ASEGURAMIENTO DE CALIDAD
MODELO OPERATIVO DE ASEGURAMIENTO DE CALIDAD EN TI

Santander Consumer Bank

Lima, Julio 2025

Tabla de contenido

1. INTRODUCCIÓN

El modelo tiene como finalidad asegurar que todas las necesidades y expectativas planteadas por el usuario solicitante sean plenamente atendidas y satisfechas durante el ciclo completo de gestión operado por el equipo de TI. Para ello, es fundamental que se genere un entregable que incluya de manera clara y detallada las especificaciones funcionales y técnicas requeridas para el desarrollo de la solución. Estos documentos deben establecer criterios de aceptación precisos que sirvan de base al equipo de Aseguramiento de Calidad, permitiéndoles ejecutar actividades exhaustivas de validación y verificación sobre los productos que se entregan en cada fase del proyecto.

2. OBJETIVOS

Establecer y mantener un marco integral de aseguramiento de calidad que permita definir, comunicar y garantizar el cumplimiento de los requisitos de calidad en todos los procesos, procedimientos y resultados asociados al desarrollo de software. El proceso busca asegurar la entrega consistente de soluciones y servicios tecnológicos que satisfagan plenamente los requisitos de calidad y las necesidades de las partes interesadas, promoviendo la confianza y la satisfacción en los resultados obtenidos.

3. OBJETIVOS ESPECIFICOS

3.1. Establecer un Marco de Calidad

- Desarrollar y documentar los requisitos de calidad que serán aplicados, asegurando que dichos requisitos sean accesibles, comprensibles y alineados con las necesidades del negocio.

3.2. Implementar Controles de Calidad

- Diseñar e implementar mecanismos de control que permitan verificar el cumplimiento de los requisitos de calidad establecidos, asegurando la correcta ejecución de los procesos y la entrega de productos conforme a los estándares definidos.

3.3. Monitoreo y Evaluación Continua

- Realizar evaluaciones periódicas del cumplimiento de los estándares de calidad y de la eficiencia de los controles implementados, facilitando la detección de oportunidades de mejora y la gestión proactiva de riesgos.

3.4. Capacitación y Sensibilización

- Desarrollar y ejecutar programas de formación y concientización dirigidos al personal involucrado en los procesos de TI, promoviendo el entendimiento de los requisitos de calidad y la adopción de las mejores prácticas en sus actividades diarias.

3.5. Fomentar la Mejora Continua

- Establecer un proceso estructurado para la revisión y optimización continua de los procesos y procedimientos de calidad, apoyándose en métricas de desempeño y en el feedback de las partes interesadas para realizar ajustes que aseguren la evolución y eficiencia de las prácticas de calidad.

4. ALCANCE

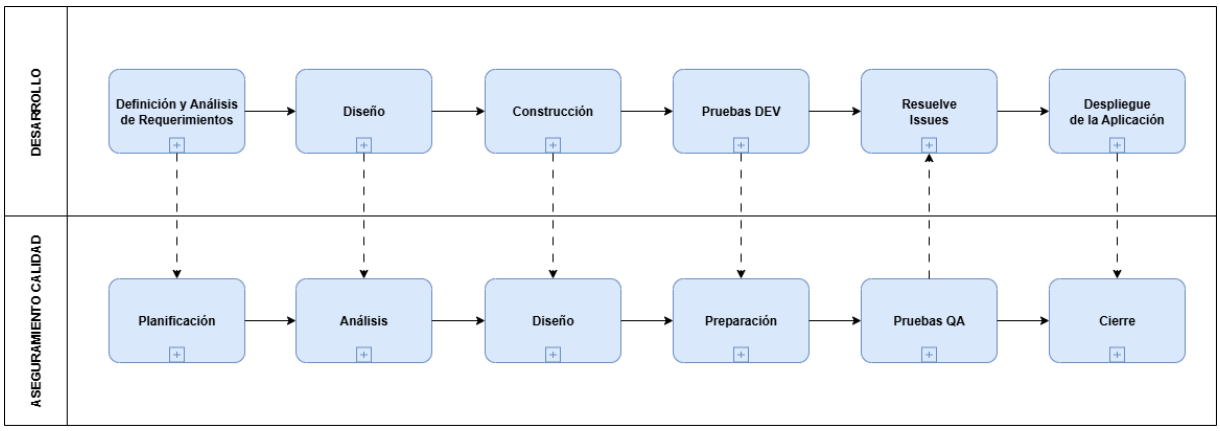
El modelo de aseguramiento de calidad en TI aplica a todas las actividades de desarrollo, mantenimiento y gestión de software, así como a proyectos de infraestructura tecnológica dentro de la organización. Incluye la definición de requisitos de calidad, implementación de controles, monitoreo y evaluación continua, capacitación al personal involucrado y acciones de mejora continua. El alcance se limita a los productos, servicios y entregables tecnológicos gestionados por el equipo de TI, con enfoque en satisfacer los requerimientos de calidad y las necesidades de las partes interesadas.

5. PROCESOS CLAVE

5.1. Establecer un proceso de aseguramiento de calidad

Establecer un proceso de aseguramiento de calidad, permite integrar actividades de QA en paralelo a cada etapa clave del desarrollo. Esto garantiza que la calidad no se revise únicamente al final, sino que sea gestionada y verificada de manera continua desde la planificación y el análisis de requerimientos hasta el cierre del proyecto.

Al hacerlo, se identifican y corrigen oportunamente posibles fallas o desviaciones, se mejora la comunicación entre equipos y se asegura que el producto final cumpla con los estándares y expectativas definidos por el negocio y los usuarios.



Proceso Desarrollo con Calidad incorporada – Adaptado para P. Helios

5.1.1. Planificación

En esta etapa se define la estrategia y el enfoque para el aseguramiento de calidad del proyecto. Se identifican los recursos necesarios, el alcance de las pruebas y se determinan los roles y responsabilidades.

