

# Prueba Técnica Developer BackEnd

## Preparación del entorno

1. Crear un repositorio en GitHub o GitLab con su nombre - prueba-node... Ejem. Pedro-Perez-prueba-node
2. Crear proyecto node.js con Express y el ORM sequelize
3. Montar la base de datos "**Market**" (anexo modelo relacional)

## Introducción

En la primera parte de la prueba se debe alimentar la base de datos creada realizando endpoints básicos de inserción.

## Endpoints a construir Primera Parte

🔴 Los productos se crean de manera general en el sistema, para esto se requiere crear un Endpoint que reciba los datos básicos de un producto y lo almacene en la tabla **PRODUCTOS** validando los datos de entrada

Validaciones

**Nombre:** Requerido - Maximo 60 caracteres

**barcode:** Requerido - Valor unico en la tabla

**presentacion:** Requerido - Maximo 25 caracteres

🔴 Asociar un producto a una Tienda, para esto se debe hacer Insert en la tabla **tiendas\_productos** asignando valores personalizados

Validaciones

**id\_producto:** Requerido - existir en tabla Productos

**id\_tienda:** Requerido - existir en la tabla Tiendas

**valor:** Requerido

**compra\_maxima:** Requerido - Numerico

En la Segunda parte se debe construir un pequeño e-commerce en el que un usuario Tipo Cliente necesita ver el catálogo de productos de varias tiendas y agregar sus productos a un carrito de compras para cada tienda, los productos tienen precios personalizados para cada tienda y posiblemente algunos tengan promoción, luego de agregar los productos al carrito el cliente procede a hacer la compra y este carrito se debe volver un pedido.

## Endpoints segunda parte

1. Se requiere construir un endPoint que permita listar los productos de una tienda específica, para esto se requiere recibir como parámetro el **id\_tienda** en el Endpoint. Los productos de la tienda se encuentran en la Tabla **Tiendas\_Productos** allí está el "**id\_producto**", "**valor**", esta tabla está relacionada con la Tabla **Productos** donde podemos obtener "**Nombre**", "**Barcode**", para obtener el **valor\_promocion** se debe verificar si el producto cuenta con una promoción disponible para esto se tiene una relación entre la tabla **Tiendas\_productos** y **Promociones** por medio del **Id\_promocion**, si esta relación existe entonces se debe verificar si la tienda tiene la promoción vigente, esto se verifica en la tabla **Tiendas\_promociones** validando el "**id\_promocion**", "**id\_tienda**", "**estado**" = 1 y que la Fecha de hoy esté entre "**inicio**" y "**Fin**"... si esto es verdadero entonces se debe obtener la información de la promoción, (**id\_promocion**, **nombre**, **porcentaje**) para el **valor\_promocion** se debe tomar el valor original y restar el porcentaje de descuento que esta en promociones.

```

PATH: /api/catalogo
TYPE: GET
PARAMS:{
  id_tienda: 6
}
RETURN: {
  "message": "consultado correctamente",
  "data": [
    {
      id_producto: 1
      id_tienda: 6
      nombre: Aguacate
      presentacion: Unidad
      barcode: 10001258
      valor: 5000
      promocion: {
        id_promocion: 25
        nombre: Descuentos especiales
        porcentaje: 25
        valor_promocion: 3750
      }
    }
  ]
}

```

2. Agregar producto al carrito. Se debe hacer Insert en la tabla **CARRITOS** con el **id\_producto**, **id\_tienda**, **id\_user** (por default 1) y cantidad (por default 1).

### 3. Realizar Pedido.

- Para esto se debe tomar todos los productos de la tienda que el cliente tiene en el carrito y pasarlos a pedido, primero se debe crear el pedido con los datos básicos

**instrucciones:** se envia en el requets

**entrega\_fecha:** Se envia en el request

**valor\_productos:** Suma del valor de los productos sin tener en cuenta la promocion

**valor\_descuento:** Sumar la diferencia entre el valor y el valor\_promocion de los productos que aplican

**valor\_envio:** se debe obtener segun la distancia registrada en la direccion del cliente y los parametros de la tienda en tiendas\_distancias

**valor\_final:** se debe calcular sumando valor\_productos - valor\_descuento + valor\_envio

**direccion:** Se obtiene de la direccion actual del cliente

**id\_tienda:** Se envia en el request

**id\_user:** se envia en el request

- También se debe hacer Insert en la tabla Pedidos\_Estados con el estado = 1
- Después de crear el Pedido se deben crear los **pedidos\_productos** segun los productos que están en el carrito
  - cantidad: La que se tenga en el carrito
  - valor\_unitario:** Valor unitario del producto antes de promocion
  - valor\_unitario\_promocion:** Valor del producto con promocion... si no tiene entonces valor unitario
  - total\_teorico:** multiplicar valor\_unitario X cantidad
  - total\_final:** multiplicar valor\_unitario\_promocion x cantidad
  - id\_promocion:** Solo si aplica en ese momento, de lo contrario NULL
  - id\_producto:** Se obtiene del carrito
  - id\_pedido:** Relacionar al pedido creado anteriormente

4. Listar los Pedidos del Cliente, tener en cuenta que el Cliente puede tener pedidos en varias tiendas entonces se debe agrupar por tienda y solo se deben listar los pedidos en estado (1,2,3).... ejemplo de respuesta

```
PATH: /api/pedidos
TYPE: GET
PARAMS:{
  id_user: 1
}
RETURN: {
  "message": "consultado correctamente",
  "data": [
    {
      id_tienda: 1
      nombre: mas x menos
      valor_pedidos: 52000
      cantidad_pedidos: 3
      pedidos: [
        id: 25
        fecha: 2024-02-15
        estado: Confirmado
        valor_final: 15400
        productos: [
          {
            id_producto: 14
            nombre: Aguacate
            presentacion: Unidad
            cantidad: 3
            valor: 5000
            valor_promocion: 3750
            valor_total: 11250
          }
        ]
      ]
    }
  ]
}
```

# Modelo Relacional

