Historial de revisiones

Fecha	Versión	Descripción	Autor
06/09/2018	1.0	Generación del documento	Miguel Tomairo
27/11/2018	1.1	Actualización del documento	Miguel Tomairo
11/01/2019	1.2	Campos adicionales	Miguel Tomairo
15/01/2019	1.2.1	Actualización de campos en tablas	Miguel Tomairo

Objetivo del documento

El objetivo de este documento es describir el proceso de integración de un comercio a la pasarela de pagos Payme Mobile de una manera dinámica y eficiente.

Este documento recoge los pasos requeridos por el comercio para realizar la integración a la pasarela de pago para poder procesar pagos con tarjetas de débito o crédito.
El lenguaje de programación de esta librería es Swift, el cual es independiente al lenguaje de la APP con el que se integre.

Contenidos

1.0 Creación de Proyecto iOS

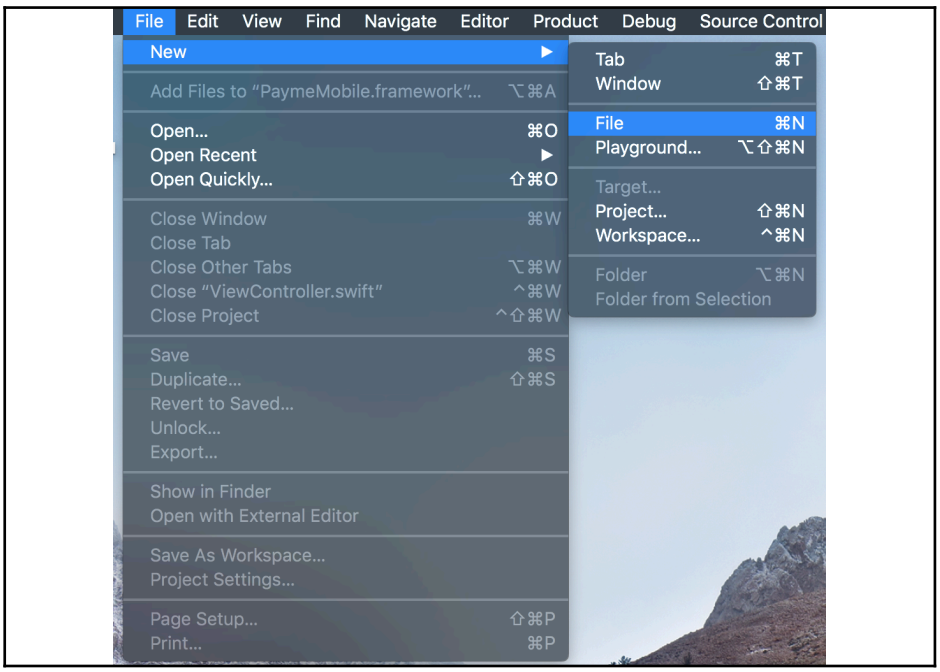
1.1 Software

IDE : Xcode 10.1

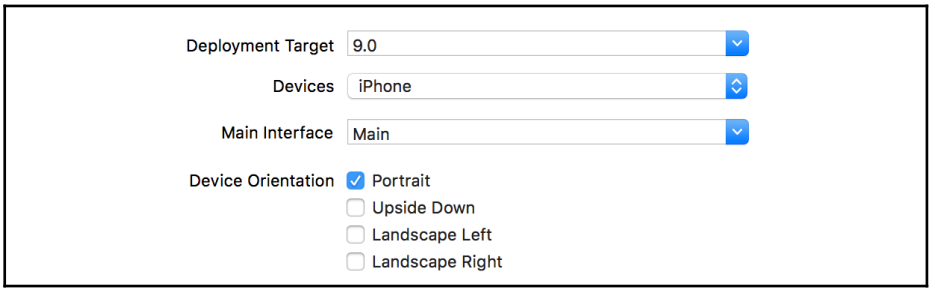
Plugin : Libreria Payme Mobile

1.2 Creación del proyecto que utilizara el API

Para crear un proyecto con Xcode, dirigirse a File > New > Project.



La versión mínima del SDK debe ser iOS 9, Luego seguir el proceso normal de creación para una aplicación nueva.



1.3 Permisos

El SDK permite escanear las tarjetas de crédito (con relieve). Para utilizarla, es necesario agregar el permiso correspondiente en el archivo plist del proyecto.

