**RETO 2**

Para asegurar la calidad en todos los ambientes (desarrollo, QA, UAT, producción), es fundamental adaptar los tipos de pruebas y enfoques para cada uno, ya que cada entorno tiene un propósito y desafíos diferentes. A continuación, te propongo cómo organizar las pruebas en cada uno de estos ambientes:

1. **Ambiente de Desarrollo (Development)**

**Objetivo:** Probar funcionalidades en su etapa inicial, mientras están en desarrollo.

**Tipos de pruebas:**

* **Pruebas unitarias:** Validar que cada componente individual funciona como se espera. Deben ser rápidas y específicas para detectar errores en el código.
* **Pruebas de integración iniciales:** Asegurarse de que los módulos trabajen correctamente juntos.
* **Pruebas estáticas:** Revisión de código, análisis estático y validación de estilo para asegurar la calidad del código.

**Herramientas:**

* PyTest para unitarias.

1. **Ambiente de QA (Quality Assurance)**

**Objetivo:** Validar que las funcionalidades cumplen con los requerimientos definidos antes de pasar a UAT.

**Tipos de pruebas:**

* **Pruebas funcionales:** Asegurarse de que las funcionalidades respondan a los requerimientos establecidos.
* **Pruebas de regresión:** Verificar que las nuevas funcionalidades no afecten las existentes.
* **Pruebas de API:** Validar los servicios backend si los tienes disponibles en esta etapa.
* **Pruebas automatizadas de UI:** Ejecutar los scripts de Selenium para validar la interfaz de usuario en diferentes navegadores y resoluciones.

**Herramientas:**

* Selenium + PyTest para pruebas funcionales de UI.
* Postman o RestAssured para pruebas de API.
* Jenkins para integrar las pruebas dentro de un pipeline de CI/CD.

1. **Ambiente de UAT (User Acceptance Testing)**

**Objetivo:** Validar que la aplicación cumple con las expectativas del cliente o usuario final antes del lanzamiento.

**Tipos de pruebas:**

* **Pruebas de aceptación:** Asegurarse de que las historias de usuario se implementaron correctamente desde la perspectiva del usuario.
* **Pruebas de flujo end-to-end:** Simular escenarios completos como lo haría el usuario final.
* **Pruebas de usabilidad:** Evaluar la facilidad de uso y experiencia del usuario.
* **Pruebas exploratorias:** Permitir al equipo de QA o a usuarios reales encontrar problemas no anticipados.

**Herramientas:**

* Selenium o Cypress para pruebas end-to-end.
* Herramientas como JIRA para el seguimiento de bugs y requerimientos encontrados en esta etapa.

1. **Ambiente de Producción (Production)**

**Objetivo:** Asegurar que el producto funcione sin errores en el entorno en vivo.

**Tipos de pruebas:**

**Pruebas de monitoreo:** Configurar scripts automatizados que verifiquen la disponibilidad y el rendimiento de los servicios en producción.

**Pruebas de smoke:** Ejecutar una pequeña batería de pruebas básicas para garantizar que la aplicación esté funcionando correctamente.

**Pruebas de rendimiento:** Evaluar la carga y el rendimiento, asegurándose de que la aplicación pueda soportar picos de usuarios.

**Herramientas:**

* Herramientas de monitoreo como New Relic, Datadog o Grafana para métricas y logs.
* Locust o JMeter para pruebas de carga y rendimiento.

**Consideraciones adicionales**

**Gestión de datos de prueba:**

* Es importante tener datasets o mockups de datos apropiados para cada entorno. Evita usar datos reales en ambientes de desarrollo o QA.

**Configuraciones de ambiente:**

En cada prueba automatizada o script, puedes manejar diferentes configuraciones (URLs, credenciales, etc.) según el ambiente en el que estés ejecutando las pruebas. Esto puede manejarse con un archivo de configuración o variables de entorno.