

Tecnología en análisis y desarrollo de software

Fase de análisis

Metodologías de desarrollo de software

Actividad.

REALIZAR PRUEBAS DE PLAN DE SOFTWARE.

GA9-220501096-AA1-EV02

Aprendices:

Jorge Eliecer Vargas lopez

Fabrizzio Martínez

Instructor:

CARLOS ALBERTO FUEL

Servicio nacional de aprendizaje Sena

Mayo del 2024

INTROUDCCION.

Las pruebas unitarias son esenciales en el desarrollo de software. Se trata de verificar el comportamiento de las unidades más pequeñas de una aplicación. Técnicamente, estas unidades pueden ser clases, métodos o incluso funciones en diferentes lenguajes de programación. Funcionalmente, funcionalmente también puede ser un conjunto de clases estrechamente relacionadas.

CONTENIDO.

- 1- Portada.**
- 2- Introducción.**
- 3- Contenido.**
- 4- Informe de pruebas unitarias y funcionalidades.**
- 5- Test de funcionalidades.**
- 6- Test de funcionalidades**
- 7- Prueba de integración, identificación de nueva funcionalidades y estrategias y criterios para las pruebas.**
- 8- Entornos de trabajo y metodologías.**
- 9- Cronograma de pruebas.**
- 10- Planificación de pruebas.**
- 11- Diseño de artefactos, selección de herramientas, riesgos y contingencias.**
- 12- Conclusiones.**
- 13- Referencias.**

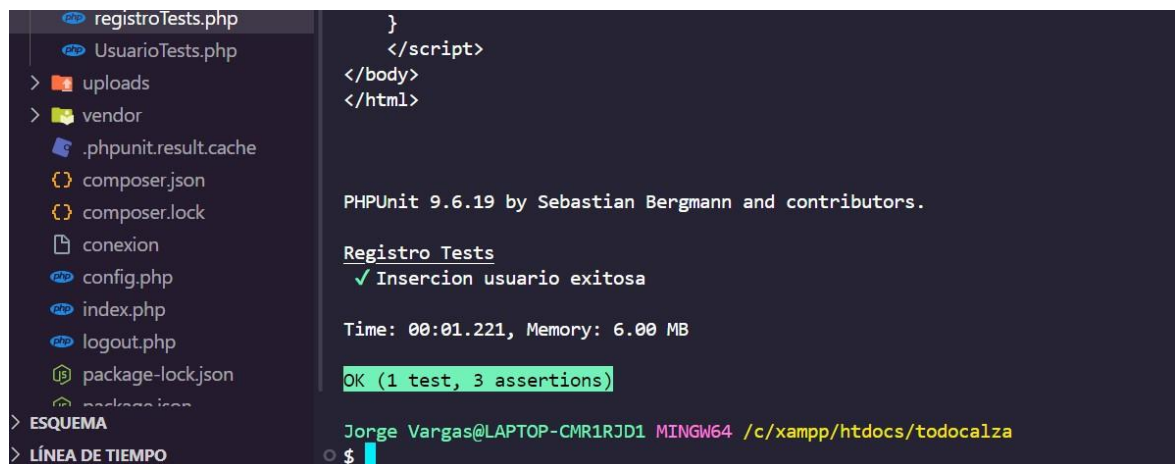
INFORME DE PRUEBAS UNITARIAS PARA APLICACIÓN SPRING BOOT.

1- Análisis de requerimientos del software

El sistema de gestión de clientes permite realizar operaciones CRUD (crear, leer, buscar, actualizar, eliminar, comprar, enviar mensajes) sobre entidades de clientes.

2- Identificación de funcionalidades existentes.