<key>NSCameraUsageDescription</key>
<string>Se necesita usar la cámara para escanear las tarjetas.</string>

Information Property List	Dictionary	(15 items)
Localization native development re...	String	$(DEVELOPMENT_LANGUAGE)
Executable file	String	$(EXECUTABLE_NAME)
Bundle identifier	String	$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle version	String	1
Application requires iPhone enviro...	Boolean	YES
Launch screen interface file base...	String	LaunchScreen
Main storyboard file base name	String	Main
Required device capabilities	Array	(1 item)
Supported interface orientations	Array	(3 items)
Supported interface orientations (i...	Array	(4 items)
Privacy - Camera Usage Description	String	Se necesita usar la camara para escanear las tarjetas.

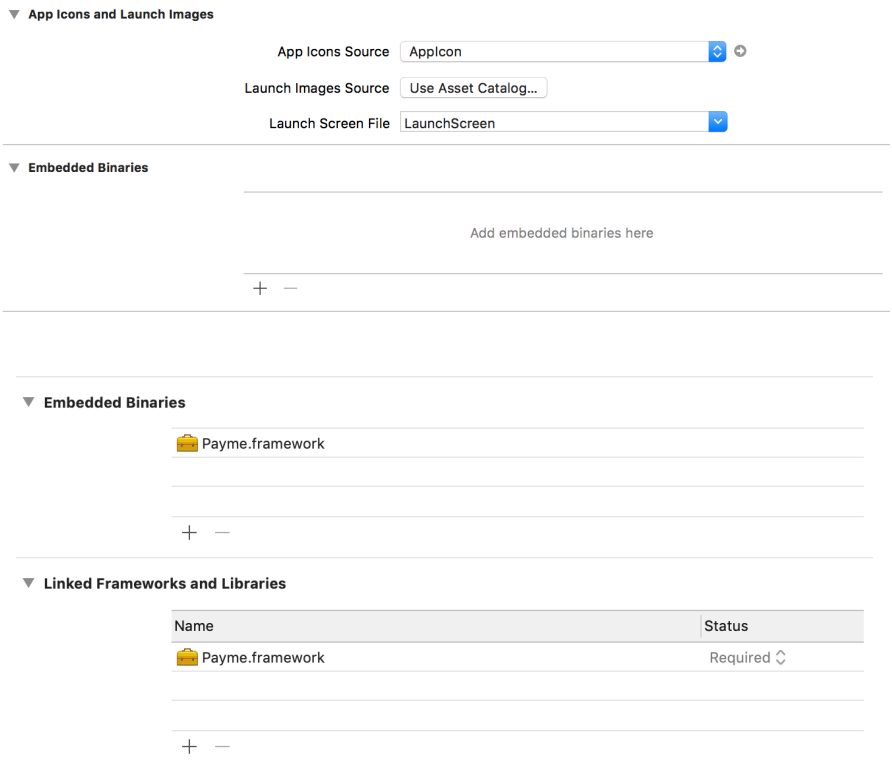
2.0 Integración Manual

2.1 Carpeta compartida

El SDK se encuentra publicado en una carpeta compartida (drive). Podrá acceder al SDK mediante el siguiente enlace.

2.2 Instalación de Framework

Descargamos el archivo Payme.framework que esta subido en la carpeta **Librería**. Luego de ello nos dirigimos al proyecto iOS qué deseamos agregar la librería. Seleccionamos el archivo y lo arrastramos y soltamos directamente en la sección de Embedded Binaries.



3.0 Modelo de datos de la librería

3.1 Módulo de pagos

A modo de prueba se crearon los propiedades para poder enviar dinámicamente los campos necesarios para consumir los servicios web propios de la librería.

- Locale
- Merchant
- Brands
- Operation
- number
- amount
- Currency
- productDes
- WalletUser.C
- PlanQuota
- Signature Key

3.2 Atributos de entrada y Salida

3.2.1 Entrada

MerchantOperationData (**Objeto principal)**

Propiedad	Tipo	Descripción
features	MerchantData	Características adicionales durante la realización de la transacción.
data	SettingsData	Información de la compra con la cual se realizará la transacción. **Obligatorio**
settings	FeaturesData	Contiene los valores de ajustes del comportamiento del formulario de pago. **Obligatorio**

MerchantData

Objeto que contiene los valores de la compra con el que se realizará la transacción.

