Frontend для формирования QR кода

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>QR</title>
</head>
<body>
<?php
/**
 * Форма для отправки запроса к бэкенду
  * для получения QR кода
*/?>
<form id="myForm" action="" method="POST">
    <input type="number" name="sum">
    <button type="submit">Отправить</button>
</form>
<?php
/**
 * Контейнер для вывода QR
 */?>
<div id="qrContainer"></div>

<script>
    const form = document.getElementById("myForm");
    const qrContainer = document.getElementById("qrContainer");

    <?php
    /**
     * Событие на отправку формы
     */?>
    form.addEventListener("submit", function (event) {
        event.preventDefault();

        const sum = form.elements.sum.value;
        const data = {
            sum: sum,
        };

        const jsonData = JSON.stringify(data);

        const xhr = new XMLHttpRequest();
        xhr.open("POST", "backend.php", true);
        xhr.setRequestHeader("Content-Type", "application/json");
        /**
         * Handles the onload event for the XMLHttpRequest.
         *
         * @param {Event} event - The onload event object.
         * @return {undefined} This function does not return a value.
         */
        xhr.onload = function () {
            if (xhr.status === 200) {
                <?php
                /**
                 * Обработка ответа, добавление QR кода в контейнер
                 */?>
                const response = JSON.parse(xhr.responseText);
                const qrImgUrl = response.results.qr_img;
                const qrImage = document.createElement("img");
                qrImage.src = qrImgUrl;
```

```
            qrImage.alt = "QR Code";
            qrContainer.innerHTML = "";
            qrContainer.appendChild(qrImage);
        } else {
            console.error(xhr.statusText);
        }
    };
    /**
     * Handles the onerror event for the xhr object.
     *
     * @param {type} - None
     * @return {type} - None
     */
    xhr.onerror = function () {
        console.error("Network error");
    };

    xhr.send(jsonData);
});
</script>

</body>
</html>
```

Backend (файл backend.php)

```
<?php
header("Content-Type: text/html; charset=UTF-8");
// Получение данных из тела запроса
$data = json_decode(file_get_contents('php://input'), true);

class curl {

    var $ch;
    var $httpget = '';
    var $head = '';
    var $is_post = false;
    var $postparams = null;
    var $httpheader = array ();
    var $cookie = array ();
    var $proxy = '';
    var $proxy_user_data = '';
    var $verbose = 0;
    var $referer = '';
    var $autoreferer = 0;
    var $writeheader = '';
    var $agent = 'Mozilla/5.0 (Windows NT 5.1; rv:23.0) Gecko/20100101 Firefox/23.0';
    var $url = '';
    var $followlocation = 1;
    var $returntransfer = 1;
    var $ssl_verifypeer = 0;
    var $ssl_verifyhost = 2;
    var $sslcert = '';
    var $sslkey = '';
    var $cainfo = '';
    var $cookiefile = '';
    var $timeout = 0;
    var $connect_time = 0;
    var $encoding = 'deflate';
    var $interface = '';
```

```
    function __construct (){
        $this->ch = curl_init();
        $this->set_httpheader(array('X-Requested-With: XMLHttpRequest', 'Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8','Accept-Language: ru-ru,ru;q=0.8,en-
us;q=0.5,en;q=0.3','Accept-Charset: windows-1251,utf-8;q=0.7,*;q=0.7'));
    }

    function get ($url){
        $this->url = $url;
        return $this->exec();
    }

    function post ($url, $postparams = null){
        $this->url = $url;
        $this->is_post = true;

        $this->postparams = $postparams;

        return $this->exec();
    }

    function set_httpget ($httpget){
        $this->httpget = $httpget;
    }

    function set_referer ($referer){
        $this->referer = $referer;
    }

    function set_autoreferer ($autoreferer){
        $this->autoreferer = $autoreferer;
    }

    function set_useragent ($agent){
        $this->agent = $agent;
    }

    function set_cookie (){
        preg_match_all('/Set-Cookie: (.*?)=(.*?);/i', $this->head, $matches, PREG_SET_ORDER);

        for ($i = 0; $i < count($matches); $i++) {
            if ($matches[$i][2] == 'deleted') {
                $this->delete_cookie($matches[$i][1]);
            } else {
                $this->cookie[$matches[$i][1]] = $matches[$i][2];
            }
        }
    }

    function add_cookie ($cookie){
        foreach ($cookie as $name => $value) {
            $this->cookie[$name] = $value;
        }
    }

