



PUESTA EN PRODUCCIÓN

Tarea 1

ESTHER CARRILLO GÁLVEZ

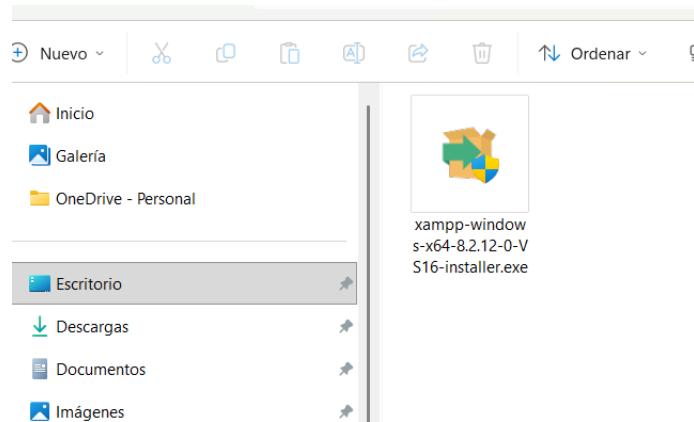
Contenido

| | |
|--|----|
| Apartado 1: Instalación de herramientas utilizadas | 3 |
| 1.1. Instalación de entorno XAMPP | 3 |
| 1.2. Instalación Visual Studio Code | 4 |
| 1.3. Instalación Composer para PHPUnit | 7 |
| 1.4. Instalación PHPUnit | 11 |
| Apartado 2: Rellenar tabla con el código del programa de pruebas | 13 |
| Apartado 4: Creación del programa de pruebas unitarias | 16 |
| Apartado 5: Verificación de software | 17 |
| Apartado 6: Responder a la pregunta..... | 20 |
| Bibliografía..... | 21 |

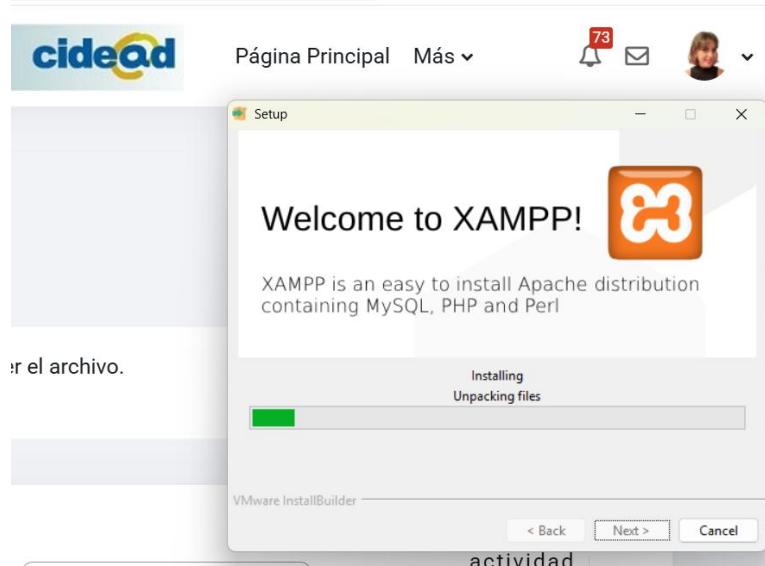
Apartado 1: Instalación de herramientas utilizadas

1.1. Instalación de entorno XAMPP

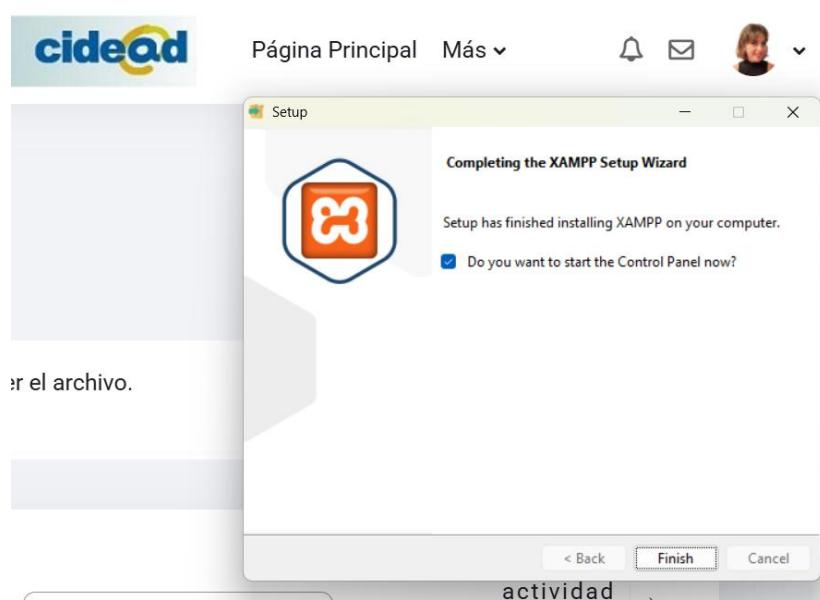
El primer paso del proceso es instalar el entorno XAMPP desde el sitio oficial <https://www.apachefriends.org/es/index.html>. En cada uno de los pasos anteriores hasta el último, se ha seleccionado la configuración por defecto pulsando el botón *Next*, así hasta llegar a *Finish*.



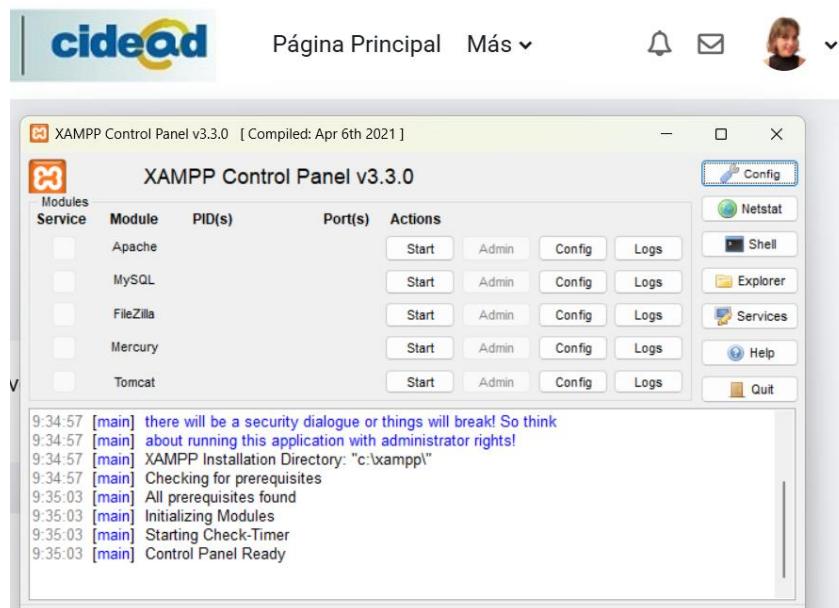
Instalación en progreso. Este es el penúltimo paso antes de la finalización.



Se deja seleccionado el botón de ejecutar el Panel de control y se finaliza.