- **Documentos de entrada:** Solicitud de aseguramiento de calidad, alcance del proyecto, cronograma.
- **Documentos de salida:** Documento de Estrategia de Pruebas / Test Plan Canvas
- **Matriz RACI:**

Actividad	PO	DEV	SCM	QA
Definición del alcance de la participación QA	A	C	C	R
Identificación de los objetivos de calidad	C	C	C	R/A
Levantamiento y análisis de requisitos de calidad	C	C	C	R
Estimación de recursos: tiempo, personal, herramientas y ambientes	A	C	C	R
Definición de la estrategia y tipos de pruebas a ejecutar	C	C	C	R/A
Establecimiento de la matriz de comunicación y canales	C	C	R	A
Programación de reuniones de seguimiento y puntos de control	I	I	R	C
Definición y documentación de los entregables de la etapa de QA	C	I	I	R/A
Revisión y aprobación del plan por las partes interesadas	A	C	C	R
Preparación del cronograma y calendario de actividades de QA	C	I	R	A

5.1.2. Análisis

Se revisan los requisitos funcionales y técnicos para comprender a fondo qué se espera del producto o solución. Se identifican los criterios de aceptación y los riesgos de calidad, asegurando una trazabilidad entre los requisitos y las pruebas.

- **Documentos de entrada:** Requerimientos del proyecto, Documento de Estrategia de Pruebas
- **Documentos de salida:** Matriz de trazabilidad / Plan de Pruebas.
- **Matriz RACI:**

Actividad	PO	DEV	SCM	QA
Recepción y revisión de los requisitos funcionales y técnicos	I	C	C	R
Identificación de criterios de aceptación de calidad	C	C	C	R/A
Análisis de la trazabilidad entre requisitos, el desarrollo y pruebas	C	C	C	R
Identificación y evaluación de riesgos de calidad	C	C	C	R/A
Validación de requisitos con Stalkeholders y equipos involucrados	A	C	C	R
Elaboración de la matriz de trazabilidad de requisitos, el desarrollo y pruebas	I	C	I	R/A
Definición de los parámetros y métricas de calidad a evaluar	C	C	C	R/A
Documentación de hallazgos y observaciones sobre los requisitos	I	I	I	R

5.1.3. Diseño

En esta fase se elaboran los casos de prueba y los escenarios que permitirán validar los requisitos definidos. Se especifica cómo y qué se va a probar, asegurando que todas las funcionalidades importantes estén cubiertas.

- **Documentos de entrada:** Diseño funcional y técnico, matriz de trazabilidad.
- **Documentos de salida:** Diseño de casos de prueba, escenarios de prueba.
- **Matriz RACI:**

Actividad	PO	DEV	SCM	QA
Revisión del diseño funcional y técnico del sistema	C	R	C	C
Identificación de elementos críticos para la calidad	C	C	C	R/A
Diseño de casos y escenarios de prueba	I	I	I	R/A
Definición de datos, precondiciones y ambientes para las pruebas	C	C	C	R
Validación del diseño de pruebas con Stalkeholders	A	C	C	R
Elaboración del plan de pruebas detallado	C	I	I	R/A
Documentación de los casos de prueba y escenarios	I	I	I	R
Comunicación del diseño de pruebas al equipo de desarrollo y Stalkeholders	A	I	R	C

5.1.4. Preparación

Consiste en configurar los ambientes y datos necesarios para ejecutar las pruebas. Se valida que todo esté listo para la fase de pruebas, minimizando posibles obstáculos técnicos durante la ejecución.

- **Documentos de entrada:** Plan de pruebas, casos de prueba, detalle de ambientes requeridos.
- **Documentos de salida:** Ambientes de prueba configurados, datos de prueba generados, validación de entorno.
- **Matriz RACI:**

Actividad	PO	DEV	SCM	QA
Configuración de ambientes de prueba	C	C	I	R/A
Generación y carga de datos de prueba	I	C	I	R
Verificación de la disponibilidad y acceso a los ambientes	C	C	I	R
Instalación y validación de herramientas necesarias para la ejecución	C	C	I	R/A
Confirmación de versiones y entregables a probar	C	R	C	A
Preparación de scripts y automatizaciones para pruebas	I	I	I	R
Validación de la documentación técnica y funcional necesaria	C	R	I	A
Comunicación de la disponibilidad de ambientes, datos y entregables	A	I	R	C

5.1.5. Pruebas QA

En esta etapa se ejecutan las pruebas de calidad sobre el producto o solución. Se documentan los resultados, se reportan los defectos encontrados y se generan informes que permiten tomar decisiones sobre la liberación del producto.

- **Documentos de entrada:** Casos de prueba, ambientes y datos configurados, versión entregada por desarrollo.
- **Documentos de salida:** Evidencias de pruebas, reporte de defectos, informe de resultados.
- **Matriz RACI:**

Actividad	PO	DEV	SCM	QA
Recepción de entregables listos para pruebas	I	R	I	A
Ejecución de casos y escenarios de prueba manuales y automáticos	I	I	I	R
Registro de resultados y evidencias de ejecución	I	I	I	R
Identificación y reporte de defectos/issues encontrados	I	I	I	R
Seguimiento y repruebas de issues corregidos	I	R	I	R
Comunicación de avances y hallazgos a los equipos involucrados	A	I	R	C
Análisis de resultados y elaboración de informe de pruebas	C	I	I	R/A
Validación del cumplimiento de criterios de aceptación	A	C	C	R
Retroalimentación al equipo de desarrollo y Stalkeholders	A	I	R	C

5.1.6. Cierre

Finalmente, se realiza la revisión final de los resultados de calidad y se documenta el cumplimiento de los requisitos. Se certifica que el producto está listo para producción y se recogen lecciones aprendidas para mejorar futuros procesos.

