Unidad Académica de Economía

Carrera

Licenciatura en Sistemas Computacionales

Grupo y Semestre

Semestre 6

Nombre estudiante

Ricardo Matos Vizcarra

Actividad

Reto IA 2

Maestro

Eligardo Cruz Sánchez

Materia

Programación Distribuida del lado Cliente

1. Referencia rápida con los 5 casos:

Predicciones iniciales

1.- El usuario intenta crear un producto pero el JSON está mal formado:

Si esta mal formado el JSON es muy probable que haya un error por lo que es necesario tener en cuenta las validaciones necesarias para que se reduzca los errores por el usuario, ya sea que este mal escrito la información que el quiere formar y hacerle saber al usuario que es lo que necesita para poder crear sin problemas el producto.

2.- El usuario intenta ver un producto que fue eliminado ayer:

Mostrarle al usuario que no se encuentra el producto que el esta buscando ya que no se encuentra en el servidor.

3.- El usuario intenta actualizar el precio de un producto pero su sesión expiró:

Si su sesión expiró lo mejor es hacerle saber que no esta con la sesión iniciada por lo que debería volver a iniciar sesión para poder actualizar y validar que es la persona indicada para poder hacer los cambios en el precio de no ser así no se le dará acceso para realizar los cambios.

4.- El servidor de base de datos está sobrecargado temporalmente:

Hacerle saber al usuario que no esta disponible de momento el servidor y que lo intente mas tarde ya que el servidor termine la sobrecarga de datos este podrá seguir con la solicitud.

5.- El usuario intenta crear un producto con un SKU que ya existe:

Hacerle saber al usuario que la ID o SKU ya esta siendo utilizada y que no puede hacer duplicaciones porque puede generar errores y a su vez no permitir que haya duplicaciones de ID's.

Respuestas correctas:

1.- El usuario intenta crear un producto pero el JSON está mal formado:

CÓDIGO: 400 Bad Request **RAZÓN:** Se utiliza cuando el servidor no puede procesar la solicitud debido a algo que es percibido como un error del cliente (sintaxis de solicitud mal formada). No usamos 422 Unprocessable Entity aquí porque el 422 se reserva para cuando el JSON es sintácticamente correcto pero lógicamente inválido (ej. un precio negativo). Si el JSON tiene una coma de más o le falta una llave, es un 400.

2.- El usuario intenta ver un producto que fue eliminado ayer:

CÓDIGO: 410 Gone RAZÓN: Aunque el famoso 404 Not Found funcionaría, el 410 es mucho más preciso para un analista experto. Indica que el recurso existía pero ya no está disponible y, lo más importante, que esta condición es permanente. Esto ayuda a que los motores de búsqueda dejen de indexar esa URL de producto de EcoMarket.

3.- El usuario intenta actualizar el precio de un producto pero su sesión expiró:

CÓDIGO: 401 Unauthorized RAZÓN: A pesar de su nombre, semánticamente significa "Unauthenticated" (no autenticado). El servidor le dice al cliente: "No sé quién eres, así que no puedo dejarte cambiar precios". No usamos 403 Forbidden porque el 403 implica que el servidor sí sabe quién eres, pero que específicamente tú no tienes permiso para esa acción.

4.- El servidor de base de datos está sobrecargado temporalmente:

CÓDIGO: 503 Service Unavailable RAZÓN: Indica que el servidor no está listo para manejar la solicitud. Es mejor que un 500 Internal Server Error porque el 503 comunica que es una condición temporal. A menudo incluye un encabezado Retry-After para decirle al cliente cuánto esperar.

5.- El usuario intenta crear un producto con un SKU que ya existe:

CÓDIGO: 409 Conflict RAZÓN: Este es el código ideal para violaciones de integridad o conflictos con el estado actual del servidor. Un SKU debe ser único en EcoMarket; intentar crear un duplicado crea un conflicto de recursos. Algunos usan 422, pero el 409 es más específico para errores de duplicidad en bases de datos.

Acción recomendada para el cliente

1.- El usuario intenta crear un producto pero el JSON está mal formado:

ACCIÓN CLIENTE: El frontend debe revisar la estructura de la petición. No tiene sentido reintentar la misma petición sin cambios; el desarrollador debe corregir el código que genera el JSON.

2.- El usuario intenta ver un producto que fue eliminado ayer:

ACCIÓN CLIENTE: El frontend debe eliminar el producto de sus cachés locales, mostrar un mensaje informativo ("Este producto ya no forma parte de nuestro catálogo") y redirigir al usuario a la lista general de productos orgánicos.

3.- El usuario intenta actualizar el precio de un producto pero su sesión expiró:

ACCIÓN CLIENTE: El frontend debe capturar este error, guardar el estado actual del formulario (para no perder el nuevo precio que el usuario escribió) y redirigir inmediatamente a la pantalla de Login.

4.- El servidor de base de datos está sobrecargado temporalmente:

ACCIÓN CLIENTE: El frontend debe mostrar una pantalla de "Estamos experimentando mucho tráfico, reintentando en unos segundos..." y programar un reintento automático (idealmente con un algoritmo de exponential backoff) en lugar de dejar al usuario abandonado.

5.- El usuario intenta crear un producto con un SKU que ya existe:

ACCIÓN CLIENTE: El frontend debe resaltar el campo del SKU en rojo y mostrar un mensaje de error específico: "Este SKU ya está registrado para otro producto. Por favor, verifica el código".

Tabla de Referencia:

Caso	Mi predicción	Código técnico	¿Dónde estuvo el ajuste?
1.- JSON Mal formado	Validar errores y avisar al usuario	**400 Bad Request**	Tu lógica es correcta. El ajuste es que el 400 detiene la petición antes de intentar validar datos; es un error de "idioma" (sintaxis).
2.- Producto Eliminado	Mostrar que no se encuentra.	**410 Gone**	Acertaste en que "no está". El ajuste es usar 410 en vez de 404 para decirle al cliente: "No me busques más, esto no volverá".
3.- Sesión Expirada	Re-iniciar sesión y validar identidad.	**401 Unauthorized**	¡Exacto! Tu flujo de redirigir al login es la implementación perfecta del código 401.
4.- Sobrecarga DB	Avisar que intente más tarde.	**503 Service Unav.**	Diste en el clavo. El 503 es precisamente el código que permite enviar un tiempo de espera (Retry-After).
5.- SKU Duplicado	Avisar que ya existe y evitar duplicados.	**409 Conflict**	¡Perfecto! Tu enfoque en la "integridad de datos" es exactamente lo que define a un conflicto 409.