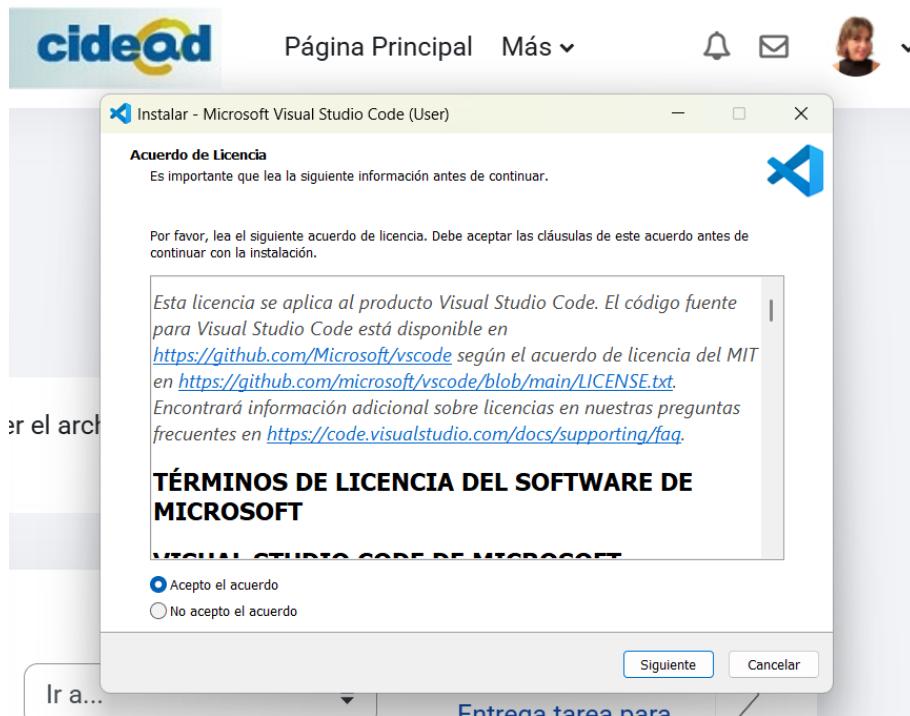


Una vez finalizado, se abrirá este panel que es la herramienta central para gestionar tu entorno de servidor local. La columna Module muestra los servicios que XAMPP instala. Es más importante y el que se va a usar para esta práctica es el primero.

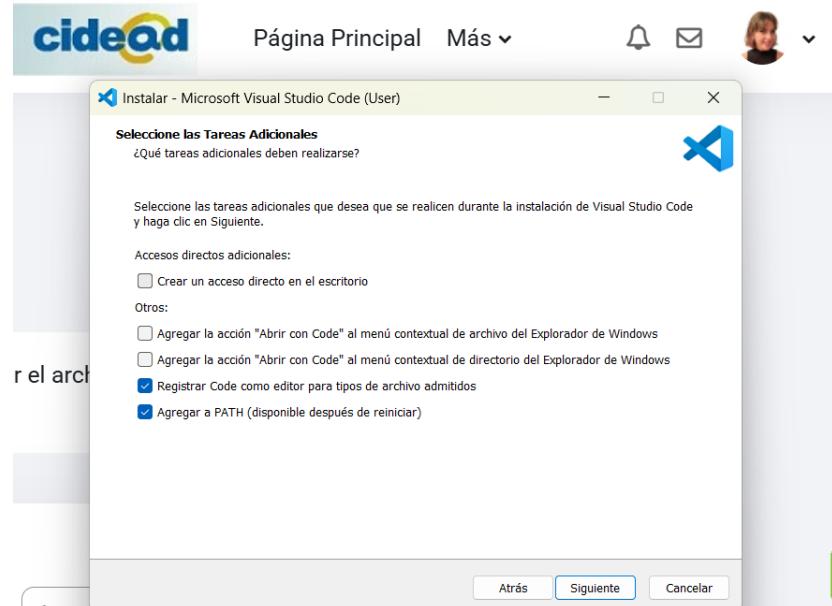


1.2. Instalación Visual Studio Code

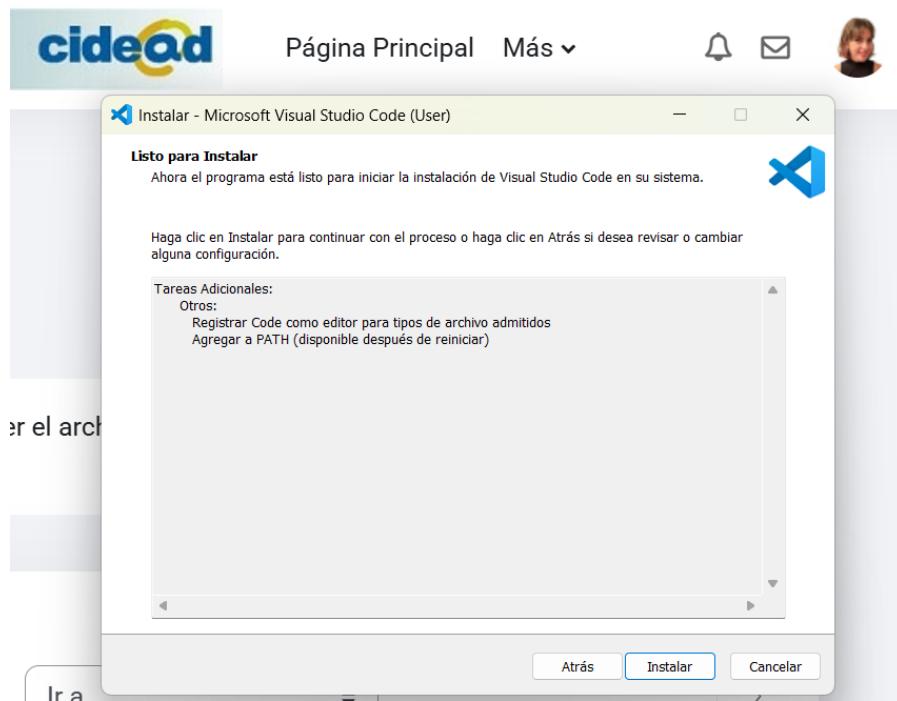
A continuación, se instala Visual Studio Code desde [Visual Studio Code - The open source AI code editor](#). En primer lugar, se aceptan los términos de licencia del software.



