- El código no es flexible y más difícil de probar o extender.

- Introduce un acoplamiento fuerte entre Program.cs y las clases SmsNotifier y EmailNotifier.

- La lógica puede crecer de forma desordenada si se añaden más tipos de notificaciones.

- El manejo de errores usa throw new Exception, que no es una práctica recomendada, ya que carece de especificidad.

- El código depende directamente de implementaciones concretas, lo que viola el principio de inversión de dependencias y aumenta el acoplamiento.

- La lógica condicional en el endpoint no es escalable. Agregar nuevos tipos de notificación requerirá modificar múltiples partes del sistema.

- No hay separación clara entre lógica y detalles de implementación, lo que dificulta realizar pruebas unitarias aisladas.

- No se utilizan interfaces ni estrategias de diseño que promuevan la extensibilidad o el desacoplamiento.

- La lógica de negocio está mezclada con la capa de configuración