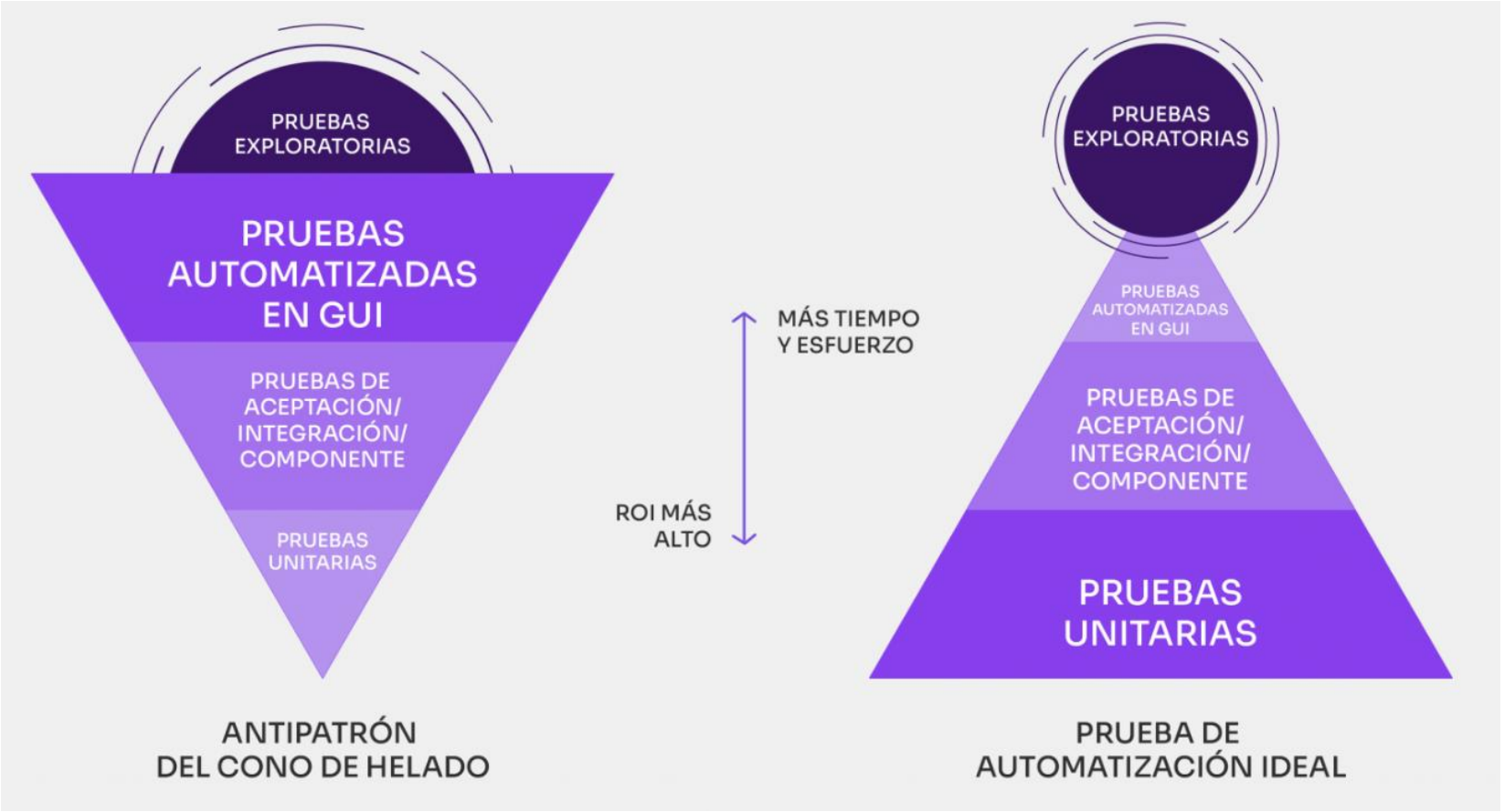

Curso Spring Boot

Temario

- **Testing**
  - Pirámide de Testing
  - TDD (Test Driven Development)
  - Introducción a herramientas: Junit, Mockito, DataJPATest y TestRestTemplate
  - Tests Unitarios
  - Tests de Integración