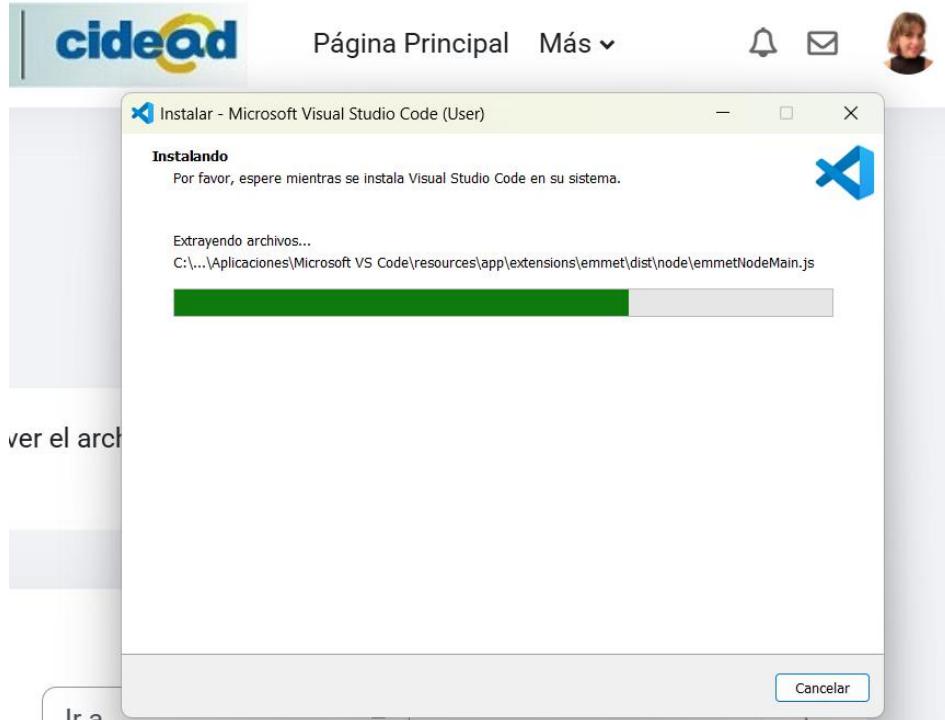
Se deben seleccionar las tareas adicionales. En este caso, se dejan las que vienen por defecto.



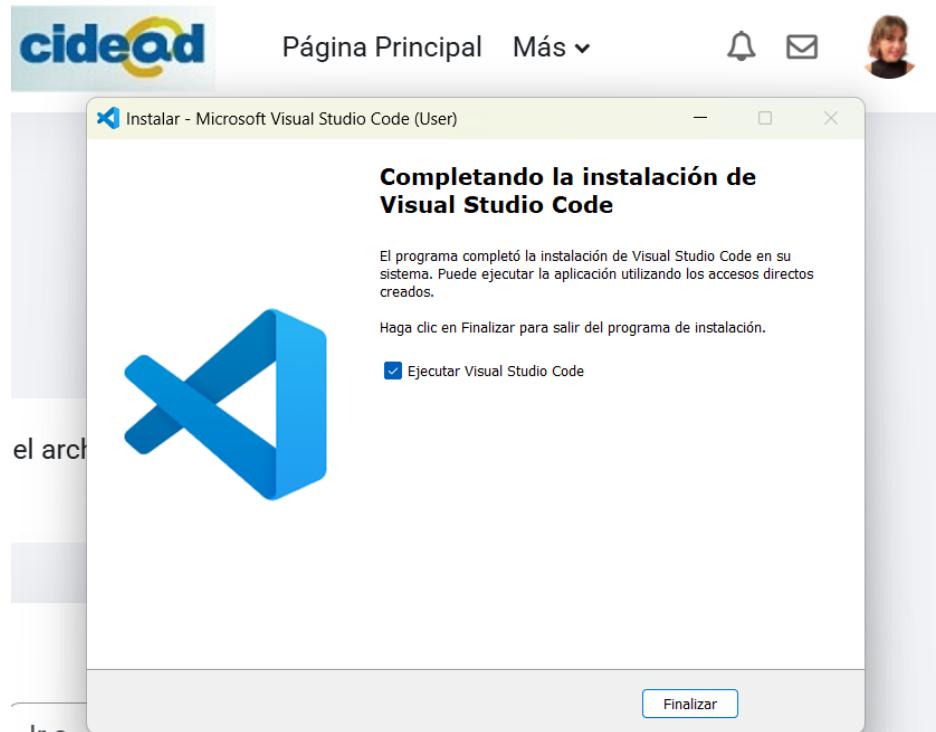
En el paso “Listo para instalar”, se muestra el resumen de la configuración.



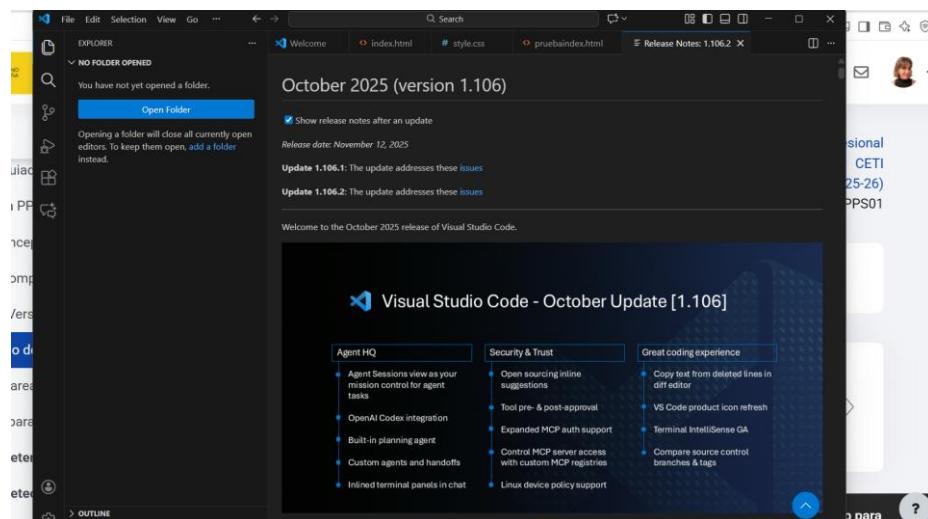
Instalación en progreso. Este es el penúltimo paso antes de la finalización.



Estado de la instalación completado. Se deja seleccionado “ejecutar Visual Studio Code”.



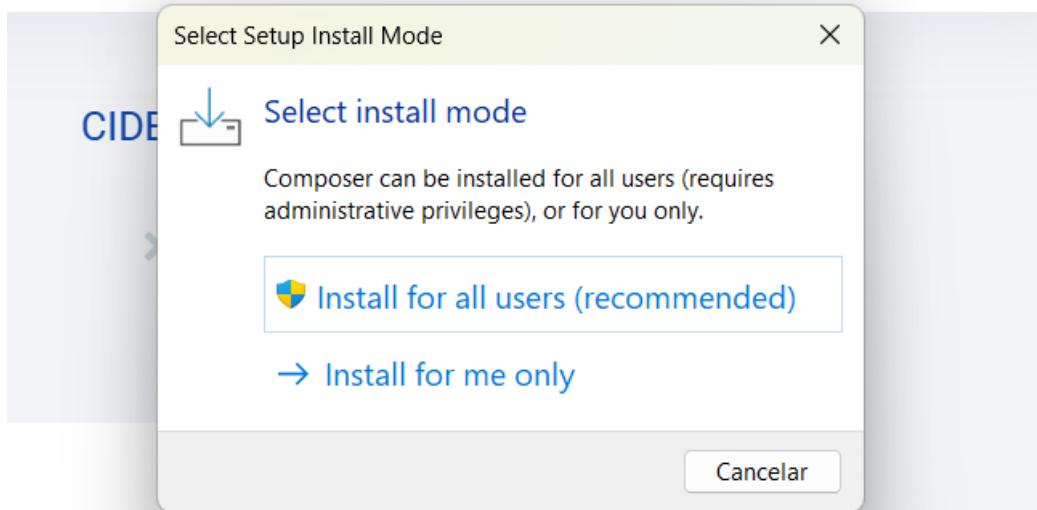
Finalmente, se ejecutará Visual Studio.



