PAGOPAR

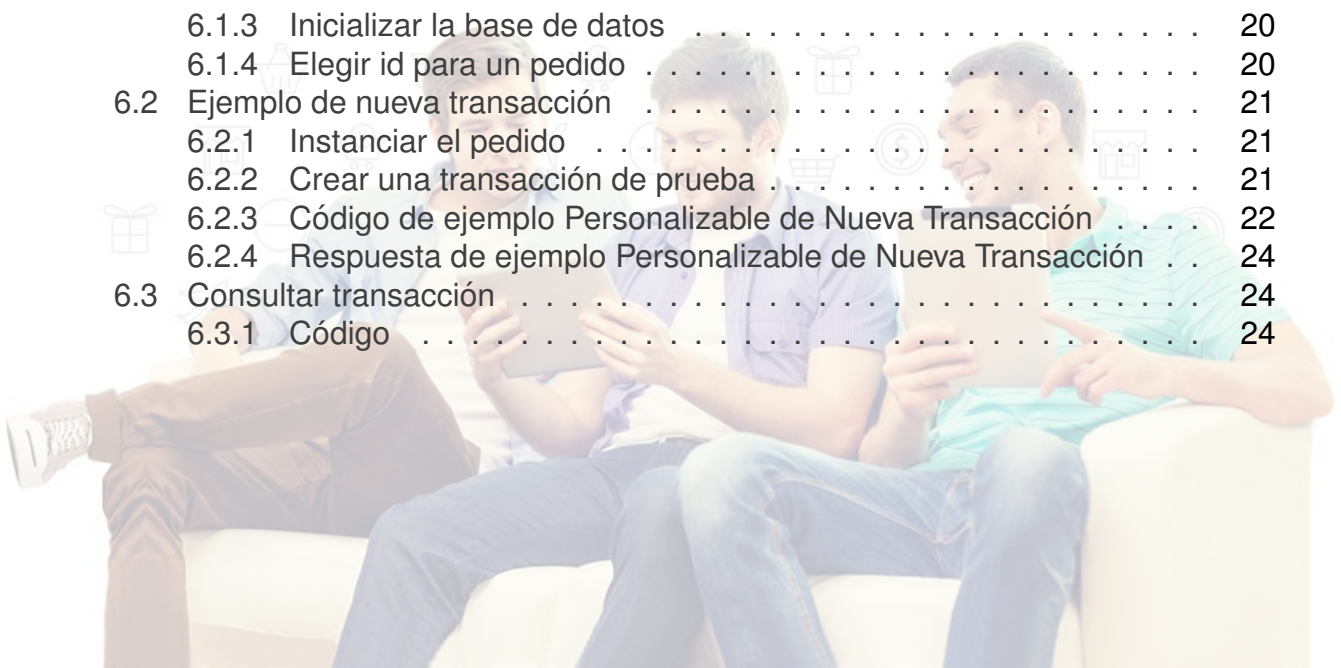
UN MÉTODO DE PAGO INTELIGENTE

DOCUMENTACIÓN TÉCNICA

Última actualización: 20/06/2017

Creado por: Pagopar <desarrollo@pagopar.com>

Tabla de Contenido



1 Nociones generales

1.1 ¿Qué es Pagopar?

Pagopar es la solución tecnológica que te brinda la posibilidad de vender online operando con todos los medios de pago sin tener que realizar ningun tipo de conexión con los mismos.

1.2 ¿Qué medios de pago se pueden utilizar?

Pagopar incluye los principales medios de pago en una sola plataforma:

TARJETAS DE CRÉDITO

Pago de forma segura desde cualquier parte del mundo con tarjetas de crédito Visa, Mastercard, Credicard y Unica.

BOCAS DE COBRANZA

Se realiza el pago en bocass de cobranza como Aquí Pago, Pago Express y Practipago.

HOMEBANKING

Pago desde cuentas bancarias a través del Homebanking de Itaú, BBVA, Regional, GNB, Atlas, Internist, Amambay, Bancop, Itapua, entre otros (Próximamente estaremos incluyendo más opciones de Homebanking).

TIGO MONEY

Pago utilizando la billetera de Tigo Money (Este método de pago estará habilitado próximamente).

PAGOPAR CARD

Pago utilizando la tarjeta de Pagopar Card con opciones de descuento y promociones.

2 ¿Cómo vender?

Para comenzar a vender en Pagopar seguimos los siguientes pasos:

1. Ingresar datos en el formulario de registro de Pagopar.
2. Verificar el mail de confirmación enviado a la cuenta de correo.
3. Integrar Pagopar al sitio web del comercio o crear un link de venta.

3 ¿Cómo comprar?

En la figura 1 podemos ver los pasos para completar el ciclo de compra de un producto en Pagopar.

1. **Ingresar al link de venta:** El usuario debe ingresar al navegador de cualquier dispositivo y pegar el link de venta del producto que quiere comprar ó, en el caso de que esté comprando en un sitio específico, el comercio le redireccionará al sitio de pago de Pagopar.
2. **Loguearse ó Registrarse:** El usuario puede realizar la acción de registro y/o logueo para que sus datos se completen automáticamente en los campos de formulario de compra.
3. **Elección de método de envío:** En el caso de que alguno o algunos de los productos cuenten con más de un método de envío[1], al usuario se le mostrará una página intermedia para que eliga el método que le parezca más adecuado para que así Pagopar pueda calcular el flete.
4. **Redirección a la aplicación web:** El usuario es redireccionado a la aplicación web, en la que debe seleccionar un método de pago (ver la sección 1.2).
5. **Completar formulario:** El usuario completa el formulario de pago con datos finales que deba requerir.
6. **Comunicación con los gateways[2] de pago:** Pagopar redirecciona a los demás sitios de pago en caso de haberse seleccionado un medio de pago distinto a Pagopar card. Ésta sería una pantalla intermedia entre la final de Pagopar y el gateway.

[1] En estos momentos sólo contamos con el "método de aex". Dentro de poco dispondremos de una opción de "envío propio".

[2] Un *payment gateway* o gateway de pago es un servicio web que autoriza el pago directo o por medio de tarjetas de crédito a un sitio web, aplicación o personas. Ejemplos de gateways serían Paypal, Amazon Pay, Procard y Bancard.

7. **Pantalla de pago final:** Se redirecciona a Pagopar con un mensaje de agradecimiento de compra y se le proporciona al usuario un certificado o número de su proceso de pago. Desde este punto el proceso de pago puede volver a repetirse.

