

API Winvisión

Documentación y uso

Versión 1.2



Sumario

| | |
|---|----|
| 1. Introducción..... | 3 |
| 1.1. Conceptos..... | 3 |
| 1.2. Estructura de un endpoint..... | 3 |
| 1.3. Estructura de los headers..... | 3 |
| 1.4. Estructura de una query..... | 4 |
| 1.5. Estructura del body..... | 4 |
| 2. Repositorio y configuración inicial..... | 4 |
| 2.1. Pasos iniciales..... | 4 |
| 2.2. Fichero .env..... | 4 |
| 2.3. Iniciar servicio..... | 6 |
| 3. Endpoints..... | 6 |
| 3.1. Familias..... | 6 |
| 3.1.1. GET..... | 6 |
| 3.2. Productos..... | 8 |
| 3.2.1. GET..... | 8 |
| 3.3. Compras..... | 12 |
| 3.3.1. GET..... | 12 |
| 3.3.2. POST..... | 16 |
| 4. Integración con WooCommerce..... | 18 |
| 4.1. Match entre Winvisión y WooCommerce..... | 20 |
| 5. Integración con Shopify..... | 20 |

1. Introducción

El propósito de este documento es el de dar a conocer el funcionamiento y uso de la API desarrollada por Larcu para la interacción de cualquier software con la aplicación Winvisión. Para una correcta integración con la API, es fundamental comprender los diversos aspectos y fundamentos de la misma y cómo administrar las respuestas obtenidas. Lo primero y más esencial es saber qué es un endpoint.

1.1. Conceptos

- **Endpoint:** URL de entrada para la comunicación con la API.
- **Headers:** Parámetros necesarios para que el endpoint permita la comunicación con la API
- **Query:** El endpoint puede ir acompañado de parámetros que identifican un recurso único. Estos se añaden tras la URL del endpoint.
- **Body:** Los parámetros pasados al endpoint a través del body son utilizados para enviar datos a la API. En caso de la API de Winvisión, son utilizados para filtrar datos.

1.2. Estructura de un endpoint

Las tablas que van a definir cada endpoint estarán compuestas por los siguientes campos:

- **Nombre:** nombre del endpoint.
- **Descripción:** definición del uso del endpoint
- **Método:** definición del método HTTP que se utilizará para llamar al endpoint. Aunque los métodos utilizados son GET, POST, PUT, PATCH y DELETE, la API de Winvisión únicamente necesitará hacer uso de GET y POST.
- **Ruta:** la URL sin la base. En caso de necesitar algún parámetro en la ruta, este se indicará entre {}.
- **Grupo:** nombre del grupo al que pertenece el endpoint.

1.3. Estructura de los headers

La definición de los headers estarán compuestos por los siguientes campos:

- **Nombre:** nombre del parámetro.
- **Descripción:** texto descriptivo del parámetro.
- **Valor:** posible valor por defecto del parámetro.
- **Tipo:** definición del tipo de parámetro. El cual puede ser string, number, date, etc.
- **Requerido:** define si un parámetro es obligatorio o no.

1.4. Estructura de una query

La definición de los parámetros de la query estarán compuestos por los siguientes campos:

- Nombre: nombre del parámetro.
- Descripción: texto descriptivo del parámetro.
- Valor: posible valor por defecto del parámetro.
- Tipo: definición del tipo de parámetro. El cual puede ser string, number, date, etc.
- Requerido: define si un parámetro es obligatorio o no.

1.5. Estructura del body

La definición de los parámetros del body estarán compuestos por los siguientes campos:

- Nombre: nombre del parámetro.
- Descripción: texto descriptivo del parámetro.
- Valor: posible valor por defecto del parámetro.
- Tipo: definición del tipo de parámetro. El cual puede ser string, number, date, etc.
- Requerido: define si un parámetro es obligatorio o no.