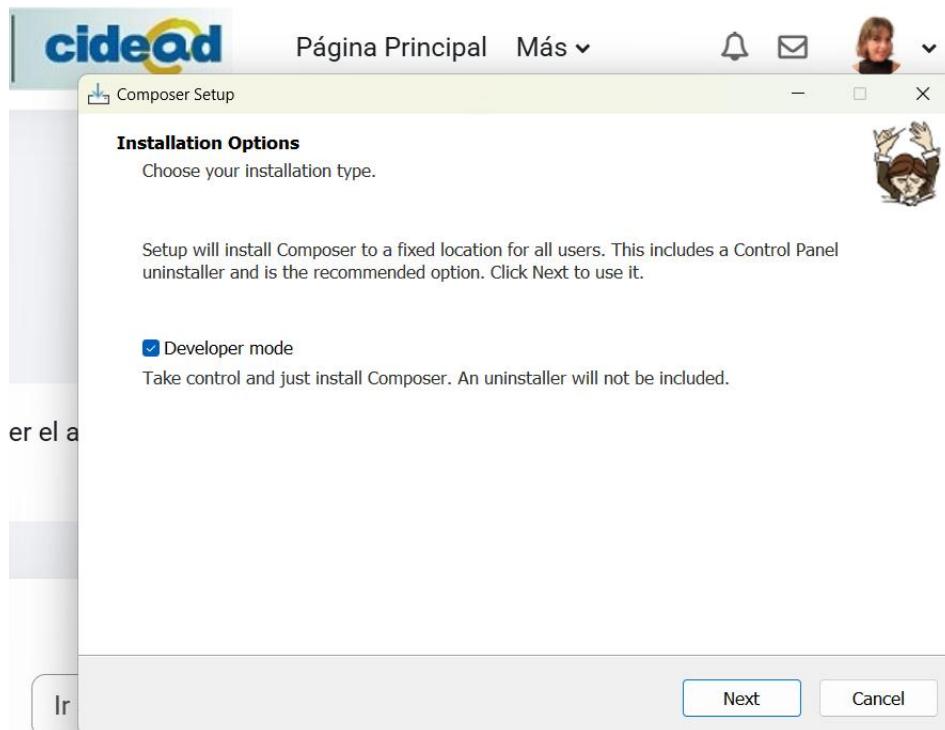
1.3. Instalación Composer para PHPUnit

Se debe instalar *Composer* antes que *PHPUnit* porque *PHPUnit* es una dependencia (o paquete de software) que se recomienda instalar y gestionar a través de *Composer*.

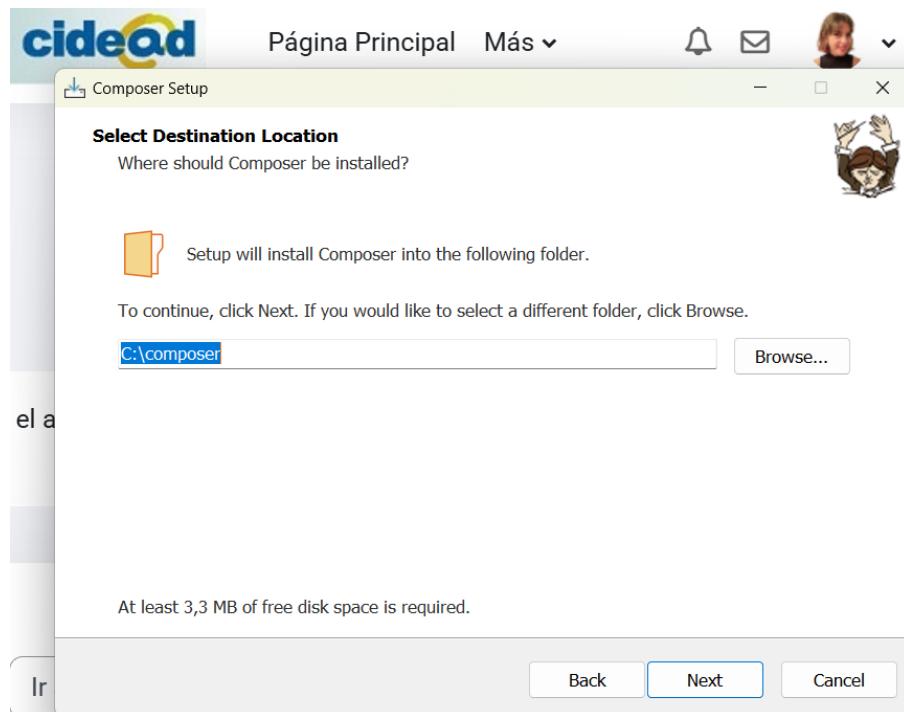
En el primer paso se debe seleccionar el modo de instalación, en este caso se instala para todos los usuarios.



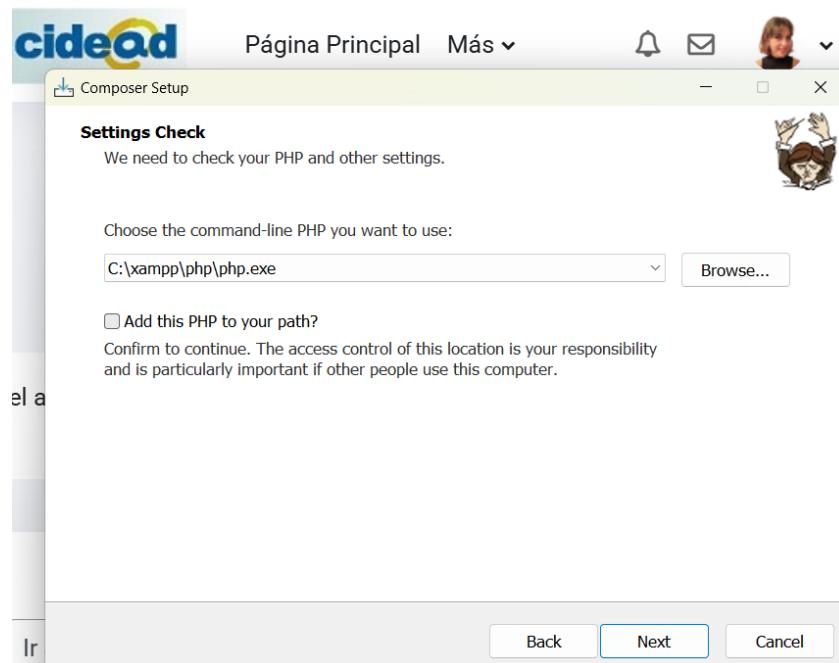
En las opciones de instalación, se marca el modo de desarrollador/a para una instalación más directa y menos "interferida".