Propiedad	Tipo	Descripción
operation	OperationData	Contiene los datos de la operación con la cual se procesará la transacción. **Obligatorio**
shipping	PersonaData	Información de datos de envió de la orden de compra.
billing	PersonaData	Información de datos de facturación de la orden de compra.
customer	PersonaData	Información del comprador que realiza la operación **Obligatorio**
signatureKey	String	Este valor es necesario para generar el **signature**, firma necesaria para validar la integridad de la trama al momento de consumir los servicios web. Este valor es un HASH Generado, dentro de la librería, con el algoritmo SHA-512 utilizando la cadena producto de concatenar los campos ***{número de operación} + {monto} + {código de moneda} + {llave para firmar}***

SettingsData

Ajustes sobre el comportamiento del formulario de pago.

Propiedad	Tipo	Descripción
locale	String	Localización del idioma que utilizará el formulario para mostrarse. **Valor Descripción** es_PE Español Perú en_US Inglés EEUU
identifier	String	Identificador del comercio.

brands	Array<String>	Contiene los valores de las marcas que serán habilitadas para el formulario de pago. **Valor** **Descripción** VISA Visa MSCD MasterCard AMEX American Express DINC Diners Club Nota: Las marcas se visualizarán en el formulario de pago de acuerdo al orden enviado en el Array. **Valor por defecto**: ['VISA']
responseType	ResponseType	Determina el tipo de respuesta que entregara el SDK. **En Swift:** - ResponseType.detail - ResponseType.extended **En Objective-c** - ResponseTypeDetail - ResponseTypeExtended

FeaturesData

Características adicionales durante la realización de la transacción.

Propiedad	Tipo	Descripción
wallet	FeatureWalletData	Contiene los valores de ajustes del comportamiento para activar Wallet
reserved	Array<FeaturedReservedData>	Contiene los valores de ajustes del comportamiento para los datos reservados que venga desde el comercio
planQuota	Boolean	Campo para poder activar planes y cuota . Nota: 1) N = no está activado. 2) Y = está activado.

FeatureWalletData

Objeto contenedor de la configuración de Wallet

Propiedad	Tipo	Descripción
userCommerce	String	dato poder activar wallet, donde el comercio definirá el valor.

FeaturedReservedData

Nombre del campo reservado y valor asignado para el comercio.

Propiedad	Tipo	Descripción
name	String	Nombre del campo reservado. **Obligatorio**
value	String	Valor del campo reservado. **Obligatorio**

OperationData

Contiene los datos de la operación con la cual se procesará la transacción.

Propiedad	Tipo	Descripción
operationNumber	String	Número de operación único en la pasarela, el cual será utilizado para realizar la transacción.
amount	String	Monto a cobrar por la pasarela de pago, se debe considerar punto como separador de decimales y máximo de dos dígitos decimales (#0.00) **Obligatorio**
productDescription	String	Descripción del producto por el cual se realiza la transacción. **Obligatorio**
currency	CurrencyData	Información de la moneda con la que se realizará la transacción. **Obligatorio**

CurrencyData

Información de la moneda con la cual se realizará a transacción.

Propiedad	Tipo	Descripción
code	String	Código de moneda ISO-4217
symbol	String	Símbolo de moneda con el cual se visualizará el monto en el formulario, no tiene ningún efecto en la transacción. Se recomienda el uso de los símbolos determinados en el ISO-4217

PersonaData

Campos que almacenan datos personales para algún propósito dentro de la pasarela.

Propiedad	Tipo	Descripción
firstName	String	Nombres
lastName	String	Apellidos
email	String	Correo electrónico
address	String	Dirección
zip	String	Código postal
city	String	Ciudad/Provincia
state	String	Estado/Departamento
country	String	País
phone	String	Teléfono

3.2.2 Salida

PaymentData

Respuesta del servidor luego de la ejecución del pago desde el comercio. Esta respuesta es referencial para el comprador, el comercio recibirá la notificación desde los servidores de Alignet en línea.

Propiedad	Tipo	Descripción
success	Boolean	Tiene valor **true** si se realizó el proceso correctamente, **false** si ocurrió algún error durante el procesamiento del pago.
messageCode	String	Código del mensaje de respuesta
message	String	Mensaje de respuesta

payment	PaymentResponse Data	Resultado del proceso de pago siempre y cuando no ocurra ningún error en el procesamiento de transacción (Propiedad success: true).
features	Features ResponseData	Características adicionales durante la realización de la transacción.

PaymentResponseData

Resultado del procesamiento del pago de acuerdo a la marca que corresponda.

