Prueba Técnica: API y Carrito de Compras

Objetivo

Construir una API básica que gestione productos y un carrito de compras, junto con un frontend que consuma esta API para permitir agregar y visualizar productos en el carrito.

Parte 1: Backend (API)

Requisitos:

1. **Tecnología**: Next JS - Ruby on Rails u otra tecnología backend.
2. **Endpoints Rest Full**:
    - `/products`: Devuelve una lista estática de productos.
    - `/cart`: Recibe el ID del producto y lo agrega al carrito.
    - `/cart`: Devuelve el carrito con los productos agregados.
3. **Estructura de los Productos**:

Lista de productos fija (sin base de datos):
json

```
[
  { "id": 1, "name": "Producto 1", "price": 100 },
  { "id": 2, "name": "Producto 2", "price": 200 }
]
```

4. **Consideraciones**:
    - El carrito puede estar en memoria (no es necesario persistir datos).
    - No necesitas manejar stock ni autenticación.

Parte 2: Frontend

Requisitos:

1. **Tecnología**: React o Next.js (opcional TypeScript).
2. **Funcionalidades**:
    - Mostrar la lista de productos (obtenidos desde `/products`).
    - Permitir agregar productos al carrito (usando `/cart`).
    - Mostrar el contenido del carrito (usando `/cart`).
3. **Diseño**:
    - Una interfaz básica con:
        - Lista de productos con botón "Agregar al carrito".
        - Vista del carrito con los productos seleccionados.

Parte 3: Lógica

Implementa una función en React que, dada una lista de productos y un presupuesto máximo, encuentre la **combinación de productos con el mayor valor total sin exceder el presupuesto**.

Detalles:

Dataset inicial:
json

```
[
  { "id": 1, "name": "Producto 1", "price": 60 },
  { "id": 2, "name": "Producto 2", "price": 100 },
  { "id": 3, "name": "Producto 3", "price": 120 },
  { "id": 4, "name": "Producto 4", "price": 70 }
]
```

1. **Requisitos**:
    - Implementa la función `findBestCombination(products, budget)`:
        - **Entrada**: Lista de productos y un presupuesto (por ejemplo, 150).
        - **Salida**: Una lista de productos cuya suma de precios sea la mayor posible sin exceder el presupuesto.
    - Muestra los productos seleccionados en pantalla.

Ejemplo: Si el presupuesto es `150`, la función debería devolver:
json

```
[
  { "id": 1, "name": "Producto 1", "price": 60 },
  { "id": 4, "name": "Producto 4", "price": 70 }
]
```

2. Con un total de `130`.

Entrega

1. Sube tu código a un repositorio público en GitHub.
2. Incluye un archivo `README.md` con:
    - Instrucciones para instalar y ejecutar el proyecto.
    - Breve descripción de la solución.
3. Proveer una URL funcional (opcional, si decides desplegarlo en Vercel, Heroku, Fly.io, etc.).