2. Repositorio y configuración inicial

Repositorio de descarga: <https://github.com/larcu/larcu-api>

2.1. Pasos iniciales

El primer paso será el de clonar el repositorio de la API en el servidor en que está instalado Winvisión. Para ello redirigimos a la documentación oficial de [GitHub](#).

Tras clonar el repositorio, nos dirigimos al nuevo directorio creado y procedemos a instalar todos los paquetes requeridos haciendo uso de `npm install`.

2.2. Fichero .env

En el fichero .env es donde se encuentra toda la configuración de la API. Es en este archivo donde se establecerán los parámetros propios de cada servidor para el correcto funcionamiento de la API. A continuación pasamos a describir cada línea del fichero.

App data

NODE_ENV = Si su valor es 'development', la API permitirá comunicación con otros dominios externos al local. En caso contrario, la comunicación solo se podrá llevar a cabo desde el mismo localhost. Para más información sobre cors [leer aquí](#).

APP_PORT = Puerto a utilizar para comunicarse con la API. Por defecto se usa el 3000

APP_VERSION=0.1

APP_MIN_VERSION=0.1

APP_NAME='API Winvision'

Database data

DB_HOST = Host de la base de datos. Por defecto 'localhost'

DB_USER = Usuario de conexión a la base de datos. Por defecto 'SYSDBA'

DB_PASS = Contraseña de acceso a la base de datos.

DB_PORT = Puerto utilizado por la base de datos. Por defecto 3050

DIRECTORIO BBDD

DB_DIR = Directorio de la base de datos. Debe acabar en "\\".

Ejemplo: 'C:\Winvision\Base de datos\'

DB_COMUN = Nombre de la base de datos común. Por defecto 'COMUN'

DB_EMPRESA = Prefijo del nombre de la base de datos de la empresa. Por defecto 'EMPRESA_'

SHOP

TIME_CHECK = Tiempo en milisegundos establecido para comprobar si se ha hecho una compra en Winvisión y actualizar stock de la web.

URL_STORE = Enlace de la tienda online

WOOCOMMERCE

SECRET_KEY = Token usado en el webhook de WooCommerce

CONSUMER_KEY = Consumer key de la API de WooCommerce

CONSUMER_SECRET = Consumer secret de la API de WooCommerce

SHOPIFY

SHOPIFY_SIGNATURE_SECRET=' '

Logging

LOG = En caso de ser true, las consultas a las bases de datos serán visibles en la consola del servidor. Al tener como valor false, estas no serán visibles.

2.3. Iniciar servicio

Finalmente, estando posicionados en una terminal en el directorio de la api, hacemos uso de `npm run init` para iniciar el servicio en el servidor.

3. Endpoints

Llegados a este punto, la API estará corriendo en el servidor esperando la comunicación con cualquier otro software. Para conocer cada grupo de acciones de la API, se describe cada endpoint a continuación. Como aclaración, al utilizar la constante `URL_BASE` nos referimos a la url configurada previamente en el fichero `.env`.

3.1. Familias

Grupo que permite interaccionar con las familias de Winvisión.

3.1.1. GET

3.1.1.1 Endpoint

| Nombre | Descripción | Método | Ruta | Grupo |
|------------------|--|--------|----------------------------------|----------|
| Obtener familias | Devuelve las familias que cumplen con los parámetros pasados | GET | <code>URL_BASE/api/family</code> | Familias |

3.1.1.2. Headers

| Nombre | Descripción | Valor | Tipo | Requerido |
|--------------|---|-------------------------------|--------|-----------|
| Content-Type | Indica el tipo de estructura que recibirá el endpoint | <code>application/json</code> | String | Sí |

| | | | | |
|---------|---|--|--------|----|
| usuario | Valor del usuario de acceso a Winvisión | | String | Sí |
| clave | Valor de la contraseña de acceso al Winvisión (distinta de vacía) | | String | Sí |