Propiedad	Tipo	Descripción
accepted	Boolean	true, cuando el pago ha sido procesado correctamente, false si se ha rechazado por alguna razón al momento de realizar la transacción. En caso sea false se puede visualizar la razón en los campos resultCode y resultMessage.
resultCode	String	Código de resultado de procesamiento de pago
resultMessage	String	Mensaje de resultado de procesamiento de pago
authorizationResult	String	Código de autorización de la transacción
referenceCode	String	Código de referencia
brand	String	Marca de las tarjetas
bin	String	Bin de la tarjeta(primero 6 dígitos)
lastPan	String	Ultimo 4 digitos de la tarjeta
transactionIdentifier	String	Identificador de transacción
errorCode	String	Este campo tiene el resultado de respuesta de la Librería
errorMessage	String	Este campo contiene el resultado de la respuesta de la libreria.
date	String	Fecha de procesamiento de la transacción.
hour	String	Hora de procesamiento de la transacción.
operationNumber	String	Numero de operation correspondiente a la transacción.

FeaturesResponseData

Resultado de las características del pago de acuerdo a la marca que corresponda.

Propiedad	Tipo	Descripción
planQuota	PlanQuotaData	Trama para identificar los planes y cuotas así como otros campos de interés.
Reserved	Array<FeaturedReservedData>	Contiene los valores de ajustes del comportamiento para los datos reservados que venga desde el comercio

PlanQuotaData

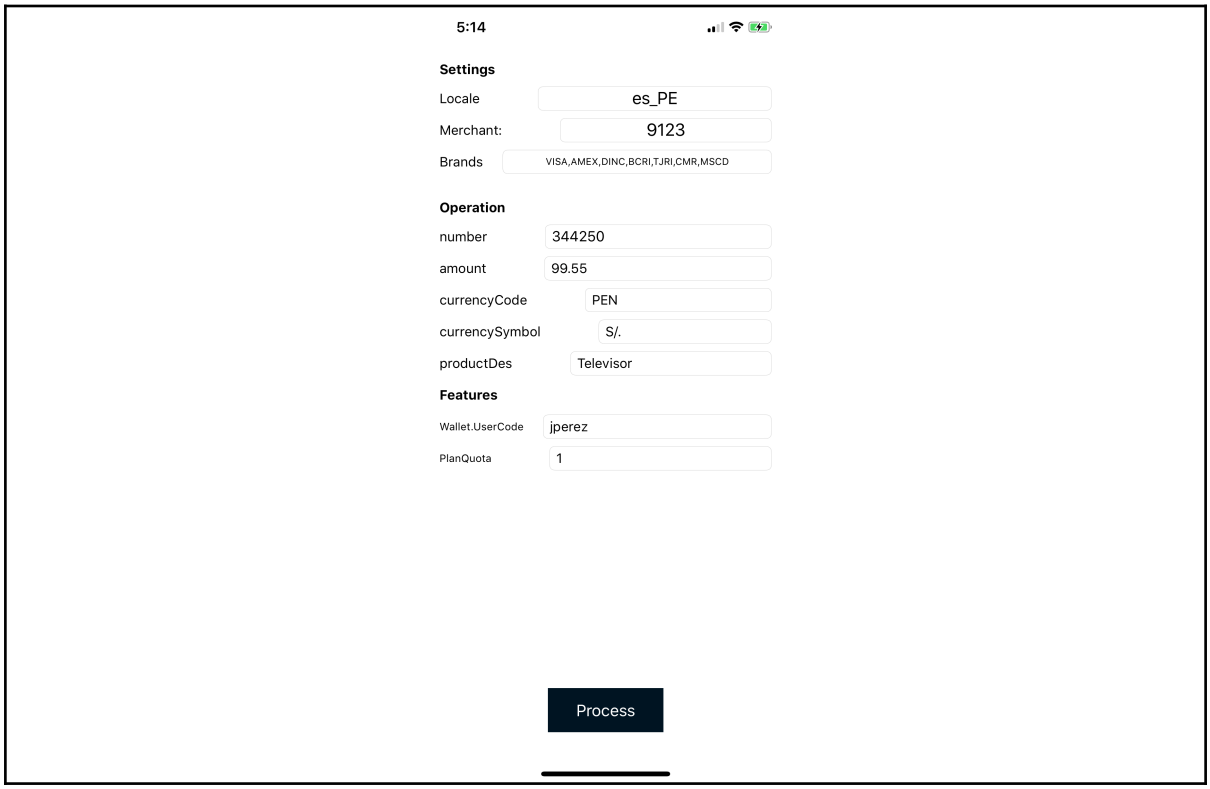
Resultado de planes y cuotas así como otros campos de interés.