- **Documentos de entrada:** Informe de resultados, registro de issues resueltos, criterios de cierre.
- **Documentos de salida:** Acta de cierre de aseguramiento de calidad, lecciones aprendidas.
- **Matriz RACI:**

Actividad	PO	DEV	SCM	QA
Revisión final de resultados y entregables de QA	A	C	C	R
Verificación del cumplimiento de criterios de calidad y aceptación	A	C	C	R
Validación del cierre de issues y defectos encontrados	A	R	C	C
Elaboración del acta de cierre de QA	C	I	I	R/A
Certificación del cumplimiento de calidad	A	I	I	R
Documentación de lecciones aprendidas y recomendaciones	C	C	C	R
Presentación de resultados finales a Stalkeholders	A	I	R	C
Comunicación formal del cierre a los equipos involucrados	A	I	R	C

6. PRINCIPIOS Y BASES

Principio/Base	ISTQB (Testing)	COBIT APO11 (Gestión de Calidad)
Prevención antes que detección	Promueve revisión y pruebas tempranas	Recomienda controles y revisiones en todo el ciclo
Mejora continua	Análisis de resultados y retroalimentación	Establece procesos de mejora sistemática
Enfoque basado en riesgos	Pruebas y QA priorizados por riesgo	Prioriza controles y recursos según impacto de riesgos
Participación de todos los actores	QA como responsabilidad compartida	Involucra Stalkeholders y define roles claros
Orientación al cliente	Satisfacción del usuario es clave	Calidad alineada a necesidades y expectativas del cliente
Políticas, roles y responsabilidades	Define roles en el proceso de pruebas	Exige políticas y responsabilidades formales
Métricas e indicadores de calidad	Propone uso de métricas de defectos y cobertura	KPI y métricas para monitorear desempeño y conformidad
Uso de estándares y buenas prácticas	Basado en estándares internacionales	Impulsa el uso de marcos y normas reconocidas

7. ROLES Y RESPONSABILIDADES

7.1. QA Lead

Descripción del Rol

- Implementar estrategia de calidad corporativa para la banca
- Gestionar equipos QA distribuidos
- Diseñar frameworks de testing para arquitecturas complejas
- Interactuar con auditores y reguladores
- Optimizar presupuestos de testing (ROI)
- Implementar innovación en procesos de QA (AI testing)
- Alinear QA con objetivos de negocio

Perfil del Puesto

- Visión estratégica del testing bancario
- Capacidad para influir en niveles ejecutivos
- Experiencia en transformación digital
- Conocimiento profundo de riesgo operacional
- Liderazgo para el cambio organizacional

Experiencia Mínima

- 8+ años en QA (5+ en liderazgo)
- 5+ años en banca retail/corporativa
- Experiencia con migraciones legacy
- Gestión de presupuestos (+500k USD)

Conocimientos Avanzados

Obligatorios:

- Gobierno de calidad (Quality Engineering)
- Testing en arquitecturas cloud/microservicios
- Automatización avanzada (Self-healing tests)
- Modelos de madurez de calidad (TMMi)
- Risk-based testing estratégico

Deseables:

- Machine Learning aplicado a QA
- Blockchain testing
- IoT en banca digital

Formación Requerida

- Certificación ISTQB Expert/Manager
- Maestría en Calidad/Tecnología (deseable)
- Cursos ejecutivos en banca digital

7.2 Quality Engineer Automation

Descripción del Rol:

- Implementar la estrategia de automatización utilizando las mejores prácticas y estándares.
- Diseñar arquitecturas escalables para frameworks de pruebas (BDD, Page Object Model, etc.).
- Optimizar suites de pruebas para reducir tiempos de ejecución en pipelines CI/CD.
- Mentorizar a QA middle y colaborar con DevOps/Devs en la calidad del software.
- Participar en la toma de decisiones técnicas relacionadas con testing.

Perfil del Puesto:

- Liderazgo técnico para guiar al equipo en automatización.
- Visión estratégica para alinear pruebas con objetivos del negocio.
- Habilidad para resolver problemas complejos de rendimiento y escalabilidad.
- Experiencia en gestión de stakeholders (coordinación con áreas técnicas y de negocio).

Experiencia Mínima:

- 6+ años en QA, con al menos 4 años en automatización avanzada.
- Experiencia liderando iniciativas de automatización en proyectos grandes.

Conocimientos Específicos:

- **Lenguajes avanzados:** Dominio de Python/Java/JavaScript con enfoque en optimización.
- **Frameworks avanzados:** Selenium Grid, Robot Framework, Karate DSL.
- **Pruebas de rendimiento:** Artillery, JMeter, Gatling o Locust (deseable).
- **Cloud/DevOps:** Pruebas en AWS, Azure o entornos serverless.
- **Seguridad:** Conocimientos básicos de OWASP y pruebas de seguridad automatizadas.
- **Metodologías:** BDD (Cucumber, Behave), TDD y Shift-Left Testing.

Formación Deseable:

- Posgrado o certificaciones avanzadas en Calidad de Software (ISTQB Expert, Agile QA Leadership).
- Experiencia con microservicios y arquitecturas escalables.
- Conocimientos en observabilidad y monitoreo (ELK, Dynatrace, Grafana).

7.3. Quality Engineer Performance

Descripción del Rol:

- Asegura la calidad del software mediante la evaluación y optimización del rendimiento del sistema.
- Diseña, ejecuta y analiza pruebas de rendimiento.
- Identifica cuellos de botella en el sistema.
- Garantiza que las aplicaciones cumplan requisitos de rendimiento y escalabilidad.
- Posee profundo conocimiento en herramientas y técnicas de pruebas de performance.
- Interpreta resultados de pruebas y recomienda mejoras técnicas.

Perfil del Puesto:

- Experiencia en aseguramiento de calidad, con enfoque en pruebas de rendimiento del software (carga, estrés, estabilidad y escalabilidad).
- Conocimiento y manejo de herramientas de pruebas de performance como JMeter, Gatling, LoadRunner, BlazeMeter, etc.
- Habilidad para diseñar, implementar y ejecutar planes de pruebas de performance alineados a los objetivos de negocio y requisitos técnicos.
- Capacidad para analizar métricas de rendimiento, identificar cuellos de botella y generar reportes claros y accionables.