Fig. 1: Proceso de pago de Pagopar

4 Ejemplos de Transacciones

4.1 Página de pago (Checkout) de Pagopar

Mostraremos un ejemplo de una compra realizada desde la página de un comercio. Usualmente debe existir una método previo de **checkout** de los producto a ser comprados por el usuario, asumiremos que dicho método existe y que todos los valores necesarios para redireccionar al sitio de Pagopar (Sean éstos: Datos del comprador y datos de los productos) son correctos.

Al iniciar la transacción, el comercio redirecciona a Pagopar.

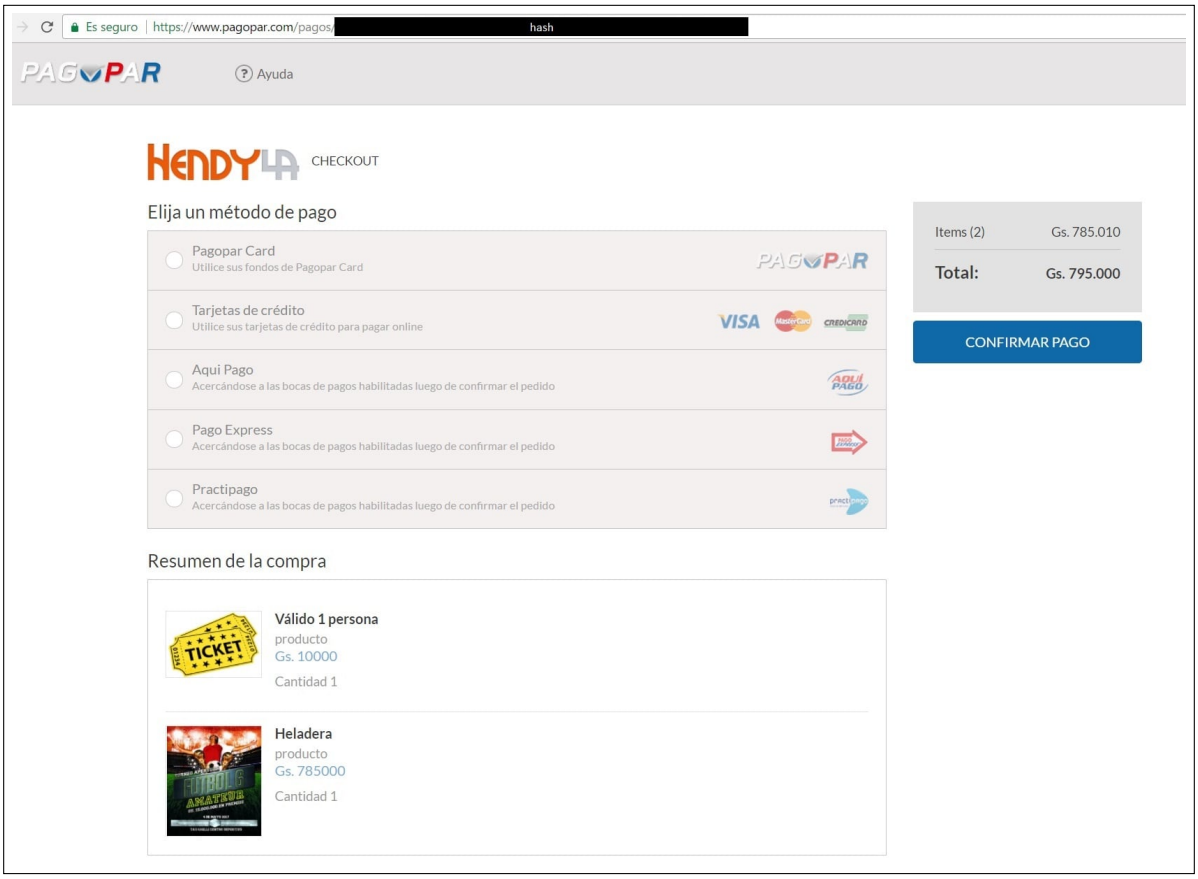


Fig. 2: Pantalla de pago de prueba de Pagopar (El comercio debe redireccionar a esta página una vez completados todos los campos necesarios para la compra)

En la Figura 2 podemos ver la pantalla de pago del sitio de Pagopar, la cual posee todos los campos de una compra de prueba, como el valor total de los productos (a la derecha, en el sidebar), una miniatura de los mismos y sus cantidades.

Nótese que la url del sitio debe poseer una cadena de caracteres al final llamada **hash**, la cual sirve para identificar la compra.

En la cabecera de la pantalla se puede ver el logo del comercio (en este caso de ejemplo, tenemos a Hendyla realizando la transacción de ejemplo), y más abajo vemos todos los métodos de pago disponibles (Pagopar card, Tarjetas de crédito, Aquí Pago, Pago Express y Practipago) con sus descripciones correspondientes.

Cuando se verifica que todos los campos son correctos podemos seleccionar algún método de pago para avanzar a la siguiente pantalla. Mostraremos un método de pago con **Tarjeta de crédito** y luego otro con **Aquí Pago**.

4.2 Tarjeta de Crédito

En el primer caso seleccionaremos la opción de Tarjetas de Crédito, y seremos redireccionados a la página de pago de **Payline**[3].

