# INSTITUTO POLITÉCTICO NACIONAL

## UNIDAD PROFESIONAL INTERDISCIPLINARIA DE INGENIERÍA Y CIENCIAS SOCIALES Y ADMINISTRATIVAS

**6NM60 Ingeniería de Pruebas**

**Plan ACS/SQA: Calculadora digital intuitiva y multifuncional.**

# MANUAL DE USUARIO


Alumnos:

García Méndez Juan Carlos

Conde Basilio Leonardo

Ramos Velázquez Felipe

Villaseñor Trejo Javier Enrique

Docente:

Cruz Martínez Ramón


Fecha: 10 de marzo del 2025

# Contenido

## Propósito

Garantizar que el desarrollo de la calculadora cumpla con los requisitos funcionales (RF), no funcionales (RNF) y estándares de calidad definidos, mediante procesos sistemáticos de auditoría, verificación y validación.

## 1.2 Alcance y Contexto

- **Stakeholders Principales**:
  - **Cliente**: Institución educativa que requiere una herramienta para enseñanza de matemáticas.
  - **Usuario Final**: Estudiantes y profesionales que necesitan realizar operaciones básicas y gráficos simples.
  - **Equipo Técnico**: Analista, diseñador UI/UX, programador Java, tester.

**Cubre todas las fases del ciclo de vida del software: análisis, diseño, codificación, pruebas y mantenimiento.** Aplica a los módulos:

- Operaciones básicas y memoria (RF01-RF11).
- Interfaz gráfica (RNF02, RNF03).
- Gráficos 2D (RF03).
- Historial de operaciones (RF04).

# 2. Objetivos del Proyecto

### 2.1 Objetivos Generales

- Entregar una calculadora funcional en 30 días.
- Garantizar una tasa de satisfacción del usuario del 90%.

### 2.2 Objetivos Específicos

| Objetivo | Métrica de Éxito |
|---|---|
| **Implementar operaciones básicas** | 100% de los casos de uso críticos (CUC01-CUC03) funcionando. |
| **Integrar gestión de memoria** | Botones M+, MR, MC operativos sin errores en pruebas de estrés. |
| **Optimizar rendimiento** | Respuesta ≤1 segundo en el 95% de las operaciones. |
| **Asegurar compatibilidad** | Funcionamiento en Windows 10, macOS 12 y navegadores modernos. |

### 3. Documentos de Referencia

- IEEE Std 730-2014 (SQA Processes).
- Especificación de Requisitos (ERS).
- Manual de Diseño Técnico.
- Plan de Pruebas.

# 3. Alcance Detallado

### 3.1 Funcionalidades Incluidas
**Módulo de Cálculo**

- **Operaciones Básicas**:
  - Suma, resta, multiplicación y división con soporte para números negativos y decimales (2 dígitos).
  - Validación de división por cero (RF02).
- **Gestión de Memoria**:
  - Almacenamiento (M+), recuperación (MR), limpieza (MC) y acumulación (RF08-RF11).

**Módulo de Interfaz**

- Diseño tipo teclado telefónico (3x4) con botones numéricos y operadores (RF06).
- Historial de operaciones visible (últimas 20) en ventana lateral (RF04).
- Feedback visual mediante colores y tooltips (RNF02).

**Módulo de Seguridad**

- Validación de entradas para evitar inyección de código (RNF04).

