TRABAJO EN GRUPO 2: AUTOMATED TESTING FRAMEWORKS EN PHP

Grupo T8

Daniel Manzano Estébanez

Inés López Baldominos

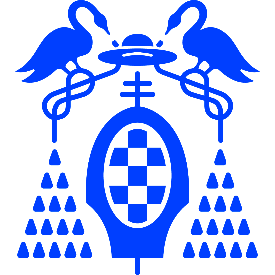
José María Sanz Górriz

Adrián de la Hoz Casanova

Cristian Abellán Madrigal

Universidad de Alcalá

Desarrollo de Tecnologías Emergentes



Contenido

[1. Autores del trabajo, planificación y entrega 3](#_Toc5657726)

[1.1 Autores 3](#_Toc5657727)

[1.2 Planificación 3](#_Toc5657728)

[1.3 Entrega 4](#_Toc5657729)

[2. Descripción de las tecnologías 4](#_Toc5657730)

[2.1 Descripción de la tecnología PHPUnit 4](#_Toc5657731)

[2.2 Descripción de la tecnología Codeception 5](#_Toc5657732)

[3. Criterios de comparación 6](#_Toc5657733)

[3.1 Categoría A: General 6](#_Toc5657734)

[3.1.1 Criterio A.1: Precio de la herramienta 6](#_Toc5657735)

[3.1.2 Criterio A.2: Idiomas de la herramienta 6](#_Toc5657736)

[3.1.3 Criterio A.3: Popularidad de la herramienta 6](#_Toc5657737)

[3.1.4 Criterio A.4: Licencia de la herramienta 6](#_Toc5657738)

[3.1.5 Criterio A.5: Documentación disponible 6](#_Toc5657739)

[3.2 Categoría B: Uso de Herramientas 7](#_Toc5657740)

[3.2.1 Criterio B.1: Dificultad para la puesta en marcha y configuración inicial 7](#_Toc5657741)

[3.2.2 Criterio B.2: Prueba de mutación con librería Infection 7](#_Toc5657742)

[3.2.3 Criterio B.3: Prueba de mutación con librería Humbug 7](#_Toc5657743)

[3.2.4 Criterio B.4: Análisis de cobertura de código con XDebug 7](#_Toc5657744)

[3.2.5 Criterio B.5: Velocidad del análisis de cobertura 8](#_Toc5657745)

[3.2.6 Criterio B.6: Complejidad para analizar los resultados de los tests 8](#_Toc5657746)

[3.3 Categoría C: Tipos de pruebas soportadas 8](#_Toc5657747)

[3.3.1 Criterio C.1: Pruebas unitarias 8](#_Toc5657748)

[3.3.2 Criterio C.2: Pruebas funcionales 8](#_Toc5657749)

[3.3.3 Criterio C.3: Pruebas de aceptación 8](#_Toc5657750)

[3.3.4 Criterio C.4 Pruebas de integración 9](#_Toc5657751)

[3.3.5 Criterio C.5 Pruebas de regresión 9](#_Toc5657752)

[3.3.6 Criterio C.6 Pruebas de estrés 9](#_Toc5657753)

[3.3.7 Criterio C.7 Pruebas de rendimiento 9](#_Toc5657754)

[3.4 Categoría D: Velocidad de ejecución de prueba 10](#_Toc5657755)

[3.4.1 Criterio D.1: Velocidad pruebas unitarias 10](#_Toc5657756)

[3.4.2 Criterio D.2: Velocidad pruebas funcionales 10](#_Toc5657757)

[3.4.3 Criterio D.3: Velocidad pruebas de aceptación 10](#_Toc5657758)

[3.4.4 Criterio D.4: Velocidad pruebas de integración 10](#_Toc5657759)

[3.4.5 Criterio D.5: Velocidad pruebas de regresión 11](#_Toc5657760)

[3.4.6 Criterio D.6: Velocidad pruebas de estrés 11](#_Toc5657761)

[3.4.7 Criterio D.7: Velocidad pruebas de rendimiento 11](#_Toc5657762)

[4. Evaluación de los criterios por tecnología 11](#_Toc5657763)

[4.1 Evaluación de los criterios para la tecnología PHPUnit 11](#_Toc5657764)

