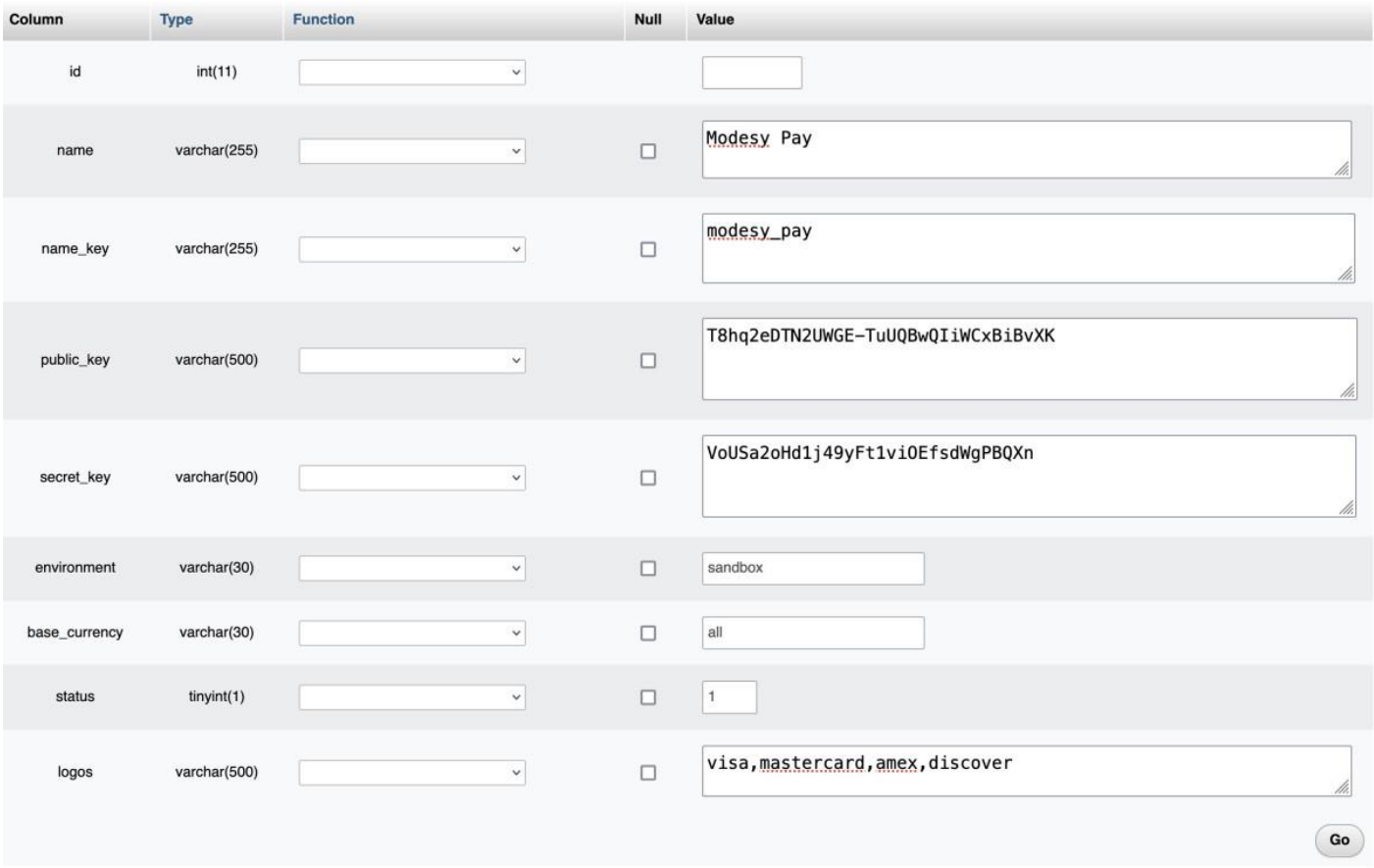
Adding a New Payment Gateway

There are 3 steps for adding a payment gateway:

1. Adding Your Payment Account to the Database

Payment systems generally work with API keys ("public_key" and "secret_key"). These key names may be different for some systems, but logically work the same way.
To add a new payment system, open the "**payment_gateways**" table in your database via phpMyAdmin and click the "**Insert**" link from the menu above to add your payment system.

Field	Description
id	Leave this field blank because the id is automatically generated.
name	Enter your payment system name. Example: Modesy Pay
name_key	Enter an unique name that does not contain special characters for your paymen Example: modesy_pay
public_key	Enter the public_key value you created in your payment account. Example: Adt9g4b-sz6NaQ_dsXGZQ76ah...
secret_key	Enter the secret_key value you created in your payment account. Example: EL6tE3pLtJ13JebQI5n3Fy57xL7...
environment	This is the payment mode option offered by your payment system. If you want to your payment system and perform payment tests, you must enter "**sandbox**" he your live API keys and receive payments after tests, you must enter "**productio** Example: sandbox
base_currency	If you want to accept payments in only one currency, you can write the short nar you want to receive payments in all active currencies on your site, you can ente the "**currencies**" table in your database for the short names of all currencies ava Example: USD
status	**1** or **0**. If you do not want the payment system to appear on your site, you need need to enter **1**. Example: 1
logos	Our script displays logos for each payment system on the payment methods pag located in the "assets / img / payment" folder. For the payment system you add, names you want to show by putting commas between them. You can upload yo folder. Example: visa,mastercard,amex,discover



After adding the necessary information, you can click the "**Go**" button and finish the adding process.