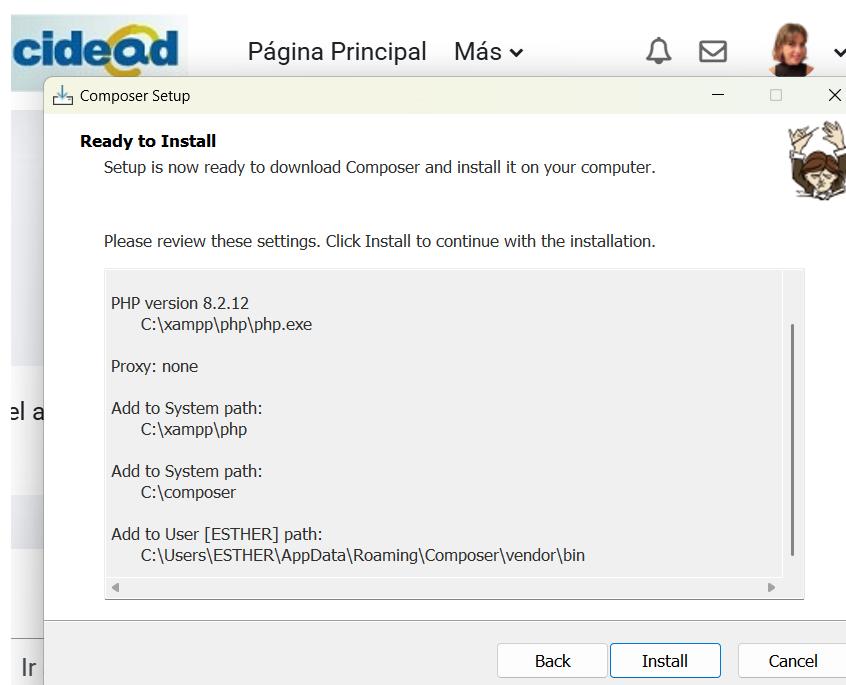
En el siguiente paso se deja por defecto el directorio para la instalación.



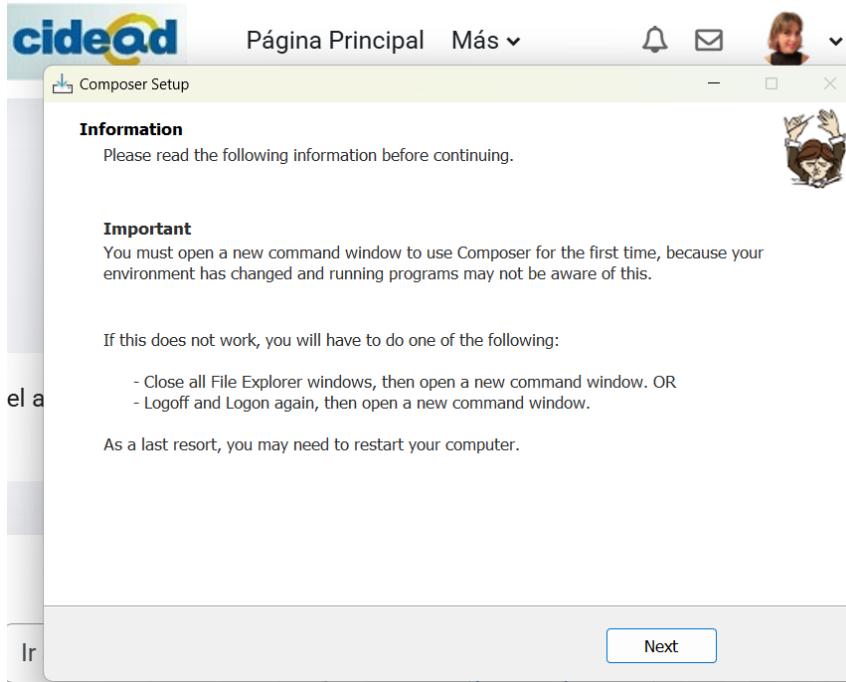
Se deja seleccionado la ruta donde se encuentra Composer.



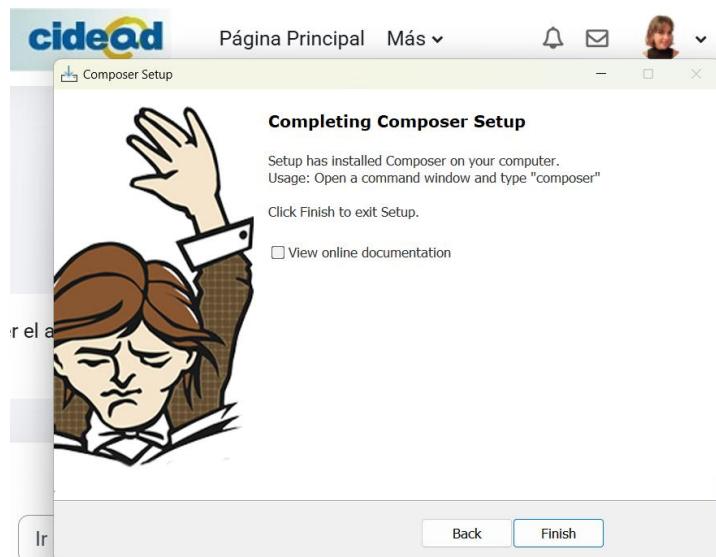
Este paso es el resumen de la configuración actual para el intérprete de PHPUnit. Pulsar el botón *install* para proceder a la instalación.



Este mensaje es un aviso sobre cómo empezar a usar Composer después de la instalación.



Esta simpática ilustración, indica que la instalación ha sido completada.



1.4. Instalación PHPUnit

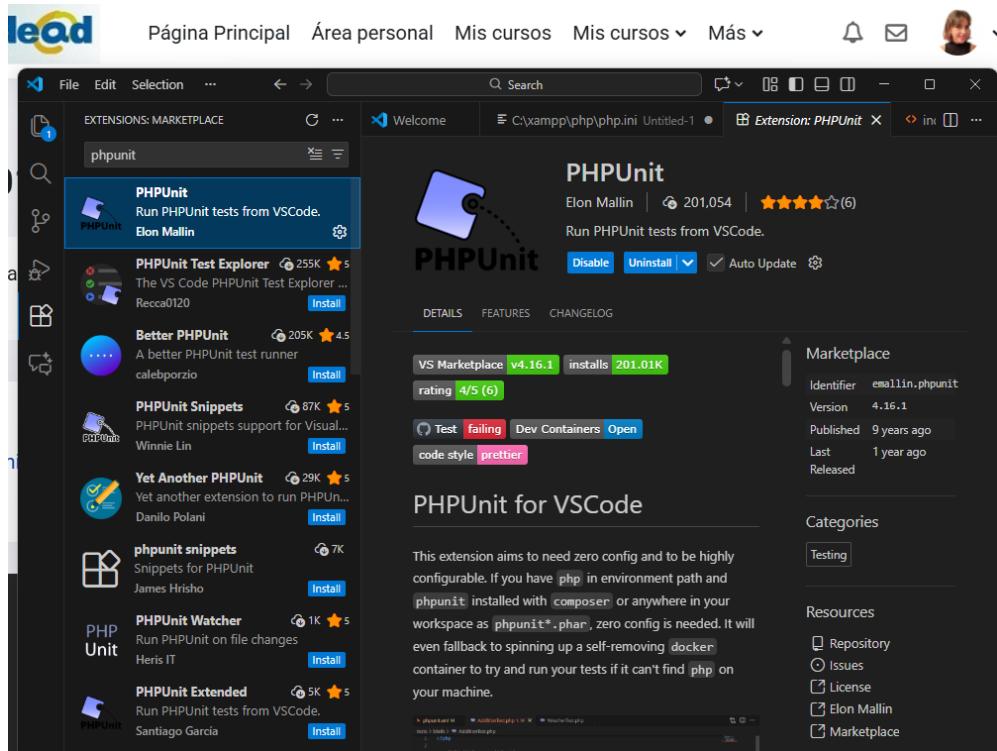
Se abre la línea de comandos y se ejecuta el comando `composer require --dev phpunit/phpunit` para instalar PHPUnit es el directorio raíz del proyecto PHP.