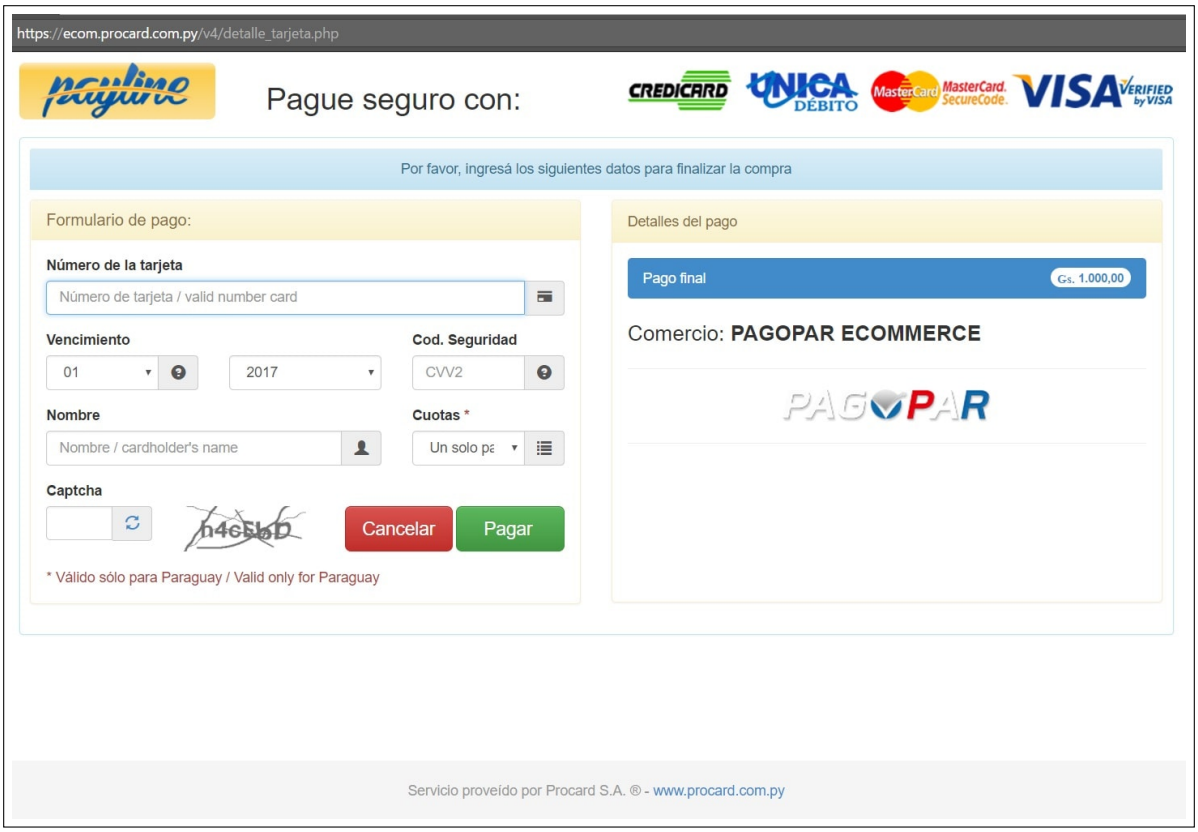


Fig. 3: Pantalla de ingreso de datos de Tarjeta de Crédito de Payline

[3]Payline es el servicio desarrollado para comercios adheridos a Procard. Más información aquí

En la Figura 3 vemos en la cabecera todos los métodos de pago posibles con Payline y, abajo a la izquierda, el formulario de pago que tiene los campos a ser llenados con la información de la Tarjeta de crédito, el Nombre del comprador, el número de cuotas y un pequeño captcha. A la derecha vemos los detalles del pago, con la cantidad total del pago, el comercio y el logo de Pagopar.

Una vez completados los datos, se da click en Pagar y si toda la transacción es correcta veremos la página de Transacción Exitosa de Payline (Figura 4), la cual posee los datos del monto de la compra y un **Número de Autorización** que, si bien será guardado por el sistema de Payline, se recomienda tomar nota del mismo ante posibles reclamos. De esta página seremos redireccionados automáticamente a la útlima pantalla de Pagopar (Figura 5), la cual es una pantalla de agradecimiento y brinda la opción de regresar al comercio.

Fig. 4: Pantalla de transacción exitosa de Payline

Nótese que al término de la url de la pantalla de la figura 5 tenemos una serie de caracteres conocida como **uuid**[4], la cual se guarda en Pagopar para una posterior identificación de la transacción.

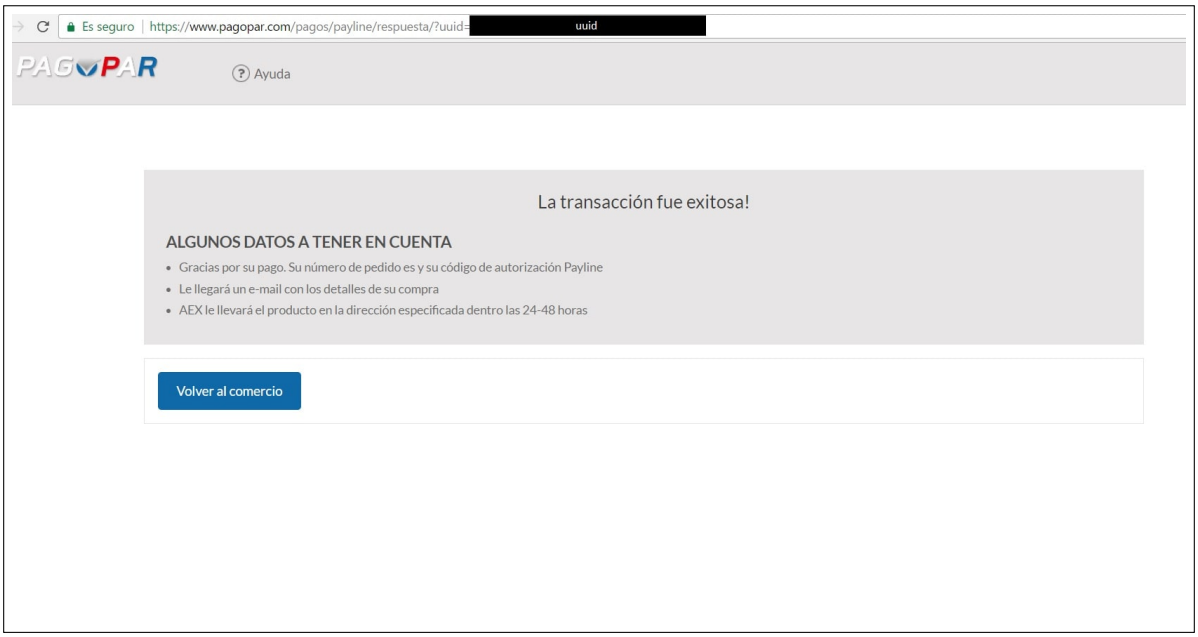


Fig. 5: Pantalla de agradecimiento de Pagopar

[4]Del inglés Universally Unique Identifier, es una cadena de 36 caracteres alfanuméricos agrupados por cuatro guiones y sirve para identificar una transacción hecha en un sistema de tarjeta de crédito (en este caso Payline). Siguen el siguiente formato: 8-4-4-4-12, ó, xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

4.3 Aquí Pago

El segundo ejemplo es el de la selección del método de pago de Aquí Pago, el cual es un proceso de pago mucho más corto (ya que el pago se realiza físicamente en alguna boca de cobranza).

En la Figura 6 vemos la pantalla de agradecimiento de Aquí Pago con las instrucciones necesarias para completar el proceso de compra.