Experiencia Mínima:

- 6+ años en QA (3+ en performance avanzado)

Conocimientos Específicos:

- Diseño y ejecución de pruebas de carga y estrés.
- Manejo avanzado de herramientas como JMeter o Artillery.
- Scripting para automatización de escenarios de performance.
- Análisis e interpretación de métricas de rendimiento.
- Identificación de cuellos de botella y reporte de resultados claros.
- Integración de pruebas de performance en pipelines de CI/CD.
- Cloud testing (AWS/Azure)
- Seguridad aplicativa (OWASP)

Formación Deseable:

- Posgrado en Calidad de Software
- Certificaciones (ISTQB Expert, AWS Certified Tester)

7.4. QA Analyst Middle

Descripción del Rol:

- Diseñar y ejecutar estrategias de pruebas funcionales y no funcionales
- Automatizar pruebas con Selenium, Cypress o similares
- Participar en revisiones de requerimientos
- Optimizar procesos de QA

Perfil del Puesto:

- Pensamiento analítico para diseño de casos de prueba
- Autonomía en ejecución de proyectos
- Habilidad para comunicar problemas técnicos

Experiencia Mínima:

- 3-5 años en QA (1-2 en automatización)

Conocimientos Específicos:

- Automatización con Selenium/Playwright/Cypress
- SQL intermedio
- Pruebas de API (Postman, RestAssured)
- CI/CD básico (Jenkins, GitLab CI)

Formación Deseable:

- Ingeniería de Software, Sistemas o similar
- Certificaciones (ISTQB Advanced, Agile Testing)

8. HERRAMIENTAS Y ESTRATEGIAS

La adopción de herramientas y técnicas adecuadas es fundamental para soportar, automatizar y fortalecer las actividades del aseguramiento de calidad. Estas permiten mejorar la eficiencia, la trazabilidad y la objetividad en la gestión y evaluación de la calidad a lo largo del ciclo de vida del producto.

8.1. Herramientas

Gestión de Requerimientos	JIRA
Trazabilidad de requisitos, matrices de cobertura, revisiones colaborativas.	
Gestión de Pruebas	XRAY
Diseño de casos de prueba, gestión de planes de prueba, automatización de pruebas, ejecución y reporte de resultados.	
Pruebas Automatizadas Servicios	FRAMEWORK BDD - REQUESTS
Pruebas Automatizadas WEB	FRAMEWORK BDD - SELENIUM
Pruebas Automatizadas Móviles	FRAMEWORK BDD – APPIUM – SAUCELABS
Pruebas de Performance	JMETER
Pruebas funcionales automatizadas, integración continua (CI), pruebas de regresión automatizadas.	
Inspección de Código	SONARQUBE
Análisis estático de código, revisiones por pares, inspección de código, detección temprana de vulnerabilidades.	
Gestión de Defectos e Incidencias	XRAY
Registro y seguimiento de bugs, análisis de causa raíz, priorización y clasificación de incidencias.	
Documentación	SHAREPOINT / CONFLUENCE
Documentación viva, wikis, gestión de conocimiento compartido.	
Métricas y Reportes	POWER BI
Definición de KPIs, elaboración de reportes automáticos, tableros de control.	

8.2. Estrategias:

Base tomada de las buenas prácticas ágiles según ISTQB:

Fuente: Cuadrante Ágil Testing – Scrum.io

8.2.1. Estrategia para pruebas APIS:

- **Pruebas unitarias:** Validan lógica interna de cada servicio (DEV)
- **Pruebas de integración:** Verifican interacción entre servicios y con sistemas externos (core bancario, terceros) (DEV)
- **Pruebas de contrato:** Aseguran que los servicios cumplan el contrato esperado por los consumidores. (QA-AUTO)

  Ejemplo: Supongamos que tienes una API que ofrece datos de usuarios. El contrato define que la respuesta debe tener los campos id, name y email, todos obligatorios. Una prueba de contrato verifica que, ante una petición válida, el servicio responde exactamente con esos campos y tipos de datos, sin faltar ni sobrar información.
- **Pruebas de carga y rendimiento:** Simulan alto volumen de solicitudes, validando tiempos de respuesta y estabilidad. (QA-PERFORMANCE)
- **Pruebas de seguridad:** Fuzzing, validación de autenticación/autorización, análisis de vulnerabilidades como inyección, fuga de datos, ethical hacking (Ciberseguridad)
- **Pruebas de disponibilidad y resiliencia:** Simulan fallos, verifican recuperación y tolerancia. (QA-PRUEBAS NEGATIVAS AUTO - STRESS PERFORMANCE)

Validaciones: Estructura del response, códigos de estado HTTP, manejo de errores, tiempos de respuesta.

8.2.2. Estrategia para pruebas WEB/MOVILES:

- **Pruebas funcionales:** Validan cada funcionalidad del portal web (login, transferencias, consulta de saldos). (QA-end to end)
- **Pruebas Contables:** Verifican que los cálculos, registros y movimientos financieros del sistema coincidan con las reglas contables y los resultados esperados.
- **Pruebas de regresión:** Aseguran que los cambios no rompen funcionalidades existentes. (QA-Auto)
- **Pruebas de compatibilidad:** Verifican funcionamiento en distintos navegadores y versiones. (QA-AUTO)
- **Pruebas de seguridad:** XSS, CSRF, control de sesiones, gestión de contraseñas. (Ciberseguridad)
- **Pruebas de rendimiento:** Validan tiempos de carga y respuesta bajo diferentes condiciones. (QA-PERFORMANCE)

Validaciones: Navegación, inputs, mensajes de error, validaciones de negocio.

8.2.3. Estrategia para pruebas aplicaciones SAAS:

- **Pruebas unitarias:** Validan la lógica de los componentes internos y servicios, considerando la reutilización multi-tenant. (DEV).
- **Pruebas de integración:** Verifican la interacción entre módulos, servicios internos y sistemas de terceros (APIs externas, pasarelas de pago, integraciones de clientes). (DEV).
- **Pruebas de contrato:** Aseguran que las APIs públicas y los endpoints expuestos respeten contratos y versiones para todos los clientes. (QA).
- **Pruebas de carga y escalabilidad:** Simulan múltiples clientes/tenants y usuarios concurrentes, validando elasticidad, partición de recursos y SLA (tiempos de respuesta bajo demanda variable). (QA).
- **Pruebas de configuración y personalización:** Validan que cada tenant pueda configurar aspectos del sistema (branding, reglas de negocio, permisos) sin afectar a otros. (QA).
- **Pruebas de Regresión:** Verifican que los despliegues de nuevas versiones no afectan la disponibilidad ni los datos de los clientes activos. (QA).
- **Pruebas de seguridad:** Control de acceso multi-tenant, aislamiento de datos, validación de autenticación/autorización, cumplimiento normativo (GDPR, SOC2), pruebas de vulnerabilidad y ethical hacking. (Ciberseguridad)
- **Pruebas de disponibilidad y resiliencia:** Simulan fallos de infraestructura, interrupciones de servicio, verifican recuperación automática, failover, backups y restauración de datos. (QA).

