Pruebas de Software

Unitarias

Integración

Cobertura

Mejorando la calidad del software a través de pruebas efectivas

Importancia de las Pruebas de Software

Calidad

- Asegura funcionamiento correcto
- Consistencia en los resultados
- Cumplimiento de requisitos

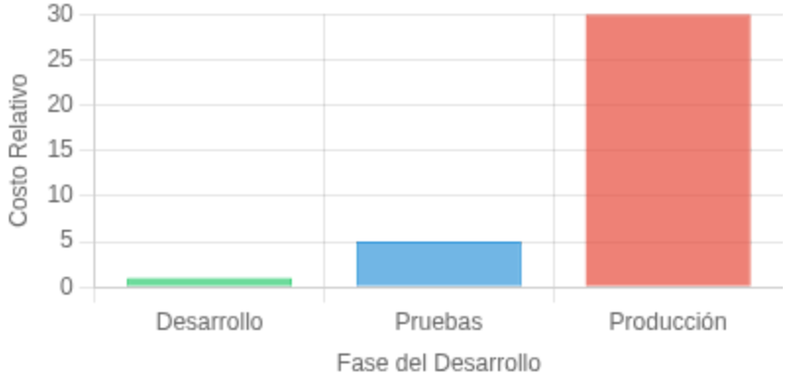
Detección Temprana

- Menor costo de corrección
- Reducción de bugs en producción
- Ciclos de desarrollo más cortos

Refactorización Segura

- Cambios sin introducir errores
- Modernización de código
- Mayor mantenibilidad

Costo de Detección de Errores por Fase



Pruebas Unitarias

Propósito

Probar de forma aislada unidades individuales de código para verificar su correcto funcionamiento.

Unidad: Función, método, clase o módulo con una responsabilidad única y claramente definida.

Buenas Prácticas

Pruebas Independientes

Cada prueba debe ejecutarse de forma aislada sin depender de otras pruebas o estados previos.

Nomenclatura Clara

Usar nombres descriptivos que indiquen qué se está probando y cuál es el resultado esperado.

Rápidas y Concisas

Las pruebas deben ejecutarse rápidamente y enfocarse en una sola funcionalidad.

Patrón AAA

Arrange (Preparar), Act (Actuar), Assert (Comprobar) para estructurar las pruebas de manera consistente.

Sin Lógica Compleja

Evitar condicionales o bucles en las pruebas para mantenerlas simples y claras.

Mock de Dependencias

Usar objetos simulados para aislar la unidad de prueba de sus dependencias externas.

Anotaciones para Deshabilitar Pruebas

JUnit 4: @Ignore

```java
import org.junit.Test;
import org.junit.Ignore;

public class CalculadoraTest {

    @Test
    public void testSuma() {
        // Test code here
    }

    @Ignore("Funcionalidad en desarrollo")
    @Test
    public void testDivision() {
        // Esta prueba no se ejecutará
    }
}
```

JUnit 5: @Disabled

```java
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.Disabled;

public class CalculadoraTest {

    @Test
    void testSuma() {
        // Test code here
    }

    @Disabled("Funcionalidad en desarrollo")
    @Test
    void testDivision() {
        // Esta prueba no se ejecutará
    }
}
```

Pruebas de Integración y Cobertura de Código

Pruebas de Integración

Definición: Verifican la interacción correcta entre diferentes componentes o módulos del sistema.

Objetivo: Detectar problemas de interfaces, comunicación y dependencias entre módulos.