- **Crear cliente:** en este caso de prueba verifica si la inserción de clientes funciona correctamente. Se insertaran utilizando mocaks para garantizar la prueba de casos válidos.



```
    }  
    </script>  
</body>  
</html>  
  
PHPUnit 9.6.19 by Sebastian Bergmann and contributors.  
  
Registro Tests  
✓ Insercion usuario exitosa  
  
Time: 00:01.221, Memory: 6.00 MB  
  
OK (1 test, 3 assertions)  
  
Jorge Vargas@LAPTOP-CMR1RJD1 MINGW64 /c/xampp/htdocs/todocalza  
$
```

- **Consultar perfil:** este caso se enfoca en verificar si la búsqueda del usuario funciona correctamente. Se busca usuarios con diferentes criterios como el rol para garantizar consultas.

```

Jorge Vargas@LAPTOP-CMR1RJD1 MINGW64 /c/xampp/htdocs/todocalza
$ vendor/bin/phpunit --testdox tests/UsuarioTests.php

// vendor/bin/phpunit --testdox tests/UsuarioTests.php

// php vendor/bin/phpunit --testdox tests/ProductTest.php
// ./vendor/bin/phpunit --configuration phpunit.xml

PHPUnit 9.6.19 by Sebastian Bergmann and contributors.

Usuario Tests
✓ Iniciar sesion admin
✓ Iniciar sesion usuario

Time: 00:00.033, Memory: 6.00 MB

OK (2 tests, 2 assertions)

Jorge Vargas@LAPTOP-CMR1RJD1 MINGW64 /c/xampp/htdocs/todocalza
$

```

- **Procesamiento del módulo de compras:** este caso de prueba verifica si la inserción de datos después de la compra funciona correctamente, se realizaron pruebas en la aplicación directamente y con phpunit, para garantizar el servicio y seguridad.

```

Jorge Vargas@LAPTOP-CMR1RJD1 MINGW64 /c/xampp/htdocs/todocalza
$ vendor/bin/phpunit --testdox tests/ModuloTests.php

PHPUnit 9.6.19 by Sebastian Bergmann and contributors.

Modulo
✓ Conexion base datos
✓ Procesamiento formulario compra

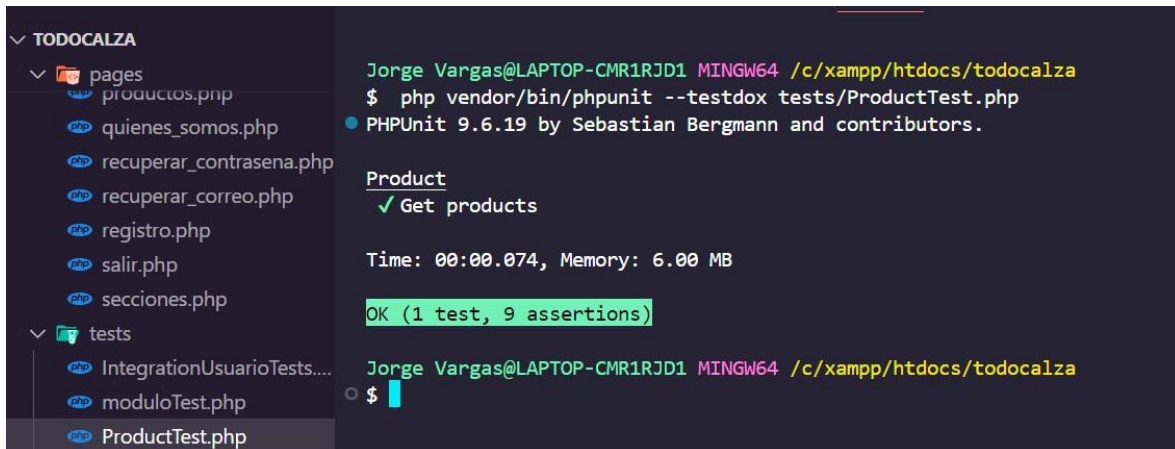
Time: 00:00.271, Memory: 6.00 MB

OK (2 tests, 3 assertions)

Jorge Vargas@LAPTOP-CMR1RJD1 MINGW64 /c/xampp/htdocs/todocalza
$

```

- **Creación de productos:** este caso de prueba se enfoca en verificar si la creación de productos funciona como se espera y realiza la inserción de los productos en el módulo y el catalogo.