Composer advierte que no usaría la versión más nueva de PHPUnit, pero logra instalar correctamente una versión anterior que es compatible con la versión de PHP que está disponible en XAMPP.

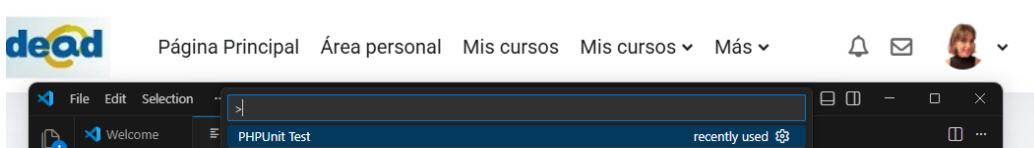
A screenshot of a terminal window titled "Símbolo del sistema". The command entered is `C:\xampp>composer require --dev phpunit/phpunit`. The terminal output shows the following message: "Cannot use phpunit/phpunit's latest version 12.4.4 as it requires php >=8.3 which is not satisfied by your platform." Below this, the command runs successfully, showing the download and installation of various PHPUnit dependencies. The output ends with "Nothing to modify in lock file" and "Writing lock file".

```
Símbolo del sistema x + | v
C:\xampp>composer require --dev phpunit/phpunit
Cannot use phpunit/phpunit's latest version 12.4.4 as it requires php >=8.3 which is not satisfied by your platform.
./composer.json has been updated
Running composer update phpunit/phpunit
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 27 installs, 0 updates, 0 removals
- Downloading staabm/side-effects-detector (1.0.5)
- Downloading sebastian/version (5.0.2)
- Downloading sebastian/type (5.1.3)
- Downloading sebastian/recursion-context (6.0.3)
- Downloading sebastian/object-reflector (4.0.1)
- Downloading sebastian/object-enumerator (6.0.1)
- Downloading sebastian/global-state (7.0.2)
- Downloading sebastian/exporter (6.3.2)
- Downloading sebastian/environment (7.2.1)
- Downloading sebastian/diff (6.0.2)
- Downloading sebastian/comparator (6.3.2)
- Downloading sebastian/code-unit (3.0.3)
- Downloading sebastian/cli-parser (3.0.2)
- Downloading phpunit/php-timer (7.0.1)
```

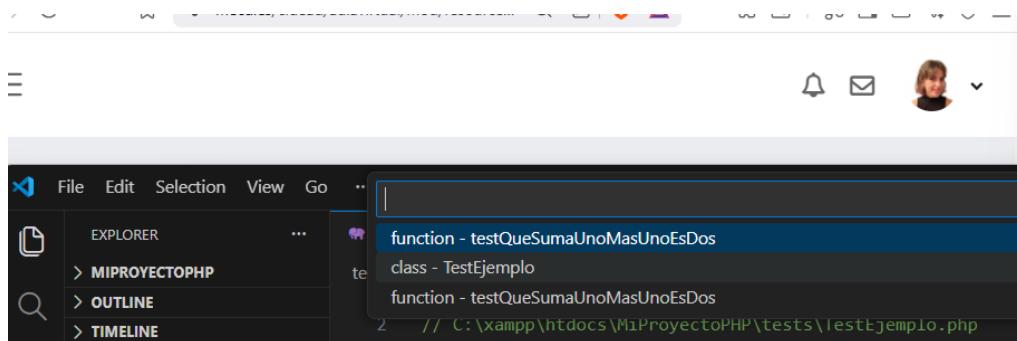
El siguiente paso es instalar la extensión de PHPUnit en Visual Studio Code.



A continuación, se configura el entorno para poder trabajar con PHP y poder realizar las pruebas unitarias en el mismo entorno. Se pulsa la combinación de teclas *Ctrl + Shift + P* para abrir una barra de búsqueda en la parte superior del editor.



Se escoge una clase o función de las que se despliegan.



Apartado 2: Rellenar tabla con el código del programa de pruebas

| PREGUNTA | RESPUESTA |
|--|---|
| Indica el número de las líneas donde hay un comentario | 3: // Importa el archivo con la función a probar 5: // utiliza la librería con PHPUnit 7: // crea una clase que hereda de TestCase 9: // crea un método para realizar las pruebas |
| Indica el número de las líneas donde hay comandos de pre-procesamiento | 2: require 'ejemplo.php'; 4: use PHPUnit\Framework\TestCase; |
| Indica el nombre de las variables utilizadas | 10: \$n 11: \$resultado Esperado 12: \$resultado obtenido |
| Indica el nombre de los métodos/funciones utilizadas | 12: obtenerPrimerosNumeros() 13: assertEquals() |
| Indica qué hace la línea 13 | 13: \$this->assertEquals(\$resultado Esperado, \$resultado obtenido); Esta línea realiza una aserción. Compara el valor esperado (\$resultado Esperado) con el valor obtenido (\$resultado obtenido) \$this indica que el método pertenece a la clase de prueba EjemploTest. |

Apartado 3: Crear un programa PHP que genere la secuencia de Fibonacci.

Código php que lanza la sucesión de Fibonacci.

