manual_de_integracion_pago_movil.pdf 1 15/11/22 2:26 p.m.

Manual de integración PaGo Móvil

credibanco

Versión 1.3

manual_de_integracion_pago_movil.pdf 2 15/11/22 2:26 p.m.

Historia de Revisiones

Fecha	Versión	Descripción	Autor
<24/01/2018>	1.0	Generación del documento	FM
<10/04/2018>	1.1	Ajustes sobre manual	FM, CV
<13/04/2018>	1.2	Ajustes sobre manual	FM
<14/06/2018>	1.3	Ajustes sobre manual	CV, FM

FM: Felipe Muñoz

CV: Camilo Valencia

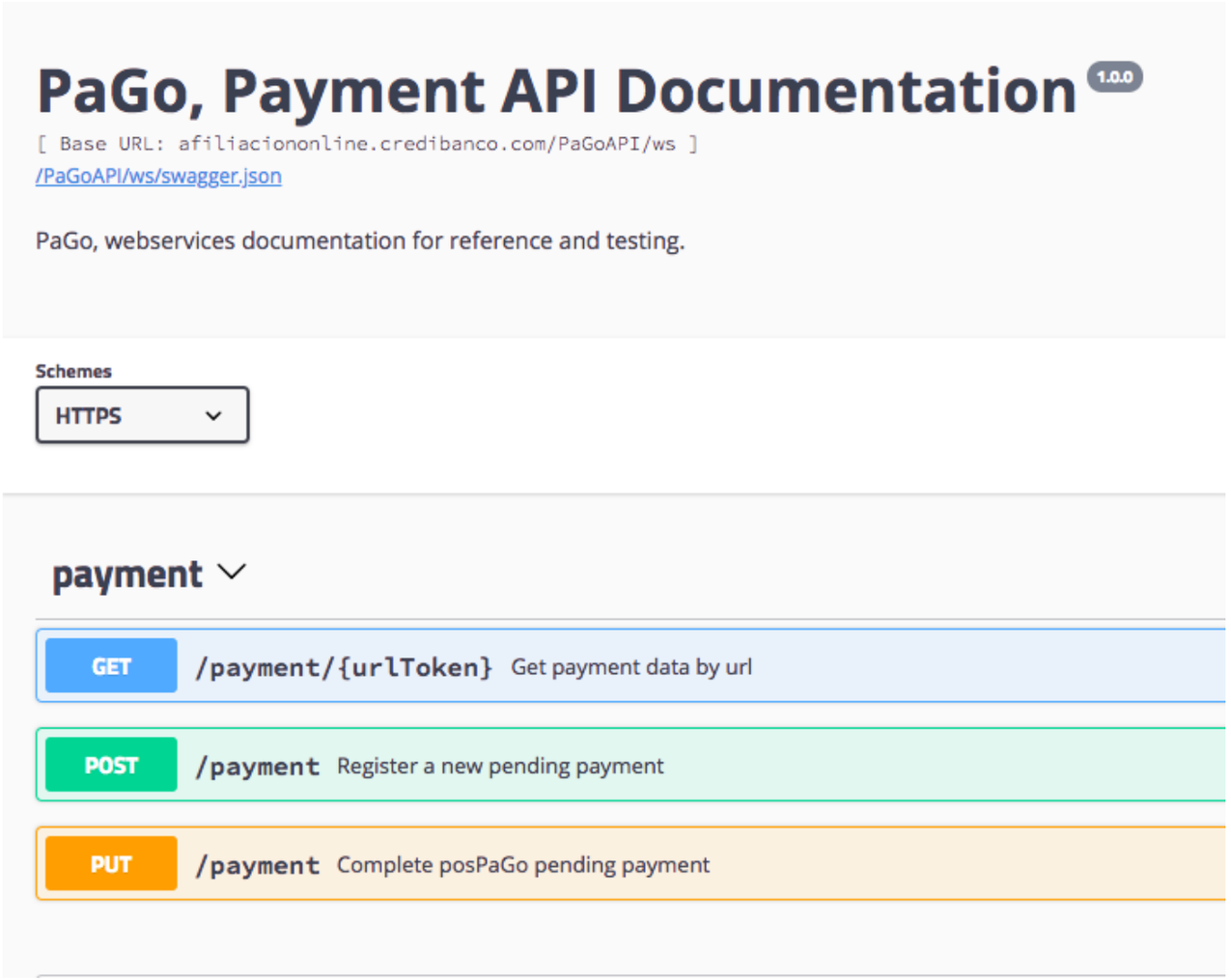
manual_de_integracion_pago_movil.pdf 3 15/11/22 2:26 p.m.

Documentación

La documentación referente a los servicios puede encontrarse en la siguiente ruta, que contiene el swagger con los servicios de implementación de PaGo.

Pruebas
https://afiliaciononline.credibanco.com/PaGoAPI/docs/#

Producción
https://pago.credibanco.com/PaGoAPI/docs/#



manual_de_integracion_pago_movil.pdf 4 15/11/22 2:26 p.m.

Implementación Android

Precondiciones Android

I. El comercio debe integrar dentro su aplicación el código fuente proporcionado por CredibanCo.
II. El comercio debe llamar a los servicios proporcionados por CredibanCo
III. La aplicación debe crear un WebView para publicar la interfaz de PaGo
IV. La aplicación debe tener un metodo en donde se reciba el resultado de la transacción para los fines de la aplicación.
V. La aplicación debe tener configurado el permiso para acceso a internet:

```
<uses-permission android:name="android.permission.INTERNET" />
```

manual_de_integracion_pago_movil.pdf 5 15/11/22 2:26 p.m.