    function delete_cookie ($name){
        if (isset($this->cookie[$name]))
            unset($this->cookie[$name]);
    }

    function get_cookie (){
        return $this->cookie;
    }
```

```
function clear_cookie (){
    $this->cookie = array ();
}

function set_httpheader ($httpheader){
    $this->httpheader = $httpheader;
}

function clear_httpheader (){
    $this->httpheader = array ();
}

function set_head ($head){
    $this->head = $head;
}

function set_encoding ($encoding){
    $this->encoding = $encoding;
}

function set_interface ($interface){
    $this->interface = $interface;
}

function set_writeheader ($writeheader){
    $this->writeheader = $writeheader;
}

function set_followlocation ($followlocation){
    $this->followlocation = $followlocation;
}

function set_returntransfer ($returntransfer){
    $this->returntransfer = $returntransfer;
}

function set_ssl_verifypeer ($ssl_verifypeer){
    $this->ssl_verifypeer = $ssl_verifypeer;
}

function set_ssl_verifyhost ($ssl_verifyhost){
    $this->ssl_verifyhost = $ssl_verifyhost;
}

function set_sslcert ($sslcert) {
    $this->sslcert = $sslcert;
}

function set_sslkey ($sslkey) {
    $this->sslkey = $sslkey;
}

function set_cainfo ($cainfo) {
    $this->cainfo = $cainfo;
}

function set_timeout ($timeout){
    $this->timeout = $timeout;
}

function set_connect_time ($connect_time){
    $this->connect_time = $connect_time;
}
```

```
function set_cookiefile ($cookiefile){
    $this->cookiefile = $cookiefile;
}

function set_proxy ($proxy){
    $this->proxy = $proxy;
}

function set_proxy_auth ($proxy_user_data){
    $this->proxy_user_data = $proxy_user_data;
}

function set_verbose ($verbose){
    $this->verbose = $verbose;
}

function get_error (){
    return curl_errno($this->ch);
}

function get_location (){
    $result = '';

    if (preg_match("/Location: (.*?)\r\n/is", $this->head, $matches)) {
        $result = end($matches);
    }

    return $result;
}

function get_http_code (){
    return curl_getinfo($this->ch, CURLINFO_HTTP_CODE);
}

function get_speed_download (){
    return curl_getinfo($this->ch, CURLINFO_SPEED_DOWNLOAD);
}

function get_content_type (){
    return curl_getinfo($this->ch, CURLINFO_CONTENT_TYPE);
}

function get_url (){
    return curl_getinfo($this->ch, CURLINFO_EFFECTIVE_URL);
}

function join_cookie() {
    $result = array ();
    foreach ($this->cookie as $key => $value)
        $result[] = "$key=$value";
    return join('; ', $result);
}

function exec (){
    curl_setopt($this->ch, CURLOPT_USERAGENT, $this->agent);
    curl_setopt($this->ch, CURLOPT_AUTOREFERER, $this->autoreferer);
    curl_setopt($this->ch, CURLOPT_ENCODING, $this->encoding);
    curl_setopt($this->ch, CURLOPT_URL, $this->url);
    curl_setopt($this->ch, CURLOPT_POST, $this->is_post);
    curl_setopt($this->ch, CURLOPT_FOLLOWLOCATION , $this->followlocation);
    curl_setopt($this->ch, CURLOPT_RETURNTRANSFER,$this->returntransfer);
    curl_setopt($this->ch, CURLOPT_SSL_VERIFYPEER, $this->ssl_verifypeer);
    curl_setopt($this->ch, CURLOPT_SSL_VERIFYHOST, $this->ssl_verifyhost);
```

```
        curl_setopt($this->ch, CURLOPT_HEADER, 1);
        curl_setopt($this->ch, CURLOPT_TIMEOUT, $this->timeout);
        curl_setopt($this->ch, CURLOPT_CONNECTTIMEOUT, $this->connect_time);
        curl_setopt($this->ch, CURLOPT_VERBOSE, $this->verbose);

        if ($this->referer)
            curl_setopt($this->ch, CURLOPT_REFERER, $this->referer);

        if ($this->interface)
            curl_setopt($this->ch, CURLOPT_INTERFACE, $this->interface);

        if ($this->httpget)
            curl_setopt($this->ch, CURLOPT_HTTPGET, $this->httpget);

