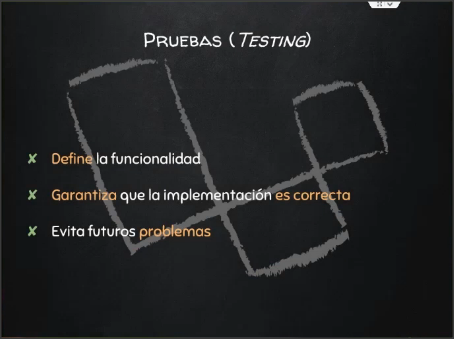
# Testing en PHP



Para realizar las pruebas se utilizará php unit, un framework orientado a hacer pruebas en php.

Como ya está instalado composer, este se descarga de la siguiente manera:



El - - dev se debe a que esto solamente lo requerimos para un ambiente de desarrollo

Existía un error de dependencia (phpunit requires ext-dom ) el cual fue solucionado con:

How do I install the dom extension for PHP7

apt-get install php-xml

Nota: además de un problema de permisos que fue solucionado con:

Composer config –g secure-http false

Composer.json nos queda de la siguiente forma:



Para utilizar phpunit, primero:

En la consola ejecutamos:

|  |
| --- |
| ./vendor/bin/phpunit |

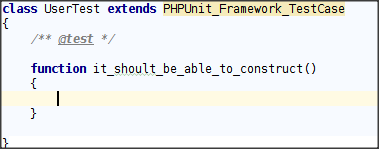
Luego en nuestra carpeta test, se usa la convención NombreClase\_test.php.

Existen dos formas de programar un test:

La primera es que se defina en un comentario que es un test y luego programar la función.

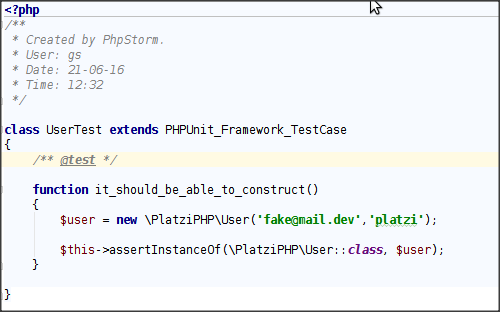
La segunda es no poner el comentario (phpDocs) y entonces el nombre de nuestra función va a estar sujeta a comenzar con test\_

Ej:



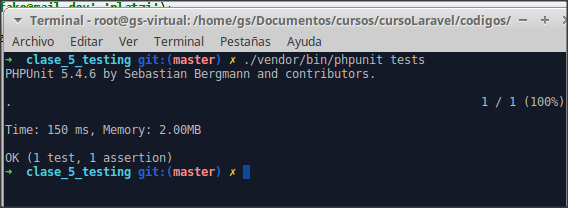
En este caso no fue necesario comenzar el nombre de la función con test, sino que el comentario fue suficiente para que el framework entienda que se trata de un test.

Ejemplo de si el objeto creado es realmente una instancia

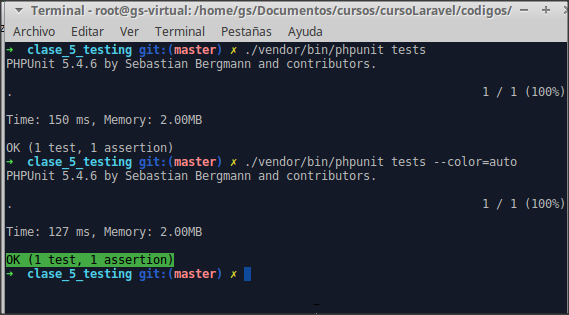


Como los test están en la carpeta tests, se ejecuta el programa por consola indicándole la carpeta.