3.1.1.3. Query

| Nombre | Descripción | Valor | Tipo | Requerido |
|-------------|--|-------|--------|-----------|
| id | Permite recuperar una familia en concreto según su identificador en la base de datos | | Number | No |
| codigo | Permite obtener una familia en concreto según su código en Winvisión | | Number | No |
| descripcion | Permite obtener una familia en concreto según su descripción en Winvisión | | String | No |

3.1.1.5. Body

| Nombre | Descripción | Valor | Tipo | Requerido |
|--------|-------------|-------|------|-----------|
| | | | | |

3.1.1.6. Salidas

Como resultado, se obtendrá un JSON formado por los siguientes campos (se incluye ejemplo de valores para una familia en concreto):

```
[
  {
    "ID_FAMILIA": 1,
    "CODIGO": 1,
    "DESCRIPCION": "MONTURAS",
```

```

    "ID_IVA_COMPRA": 14,
    "ID_IVA_VENTA": 15,
    "DESCUENTO": 0,
    "ID_GRUPO": 1,
    "CAMBIO_COSTE": 2,
    "DESCUENTO_DESDE": "1899-12-30T00:00:00.000Z",
    "DESCUENTO_HASTA": "1899-12-30T00:00:00.000Z",
    "ID_CUENTA_COMPRA": 303,
    "ID_CUENTA_VENTA": 451,
    "VENDER_NEGATIVO": 1
  }
]

```

3.1.5. Códigos de error

| Código | Texto | Descripción |
|--------|-------|----------------|
| 500 | Error | Error interno. |

3.2. Productos

Grupo que permite interaccionar con los artículos de Winvisión.

3.2.1. GET

3.2.1.1 Endpoint

| Nombre | Descripción | Método | Ruta | Grupo |
|--------------------|---|--------|----------------------------------|-----------|
| Obtener artículos | Devuelve los artículos que cumplen con los parámetros pasados en el body | GET | URL_BASE/ api/item | Productos |
| Exportar artículos | Exporta los artículos que cumplen con los parámetros pasados en el body a csv | GET | URL_BASE/ api/item/ export | Productos |

3.2.1.2. Headers

| Nombre | Descripción | Valor | Tipo | Requerido |
|--------------|---|------------------|--------|-----------|
| Content-Type | Indica el tipo de estructura que recibirá el endpoint | application/json | String | Sí |
| usuario | Valor del usuario de acceso a Winvisión | | String | Sí |
| clave | Valor de la contraseña de acceso al Winvisión (distinta de vacía) | | String | Sí |

3.2.1.3. Query

| Nombre | Descripción | Valor | Tipo | Requerido |
|------------|--|-------|--------|-----------|
| id | Permite recuperar un artículo en concreto según su identificador en la base de datos | | Number | No |
| referencia | Permite obtener un artículo en concreto según su referencia en Winvisión | | String | No |
| familia | Permite obtener un artículo o grupo de artículos según la descripción de su familia en Winvisión | | String | No |

3.2.1.5. Body

| Nombre | Descripción | Valor | Tipo | Requerido |
|--------|-------------|-------|------|-----------|
| | | | | |

3.2.1.6. Salidas

Como resultado, se obtendrá un JSON formado por los siguientes campos (se incluye ejemplo de valores para un modelo de gafa):

```
[{
  "ID_ARTICULO": 7276,
  "REFERENCIA": "10.211.096",
  "CODIGO_BARRAS": 9990000232420,
  "REFERENCIA_PROVEEDOR": null,
  "DESCRIPCION": "GAFA Mod. 551 51-20 31 54 LASTES",
  "ID_FAMILIA": 102,
  "ID_IVA_COMPRA": 14,
  "PRECIO_PVP1_PVP2": 0,
  "FECHA_GARANTIA": "2079-05-29T23:00:00.000Z",
  "FECHA_CADUCIDAD": "2079-05-29T23:00:00.000Z",
  "FECHA_ULTIMA_COMPRA": "2018-05-23T23:00:00.000Z",
  "FECHA_ULTIMA_VENTA": "2020-08-12T23:00:00.000Z",
  "MODELO": null,
  "RADIO_1": null,
  "RADIO_2": null,
  "COLOR": null,
  "VARILLA": null,
  "OFERTA_DESDE": "2079-05-29T23:00:00.000Z",
  "OFERTA_HASTA": "2079-05-29T23:00:00.000Z",
  "NUEVO_DESDE": "2079-05-29T23:00:00.000Z",
  "IMAGEN": null,
  "ID_PROVEEDOR": 22,
  "ID_TARTICULO": 0,
  "ID_PROVEEDOR_ULTIMO": 22,
  "ID_TIPO": -1,
```

