Instituto Politécnico Superior – 6º Informática 9/6/2019
Aseguinolaza Luis
Uliassi Manuel

Trabajo Practico Integrador

Informe

1)

El archivo *.gitignore* es una herramienta de GIT que se utiliza para ignorar todos aquellos directorios y archivos que estén especificados dentro de este. El archivo debe estar en la carpeta raíz del repositorio, con este nombre y GIT automáticamente no guardara cambios de esos archivos. Una excepción, es cuando subimos un archivo y luego lo ignoramos. Al suceder esto, GIT seguirá tomando los cambios de esos archivos independientemente de estar en *.gitignore* o no.

2)

travis.yml se utiliza como un archivo que contiene las instrucciones para que Travis.CI haga las pruebas correspondientes al repositorio en donde creemos este archivo y ejecutemos la aplicación Travis.CI. El archivo se debe crear en el repositorio donde queremos hacer los tests. Sin este archivo Travis.CI no va a poder ejecutar su test.

Líneas:

1: se especifica el lenguaje del archivo al que testea.

2 y 3: se especifica la versión del lenguaje

5 y 6: Composer es una herramienta de instalación de paquetes de PHP.

Con el argumento "--prefer-source" se le solicita que actualice los paquetes que se encuentran dentro del archivo *composer.lock* dentro de la misma carpeta donde se encuentra el archivo *.travis.yml*

8 y 9:el comando script especifica el script a ejecutar para la construcción del proyecto para testear.

- php vendor/bin/phpunit --color tests especifica que script y que la salida de los tests siempre tengan colores.

3)

composer.json define las dependencias y requerimientos de un proyecto. *autoload* sirve como un cargador automático de clases para nuestro proyecto.

"psr-4": { "Bingo\\": "src" } está indicando que las clases del namespace "Bingo" (que se encuentran en ambos tests) se encuentran en el directorio "src" que está en la raíz.

"require": { "phpunit/phpunit": "^6" } esta diciendo que phpunit requiere la version 6 en este proyecto.

composer.lock registra todos los paquetes instalados y la versión instalada de cada uno de ellos. Para ello usa una serie grande de especificaciones en donde podemos encontrar por ejemplo el nombre de la versión, la fecha, el campo "autoload", el campo "psr-4", el campo "require", etc.

Lo que sucede al ejecutar un test es que la primera vez lee las dependencias de nuestro proyecto en *composer.json*, instala los paquetes, crea el archivo *composer.lock* en donde va anotando todos los paquetes instalados y las versiones instaladas de cada uno de ellos y las próximas veces que se ejecuta el test instala los paquetes con sus versiones correspondientes especificados en el archivo *composer.lock*.

4)

Una alternativa para Node.js es el gestor de paquetes NPM (Node Packaged Manager) la cual se maneja a través de módulos llamados NPMs (Node Packaged Modules) y estos corresponden a librerías que pueden ser compartidas y reutilizadas entre diferentes proyectos.

Una alternativa para Ruby es Bundler es un entorno para los proyectos de Ruby para gestionar versiones de un proyecto de Ruby.

5)

Namespace: Espacio de nombre.

Se utiliza para referenciar archivos entre si. Solo sirve para ordenar archivos. Por ejemplo, Si dos archivos se encuentran dentro del mismo namespace, pueden utilizar (de ser públicas) las funciones del otro archivo. De esta manera podemos tener funciones con el mismo nombre, o archivos con el mismo nombre, pero podemos hacerles referencias mediantes distintos namespaces.

Si lo quitamos, las clases que necesiten implementar interfaces fallaran, ya que no encontraran la interface.

6)

// {@inheritdoc}: Inherit: herencia / doc: Documentacion.

Esta etiqueta hace referencia a la documentación de la función "padre". Es decir, cuando estemos documentando una función, podemos importar la documentación de la función de la cual hereda. De esta manera, tenemos una descripción más amplia en la función padre, y luego una más específica en la función heredera, conteniendo a la documentación de la función padre.

7)

extends TestCase:

Las clases del directorio tests extienden de la clase TestCase, porque justamente son tests que PHPUnit debe correr. Necesitan extender de esta clase para poder utilizar el método *assertTrue().*

Que una clase extienda de otra significa que la clase hijo hereda todos los métodos y propiedades de la clase padre, es decir, que la clase hijo puede utilizar todas las funciones y variables que estén declarados dentro de la clase padre.