Integración PaGo Android

1. Crear una clase PaGoLib Ej:

```
public class PaGoLib {

    final static String url = "https://"; //url provista por CredibanCo
    final static String bearerAuth = ""; //Codigo de Autenticación provisto por CredibanCo
    final static String uniqueCode = "0000000000"; //Codigo único provisto por CredibanCo
    static String responseWs, paymentState; //variables correspondientes al token de sesión y al
estado del pago
    static StringEntity entity;
    static WebView wV;

    //Método inicial para invocar la transacción de pago
    //Parámetros:
    //Activity activity : Actividad de Android desde la que se lanza la petición y sobre la que se
mostrará el WebView
    //[parametros]: Parámetros con los que se construirá el objeto JSON de la transacción
    public static void invokeWS(final Activity activity, Object [parametros]) {
        try {
            /*
             * Se crea el objeto JSON que se enviará al servicio para recibir la dirección del
pago
             * El formato del objeto JSON para la transaccion se relaciona en el ANEXO 1
             */
            JSONObject json = new JSONObject();
            if (!commerceName.equals("")) json.put("commerceName", commerceName);
            if (!observations.equals("")) json.put("observations", observations);
            if (!currencyCode.equals("")) json.put("currencyCode", currencyCode);
            if (!purchaseCode.equals("")) json.put("purchaseCode", purchaseCode);
            if (!totalAmount.equals("") && !posPago) json.put("totalAmount", totalAmount);
            if (posPago) {
                json.put("posPago", "true");
                json.put("totalAmount", 0);
                json.put("iacVal", 0);
                json.put("iva", 0);
            }
            if (!terminalCode.equals("")) json.put("terminalCode", terminalCode);
            if (!iva.equals("") && !posPago) json.put("iva", iva);
            if (!ref1.equals("")) json.put("ref2", ref2);
            if (!ref2.equals("")) json.put("ref3", ref3);
            if (!ref3.equals("")) json.put("ref1", ref1);
            if (!ref4.equals("")) json.put("ref4", ref4);
            if (!ref5.equals("")) json.put("ref5", ref5);
            if (!ref6.equals("")) json.put("ref6", ref6);
            if (!ref7.equals("")) json.put("ref7", ref7);
            if (!ref8.equals("")) json.put("ref8", ref8);
            if (!ref9.equals("")) json.put("ref9", ref9);
            if (!ref10.equals("")) json.put("ref10", ref10);
            if (!iacId.equals("")) json.put("iacId", iacId);
            if (!iacName.equals("")) json.put("iacName", iacName);
            if (!iacVal.equals("") && !posPago) json.put("iacVal", iacVal);
            if (!convenio.equals("")) json.put("convenio", convenio);
            if (!entidad.equals("")) json.put("entidad", entidad);

            if(airline != null) {
                JSONObject airlineDataJ = new JSONObject();
                airlineDataJ.put("idTaxFl", airline.getIdTaxFl());
                airlineDataJ.put("nameTaxFl", airline.getNameTaxFl());
                airlineDataJ.put("valueTaxFl", airline.getValueTaxFl());
                airlineDataJ.put("codeFl", airline.getCodeFl());
```

```
        airlineDataJ.put("codeAirline", airline.getCodeAirline());
        airlineDataJ.put("departDate", airline.getDepartDate());
        airlineDataJ.put("departHour", airline.getDepartHour());
        airlineDataJ.put("arriveDate", airline.getArriveDate());
        airlineDataJ.put("arriveHour", airline.getArriveHour());
        airlineDataJ.put("airportDeCode", airline.getAirportDeCode());
        airlineDataJ.put("countryDepCode", airline.getCountryDepCode());
        airlineDataJ.put("cityDepCode", airline.getCityDepCode());
        airlineDataJ.put("airportArriCode", airline.getAirportArriCode());
        airlineDataJ.put("countryArriCode", airline.getCountryArrCode());
        airlineDataJ.put("cityArriCode", airline.getCityArriCode());
        airlineDataJ.put("namePass1", airline.getNamePass1());
        airlineDataJ.put("lastPass1", airline.getLastPass1());
        airlineDataJ.put("documentPass1", airline.getDocumentPass1());
        airlineDataJ.put("namePass2", airline.getNamePass2());
        airlineDataJ.put("lastPass2", airline.getLastPass2());
        airlineDataJ.put("documentPass2", airline.getDocumentPass2());
        json.put("airlineData", airlineDataJ);
    }
    if(agency != null){
        JSONObject agencyDataJ = new JSONObject();
        agencyDataJ.put("idTaxFlAg", agency.getIdTaxFlAg());
        agencyDataJ.put("nameTaxFlAg", agency.getNameTaxFlAg());
        agencyDataJ.put("valueTaxFlAg", agency.getValueTaxFlAg());
        agencyDataJ.put("codeFlAg", agency.getCodeFlAg());
        agencyDataJ.put("codeAirlineFlAg", agency.getCodeAirlineFlAg());
        agencyDataJ.put("departDateFlAg", agency.getDepartDateFlAg());
        agencyDataJ.put("departHourFlAg", agency.getDepartHourFlAg());
        agencyDataJ.put("arriveDateFlAg", agency.getArriveDateFlAg());
        agencyDataJ.put("arriveHourFlAg", agency.getArriveHourFlAg());
        agencyDataJ.put("airportDeCodeAg", agency.getAirportDeCodeAg());
        agencyDataJ.put("countryDepCodeAg", agency.getCountryDepCodeAg());
        agencyDataJ.put("cityDepCodeAg", agency.getCityDepCodeAg());
        agencyDataJ.put("airportArriCodeAg", agency.getAirportArriCodeAg());
        agencyDataJ.put("countryArriCodeAg", agency.getCountryArriCodeAg());
        agencyDataJ.put("cityArriCodeAg", agency.getCityArriCodeAg());
        agencyDataJ.put("namePass1Ag", agency.getNamePass1Ag());
        agencyDataJ.put("lastPass1Ag", agency.getLastPass1Ag());
        agencyDataJ.put("documentPass1Ag", agency.getDocumentPass1Ag());
        agencyDataJ.put("namePass2Ag", agency.getNamePass2Ag());
        agencyDataJ.put("lastPass2Ag", agency.getLastPass2Ag());
        agencyDataJ.put("documentPass2Ag", agency.getDocumentPass2Ag());
        agencyDataJ.put("ivatasa", agency.getIvatasa());
        agencyDataJ.put("codetasa", agency.getCodeAirlineFlAg());
        agencyDataJ.put("valetasa", agency.getValetasa());
        json.put("agencyData", agencyDataJ);
    }
    if(restaurant != null)
    {
        JSONObject restaurantDataJ = new JSONObject();
        restaurantDataJ.put("idTaxRest", restaurant.getIdTaxRest());
        restaurantDataJ.put("nameTaxRest", restaurant.getNameTaxRest());
        restaurantDataJ.put("valueTaxRest", restaurant.getValueTaxRest());
        json.put("restaurantData", restaurantDataJ);
    }

    entity = new StringEntity(json.toString());
    entity.setContentType(new BasicHeader(HTTP.CONTENT_TYPE, "application/json"));

    //Se crea el cliente Http asíncrono con el que se realizará el llamado al servicio
    AsyncHttpClient client = new AsyncHttpClient();

    //Se agregan los headers correspondientes:
    //Código del comercio
    client.addHeader("Commerce", uniqueCode);
    //Código de autenticación
    client.addHeader("Authorization", bearerAuth);
```