[4.2 Evaluación de los criterios para la tecnología Codeception 12](#_Toc5657765)

[5. Comparación de las tecnologías 12](#_Toc5657766)

[6. Recomendaciones 15](#_Toc5657767)

[6.1 Situación 1 15](#_Toc5657768)

[6.1.1 Descripción de la situación 15](#_Toc5657769)

[6.1.2 Recomendación de tecnología a utilizar 15](#_Toc5657770)

[6.2 Situación 2 16](#_Toc5657771)

[6.2.1 Descripción de la situación 16](#_Toc5657772)

[6.2.2 Recomendación de tecnología a utilizar 16](#_Toc5657773)

# 1. Autores del trabajo, planificación y entrega

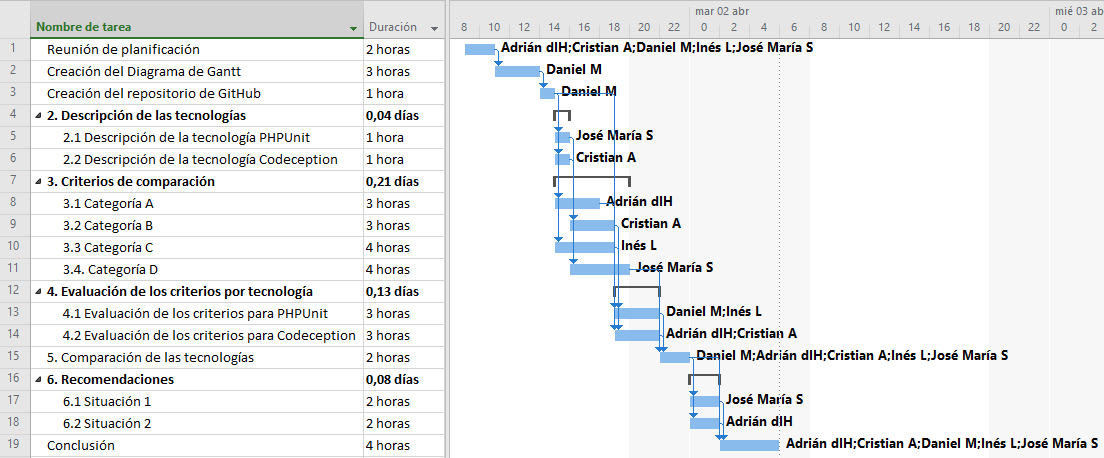
## 1.1 Autores

Grupo T8

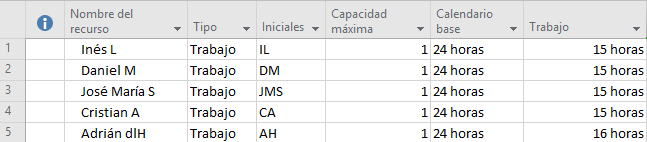
* Daniel Manzano Estébanez (Coordinador)
* Cristián Abellán Madrigal
* Inés López Baldominos
* José María Sanz Górriz
* Adrián de la Hoz Casanova

## 1.2 Planificación

Para establecer la planificación se ha elaborado un diagrama de Gantt con Microsoft Project. Al igual que en el trabajo anterior, cada miembro del grupo debe sumar por lo menos 15 horas de trabajo entre las tareas que tenga asignadas. El diagrama es el siguiente.



La última tarea se corresponde con la unificación y comprobación del trabajo, que es realizada por todos los miembros del grupo.



En esta última imagen se puede apreciar el total de horas que acumula cada participante, quedando los totales balanceados en torno a las 15 horas.

El archivo de Microsoft Project correspondiente a la planificación se encontrará en el repositorio del trabajo, cuya dirección se puede encontrar en el siguiente apartado de este documento.

## 1.3 Entrega

Del mismo modo que en el trabajo TG1, se ha creado un repositorio en GitHub siguiendo las mismas pautas, donde se encontrará la plantilla del trabajo, el trabajo individual de cada miembro y el documento final y presentación del mismo.

