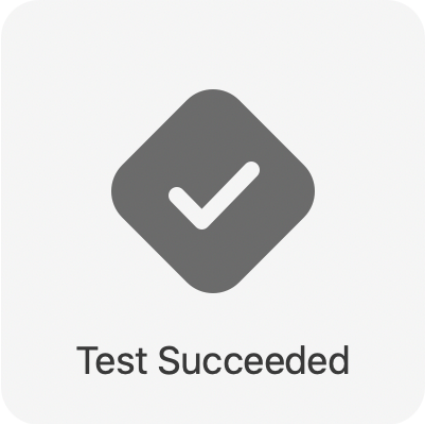
iOS Avanzado

KEEPCODING
Tech School

UnitTesting

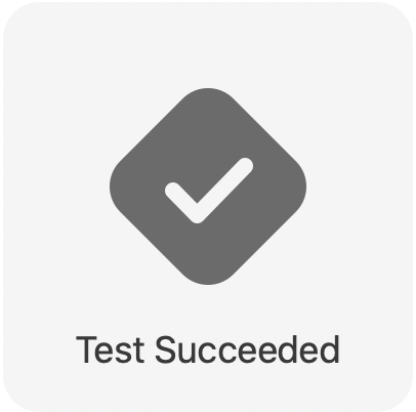
Esencialmente, una prueba unitaria o unit test es un método que crea una instancia de una pequeña porción de código de nuestra aplicación y comprueba su correcto comportamiento, independientemente de otras partes.

El Unit Testing es una herramienta esencial para el desarrollador de software, sin embargo, a veces puede ser un poco complicado de entender e implementar.



UnitTesting

Tal vez tengas una aplicación “funcional”, sin embargo, la corrección de errores, bugs, reutilización de código y constante escalabilidad del mismo pueden llegar a complicar el desarrollo a futuro. El Unit Testing nos puede ayudar a evitar éste tipo de problemas.



UnitTesting

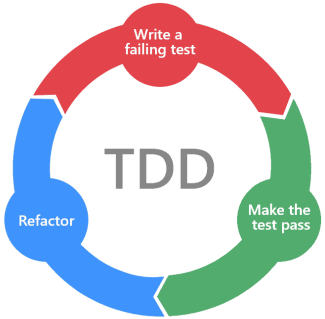
El acrónimo FIRST describe un conciso set de criterios para una efectiva prueba unitaria, éstos son:

- **Fast:** Las pruebas no deben tardar en terminar de ejecutarse.
- **Independent/Isolated:** Las pruebas no deben compartir o depender de algún estado de otra prueba.
- **Repeatable:** Debes obtener los mismos resultados cada vez que ejecutes la misma prueba, a menos que cambies alguna característica específica, ajena a datos externos que no dependan de tí.
- **Self-validating:** Las pruebas deben ser completamente automatizadas, el output debe ser solamente “pasó” o “falló”, en lugar de tener que ser interpretado por el programador.
- **Timely:** Idealmente, las pruebas deben ser escritas siempre antes del código en producción. Esto quiere decir, cuando se va a desarrollar una característica de la aplicación, primero se escriben las pruebas necesarias, y después, basándonos en éstas, implementar la característica deseada. De esta forma, las pruebas no serán obsoletas, además, las pruebas nos obligan a implementar las buenas prácticas en nuestro código.

UnitTesting - Flujo de las pruebas unitarias

Esencialmente la prueba unitaria consta de 3 fases:

- **Primero:** Se inicializa una pequeña parte del código de la aplicación que se quiere probar.
- **Luego:** Se aplica un estímulo (ejecutando dicha parte de código con ciertas características).
- **Finalmente:** Se observa el resultado final de la prueba.



UnitTesting - iOS Unit Testing Bundle

Las pruebas unitarias se ejecutan en un Target diferente al principal, dicho Target es llamado iOS Unit Testing Bundle.

Para poder hacer uso de los métodos y variables de nuestro Target principal se debe agregar un @testable import de dicho Target, sin embargo, los componentes privados seguirán siendo inaccesibles.

