edix
Digital Workers

Actividad de desarrollo: Unit testing

DEVOPS - UF2

Unit testing

Añadir nuevas capacidades a nuestros sistemas de forma robusta y rápida es posible si se cuenta con una buena batería de tests.

Enunciado

Se pide al alumno extender la aplicación web creada en el anterior ejercicio:

- Se añadirán test de tipo **unitario** que nos ayuden a verificar el funcionamiento de nuestros endpoints.
- Se añadirá un script que facilite la provisión del servicio, cuya invocación permite comenzar a hacer peticiones al mismo.

Para saber si nuestra batería de test es suficiente, se utilizan herramientas de **medición de cobertura**, que miden qué porcentaje de nuestro código se recorre con los test que ejecutamos.

Herramientas de medición de cobertura

Se exponen ejemplos de librerías o utilidades comunes de medición de la cobertura de testing (entre otras características) en los tres lenguajes que utilizamos en el curso.

Si el alumno está familiarizado con cualquier otra herramienta que provea la misma funcionalidad, es totalmente válido su uso, siempre y cuando esté documentado cómo conseguir un reporte.

Python

Existe la librería coverage, muy sencilla de usar y que funciona con los runners habituales. Un ejemplo con *py.test:*

```
pip install 'coverage==5.5'
coverage run -m pytest --source .
coverage report -m
coverage html
# En Mac es open, en Linux se usa xdg-open
open htmlcov/index.html
```

Java

Se puede emplear JaCoCo, fácilmente configurable en Maven y Gradle, que permite elaborar reportes en formato HTML, al igual que la herramienta anterior.

Formato de entrega

Se entregará una carpeta que contendrá el código fuente pedido, así como instrucciones para su construcción y validación:

- Debe especificarse el **sistema operativo** necesario para esto:
    - Alternativamente, en cuál(es) no se ha podido probar o en los que se sabe que **no** funciona.
- En caso de ser necesario, usar alguna herramienta de gestión de ciclo de vida (Maven, PIP, npm…) deben figurar las instrucciones para su instalación.
- En caso de utilizar Docker, deben figurar los comandos empleados para construir y levantar el servicio.

Alternativamente, se puede subir el código a cualquier VCS público:

- No es necesario que sea un repositorio privado,que normalmente están limitados o son de pago.

Criterios de calificación

La actividad será correcta siempre que:

- Existan test que garanticen una cobertura de código **de al menos el 50%.**
- Se haya aportado y documentado un script para levantar el servicio de forma autónoma.

Cumplidos estos criterios, **la nota será igual al porcentaje de cobertura** que arrojen las herramientas de análisis.

Se valorará de forma **muy** positiva:

- La presencia de otros tipos de test.
- La claridad y documentación en el código entregado.
- Que se haya refactorizado el código entregado previamente para facilitar el testing.
- La presencia de cualquier otro *script* de Bash/Powershell, como por ejemplo para parar/reiniciar el servicio o cambiar parámetros de ejecución: puerto, número de instancias, nombre de fichero de datos...