"PRECIO_COSTE_M1": 23.7,
"PRECIO_COSTE_M2": 3944,
"PRECIO_VENTA1_M1": 94.96,
"PRECIO_VENTA1_M2": 15800.01,
"PRECIO_VENTA2_M1": 119.3,
"PRECIO_VENTA2_M2": 19849.85,
"PRECIO_COSTE_MED_M1": 24.53,
"PRECIO_COSTE_MED_M2": 4082,
"DESCUENTO_COMPRA": 0,
"MARGEN_BENEFICIO": 274,
"STOCK_MAXIMO": 0,
"STOCK_MINIMO": 0,
"STOCK_PEDIDO": 0,
"ULTIMO_DESCUENTO": 0,
"COMPRADAS_UNIDADES": 0,
"COMPRADAS_IMP_M1": 0,
"COMPRADAS_IMP_M2": 0,
"VENDIDAS_UNIDADES": 0,
"VENDIDAS_IMP_M1": 0,
"VENDIDAS_IMP_M2": 0,
"DIAMETRO": 0,
"ESFERA": 0,
"CILINDRO": 0,
"TAMANO": "0",
"ID_TRATAMIENTO_1": -1,
"ID_TRATAMIENTO_2": -1,
"ID_TRATAMIENTO_3": -1,

```

    "ID_MARCA": -1,
    "OFERTA_IMPORTE_M1": 0,
    "OFERTA_IMPORTE_M2": 0,
    "NUEVO_PRECIO_M1": 0,
    "NUEVO_PRECIO_M2": 0,
    "ID_IVA_VENTA": 15,
    "FECHA_CREACION": "1899-12-30T00:00:00.000Z",
    "PREGUNTAR_PVP": null,
    "VENDER_SIN_EXISTENCIAS": null,
    "TELEMATICO": 1,
    "PRECIO_VENTA_TIENDA_VIRTUAL": 94.96,
    "NOMBRE_FAMILIA": "MONTURAS",
    "STOCK": 2
  }
]

```

3.2.1.7. Códigos de error

| Código | Texto | Descripción |
|--------|---------------------|--|
| 401 | Non existent family | La familia pasada como parámetro en el body no existe. |
| 500 | Error | Error interno. |

3.3. Compras

Grupo que permite interaccionar con las compras de artículos.

3.3.1. GET

3.3.1.1 Endpoint

| Nombre | Descripción | Método | Ruta | Grupo |
|--------------------------------|--|--------|---------------------------|---------|
| Obtener movimientos de compras | Devuelve las compras realizadas que cumplen con los parámetros pasados en el | GET | URL_BASE/ api/purchase | Compras |

| | | | | |
|--|------|--|--|--|
| | body | | | |
|--|------|--|--|--|

3.3.1.2. Headers

| Nombre | Descripción | Valor | Tipo | Requerido |
|--------------|---|------------------|--------|-----------|
| Content-Type | Indica el tipo de estructura que recibirá el endpoint | application/json | String | Sí |
| usuario | Valor del usuario de acceso a Winvisión | | String | Sí |
| clave | Valor de la contraseña de acceso al Winvisión (distinta de vacía) | | String | Sí |