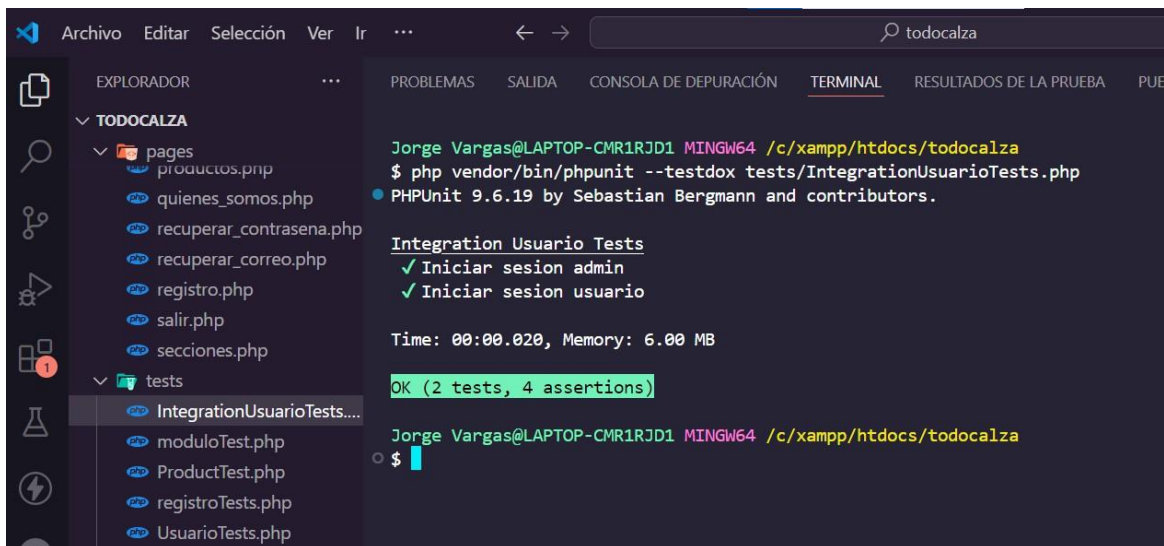
```
Jorge Vargas@LAPTOP-CMR1RJD1 MINGW64 /c/xampp/htdocs/todocalza
$ php vendor/bin/phpunit --testdox tests/ProductTest.php
PHPUnit 9.6.19 by Sebastian Bergmann and contributors.

Product
✓ Get products

Time: 00:00.074, Memory: 6.00 MB

OK (1 test, 9 assertions)
```

Prueba de integración: este caso de prueba se enfoca en verificar si los diferentes componentes del código funcionan bien entre si y no hay errores o contradicciones que impidan su correcto funcionamiento



```
Jorge Vargas@LAPTOP-CMR1RJD1 MINGW64 /c/xampp/htdocs/todocalza
$ php vendor/bin/phpunit --testdox tests/IntegrationUsuarioTests.php
PHPUnit 9.6.19 by Sebastian Bergmann and contributors.

Integration Usuario Tests
✓ Iniciar sesion admin
✓ Iniciar sesion usuario

Time: 00:00.020, Memory: 6.00 MB

OK (2 tests, 4 assertions)
```

3- Identificación de funcionalidades nuevas a probar.

Se identifican las siguientes funcionalidades nuevas a probar.

- **Autenticación de usuarios.**
Se probará si el sistema permite el inicio de sesión de usuarios válidos y el bloqueo de usuarios.
- **Control de acceso:** Se probará si el sistema permite el acceso a funcionalidades específicas solo para usuarios autorizados.

4- definición de estrategias y criterios para realizar pruebas.

- **Estrategia:** pruebas unitarias utilizando phpJUnit y Mockito para simular la interacción con el servicio **`y todas sus funcionalidades`**.
- **Criterios de aceptación:** validar que todas las operaciones CRUD funcionen correctamente y que se manejen adecuadamente los casos de error y éxito.

5- IDENTIFICACION DE ESTORNOS DE TRABAJO REQUERIDOS.

