
Unidad 9: QA Ágil y DevOps

Asignatura: IDS325 – Aseguramiento de la Calidad del Software
Integrantes:
– Pedro Encarnación – 1121181
– Brady Rodríguez – 1124979
– Ángel Concepción – 1124530
Fecha: 29 de junio de 2025

Introducción: Rol del QA en Entornos Ágiles y DevOps

En los entornos ágiles y DevOps, el aseguramiento de la calidad (QA) deja de ser una fase final y se convierte en un proceso continuo e integrado en todo el ciclo de vida del desarrollo. El enfoque se centra en la colaboración constante entre desarrolladores (Dev), testers (QA), operaciones (Ops) y el Product Owner (PO), con el objetivo de entregar software confiable, funcional y de alto valor al usuario. La integración de estrategias como **shift-left** y **shift-right** permite detectar y corregir errores en fases tempranas, y al mismo tiempo, mantener una supervisión activa en producción, logrando un sistema resiliente y de mejora continua.

Entregable 1: Shift-Left en Ciclos Ágiles

Planificación Colaborativa

User Story seleccionada:

Como usuario, quiero filtrar productos por categoría para visualizar solo los artículos de mi interés.

Sesión de planificación colaborativa:
Participaron los roles de **Dev**, **QA** y **PO** para definir los criterios de aceptación y planificar las pruebas tempranas antes del desarrollo. Se utilizó la metodología **BDD (Behavior Driven Development)** para asegurar una comprensión común del comportamiento esperado.

Criterios de Aceptación (Formato Given-When-Then)

Escenario	Given (Dado que)	When (Cuando)	Then (Entonces)
Filtrado exitoso por categoría	Dado que el usuario se encuentra en la página de productos	Cuando selecciona una categoría en el filtro	Entonces el sistema muestra solo los productos correspondientes a esa categoría
Filtro sin resultados	Dado que el usuario selecciona una categoría sin productos asociados	Cuando aplica el filtro	Entonces el sistema muestra un mensaje de “No hay productos disponibles”
Restablecer filtros	Dado que el usuario aplicó un filtro	Cuando selecciona la opción “Restablecer”	Entonces el sistema muestra todos los productos nuevamente

---

Diseño Shift-Left y Automatización

Antes de implementar la funcionalidad, se diseñaron las pruebas unitarias y de integración usando el enfoque **TDD (Test Driven Development)**.
Se elaboró una prueba automatizada con **pytest** que verifica que el filtrado devuelva resultados coherentes con la categoría seleccionada.

Ejemplo de prueba automatizada:

```
def test_filtrar_por_categoria(api_client):
  response = api_client.get('/productos?categoria=Tecnologia')
  assert response.status_code == 200
  for producto in response.json():
    assert producto['categoria'] == 'Tecnologia'
```

Reflexión: Impacto del Shift-Left

Aplicar la estrategia **shift-left** reduce los costos de corrección de defectos al detectar errores antes del desarrollo o despliegue. Esto permite prevenir fallos lógicos, inconsistencias de datos y retrabajos costosos. La colaboración temprana

mejora la comunicación entre roles, acelera los ciclos de entrega y aumenta la confianza en la calidad del producto desde el inicio.

Entregable 2: Shift-Right y Pipeline DevOps

Configuración Shift-Right

Se simuló el monitoreo del entorno de **staging** utilizando **Sentry**, una herramienta de rastreo de errores en producción.
Durante la simulación, se provocó un error controlado ("NullReferenceException") en el módulo de filtrado. Sentry generó una alerta automática, registrando el tipo de excepción, la ruta afectada y la traza del stack, lo que permitió al equipo responder rápidamente.

Evidencia simulada:

- Notificación automática a canal DevOps (Slack o correo).
- Registro del error con timestamp y usuario afectado.
- Acción correctiva inmediata del equipo QA/Dev.

Pipeline DevOps Completo

El pipeline se extendió con etapas de **CI/CD** que incluyen pruebas, despliegue, monitoreo y retroalimentación:

Diagrama de Pipeline QA-DevOps

Commit → Build → Pruebas Unitarias → Integración → Despliegue en Staging →

Health Check → Pruebas de Humo → Monitoreo (Sentry/Prometheus) → Retroalimentación

Componentes clave:

- **Shift-Left:** Validación automática de pruebas unitarias y revisión de código en cada commit.
- **Shift-Right:** Monitoreo continuo del entorno de producción, alertas automáticas y métricas de rendimiento.

Reflexión: Mejora de la Resiliencia del Sistema

El enfoque **shift-right** permite identificar fallos en escenarios reales de uso, proporcionando información valiosa sobre el rendimiento y estabilidad del sistema.
Al integrar monitoreo, health-checks y alertas automáticas, el equipo puede reaccionar proactivamente ante incidentes, reduciendo el tiempo medio de resolución (MTTR) y fortaleciendo la **resiliencia del sistema** frente a cambios y fallos imprevistos.

Conclusiones

La combinación de **shift-left** y **shift-right** consolida la calidad como un proceso transversal y continuo dentro del ciclo DevOps.
Mientras el shift-left permite **prevenir defectos desde las fases iniciales**, el shift-right facilita **detectar, analizar y aprender de los errores en producción**.
Ambas estrategias, sumadas a la colaboración entre **Dev, QA, Ops y Product Owner**, garantizan un producto más estable, predecible y alineado con las necesidades reales del usuario final.
La cultura de mejora continua se convierte en el pilar fundamental del aseguramiento de la calidad en entornos modernos.
