**Análisis Según SOLID**  
1. S – Single Responsibility (Responsabilidad Única).  
Problema:

El WebComponent **UsuariosTabla** hace demasiadas cosas a la vez:  
- Crea la Interfaz (Botón, tabla, modal).  
- Realiza Solicitudes HTTP(**fetch**).  
- Maneja la Lógica de eventos y visualización de datos.  
- Controla el Modal.  
  
- Esto rompe el principio de responsabilidad única porque un solo componente tiene múltiples razones para cambiar (cambio de estilo, cambio en la API, cambio en la lógica del model, etc).  
  
Refactorización:  
Separar en clases/métodos especializados:  
- **UsuarioService** -> encargado de las Solicitudes HTTP.  
- **TablaUsuarios** -> encargado de renderizar la tabla.  
- **ModalUsuario** -> encargado de renderizar y mostrar el modal.  
- **UsuariosTabla** -> solo orquestador que conecta todo.  
  
  
**2. O – Open/Closed(Abierto/Cerrado)**  
- Problema:  
Si mañana quisiera mostrar más columnas en la tabla (por ejemplo, **company**), debería modificar directamente el código del componente.  
Esto significa que el código no está abierto a extensión sin modificar la clase existente.  
- Refactorización:  
- Crear una configuración de columnas (un array de objetos con **key** y **label**).  
- El Renderizado de la tabla iterará sobre esta configuración.  
- Así podemos extender agregado columnas nuevas sin modificar la clase principal.  
  
**3. L – Liskov Substitution Principle (No Aplica)**  
- No aplica porque no tengo jerarquías de herencia que sustituir.  
Todo está basado en un único WebComponent sin subclases.  
  
**4. I – Interface Segregation (No Aplica)**  
- No aplica porque en JavaScript no trabajo con interfaces explícitas.  
  
**5. D – Dependency Inversion (Inversión de Dependencias)**

- Problema:  
El WebComponent está fuertemente acoplado al **fetch** y al endpoint.  
Si mañana queremos cambiar la fuente de datos, tendríamos que reescribir el código del componente.  
- Refactorización:  
- El Componente debería depender de una abstracción de servicio (**UsuariosService**) que reciba la URL o un objeto de configuración.  
- Así, el WebComponent no depende de **fetch** directamente, sino de un servicio inyectado que se encarga de la obtención de datos.