The screenshot shows a Microsoft Visual Studio Code interface. The title bar reads "Aula Virtual Centro Integrado de Enseñanzas Regladas a Distancia" and "cidead". The code editor window has "fibo.php" selected. The code itself is as follows:

```
ejemplotest > ● fibo.php
1 <?php
2 // Nombre: Esther Carrillo Gálvez
3 /**
4 * Tarea PPS01,
5 * Generar secuencia Fibonacci en PHP
6 */
7
8 function fibonacci($n) {
9     // Si el valor indicado es menor o igual a 1,
10    // devolvemos un array con el parámetro
11    if ($n <= 1) {
12        return [$n];
13    }
14
15    // Inicializamos dos variables para almacenar el comienzo de la secuencia
16    $a = 0; // Primer valor
17    $b = 1; // Segundo valor
18    $secuencia = [$a, $b];
19
20    try {
21        // Generamos nuevos valores con un bucle for
22        for ($i = 2; $i < $n; $i++) {
23            // Calculamos el siguiente valor
24            $c = $a + $b;
25
26            // Asignamos los nuevos valores
27            $a = $b;
28            $b = $c;
29
30            // Añadimos nuevo valor a la secuencia
31            // Nota: El comando array_push no se ve en el editor,
32            // pero se transcribe la línea completa que aparece en la imagen inferior
33            array_push($secuencia, $c);
34        }
35
36    } catch (Exception $e) {
37        // Si el valor es erróneo lo devolvemos vacío
38    }
39
40
41    // Devolvemos la secuencia
42    return $secuencia;
43 }
44
45 // Indicamos el número de valores a generar
46 $n = 10;
47 $secuencia = fibonacci($n);
48 // Mostramos los valores
49 if ($n >= 0) {
50     echo implode(",", $secuencia);
51 } else {
52     echo "El valor de n debe ser mayor o igual a 1";
53 }
54 ?>
```

The terminal below shows the command line and its output:

```
PS C:\xampp\htdocs\MiProyectoPHP> cd ejemplotest
PS C:\xampp\htdocs\MiProyectoPHP\ejemplotest> php fibo.php
0,1,1,2,3,5,8,13,21,34
PS C:\xampp\htdocs\MiProyectoPHP\ejemplotest> []
```

This screenshot shows the same Microsoft Visual Studio Code interface as the previous one, but with a different version of the code in the editor. The code has been modified, particularly in the main loop and the final output section.

```
ejemplotest > ● fibo.php
8 function fibonacci($n) {
9     try {
10
11         } catch (Exception $e) {
12             // Si el valor es erróneo lo devolvemos vacío
13             return [];
14         }
15
16         // Devolvemos la secuencia
17         return $secuencia;
18     }
19
20     // Indicamos el número de valores a generar
21     $n = 10;
22     $secuencia = fibonacci($n);
23     // Mostramos los valores
24     if ($n >= 0) {
25         echo implode(",", $secuencia);
26     } else {
27         echo "El valor de n debe ser mayor o igual a 1";
28     }
29
30 ?>
```

The terminal output is identical to the previous screenshot:

```
PS C:\xampp\htdocs\MiProyectoPHP> cd ejemplotest
PS C:\xampp\htdocs\MiProyectoPHP\ejemplotest> php fibo.php
0,1,1,2,3,5,8,13,21,34
PS C:\xampp\htdocs\MiProyectoPHP\ejemplotest> []
```

Código php completo.

```
<?php
// Nombre: Esther Carrillo Gálvez
/**
 * Tarea PPS01.
 * Generar secuencia Fibonacci en PHP
 */

function fibonacci($n) {
    // Si el valor indicado es menor o igual a 1,
    // devolvemos un array con el parámetro
    if ($n <= 1) {
        return [$n];
    }

    // Inicializamos dos variables para almacenar el comienzo de la secuencia
    $a = 0; // Primer valor
    $b = 1; // Segundo valor
    $secuencia = [$a, $b];

    try {
        // Generamos nuevos valores con un bucle for
        for ($i = 2; $i < $n; $i++) {
            // Calculamos el siguiente valor
            $c = $a + $b;

            // Asignamos los nuevos valores
            $a = $b;
            $b = $c;

            // Añadimos nuevo valor a la secuencia
            // Nota: El comando array_push no se ve en el editor,
            // pero se transcribe la línea completa que aparece en la imagen inferior
            array_push($secuencia, $c);
        }
    } catch (Exception $e) {
        // Si el valor es erróneo lo devolvemos vacío
        return [];
    }

    // Devolvemos la secuencia
    return $secuencia;
}

// Indicamos el número de valores a generar
$n = 10;
$secuencia = fibonacci($n);
// Mostramos los valores
if ($n >= 0) {
    echo implode(",", $secuencia);
} else {
    echo "El valor de n debe ser mayor o igual a 1";
}
?>
```

Apartado 4: Creación del programa de pruebas unitarias

Código PHP para realizar la prueba unitaria sobre el fichero *fibo.php*.



Enunciado de la tarea para PPS01

CIDEAD 2025/26 > Formación Profesional > Cursos de Especialización > CETI > Puesta en Producción Segura (25-26) > Enunciado de la tarea para PPS01

```
<?php
//Importar el archivo con la función fibo
require 'fibo.php';

//Utilizamos la librería PHPUnit
use PHPUnit\Framework\TestCase;

//Creamos la clase para los test
class testFibo extends TestCase
{
    //se crea el método para las pruebas
    public function testFibonacci5(){
        $secuencia = fibonacci(5);

        $this->assertEquals([0,1,1,2,3], $secuencia, "la secuencia no coincide");
    }

    public function testFibonacci()
    {
        //Crear los parámetros para $posición y $valor
        $posición = 4;
        $valor = 3;
        //Obtener la secuencia
        $secuencia = fibonacci(5);
        //extraemos la posición
        $valor_secuencia = $secuencia[$posición];

        $this->assertEquals($valor, $valor_secuencia, "El valor {$valor_secuencia} en la posición $posición de la secuencia no coincide");
    }
}
```

Código php completo.

```
<?php
//Importar el archivo con la función probar
require 'fibo.php';

//Utilizamos la librería PHPUnit
use PHPUnit\Framework\TestCase;

//Creamos la clase para los test
class testFibo extends TestCase
{
    //se crea el método para las pruebas
    public function testFibonacci(){
        $secuencia = fibonacci(5);

        $this->assertEquals([0,1,1,2,3], $secuencia, "la secuencia no coincide");
    }

    public function testFibonacci()
    {
        //Crear los parámetros para $posición y $valor

        $posición = 4;
        $valor = 3;
        //Obtener la secuencia
        $secuencia = fibonacci(5);
        //extraemos la posición
        $valor_secuencia = $secuencia[$posición];

        $this->assertEquals($valor, $valor_secuencia, "El valor {$valor_secuencia} en la posición $posición de la secuencia no coincide");
    }
}
```

Apartado 5: Verificación de software

Se ejecuta el comando `vendor\bin\phpunit testFibo.php` para correr todas las pruebas contenidas en el archivo. Es exitoso.