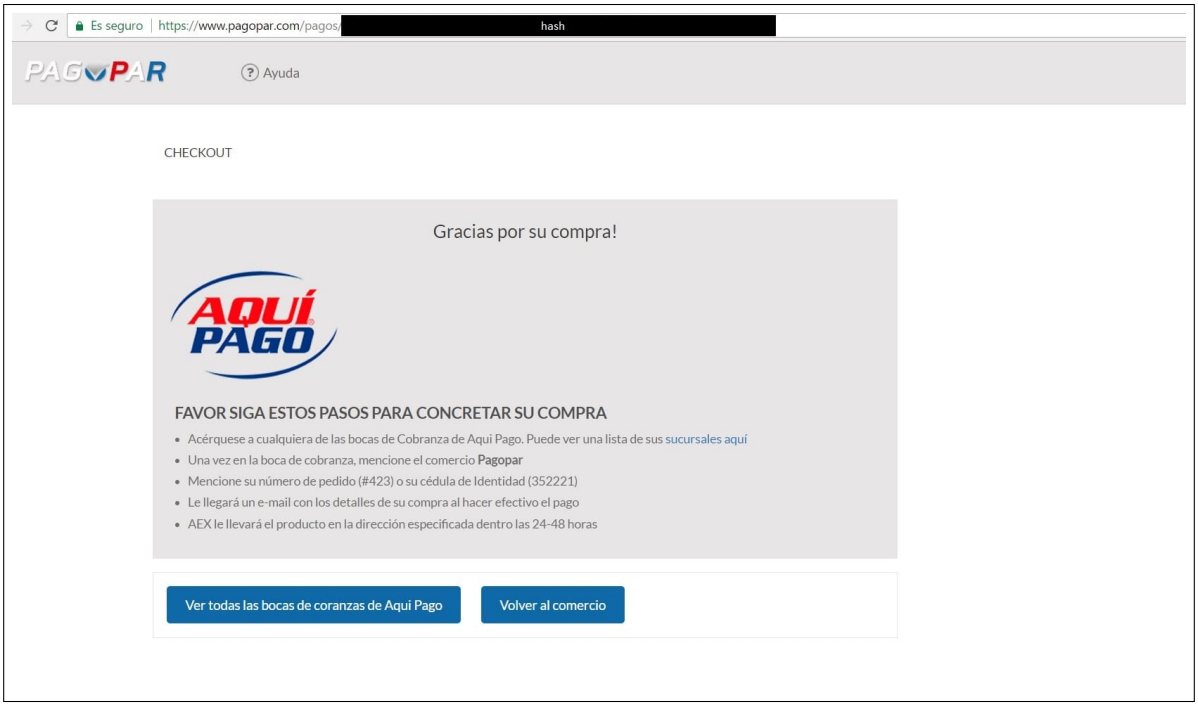


Fig. 6: Pantalla de agradecimiento de Pagopar con el método de pago Aquí Pago

5 API

La API de Pagopar puede ser utilizada con cualquier lenguaje de programación web.

Esta sección está orientada a desarrolladores, términos a tener en cuenta son: vector[5], cadena[6], entero[7], formato "Y-m-d H-i-s"[8] y SHA-1[9].

A continuación listamos y describimos los métodos necesarios para la integración con el servicio de Pagopar.

5.1 Generar transacción

Generar una nueva transacción es el método por el cual creamos un nuevo pedido de Pagopar. Para esto debemos realizar un *request*[10] cURL[11] a distintas urls del API de Pagopar. Los demás métodos que veremos más adelante siguen el mismo principio.

Los demás métodos que veremos más adelante siguen el mismo principio, deben invocar una función similar a runCurl (ver más adelante) con parámetros específicos (*$args* serían los parámetros del método y *$url* el link de pagopar que se necesita para ese método) y se obtiene como respuesta un JSON[12]. Nótese que la función similar a runCurl debe convertir los argumentos *$args* en un JSON antes de hacer el request cURL.

A continuación, vemos un ejemplo de cómo correr un request cURL a una url con el método POST en el lenguaje de programación PHP. La función *runCurl* recibe como los parámetros y la url a invocar. Retorna una cadena en formato JSON.

[5]Del Inglés, Array. Un Vector o Array es un arreglo ordenado de elementos dispuestos en filas o columnas. Por conveniencia sólo usaremos palabras en español para referirnos a términos de programación específicos.

[6]Es una serie de caracteres agrupados para formar un palabra o una oración

[7]Número entero. Cada vez que nos refiramos a un elemento como entero, queremos decir un número entero mayor a cero

[8]Formato de fecha utilizado en pagopar, indica "year-month-day hour-minute-second" ó, en español, "año-mes-día hora-minuto-segundo". El formato de fecha es el de 24 horas, esto es, 0 a 23 horas

[9]Del inglés, *Secure Hash Algorithm*, Algoritmo de Hash Seguro es un método para hacer cifrado de una cadena de caracteres. SHA-1 produce una cadena de caracteres de 160 bits (20 bytes). Más información aquí

[10]Inglés, significa petición o solicitud. Un request es una petición que realiza un servicio web a otro. De ahora en adelante usaremos este término cada vez que querramos indicar que se realiza una petición a un servicio web

[11]Es un software que permite la transferencia de datos entre distintos servicios webs. Más información aquí

[12]Del inglés JavaScript Object Notation, es un tipo de formato de archivo que utiliza una estructura de pares para poder leer un objeto. Más información aquí.

```php
<?php
    function runCurl($args, $url){
        $args = json_encode($args);

        $ch = curl_init();
        $headers= array('Accept:application/json',
        'Content-Type:application/json');
        curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
        curl_setopt($ch, CURLOPT_URL, $url);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
        curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
        curl_setopt($ch, CURLOPT_POST, true);
        curl_setopt($ch, CURLOPT_POSTFIELDS, $args);
        $response = curl_exec($ch);
        $error = curl_error($ch);
        curl_close($ch);

        return $response;
    }
?>
```

5.1.1 Parámetros

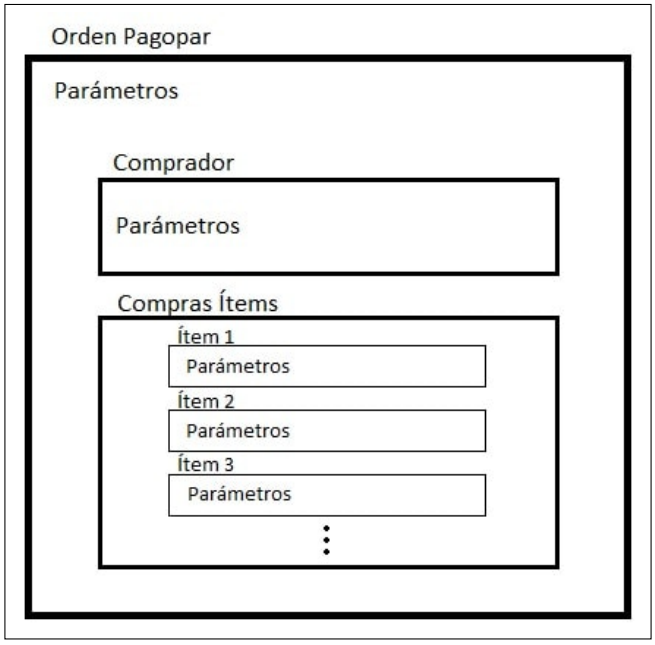


Fig. 7: Resumen de la estructura del nuevo pedido de Pagopar

En la figura 7 resumimos la estructura del vector de pedido que describiremos en las tablas 1 y 2. Una vez que tengamos el pedido de esta manera, debemos pasarlo como parámetro a una función similar a *runCurl*, la cual lo transformará a una cadena JSON y hará un request a la url de generar transacción de Pagopar.