manual_de_integracion_pago_movil.pdf 7 15/11/22 2:26 p.m.


```
            //Se realiza el llamado con el método POST al servicio de PaGo, disponiendo un Handler
para la respuesta
            client.post(activity.getApplicationContext(), url + "/ws/payment", entity,
"application/json", new AsyncHttpResponseHandler() {

                //Método de respuesta en caso de éxito en el llamado
                public void onSuccess(String response) {
                    try {

                        //Se crea un objeto que recibirá la respuesta del servicio
                        JSONObject obj = new JSONObject(response);

                        /*
                         * El formato del objeto JSON de la transaccion se relaciona en el ANEXO 2
                        */

                        //Si el código de respuesta es "01", el servicio se ejecutó correctamente
                        if (obj.getString("responseCode").equals("01")) {

                            //Captura del token de sesión del usuario
                            responseWs =
obj.getJSONObject("responseEntity").getString("urlToken");

                            //Creación y parametrización del WebView en el que se mostrará la
interfaz del pago
                            wV = new WebView(activity);
                            wV.setWebViewClient(new WebViewClient());
                            wV.setWebChromeClient(new WebChromeClient());
                            wV.getSettings().setJavaScriptEnabled(true);

                            //Carga de la interfaz del servicio
                            wV.loadUrl(url + "/pay?paymentId=" + responseWs);
                            activity.setContentView((wV));

                            //Llamado al servicio que realizará la comprobación del pago
                            invokeTrxWS(activity, responseWs);

                        }

                        //Si el código de respuesta es diferente a "01" ha ocurrido un error en el
servicio
                        else
                        //Sección para controlar errores en el llamado al servicio
                    }
                    //Captura de excepción en caso de problemas con la librería de JSON
                    catch (JSONException e) {
                        e.printStackTrace();
                    }
                }

                //Método de respuesta en caso de falla en el llamado
                //Los parámetros para este método son:
                //int statusCode: Código de estado para el error, los valores más comunes son 404
(Servicio no encontrado) o 500 (Error de servidor)
                //Throwable error: Error provisto por el método
                //String content: Contenido de error provisto por el método
                @Override
                public void onFailure(int statusCode, Throwable error, String content) {

                    //En esta sección deben implementarse las acciones en el evento de falla del
servicio
                    if (statusCode == 404) {
                    } else if (statusCode == 500) {
                    } else {
                    }
                }
```

```
            });
        }
        //Captura de excepción en caso de problemas con la librería de JSON
        catch (JSONException e) {
            e.printStackTrace();
        }
        //Captura de excepción en caso de problemas con la codificación del objeto JSON
        catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    }

    //Método que realiza la repetición de la consulta al servicio de pago
    //Los parámetros para este método son:
    //Activity activity : Actividad sobre la cual se realiza el llamado al servicio de estado del
pago
    private static void checkPayment(final Activity activity) {
        //Se instancia una cuenta regresiva con duración de 2 segundos para realizar la consulta
al servicio
        new CountDownTimer(5000, 1000) {

            public void onTick(long millisUntilFinished) {
            }

            public void onFinish() {
                invokeTrxWS(activity, responseWs);
            }
        }.start();
    }

    //Método que realiza la llamada al servicio de estado del Pago
    //Los parámetros para este método son:
    //Activity activity : Actividad sobre la cual se realiza el llamado al servicio de estado del
pago
    //String token : Token de sesión del usuario
    public static void invokeTrxWS(final Activity activity, String token) {
        //Se crea el cliente Http asíncrono con el que se realizará el llamado al servicio
        AsyncHttpClient client = new AsyncHttpClient();
        //Se agregan los headers correspondientes:
        //Código del comercio
        client.addHeader("Commerce", String.valueOf(uniqueCode));
        //Código de autenticación
        client.addHeader("Authorization", bearerAuth);

        //Se realiza el llamado con el método GET al servicio de PaGo, disponiendo un Handler para
la respuesta
        client.get(url + "/ws/payment/" + token, new AsyncHttpResponseHandler() {

            //Método de respuesta en caso de éxito en el llamado
            public void onSuccess(String response) {
                try {
                    //Se crea un objeto que recibirá la respuesta del servicio
                    JSONObject obj = new JSONObject(response);

                    /*
                     * El formato del objeto JSON para la transaccion se relaciona en el ANEXO 3
                     */
                    //Si el código de respuesta es "01", el servicio se ejecutó correctamente
                    if (obj.getString("responseCode").equals("01")) {
                      //Captura del estado del pago
                      paymentState =
obj.getJSONObject("responseEntity").getString("paymentState");

                        //Si el código de transacción es 00, el pago se realizó exitosamente, si
el código es “NoCompra”, es una transacción PosPago en espera de terminación
                        if (paymentState.equals("00") || paymentState.equals("NoCompra")) {
                            if(wV != null)
                                wV.destroy();
```

```
                            if(paymentState.equals("NoCompra"))
                                //Sección para controlar el pago en espera de terminación

                            else
                                //Sección para controlar el pago exitoso

                        }
                        //Si el código de transacción es P, la transacción no se ha llevado a cabo
                        else if (paymentState.equals("P"))
                            checkPayment(activity);
                    //Sección para implementar acciones de acuerdo al código de transacción
recibido
                        //else

                     }

                     //Si el código de respuesta es diferente a "01" ha ocurrido un error en el
servicio
                     else {
                       //Sección para controlar errores en el llamado al servicio
                     }
                }
                //Captura de excepción en caso de problemas con la librería de JSON
                catch (JSONException e) {
                    e.printStackTrace();
                    checkPayment(activity);
                }
            }

            //Método de respuesta en caso de falla en el llamado
            //Los parámetros para este método son:
            //int statusCode: Código de estado para el error, los valores más comunes son 404
(Servicio no encontrado) o 500 (Error de servidor)
            //Throwable error: Error provisto por el método
            //String content: Contenido de error provisto por el método
            @Override
            public void onFailure(int statusCode, Throwable error, String content) {
                //En esta sección deben implementarse las acciones en el evento de falla del
servicio
                if (statusCode == 404) {

                } else if (statusCode == 500) {

                } else {

                }
            }
        });
    }


    //Método que realiza la llamada al servicio para la finalización de una transacción PosPaGo
    //Los parámetros para este método son:
    //Activity activity : Actividad sobre la cual se realiza el llamado al servicio de estado del
pago
    //String totalAmount : Valor total de la transacción
    //String terminalCode : Codigo de la terminal sobre la que se realiza la transacción
    //String iva : Iva de la transacción
    //String iac : Iac de la transacción

    public static void finishPosPagoTrx(final Activity activity, String totalAmount, String
terminalCode, String iva, String iac)
    {

        try {

            //Se crea el cliente Http asíncrono con el que se realizará el llamado al servicio
```

```
            AsyncHttpClient client = new AsyncHttpClient();
            //Se agregan los headers correspondientes:
            //Código del comercio
            client.addHeader("Commerce", String.valueOf(uniqueCode));
            //Código de autenticación
            client.addHeader("Authorization", bearerAuth);

            /*
             * El formato del objeto JSON para la transaccion se relaciona en el ANEXO 4
             */

            JSONObject json = new JSONObject();
            if (!totalAmount.equals("")) json.put("totalAmount", totalAmount);
            if (!terminalCode.equals("")) json.put("terminalCode", terminalCode);
            if (!iva.equals("")) json.put("iva", iva);
            if (!iac.equals("")) json.put("iacVal", iac);
            entity = new StringEntity(json.toString());
            entity.setContentType(new BasicHeader(HTTP.CONTENT_TYPE, "application/json"));


            //Se realiza el llamado con el método PUT al servicio de PaGo, disponiendo un Handler
para la respuesta
          client.put(activity.getApplicationContext(), url + "/ws/payment/" + responseWs, entity,
"application/json", new AsyncHttpResponseHandler() {

              //Método de respuesta en caso de éxito en el llamado
              public void onSuccess(String response) {
                  try {
                      //Se crea un objeto que recibirá la respuesta del servicio
                      JSONObject obj = new JSONObject(response);

                     //Si el código de respuesta es "01", el servicio se ejecutó correctamente
                     if (obj.getString("responseCode").equals("01")) {
                      //Captura del estado del pago
                     paymentState = obj.getJSONObject("responseEntity").getString("paymentState");
                            //Sección para controlar el pago exitoso
                            //El formato del objeto JSON recibido se encuentra en el ANEXO 3
}

               //Si el código de respuesta es diferente a "01" ha ocurrido un error en el servicio
                         else {
                        //Sección para controlar errores en el llamado al servicio
                         }
                     }
                     //Captura de excepción en caso de problemas con la librería de JSON
                     catch (JSONException e) {
                         e.printStackTrace();
                     }
                 }

                 //Método de respuesta en caso de falla en el llamado
                 //Los parámetros para este método son:
                 //int statusCode: Código de estado para el error, los valores más comunes son 404
(Servicio no encontrado) o 500 (Error de servidor)
                 //Throwable error: Error provisto por el método
                 //String content: Contenido de error provisto por el método
                 @Override
                 public void onFailure(int statusCode, Throwable error, String content) {
             //En esta sección deben implementarse las acciones en el evento de falla del servicio
                     if (statusCode == 404) {

                     } else if (statusCode == 500) {

                     } else {

                     }
                 }
             });
```