The screenshot shows a web browser window with the following details:

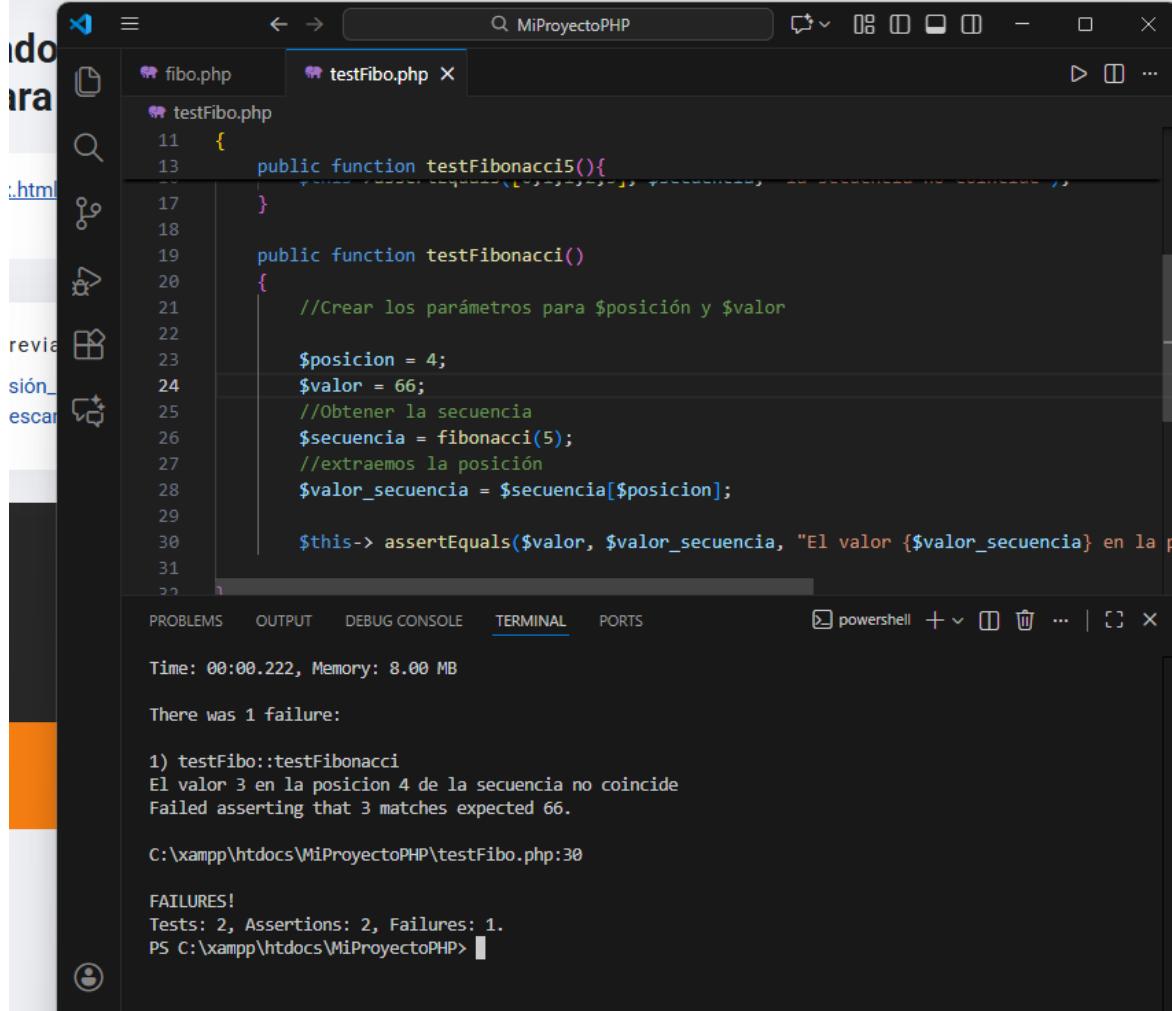
- Top Bar:** Aula Virtual, Centro Integrado de Enseñanzas Regladas a Distancia, cidead logo, Página Principal, Más, notifications, and user profile.
- Code Editor:** An IDE-like interface with tabs for "fibo.php" and "testFibo.php". The "testFibo.php" tab is active, displaying PHP code for a Fibonacci sequence test. The code includes assertions for positions 4 and 5 of the sequence.
- Terminal:** Shows the command "vendor\bin\phpunit testFibo.php" being run in a powershell terminal. The output indicates the test passed with 2 tests, 2 assertions, and 0 failures.
- Status Bar:** Shows runtime information (PHP 8.2.12), time (00:00.086), memory usage (8.00 MB), and file statistics (OVR, Ln 33, Col 2, Spaces: 4, UTF-8, CRLF).

Comprobación de la capacidad del programa de detectar errores mediante el cambio del valor 3 por el valor 66 en línea 24.

- El mensaje final del test: **FAILURES! Tests: 1, Assertions: 1, Failures: 1.**

La aparición del mensaje de error es la comprobación de que el test detecta errores.

Aula Virtual | cidead | Página Principal | Más | 



The screenshot shows a code editor interface with the following details:

- Title Bar:** MiProyectoPHP
- File List:** fibo.php, testFibo.php (active)
- Code Editor:** Contains PHP test code for a Fibonacci sequence function.

```
11  {
13      public function testFibonacci5(){
17      }
18
19      public function testFibonacci()
20      {
21          //Crear los parámetros para $posición y $valor
22
23          $posición = 4;
24          $valor = 66;
25          //Obtener la secuencia
26          $secuencia = fibonacci(5);
27          //extraemos la posición
28          $valor_secuencia = $secuencia[$posición];
29
30          $this->assertEquals($valor, $valor_secuencia, "El valor {$valor_secuencia} en la p
31
32 }
```
- Terminal Tab:** TERMINAL (selected)
- Terminal Output:**

```
Time: 00:00.222, Memory: 8.00 MB

There was 1 failure:

1) testFibo::testFibonacci
El valor 3 en la posición 4 de la secuencia no coincide
Failed asserting that 3 matches expected 66.

C:\xampp\htdocs\MiProyectoPHP\testFibo.php:30

FAILURES!
Tests: 2, Assertions: 2, Failures: 1.
PS C:\xampp\htdocs\MiProyectoPHP>
```

Apartado 6: Responder a la pregunta

Esta vulnerabilidad se presenta cuando las credenciales están escritas en el código fuente de la aplicación. Lo que ocurre en el código de ejemplo es que un atacante podría vulnerar el sistema si tuviera el código fuente, ya que las líneas 3, 4, 5 y 6 revelan información muy sensible. En estas líneas se leen cuatro variables relacionadas con el nombre del servidor (`$servername = "db1"`), el nombre de la base de datos (`$database = "databasename"`), y las credenciales de usuario (`$username = "username"; $password = "password"`).

```
<?php
require_once 'pdoconfig.php';
$servername = "db1";
$databasename = "databasename";
$username = "username";
$password = "password";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$database", $username, $password);
    echo "Conectado a $database de $servername con éxito.";
} catch (PDOException $pe) {
    die("No es posible la conexión a $database : " . $pe->getMessage());
}
?>
```

Para resolver el problema de credenciales fijas, la práctica recomendada es almacenar las credenciales en un gestor Secrets Manager como Azure Key Vault. Además de cargar las credenciales en un archivo de configuración local (.env, config.ini) fuera del directorio web (htdocs).

Bibliografía

Apache Friends. (s. f.). *XAMPP*. Recuperado el 1 de diciembre de 2025, de <https://www.apachefriends.org/es/index.html>

SonarSource. (s. f.). *Rule RSPEC-2068: Passwords should not be hardcoded*. Recuperado el 1 de diciembre de 2025, de <https://rules.sonarsource.com/typescript/tag/cwe/RSPEC-2068/>

Visual Studio Code. (s. f.). *Code editing. Redefined*. Recuperado el 1 de diciembre de 2025, de <https://code.visualstudio.com/>