2. Adding Payment System Codes

Payment systems generally have 2 different integration methods:

A. JavaScript Integration: This method allows you to accept payments with the help of a popup or a previously created form. After the payment is completed, your payment system will return the payment information if the transaction is successful and you can complete the transaction by sending this information to the server side with the help of AJAX.

B. Web Redirect Integration: This method allows you to accept payments on an external web page. After the payment is completed, the payment system sends the payment result to the server side.

You need to read the documentation of the payment gateway you want to add, learn how it works, and decide which method you want to add. Payment methods usually have several integration methods.
After reading the API documentation of the payment gateway and understanding the

working logic, you can start the integration process.

1. Creating the View

First you need to add a "**view**" for the payment method. All HTML, CSS and JavaScript codes of the payment method should be added to this file. To do this, open the "**app/Views/cart/payment_methods**" folder and create a PHP file for your payment method. Your "**name_key**" value that in your database and your file name must be the same. For example, if your "name_key" value is "**modesy_pay**", you must create a file named "**_modesy_pay.php**".

2. Creating the Function

You need to add a function to the "**app/Controllers/CartController.php**" file to verify the payment and add the order to the database after the payment is made. Open this file and create your function. For example, if your "name_key" is "**modesy_pay**", you can create this fuction like this:

```
/**
 * Payment with Modesy Pay
 */
public function modesyPayPaymentPost()
{


}
```

You can add this code after the "**mercadoPagoPaymentPost**" function in this file.

3. Defining a Route to Access This Function

You need to define a route to access this fuction. To do this, open the "**app/Config/RoutesStatic.php**" file and add your route. For example, if the name of the function that you created is "**modesyPayPaymentPost**", you can add your route like this:

```
$routes->post('modesy-pay-payment-post',
'CartController::modesyPayPaymentPost');
```

Modesy has CSRF protection. When this protection is active, a request cannot be sent to your site from any external source via POST method. If your payment system sends the payment result to your site with POST method, this request may not reach your site. To solve this problem, you can exclude the requests coming from the route you created. To do this, open the "**app/Config/Filters.php**" file and add the route you created as in the example below.

Example:

```
    public array $globals = [
        'before' => [
            'csrf' => ['except' => ['modesy-pay-payment-post']],
        ],
    ...
```

After following these steps, you can add your payment gateway codes to the "_modesy_pay.php" file. The following PHP variables are available in this file and you need to use them for payment process:

- **$totalAmount:** The total amount of the items in the shopping cart. Amount to be paid. This is a decimal variable (Example: 10.50). Some payment systems require the payment amount as an integer number. In this case, you can multiply the total amount by 100.
- **$currency:** The currency of the payment. (Example: USD)
- **$mdsPaymentType**: There are 3 types of payments in the Modesy. There are "sale", "membership" and "promote" payments. You can use this variable if you need to make some differences according to payment type.
- **$mdsPaymentToken:** This variable has a unique token created for the payment. This token can be used to verify the payment.
- **$paymentGateway:** This variable is the payment method object chosen by the user. This variable contains all the values for the payment method you have added to the database. For example, if you want to access the "name_key" value for the selected payment method, you can access it with "$paymentGateway->name_key" code.

We will use PayPal JavaScript integration codes as an example to show how the payment gateway codes can be added to the "_modesy_pay.php" file.

The codes below are PayPal JavaScript integration codes. If you add these codes to the "_modesy_pay.php" file, you will see the PayPal payment button in the payment section. After the payment approved, you can send the paymet details to the function you created with AJAX POST method.

_modesy_pay.php

```
    <script src="https://www.paypal.com/sdk/js?client-id=<?=
$paymentGateway->public_key; ?>&currency=<?= $currency; ?>"></script>
    <script>
        paypal.Buttons({
            createOrder: function (data, actions) {
                return actions.order.create({
                    purchase_units: [{
                        amount: {
                            value: '<?= $totalAmount; ?>'
                        }
                    }]
                });
            },
            onApprove: function (data, actions) {
                return actions.order.capture().then(function (details) {
                    $('.paypal-loader').show();
                    var dataArray = {
                        'payment_id': data.orderID,
                        'currency': '<?= $currency; ?>',
                        'payment_amount': '<?= $totalAmount; ?>',
                        'payment_status': details.status,
                        'mds_payment_type': '<?= $mdsPaymentType; ?>'
                    };
                    $.ajax({
                        type: 'POST',
                        url: MdsConfig.baseURL + '/paypal-payment-post',
                        data: setAjaxData(dataArray),
                        success: function (response) {
                            var obj = JSON.parse(response);
                            if (obj.result == 1) {
                                window.location.href = obj.redirectUrl;
                            } else {
                                location.reload();
```

```
                            }
                        }
                    });
                });
            },
            onError: function (error) {
                alert(error);
            }
        }).render('#paypal-button-container');
    </script>
```