El enlace URL es el siguiente: <https://github.com/daniel-manzanoe/TG2>

# 2. Descripción de las tecnologías

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

## 2.1 Descripción de la tecnología PHPUnit

PHPUnit es un framework open source para el desarrollo, orientado a pruebas o Test-driven development (TDD) para cualquier código PHP. Es decir, es un framework que nos ayuda a probar nuestro código creando pequeñas unidades que revisan funcionalidades puntuales del código y probando que funcionan como deben, además de la posibilidad de automatizar estas pruebas para ejecutarlas frecuentemente, tanto como el código cambie.



Figura 1

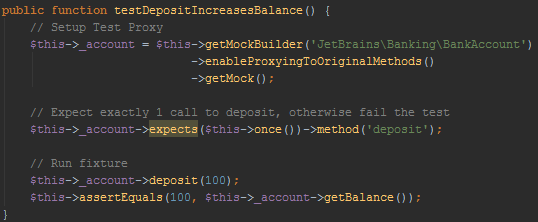


Figura 2

URL1,2: <https://blog.jetbrains.com/phpstorm/2014/05/phpunit-4-in-phpstorm-8-eap/>

## 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.

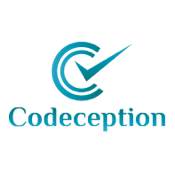


Figura 1

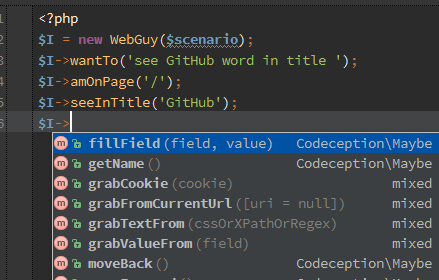


Figura 2

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

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

# 3. Criterios de comparación

Para comparar ambas tecnologías y tener una visión general de las funcionalidades, ventajas e inconvenientes de cada una se establecen los siguientes criterios:

## 3.1 Categoría A: General

### 3.1.1 Criterio A.1: Precio de la herramienta

Nombre del criterio: Precio

Descripción: se trata del coste que supone la adquisición de la herramienta a utilizar.

Tipo de valor: Alto / Medio / Bajo.

### 3.1.2 Criterio A.2: Idiomas de la herramienta

Nombre del criterio: Idiomas

Descripción: indica los idiomas en los que se encuentra disponible la herramienta que se va a utilizar.

Tipo de valor: Texto libre.

### 3.1.3 Criterio A.3: Popularidad de la herramienta

Nombre del criterio: Popularidad

Descripción: muestra la popularidad o el renombre con el que cuenta la herramienta utilizada en el sector que nos encontramos, en este caso Testing PHP.

Tipo de valor: Alto / Medio / Bajo.

### 3.1.4 Criterio A.4: Licencia de la herramienta

Nombre del criterio: Licencia

Descripción: muestra la licencia de software que posee la herramienta.

Tipo de valor: BSD / Freeware / Shareware / Software Comercial / OpenSource.

### 3.1.5 Criterio A.5: Documentación disponible

Nombre del criterio: Documentación

Descripción: identifica la cantidad de documentación disponible para la utilización de una herramienta.

Tipo de valor: Alto / Medio / Bajo.

## 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

## 3.3 Categoría C: Tipos de pruebas soportadas

### 3.3.1 Criterio C.1: Pruebas unitarias

Nombre del criterio: Soporta pruebas unitarias

Descripción: Define si el entorno de testeo automatizado para PHP es capaz de ejecutar las pruebas unitarias. Se basan en comprobar que una función devuelve el resultado esperado en base a unas condiciones conocidas.

Tipo de valor: Booleano (Si/No).

### 3.3.2 Criterio C.2: Pruebas funcionales

Nombre del criterio: Soporta pruebas funcionales

Descripción: Define si el entorno de testeo automatizado para PHP es capaz de ejecutar las pruebas funcionales. En este caso, se comprueba el comportamiento de cada función con pruebas de caja negra.

Tipo de valor: Booleano (Si/No).

### 3.3.3 Criterio C.3: Pruebas de aceptación

Nombre del criterio: Soporta pruebas de aceptación

Descripción: Define si el entorno de testeo automatizado para PHP es capaz de ejecutar las pruebas de aceptación. Estas pruebas determinan si el código cumple con los requisitos de las empresas y sus usuarios. ​

Tipo de valor: Booleano (Si/No).