manual_de_integracion_pago_movil.pdf 11 15/11/22 2:26 p.m.


```
        }
        //Captura de excepción en caso de problemas con la librería de JSON
        catch (JSONException e) {
            e.printStackTrace();
        }
        //Captura de excepción en caso de problemas con la codificación del objeto JSON
        catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

2. Para la clase PaGoLib se deben realizar los siguientes import:

```
import android.app.Activity;
import android.os.CountDownTimer;
import android.webkit.WebChromeClient;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.TextView;

import org.apache.http.entity.StringEntity;
import org.apache.http.message.BasicHeader;
import org.apache.http.protocol.HTTP;
import org.json.JSONException;
import org.json.JSONObject;

import com.credibanco.sm.pagoplusdummy.entities.*;
import com.loopj.android.http.AsyncHttpClient;
import com.loopj.android.http.AsyncHttpResponseHandler;

import java.io.UnsupportedEncodingException;
```

3. Dentro del metodo de pago se debe instanciar la clase PaGoLib y llamar a los métodos invokeWS() y finishPosPagoTrx(), pasando los parámetros necesarios.

```
PaGoLib lib = new PaGoLib();

lib.invokeWS(MainActivity.this, [parametros]);

lib.finishPosPagoTrx(MainActivity.this, total, terminal, iva, iac);
```

Para acceder a los datos de la transacción, se pueden encapsular para su consulta.

manual_de_integracion_pago_movil.pdf 12 15/11/22 2:26 p.m.

Implementación iOS

Precondiciones iOS

I. El comercio debe integrar dentro su aplicación el código fuente proporcionado por CredibanCo.
II. El comercio debe llamar a los servicios proporcionados por CredibanCo
III. La aplicación debe crear un WebView para publicar la interfaz de PaGo
IV. La aplicación debe tener un metodo en donde se reciba el resultado de la transacción para los fines de la aplicación.
V. La aplicación debe tener la siguiente etiqueta en el archivo info.plist:

```
<key>NSAppTransportSecurity</key>
<dict>
   <key>NSAllowsArbitraryLoads</key>
   <true/>
</dict>
```

manual_de_integracion_pago_movil.pdf 13 15/11/22 2:26 p.m.

Integración PaGo iOS

1. Crear los siguientes métodos en la clase que implementa el pago:

```
//Realiza el llamado al primer servicio, obteniendo el token de sesión del
usuario
    func callUrlWithValue()
    {
        /*
         * Se crea el objeto JSON que se enviará al servicio para recibir la
dirección del pago:
             * El formato del objeto JSON para la transaccion se relaciona en el
ANEXO 1         */

        let jsonData = try? JSONSerialization.data(withJSONObject: dict, options:
[.prettyPrinted]);

        //Se crea el request al servicio usando el método POST
        let request = NSMutableURLRequest(url: URL(string: _serviceUrl +
"/ws/payment")!);
        request.httpMethod = "POST";

        //Se incluye el objeto JSON como cuerpo del mensaje y se agregan las
cabecerasa
        request.httpBody = jsonData;
        request.addValue("application/json", forHTTPHeaderField:"Content-Type")
        request.setValue(_serviceAuth, forHTTPHeaderField:"Authorization")
        request.setValue(String(_uniqueCode), forHTTPHeaderField: "Commerce")

        //Se ejecuta la tarea usando el request creado
        let task = URLSession(configuration:
URLSessionConfiguration.default).dataTask(with: request as URLRequest,
completionHandler: {(data, response, error) in

            //Se formatea el objeto data recibido para trabajar con el contenido
            let json = try? JSONSerialization.jsonObject(with: data!, options:
.allowFragments) as! [String:AnyObject]
            /*
                            * El formato del objeto JSON para la transaccion se
relaciona en el ANEXO 2*/

            //Si el código de respuesta es "01" se ejecutó correctamente el
método
            if("01" == (json!["responseCode"] as! String))
            {
                //Se extrae el token de la respuesta para definir el servicio de
consulta del estado del pago
                let responseEntity = json!["responseEntity"] as! NSDictionary;
                token = responseEntity["urlToken"] as! String;
                _url = _serviceUrl + "/pay?paymentId=" + token;

                //Se crea el webview con la interfaz de pago en el hilo
principal, y se carga con la url generada a partir del token de autenticación
                DispatchQueue.main.async(group: DispatchGroup.init(), execute: {
                    wView = UIWebView(frame: CGRect(x: 0, y: 0, width:
UIScreen.main.bounds.width, height: UIScreen.main.bounds.height));
```

manual_de_integracion_pago_movil.pdf 14 15/11/22 2:26 p.m.


```
                    wView.loadRequest( URLRequest(url: URL(string: _url)!));

                    self.view.addSubview(wView);
                    self.invokeTrxWS()
                });
            }
            else
            {
                //En esta sección deben implementarse las acciones en el evento
de falla del servicio
            }
        });
        task.resume()
    }

    //Método que dispara la validación del estado del pago
    func checkPayment() {
        //Se crea el Timer con un intervalo de 5 segundos para evitar usar
demasiado el tráfico del teléfono
        timer = Timer.scheduledTimer(withTimeInterval: 5, repeats: false, block:
{ (void) in self.invokeTrxWS() })
        timer.fire()
    }

    //Método que realiza el llamado al servicio que comprueba el estado de la
transacción
    func invokeTrxWS()
    {
        //Se crea el request con el que se consultará el estado del pago, por
medio del método GET
        let request = NSMutableURLRequest(url: URL(string: _serviceUrl +
"/ws/payment/"+token)!);
        request.httpMethod = "GET";
        //Se agregan las cabeceras de autenticación y formato del servicio
        request.addValue("application/json", forHTTPHeaderField:"Content-Type")
        request.setValue(_serviceAuth, forHTTPHeaderField:"Authorization")
        request.setValue(String(_uniqueCode), forHTTPHeaderField: "Commerce")

        //Se inicia la tarea que llamará al servicio
        let task = URLSession(configuration:
URLSessionConfiguration.default).dataTask(with: request as URLRequest,
completionHandler: {(data, response, error) in

            //Se formatea el objeto data recibido para trabajar con el contenido
            let json = try? JSONSerialization.jsonObject(with: data!, options:
.allowFragments) as! [String:AnyObject]
            /*
            * El formato del objeto JSON para la transaccion se relaciona en el
ANEXO 3
             */

            //Si el código de respuesta es "01" se ejecutó correctamente el
método
            if("01" == (json!["responseCode"] as! String))
            {
                //Se extraen los datos del estado del pago
                let responseEntity = json!["responseEntity"] as! NSDictionary;
                let paymentState = responseEntity["paymentState"] as!
NSDictionary;
```

```
                //Si el estado del pago es 3, el pago se realizó correctamente
                if((paymentState["paymentStateId"] as! Int64) == 3)
                {
                    //Se retorna al hilo principal para realizar el cerrado del
WebView y la ejecución de otras tareas pertinentes
                    DispatchQueue.main.async(group: DispatchGroup.init(),
execute: { wView.removeFromSuperview(); });
                }
                    //Si el código de transacción es 1, la transacción no se ha
llevado a cabo
                else if((paymentState["paymentStateId"] as! Int64) == 1)
                { self.checkPayment() }
                else{
                    //Sección para implementar acciones de acuerdo al código de
transacción recibido
                }
            }
            else
            {
                //En esta sección deben implementarse las acciones en el evento
de falla del servicio
            }
        });
        task.resume()
    }
```

2. Crear las siguientes variables en la misma clase:

```
var _url = String();
let _serviceUrl = "https://pago.credibanco.com/PaGoAPI";
let _serviceAuth = "Bearer de autenticación provisto por CredibanCo ";
var _uniqueCode = Código único provisto por CredibanCo en formato numérico;
var timer = Timer();
var token = String();
var wView = UIWebView();
```

3. Realizar el llamado al método en el evento deseado:

```
callUrlWithValue();
```

manual_de_integracion_pago_movil.pdf 16 15/11/22 2:26 p.m.

Códigos de Respuesta del Servicio

Los siguientes son los códigos de respuesta que provee el servicio en el campo paymentState. Estos son los códigos que entrega VPayment en la autorización del pago.

codRespuesta	Message
00	Aprobada
01	Negada, comuniquese con su entidad
02	Negada, comuniquese con su entidad
03	Negada, comercio inválido
04	Negada, retener tarjeta
05	Negada, puede ser tarjeta bloqueada o timeout
06	Negada, no se pudo procesar la transacción
07	Negada, retener tarjeta
08	Aprobada, solicitar más información
09	Negada, transacción duplicada
11	Aprobada, vip
12	Negada, transacción inválida
13	Negada, monto inválido
14	Negada, estado de la tarjeta inválido
15	Negada, la institución no está en el IDF
16	Negada, Numero cuotas invalidas
30	Negada, error en edición de mensaje
31	Negada, el emisor no es soportado por el Sistema
33	Negada, tarjeta vencida con orden de retención
34	Negada, retener/capturar
35	Negada, retener/capturar
36	Negada, retener tarjeta
37	Negada, tarjeta bloqueada retener/capturar
38	Negada, número de intentos del PIN excedidos

manual_de_integracion_pago_movil.pdf 17 15/11/22 2:26 p.m.

39	Negada, puede ser tarjeta bloqueada o timeout
41	Negada, tarjeta robada o extraviada
43	Negada, estado en archivo de tarjetahabientes CAF
51	Fondos insuficientes
54	Negada, tarjeta vencida
55	Negada, PIN inválido
56	Negada, no se encontro CAF
57	Negada, transacción no permitida a esta tarjeta
58	Negada, transacción Inválida
61	Negada, excede el monto máximo
62	Negada, tarjeta restringida
65	Negada, límite de usos por período excedido
68	Negada, TIMEOUT
70	Negada, tarjeta vencida
71	Negada, El tipo de cuenta no corresponde
75	Negada, número de intentos de PIN excedidos
76	Aprobada, (Privado)
77	Aprobada, pendiente identificación firma del comp
78	Aprobada a ciegas
79	Aprobada, transacción administrativa
80	Aprobada por boletín de seguridad
81	Aprobada por el establecimiento
82	Negada, no hay módulo de seguridad
83	Negada, no hay cuenta para la tarjeta
84	Negada, no existe el archivo de saldos PBF
85	Negada, error en actualización de archivo saldo
86	Negada, tipo de autorización errado
87	Negada, track 2 errado
88	Negada, error en log de transacciones PTLF

manual_de_integracion_pago_movil.pdf 18 15/11/22 2:26 p.m.

89	Negada, inválida la ruta de servicio
90	Negada, no es posible autorizar
91	Negada, no es posible autorizar
92	Negada, puede ser tarjeta bloqueada o timeout
93	Negada, no es posible autorizar
94	Negada, transacción duplicada
96	Negada, no se pudo procesar la transacción
97	Negada, Número de documento inválido
98	Negada, CVV2 invalido
99	Error de comunicaciones
B1	Numero de Factura invalida
B2	Factura vencida
B3	Factura pagada
F1	Filtro por Comercio
F2	FILTRO POR AGENCIA
F3	Bin no permitido para esta aerolínea
No	Negada, no es posible autorizar
N1	Negada, longitud del número de PAN inválida
N2	Negada, se llenó el archivo de preautorizaciones
N3	Negada, límite de retiros en línea excedido
N4	Negada, límite retiros fuera de línea excedido
N5	Negada, límite de crédito por retiro excedido
N6	Negada, límite de retiros de crédito excedido
N7	Negada, customer selected negative file reason
N8	Negada, excede límite de piso
N9	Negada, maximum number of refund credit
00	Negada, referral file full
01	Negada, NEG file problem
02	Negada, advances less than minimum

manual_de_integracion_pago_movil.pdf 19 15/11/22 2:26 p.m.

03	Negada, delinquent
04	Negada, over limit table
05	Negada, PIN required
06	Negada, mod 10 check
07	Negada, force post
08	Negada, bad PBF
09	Negada, NEG file problem
P0	Negada, CAF problem
P1	Negada, over daily limit
P3	Negada, advance less than minimum
P4	Negada, number times used
P5	Negada, delinquent
P6	Negada, over limit table
P7	Negada, advance less than minimum
P8	Negada, administrative card needed
P9	Negada, enter lesser amount
Q0	Negada, invalid transaction date
Q1	Negada, Fecha de vencimiento invalida
Q2	Negada, invalid transaction code
Q3	Negada, valor del avance menor que el mínimo
Q4	Negada, excedido el número de usos por período
Q5	Negada, delinquent
Q6	Negada, tabla de límites excedida
Q7	Negada, el valor excede al máximo
Q8	Negada, no se encuentra la tarjeta administrativa
Q9	Negada, tarjeta administrativa no está permitida
R0	Negada, transacción admin aprobada en ven
R1	Negada, transacción admin aprobada fuera
R2	Negada, transacción administrativa aprobada

manual_de_integracion_pago_movil.pdf 20 15/11/22 2:26 p.m.

R3	Negada, la transacción Chargeback es aprobada
R4	Negada, devolución/drchivo de usuario actualizado
R5	Negada, devolución/número de prefijo incorrecto
R6	Negada, devolución código de rspta incorrecto
R7	Negada, transacción administrativa no soportada
R8	Negada, la tarjeta está en archivo de negativos
S4	Negada, PTLF full
S5	Negada, devolución aprobada, archivo de cliente n
S6	Negada, devolución aprobada, archivo de cliente n
S7	Negada, devolución aceptada, destino incorrecto
S8	Negada, ADMIN file problem
S9	Negada, unable to validate PIN, security module is
T1	Negada, tarjeta de crédito inválida
T2	Negada, fecha de transacción inválida
T3	Negada, card not supported
T4	Negada, amount over maximum
T5	Negada, CAF status = 0 or 9
T6	Negada, Bad UAF
T7	Negada, límite diario excedido en el Cash back
T8	Negada, el enlace esta caido
T0	Negada, time out
1001	Invalid Data sent by Commerce
1002	Invalid Currency for Commerce
1003	Invalid Commerce Code
1004	Duplicated order number
1101	Communication Problem
1102	Processing Problem
1103	Communication Problem
1104	Invalida Data

manual_de_integracion_pago_movil.pdf 21 15/11/22 2:26 p.m.

1107	Invalid VCI
1114	Card Number does not belong to selected brand
2100	Implementor Class not existent
2101	Default Plan Quotas inexistent
2102	Existing BIN for the acquirer but does not have Plan Quotas assigned
2200	Acquirer must need more data sent in the Plug-In
2201	Plan Quotas not sent by Commerce
2202	Duplicated order number
2308	Card Brand is not correct
2309	Card Number is not correct
2310	Card Expiry Date is not correct
2311	Card Security Code is not correct
2312	Card Number is not present and is required
2313	Commerce not well configured
2314	Acquirer not well configured
2315	Will not go to authorization due to Pre-Authorization Rules
2316	Transaction has been authorized but Post-Authentication Rules refused it
2317	Transaction has been authorized but has been reversed by Post-Authentication Rules
2303	PurchaseCode Field must not be greater than 12 characters
2305	Commerce is not active
2306	Acquirer is not active
2307	Transaction has been processed
2318	PurchaseOperationNumber is greater than 8 characters (Maximum for Associated Commerce)
2319	PurchaseOperationNumber is greater than 12 characters
2400	Transaction is rejected and will not be authorized due to pre-authorization rules
2401	Pre Authentication rules not approved

manual_de_integracion_pago_movil.pdf 22 15/11/22 2:26 p.m.

2402	Error processing authorization
2403	Maximum Monthly accumulated amount has been reached
2404	Maximum Daily accumulated amount has been reached
2405	Maximum Daily order number has been reached
2406	Maximum Monthly order number has been reached
2410	Mall cannot start transactions
2411	Shipper is needed but not found
2412	Shipping amount is not valid
2413	Purchase amount is not valid
2414	Associated Commerce amount is not valid
2415	Purchase Amount and Associated Commerce amount are not valid
2417	Iva amount is not valid
2418	Dev Iva amount is not valid
2500	Invalid amount format
2501	Purchase amount is less than iva
2502	Maximum iva exceeded
2503	Tip amount is not valid
2504	Maximum tip exceeded
2505	Airport tax is not valid
2506	Maximum airport tax exceeded
2507	Purchase amount is less than iva + dev iva
3001	Rejected by Ecommerce rule
3002	Rejected by Ecommerce rule Error\t
3003	Rejected by Ecommerce rule Comm. Error
5000	Certificate does not belong to commerceId sent
2320	Commerce does not belong to Acquirer Sent
2304	Commerce does not exist
2203	Currency does not exists
2203	Currency does not exists

manual_de_integracion_pago_movil.pdf 23 15/11/22 2:26 p.m.

4006	Financial associated with the card not registered in the system
4007	Airline is not registered in the system
CB01	Lo sentimos, la transacción fue bloqueada por listas.
CB02	Lo sentimos, la transacción no puede ser procesada por parámetros de seguridad.
CB03	Lo sentimos, la transacción fue bloqueada por reintentos denegados.
CB04	Lo sentimos, la transacción no puede ser procesada Filtro por Agencia.
CB05	Lo sentimos, BIN no permitido para esta aerolínea.

manual_de_integracion_pago_movil.pdf 24 15/11/22 2:26 p.m.

Servicio PosPaGo

Este servicio consiste en una transacción que se envía sin valor donde el cliente selecciona el método de PaGo para consumir el servicio, una vez seleccionado el método de pago, el comercio invoca un servicio donde envía el valor de la transacción para finalizar el pago.

Inicialización del pago:

La implementación se realiza con el mismo flujo de una compra e invocando el mismo servicio, pero en el campos **posPago** se envía un "true". En los campos de valor de la transacción (totalAmount, iva, iac) se debe enviar el valor o.