Pirámide de Testing



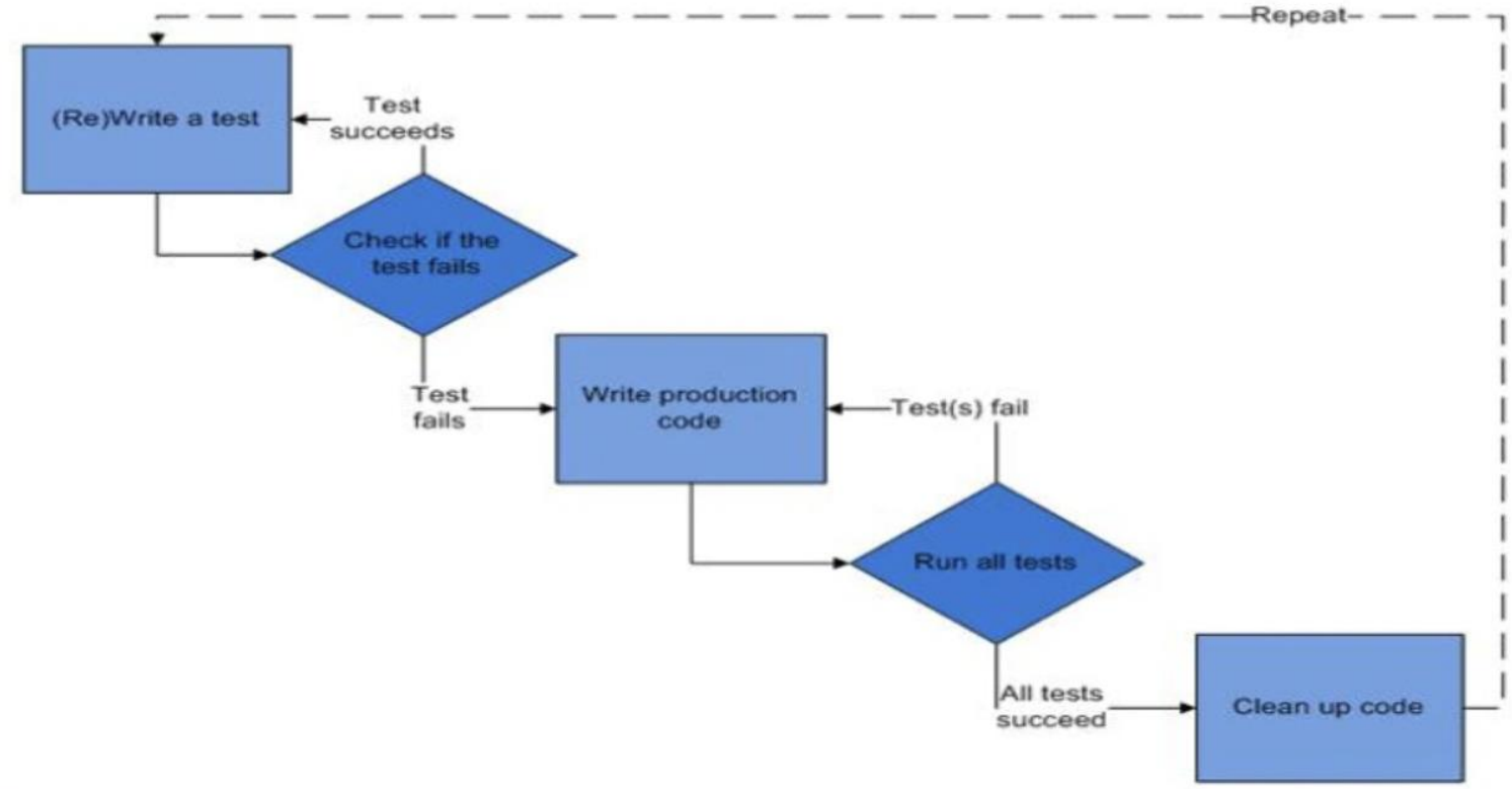
TDD

- ¿Qué es?
  - Es una práctica de programación que consiste en escribir primero las pruebas unitarias y después escribir el código fuente que pase la prueba satisfactoriamente y, por último, refactorizar el código escrito (refactoring).
  - Es una práctica que envuelve el desarrollo en su conjunto, especialmente el diseño de Software

- Objetivo y propósito
  - Lograr un código limpio y robusto que funcione (Diseño manejado por las pruebas)

TDD

- Ciclo de desarrollo conducido por pruebas



TDD

- Antes de empezar el ciclo se define una lista de requisitos

1. **Elegir un requisito**: se elige el requisito que aporte mayor conocimiento y a la vez sea de fácil implementación.
2. **Escribir una prueba**: Se comienza escribiendo una prueba para el requisito. Para ello el programador debe entender claramente las especificaciones y los requisitos de la funcionalidad que está por implementar.
3. **Verificar que la prueba falla**: Si la prueba no falla es porque el requerimiento ya estaba implementado o porque la prueba es errónea.
4. **Escribir la implementación**: Escribir el código más sencillo que haga que la prueba funcione

TDD

- Antes de empezar el ciclo se define una lista de requisitos

5. **Ejecutar las pruebas automatizadas**: Verificar si todo el conjunto de pruebas funciona correctamente.
6. **Eliminación de duplicación**: El paso final es la refactorización, que se utilizará principalmente para eliminar código duplicado.
7. **Actualización de la lista de requisitos**: Se actualiza la lista de requisitos tachando el requisito implementado. Asimismo se agregan requisitos que se hayan visto como necesarios durante este ciclo y se agregan requerimientos de diseño.

Este ciclo se suele abreviar como **rojo-verde-refactor**

TDD

- Características
  - Cuando los equipos ejecutan TDD consistentemente en sus mentes, las características se producen con mayor rapidez. Características más rápidas significan menos tiempo al mercado. Reducción del período de tiempo de salida al mercado significa mayor posibilidades de obtener más clientes. Lo mejor de todo es que la velocidad de comercialización está respaldado por una red de seguridad de las pruebas que permiten un cambio rápido en la misma cantidad de tiempo
  - Proporciona un gran valor añadido en la creación de software, produciendo aplicaciones de más calidad y en menos tiempo. Ofrece más que una simple validación del cumplimiento de los requisitos, también puede guiar el diseño de un programa
  - Sólo se implemente el código necesario para resolver un caso de prueba concreto (pasar la prueba) hace que el código creado sea el mínimo necesario, reduciendo redundancia
  - El código escrito con TDD se respalda en pruebas, cualquier cambio que rompe el código se descubre rápidamente. En esencia, el código escrito con TDD es más fácil de cambiar y más fácil de arreglar.

TDD

- Inconvenientes y limitaciones
  - El desarrollo guiado por pruebas requiere que las pruebas puedan automatizarse. Esto resulta complejo en los siguientes casos:
  - Bases de datos. Hacer pruebas de código que trabaja con base de datos es complejo porque requiere generar unos datos conocidos antes de hacer las pruebas y verificar que el contenido de la base de datos es el esperado después de la prueba. Los objetos simulados (MockObjects) son otra opción
  - Interfaces gráficas de usuarios (GUIs). aunque hay soluciones parciales propuestas