apellido: 'B'
});

expect(res.redirect).toHaveBeenCalledWith('/panel');
});
});
```

## 5. Conclusión

La función login pasó exitosamente todas las pruebas unitarias realizadas. Esto confirma que:

- La validación de credenciales funciona correctamente.
- Las respuestas ante errores cumplen lo esperado.
- La sesión se genera correctamente en caso de autenticación válida.
- No se detectaron vulnerabilidades lógicas dentro del proceso.

El módulo está listo para integración con el resto del sistema.