Escenario de Escalabilidad:

Cualidad: Escalabilidad

- **Requisitos No Funcionales:** El sistema debe manejar 100, 900 y 3000 usuarios simultáneos a nivel local, departamental y nacional respectivamente.
- **Escenario de Calidad:** Cuando la cantidad de usuarios aumenta, el sistema debe mantener un rendimiento constante y rápido sin ralentizarse.
- **Solución Propuesta:** Emplear técnicas de escalabilidad horizontal y vertical.
    - **Escalabilidad Horizontal:** Distribuir la carga entre servidores y replicar la infraestructura para balancear la demanda.
    - **Escalabilidad Vertical:** Optimizar el rendimiento del servidor y base de datos para manejar más usuarios sin perder velocidad.
- **Evaluación:** Realizar pruebas de estrés y monitorear el sistema para detectar y solucionar problemas de rendimiento bajo diferentes cargas.

Escenario de Modificabilidad:

Cualidad: Modificabilidad

- **Requisitos No Funcionales:** El sistema debe ser flexible para cambiar la base de datos o introducir nuevos indicadores sin complicaciones.
- **Escenario de Calidad:** Al realizar cambios en la base de datos o indicadores, el sistema debe permitir estas modificaciones sin afectar su funcionamiento y con esfuerzo y tiempo razonables.
- **Solución Propuesta:**
    - **Diseño Modular:** Separar claramente las partes del sistema para facilitar cambios.
    - **Patrones de Diseño:** Utilizar patrones como Inyección de Dependencias para facilitar la sustitución de componentes.
    - **Documentación Detallada:** Proporcionar documentación clara para que los desarrolladores comprendan fácilmente el sistema.
- **Evaluación:** Medir el tiempo y esfuerzo necesarios para realizar cambios específicos y evaluar el impacto en el sistema.

Tácticas a utilizar:

Escalabilidad:

1. **Tácticas de Escalabilidad:**
    - **Escalabilidad Horizontal:**
        - **Balanceo de Carga:** Implementar un balanceador de carga para distribuir la demanda entre varios servidores. El patrón **"Proxy"** puede ser útil aquí para proporcionar un punto de entrada único.
        - **Particionamiento de Datos:** Dividir grandes conjuntos de datos en particiones más pequeñas distribuidas en múltiples nodos. El patrón **"Sharding"** ayuda a distribuir y gestionar estos fragmentos de datos.

Modificabilidad:

1. **Tácticas de Modificabilidad:**
    - **Diseño Modular:**
        - **Separación de Capas:** Aplicar la arquitectura de capas (como la arquitectura **"MVC" - Modelo-Vista-Controlador**) para separar la lógica de negocios, la interfaz de usuario y el acceso a datos. Esto facilita la modificación de una capa sin afectar las demás.
        - **Desacoplamiento:** Utilizar el patrón **"Observer"** para permitir que los componentes interesados sean notificados de cambios en otros componentes sin estar directamente acoplados.