Propiedad	Tipo	Descripción
plan	String	00=> Normal, 02=> Diferido
Quota	String	cantidad de cuotas
QuotaProcessed	String	Proceso de cuota
amount	String	Valor de cuota
dueDate	String	Fecha de primer vencimiento en formato yyyyMMdd
currency	String	Moneda
Interest	String	Intereses

4.0 Proyecto Ejemplo

Para realizar el consumo del API, tomaremos como base el proyecto de ejemplo, disponibles en tres versiones. Lo pueden encontrar en el drive del SDK en la carpeta Ejemplo/ iOS.

A continuación, se muestra la pantalla del carrito que se construyo con el fin de ingresar datos variables para consumir la librería. En el ViewController de la pantalla del carrito podrá identificar los métodos delegados que requiere implementar la aplicación para invocar la librería.



4.1 Importación de Libreria

Se ha construido una serie de proyectos de ejemplo que permitirán guiarlo según el lenguaje que desea utilizar para realizar la integración en su aplicación móvil.
Lo primero que haremos será hacer el import de la librería y seguidamente implementar las clases y métodos delegados que esta solicita. Se proveen los siguientes tipos para realizar la importación de la librería en su proyecto iOS.

4.1.1 Con Swift

Nos ubicamos en la parte superior del viewcontroller e importamos la librería para que sus métodos y entidades puedan ser reconocidas.

```swift
import UIKit
import Payme

class ViewController: UIViewController {
```

4.1.2 Con Objective-c

Nos ubicamos en el header(.h) o archivo de implementación(.m) y agregamos el import de la librería.

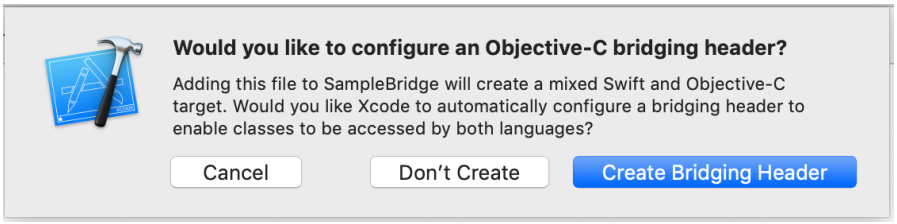
```objc
#import <UIKit/UIKit.h>
#import <Payme/Payme.h>

@interface ViewController : UIViewController<PaymeMobileDelegate>

@end
```

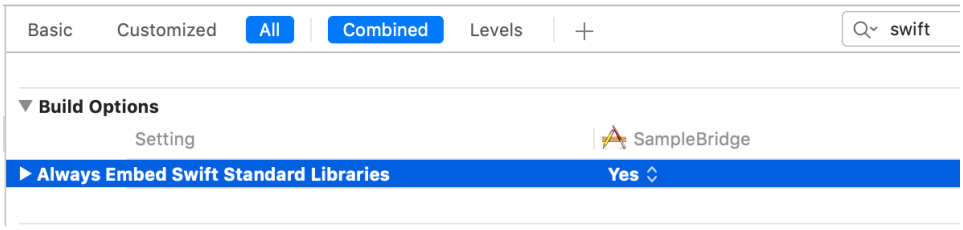
4.1.3 Con Bridge

Si deseamos consumir directamente un archivo Swift en nuestro proyecto de Objective-c. Lo que tenemos que hacer es agregar un archivo Swift en el proyecto. Inmediatamente le aparecerá la siguiente ventana. Hacemos click en **Créate Bridging Header.** Finalmente, realizamos la implementación como si fuera un proyecto Swift.



Nota:

Para proyectos hechos con Objective-c es necesario realizar la siguiente configuración en **Build Settings**



4.2 Delegados del SDK

Luego de importar correctamente el SDK procedemos a configurar los dato de entrada que son necesarios para invocar la librería.

4.2.1 Con Swift

Declaramos un extension y agregamos el delegado del SDK llamado PaymeMobileDelegate. El delegado cuenta con una serie de valores y métodos que detallamos a continuación:

- **setEnviroment**: Este variable nos permitirá setear el ambiente (production, development o sandbox)
- **setParamsMerchant:** Esta función nos permite instancias las clases propias del SDK, necesarias para iniciar el flujo de pago.
- **getResponsePay:** Este método retorna las respuesta luego de haber efectuado una transacción, sea exitosa o no.