3.3.1.3. Query

| Nombre | Descripción | Valor | Tipo | Requerido |
|--------|--|------------|--------|-----------|
| inicio | Permite obtener una compra o grupo de compras hechas desde la API desde una fecha en concreto. | mm/dd/yyyy | String | No |
| fin | Permite obtener una compra o grupo de compras hechas desde la API hasta una fecha en concreto. | mm/dd/yyyy | String | No |

3.3.1.5. Body

| Nombre | Descripción | Valor | Tipo | Requerido |
|--------|-------------|-------|------|-----------|
| | | | | |

3.3.1.6. Salidas

Como resultado, se obtendrá un JSON formado por los siguientes campos (se incluye ejemplo de valores para una compra en concreto):

[

```
{  
  "ID_MOVIMIENTO": 162730,  
  "ID_TMOVIMIENTO": 4,  
  "DESCRIPCION_MOV": "Venta de tienda virtual",  
  "FECHA": "2021-01-28T11:28:25.000Z",  
  "FECHA_SISTEMA": null,  
  "FECHA_OPERACION": null,  
  "REFERENCIA": "10.211.096",  
  "DESCRIPCION": "GAFA Mod. 551 51-20 31 54 LASTES",  
  "UNIDADES": 1,  
  "PRECIO_COSTE_M1": 23.7,  
  "PRECIO_COSTE_M2": 3944,  
  "PRECIO_VENTA_M1": 94.96,  
  "PRECIO_VENTA_M2": 0,  
  "IVA_COMPRA_PORC": 0,  
  "IVA_COMPRA_IMP_M1": 0,  
  "IVA_COMPRA_IMP_M2": 0,  
  "IVA_VENTA_PORC": 10,  
  "IVA_VENTA_IMP_M1": 9.5,  
  "IVA_VENTA_IMP_M2": 0,  
  "ID_ALMACEN": 1,  
  "PROVEEDOR_CODIGO": 0,  
  "PROVEEDOR_DESCRIPCION": null,  
  "FAMILIA_CODIGO": 115,  
  "FAMILIA_DESCRIPCION": "MONTURAS CON 50% DE DESCUENTO",  
  "AGENTE_1": 0,  
  "AGENTE_1_COMISION": 0,
```

"AGENTE_1_NOMBRE": null,
"AGENTE_2": 0,
"AGENTE_2_COMISION": 0,
"AGENTE_2_NOMBRE": null,
"AGENTE_3": 0,
"AGENTE_3_COMISION": 0,
"AGENTE_3_NOMBRE": null,
"CLIENTE_CODIGO": 0,
"CLIENTE_NOMBRE": null,
"NUMERO_ALBARAN": null,
"ALBARAN_RECIBIDO": 0,
"GRUPO_VENTA_DESCRIPCION": null,
"DESCUENTO_1": 0,
"DESCUENTO_IMPORTE_1_M1": 0,
"DESCUENTO_IMPORTE_1_M2": 0,
"DESCUENTO_2": 0,
"DESCUENTO_IMPORTE_2_M1": 0,
"DESCUENTO_IMPORTE_2_M2": 0,
"DESCUENTO_3": 0,
"DESCUENTO_IMPORTE_3_M1": 0,
"DESCUENTO_IMPORTE_3_M2": 0,
"GRUPO_VENTA_CODIGO": null,
"ID_TIENDA": 1,
"PRECIO_COSTE_MEDIO_M1": 0,
"PRECIO_COSTE_MEDIO_M2": 0,
"DESCUENTO_LINEA": 0,
"DESCUENTO_LINEA_M1": 0,

```

    "DESCUENTO_LINEA_M2": 0,
    "LIQUIDO_M1": 104.46,
    "LIQUIDO_M2": 0,
    "BI_BASE_M1": 94.96,
    "BI_BASE_M2": 0,
    "BI_IVA_M1": 9.5,
    "BI_IVA_M2": 0,
    "BI_RECARGO_M1": 0,
    "BI_RECARGO_M2": 0,
    "ID_DETALLE": -1,
    "TIPO_DETALLE": 0
  }
]

