Introducción a TDD

Workshop

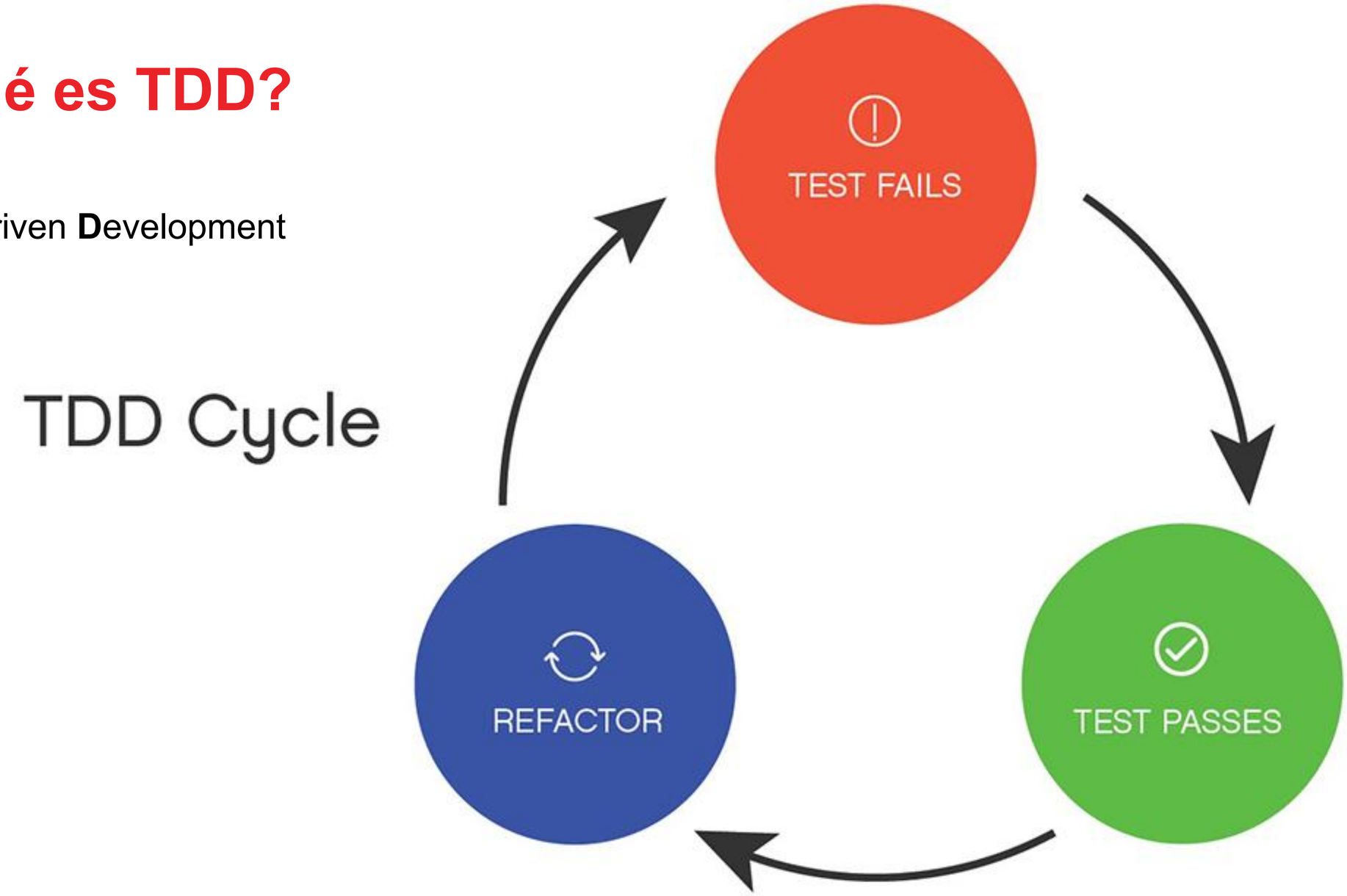
(+34) 986 120 430 | gradiant@gradiant.org | www.gradiant.org

¿Por qué TDD?

- **Entregar más valor más rápido**
- **Correlación entre Calidad interna y Productividad**
  - ¿Es fácil es encontrar el código responsable de una tarea?
  - ¿Es fácil es modificar ese código?
  - ¿Es fácil añadir es testear los cambios?
- **Foco**
  - TDD nos ayuda a enfocarnos en un solo problema pequeño en cada paso y alcanzar la solución final paso a paso
  - Evitar el overthinking
  - Evitar hacer cosas innecesarias
- **Sostenibilidad**
  - Reduce el coste de añadir cambios o nuevas funcionalidades

¿Qué es TDD?

Test **D**riven **D**evelopment



¿Qué es TDD?

- **RED** -> Foco en el problema, ¿que debemos resolver?
- **GREEN** -> Foco en la solución ¿cómo lo resolvemos?
- **REFACTOR** -> Foco en la calidad ¿ahora que funciona podemos mejorarlo?

¿Qué es TDD?

- Reglas básicas
  - Nunca añadir nueva funcionalidad sin un test fallando
  - Nunca refactorizar con un test fallando
  - Baby steps

Tipos de tests

- **Unitarios**: Prueban un comportamiento de forma aislada. En memoria sin involucrar I/O o comportamientos externos.
  - Definición de Unidad / No siempre es una clase o método
- **Integración**: Prueba la interacción entre distintos elementos. Típicamente involucra sistemas externos y I/O
- **End to End / Aceptación / Customer tests**: Prueban el sistema como un todo. Típicamente “ejercitan” el sistema como lo haría el cliente.

