**SOLUCIÓN PUNTO 5**

**Proceso de desarrollo de software utilizando metodologías tradicionales y ágiles:**

* Metodologías Tradicionales (Cascada): Enfoque secuencial donde cada fase (requerimientos, diseño, implementación, pruebas, despliegue y mantenimiento) se realiza de forma lineal. Es adecuado cuando los requisitos son claros y no cambian, pero es inflexible ante modificaciones.
* Metodologías Ágiles: Se basan en iteraciones cortas (sprints) y la entrega continua de valor. Priorizan la flexibilidad, la colaboración y la adaptación a cambios. Las entregas son incrementales, y el trabajo se revisa y ajusta continuamente en base al feedback.

**Rol del analista de calidad en un entorno ágil:**

El analista de calidad participa activamente en todas las fases del desarrollo ágil. Sus responsabilidades incluyen la planificación de pruebas, la automatización de pruebas, la ejecución continua de pruebas y proporcionar retroalimentación rápida al equipo. Trabaja estrechamente con desarrolladores y otros miembros del equipo para garantizar que la calidad sea un objetivo constante.

**Automatización y POO:**

* **Pilares de la POO:**
* Encapsulamiento: Agrupa datos y funciones, ocultando detalles de implementación.
* Abstracción: Oculta la complejidad y muestra solo lo esencial.
* Herencia: Permite que una clase herede características de otra.
* Polimorfismo: Permite que un objeto adopte múltiples formas según el contexto.
* **Consideraciones al automatizar pruebas:**  
  Selección de herramientas adecuadas, mantenibilidad del código de pruebas, cobertura completa del sistema, repetibilidad de las pruebas y tiempo de ejecución.
* **Principios SOLID en automatización de pruebas:**  
  Los principios SOLID (Responsabilidad única, Abierto/Cerrado, Sustitución de Liskov, Segregación de interfaz y Dependencia de abstracciones) ayudan a crear pruebas escalables, mantenibles y fáciles de modificar.
* **Pirámide de automatización de Cohn:**  
  Propone organizar las pruebas en tres niveles: unitarias en la base (rápidas y frecuentes), de integración en el medio (menos frecuentes) y end-to-end en la cima (lentamente y costosas).
* **Uso de SonarQube:**  
  SonarQube analiza el código en busca de problemas de calidad como complejidad, cobertura y duplicación. Ayuda a mantener un código limpio y sostenible, reduciendo errores y mejorando la mantenibilidad.

**Pruebas No Funcionales:**

* **Pruebas de Seguridad y Performance:**

Seguridad: Detecta vulnerabilidades como inyecciones SQL, XSS, y problemas de autenticación/autorización.

Performance: Mide el rendimiento, carga, estrés y escalabilidad del sistema bajo condiciones diversas de uso.

* **Herramientas para pruebas de seguridad y performance:**

Seguridad: OWASP ZAP, Burp Suite, Nessus.

Performance: JMeter, Gatling, LoadRunner.

* **Función de las pruebas no funcionales:**  
  Validan aspectos del sistema que no están relacionados con su funcionalidad directa, como la seguridad, el rendimiento, la fiabilidad y la escalabilidad, asegurando que el sistema cumpla con los requisitos no funcionales y pueda manejar condiciones del mundo real.