En la tabla 1 podemos ver los parámetros necesarios para generar una nueva transacción (generar un nuevo pedido u orden, usaremos los términos indistintamente).

Método	Parámetro	Descripción
POST	orderPagopar	Vector con los siguientes índices: • tipo_pedido (cadena, Tipo de pedido) • fecha_maxima_pago (cadena de la fecha en formato ISO 8601 ó "Y-m-d H:i:s") • public_key (cadena, Clave pública) • id_pedido_comercio (entero, Id de un nuevo pedido) • monto_total (entero, Monto total de los ítems) • token (cadena, Hash SHA-1 del Pedido, utiliza la Clave privada y una cadena concatenada que especifica el tipo de transacción, para el caso de esta operación se usa la cadena "VENTA-COMERCIO") • descripcion_resumen (cadena, Descripción del nuevo pedido) • comprador (vector, Información del vendedor. Ver tabla 2) • compras_items (vector, Listado de los items del pedido. Ver tabla 2)
	link	Url de Pagopar a la que se debe hacer un request cURL con el método POST. **URL: https://api.pagopar.com/api/comercios/1.1/iniciar-transaccion**

Tab. 1: Parámetros para generar una nueva transacción

En la tabla 2 podemos ver los índices de los vectores de "comprador" y "compras_items" de la tabla 1. Vemos que el *token* que se debe enviar es una convinación de la clave pública y una cadena que indica el tipo de operación. Un ejemplo de cómo generar el token en PHP es éste:

```
<?php
    $token = sha1('clave_privada'.'tipo_de_token');
?>
```

Es importante resaltar que el vector de "compras_items" es un vector multidimensional (ver figura 7), donde cada sub-vector es un ítem distinto del pedido, por lo que en la descripción vemos los parámetros que debe tener cada uno de estos ítems.

Índices	Descripción
comprador	Vector con los siguientes índices: • nombre (cadena, Nombre del comprador. Obligatorio) • email (cadena, Correo electrónico del comprador. Obligatorio) • ciudad_id (entero, Identificador de la ciudad del comprador. Obligatorio) • telefono (cadena, Teléfono del comprador) • tipo_documento (cadena, Tipo del documento. Puede ser la palabra "CI", por ejemplo) • documento (cadena, Valor del documento del comprador) • direccion (cadena, Dirección del comprador) • direccion_referencia (cadena, Referencia de la dirección del comprador) • coordenadas (cadena, Coordenadas: latitud y longitud separadas por coma) de la dirección del comprador) • ruc (cadena, RUC del comprador) • razon_social (cadena, Razón social del comprador)
compras_items	Vector multidimensional, cada uno de sus sub-vectores (ítems, productos) posee los siguientes índices: • nombre (cadena, Nombre del producto. Obligatorio) • cantidad (entero, Cantidad de unidades del producto. Obligatorio) • precio_total (entero, Suma total de los precios de los productos (precio individual x cantidad. Obligatorio) • ciudad_id (entero, Identificador de la ciudad. Obligatorio) • descripcion (cadena, Descripción del producto) • url_imagen (cadena, Enlace a la imagen del producto) • peso (entero, Peso del producto) • largo (entero, Largo del producto) • ancho (entero, Ancho del producto) • alto (entero, Altura del producto) • categoria (entero, Identificador de la categoría del producto. Obligatorio) • producto_id (cadena, Identificador del producto. Elegido por el comercio. Obligatorio) • vendedor_telefono (cadena, Teléfono del vendedor) • vendedor_email (cadena, Correo electrónico del vendedor) • vendedor_direccion (cadena, Dirección del vendedor) • vendedor_direccion_referencia (cadena, Referencia de la dirección del vendedor) • vendedor_direccion_coordenadas (cadena, Coordenadas: latitud y longitud separadas por coma de la dirección del vendedor) • vendedor_clave_publica (cadena, Clave pública del vendedor. Obligatorio)

Tab. 2: Índices de comprador y compras_item

5.1.2 Respuesta

Todas las respuestas de Pagopar son enviadas en formato JSON y poseen dos campos, **respuesta**, un booleano, y **resultado**, los datos de respuesta de la operación.

En la tabla 3 observamos la estructura de la respuesta de una nueva transacción.

Formato	Atributo	Descripción
JSON	respuesta	Dato booleano (*true* ó *false*)
	resultado	Éxito: Vector con el campo *data* que contiene el Hash de la transacción. El hash tiene 64 caracteres. Ejemplo de hash: b92a3c6e319f08e49500328cbd342db19cf1cf07eab118414716a5f66d20cee3 Error: Cadena "Sin datos".

Tab. 3: Respuesta de la operación de generar nueva transacción

5.2 Consultar transacción

5.2.1 Parámetros

Método	Parámetro	Descripción
POST	argumentos	Vector con los siguientes índices: • hash_pedido (cadena, Hash proveído por Pagopar al finalizar el proceso de pago) • token (cadena, Hash, utiliza la Clave privada y la cadena "CONSULTA" concatenada • token_publico (cadena, Clave pública)
	link	Url de Pagopar a la que se debe hacer un request cURL con el método POST. **URL: https://api.pagopar.com/api/pedidos/1.1/traer**

Tab. 4: Parámetros para consultar un pedido

5.2.2 Respuesta

Formato	Atributo	Descripción
JSON	respuesta	Dato booleano (*true* ó *false*)
	resultado	Éxito: Vector, donde cada elemento tiene los siguientes campos: • pagado (booleano, Indica si se pagó el pedido) • forma_pago (cadena, Nombre del método de pago) • fecha_pago (cadena o null, Fecha de pago en formato ISO 8601. En caso de null, aún no se procedió al pago) • monto (cadena, Monto total del pedido, incluye los posibles fletes) • fecha_maxima_pago (cadena, Fecha máxima para el pago del pedido en formato ISO 8601. Al pasar este tiempo, el campo resultado valdrá *false*) • hash_pedido (cadena, Hash del pedido, se reenvía por motivos de comparación) Error: Cadena “Sin datos”

