

PRUEBAS UNITARIAS			
N° prueba	Descripción	Código	Test
PU1	Validación de credenciales de usuario	<pre>function validateUserCredentials(\$email, \$password, \$dbConnection) { \$stmt = \$dbConnection->prepare("SELECT * FROM users WHERE email = ?"); \$stmt->execute([\$email]); \$user = \$stmt->fetch(); if (\$user && password_verify(\$password, \$user['password'])) { return true; // credenciales correctas } return false; // credenciales incorrectas }</pre>	<pre><?php use PHPUnit\Framework\TestCase; class LoginTest extends TestCase { public function testValidateUserCredentials() { \$mockDb = \$this->createMock(PDO::class); \$mockStmt = \$this->createMock(PDOStatement::class); \$mockDb->method('prepare')->willReturn(\$mockStmt); \$mockStmt->method('execute')->willReturn(true); \$mockStmt->method('fetch')->willReturn(['email' => 'test@example.com', 'password' => password_hash('password123', PASSWORD_DEFAULT),]); \$result = validateUserCredentials('test@example.com', 'password123', \$mockDb); \$this->assertTrue(\$result); \$result = validateUserCredentials('test@example.com', 'wrongpassword', \$mockDb); \$this->assertFalse(\$result); } }</pre>
PU2	Cálculo de totales en el carrito	<pre>function calculateOrderTotal(\$cartItems, \$taxRate = 0.15) { \$subtotal = 0; foreach (\$cartItems as \$item) { \$subtotal += \$item['price'] * \$item['quantity']; } \$tax = \$subtotal * \$taxRate; \$total = \$subtotal + \$tax; return ['subtotal' => round(\$subtotal, 2), 'tax' => round(\$tax, 2), 'total' => round(\$total, 2),]; }</pre>	<pre><?php use PHPUnit\Framework\TestCase; require_once 'path/to/your/functions.php'; class CheckoutTest extends TestCase { public function testCalculateOrderTotal() { \$cartItems = [['price' => 100.00, 'quantity' => 2], ['price' => 50.00, 'quantity' => 1],]; \$result = calculateOrderTotal(\$cartItems); \$this->assertEquals(250.00, \$result['subtotal']); \$this->assertEquals(37.50, \$result['tax']); \$this->assertEquals(287.50, \$result['total']); } }</pre>
PU3	Simulación de colocación de pedidos	<pre>function placeOrder(\$clientId, \$orderDetails, \$dbConnection) { \$stmt = \$dbConnection->prepare("INSERT INTO orders (client_id, total_amount, delivery_address) VALUES (?, ?, ?)"); return \$stmt->execute([\$clientId, \$orderDetails['total'], \$orderDetails['delivery_address']]); }</pre>	<pre><?php use PHPUnit\Framework\TestCase; class PlaceOrderTest extends TestCase { public function testPlaceOrder() { \$mockDb = \$this->createMock(PDO::class); \$mockStmt = \$this->createMock(PDOStatement::class); \$mockDb->method('prepare')->willReturn(\$mockStmt); \$mockStmt->method('execute')->willReturn(true); \$orderDetails = ['total' => 287.50, 'delivery_address' => '123 Calle Principal, Ciudad, Pais',]; }</pre>
PU4	Envío email de confirmación	<pre>function getEmailProvider(\$email) { \$email = strtolower(\$email); if (strpos(\$email, '@gmail.com') !== false) { return 'gmail'; } elseif (strpos(\$email, '@outlook.com') !== false strpos(\$email, '@hotmail.com') !== false) { return 'outlook'; } return 'default'; }</pre>	<pre><?php use PHPUnit\Framework\TestCase; class EmailProviderTest extends TestCase { public function testGetEmailProvider() { \$email = "user@gmail.com"; \$outlook = "user@outlook.com"; \$default = "user@customdomain.com"; \$this->assertEquals('gmail', getEmailProvider(\$email)); \$this->assertEquals('outlook', getEmailProvider(\$outlook)); \$this->assertEquals('default', getEmailProvider(\$default)); } }</pre>
PU5	Validación del manejo de errores y redirección en el inicio de sesión	<pre>describe("Login Form", () => { test("Handles successful login", () => { const mockAjax = jest.fn().mockImplementation((options) => { options.success({ status: "success" }); }); // Simular el redireccionamiento global.location = { reload: jest.fn() }; // Usar la función simulada \$.ajax = mockAjax; const loginForm = \$("#login-form"); loginForm.trigger("submit"); expect(mockAjax).toHaveBeenCalled(); expect(global.location.reload).toHaveBeenCalled(); }); test("Handles incorrect credentials", () => { const mockAjax = jest.fn().mockImplementation((options) => { options.success({ status: "incorrect" }); }); \$.ajax = mockAjax; const loginForm = \$("#login-form"); loginForm.trigger("submit"); // Verifica que muestra el mensaje de error expect(\$(".err-msg").length).toBe(1); expect(\$(".err-msg").text()).toBe("Incorrect Credentials."); }); });</pre>	