### 3.2 Exclusiones

- **No Incluye**:
  - Persistencia de datos tras cerrar la aplicación (limitación técnica aceptada).
  - Funciones trigonométricas o estadísticas (priorizadas para una fase futura).
  - Soporte para dispositivos móviles (iOS/Android). Entregables y

## 4. Objetivos de Calidad

| Objetivo | Métrica de Cumplimiento | Estándar |
|---|---|---|
| **Funcionalidad Correcta** | 100% de los RF implementados. | ISO/IEC 25010 |
| **Rendimiento Óptimo** | ≤1 seg. por operación (RNF01). | Benchmark interno |
| **Usabilidad Intuitiva** | 90% de usuarios navegan sin manual (RNF02). | SUS Score ≥75 |
| **Compatibilidad Garantizada** | Funciona en Windows 10+, macOS 12+ (RNF03). | Checklist SO |
| **Código Mantenible** | 85% de cobertura con JUnit. | ISO/IEC 5055 |

# 5. Organización del SQA

### 5.1 Roles y Responsabilidades

| Rol | Responsabilidades |
|---|---|
| **SQA Manager** | Supervisar el cumplimiento del plan ACS/SQA. |
| **SQA Auditor** | Realizar auditorías de código y procesos. |
| **Equipo de Desarrollo** | Implementar estándares de codificación. |
| **Equipo de Pruebas** | Ejecutar y documentar casos de prueba. |

### 5.2 Estructura del Equipo

Copy

SQA Manager → SQA Auditor

↓

Equipo de Desarrollo ↔ Equipo de Pruebas

# 6. Tareas y Actividades de SQA

### 6.1 Auditorías de Proceso

- **Revisión de Requisitos**: Validar que los RF/RNF estén alineados con las necesidades del cliente (semanal).
- **Inspecciones de Diseño**: Verificar coherencia entre diagramas UML y el código (tras cada sprint).

### 6.2 Auditorías de Producto

- **Código Fuente**:
  - Cumplimiento de convenciones Java (camelCase, Javadoc).
  - Uso adecuado de BigDecimal para cálculos precisos.
- **Interfaz Gráfica**:
  - Validación de disposición 3x4 (RF06) con herramientas como **Figma**.

### 6.3 Actividades Clave por Fase

| Fase del Proyecto | Actividades SQA |
|---|---|
| **Análisis** | Validar ERS con stakeholders. |
| **Diseño** | Revisar diagramas de arquitectura y flujos. |
| **Codificación** | Auditorías de código semanales. |
| **Pruebas** | Verificar cobertura y casos de prueba. |
| **Mantenimiento** | Monitorear bugs reportados post-lanzamiento |

# 7. Estándares y Procedimientos

## 7.1 Estándares Técnicos

- **Codificación**:
  - Convenciones Java (Oracle Code Conventions).
  - Documentación Javadoc en métodos públicos.
- **Pruebas**:
  - Nomenclatura de pruebas: [Clase]Test.java (ej: CalculadoraTest.java).
  - Uso de JUnit 5 y AssertJ para legibilidad.

## 7.2 Procedimientos de Control de Calidad

- **Control de Cambios**:
  1. Solicitud de cambio vía GitHub Issues.
  2. Evaluación de impacto por el SQA Manager.
  3. Aprobación por el comité de cambios.
- **Gestión de Configuración**:
  - Uso de Git con ramas: main (producción), develop (desarrollo), feature/* (funcionalidades).

# 8. Revisiones y Auditorías

## 8.1 Revisiones Técnicas Formales

| Tipo de Revisión | Frecuencia | Participantes |
|---|---|---|
| **Revisión de Diseño** | Por sprint | Arquitecto, Programador, SQA. |
| **Revisión de Código** | Semanal | 2 programadores + SQA Auditor. |
| **Revisión de Pruebas** | Post-release | Tester, Cliente, SQA Manager. |

## 8.2 Checklist de Auditoría de Código

- Variables con nombres descriptivos (ej: valorMemoria).
- Métodos cortos (<30 líneas).
- Manejo de excepciones (try-catch para DivisionByZero).
- Ausencia de código comentado o muerto.

