# Test Driven Development

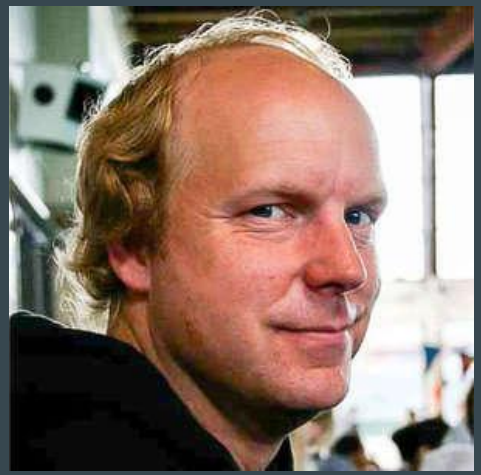

Magariños Leonardo-Bianco Manuel

# Que es Test Driven Development?

TDD es una práctica de programación, dividida en 3 partes:

- Escribir las pruebas
- Escribir el código fuente
- Refactorizar el código escrito

Fue creado por
Kent Beck



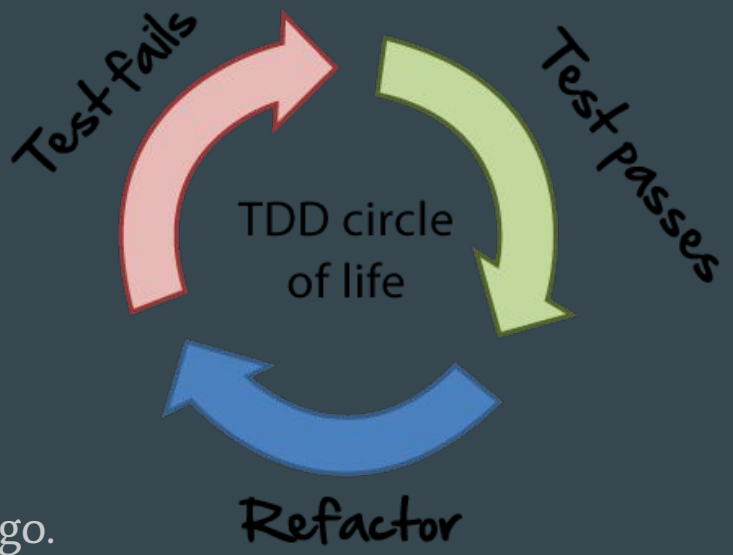
# Que se logra al utilizar TDD?

Con esta práctica se consigue entre otras cosas:

- Código más robusto.
- Más seguro.
- Más mantenible.
- Mayor rapidez en el desarrollo.

# Ciclo de desarrollo

1. Escribir un test para el requisito.
2. Probarlo y ver que falle.
3. Escribir el código para pasar el test.
4. Ejecutarlo y comprobar que funcione.
5. Refactorizar para evitar duplicación de código.



# Como se realizan las pruebas en PHP?

Las pruebas de los diversos requerimientos se realizan en Unit Test(Pruebas Unitarias).  En el caso de PHP, PHPUnit es uno de los entornos más utilizados para realizar pruebas unitarias, se puede encontrar en GitHub y ha sido creado por Sebastian Bergmann.

Como todos los frameworks de pruebas unitarias, PHPUnit utiliza assertions para verificar que el comportamiento de una unidad de código es el esperado.

