Tarea Técnica Flapp

Introducción

El proyecto consiste en el desarrollo de una aplicación frontend-backend para simular el comportamiento de una compra a través de un e-commerce llamado “Flapp”.

Para obtener los recursos requeridos se hará uso de la API: https://dummyjson.com/

Requerimientos Backend

1. La aplicación debe proveer el endpoint `POST /api/cart` el cual:
    a. Recibe un listado de productos a “comprar” y datos de envío con el siguiente formato:

```
{
  "products": [
    {
      "productId": "1",
      "price": 50,
      "quantity": 2,
      "discount": 10
    },
    {
      "productId": "3",
      "price": 15,
      "quantity": 3,
      "discount": 2
    },
    ...
  ],
  "customer_data": {
    "name": "Juan Pérez",
```

```
        "shipping_street": "Calle Falsa 123",
        "commune": "Vitacura",
        "phone": "+56900000000"
    }
}
```

b. Obtiene la totalidad de los productos existentes a través de la API dummyjson, siendo esto el equivalente a realizar una consulta a la base de datos, utilizando paginación de 10 en 10.

c. Dada la totalidad de los productos obtenidos, se obtiene el nombre, el stock y el rating de cada uno de los productos solicitados del carrito. Sea **St** el stock obtenido y **r** el rating obtenido, se calcula el stock real Sr en base a la siguiente fórmula:

$$Sr = \lfloor St/r \rfloor$$

d. Imprime en consola el carrito recibido, indicando id, nombre, precio por unidad, descuento total, cantidad solicitada, stock obtenido, rating y stock real para cada producto.

e. Verifica que se pueda satisfacer la cantidad solicitada de cada producto con el pedido con el stock real.

f. Realiza una tarificación a los couriers a TraeloYa y Uder (la documentación de estos se encuentra en el documento “Tarificación de Envíos”) e **imprime en consola su respuesta**. Para la tarificación, utiliza los datos recibidos en la request además de la siguiente información como datos de origen del envío (Estos deben ser enviados a cada courier en el formato que corresponda para cada uno):

```
Nombre: Tienda Flapp
Teléfono: +569 1234 5678
Dirección: Juan de Valiente 3630
Comuna: Vitacura
```

g. Con los resultados de las tarificaciones, obtiene el precio más bajo de las tarifas encontradas y retorna un json con los campos:

- `price`: Un **float** que representa el precio de la tarifa más barata encontrada.
- `courier`: Un **string** con el nombre del courier del cual se escogió la tarifa más barata.

```
{
   courier: 'TraeloYa'
   price: 5990.00
}
```

h. En caso de no poder satisfacer el stock para algún producto o de no encontrar tarifas disponibles se debe retornar un error 400 con la explicación del error.

Requerimientos Frontend

1. Vista inicial simple, donde se desplieguen dos botones. **“Generar carrito”**, el cual gatilla la obtención de un carrito aleatorio de la API de dummyjson, y **“Finalizar compra”** para navegar a la vista de checkout en caso de ya existir un carrito cargado.
2. Vista de checkout más detallada, donde se despliega, considerando criterios de usabilidad y experiencia de usuario, el carrito anteriormente obtenido, simulando el resumen de un carrito de compra en un e-commerce. Además de los botones **“Cotizar despacho”**, **“Limpiar carrito”** y **“Volver”**.
3. Una vista para ingresar datos de envío del comprador, incluyendo al menos **nombre**, **dirección de destino** (nombre de la calle y comuna) y **teléfono.**
4. El botón **“Cotizar despacho”** debe hacer envío del carrito obtenido y los datos de envío al backend, y luego mostrar:
   - En caso de éxito: Envío Flapp con `{courier_seleccionado}` ⚡ - $`{precio_del_envio}`.
   - En caso de fallo: No hay envíos disponibles :(

5. El botón **“Limpiar carrito”** debe volver a la vista inicial limpiando el carrito actual.
6. El botón **“Volver”** debe volver a la vista inicial sin limpiar el carrito actual.

Bonus

Para hacer tu tarea aún más *épica*, te proponemos los siguientes desafíos:

- Dockerizar tu aplicación.
- Testear el endpoint de tarificación.
- *Deployear* tu aplicación a algún servicio.

Consideraciones adicionales

- La tarea (tanto frontend como backend) la puedes realizar en el stack que consideres más adecuado o más te acomode.
- Debes dejar un archivo README.md con instrucciones sobre cómo ejecutar tu tarea.
- Puedes hacer los supuestos que estimes conveniente, sin límite de estos. Eso sí, estos deben estar documentados en el README.

 ¡Estaremos esperando tu respuesta!