# 9. Estrategia de Pruebas

### 9.1 Tipos de Pruebas

| Tipo | Herramienta | Criterios de Éxito |
|---|---|---|
| **Unitarias** | JUnit 5 | 100% de los RF validados. |
| **Integración** | Mockito | Módulos GUI + Lógica funcionan. |
| **Sistema** | Jenkins + Selenium | Flujos completos sin errores. |
| **Rendimiento** | JMeter | ≤1 seg. por operación (pico: 100 usuarios). |
| **Usabilidad** | Figma + Hotjar | 90% de usuarios completan tareas sin ayuda. |

### 9.2 Casos de Prueba Críticos

| ID | Descripción | Entrada | Resultado Esperado |
|---|---|---|---|
| **CP01** | División por cero | 5 / 0 | Mensaje de error claro. |
| **CP02** | Acumulación en memoria | M+5 → M+3 → MR | 8.00 |

# 10. Gestión de la Configuración

### 10.1 Elementos Configurables

- **Código Fuente**: Repositorio GitHub con ramas protegidas.
- **Documentación**: Confluence para manuales técnicos y de usuario.
- **Entornos**:
  - Desarrollo: NetBeans + Java 17.
  - Producción: JAR ejecutable + JRE 17.

### 10.2 Control de Versiones

- **Etiquetado**: Versiones semánticas (ej: v1.2.3).
- **Políticas**:
  - main solo acepta merges desde develop tras aprobación de SQA.
  - Commits con mensajes descriptivos (ej: "FEAT: Añadido botón M+").

# 11. Gestión de Riesgos de Calidad

### 11.1 Identificación de Riesgos

| Riesgo | Impacto | Probabilidad | Mitigación |
|---|---|---|---|
| **Errores de redondeo en cálculos** | Alto | Media | Usar BigDecimal con precisión de 4 decimales. |
| **Incompatibilidad en navegadores** | Crítico | Baja | Pruebas en Chrome, Firefox y Safari. |
| **Baja usabilidad del historial** | Medio | Alta | Prototipado temprano en Figma. |

### 11.2 Plan de Contingencia

- **Escenario**: Fallo en renderizado de gráficos.
  - **Acción**: Rollback a la última versión estable y depuración con Matplotlib.
- **Escenario**: Bajo rendimiento en equipos antiguos.
  - **Acción**: Optimizar uso de memoria con LinkedList en lugar de ArrayList.

# 12. Herramientas y Recursos

### 12.1 Herramientas SQA

| Categoría | Herramienta | Uso |
|---|---|---|
| **Gestión de Código** | GitHub + Git | Control de versiones. |
| **Pruebas** | JUnit, JMeter, Selenium | Automatización y rendimiento. |
| **Monitoreo** | SonarQube | Análisis estático de código. |
| **Documentación** | Confluence | Manuales técnicos y de usuario. |

### 12.2 Recursos Humanos

- **SQA Manager**: 20 horas/semana.
- **Auditor Externo**: 10 horas/revisión.

# 13. Capacitación

### 13.1 Plan de Capacitación

| Público Objetivo | Tema | Duración | Método |
|---|---|---|---|
| **Desarrolladores** | Buenas prácticas en Java | 8 horas | Taller práctico. |
| **Testers** | Uso de JUnit y JMeter | 6 horas | Sesión virtual. |
| **Usuarios Finales** | Tutorial de la calculadora | 1 hora | Video interactivo. |

# 14. Mantenimiento y Mejora Continua

### 14.1 Post-Implementación

- **Soporte Nivel 1**: Corrección de bugs críticos en ≤24 horas.
- **Actualizaciones**: Ciclo de lanzamientos mensuales para nuevas funciones.

### 14.2 Retroalimentación y Mejora

- **Encuestas de Usuario**: Recolección trimestral de feedback.
- **Métricas Clave**:
  - Tasa de errores post-lanzamiento.
  - Tiempo promedio de resolución de incidencias.

# 15. Anexos

### 15.1 Plantillas de Documentos

- Informe de Auditoría de Código
- Checklist de Pruebas de Usabilidad.

### 15.2 Registro de Cambios

| Versión | Fecha | Cambios | Responsable |
|---|---|---|---|
| **v1.0** | 25/02/2025 | Aprobación inicial del plan. | SQA Manager |