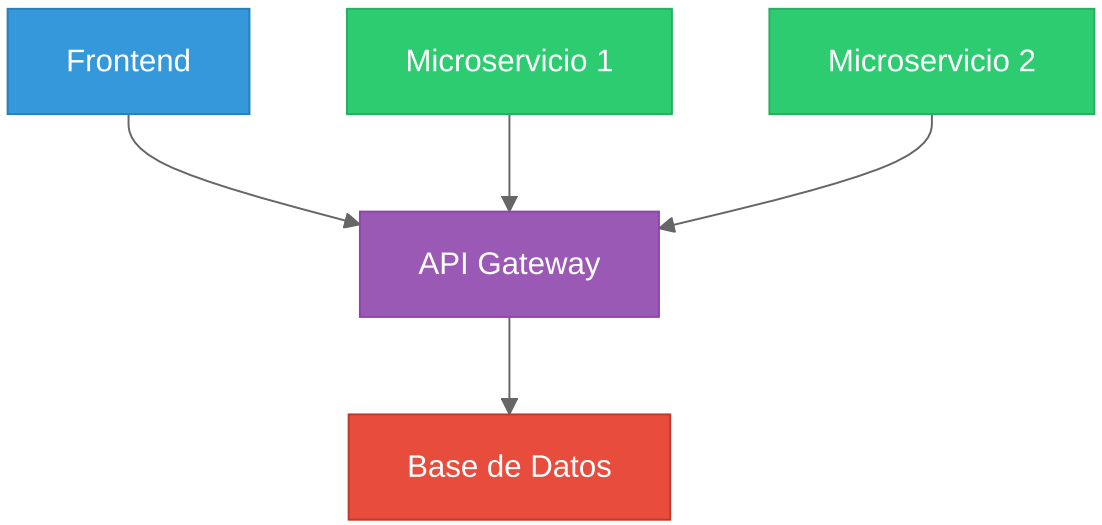
Tipos de Integración

Integración Incremental
Componentes se integran uno a uno, permitiendo identificar errores específicos.

Integración Big Bang
Todos los componentes se integran simultáneamente para pruebas.

Integración Descendente
Comienza con módulos de alto nivel hasta llegar a los de bajo nivel.

Integración Ascendente
Comienza con módulos de bajo nivel hacia los de alto nivel.



Cobertura de Código

Métrica que indica el porcentaje de código fuente que es ejecutado durante las pruebas automatizadas.

Principales Métricas

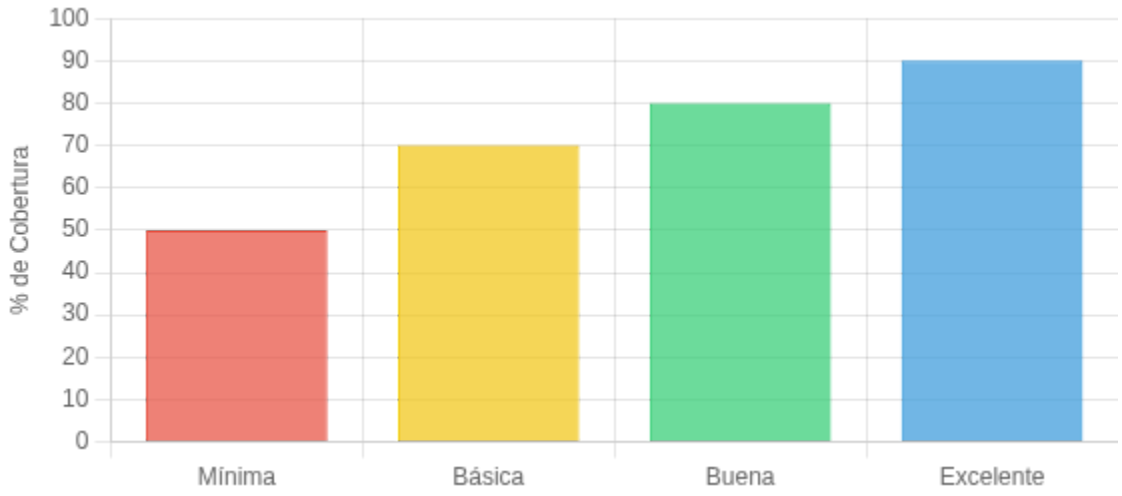
Cobertura de Líneas
Porcentaje de líneas de código ejecutadas durante las pruebas.

Cobertura de Ramas
Porcentaje de ramas condicionales (if/else, switch) que han sido ejecutadas.

Cobertura de Funciones
Porcentaje de métodos o funciones ejecutados durante las pruebas.

Cobertura de Caminos
Porcentaje de posibles caminos de ejecución cubiertos por las pruebas.

Interpretación de Cobertura



⚠ 100% de cobertura no garantiza ausencia de errores