        if ($this->writeheader != '')
            curl_setopt($this->ch, CURLOPT_WRITEHEADER, $this->writeheader);

        if ($this->is_post) {
            curl_setopt($this->ch, CURLOPT_POSTFIELDS, $this->postparams);
        }

        if ($this->proxy)
            curl_setopt($this->ch, CURLOPT_PROXY, $this->proxy);

        if ($this->proxy_user_data)
            curl_setopt($this->ch, CURLOPT_PROXYUSERPWD, $this->proxy_user_data);

        if ($this->cookie)
            curl_setopt($this->ch, CURLOPT_COOKIE, $this->join_cookie());

        if (count($this->httpheader))
            curl_setopt($this->ch, CURLOPT_HTTPHEADER, $this->httpheader);

        if ($this->sslcert)
            curl_setopt($this->ch, CURLOPT_SSLCERT, $this->sslcert);

        if ($this->sslkey)
            curl_setopt($this->ch, CURLOPT_SSLKEY, $this->sslkey);

        if ($this->cainfo)
            curl_setopt($this->ch, CURLOPT_CAINFO, $this->cainfo);

        if ($this->cookiefile) {
            curl_setopt($this->ch, CURLOPT_COOKIEFILE, $this->cookiefile);
            curl_setopt($this->ch, CURLOPT_COOKIEJAR, $this->cookiefile);
        }

        $response = curl_exec($this->ch);
        $this->set_head(substr($response, 0, curl_getinfo($this->ch, CURLINFO_HEADER_SIZE)));
        $response = substr($response, curl_getinfo($this->ch, CURLINFO_HEADER_SIZE));
        $this->set_cookie();

        $this->postparams = null;
        $this->is_post = false;

        return $response;
    }

    function __destruct (){
        curl_close($this->ch);
    }
}
class QR {
    var $curl = null;
```

```
var $domain = 'https://app.wapiserv.qrm.ooo/';
var $qrSize = 200;
var $paymentPurpose ='';
var $notificationUrl = '';
var $access_token = 'J8LUjbGJ.UBZksq5n9O443l3Nw93rbqSdDjIb0ZpZ';

private $data = array();

/**
 * Initializes a new instance of the class.
 *
 * @throws Some_Exception_Class description of exception
 * @return void
 */
function __construct() {

    // $this->domainAuth = $domainAuth;
    // $this->token = $token;
    $this->curl = new curl();
    $this->curl->set_httpheader(array('Content-Type: application/json'));
    $this->curl->set_cookiefile(__DIR__ . '/cookie.dat');

}
/**
 * Set the QR size.
 *
 * @param mixed $data The value to set as the QR size.
 */
public function setQrSize($data) {
    $this->qrSize = $data;
}
/**
 * Sets the payment purpose.
 *
 * @param mixed $data The payment purpose data.
 */
public function setPaymentPurpose($data) {
    $this->paymentPurpose = $data;
}
/**
 * Sets the notification URL.
 *
 * @param mixed $data The data to set as the notification URL.
 * @return void
 */
public function setNotificationUrl($data) {
    $this->notificationUrl = $data;
}

/**
 * Sets the data for the function.
 *
 * @param array $data The data to be set.
 * @return void
 */
function setData($data){
    if($this->qrSize>0){
        $data['qr_size'] = $this->qrSize;
    }
    if($this->paymentPurpose!=''){
        $data['payment_purpose'] = $this->paymentPurpose;
    }
    if($this->notificationUrl!=''){
        $data['notification_url'] = $this->notificationUrl;
    }
```

```php
        $this->data = $data;
    }
    /**
     * Creates a QR code with the given data.
     *
     * @param mixed $data The data to be encoded in the QR code.
     * @throws Some_Exception_Class A description of the exception that can be thrown.
     * @return mixed The decoded response from the API call.
     */
    function createQRcode($data){
        $this->setData($data);

        //echo'<pre>';print_r($this->data);echo'</pre>';
        $this->curl->set_httpheader(array('Content-Type: application/json', 'X-Api-Key: '.$this->access_token));
        $response = $this->curl->post($this->domain.'operations/qr-code/', json_encode($this->data));
        return json_decode($response);
    }
}



$QR = new QR();
$QR->setQrSize(400);
$QR->setPaymentPurpose('Иванов И.И. Договор №345567356324, плата за обучение');
$QR->setNotificationUrl('http://test.com/callback-qr-status/');
$res = $QR->createQRcode($data);

echo json_encode($res);
```