```

3.3.1.7. Códigos de error

| Código | Texto | Descripción |
|--------|-------|----------------|
| 500 | Error | Error interno. |

3.3.2. POST

3.3.2.1 Endpoint

| Nombre | Descripción | Método | Ruta | Grupo |
|--------------------------------|--|--------|---|---------|
| Realizar compra | Crea una compra y registra su movimiento según la configuración en Winvisión | POST | URL_BASE/ api/purchase | Compras |
| Compra a través de WooCommerce | Endpoint usado para reducir el stock de Winvisión al hacer una compra online | POST | URL_BASE/ api/ purchase/ woocommerce | Compras |

3.3.2.2. Headers

| Nombre | Descripción | Valor | Tipo | Requerido |
|--------------|---|------------------|--------|-----------|
| Content-Type | Indica el tipo de estructura que recibirá el endpoint | application/json | String | Sí |
| usuario | Valor del usuario de acceso a Winvisión | | String | Sí |
| clave | Valor de la contraseña de acceso al Winvisión (distinta de vacía) | | String | Sí |

3.3.2.3. Query

| Nombre | Descripción | Valor | Tipo | Requerido |
|--------|-------------|-------|------|-----------|
| | | | | |

3.3.2.5. Body

| Nombre | Descripción | Valor | Tipo | Requerido |
|------------|--|-------|--------|-----------|
| referencia | La compra se realiza del artículo con la referencia dada | | String | Sí |
| unidades | Se reducirán del stock tantas unidades como indique este parámetro | | Number | Sí |
| id | Id del artículo comprado | | Number | No |

3.3.2.6. Salidas

Como resultado, se obtendrá un **ok** en caso de que la compra haya finalizado correctamente. En caso contrario, a continuación se indican los códigos de error.

3.3.2.7. Códigos de error

| Código | Texto | Descripción |
|--------|--------------|--|
| 400 | Bad request | Faltan en el body los parámetros referencia, unidad o ambos. |
| 400 | Out of stock | No hay stock disponible. |

| | | |
|-----|----------------------|--|
| 400 | Store does not exist | El almacén configurado en Winvisión no existe. |
| 500 | Error | Error interno. |

4. Integración con WooCommerce

Para una correcta integración con WordPress y su plugin para tiendas online WooCommerce, es necesario completar los datos del fichero `.env` bajo el título de `#WOOCOMMERCE`. Para ello, tras haber instalado el plugin de WooCommerce en la web que se quiere integrar con Winvisión, es necesario dirigirse a la siguiente dirección dentro del panel de WordPress:

WooCommerce > Ajustes > Avanzado > API REST y solicitar las claves. Las cuales se pegarán en los campos `CONSUMER_KEY` Y `CONSUMER_SECRET` del fichero `.env`.

Tras haber realizado este paso, el siguiente será crear el webhook que avisará a Winvisión de que se ha hecho una compra a través de la web. De esta forma, se podrá reducir el stock del artículo en Winvisión. Para ello es necesario dirigirse a la siguiente dirección dentro del panel de WordPress:

WooCommerce > Ajustes > Avanzado > Webhooks y añadir uno nuevo. Los campos solicitados para crear el webhook son los siguientes:

- Nombre: nombre con el que se conocerá el webhook.
- Estado: el cual se deberá establecer como “Activo” para que el webhook esté escuchando.
- Tema: acción que disparará el webhook. En este caso se debe establecer el valor de “Pedido creado”
- URL de entrega: La cual se establecerá como el endpoint:
`URL_BASE/api/purchase/woocommerce`

Es importante que la URL base no tenga el puerto incluida en la dirección. Para ello, en caso de que la API de Winvisión se encuentre corriendo en el puerto 3000. Se deberá abrir un puerto en el router apuntando a la IP que corre la API de Winvisión. En el puerto local se establecerá el 3000, en el puerto externo se establecerá el 80.