Tab. 5: Respuesta de la operación de consultar pedido

5.3 Consultar categorías

5.3.1 Parámetros

Método	Parámetro	Descripción
POST	argumentos	Vector con los siguientes índices: • token (cadena, Hash, utiliza la Clave privada y la cadena “CATEGORIAS” concatenada • token_publico (cadena, Clave pública)
	link	Url de Pagopar a la que se debe hacer un request cURL con el método POST. **URL: https://api.pagopar.com/api/categorias/1.1/traer**

Tab. 6: Parámetros para obtener las categorías

5.3.2 Respuesta

Formato	Atributo	Descripción
JSON	respuesta	Dato booleano (*true* ó *false*)
	resultado	Éxito: Vector, donde cada elemento tiene los siguientes campos: • categoría (cadena, Id de la categoría) • descripcion (cadena, Nombre de la categoría) • padre (cadena o null, Id de la categoría padre. En caso de null, la categoría es padre) • orden (cadena, Permite ordenar las categorías en tres niveles, sirve para el comercio. Por ejemplo: Productos (1), Servicios (2), Otros (3)) • producto_fisico (booleano, Indica si la categoría es de un producto físico, *true* ó *false*) • medidas (booleano, Indica si las medidas del producto son necesarias para la categoría, *true* ó *false*) Error: Cadena "Sin datos"

Tab. 7: Respuesta de la operación de consultar categorías

5.4 Consultar ciudades

5.4.1 Parámetros

Método	Parámetro	Descripción
POST	argumentos	Vector con los siguientes índices: • token (cadena, Hash, utiliza la Clave privada y la cadena "CIUDADES" concatenada • token_publico (cadena, Clave pública)
	link	Url de Pagopar a la que se debe hacer un request cURL con el método POST. **URL: https://api.pagopar.com/api/ciudades/1.1/traer**

Tab. 8: Parámetros para obtener las ciudades

5.4.2 Respuesta

Formato	Atributo	Descripción
JSON	respuesta	Dato booleano (*true* ó *false*)
	resultado	Éxito: Vector, donde cada elemento tiene los siguientes campos: • ciudad (cadena, Id de la ciudad) • descripcion (cadena, Nombre de la ciudad) Error: Cadena “Sin datos”

Tab. 9: Respuesta de la operación de consultar categorías

5.5 Calcular Flete

5.5.1 Parámetros

Método	Parámetro	Descripción
POST	argumentos	Vector con los siguientes índices: • token (cadena, Hash, utiliza la Clave privada y la cadena “CALCULAR-FLETE” concatenada) • token_publico (cadena, Clave pública) • dato (cadena, pedido en formato JSON. Los parámetros para el pedido son los mismos que los de la tabla 1)
	link	Url de Pagopar a la que se debe hacer un request cURL con el método POST. **URL: https://api.pagopar.com/api/calcular-flete/1.1/traer**

Tab. 10: Parámetros para calcular el flete de un pedido

6 SDK

El SDK (versión PHP) puede obtenerse descargando o clonando los archivos del repositorio de Pagopar[13]

6.1 Primeros pasos

6.1.1 Incluir la clase de Pagopar

Incluimos la clase Pagopar que se encuentra en el archivo *Pagopar.php*.

```
<?php
 require 'Pagopar.php';
?>
```

6.1.2 Usar una base de datos

Es necesario crear una base de datos MySQL o usar una ya existente antes de implementar el SDK de Pagopar.

6.1.3 Inicializar la base de datos

Instanciamos la clase *DBPagopar* con los datos necesarios para inicializar la base de datos. El SDK automáticamente crea una tabla llamada *transactions* en la que se guardarán los pedidos. Los parámetros a ser pasados son: *dbname* (el nombre de la base de datos), *dbuser* (el usuario) y *dbpass* (la contraseña).

```
<?php
    $db = new DBPagopar( "dbname" , "dbuser" , "dbpass" );
?>
```

Un ejemplo podría ser una base de datos llamada *dbpagopar*, con usuario *root* y sin contraseña:

```
<?php
    $db = new DBPagopar( 'dbpagopar' , 'root' , '' );
?>
```

6.1.4 Elegir id para un pedido

Debemos elegir un id para realizar un nuevo pedido o consultar una transacción realizada anteriormente. En ambos casos el id debe ser un entero mayor a cero.

En el caso de una nueva transacción, debemos elegir un valor superior al id del último pedido realizado.

[13] Próximamente tendremos disponibles otras herramientas para integrar el API a diferentes plataformas como Woocommerce, Magento, OpenCart, entre otros.

6.2 Ejemplo de nueva transacción

6.2.1 Instanciar el pedido

Lo primero que hacemos es crear una variable, en este caso *pedidoPagopar*, e instanciamos la clase de Pagopar, pasándole como parámetro el id del nuevo pedido[14].

```
<?php
    $pedidoPagoPar = new Pagopar(idNuevoPedido, $db);
?>
```

6.2.2 Crear una transacción de prueba

En el API se encuentra creado un método que realiza una transacción de prueba (ya con valores precargados) llamada *newTestPagoparTransaction*. Para nuestro ejemplo nos basta usar esta función, pero si se desea realizar una transacción con valores customizables se puede usar el código de la siguiente subsección, en dónde vemos todos los parámetros necesarios para pasar al método *newPagoparTransaction*.

```
<?php
    $pedidoPagoPar->newTestPagoparTransaction();
?>
```

Así, para generar un nuevo pedido de prueba de manera sencilla tenemos que:

```
<?php

    require 'Pagopar.php';

    $db = new DBPagopar( 'dbname' , 'dbuser' , 'dbpass' );

    /*Generar nuevo pedido*/
    //Id mayor al Id del ultimo pedido, solo para pruebas
    $idNuevoPedido = nuevo_id;
    //Generamos el pedido
    $pedidoPagoPar = new Pagopar($idNuevoPedido, $db);
    $pedidoPagoPar->newTestPagoparTransaction();

?>
```

[14] Número entero, ver la subsección 6.1.4 presentada anteriormente.

