ADR: Rediseño y Estandarización del Cliente HTTP

Contexto 🏗️

El cliente HTTP inicial de EcoMarket presentaba limitaciones técnicas que comprometían la escalabilidad y la experiencia de usuario:

- Uso de `fetch()` nativo sin envolturas para tiempos de espera (*timeouts*) o gestión de estados pendientes.
- Variables de configuración (URLs) definidas manualmente en el código, lo que incrementa el riesgo de errores humanos en despliegues a producción.
- Manejo de errores genérico que no permitía a la interfaz de usuario distinguir entre fallos de red, errores de autorización (401) o caídas del servidor (500), resultando en mensajes vagos para el usuario.

Decisión ✅

Se ha decidido implementar un cliente robusto basado en **Axios** con las siguientes especificaciones:

1. **Gestión de Entornos:** Migración de URLs base y claves a variables de entorno (`.env`) para separar configuraciones de desarrollo, mock y producción.
2. **Propagación Enriquecida:** Creación de una clase `EcoMarketError` que centraliza la lógica de traducción de errores mediante interceptores de Axios.
3. **Mapeo de Mensajes:** Los errores se transformarán en mensajes amigables predefinidos (Ej: *"No tienes autorización"* para 401, *"Servicio no disponible"* para 500), preservando el error técnico original en una propiedad `details` para debugging.
4. **Lógica de Idempotencia:** El error incluirá una bandera `isRetryable` calculada automáticamente: será `true` para fallos de red en métodos **idempotentes** (`GET`, `PUT`, `DELETE`) y `false` para operaciones críticas (`POST`) para evitar duplicidades.

Alternativas Consideradas 🔄

- **Seguir con `fetch()` nativo:** Se descartó porque, aunque reduce el tamaño del paquete (*bundle size*), requiere programar manualmente interceptores y lógica de *retry*, lo que aumenta el tiempo de desarrollo y la probabilidad de bugs.
- **Manejo de Errores por Componente:** Se consideró dejar que cada pantalla manejara sus excepciones, pero se descartó en favor del enfoque de **"Propagación Enriquecida"** para garantizar que toda la aplicación responda de manera predecible ante fallos.
- **Variables de URL Manuales:** Se evaluó mantenerlas por su "simplicidad de compilación", pero el riesgo de enviar datos a servidores de prueba o exponer la URL de producción fue considerado inaceptable.

Consecuencias 📈

- **Positivas:** * **Seguridad:** Eliminación de credenciales y URLs en el código fuente.
    - **Mantenibilidad:** La lógica de errores se modifica en un solo lugar (el interceptor).
    - **UX Superior:** El usuario recibe instrucciones claras de qué hacer cuando algo falla, y la interfaz sabe cuándo es seguro ofrecer un botón de reintento.
- **Negativas:**
    - **Tamaño del Bundle:** Ligero incremento por la librería Axios.
    - **Complejidad Inicial:** Requiere que los desarrolladores aprendan la estructura de `EcoMarketError` en lugar de usar el `Error` estándar de JS.