Luvik Middleware Docs

Autenticacion
Cada peticion a realizar debera ser realizada enviando una clave en los headers de cada una de estas.
Headers:
- optional Accept-Encoding: gzip, defilate, br
- optional Cache-Control: no-cache
- optional Connection: keep-alive
- required Content-Type: application/json
- required x-auth-key: ${AUTH_KEY}

Productos
POST /products/batch
Descripción: Maneja el lote de operaciones Create, Update, Delete de un conjunto de productos.
Tipo de Petición: POST
Datos a Enviar:

```
{
  "created": [{
    // Estructura de datos del lote
  }],
  "updated": [{
    // Estructura de datos del lote
  }],
  "deleted": [{
    // Estructura de datos del lote
  }]
}
```

Controlador: `productController.handleBatch()`
Estructura de Datos:
- Created: Conjunto de Productos a ser creado en Shopify.
productData required --> productData Producto
No pasar ID. Este es creado automaticamente en Shopify
No repetir titulos. Devuelve estado 422

- Updated: Conjunto de Productos a ser editados.
id required
productData optional --> Se pasan las propiedades junto a sus valores asociados a ser editados

- Deleted: Conjunto de Productos a ser eliminados.
id required

Importante: Esta peticion maneja las 3 operaciones de manera simultanea. No es necesario realizarlas de manera simultanea. Se pueden realizar de manera independiente unicamente especificando la operacion junto a sus datos.

Siempre se debe pasar el precio mas alto de los 3 tipos de cliente. --> Super importante

El scope de Product de la API de Shopify no incluye la propiedad "collections". Igualmente es posible asignarle la coleccion pasando la propiedad "collection" junto al ID de la colecciones. Esta propiedad recibe un [] de IDs de colecciones tanto en Created como en Updated. Si se desea actualizar las colecciones de un producto, se debera pasar todas las colecciones resultantes y NO solo las nuevas ya que el metodo elimina las colecciones anteriores. Esto para evitar duplicaciones.

Manejo de datos de precios y ofertas:
- Si existe oferta se debe cargar el precio de oferta en price y el precio real en compare_at_price
- Caso contrario, se carga el precio en price

Ejemplo + detallado:

```
{
  "created": [
    {
      "product": {
        "lumps": 10,
        "collection": ["collection ID`s"]
        "title": "Title",
        // ... product keys
        "variants": [{
          "compare_at_price": "150",
          "price": "100",
          "sku": "ABC123"
          "inventory_managment": "shopify", --> required. sin esto, no es posible manejar stock
          "stock": 10,
          // ... product variant keys
        }]
      }
```

```
    }
  ],
    "updated": [
      {
        "product": {
          "collection": ["collection ID`s"]
          "id": "1" --> required
          "title": "New Title",
           "variants": [{
              "compare_at_price": "150",
              "price": "100",
              "stock": 10
              // ... product variant keys
            }]
          // ...product keys
        }
        // ...product props
      }
    ],
    "deleted": [
      "1",
      "2"
      // ...product id's
    ]
}
```

POST /products/update-stock/:id/:newStock --> ERP a Shopify
Descripción: Actualiza el stock de un producto específico.
Parámetros en la URL:
id (number): El ID del producto cuyo stock se va a actualizar.
newStock (number): La nueva cantidad de stock.
Datos a Enviar: No se requiere un cuerpo en la solicitud, ya que los datos necesarios están en los parámetros de la URL.
Controlador: `productController.updateProductStock`

GET /products/list
Descripción: Obtiene y lista todos los productos de la tienda.
Datos a Enviar: No se requiere un cuerpo en la solicitud.
Controlador: `productController.listProducts`

GET /products/list-by-id/:id
Descripción: Obtiene y lista un producto en especifico
Datos a Enviar: ID Producto.
Controlador: `productController.listProductById`

GET /products/list-collections
Descripción: Obtiene y lista ID y Nombre todas las colecciones de la tienda.
Datos a Enviar: No se requiere un cuerpo en la solicitud.
Controlador: `productController.listCollections`

GET /products/get-id-by-name/:name
Descripción: Busca productos por nombre y devuelve los IDs de los productos coincidentes.
Parámetros en la URL:
name (string): El nombre del producto a buscar.
Datos a Enviar: No se requiere un cuerpo en la solicitud, ya que el nombre del producto está en el parámetro de la URL.
Controlador: `productController.getProductIDsByName`

Usuarios
GET /customers/list
Descripción: Obtiene y lista todos los usuarios de la tienda.
Datos a Enviar: No se requiere un cuerpo en la solicitud.
Controlador: `customerController.listUsers`

****GET /customers/list/:id****
Descripción: Obtiene los detalles de un usuario específico por ID.
Tipo de Petición: GET
Parámetros en la URL:
id (number): El ID del usuario cuyos detalles se van a obtener.
Datos a Enviar: No se requiere un cuerpo en la solicitud, ya que el ID del usuario está en el parámetro de la URL.
Controlador: `customerController.getUserByID`

****GET /customers/get-id-by-name/:name****
Descripción: Busca usuarios por nombre y devuelve los IDs de los usuarios coincidentes.
Tipo de Petición: GET
Parámetros en la URL:
name (string): El nombre del usuario a buscar.
Datos a Enviar: No se requiere un cuerpo en la solicitud, ya que el nombre del usuario está en el parámetro de la URL.

Controlador: `customerController.getUserIDByName`

POST /customers/delete/:id
Descripción: Elimina un usuario específico por ID.
Tipo de Petición: POST
Parámetros en la URL:
id (number): El ID del usuario que se va a eliminar.
Datos a Enviar: No se requiere un cuerpo en la solicitud, ya que el ID del usuario está en el parámetro de la URL.
Controlador: `customerController.deleteUser`

POST /customers/update/:id
Descripción: Actualiza los detalles de un usuario específico por ID.
Tipo de Petición: POST
Parámetros en la URL:
id (number): El ID del usuario cuyos detalles se van a actualizar.
Datos a Enviar:

```
{
  userData: {
    // Estructura de los datos del usuario a actualizar
  }
}
```

Controlador: `customerController.updateUser`

Orders
WEBHOOK POST /orders/new/*
Descripción: Cada vez que se crea una orden, Shopify envia una notificacion con todos los datos de esta. La firma del Webhook ha sido cifrada correctamente, esto con el objetivo de evitar una posible vulnerabilidad. Este dato es posible utilizarlo al gusto del ERP, con este pueden cargarlo en una db, analizarlo, etc.
Tipo de Petición: POST
Datos que recibe: Order Data