6.2.3 Código de ejemplo Personalizable de Nueva Transacción

El siguiente código posee los mismos valores que el método *newTestPagoparTransaction*, pero con algunas cadenas que deben ser reemplazadas por valores reales.

```
<?php
    require 'Pagopar.php';

    $db = new DBPagopar( 'dbname' , 'dbuser' , 'dbpass' );

    //Id mayor al Id del ultimo pedido, solo para pruebas
    $idNuevoPedido = nuevo_id;
    //Generamos el pedido
    $pedidoPagoPar = new Pagopar($idNuevoPedido, $db);

    //Creamos el comprador
    $buyer = new BuyerPagopar();
    $buyer->name            = "Juan Perez";
    $buyer->email           = 'correo_electronico';
    $buyer->cityId          = 1;
    $buyer->tel             = "0972200046";
    $buyer->typeDoc         = "CI";
    $buyer->doc             = "352221";
    $buyer->addr            = "Mexico 840";
    $buyer->addRef          = "al lado de Credicentro";
    $buyer->addrCoo         = "-25.2844638,-57.6480038";
    $buyer->ruc             = null;
    $buyer->socialReason    = null;
    //Agregamos el comprador
    $pedidoPagoPar->order->addPagoparBuyer($buyer);

    //Creamos los productos
    $item1 = new ItemPagopar();
    $item1->name                = "Valido 1 persona";
    $item1->qty                 = 1;
    $item1->price               = 10000;
    $item1->cityId              = 1;
    $item1->desc                = "producto";
    $item1->url_img             = "url_imagen";
    $item1->weight              = 0.1;
    $item1->category            = 3;
    $item1->productId           = "100";
    $item1->sellerPhone         = "0985885487";
    $item1->sellerAddress       = 'lorep ipsum';
    $item1->sellerAddressRef    = '';
    $item1->sellerAddressCoo    = '-28.75438,-57.1580038';
    $item1->sellerPublicKey     = "clave_publica";
```

```php
$item2 = new ItemPagopar();
$item2->name                = "Heladera";
$item2->qty                 = 1;
$item2->price               = 785000;
$item2->cityId              = 1;
$item2->desc                = "producto";
$item2->url_img             = "url_imagen";
$item2->weight              = 5.0;
$item2->width               = 2.0;
$item2->height              = 5.0;
$item2->large               = 5.0;
$item2->category            = 8;
$item2->productId           = "2";
$item2->sellerPhone         = '0985885487';
$item2->sellerAddress       = "lorep ipsum dolor";
$item2->sellerAddressRef    = '';
$item2->sellerAddressCoo    = '-28.75438,-57.1580038';
$item2->sellerPublicKey     = "clave_publica";

//Agregamos los productos al pedido
$pedidoPagoPar->order->addPagoparItem($item1);
$pedidoPagoPar->order->addPagoparItem($item2);

//Pasamos los parametros para el pedido
$pedidoPagoPar->order->publicKey = "clave_publica";
$pedidoPagoPar->order->privateKey = "clave_privada";
$pedidoPagoPar->order->typeOrder = 'VENTA-COMERCIO';
$pedidoPagoPar->order->desc = "Entrada Retiro";
$pedidoPagoPar->order->periodOfDaysForPayment = 1;
$pedidoPagoPar->order->periodOfHoursForPayment = 0;

//Obtiene las opciones de envio y, si es algun item que
//tiene opcion_envio metodo_aex y propio, entonces
//retornamos un json en el que el usuario debe seleccionar
//el metodo de pago para el item especifico
$json_pedido = $pedidoPagopar->getMethodsOfShipping();

if( !$json_pedido ){
    $pedidoPagopar->newPagoparTransaction();
}else{
    //Seleccionamos los metodos de envio y despues generamos
    //la nueva transaccion.
    //Metodo para seleccionar el envio, debe ser una pantalla
    //intermedia y hay que llamar a la funcion
    //newPagoparTransaction con un json como  parametro.
```

```
        //El json debe tener esta forma:
        // {
        //     id_producto_1 : metodo_seleccionado_1,
        //     id_producto_2 : metodo_seleccionado_2,
        //     ...           : ...
        // }
        $json = '{"100":"aex"}';
        $pedidoPagopar->newPagoparTransaction($json);
    }
?>
```

6.2.4 Respuesta de ejemplo Personalizable de Nueva Transacción

Fig. 8: Ejemplo de respuesta de una nueva transacción

6.3 Consultar transacción

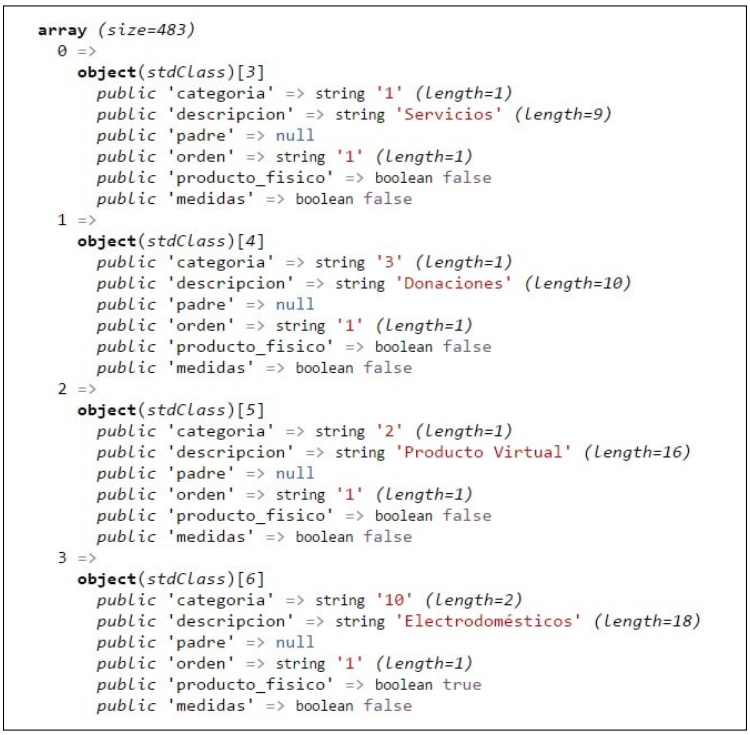
Para consultar un pedido realizado previamente, debemos conocer el *id*[15] del mismo. Una vez que lo tenemos, basta con llamar a *getPagoparOrderStatus*, la cual nos retorna un JSON (ver fig. 9) con las indicaciones del estado de la transacción.

6.3.1 Código

```
<?php
    $db = new DBPagopar( 'dbname' , 'dbuser' , 'dbpass' );
    /*Consultar pedido*/
    $idPedido = id_pedido;
    $pedidoPagoPar = new Pagopar($idPedido, $db);
    $pedidoPagoPar->publicKey = 'public_key';
    $pedidoPagoPar->privateKey = 'private_key';
    $pedidoPagoPar->getPagoparOrderStatus($idPedido);
?>
```

[15]Es posible mirar la base de datos generada por el SDK para obtener el *id* de un pedido, ver sección 6.1.3 para más datos de cómo usar la base de datos.