```
{
  ...
  "totalAmount": "0", //Total de la compra
  ...
  "iva": "0", //IVA
  ...
  "iacVal": "0", //Valor del IAC
  "posPago": "true", // Pospago
  ...
}
```

El comercio abre la url de pago dentro del webview y el usuario podrá seleccionar un método de pago. Y seguir con el flujo de una transacción normal con la diferencia de que no se hará ninguna autorización de pago.

Consulta del pago:

Se consume el mismo servicio de consulta de pago y si la selección de forma de pago fue exitosa en el campo paymentState debe aparecer el código **NoCompra** y en el campo paymentStateDesc el valor **Esquema Pospago**.

Finalización del PaGo

Se debe consumir el webservice descrito para finalizar la transacción y obtener los datos de autorización del pago.
Metodo PUT, path: /payment/[urlToken]

El cuerpo del mensaje se encuentra en el ANEXO 4
Este servicio devuelve el mismo resultado del servicio obtener pago con el estado actual de la transacción

manual_de_integracion_pago_movil.pdf 25 15/11/22 2:26 p.m.

Anexo 1

*Campos Obligatorios

```
{
  "commerceName": "string", //Nombre del comercio
  "observations": "string", //Observaciones del pago
  "currencyCode": "string", //Código de la moneda *
  "purchaseCode": "string", //Código de compra
  "totalAmount": "string", //Total de la compra *
  "terminalCode": "string", //Código de la terminal *
  "iva": "string", //IVA *
  "ref1": "string", //Referencia 1
  "ref2": "string", //Referencia 2
  "ref3": "string",//Referencia 3
  "ref4": "string", //Referencia 4
  "ref5": "string", //Referencia 5
  "ref6": "string", //Referencia 6
  "ref7": "string", //Referencia 7
  "ref8": "string", //Referencia 8
  "ref9": "string", //Referencia 9
  "ref10": "string", // Referencia 10
  "iacId": "string", //Id del IAC
  "iacName": "string", //Nombre del IAC
  "iacVal": "string", //Valor del IAC
  "posPago": "string", // Pospago
  "convenio": "string", //Convenio
  "entidad": "string", //Entidad
//Si el comercio es una aerolínea, se añade el siguiente campo, de lo contrario se puede omitir
  "airlineData": {
    "idTaxFI": "string", //Id cargo
    "nameTaxFI": "string", //Nombre cargo
    "valueTaxFI": "string", //Valor cargo
    "codeFI": "string", //Codigo de vuelo
    "codeAirline": "string",  //Codigo de la aerolinea
    "departDate": "string", //Fecha del vuelo
    "departHour": "string", //Hora del vuelo
    "arriveDate": "string", //Fecha de llegada
    "arriveHour": "string", //Hora de llegada
    "airportDeCode": "string", //Aeropuerto de salida
    "countryDepCode": "string", //Pais de salida
    "cityDepCode": "string", //Ciudad de salida
    "airportArriCode": "string", //Aeropuerto de llegada
    "countryArriCode": "string", //Pais de llegada
    "cityArriCode": "string", //Ciudad de llegada
    "namePass1": "string", //Nombre pasajero
    "lastPass1": "string", //Apellido pasajero
```

manual_de_integracion_pago_movil.pdf 26 15/11/22 2:26 p.m.