3. Verifying the Payment and Adding the Order to the Database

Once the payment has been received, you must verify this payment with its id before adding the order to the database. Each payment system has a different payment verification system. You can find the necessary codes in the API documentation of your payment system. You need to add these codes to the "**modesyPayPaymentPost**" function that you created in the "**app/Controllers/CartController.php**" file.

You can get the variables you send with AJAX POST
with **inputPost()** and **inputGet()** functions in the function you created. Also, if your payment system accepts the payments on an external web page and then returns the necessary variables, you can receive these variables in the same way.

*** First you need to check the payment method. If the payment method is not exist or enabled, you need to return an error.**

```
$paypal = getPaymentGateway('paypal');
if (empty($paypal)) {
    setErrorMessage("Payment method not found!");
    echo json_encode([
        'result' => 0
    ]);
    exit();
}
```

*** Then you need to validate the payment with the "payment_id" variable that PayPal returned to your function with POST method.**

```
$paymentId = inputPost('payment_id');
loadLibrary('Paypal');
$paypalLib = new \Paypal($paypal);
//validate the order
if ($paypalLib->getOrder($paymentId)) {
    //payment is valid
}
```

*** After validating the payment, you can add the order to the database by calling the "executePayment()" function.**

```
if ($paypalLib->getOrder($paymentId)) {
    $dataTransaction = [
        'payment_method' => 'PayPal',
        'payment_id' => $paymentId,
        'currency' => inputPost('currency'),
        'payment_amount' => inputPost('payment_amount'),
        'payment_status' => inputPost('payment_status'),
    ];
    $mdsPaymentType = inputPost('mds_payment_type');
    //add order
    $response = $this->executePayment($dataTransaction, $mdsPaymentType,
langBaseUrl());
    if ($response->result == 1) {
        setSuccessMessage($response->message);
        echo json_encode([
            'result' => 1,
            'redirectUrl' => $response->redirectUrl
        ]);
    } else {
        setErrorMessage($response->message);
        echo json_encode([
            'result' => 0
```

```
        ]);
    }
}
```

After combining all these, the function will be like this:

```
    /**
     * Payment with Paypal
     */
    public function paypalPaymentPost()
    {
        $paypal = getPaymentGateway('paypal');
        if (empty($paypal)) {
            setErrorMessage("Payment method not found!");
            echo json_encode([
                'result' => 0
            ]);
            exit();
        }
        $paymentId = inputPost('payment_id');
        loadLibrary('Paypal');
        $paypalLib = new \Paypal($paypal);
        //validate the order
        if ($paypalLib->getOrder($paymentId)) {
            $dataTransaction = [
                'payment_method' => 'PayPal',
                'payment_id' => $paymentId,
                'currency' => inputPost('currency'),
                'payment_amount' => inputPost('payment_amount'),
                'payment_status' => inputPost('payment_status'),
            ];
            $mdsPaymentType = inputPost('mds_payment_type');
            //add order
```

```
            $response = $this->executePayment($dataTransaction,
$mdsPaymentType, langBaseUrl());
            if ($response->result == 1) {
                setSuccessMessage($response->message);
                echo json_encode([
                    'result' => 1,
                    'redirectUrl' => $response->redirectUrl
                ]);
            } else {
                setErrorMessage($response->message);
                echo json_encode([
                    'result' => 0
                ]);
            }
        } else {
            setErrorMessage(trans("msg_error"));
            echo json_encode([
                'result' => 0
            ]);
        }
    }
```

Redirecting to the Order Details Page
Once the entire payment process is complete, our codes will return a "**redirectUrl**" variable. You need to redirect the page to this URL.

About Updates: We are constantly making new updates for our script. If we upload a major update, you will need to update all your files to update your site. After updating your site, if you copy your "_modesy_pay.php" file, your "modesyPayPaymentPost" function and your new route to the new version, your payment system will continue to work. Updates do not affect the database, so you do not need to add the same record to the database again. Also, if you've made an addition to the "**csrf**" array in the **"app/Config/Filters.php"** file for your payment system, don't forget to add it again for the new version as well.

Important: We have prepared this documentation to show you how you can add a payment system to our script. We have explained all the details you need to know about our codes. However, adding a payment system can be difficult because almost all payment systems work differently and have different codes. If you are unable to add your payment system by following these steps, or if you encounter errors while adding your payment system, we recommend that you to work with a developer. As we do not have a support or customization service, we will not be able to help you to add your payment system or solve the problems you see.