6.3.2 Respuesta



```
array (size=483)
  0 =>
    object(stdClass)[3]
      public 'categoria' => string '1' (length=1)
      public 'descripcion' => string 'Servicios' (length=9)
      public 'padre' => null
      public 'orden' => string '1' (length=1)
      public 'producto_fisico' => boolean false
      public 'medidas' => boolean false
  1 =>
    object(stdClass)[4]
      public 'categoria' => string '3' (length=1)
      public 'descripcion' => string 'Donaciones' (length=10)
      public 'padre' => null
      public 'orden' => string '1' (length=1)
      public 'producto_fisico' => boolean false
      public 'medidas' => boolean false
  2 =>
    object(stdClass)[5]
      public 'categoria' => string '2' (length=1)
      public 'descripcion' => string 'Producto Virtual' (length=16)
      public 'padre' => null
      public 'orden' => string '1' (length=1)
      public 'producto_fisico' => boolean false
      public 'medidas' => boolean false
  3 =>
    object(stdClass)[6]
      public 'categoria' => string '10' (length=2)
      public 'descripcion' => string 'Electrodomésticos' (length=18)
      public 'padre' => null
      public 'orden' => string '1' (length=1)
      public 'producto_fisico' => boolean true
      public 'medidas' => boolean false
```

Fig. 9: Ejemplo de retorno de *getPagoparOrderStatus*

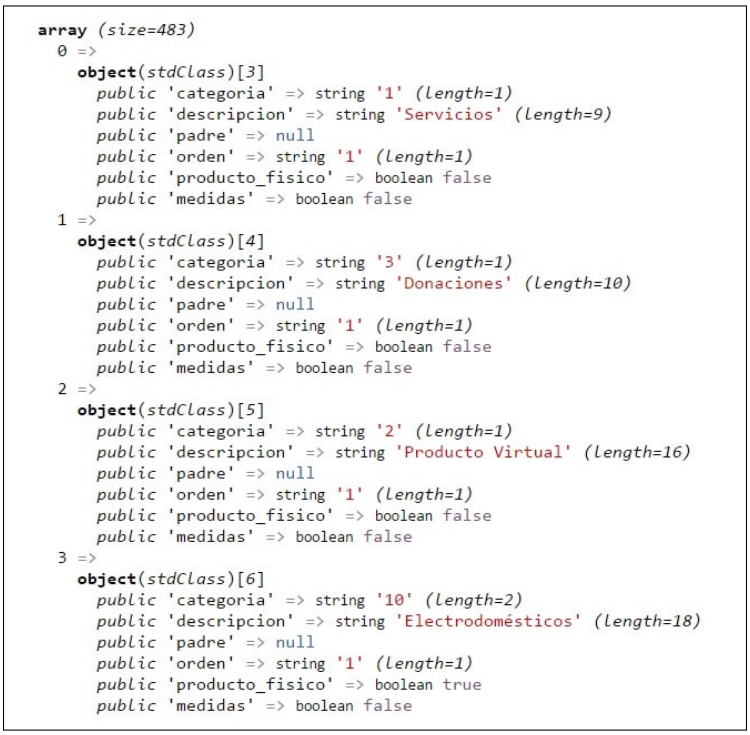
6.4 Consultar Categorías

Esta función no requiere del uso interno de una base de datos y retorna un vector de objectos con todas las categorías de los producto como indica la fig. 10. Para más detalles sobre los valores de retorno, ver la tabla 6.

6.4.1 Código

```php
<?php
    $consultaPagoPar = new ConsultPagopar();
    $consultaPagoPar->publicKey = 'public_key';
    $consultaPagoPar->privateKey = 'private_key';
    var_dump($consultaPagoPar->getProductCategories());
?>
```

6.4.2 Respuesta



```
array (size=483)
  0 =>
    object(stdClass)[3]
      public 'categoria' => string '1' (length=1)
      public 'descripcion' => string 'Servicios' (length=9)
      public 'padre' => null
      public 'orden' => string '1' (length=1)
      public 'producto_fisico' => boolean false
      public 'medidas' => boolean false
  1 =>
    object(stdClass)[4]
      public 'categoria' => string '3' (length=1)
      public 'descripcion' => string 'Donaciones' (length=10)
      public 'padre' => null
      public 'orden' => string '1' (length=1)
      public 'producto_fisico' => boolean false
      public 'medidas' => boolean false
  2 =>
    object(stdClass)[5]
      public 'categoria' => string '2' (length=1)
      public 'descripcion' => string 'Producto Virtual' (length=16)
      public 'padre' => null
      public 'orden' => string '1' (length=1)
      public 'producto_fisico' => boolean false
      public 'medidas' => boolean false
  3 =>
    object(stdClass)[6]
      public 'categoria' => string '10' (length=2)
      public 'descripcion' => string 'Electrodomésticos' (length=18)
      public 'padre' => null
      public 'orden' => string '1' (length=1)
      public 'producto_fisico' => boolean true
      public 'medidas' => boolean false
```

Fig. 10: Ejemplo de retorno de *getProductCategories*

6.5 Consultar Ciudades

Esta función no requiere del uso interno de una base de datos y retorna un vector de objectos con todas las categorías de los producto como indica la fig. 11. Para más detalles sobre los valores de retorno, ver la tabla 10.

6.5.1 Código

```php
<?php
    require 'Pagopar.php';

    $consultaPagoPar = new ConsultPagopar();
    $consultaPagoPar->publicKey = 'public_key';
    $consultaPagoPar->privateKey = 'private_key';
    var_dump($consultaPagoPar->getCities());
?>
```

6.5.2 Respuesta

```
array (size=127)
  0 =>
    object(stdClass)[3]
      public 'ciudad' => string '114' (length=3)
      public 'descripcion' => string '25 De Diciembre' (length=15)
  1 =>
    object(stdClass)[4]
      public 'ciudad' => string '182' (length=3)
      public 'descripcion' => string 'Acahay' (length=6)
  2 =>
    object(stdClass)[5]
      public 'ciudad' => string '207' (length=3)
      public 'descripcion' => string 'Alberdi' (length=7)
  3 =>
    object(stdClass)[6]
      public 'ciudad' => string '124' (length=3)
      public 'descripcion' => string 'Altos' (length=5)
  4 =>
    object(stdClass)[7]
      public 'ciudad' => string '109' (length=3)
      public 'descripcion' => string 'Antequera' (length=9)
```

Fig. 11: Ejemplo de retorno de *getCities*