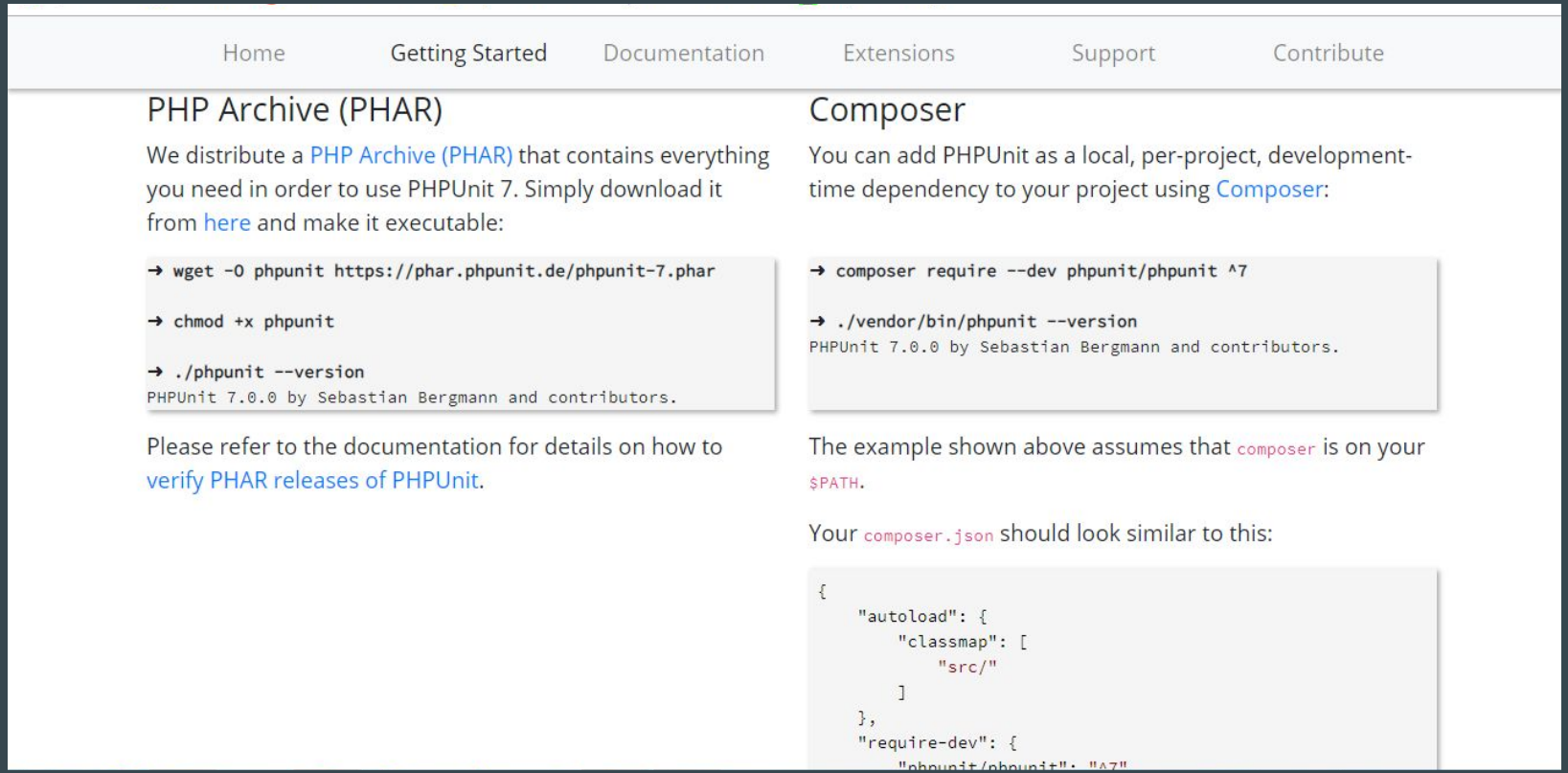
# PHPUnit

# Instalación


Archivo Phar


Composer

# Instalación



Home | Getting Started | Documentation | Extensions | Support | Contribute

## PHP Archive (PHAR)

We distribute a PHP Archive (PHAR) that contains everything you need in order to use PHPUnit 7. Simply download it from here and make it executable:

```
➜ wget -O phpunit https://phar.phpunit.de/phpunit-7.phar

➜ chmod +x phpunit

➜ ./phpunit --version
PHPUnit 7.0.0 by Sebastian Bergmann and contributors.
```

Please refer to the documentation for details on how to verify PHAR releases of PHPUnit.

## Composer

You can add PHPUnit as a local, per-project, development-time dependency to your project using Composer:

```
➜ composer require --dev phpunit/phpunit ^7

➜ ./vendor/bin/phpunit --version
PHPUnit 7.0.0 by Sebastian Bergmann and contributors.
```

The example shown above assumes that `composer` is on your `$PATH`.

Your `composer.json` should look similar to this:

```
{
    "autoload": {
        "classmap": [
            "src/"
        ]
    },
    "require-dev": {
        "phpunit/phpunit": "^7"
```

# Ejemplo

## Code

**src/Money.php**

```php
<?php
class Money
{
    private $amount;

    public function __construct($amount)
    {
        $this->amount = $amount;
    }

    public function getAmount()
    {
        return $this->amount;
    }

    public function negate()
    {
        return new Money(-1 * $this->amount);
    }

    // ...
}
```

## Test Code

**tests/MoneyTest.php**

```php
<?php
use PHPUnit\Framework\TestCase;

class MoneyTest extends TestCase
{
    // ...

    public function testCanBeNegated()
    {
        // Arrange
        $a = new Money(1);

        // Act
        $b = $a->negate();

        // Assert
        $this->assertEquals(-1, $b->getAmount());
    }

    // ...
}
```

# VENTAJAS

- Mayor calidad
- Diseño enfocado en las necesidades
- Mayor simplicidad en el diseño:
- El diseño se va adaptando al entendimiento del problema.
- Mayor productividad.
- Menos tiempo invertido en debugging de errores.

# DESVENTAJAS

- Es difícil implementar TDD en interfaces de usuario.
- Errores no identificados.
- Trabajar con bases de datos es complicado.
- Pronunciada curva de aprendizaje.

# Links útiles

- PHPUnit(Instalación) : https://phpunit.de/getting-started/phpunit-7.html
- Documentacion PHPUnit(en español) : https://phpunit.readthedocs.io/es/latest/

# PREGUNTAS?