Actividad 5: Estrategia de Control de Calidad y Pruebas Unitarias

Para garantizar la calidad del software, es fundamental establecer una estrategia de pruebas unitarias y validación del código.

1. **Enfoque en Pruebas Unitarias**
   Estas verificarán que el código funcione de manera correcta los componentes individuales y que estén libres de errores.
   Las pruebas unitarias se realizaran con JUnit, el estándar moderno para pruebas en java.
   Las pruebas unitarias estarán organizadas en un paquete separado (src/test/java), siguiendo esta estructura:

   Principios Clave
   - Aplicación de TDD (Test-Driven Development) para escribir pruebas antes del código funcional.
   - Uso de Mockito para mocking y pruebas de componentes desacoplados.
   - Enfoque en pruebas de caja negra (comprobar resultados esperados sin conocer la implementación interna).

2. **Validación del Código y Control de Calidad**
   Para mantener estándares de calidad, se integrarán herramientas de análisis estático y revisión de código.

   Análisis Estático con Checkstyle y SpotBugs:
   - Checkstyle: Asegura que el código cumple con las convenciones de estilo de Java.
   - SpotBugs: Detecta defectos en el código, como posibles NullPointerException o fugas de memoria.
   - SonarQube: Proporciona análisis avanzado de calidad de código y cobertura de pruebas.

   Para la revision del codigo:
   - Se exigirá un Pull Request obligatorio antes de fusionar cambios a la rama principal.
   - Se aplicará Pair Programming en tareas críticas.
   - Se utilizarán Reglas de Merge en GitHub/GitLab para asegurar revisiones antes de integrar código.

   Pruebas en Entornos de Integración(CI/CD)
   - Se integrará GitHub Actions o Jenkins para ejecutar pruebas automáticamente en cada push.
   - Se aplicarán pruebas automatizadas en staging antes del despliegue en producción.

Conclusión
Implementar una estrategia sólida de pruebas unitarias y validación del código es clave para garantizar la calidad, estabilidad y mantenibilidad del software. Al utilizar JUnit para pruebas unitarias, combinadas con principios de TDD y herramientas como Mockito, aseguramos un código modular y libre de errores desde su desarrollo inicial.

Además, la integración de herramientas de análisis estático como Checkstyle, SpotBugs y SonarQube permite detectar problemas tempranos y mantener estándares de calidad. La implementación de revisiones de código, mediante Pull Requests obligatorios y Pair Programming, refuerza la colaboración y previene la acumulación de deuda técnica.

Finalmente, la automatización de pruebas en entornos de CI/CD con GitHub Actions o Jenkins garantiza despliegues confiables y eficientes. Con este enfoque integral, aseguramos que la plataforma de gestión de proyectos sea robusta, escalable y alineada con las mejores prácticas de desarrollo moderno.