🛡️ Informe técnico – Exposición de información y riesgo de manipulación de datos en endpoints públicos

Resumen del problema

Actualmente, la aplicación en producción permite que el frontend realice operaciones **CRUD completas** (crear, leer, actualizar y borrar) sobre los grupos mediante llamadas directas a los endpoints:

- `GET /groups`
- `POST /groups`
- `PUT /groups/:id`
- `DELETE /members`
- `POST /groups/:id`
- entre otros…

Estos endpoints están **expuestos públicamente** y **no requieren autenticación**, lo cual implica que **cualquier persona externa**, mediante herramientas como `curl`, Postman o un simple script automatizado, puede:

- Obtener toda la información almacenada.
- Crear grupos falsos.
- Modificar grupos existentes.
- Borrar grupos sin ningún tipo de control.
- Realizar cargas masivas de datos fraudulentos mediante CSV.

Esto representa una **vulnerabilidad crítica**, ya que el usuario final únicamente debería poder **crear su grupo y subir su presentación**, pero no existe ninguna funcionalidad que requiera que el usuario pueda leer, editar o borrar datos.
Es decir: **el diseño original del producto NO contempla interacción posterior del usuario con sus datos**, sin embargo la API sí lo permite.

Solución propuesta (mínima y compatible con producción)

Dado que:

- El frontend **solo necesita poder crear grupos mediante POST**.
- El usuario **no necesita leer, actualizar ni borrar datos**.
- No tenemos acceso al frontend para modificar su lógica interna.

La solución más sencilla y efectiva es **replicar la protección de tokens usada en `/users`** y aplicarla parcialmente a los endpoints de grupos.

/mia/routes/users.py

```
@router.get("/users") =>
def get_users(request: Request,
        state: str | None = None,
        page: str | None = None,
        page_size: str | None = None
    ):
    try:
        logger.info(f"GET /users endpoint called")
        payload = Auth.read_jwt_payload(request)
        if not isinstance(payload, dict):
            return payload
        tenant = payload.get("tenant_id")
```

implementar esta misma logica en los endponits que se deben proteger

🛠 Implementación recomendada

1. **Mantener público únicamente:**
    - `POST /groups`
        - Permite que el usuario siga creando su grupo sin autenticación.
        - No rompe la lógica del frontend.
2. **Proteger con token todos los demás endpoints:**
    - `GET /groups`
    - `PUT /groups/:id`
    - etc.

    Solo el administrador (o procesos internos) podrán utilizarlos.
3. **Reutilizar la misma lógica de token que ya funciona en `/users`**
   Esto garantiza consistencia y reduce el riesgo de generar más fallos.

Resultado esperado

- Un usuario final solo puede **crear su grupo**.
- Ningún usuario sin token podrá:
    - Listar información.

    - Modificar datos.
    - Borrar grupos.
    - Subir CSV masivos.
- Se elimina la vulnerabilidad crítica de exposición y manipulación de información.

> Nota: Todas las recomendaciones deben aplicarse primero en un entorno de prueba antes de pasar a producción para evitar corrupción de datos.