- Secreto: token personal que también deberá ser copiado en el campo `SECRET_KEY` del fichero `.env`.
- Versión API: Integración con WP REST API v3

Una vez finalizada la configuración de los ajustes de WooCommerce, se hará una petición GET al endpoint `URL_BASE/api/item/export` para descargar un fichero csv con todos los artículos marcados como vendibles en la tienda online. También existe la posibilidad de ejecutar un script en consola para que se genere el fichero en el directorio “doc” de forma automática. Para ello el comando a ejecutar es `npm run exportcsv` (sin comillas). Este fichero CSV generado deberá ser importado en la siguiente dirección del menú de WordPress: Productos > Importar.

Las cabeceras quedarán asignadas de la siguiente forma a los campos del CSV:

| | |
|---|--------------------------|
| ID Ejemplo: 2656 | ID |
| sku Ejemplo: ACUVUE1 | SKU |
| post_title Ejemplo: ACUVUE OASYS 1 DAY PACK DE 30 LEN TILLAS | Nombre |
| post_content Ejemplo: ACUVUE OASYS 1 DAY PACK DE 30 LEN TILLAS | Descripción |
| regular_price Ejemplo: 32 | Precio normal |
| sale_price Ejemplo: 32 | Precio rebajado |
| manage_stock Ejemplo: true | ¿En inventario? |
| stock_quantity | Inventario |
| weight | Peso (kg) |
| Images Ejemplo: https://images.ray-ban.com/is/image/RayBan/805289602057__001.png | Imágenes |
| categories Ejemplo: LENTILLAS | Categorías |
| meta:attribute_color | Importar como datos meta |

Al finalizar la importación, se habrán creado todos los artículos y categorías incluidos en el fichero CSV. La API de Winvisión podrá iniciarse para comenzar a recibir peticiones y que esta sea capaz de tener el stock de los artículos sincronizado tanto en Winvisión como en WooCommerce.

4.1. Match entre Winvisión y WooCommerce

La API dispone de un script para subir a la tienda online los nuevos productos dados de alta en Winvision y actualizar el stock de los mismos (si se han comprado nuevas unidades). De esta forma, ejecutando el comando “`npm run match`” (sin comillas) tras hacer un tratamiento del inventario o alguna recepción de productos, se mantendrá actualizada la base de datos de productos de WooCommerce de forma automática.

5. Integración con Shopify

Para una correcta integración con Shopify y sus webhooks, es necesario completar los datos del fichero `.env` bajo el título de `#SHOPIFY`. Para ello, tras haber ingresado en el apartado de Configuración > Notificaciones > Webhooks de la tienda de Shopify que se quiere integrar con Winvisión, es necesario configurar el campo `SHOPIFY_SIGNATURE_SECRET` del fichero `.env` con la clave de firma dada por Shopify para verificar la integridad.

Tras haber realizado este paso, el siguiente será crear el webhook que avisará a Winvisión de que se ha hecho una compra a través de la web. De esta forma, se podrá reducir el stock del artículo en Winvisión. Para ello hay que crear un webhook nuevo en el apartado en el que hemos localizado la clave de verificación indicando:

- Evento: en el que se selecciona “Creación de pedido”.
- Formato: en el que se selecciona JSON.
- URL de entrega: La cual se establecerá como el endpoint:
`URL_BASE:APP_PORT/api/purchase/shopify`

En caso de que la API de Winvisión se encuentre corriendo en el puerto 3000. Se deberá abrir este puerto en el router apuntando a la IP que corre la API de Winvisión.

- Versión de la API de webhooks: La última estable

Una vez finalizada la configuración de los ajustes de Shopify, se hará una petición GET al endpoint `URL_BASE:APP_PORT/api/item/export` para descargar un fichero csv con todos los artículos marcados como vendibles en la tienda online. También existe la posibilidad de ejecutar un script en consola para que se genere el fichero en el directorio “doc” de forma automática. Para ello el comando a ejecutar es “`npm run exportcsv`” (sin comillas).