**2. Descripción de las tecnologías**

A continuación, vamos a analizar brevemente cada una de las tecnologías seleccionadas en el trabajo anterior, PHPUnit y Codeception.

**2.2 Descripción de la tecnología Codeception**

Codeception recopila y comparte las mejores prácticas y soluciones para probar aplicaciones web PHP. Con un conjunto flexible de módulos incluidos, las pruebas son fáciles de escribir, de usar y de mantener. Codeception anima a los desarrolladores e ingenieros de control de calidad a concentrarse en las pruebas y no en crear un conjunto de pruebas. Permite realizar los tres grandes tipos de pruebas: unitarias, funcionales y de aceptación. Permite escribir pruebas de unidad, funcionales, de integración y de aceptación en un estilo único y coherente. Todas las pruebas de Codeception están escritas de manera descriptiva, es decir, con solo mirar el cuerpo de prueba, se puede comprender claramente qué se está probando y cómo se realiza.



Figura 1

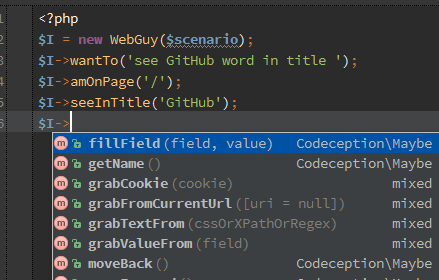


Figura 2

URL1: <https://leehblue.com/how-to-use-stepobjects-in-cest-classes/>

URL2: <https://codeception.com/11-20-2013/webdriver-tests-with-codeception.html>

**3.2 Categoría B: Uso de Herramientas**

3.2.1 Criterio B.1: Dificultad para la puesta en marcha y configuración inicial

Nombre del criterio: Puesta en marcha

Descripción: este criterio identifica la complejidad a la hora de instalar, configurar e iniciar el framework.

Tipo de valor: Alta / media / baja

3.2.2 Criterio B.2: Prueba de mutación con librería Infection

Nombre del criterio: Uso de librería Infection

Descripción: Las pruebas de mutación son un tipo de pruebas realizadas sobre los test unitarios que de manera intencionada añaden defectos a tu código y detectan si tus tests reaccionan a estos cambios. Si un código modificado hace fallar el test, se denomina que se ha matado a la mutación, en caso contrario significa que tu test no cubre todos los casos de uso y no cubre la regresión de las pruebas.

Con este tipo de pruebas se pretende comprobar la calidad de los test y no su cobertura, demostrando que tener un 100% de cobertura de código no implica que tu código esté probado correctamente, sólo indica que las ejecuciones de los test han pasado por todas las líneas de código al menos una vez, pero no que estas líneas hayan sido probadas de manera certera.

Tipo de valor: Booleano (Si/No).

3.2.3 Criterio B.3: Prueba de mutación con librería Humbug

Nombre del criterio: Uso de librería Humbug

Descripción: Las pruebas de mutación son un tipo de pruebas realizadas sobre los test unitarios que de manera intencionada añaden defectos a tu código y detectan si tus tests reaccionan a estos cambios. Si un código modificado hace fallar el test, se denomina que se ha matado a la mutación, en caso contrario significa que tu test no cubre todos los casos de uso y no cubre la regresión de las pruebas.

Con este tipo de pruebas se pretende comprobar la calidad de los test y no su cobertura, demostrando que tener un 100% de cobertura de código no implica que tu código esté probado correctamente, sólo indica que las ejecuciones de los test han pasado por todas las líneas de código al menos una vez, pero no que estas líneas hayan sido probadas de manera certera.

Tipo de valor: Booleano (Si/No).

3.2.4 Criterio B.4: Análisis de cobertura de código con XDebug

Nombre del criterio: Análisis de cobertura de código de los test realizados utilizando XDebug

Descripción: Este criterio indica si el framework permite analizar el % del código que cubren los test realizados en un proyecto php. Para ello vamos a utilizar la librería XDebug, de uso común para este tipo de métricas.

Tipo de valor: Booleano (Si/No).

3.2.5 Criterio B.5: Velocidad del análisis de cobertura

Nombre del criterio: Velocidad del análisis de cobertura de código

Descripción: Este criterio mide la velocidad con la que se ejecuta el análisis de cobertura de código descrito en el apartado anterior.

Tipo de valor: Alta / media / baja

3.2.6 Criterio B.6: Complejidad para analizar los resultados de los tests

Nombre del criterio: Complejidad para analizar los resultados de los test.

Descripción: En este criterio se evalúa la facilidad de interpretación de los resultados de los tests que se han lanzado.