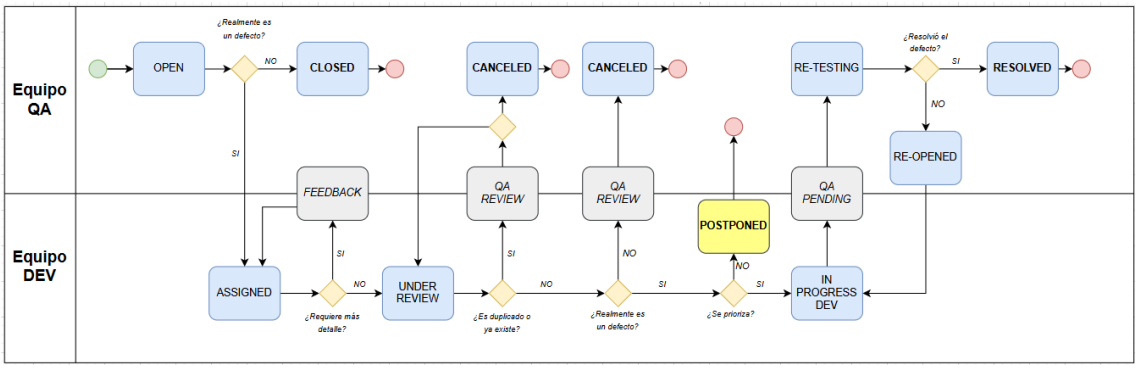
Validaciones adicionales:

- Aislamiento de datos entre tenants
- Cumplimiento de niveles de servicio (SLA)
- Adaptabilidad ante cambios de escala
- Soporte para personalizaciones/configuraciones independientes

- UX consistente para diferentes perfiles de usuario
- Integridad y consistencia de datos a través de actualizaciones

9. GESTIÓN DE DEFECTOS

La gestión de defectos es un proceso fundamental en el aseguramiento de calidad de software, pues permite asegurar que los productos entregados cumplen con los estándares requeridos y satisfacen las necesidades del cliente.



Transición de estados de Defectos – Adaptado para P. Helios

La gestión de defectos es el conjunto de actividades orientadas a identificar, registrar, priorizar, analizar, asignar, resolver y cerrar los defectos detectados durante las fases de desarrollo y pruebas de un proyecto de software. Este proceso involucra tanto al equipo de QA como al equipo de desarrollo, promoviendo la colaboración y la mejora continua.

9.1. Flujo de Gestión de Defectos

El ciclo de vida de un defecto se representa mediante un flujo de estados y decisiones, donde cada transición está condicionada por la verificación, análisis y resolución del defecto. El flujo puede dividirse en dos grandes responsables: el equipo de QA y el equipo de Desarrollo (DEV).

9.2. Estados y Transiciones

Equipo QA

- **OPEN:** El defecto es reportado y registrado por el equipo de QA. Se inicia el proceso de análisis.
    - **¿Realmente es un defecto?**
        - **NO:** El defecto se cierra (CLOSED).
        - **SÍ:** Se transfiere al equipo de desarrollo (ASSIGNED).

Equipo DEV

- **ASSIGNED:** El defecto es asignado a un desarrollador para su análisis.
    - **¿Requiere más detalle?**
        - **SÍ:** Se solicita retroalimentación al equipo QA (FEEDBACK).
        - **NO:** Se inicia el análisis del defecto (UNDER REVIEW).

- **UNDER REVIEW:** El desarrollador revisa el defecto.
    - **¿Es duplicado o ya existe?**
        - **SÍ:** El defecto se cancela (CANCELED).
        - **NO:** Continúa la revisión por QA (QA REVIEW).
- **QA REVIEW:** El equipo QA evalúa el estado del defecto.
    - **¿Realmente es un defecto?**
        - **NO:** Se cancela (CANCELED).
        - **SÍ:** Se evalúa la prioridad.
    - **¿Se prioriza?**
        - **NO:** El defecto se pospone (POSTPONED).
        - **SÍ:** El equipo de desarrollo trabaja en la resolución (IN PROGRESS DEV).
- **POSTPONED:** El defecto queda en espera hasta que se le asigne prioridad.
- **IN PROGRESS DEV:** El defecto está siendo solucionado por el equipo de desarrollo.
- **QA PENDING:** El defecto espera la validación por el equipo QA tras la corrección.

Equipo QA

- **RE-TESTING:** Se realiza la verificación para comprobar si el defecto fue resuelto.
    - **¿Se resolvió el defecto?**
        - **SÍ:** Se considera resuelto (RESOLVED).
        - **NO:** Se reabre el defecto (RE-OPENED).
- **RE-OPENED:** El defecto se reabre y regresa a estado de corrección por el equipo de desarrollo.

9.3. Estados Finales

- **CLOSED:** El defecto no procede por no ser válido.
- **CANCELED:** El defecto es duplicado o no aplicable.
- **POSTPONED:** El defecto queda en espera por baja prioridad.
- **RESOLVED:** El defecto ha sido corregido y validado.

9.4. Involucrados

- **Equipo QA:**
    - Identificar y reportar defectos con información clara y suficiente.
    - Validar la corrección de los defectos.
    - Colaborar en la retroalimentación y clarificación de los reportes.
- **Equipo DEV:**
    - Analizar y resolver los defectos asignados.
    - Solicitar información adicional cuando sea necesario.
    - Comunicar el estado y resultados de las correcciones.

9.5. Decisiones y Criterios

- **Validez del defecto:** Determinar si el incidente reportado corresponde a un defecto real del producto.
- **Duplicidad:** Verificar si el defecto ya ha sido reportado previamente.
- **Prioridad:** Evaluar la urgencia y el impacto del defecto para decidir su resolución inmediata o su postergación.
- **Cierre:** Cerrar el defecto solo cuando esté completamente validada su resolución.

