Universidad Nacional de La Matanza

Testing

En Kotlin Multiplatform Mobile

Agenda

- Repaso testing
- Intro a kotlin.test
- Testeando el código compartido
- Testeando el código específico de plataforma

Repaso Testing

Testing

Cuando hablamos de Testing, una de las primeras palabras que se nos viene a la cabeza es la de Calidad.

¿Qué es la calidad?

¿Cómo medimos la calidad?

Testing

Una de las formas que tenemos de medir la calidad es realizando pruebas (tests)

- La prueba consiste en la búsqueda de errores en el sistema.
- Las pruebas muestran la presencia, no la ausencia de errores (*bugs*)

¿Cómo armamos una prueba?

1. Se define una situación inicial.
2. Se realiza una operación sobre el sistema.
3. Se evalúa la situación final, comparándola con la esperada.

Este proceso es análogo a la revisión de un contrato.

Testing

Características de los tests

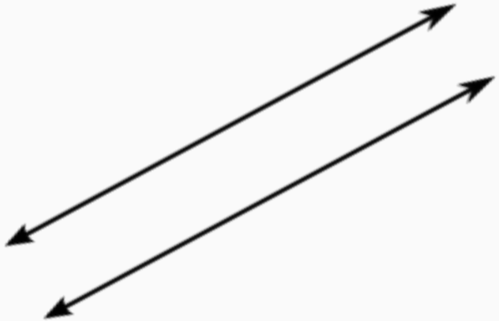
- *Fast* (Rápido)
- *Independent* (Independiente)
- *Repeatable* (Repetible)
- *Self-validating* (Auto-validado)
- *Timely* (Oportuno)

Fast

- *Ejecutar una gran cantidad de test en cuestión de segundos*
- *Encontrar bugs y asegurar el funcionamiento de una forma rápida*

Independent

- *Todas las pruebas deben ser independientes de otras*
- *La ejecución y resultado de un test no debe influenciar en otro test*



Repeatable

- *Una prueba debe poder ejecutarse y replicarse en cualquier entorno*
- *El resultado de una prueba debe ser el mismo independientemente del servidor*

Self-Validating

- *Cada test debe ser capaz de validar si el resultado obtenido es correcto o no y a su vez, ofrecer un resultado claro de lo que ha pasado.*

Timely

- *Hay que ser oportunos a la hora de hacer testing, las pruebas deben ser escritas en el momento justo que se necesitan*

Testing

Tipos de tests

- Unitarias
- Integración
- Desempeño
- Funcionales
- Aceptación
- Instalación

Testing

“Encontrar y arreglar un problema de software luego de la entrega es con frecuencia 100 veces más caro que encontrarlo y arreglarlo durante la fase de requerimientos y diseño.” - Barry Boehm

kotlin.test

kotlin.test

- Es una librería de Unit Testing **multiplataforma** escrita en Kotlin
- Permite escribir tests para el código compartido (commonTest) como para el código específico de cada plataforma (iosTest y androidTest)
- Tiene módulos de integración con JUnit5, TestNG, libs de JS, entre otras.

kotlin.test - Integración

En el archivo build.gradle del módulo compartido (shared) incluir la dependencia en el *sourceSet commonTest*:

```
kotlin {
    ...
    sourceSets {
        val commonMain by getting
        val commonTest by getting {
            dependencies {
                implementation(kotlin("test"))
            }
        }
        val androidMain by getting
        val androidTest by getting
        val iosMain by getting
        val iosTest by getting
    }
}
```

El plugin de Gradle inferirá las demás dependencias para los otros sourceSet

kotlin.test - Escribiendo tests

Crear un archivo .kt dentro del directorio ***commonTest*** y allí comenzar a escribir los tests como funciones.

Se debe anotar cada uno con la anotación **@Test**. Ej:

```
import kotlin.test.Test
import kotlin.test.assertEquals

class TestCalculadora {

    @Test
    fun Test_suma() {
        val calculadora = Calculadora()

        val resultado = calculadora.sumar(3, 2)

        assertEquals(5, resultado)
    }

}
```

kotlin.test - Escribiendo tests

También podemos utilizar @BeforeTest y @AfterTest para ejecutar código antes y después de cada test, respectivamente.

```
import kotlin.test.BeforeTest
import kotlin.test.AfterTest

class TestDeRepositorio {

    @BeforeTest
    fun antesDeCadaTest() {
        abrirBaseDeDatos()
    }

    @AfterTest
    fun luegoDeCadaTest() {
        cerrarBaseDeDatos()
    }

    ...

    private fun abrirBaseDeDatos() {
        // Codigo
    }

    private fun cerrarBaseDeDatos() {
        // Codigo
    }
}
```

kotlin.test - Escribiendo tests

kotlin.test trae integrado un Asserter con funciones que nos permiten escribir las validaciones en los tests:

```
assertEquals() / assertNotEquals()

assertNull() / assertNotNull()

assertTrue() / assertFalse()

assertContains()
```

kotlin.test - Ejecutando tests

El comando **check** nos permite correr *todos* los tests del proyecto. Para ello en la terminal escribimos desde el directorio raíz del proyecto:

```
./gradlew check
```

Esto ejecutará todos los tests incluidos en el módulo compartido

> *"Cada vez que ejecutamos el comando check, los tests del directorio commonTest se ejecutan 1 vez por cada plataforma"*

Links útiles

Testing en proyectos Kotlin multiplataforma

Dependencias de test

kotlin.test

Módulos kotlin.test