**Middleware en 20minCoach: Documentación y Código**

En el archivo **README.md**, se creará una sección dedicada a la capa de middleware. Se explicará la clave, el propósito de cada componente, proporcionar un flujo de trabajo y ofrecer plantillas de código listas para usar.

**1. Flujo y Propósito General**

Se empieza con una breve descripción de por qué se usa middleware, esto para interceptar las peticiones y respuestas, permitiendo que la lógica de negocio se mantenga limpia y enfocada. Además, un diagrama de flujo para visualizar el proceso.

**2. Plantillas de Código y Patrones de Diseño**

Aquí es donde se detalla la implementación para cada requisito. Cada middleware debe tener su propia subsección con una breve descripción y una plantilla de código. Por ejemplo:

src/

└── middleware/

├── interceptors.js

├── permissionsMiddleware.js

├── errorHandler.js

└── loggingMiddleware.js

**2.1 Request/Response Interceptors**

**Descripción:**  
En esta plantilla de código se muestra cómo interceptar las peticiones antes de que se envíen a la API. Es bueno para agregar automáticamente el token de autenticación del usuario, gestionar los encabezados o transformar los datos de las respuestas.

**Guía para desarrolladores:**  
"Se deben importar estos interceptores y adjuntarlos a cada cliente de API o a la capa de servicios para que se ejecuten automáticamente."

**2.2 Permissions Validation Middleware**

**Descripción:**  
Este middleware es crucial para la seguridad. Se encargará de verificar los permisos del usuario (definidos en el PoC de seguridad) antes de permitir el acceso a ciertas funcionalidades o rutas de la aplicación.

**Guía para desarrolladores:**  
"Usa esta función para proteger componentes o acciones específicas."

**2.3 Error Handling Middleware**

**Descripción:**  
Este componente es un punto centralizado para gestionar las excepciones. Su propósito es interceptar los errores, registrarlos de manera estructurada y mostrar un mensaje amigable al usuario, evitando que se expongan detalles técnicos.

**Guía para desarrolladores:**  
"Este middleware debe ser llamado en cada bloque catch para asegurar que todos los errores se manejen de forma consistente. Su salida debe ser utilizada para actualizar el estado de la UI y mostrar el mensaje al usuario."

**2.4 Logging Events Middleware (Patrón Strategy)**

**Descripción:**  
El sistema de logging utiliza el **patrón Strategy** para permitir múltiples "proveedores" de registro sin cambiar la lógica principal. Esto hace que sea fácil cambiar de un simple console.log a un servicio de logging externo como Sentry o Datadog en el futuro.

**Guía para desarrolladores:**  
"Para registrar un evento, simplemente importa loggingStrategy y llama al método log() con un objeto estructurado. Para cambiar el proveedor de logging, solo necesitas cambiar la instancia de Logger."