Tipo de valor: Alta / media / baja

**4.2 Evaluación de los criterios para la tecnología Codeception**

|  |  |
| --- | --- |
| **CRITERIOS** | **EVALUACIÓN** |
| Criterio A.1: Precio de la herramienta | Gratuita |
| Criterio A.2: Idioma de la herramienta | Inglés |
| Criterio A.3: Popularidad de la herramienta | Alta |
| Criterio A.4: Licencia de la herramienta | Open Source (Licencia MIT) |
| Criterio A.5: Documentación disponible | Media |
| Criterio B.1: Dificultad para la puesta en marcha | Alta |
| Criterio B.2: Prueba de mutación con librería Infection | No |
| Criterio B.3: Prueba de mutación con librería Humbug | Si |
| Criterio B.4: Análisis de cobertura de código con XDebug | Sí |
| Criterio B.5: Velocidad del análisis de cobertura de código | Alta |
| Criterio B.6: Complejidad para analizar los resultados | Baja |
| Criterio C.1: Velocidad pruebas unitarias | Alta |
| Criterio C.2: Velocidad pruebas funcionales | Alta |
| Criterio C.3: Velocidad pruebas de aceptación | Baja |
| Criterio C.4: Velocidad pruebas de integración | Media |
| Criterio C.5: Velocidad pruebas de regresión | Baja |
| Criterio C.6: Velocidad pruebas de estrés | Media |
| Criterio C.7: Velocidad pruebas de rendimiento | Alta |
| Criterio D.1: Pruebas unitarias | Si |
| Criterio D.2: Pruebas funcionales | Si |
| Criterio D.3: Pruebas de aceptación | Si |
| Criterio D.4: Pruebas de integración | Si |
| Criterio D.5: Pruebas de regresión | Si |
| Criterio D.6: Pruebas de estrés | Si |
| Criterio D.7: Pruebas de rendimiento | Si |

**5. Comparación de las tecnologías**

|  |  |  |  |
| --- | --- | --- | --- |
| **CRITERIOS** | **PHPUnit** | **Codeception** | **COMENTARIOS** |
| A.1 | Gratis | Gratis | Se pueden hacer donativos. |
| A.2 | Inglés | Inglés |  |
| A.3 | Alta | Alta |  |
| A.4 | BSD | Open Source | Ambas licencias están relacionadas con el software libre. En el caso de Codeception es MIT license. |
| A.5 | Alta | Media | En el caso de codeception se limita a prácticamente a la documentación oficial. |
| B.1 | Baja | Alta | Codeception requiere una configuración inicial más compleja. |
| B.2 | Si | No | La librería Infection estará próximamente disponible en Codeception. |
| B.3 | Si | Si |  |
| B.4 | Si | Si |  |
| B.5 | Baja | Alta | Un análisis con Codeception puede tardar algo más de 1 min, mientras que en PHPunit estaría en torno a 15 min. |
| B.6 | Media | Baja | Los resultados con codeception son fácilmente interpretables por usuarios sin conocimientos técnicos. |
| C.1 | Alta | Alta | Al ser pruebas unitarias (generales) se realizan rápidamente, no se obtienen diferencias notables. |
| C.2 | Alta | Alta | Del mismo modo que en el caso anterior, ambas pruebas funcionales se realizan rápidamente, no encontramos diferencias destacables como para decantarnos por una tecnología u otra por esta característica. |
| C.3 | Baja | Media | En este caso si tenemos una pequeña diferencia de rapidez en las pruebas de aceptación. Son más rápidas en el caso de Codeception. |
| C.4 | Baja | Media | Las pruebas de integración combinan los distintos módulos de una aplicación. Esto es una tarea compleja que depende en cierta medida de como estén codificadas las pruebas. Además, también influye como de optimizadas estén las librerías que use el entorno de pruebas y por lo tanto la velocidad de las mismas está influida por el tipo de entorno que se esté usando. En este caso Codeception usa librerías más optimizadas. |
| C.5 | Baja | Baja | La velocidad de las pruebas de regresión está fuertemente relacionada con lo optimizado que se encuentre el entorno de trabajo que se esté utilizando. Las pruebas de regresión consisten en probar completamente una versión nueva de una aplicación y por lo tanto suele ser un proceso lento. |
| C.6 | Media | Media | Las pruebas de estrés consisten en llevar a condiciones extremas a la aplicación. El gran volumen de datos hace que esta tarea conlleve un tiempo de ejecución medio. Ambos entornos de trabajo están preparados para optimizar el tiempo en este tipo de pruebas. |
| C.7 | Alta | Alta | Las pruebas de rendimiento están relacionadas con las pruebas de estrés. La diferencia es que no suelen llevar a condiciones extremas a la aplicación, sino que se estudia el tiempo de respuesta de la misma. En este sentido, ambos entornos están muy optimizados en este tipo de pruebas. |
| D.1 | Si | Si | Ambos entornos permiten configurar mediante una plantilla genérica cualquier tipo de pruebas. |
| D.2 | Si | Si | Ambos entornos permiten configurar mediante una plantilla genérica cualquier tipo de pruebas. |
| D.3 | Si | Si | Ambos entornos permiten configurar mediante una plantilla genérica cualquier tipo de pruebas. |
| D.4 | Si | Si | Ambos entornos permiten configurar mediante una plantilla genérica cualquier tipo de pruebas. |
| D.5 | Si | Si | Ambos entornos permiten configurar mediante una plantilla genérica cualquier tipo de pruebas. |
| D.6 | Si | Si | Ambos entornos permiten configurar mediante una plantilla genérica cualquier tipo de pruebas. |
| D.7 | Si | Si | Ambos entornos permiten configurar mediante una plantilla genérica cualquier tipo de pruebas. |