Este sería el output



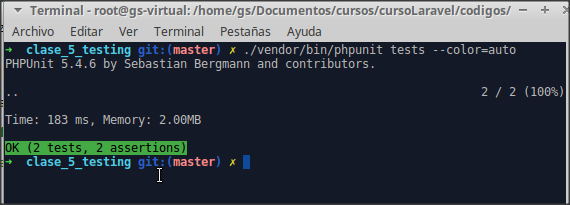
Para que sea un caso de mejor usabilidad se le puede agregar color:



Se pueden agregar más tests



Luego:



Nota:

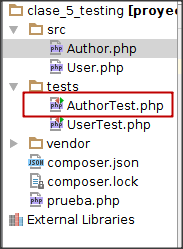
Además existe la opción de realizar esta tarea al revés, ósea realizar un “test driven development”

En donde se programan los tests, esperando un funcionamiento y luego se crea ese funcionamiento.

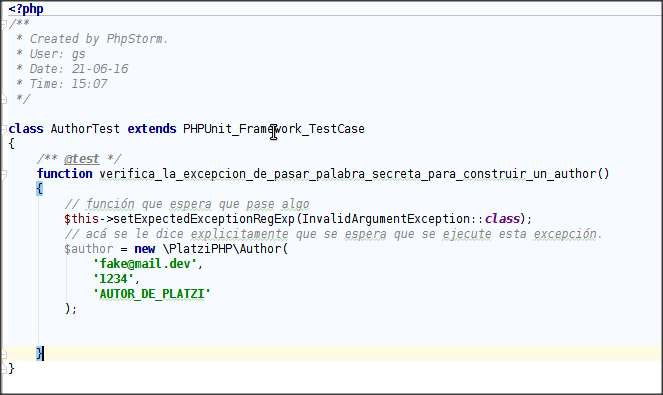
Además se puede generar una prueba que realice una verificación de un manejo de error:

Observar el siguiente ejemplo:

Se crea en la carpeta tests el siguiente archivo.

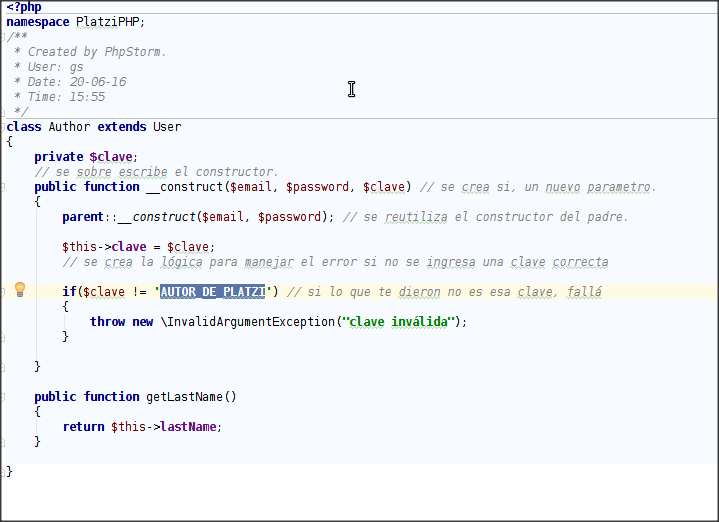


Luego:

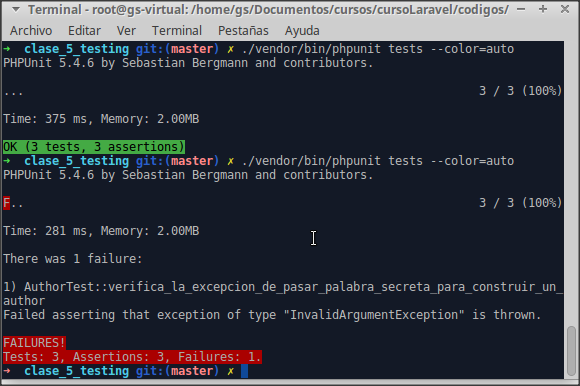


Se crea el objeto author y se le da tres parámetros.

En la clase Author vemos lo siguiente:



En este caso, está correcto por tanto la excepción no se lanza. Se podría decir que está correcto, pero en el test eso no es lo que queremos, por tanto al ejecutar sucederá lo siguiente:



Pero si creamos el objeto con el parámetro erróneo, lo que ocurre es que se ejecuta la excepción y el test pasa la prueba.