### 3.3.4 Criterio C.4 Pruebas de integración

Nombre del criterio: Soporta pruebas de integración

Descripción: Define si el entorno de testeo automatizado para PHP es capaz de ejecutar las pruebas de integración. Lo que prueban es que todos los elementos unitarios que componen el software, funcionan juntos correctamente probándolos en grupo.

Tipo de valor: Booleano (Si/No).

### 3.3.5 Criterio C.5 Pruebas de regresión

Nombre del criterio: Soporta pruebas de regresión

Descripción: Define si el entorno de testeo automatizado para PHP es capaz de ejecutar las pruebas de regresión. Las pruebas de regresión son pruebas de un programa o componente previamente probado que ha sufrido modificaciones, para asegurarse que no se han introducido o descubierto defectos en áreas del software que no han sido modificadas como resultado de los cambios realizados.

Tipo de valor: Booleano (Si/No).

### 3.3.6 Criterio C.6 Pruebas de estrés

Nombre del criterio: Soporta pruebas de estrés

Descripción: Define si el entorno de testeo automatizado para PHP es capaz de ejecutar las pruebas de estrés. En este tipo de prueba se va aumentando el número de usuarios de la aplicación hasta que deja de funcionar.

Tipo de valor: Booleano (Si/No).

### 3.3.7 Criterio C.7 Pruebas de rendimiento

Nombre del criterio: Soporta pruebas de rendimiento

Descripción: Define si el entorno de testeo automatizado para PHP es capaz de ejecutar las pruebas de rendimiento. Permiten medir la velocidad de ejecución de una serie de tareas en un sistema, fijando distintas condiciones. ​

Tipo de valor: Booleano (Si/No)

## 3.4 Categoría D: Velocidad de ejecución de prueba

### 3.4.1 Criterio D.1: Velocidad pruebas unitarias

Nombre del criterio: Velocidad de la ejecución al realizar las pruebas unitarias

Descripción: Velocidad que se demora el entorno de testeo automatizado para PHP en ejecutar las pruebas unitarias. Este valor se determina en función del tiempo que tarda el resto de software similares a este en realizar el mismo test unitario, considerándose como valores altos los más rápidos con mínimas diferencias y lentos los que más tiempo necesitan.

Tipo de valor: Alto / Medio / Bajo.

### 3.4.2 Criterio D.2: Velocidad pruebas funcionales

Nombre del criterio: Velocidad de la ejecución al realizar las pruebas funcionales

Descripción: Velocidad que se demora el entorno de testeo automatizado para PHP en ejecutar las pruebas funcionales. Este valor se determina en función del tiempo que tarda el resto de software similares a este en realizar el mismo test funcional, considerándose como valores altos los más rápidos con mínimas diferencias y lentos los que más tiempo necesitan.

Tipo de valor: Alto / Medio / Bajo.

### 3.4.3 Criterio D.3: Velocidad pruebas de aceptación

Nombre del criterio: Velocidad de la ejecución al realizar las pruebas de aceptación

Descripción: Velocidad que se demora el entorno de testeo automatizado para PHP en ejecutar las pruebas de aceptación. Este valor se determina en función del tiempo que tarda el resto de software similares a este en realizar el mismo test de aceptación, considerándose como valores altos los más rápidos con mínimas diferencias y lentos los que más tiempo necesitan.

Tipo de valor: Alto / Medio / Bajo.

### 3.4.4 Criterio D.4: Velocidad pruebas de integración

Nombre del criterio: Velocidad de la ejecución al realizar las pruebas de integración

Descripción: Velocidad que se demora el entorno de testeo automatizado para PHP en ejecutar las pruebas de integración. Este valor se determina en función del tiempo que tarda el resto de software similares a este en realizar el mismo test unitario, considerándose como valores altos los más rápidos con mínimas diferencias y lentos los que más tiempo necesitan.

Tipo de valor: Alto / Medio / Bajo.

### 3.4.5 Criterio D.5: Velocidad pruebas de regresión

Nombre del criterio: Velocidad de la ejecución al realizar las pruebas de regresión

Descripción: Velocidad que se demora el entorno de testeo automatizado para PHP en ejecutar las pruebas de regresión. Este valor se determina en función del tiempo que tarda el resto de software similares a este en realizar el mismo test unitario, considerándose como valores altos los más rápidos con mínimas diferencias y lentos los que más tiempo necesitan.