UnitTesting

```
import Foundation

class Cart {
  var products = 0

  func addProduct() {
    products += 1
  }

  func removeProduct() {
    products = (products > 0) ? products - 1 : products
  }

  func removeAllProducts() {
    products = 0
  }
}
```

UnitTesting

```
import XCTest
@testable import UnitTesting

class CartTests: XCTestCase {

  func testAddProduct() {
    let cart = Cart()
    cart.addProduct()
    cart.addProduct()
    cart.addProduct()
    XCTAssertEqual(cart.products, 2, "Add product failed")
  }

  func testRemoveProduct() {
    let cart = Cart()
    cart.addProduct()
    cart.addProduct()
    cart.addProduct()
    cart.removeProduct()
    XCTAssertEqual(cart.products, 3, "Remove product failed")
  }

  func testRemoveAllProducts() {
    let cart = Cart()
    cart.addProduct()
    cart.addProduct()
    cart.addProduct()
    cart.removeAllProducts()
    XCTAssertEqual(cart.products, 0, "Remove products failed")
  }
}
```

UnitTesting - ¿Cómo corro las pruebas?

```
 8  import XCTest
 9  @testable import UnitTesting
10
    class CartTests: XCTestCase {
12
      func testAddProduct() {
14      let cart = Cart()
15      cart.addProduct()
16      cart.addProduct()
17      cart.addProduct()
18      XCTAssertEqual(cart.products, 2, "Add product failed")   XCTAssertEqual failed: ("3") is not equal to ("2") - Add product failed
19    }
```

```
$24.02
Test Suite 'Selected tests' started at 2020-11-01 23:42:20.013
Test Suite 'UnitTestingTests.xctest' started at 2020-11-01 23:42:20.014
Test Suite 'CartTests' started at 2020-11-01 23:42:20.015
Test Case '-[UnitTestingTests.CartTests testAddProduct]' started.
/Users/adamjensen/Documents/Git/XCode/UnitTesting/UnitTestingTests/Cases/CartTests.swift:18: error: -[UnitTestingTests.CartTests
    testAddProduct] : XCTAssertEqual failed: ("3") is not equal to ("2") - Add product failed
Test Case '-[UnitTestingTests.CartTests testAddProduct]' failed (0.086 seconds).
Test Suite 'CartTests' failed at 2020-11-01 23:42:20.102.
   Executed 1 test, with 1 failure (0 unexpected) in 0.086 (0.088) seconds
Test Suite 'UnitTestingTests.xctest' failed at 2020-11-01 23:42:20.103.
   Executed 1 test, with 1 failure (0 unexpected) in 0.086 (0.089) seconds
Test Suite 'Selected tests' failed at 2020-11-01 23:42:20.104.
   Executed 1 test, with 1 failure (0 unexpected) in 0.086 (0.091) seconds
```

UnitTesting - Pruebas exitosas

```
8  import XCTest
9  @testable import UnitTesting
10
11 class CartTests: XCTestCase {
12
13   func testAddProduct() {
14     let cart = Cart()
15     cart.addProduct()
16     cart.addProduct()
17     cart.addProduct()
18     XCTAssertEqual(cart.products, 3, "Add product failed")
19   }
```

```
$24.02
Test Suite 'Selected tests' started at 2020-11-01 23:45:32.300
Test Suite 'UnitTestingTests.xctest' started at 2020-11-01 23:45:32.300
Test Suite 'CartTests' started at 2020-11-01 23:45:32.301
Test Case '-[UnitTestingTests.CartTests testAddProduct]' started.
Test Case '-[UnitTestingTests.CartTests testAddProduct]' passed (0.009 seconds).
Test Suite 'CartTests' passed at 2020-11-01 23:45:32.310.
   Executed 1 test, with 0 failures (0 unexpected) in 0.009 (0.009) seconds
Test Suite 'UnitTestingTests.xctest' passed at 2020-11-01 23:45:32.310.
   Executed 1 test, with 0 failures (0 unexpected) in 0.009 (0.010) seconds
Test Suite 'Selected tests' passed at 2020-11-01 23:45:32.311.
   Executed 1 test, with 0 failures (0 unexpected) in 0.009 (0.011) seconds
```

UnitTesting - Refactorizar

```
class CartTests: XCTestCase {

  var cart: Cart? // 1

  override func setUp() { // 2
    super.setUp()
    cart = Cart() // 3
  }

  func testAddProduct() {
    cart?.addProduct()
    cart?.addProduct()
    cart?.addProduct()
    XCTAssertEqual(cart?.products, 3, "Add product failed")
  }

  func testRemoveProduct() {
    cart?.addProduct()
    cart?.addProduct()
    cart?.addProduct()
    cart?.removeProduct()
    XCTAssertEqual(cart?.products, 2, "Remove product failed")
  }

  func testRemoveAllProducts() {
    cart?.addProduct()
    cart?.addProduct()
    cart?.addProduct()
    cart?.removeAllProducts()
    XCTAssertEqual(cart?.products, 0, "Remove products failed")
  }

  override func tearDown() { // 4
    super.tearDown()
    cart = nil // 5
  }
}
```

KEEPCODING

Tech School

Madrid | Barcelona | Bogotá

Datos de contacto