- **Entorno de desarrollo:** IDE (como visual studio code).
- **Entorno de pruebas:** phpJunit y mockito.

6- Metodologías, procedimientos, cronograma y planificación de las pruebas.

Se establecerán metodologías de prueba como pruebas unitarias, pruebas de integración, pruebas de sistemas y pruebas de aceptación. Se establecerán procedimientos para cada prueba, como la documentación de casos de prueba, la revisión de pruebas y resolución de problemas. El cronograma y la planificación de las pruebas se establecerán de acuerdo con la cronología del proyecto.

- **CRONOGRAMA DE**

Semana	Tarea
1-2	Definición de casos de prueba
	- Identificar las funcionalidades clave de la aplicación.
	- Crear escenarios de prueba para cada función.
3-4	Preparación del entorno de pruebas
	- Configurar el entorno de pruebas (base de datos, servidores, etc.).
	Preparar datos de prueba (clientes ficticios, interacciones simuladas, etc.).
5-6	Ejecución de pruebas unitarias
	Realizar pruebas en módulos individuales (registro de clientes, gestión de contactos, etc.
	- Verificar que las funciones se comporten según lo esperado.
7-8	Corrección de errores y optimización
	- Identificar y corregir defectos encontrados durante las pruebas.
	- Optimizar el rendimiento de las funciones.
9-10	Validación de seguridad y autenticación
	- Verificar que los mecanismos de autenticación sean sólidos.
	- Evaluar la seguridad contra posibles vulnerabilidades.
11-12	Documentación y cierre de pruebas
	- Documentar los resultados de las pruebas unitarias.
	- Preparar informes para el equipo de desarrollo.
hecho por: jorge vargas y fabrizio martinez	

- **PLANIFICACION DE PRUEBAS.**

Semana	Tarea
1-2	Análisis del Producto
	- Comprender los requisitos funcionales y no funcionales.
	- Identificar las áreas críticas para pruebas.
3-4	Estrategia de Pruebas
	- Definir el alcance de las pruebas.
	- Identificar el tipo de pruebas a realizar.
5-6	Objetivos de Pruebas
	- Validar la funcionalidad de registro de clientes.
	- Verificar la gestión de contactos.
	- Evaluar la generación de informes.
7-8	Criterios de Prueba
	- Establecer criterios de suspensión y salida.
9-10	Recursos
	- Asignar recursos humanos y preparar recursos del sistema.
11-12	Entorno de Pruebas
	- Definir y configurar el entorno de prueba.
13-14	Programación y Estimación
	- Crear un cronograma detallado.
	- Estimar el tiempo necesario para cada tipo de prueba.
15-16	Entregables de Pruebas
	- Documentar resultados y preparar entregables.

7- Diseño de artefactos e instrumentos para el registro de pruebas.

- Utilización de anotaciones phpJunit para identificar y organizar casos de prueba.
- Uso de mockito para simular objetos y comportamiento de las pruebas.

8- Selección de utilidades o herramientas para implementar las pruebas.

- **PhpJUnit:** para escribir y ejecutar pruebas unitarias.
- **Mockito:** para simular objetos y comportamientos de las pruebas.

9- Identificación de riesgos y contingencias:

- **Riesgos:** fallos en la lógica de negocios de las operaciones CRUD.
- **Contingencia:** realizar pruebas exhaustivas con casos de prueba que cubran diferentes escenarios (Éxito, error) y realizar pruebas de integración después de las pruebas unitarias

CONCLUSIONES.

El propósito real de las pruebas unitarias va más allá de validar que cada unidad de trabajo se comporte como se espera. Su verdadera ventaja es proporcionarte retroalimentación casi instantánea sobre el diseño y la implementación de tu código. La facilidad con la que puede crear pruebas unitarias te dice mucho sobre la calidad de tu código y como las has diseñado.

En resumen las pruebas unitarias son una herramienta poderosa para garantizar la calidad y funcionalidad de tu código.

REFERENCIAS.

1- Google académico:

<https://scholar.google.es/schhp?hl=es>

2- Biblioteca Sena.