P01 - Pruebas (práctica)

Entornos de Desarrollo

1º DAW

Yun Elia Gil Martínez

Rizwan Baig

Curso 2024/2025

Índice

[1. Qué son las pruebas, para qué sirven y por qué son tan importantes en el ciclo de desarrollo de software 3](#__RefHeading___Toc128_2340594174)

[1.1 ¿qué son? 3](#__RefHeading___Toc140_2340594174)

[1.2 ¿Para qué sirven? 3](#__RefHeading___Toc132_2340594174)

[1.3 ¿Por qué son tan importantes? 3](#__RefHeading___Toc134_2340594174)

[2. Qué diferencia hay entre pruebas unitarias, pruebas de integración y pruebas funcionales. 3](#__RefHeading___Toc114_2340594174)

[1.1 Pruebas unitarias 3](#__RefHeading___Toc157_2340594174)

[1.2 Pruebas de integración 3](#__RefHeading___Toc144_2340594174)

[1.3 Pruebas funcionales 4](#__RefHeading___Toc146_2340594174)

[1.4 Diferencias entre las pruebas 4](#__RefHeading___Toc259_1570237854)

[3. Qué es el "TDD" (Desarrollo Guiado por Pruebas) 4](#__RefHeading___Toc116_2340594174)

[4. Qué es Junit 6](#__RefHeading___Toc118_2340594174)

[5. Qué diferencias hay entre JUnit 5 y JUnit 4 6](#__RefHeading___Toc120_2340594174)

[6. Qué buenas prácticas se deben seguir para hacer un buen diseño de pruebas 6](#__RefHeading___Toc122_2340594174)

[7. Qué futuro se puede esperar de las pruebas con la llegada de la IA 7](#__RefHeading___Toc124_2340594174)

[8. Bibliografía 8](#__RefHeading___Toc126_2340594174)

1. Qué son las pruebas, para qué sirven y por qué son tan ****importantes**** en el ciclo de desarrollo de software

## 1.1 ¿qué son?

* Las pruebas son los procesos con los cuales se pone a prueba una aplicación en proceso de creación y ya terminada para detectar errores o anomalías tanto para su funcionamiento como para la mejora de su código.

## 1.2 ¿Para qué sirven?

* Sirven para verificar el funcionamiento de las aplicaciones o y probar casos extremos de uso con el fin de minimizar errores y mejoras en su fiabilidad.

## 1.3 ¿Por qué son tan importantes?

* Permiten aplicar mejoras y no interrumpir el funcionamiento existente, de esa manera se puede prevenir de los fallos para que no afecte la seguridad de app.

# **Qué diferencia hay entre pruebas unitarias, pruebas de integración y pruebas funcionales**.

## 1.1 Pruebas unitarias

* Son pruebas que se hacen sobre los métodos o funciones concretas para comprobar su funcionalidad.
* Es un bloque de código que verifica la precisión de un bloque más pequeño y aislado de código de aplicación. Está diseñada para verificar que el bloque de código se ejecuta según lo esperado, de acuerdo con la lógica que hayamos aplicado. Solo interactúa con el bloque de código a través de entradas y salidas.
* Son menos útiles cuando el tiempo es limitado, para incluir código heredado, cuando el código evoluciona muy rápidamente, etc.

## 1.2 Pruebas de integración

* Son pruebas que se hacen a un conjunto de métodos que forman parte del mismo sistema para comprobar su comportamiento.
* Se usan para probar la infraestructura de la aplicación y todo el marco, como las bases de datos, el sistema de archivos, los dispositivos de red, etc.
* Usan los componentes reales que emplea la aplicación en producción, necesitan más código y procesamiento de datos y tardan más en ejecutarse.

## 1.3 Pruebas funcionales

* Son aquellas las que evalúan si la app o sistema creado cumple con los parámetros que se exige sin importar como este implementado el código.
* Garantiza que el software funcione correctamente antes de ser lanzado, esto evita que los usuarios finales se encuentren con problemas o errores. Al solucionar los problemas al principio del desarrollo permite una corrección temprana y ahorran tiempo y recursos.

## 1.4 Diferencias entre las pruebas

* Las pruebas unitarias, de integración y funcionales se diferencian principalmente en el **momento en que se realizan** y en el **objetivo que persiguen** dentro del proceso de desarrollo de software.
* **Si quieres comprobar que todo el sistema funciona como se espera, lo mejor son las pruebas funcionales.  
  Si has unido tu código con otros y necesitas asegurarte de que no haya fallos, haz pruebas de integración.  
  Y si acabas de escribir una parte del código y quieres ver que funciona bien por sí sola, haz pruebas unitarias, son más rápidas de generar y así garantiza el buen funcionamiento a cada paso del código.**

# **Qué es el "TDD" (Desarrollo Guiado por Pruebas)**

Es una práctica de ingeniería de software que involucra otras dos prácticas.