9.6. Herramienta y Registro de Defectos

Todos los defectos deben ser registrados en una herramienta de gestión (por ejemplo, Jira, XRAY), con los siguientes datos mínimos:

- Identificador único
- Descripción detallada
- Estado actual
- Responsable asignado
- Prioridad
- Historial de cambios y comentarios

10. ARTEFACTOS DE CERTIFICACIÓN

10.1. Plan de pruebas detallado

- **Introducción**: El Plan de pruebas describe el alcance, el enfoque, los recursos y el cronograma de todas las actividades de prueba. Su objetivo es garantizar que el software cumpla con los requisitos técnicos, funcionales y comerciales.
- **Alcance**: El alcance de este plan de prueba incluye pruebas funcionales, pruebas no funcionales (performance, portabilidad, compatibilidad, usabilidad) y pruebas de aceptación de usuario (UAT)
- **Objetivo de las Pruebas**: Asegurar que la aplicación bajo prueba cumpla con los requisitos funcionales y no funcionales, y que los errores se identifiquen y solucionen antes de su lanzamiento.
- **Metodología de Pruebas**: La metodología implica la realización de los tres niveles de Prueba: Unitarias, integración, sistema, aceptación.
- **Clasificación de Errores**: Críticos, mayores, menores.
- **Criterios de Suspensión y Reanudación**: Definidos según la gravedad de los errores.
- **Roles**: QA, Tester, Especialista Infraestructura, Desarrolladores, Especialista de Seguridad.

10.2. Matriz de Trazabilidad

- Objetivo:
  Este documento asegura que todos los requisitos definidos para un sistema o software estén correctamente implementados, verificados y validados. Permite:

- Hacer seguimiento al cumplimiento de los requisitos a lo largo del ciclo de vida del proyecto.
- Garantizar que cada requisito tenga cobertura en el desarrollo, pruebas y validación.
- Permitir la identificación rápida de brechas, cambios y el estado de cada requisito.
- Facilitar la comunicación entre las áreas involucradas (negocio, desarrollo, QA, etc.).

- Metodología de aplicación:

  - **Identificación de Requisitos:** Cada requisito funcional o no funcional se documenta con un identificador único (por ejemplo, REQ-001).
  - **Registro en la Matriz:** Se ingresa cada requisito en la matriz, asociándolo con su descripción, necesidad comercial y el componente del sistema donde se implementará.
  - **Seguimiento del Ciclo de Vida:** Se documenta el resultado de la implementación (por ejemplo, versión en la que se implementó), los casos de prueba asociados y el estado actual (completado, en progreso, pendiente).
  - **Vinculación con Casos de Prueba:** Se asocian los requisitos con los casos de prueba específicos que validarán su cumplimiento (por ejemplo, TC-001, TC-002).
  - **Actualización Continua:** La matriz se actualiza a medida que avanza el proyecto, reflejando cambios en los requisitos, implementación y resultados de pruebas.
  - **Revisión y Auditoría:** Se utiliza la matriz para auditorías internas o externas, asegurando la trazabilidad desde el requisito hasta su validación final.

10.3. Matriz de Casos de Prueba

- Objetivo:

  El objetivo principal de una matriz de casos de prueba es organizar, documentar y gestionar todas las pruebas que se realizarán sobre un sistema o aplicación. Sus propósitos específicos son:
  - Asegurar la cobertura de los requisitos y funcionalidades a través de casos de prueba detallados.
  - Facilitar el seguimiento y control de la ejecución de las pruebas.
  - Servir como evidencia de validación del sistema ante auditorías o revisiones.
  - Identificar fácilmente el estado de cada caso de prueba (pasó, falló, pendiente).
  - Asignar responsabilidades y priorizar esfuerzos de pruebas.

- Metodología para la aplicación:

  - **Identificación y Registro de Casos de Prueba:** Cada caso de prueba se documenta con un identificador único (por ejemplo, TC-001), asociado a una funcionalidad y una descripción clara del objetivo de la prueba.

- **Definición de Precondiciones:** Se especifican las condiciones que deben cumplirse antes de ejecutar el caso de prueba (por ejemplo, “usuario existe”).
- **Pasos a Seguir:** Se detallan los pasos secuenciales que debe realizar el tester para ejecutar el caso de prueba, usando el formato Given-When-Then o una lista numerada.

- **Datos de Prueba:** Se definen los datos específicos a utilizar durante la ejecución (usuarios, contraseñas, correos, etc.).

- **Resultado Esperado:** Se documenta el resultado esperado para poder comparar con el resultado real y determinar si el caso pasa o falla.

- **Ejecución y Seguimiento:** Durante la ejecución del testing, se llenan las columnas de resultado real, estado (Pass/Fail), severidad, prioridad y responsable, además de la fecha de ejecución y observaciones.

- **Control y Gestión:** La matriz se utiliza para revisar el avance de la ejecución de pruebas, identificar casos críticos o bloqueados, y tomar acciones correctivas si es necesario.

10.4. Reporte consolidado de Defectos

- Objetivo:
  El objetivo principal de un reporte de defectos es registrar, organizar y dar seguimiento a los errores encontrados durante el ciclo de pruebas de software. Permite:
  - Identificar y describir de forma clara los defectos detectados.
  - Priorizar la resolución de los defectos según su severidad e impacto.
  - Asignar responsables para su análisis y corrección.
  - Hacer seguimiento al estado y ciclo de vida de cada defecto (abierto, cerrado, pendiente).
  - Facilitar la toma de decisiones y la mejora continua del proceso de calidad.

- Metodología para la aplicación:

  - **Registro del Defecto:** Cuando se detecta un defecto, se documenta inmediatamente en el reporte.
  - **Asignación de responsable:** Se designa a una persona responsable de analizar y corregir el defecto.
  - **Seguimiento del Estado:** Se actualiza el estado del defecto conforme avanza su resolución.
  - **Gestión de Fechas:** Se registra la fecha de detección y, cuando corresponde, la fecha de cierre del defecto.
  - **Observaciones y Notas:** Se añaden observaciones relevantes, bloqueos, o cualquier información adicional útil para el seguimiento.