Tipo de valor: Alto / Medio / Bajo.

### 3.4.6 Criterio D.6: Velocidad pruebas de estrés

Nombre del criterio: Velocidad de la ejecución al realizar las pruebas de estrés

Descripción: Velocidad que se demora el entorno de testeo automatizado para PHP en ejecutar las pruebas de estrés. Este valor se determina en función del tiempo que tarda el resto de software similares a este en realizar el mismo test unitario, considerándose como valores altos los más rápidos con mínimas diferencias y lentos los que más tiempo necesitan.

Tipo de valor: Alto / Medio / Bajo.

### 3.4.7 Criterio D.7: Velocidad pruebas de rendimiento

Nombre del criterio: Velocidad de la ejecución al realizar las pruebas de rendimiento

Descripción: Velocidad que se demora el entorno de testeo automatizado para PHP en ejecutar las pruebas de rendimiento. Este valor se determina en función del tiempo que tarda el resto de software similares a este en realizar el mismo test unitario, considerándose como valores altos los más rápidos con mínimas diferencias y lentos los que más tiempo necesitan.

Tipo de valor: Alto / Medio / Bajo.

# 4. Evaluación de los criterios por tecnología

## 4.1 Evaluación de los criterios para la tecnología PHPUnit

|  |  |
| --- | --- |
| 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 | BSD |
| Criterio A.5: Documentación disponible | Alta |
| Criterio B.1: Dificultad para la puesta en marcha | Baja |
| Criterio B.2: Prueba de mutación con librería Infection | Si |
| Criterio B.3: Prueba de mutación con librería Humbug | Si |
| Criterio B.4: Análisis de cobertura de código con XDebug | Si |
| Criterio B.5: Velocidad del análisis de cobertura de código | Baja |
| Criterio B.6: Complejidad para analizar los resultados | Media |
| Criterio C.1: Pruebas unitarias | Si |
| Criterio C.2: Pruebas funcionales | Si |
| Criterio C.3: Pruebas de aceptación | Si |
| Criterio C.4: Pruebas de integración | Si |
| Criterio C.5: Pruebas de regresión | Si |
| Criterio C.6: Pruebas de estrés | Si |
| Criterio C.7: Pruebas de rendimiento | Si |
| Criterio D.1: Velocidad pruebas unitarias | Alta |
| Criterio D.2: Velocidad pruebas funcionales | Alta |
| Criterio D.3: Velocidad pruebas de aceptación | Media |
| Criterio D.4: Velocidad pruebas de integración | Baja |
| Criterio D.5: Velocidad pruebas de regresión | Baja |
| Criterio D.6: Velocidad pruebas de estrés | Media |
| Criterio D.7: Velocidad pruebas de rendimiento | Alta |

## 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: Pruebas unitarias | Si |
| Criterio C.2: Pruebas funcionales | Si |
| Criterio C.3: Pruebas de aceptación | Si |
| Criterio C.4: Pruebas de integración | Si |
| Criterio C.5: Pruebas de regresión | Si |
| Criterio C.6: Pruebas de estrés | Si |
| Criterio C.7: Pruebas de rendimiento | Si |
| Criterio D.1: Velocidad pruebas unitarias | Alta |
| Criterio D.2: Velocidad pruebas funcionales | Alta |
| Criterio D.3: Velocidad pruebas de aceptación | Baja |
| Criterio D.4: Velocidad pruebas de integración | Media |
| Criterio D.5: Velocidad pruebas de regresión | Baja |
| Criterio D.6: Velocidad pruebas de estrés | Media |
| Criterio D.7: Velocidad pruebas de rendimiento | Alta |