```
    "documentPass1": "string",//Documento pasajero
    "namePass2": "string",//Nombre pasajero 2
    "lastPass2": "string",//Apellido pasajero 2
    "documentPass2": "string" //Documento pasajero 2
  },
//Si el comercio es una agencia se incluye el siguiente campo, de lo contrario, se puede omitir
  "agencyData": {
    "idTaxFlAg": "string", //Id cargo
    "nameTaxFlAg": "string", //Nombre cargo
    "valueTaxFlAg": "string", //Valor cargo
    "codeFlAg": "string", //Codigo agencia
    "codeAirlineAg": "string", //Codigo aerolinea
    "departDateAg": "string", //Fecha de salida
    "departHourAg": "string", //Hora de salida
    "arriveDateAg": "string", //Fecha de llegada
    "arriveHourAg": "string", //Hora de llegada
    "airportDeCodeAg": "string", //Aeropuerto de salida
    "countryDepCodeAg": "string", //Pais de salida
    "cityDepCodeAg": "string", //Ciudad de salida
    "airportArriCodeAg": "string", //Aeropuerto de llegada
    "countryArriCodeAg": "string", //Pais de llegada
    "cityArriCodeAg": "string", //Ciudad de llegada
    "namePass1Ag": "string", //Nombre pasajero 1
    "lastPass1Ag": "string", //Apellido pasajero 1
    "documentPass1Ag": "string", //Documento pasajero 1
    "namePass2Ag": "string", //Nombre pasajero 2
    "lastPass2Ag": "string",// Apellido pasajero 2
    "documentPass2Ag": "string", //Documento pasajero 2
    "ivatasa": "string", //IVA
    "codetasa": "string", //Codigo tasa
    "valetasa": "string" //Vale tasa
  },
//Si el comercio es un restaurante se incluye el siguiente campo, de lo contrario, se puede omitir
  "restaurantData": {
    "idTaxRest": "string" //Id cargo,
    "nameTaxRest": "string", //Nombre cargo
    "valueTaxRest": "string" //Valor cargo
  }
}
```

manual_de_integracion_pago_movil.pdf 27 15/11/22 2:26 p.m.

Anexo 2

```
{
  "responseCode": "01", //Codigo de respuesta
  "responseDescription": "ABCD", //Descripcion de la respuesta
  "responseEntity": {
    "commerceName": "ABCD", //Nombre del comercio
    "observations": "ABCD", //Observaciones del pago
    "currencyCode": "ABCD",  //Codigo de la moneda
    "purchaseCode": "ABCD", //Codigo de la compra
    "totalAmount": "000", //Total de la compra
    "terminalCode": "ABCD", //Codigo de la terminal
    "iva": " ABCD ", //IVA
    "ref1": " ABCD ", //Referencias 1 a 10
    "ref2": " ABCD ",
    "ref3": " ABCD ",
    "ref4": " ABCD ",
    "ref5": " ABCD ",
    "ref6": " ABCD ",
    "ref7": " ABCD ",
    "ref8": " ABCD ",
    "ref9": " ABCD ",
    "ref10": " ABCD",
    "iacId": " ABCD", //Id IAC
    "iacName": "ABCD", //Nombre IAC
    "iacVal": "ABCD", //Valor IAC
    "posPago": " ABCD ", //Pospago
    "convenio": " ABCD ", //Convenio
    "entidad": " ABCD ", //Entidad
    "urlToken": "ABCD", //Token del pago
    "paymentState": "P" //Estado del pago
  }
}
```

manual_de_integracion_pago_movil.pdf 28 15/11/22 2:26 p.m.

Anexo 3

```
{
  "responseCode": "01", //Codigo de respuesta
  "responseDescription": "ABCD", //Descripcion de la respuesta
  "responseEntity": {
    "commerceName": "ABCD", //Nombre del comercio
    "observations": "ABCD", //Observaciones del pago
    "currencyCode": "ABCD",  //Codigo de la moneda
    "purchaseCode": "ABCD", //Codigo de la compra
    "totalAmount": "000", //Total de la compra
    "terminalCode": "ABCD", //Codigo de la terminal
    "iva": " ABCD ", //IVA
    "ref1": " ABCD ", //Referencias 1 a 10
    "ref2": " ABCD ",
    "ref3": " ABCD ",
    "ref4": " ABCD ",
    "ref5": " ABCD ",
    "ref6": " ABCD ",
    "ref7": " ABCD ",
    "ref8": " ABCD ",
    "ref9": " ABCD ",
    "ref10": " ABCD",
    "iacId": " ABCD", //Id IAC
    "iacName": "ABCD", //Nombre IAC
    "iacVal": "ABCD", //Valor IAC
    "posPago": " ABCD ", //Pospago
    "convenio": " ABCD ", //Convenio
    "entidad": " ABCD ", //Entidad
    "urlToken": "ABCD", //Token del pago
    "paymentState": "P" //Estado del pago,
    "lastCard": "0003", //últimos 4 digitos de la tarjeta con que se realizó el pago
    "cardFranchise": "VISA", // Franquicia de la tarjeta
    "dues": 1 , // No de cuotas
  }
}
```

manual_de_integracion_pago_movil.pdf 29 15/11/22 2:26 p.m.

Anexo 4

```
{
  "totalAmount": "10000", // Valor final de la transacción
  "iva": "0", // iva
  "iacVal": "0", // iac
}
```