Tipos de tests

Tests Unitarios

- Características de un Unit test F.I.R.S.T
  - **F**ast
    - Se ejecutan rápido
  - **I**solated
    - Un caso de prueba nunca puede depender de otro caso de prueba
  - **R**epeatable
    - Determinista. El resultado debe ser siempre el mismo. El resultado no puede depender de factores externos.
  - **S**elf-validating
    - No pueden requerir una interpretación manual
  - **T**imely
    - Debe ser fácil/rápido añadir nuevos casos

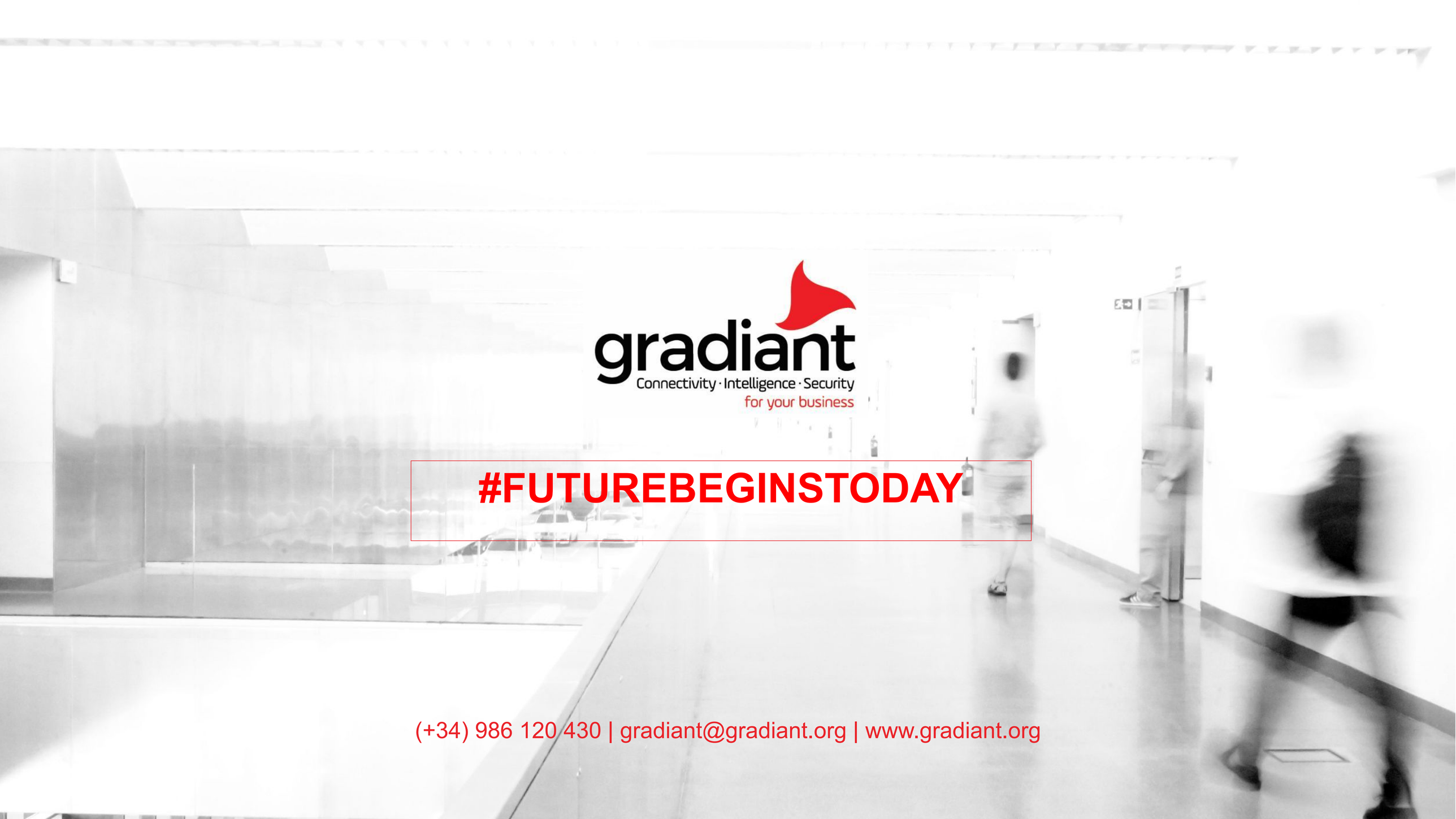
Tests Unitarios

```php
class Person
{
    /** @var int */
    private $age;

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

    public function hasLegalAge(): bool
    {
        return $this->age >= 18;
    }
}
```

```php
class PersonTest extends TestCase
{
    /** @test */
    public function has_not_legal_age_when_person_is_17()
    {
        $person = new Person( age: 17);

        $this->assertFalse($person->hasLegalAge());
    }

    /** @test */
    public function has_legal_age_when_person_is_18()
    {
        $person = new Person( age: 18);

        $this->assertTrue($person->hasLegalAge());
    }
}
```


Kata

- Kata FizzBuzz https://github.com/CodiumTeam/tdd-training-java
- Pair programming
- Baby steps



(+34) 986 120 430 | gradiant@gradiant.org | www.gradiant.org