PRUEBAS UNITARIAS

Las pruebas unitarias o unit testing, forman parte de los diferentes procedimientos que se pueden llevar a cabo dentro de la metodología ágil. Son principalmente trozos de código diseñados para comprobar que el código principal está funcionando como esperábamos. Pequeños test creados específicamente para cubrir todos los requisitos del código y verificar sus resultados.

El proceso que se lleva a cabo, consta de tres partes. El Arrange, donde se definen los requisitos que debe cumplir el código principal. El Act, el proceso de creación, donde vamos acumulando los resultados que analizaremos. Y el Asert, que se considera el momento en que comprobamos si los resultados agrupados son correctos o incorrectos. Dependiendo del resultado, se valida y continúa, o se repara, de forma que el error desaparezca.

CARACTERÍSTICAS

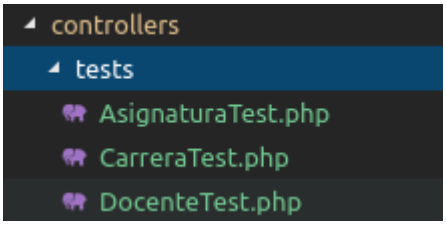
- Automatizable; Aunque los resultados deben ser específicos de cada test unitario desarrollado, los resultados se pueden automatizar, de forma que podemos hacer las pruebas de forma individual o en grupos.
- Completas; El proceso consta de pequeños test sobre parte del código, pero al final, se debe comprobar su totalidad.
- Repetibles; En el caso de repetir las pruebas de forma individual o grupal, el resultado debe ser siempre el mismo dando igual el orden en que se realicen los test, los tests se almacenan para poder realizar estas repeticiones o poder usarlos en otras ocasiones.
- Independientes; Es un código aislado que se ha creado con la misión de comprobar otro código muy concreto.

VENTAJAS

1. Proporciona un trabajo ágil; Como procedimiento ágil que es, permite poder detectar los errores a tiempo, de forma que se pueda reescribir el código o corregir errores sin necesidad de tener que volver al principio y rehacer el trabajo.
2. Calidad del código; Al realizar pruebas continuamente y detectar los errores, cuando el código está terminado, es un código limpio y de calidad.
3. Detectar errores rápido; A diferencias de otros procesos, los tests unitarios nos permiten detectar los errores rápidamente, analizamos el código por partes, haciendo pequeñas pruebas y de manera periódica, además, las pruebas se pueden realizar las veces que hagan falta hasta obtener el resultado óptimo.
4. Facilita los cambios y favorece la integración; nos permiten modificar partes del código sin afectar al conjunto, simplemente para poder solucionar bugs que nos encontramos por el camino. Al estar desglosados en bloques individuales permiten la integración de nuevas aportaciones para hacer un código más complejo o actualizarlo en función de lo que el cliente demande.
5. Proporciona información; debido al continuo flujo de información y la superación de errores, se puede recopilar gran cantidad de información para evitar bugs venideros.
6. Testeando una porción del código, también se puede saber qué requisitos se deben cumplir, y por eso será mucho más fácil llegar a una cohesión entre el código y el diseño.
7. Reduce costos; partiendo de la base que los errores se detectan a tiempo, lo cual implica tener que escribir menos código, poder diseñar a la vez que se crea y optimizar los tiempos.

IMPLEMENTACIÓN

Para el caso, se utilizó la herramienta phpUnit que viene incorporada al framework Codeigniter, en el cual creamos una carpeta llamada tests dentro de la carpeta controllers y procedemos a crear clases del tipo test, una para cada clase que tengamos entre los controladores



Dentro de la clase asignaturaTest tenemos un constructor que carga una librería perteneciente a unit test, un acceso a la base de datos y por ultimo el modelo de la asignatura a la que haremos referencia luego.

```php
class AsignaturaTest extends CI_Controller
{

    public function __construct()
    {
        parent::__construct();
        $this->load->library('unit_test');
        $this->load->database();
        $this->load->model("Asignatura_model");
    }

    public function index()
    {
        echo "Pruebas Unitarias Asignaturas";

        $this->unit->run($this->asignaturaIdNombre(1),
                         "1 - Análisis Matemático I",
                         "Nombre de asignatura");

        $this->unit->run($this->asignaturaIdEquipoDocente(8),
                         "Asignatura 8- Chade, Pablo - Barros Olivera, Ruy Enio - Gómez, Fabian ",
                         "Equipo docente de una asignatura");

        $this->load->view('tests/tests');
    }
```

Luego se encuentra una función llamada index donde se realizan las pruebas. Se utiliza la función unit->run propia de la librería unit_test, la cual recibe 3 parámetros:

```
$this->unit->run($this->asignaturaIdNombre(1),
                 "1 - Análisis Matemático I",
                 "Nombre de asignatura");
```

1- $this→asignaturaIdNombre - La función a evaluar

2- “1 – Análisis Matemático I” - El resultado esperado

3- “Nombre de asignatura” - El nombre de la prueba

Con la función load->view se carga la vista:

Pruebas Unitarias Asignaturas

Nombre de prueba	Nombre de asignatura
Tipo de datos de prueba	String
Tipo de datos esperados	String
Resultado	Aprobado
Nombre fichero	/var/www/html/eticPHP/application/controllers/tests/AsignaturaTest.php
Número de línea	20
Notas	

Nombre de prueba	Equipo docente de una asignatura
Tipo de datos de prueba	String
Tipo de datos esperados	String
Resultado	Aprobado
Nombre fichero	/var/www/html/eticPHP/application/controllers/tests/AsignaturaTest.php
Número de línea	24
Notas	

En la vista se puede ver el nombre de la prueba, el tipo de datos de prueba, el tipo de datos esperado y el resultado el cual en ambos casos es Aprobado.