

El futuro digital es de todos





CICLO II:

Programación Básica en Java







Sesión 17: Introducción a Java

Conexión a Base de Datos







Objetivos de la sesión

Al finalizar esta sesión estarás en capacidad de:

- 1. Manipular la base de datos relacional de más de una tabla con relaciones construida en SQLite
- Desarrollar una aplicación constituida de varios métodos utilizando entorno gráfico que conecte a base de datos relacional y lleve a cabo operaciones sobre esta
- 3. Definir y aplicar pruebas unitarias sobre los distintos métodos desarrollados en una aplicación.





El futuro digital es de todos



lotivos para realizar una Prueba Unitaria

- Las pruebas unitarias demuestran que la lógica del código está en buen estado y que funcionará en todos los casos.
- Aumentan la legibilidad del código y ayudan a los desarrolladores a entender el código base, lo que facilita hacer cambios más rápidamente.
- Las pruebas unitarias bien realizadas sirven como documentación del proyecto.
- Se realizar en pocos milisegundos, por lo que se podrá realizar cientos de ellas en muy poco tiempo.





MinTIC



Motivos para realizar una Prueba Unitaria

- Las pruebas unitarias permiten al desarrollador refactorizar el código más adelante y tener la garantía de que el módulo sigue funcionando correctamente. Para ello se escriben casos de prueba para todas las funciones y métodos, para que cada vez que un cambio provoque un error, sea posible identificarlo y repararlo rápidamente.
- La calidad final del código mejorará ya que, al estar realizando pruebas de manera continua, al finalizar, el código será limpio y de calidad.
- Como las pruebas unitarias dividen el código en pequeños fragmentos, es posible probar distintas partes del proyecto sin tener que esperar a que otras estén completadas.







Las 3 A's del unit testing

- Arrange (organizar). Es el primer paso de las pruebas unitarias. En esta parte se definen los requisitos que debe cumplir el código.
- Act (actuar). Es el paso intermedio de las pruebas, el momento de ejecutar el test que dará lugar a los resultados que deberás analizar.
- Assert (afirmar). En el último paso, es el momento de comprobar si los resultados obtenidos son los que se esperaban. Si es así, se valida y se sigue adelante. Si no, se corrige el error hasta que desaparezca.







Cómo llevar a cabo Pruebas Unitarias

El proceso de los tests unitarios puede realizarse de manera manual, aunque lo más común es automatizar el procedimiento a través de herramientas.

- <u>xUnit</u>: se trata de una herramienta de pruebas unitarias para el framework .NET.
- <u>Junit</u>: es un conjunto de bibliotecas para realizar pruebas unitarias de aplicaciones Java.
- <u>NUnit</u>: inicialmente portado desde JUnit, NUnit 3 se ha reescrito por completo para dotarlo de nuevas características y soporte para una amplia gama de plataformas .NET.
- <u>PHPUnit</u>: entorno de pruebas unitarias en el lenguaje de programación PHP.







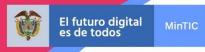


Pruebas Unitarias - JUnit

JUnit es una librería desarrollada para poder probar el funcionamiento de las clases y métodos que componen una aplicación java, y asegurarnos de que se comportan como deben ante distintas situaciones de entrada.

JUnit es un marco simple para escribir pruebas repetibles para el <u>lenguaje de</u> <u>programación Java</u>. Es una instancia de la arquitectura xUnit para marcos de prueba de unidades.







Pruebas Unitarias - JUnit

Las características principales son:

- Afirmaciones: que le permiten personalizar cómo probar los valores en sus pruebas
- Corredores de prueba: que le permiten especificar cómo ejecutar las pruebas en su clase
- Reglas: que le permiten modificar con flexibilidad el comportamiento de las pruebas en su clase
- Suites: que le permiten construir juntos un conjunto de pruebas de muchas clases diferentes







Buenas prácticas para las pruebas unitarias

- Las pruebas unitarias deberían ser independientes.
- Probar sólo un código a la vez.
- Seguir un esquema claro.
- Cualquier cambio necesita pasar el test.
- Corregir los bugs identificados durante las pruebas antes de continuar
- Realizar pruebas regularmente mientras se programa.





Pruebas unitarias - Ejemplo

```
public class Calculadora {
    public static int suma(int a, int b) {
        return a + b;
    public static int resta(int a, int b) {
        return a - b;
```





Pruebas unitarias – Aplicando @Test en el Ejemplo

```
@Test
public void testSuma() {
    System.out.println("suma");
    int a = 5;
    int b = 2;
    int expResult = 7;
    int result = Calculadora.suma(a, b);
    assertEquals(expResult, result);

if (result != expResult) {
    fail("The test case is a prototype.");
}
```

```
@Test
public void testResta() {
    System.out.println("resta");
    int a = 5;
    int b = 2;
    int expResult = 4;
    int result = Calculadora.resta(a, b);
    assertEquals(expResult, result);

if (result != expResult) {
    fail("The test case is a prototype.");
}
```

A tener en cuenta: El condicional if antes del método fail es agregado. El bloque if puede estar en comentarios para que la verificación se haga por asserEquals.







Los métodos assertEquals y fail

- Método assertEquals: Se utiliza para comparar dos tipos de datos u objetos y afirmar que son iguales.
- Método fail: Permite indicar que hay un fallo en la prueba.





IGRACIASPOR SER PARTE DE ESTA EXPERIENCIA DE APRENDIZAJE!