* Test First Development
* Refactoring

Consiste en primero escribir las pruebas y luego realizar la implementación necesaria para que estas pruebas pasen y finalmente refactorizar el código escrito.

De forma habitual las pruebas se realizan sobre el código directamente, pero en TDD, las pruebas se escriben antes para prevenir los errores.

# **Qué es Junit**

Junit es un framework (Conjunto de herramientas) exclusivo para java que permite la ejecución de clases de manera controlada para comprobar que los métodos realizan su función correctamente. Dicho de otra manera es una plataforma para crear y aplicar pruebas sobre el código que generamos en java.

Es de código abierto creado por Kent Beck y Erich Gamma.

# **Qué diferencias hay entre JUnit 5 y JUnit 4**

Ambos son frameworks de pruebas basados en java, el Junit 5 es el modelo de pruebas más avanzado comparado con Junit4.

En tema de *arquitectura* Junit5 admite diseños con una arquitectura más modular y extensible, admite características de java 8 y también puede soportar expresiones de lambda, en cambio Junit4 sigue una arquitectura monolítica y tiene soporte limitado para expresiones lambda.

En Junit5 introduce nuevas anotaciones como @BeforeEach, @AfterEach, @BeforeAll y @AfterAll que Junit4 no tiene.

En Junit5 tienen un modelo de extensión bastante poderoso llamado @ExtendWith y Junit4 carece de este modelo de soporte.

# **Qué buenas prácticas se deben seguir para hacer un buen diseño de pruebas**

Escribir pruebas unitarias, integrales y funcionales es muy importante para conocer el flujo que queremos comprobar y para que estas pruebas sean eficaces, deben seguir un tipo de estándar para tener los resultados esperados, a continuación, describo algunas reglas para poder tener un buen diseño.

* **Escoge una buena convención de nombres**

Procura que los nombres que le des a tus pruebas sigan alguna clase de estándar, elegir un estándar facilita el entendimiento del propósito.

* **Estructura de Pruebas**

Procurar que el código dentro de las pruebas tenga una estructura clara. Debe de ser posible identificar, por lo menos, las siguientes secciones en orden: los datos de prueba, las operaciones y las validaciones

* **Evitar crear instancias dentro de las pruebas**

No de recomienda crear instancias dentro de las pruebas dado que limita la mantenibilidad de nuestras pruebas en el tiempo.

* **Evitar hacer inserciones manuales a la base de datos**

Debemos evitar insertar la data directamente en nuestras pruebas y en su lugar hacer uso de la anotación de @sql de Spring Data.

* **Crear abstracciones para las configuraciones**

Conforme que vamos agregando nuevas funcionalidades a nuestra aplicación, el número de clases para pruebas de flujos concretos también crecerá. a veces las clases se comparten una misma configuración, para evitar copiar y pegar estas anotaciones en cada clase, debemos agruparlas en una clase abstracta y hacer uso de ellas. Una cosa

# **Qué futuro se puede esperar de las pruebas con la llegada de la IA**

Con la llegada de la IA en mi opinión, estas pruebas pueden desaparecer muy pronto por los siguientes motivos:

* + IA funciona mucho más rápido que generar estas pruebas.
  + Es más económico ya que reduce la necesidad de programar manualmente que emplea menos intervención humana.
  + IA da una cobertura más amplia ya que tiene menos riesgos de fallos.
  + Automatización de diagramas con IA puede generar UML directamente con el código fuente.

También es posible que no desaparezcan las pruebas directas si no que la inteligencia artificial lo crea automáticamente a partir del código fuente con lo cual se perjudicaría el uso de plataformas como JUnit etc.

# **Bibliografía**

* https://www.wearetesters.com/investigacion-ux/pruebas-qa-funcionales/
* https://learn.microsoft.com/es-es/aspnet/core/test/integration-tests?view=aspnetcore-9.0
* https://aws.amazon.com/es/what-is/unit-testing/
* https://medium.com/somos-pragma/buenas-pr%C3%A1cticas-y-recomendaciones-para-tus-pruebas-unitarias-con-spring-boot-7bd7b7861119
* https://www.loadview-testing.com/es/blog/tipos-de-pruebas-de-software-diferencias-y-ejemplos/