# 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 | Si | Si | Ambos entornos permiten configurar mediante una plantilla genérica cualquier tipo de pruebas. |
| C.2 | Si | Si | Ambos entornos permiten configurar mediante una plantilla genérica cualquier tipo de pruebas. |
| C.3 | Si | Si | Ambos entornos permiten configurar mediante una plantilla genérica cualquier tipo de pruebas. |
| C.4 | Si | Si | Ambos entornos permiten configurar mediante una plantilla genérica cualquier tipo de pruebas. |
| C.5 | Si | Si | Ambos entornos permiten configurar mediante una plantilla genérica cualquier tipo de pruebas. |
| C.6 | Si | Si | Ambos entornos permiten configurar mediante una plantilla genérica cualquier tipo de pruebas. |
| C.7 | Si | Si | Ambos entornos permiten configurar mediante una plantilla genérica cualquier tipo de pruebas. |
| D.1 | Alta | Alta | Al ser pruebas unitarias (generales) se realizan rápidamente, no se obtienen diferencias notables. |
| D.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. |
| D.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. |
| D.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. |
| D.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. |
| D.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. |
| D.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. |

# 6. Recomendaciones

Basándonos en los criterios de valoración de cada tecnología planteamos las siguientes posibles situaciones:

## 6.1 Situación 1

### 6.1.1 Descripción de la situación

La empresa Knuto desarrolla aplicaciones de Gestión Medioambiental en su propio departamento de desarrollo. La complejidad de estas aplicaciones es bastante alta, incluso incorporan tecnología GIS. La empresa desea conocer si los programas desarrollados son óptimos y fiables para poder ser lanzados posteriormente al mercado. Los resultados obtenidos los requieren también desde el departamento legal para justificar la adecuación de la aplicación a la normativa establecida. El equipo que se va a encargar de la automatización del testing tiene amplios conocimientos y experiencia. Para ello, plantean utilizar una de las dos herramientas que ha propuesto el jefe de departamento de desarrollo, que son Codeception y PHPUnit.

### 6.1.2 Recomendación de tecnología a utilizar

En esta situación el departamento se decanta por elegir Codeception, las ventajas que se han encontrado y por las que se decantan por esta tecnología las mostramos a continuación en una tabla:

|  |  |  |
| --- | --- | --- |
| Criterios relevantes para la decisión | Ventajas PHPUnit | Ventajas Codeception |
| Velocidad de ejecución |  | Al tratarse de aplicaciones de una gran cantidad de líneas, Codeception tarda menos tiempo en realizarlas. |
| Dificultad para la puesta en marcha |  | La dificultad de la configuración es más compleja en Codeception, pero no supone una desventaja, ya que sus trabajadores son expertos en testing automático. |
| Complejidad para analizar los resultados |  | En este caso es más sencillo analizar los resultados. Dado que estos son más intuitivos, será más rápido analizar las conclusiones. |

## 6.2 Situación 2

### 6.2.1 Descripción de la situación

Una pequeña empresa de tecnología llamada HackingLife tiene la necesidad de llevar a cabo el desarrollo de una pequeña aplicación en PHP para gestionar el registro de los usuarios a través de su página web. Dicha aplicación debe interactuar con el front-end y back-end de la web para que puedan ser almacenados de manera segura en la base de datos. El equipo que va a llevar a cabo el desarrollo será un equipo de jóvenes programadores debido a la sencillez y baja complejidad que tiene la aplicación. El departamento de software y tecnología propone el uso de alguno de estos dos entornos de trabajo: PHPUnit y Codeception.

### 6.2.2 Recomendación de tecnología a utilizar

Para esta situación se ha decidido utilizar la tecnología PHPUnit ya que contamos con una serie de ventajas frente a Codeception que son desglosadas en el cuadro siguiente:

|  |  |  |
| --- | --- | --- |
| Criterios relevantes para la decisión | Ventajas PHPUnit | Ventajas Codeception |
| Velocidad de ejecución | Tratándose de un código sencillo y de baja complejidad ya que no cuenta con gran número de líneas de código, la velocidad entre ambas herramientas es similar. |  |
| Dificultad para la puesta en marcha | La dificultad de la configuración es más sencilla en PHPUnit, por lo que dado que se cuenta con un grupo joven de programadores que no cuentan con una gran experiencia es más recomendable su uso. |  |
| Complejidad para analizar los resultados |  | Teniendo en cuenta que son programadores que no cuentan con experiencia y no están acostumbrados a analizar resultados, la herramienta Codeception cuenta con apariencias más intuitivas y es más rápido a la hora de analizar las conclusiones. |