- **Revisión Periódica:** El reporte se revisa en reuniones de seguimiento para asegurar que los defectos críticos se atiendan oportunamente y que no queden casos sin resolver.

10.5. Acta de Cierre

- Objetivo:
  Documento que certifica la finalización del proceso de pruebas de calidad en un proyecto de software. Su objetivo es dejar constancia de que se han ejecutado las pruebas planificadas, documentar los resultados, registrar los defectos. También suele ser requerido para liberar el software a producción.

- Metodología para la aplicación:

  - **Consolidación de Información y Resultados:** Se recopilan todos los resultados de ejecución de pruebas, matrices de casos de prueba, reportes de defectos y documentación relevante.

  - **Revisión y Verificación de Cobertura:** Valida que todos los requisitos y funcionalidades planificadas hayan sido cubiertos por los casos de prueba. Verifica que los defectos críticos hayan sido resueltos o documentados.

  - **Documentación del Acta:** Se llena el acta de cierre con la siguiente información:

    - Datos generales del proyecto, versión, responsables y fechas.
    - Alcance de pruebas y módulos cubiertos.
    - Resumen cuantitativo de la ejecución: Nº de casos ejecutados, aprobados, fallidos, pendientes.
    - Estado de defectos: Nº de defectos reportados, abiertos, cerrados, críticos, etc.
    - Riesgos, observaciones y recomendaciones.
    - Conclusión y declaración de cierre.

11. INDICADORES

11.1. KPI: Porcentaje de Eficiencia en la Revisión de Casos de Prueba

- **Fórmula:**

% Eficacia = (Número de casos de prueba aprobados en la última revisión / Número total de casos de prueba diseñados) x 100

- **Detalles:**
  - **Número de casos de prueba aprobados en la última revisión:**
    Corresponde a los casos de prueba que, tras una revisión formal (por parte de

QA, líder técnico o usuario), cumplen plenamente con los criterios de aceptación y estándares de calidad establecidos.
    - **Número total de casos de prueba diseñados:**
      Es la totalidad de los casos de prueba planificados y documentados para una funcionalidad, sprint o ciclo de pruebas.

- **Descripción:**
  El **Porcentaje de Eficacia en las Revisiones de Casos de Prueba** es un indicador clave de desempeño que mide la calidad de los casos de prueba diseñados durante el proceso de aseguramiento de calidad. Este KPI refleja el grado en que los casos de prueba diseñados cumplen con los criterios de aceptación, estándares de calidad y cobertura funcional definida, según lo validado en la última revisión formal del ciclo de pruebas.

- **¿Qué Mide esta Indicador?**
    - **Calidad del diseño de pruebas:** Evalúa qué tan efectivos y correctos son los casos de prueba diseñados antes de su ejecución.
    - **Nivel de preparación:** Permite identificar si los casos de prueba están listos y alineados con los requerimientos funcionales y no funcionales.
    - **Madurez del proceso de pruebas:** Un porcentaje alto indica que el proceso de diseño de pruebas es robusto y maduro, mientras que un valor bajo puede señalar la necesidad de mejorar la capacitación, la documentación o la comprensión de los requisitos.

- **¿Por qué es importante?**
    - Permite asegurar que los casos de prueba sean relevantes, completos y correctamente diseñados antes de ser ejecutados.
    - Reduce el riesgo de encontrar defectos de prueba (casos incorrectos, ambiguos o incompletos) durante la ejecución.
    - Ayuda a optimizar los recursos y el tiempo destinado a las actividades de QA.
    - Favorece la mejora continua del proceso de diseño de pruebas.

- **Buenas Prácticas:**
    - Definir criterios de aprobación claros y objetivos para las revisiones.
    - Realizar revisiones periódicas y colaborativas de los casos de prueba, involucrando a los principales Stalkeholders.
    - Documentar las causas por las que un caso de prueba no fue aprobado para alimentar la mejora continua.

11.2. KPI: Porcentaje de Defectos Descubiertos

- **Fórmula:**

% Casos Descubiertos = (Número de casos de prueba impactados por defectos en el primer ciclo / Número total de casos de prueba) x 100

- **Detalles:**

- **Número de casos de prueba impactados por defectos en el primer ciclo:**
  Es la cantidad de casos de prueba que, al ejecutarse por primera vez, han detectado al menos un defecto. Se cuenta cada caso de prueba solo una vez, aunque haya encontrado múltiples defectos.

- **Número total de casos de prueba:**
  Es la totalidad de los casos de prueba ejecutados en el primer ciclo.

- **Descripción:**
  El **Porcentaje de Defectos Descubiertos** es un KPI que indica la proporción de casos de prueba que han detectado al menos un defecto durante el primer ciclo de pruebas. Este indicador ayuda a evaluar la efectividad de los casos de prueba para identificar errores tempranamente en el ciclo de validación.

- **¿Qué Mide esta Indicador?**
  - **Eficacia de los casos de prueba:** Mide la capacidad de los casos de prueba para encontrar defectos reales en el software durante la primera ejecución.
  - **Calidad del software entregado:** Un porcentaje alto puede indicar que existen varios defectos en la versión entregada, mientras que un porcentaje bajo podría significar que el producto es más robusto o que los casos de prueba no están detectando suficientes problemas.
  - **Madurez del proceso de pruebas:** Permite ajustar la estrategia de pruebas, el diseño de casos y la cobertura según los resultados obtenidos.

- **¿Por qué es importante?**
  - Permite identificar la efectividad inicial del ciclo de pruebas y la calidad del producto recibido para pruebas.
  - Ayuda a determinar si la estrategia de pruebas y el diseño de casos son adecuados para detectar fallas relevantes desde el inicio.
  - Es útil para evaluar la tendencia de la calidad en distintas iteraciones o versiones, apoyando la mejora continua.
- **Buenas Prácticas:**
  - Asegurarse de que los casos de prueba estén bien diseñados y alineados con los requerimientos.
  - Analizar las causas de defectos frecuentes para retroalimentar el proceso de desarrollo y pruebas.
  - Monitorear este KPI en cada ciclo para identificar tendencias y oportunidades de mejora.

11.3. KPI: Porcentaje de Desviación en Tiempos de Entrega

- **Fórmula:**

% Desviación de Tiempo = ((Fecha de cierre real - fecha de cierre programada) / fecha de cierre programada) X 100

- **Detalles:**

- **Fecha de cierre real:**
  Es la fecha en la que realmente se finalizó la tarea, proyecto o entrega.

- **Fecha de cierre programada:**
  Es la fecha que se había comprometido o planificado inicialmente para la finalización.

- **Cálculo:**
  La diferencia entre la fecha real y la programada se divide entre la fecha programada, multiplicando el resultado por 100 para obtener el porcentaje de desviación.

  - Si el resultado es positivo, indica retraso (la entrega se completó después de lo programado).
  - Si es negativo, indica adelanto (la entrega se completó antes de lo programado).
  - Si es cero, la entrega fue justo a tiempo.

- **Descripción:**
  El **Porcentaje de Desviación en Tiempos de Entrega** es un indicador clave que mide la diferencia porcentual entre la fecha en que realmente se concluyó un proyecto, tarea o entrega, y la fecha que originalmente se había planificado. Este KPI permite identificar la capacidad del equipo o la organización para cumplir con los compromisos de tiempo establecidos, proporcionando visibilidad sobre retrasos o adelantos en la ejecución de actividades.

- **¿Qué Mide esta Indicador?**
  - **Cumplimiento de plazos:** Evalúa la exactitud y disciplina en la gestión del tiempo, mostrando si las entregas ocurren según lo planificado.
  - **Eficiencia en la gestión de proyectos:** Un alto porcentaje de desviación puede indicar problemas en la planificación, estimación, gestión de riesgos o ejecución.
  - **Capacidad de respuesta:** Permite identificar tendencias de retrasos o adelantos, facilitando la toma de decisiones y ajustes en la gestión de futuras entregas.

- **¿Por qué es importante?**
  - Ayuda a controlar y mejorar la gestión del tiempo en los proyectos.
  - Permite identificar proyectos o equipos que consistentemente presentan desviaciones, facilitando la implementación de acciones correctivas.
  - Aporta información valiosa para la mejora continua en la estimación y planificación de futuros proyectos.
  - Es útil para reportes de gestión y comunicación con Stalkeholders.
- **Buenas Prácticas:**
  - Usar unidades de tiempo consistentes (días, horas, semanas).

- Analizar las causas de las desviaciones para implementar acciones preventivas o correctivas.
- Monitorear este KPI periódicamente para identificar tendencias y mejorar la capacidad de planificación.
- Comunicar los resultados a los equipos y partes interesadas para fomentar la responsabilidad y la mejora continua.

11.4. KPI: Porcentaje de Cobertura de Pruebas Automatizadas

- **Fórmula:**

% Cobertura Automatizada = (Número de pruebas automatizadas / Número total de pruebas) x 100

- **Detalles:**
  - **Número de pruebas automatizadas:**
    Es la cantidad de casos de prueba que pueden ejecutarse automáticamente mediante frameworks, scripts o herramientas de automatización.
  - **Número total de pruebas:**
    Incluye tanto las pruebas manuales como las automatizadas diseñadas para cubrir los requerimientos funcionales y no funcionales en un periodo de tiempo determinado (sprint, release, etc.).
- **Descripción:**
  El **Porcentaje de Cobertura de Pruebas Automatizadas** es un indicador clave que mide la proporción de los casos de prueba gestionados y ejecutados de manera automática respecto al total de pruebas definidas para un sistema, producto, módulo o proyecto. Este KPI refleja el grado de automatización en el proceso de pruebas, mostrando hasta qué punto la validación del software se apoya en herramientas automáticas frente a la ejecución manual.
- **¿Qué Mide esta Indicador?**
  - **Nivel de automatización:** Indica el porcentaje de pruebas que están automatizadas, facilitando la identificación de áreas que aún requieren intervención manual.
  - **Optimización de recursos y tiempos:** Un alto porcentaje suele asociarse a procesos de QA más eficientes, reducción de tiempos de ciclo y mayor capacidad de ejecución repetitiva.
  - **Madurez del proceso de pruebas:** Equipos con alta cobertura automatizada suelen tener procesos más robustos, repetibles y menos sujetos a errores humanos.
- **¿Por qué es importante?**
  - Permite monitorear el avance de la automatización y priorizar esfuerzos en la cobertura de áreas críticas del sistema.
  - Ayuda a justificar inversiones en herramientas, capacitación y mantenimiento de pruebas automatizadas.

  - Reduce el riesgo de regresión y mejora la confiabilidad de los resultados, ya que las pruebas automatizadas pueden ejecutarse con mayor frecuencia y rapidez.
  - Es fundamental para entornos DevOps y CI/CD, donde la integración y entrega continua requieren altos niveles de automatización para mantener la calidad.
- **Buenas Prácticas:**
  - Seleccionar para automatización aquellas pruebas que sean repetitivas, de alto impacto y estables, priorizando regresión, pruebas críticas y flujos básicos del sistema.
  - Mantener actualizados y bien documentados los scripts y marcos de automatización.
  - Revisar periódicamente la cobertura automatizada para identificar obsolescencia y nuevas oportunidades de automatización.

12. ANEXOS

- Documento de Plan de Pruebas
- Matriz de Trazabilidad
- Matriz de Casos y Escenarios de Prueba (*mejora registro en XRAY*)
- Documento de Reporte Consolidado de Defectos
- Documento de Acta de Cierre de Pruebas y Lecciones Aprendidas.

13. CONCLUSIONES

Implementar y mantener un modelo de aseguramiento de calidad robusto no solo contribuye a reducir errores, mejorar la satisfacción del cliente y optimizar recursos, sino que también posiciona al equipo de TI como un socio estratégico para el negocio, capaz de responder ágilmente a los desafíos del mercado y de garantizar el éxito sostenido de las iniciativas tecnológicas.

“La calidad no es un acto, es un hábito.” – Aristóteles