El carrito de ejemplo provee parámetros configurables por medio de cajas de texto (TextField). También se provee un método random para generar el numero de operación, pues este valor debe ser único por transacción.
El comercio es responsable de proveer los valores necesarios y correctos para poder realizar un pago. El resultado puede ser aceptado o denegado y esto dependerá de las tarjetas validas(bloqueadas, sin saldo) o que los ambientes se encuentren desplegados.

```
// MARK: - PaymeMobileDelegate methods
extension ViewController: PaymeMobileDelegate {

    func dismissed() {
        print("Hacer algo ....")
    }

    var setEnviroment: Enviroment {
        return .development
    }

    func setParamsMerchant() -> ModelMerchant {

        let modelMerchantSettings = ModelMerchantSettings(
            locale: self.settingLocale.text!,
            identifier: self.settingIdentifier.text!,
            brands: self.settingBrands.text!.components(separatedBy: ","),
            signatureKey: self.signatureKey(merchantId: self.settingIdentifier.text!),
            responseType: ResponseType.extended
        )

        let modelMerchantFeatures = ModelMerchantFeatures(
            wallet: ModelMerchantFeaturesWallet(
                userCommerce: self.sendWallet(merchant: self.settingIdentifier.text!)
            ),
            reserved: [
                ModelMerchantFeaturesReserved(name: "reserved1", value: "1"),
                ModelMerchantFeaturesReserved(name: "reserved2", value: "2"),
                ModelMerchantFeaturesReserved(name: "reserved3", value: "3")
            ],
            planQuota: (self.featuresPlanQuota.text != "0" ? true : false)
        )

        let modelMerchantDataOperation = ModelMerchantDataOperation(
            operationNumber: self.operationNumber.text!,
            amount: self.operationAmount.text!,
            currency: ModelMerchantDataOperationCurrency(
                code: self.operationCurrencyCode.text!,
                symbol: self.operationCurrencySymbol.text!
            ),
            productDescription: self.operationProduct.text!
        )
```

```
        let modelMerchantDataPerson = ModelMerchantDataPerson(
            firstName: "Javier",
            lastName: "Perez",
            email: "jperez@alignet.com",
            address: "Av casimiro Ulloa 333",
            zip: "000",
            city: "Lima",
            state: "Lima",
            country: "Peru",
            phone: "998888444"
        )

        let modelMerchantData = ModelMerchantData(
            operation: modelMerchantDataOperation,
            shipping: modelMerchantDataPerson,
            billing: modelMerchantDataPerson,
            customer: modelMerchantDataPerson,
            signature: ""
        )

        let merchant = ModelMerchant(
            data: modelMerchantData,
            settings: modelMerchantSettings,
            features: modelMerchantFeatures
        )

        return merchant
    }

    func getResponsePay(response: ModelPayment?) {
        if let response = response {
            print("brand: ", response.payment?.brand ?? "")
            print("lastPan: ", response.payment?.lastPan ?? "")
            print("bin: ", response.payment?.bin ?? "")

        } else {
            print("response: ", "another error")
        }
    }
```

4.2.2 Con objective-c

De manera análoga se realiza la declaración del delegado y los métodos necesario para invocar la librería.

Para setear el tipo de ambiente será necesario declarar la variable y asignar el tipo de Environment correspondiente (production, development, sandbox). Esta asignación se puede realizar directamente en viewDidLoad o por medio de una función que declara valores iniciales para el viewcontroller.

```
#pragma mark - PaymeMobileDelegate method

@synthesize setEnviroment;

- (void)dismissed {
    NSLog(@"Hacer algo ...");
}

- (void)getResponsePayWithResponse:(ModelPayment * _Nullable)response {

    if (response != NULL) {
        NSLog(@"Response: %@",response);
//        NSLog(@"success: %s",response.success ? "true" : "false");
//        NSLog(@"messageCode: %@ \n",response.messageCode);
//        NSLog(@"message: %@ \n",response.message);
//        NSLog(@"accepted: %s \n",response.payment.accepted ? "true" : "false");
//        NSLog(@"resultCode: %@ \n",response.payment.resultCode);
//        NSLog(@"resultMessage: %@",response.payment.resultMessage);
    }else {
        NSLog(@"Response: another error");
    }
}

- (ModelMerchant * _Nonnull)setParamsMerchant {

    ModelMerchantDataOperationCurrency *currency = [[ModelMerchantDataOperationCurrency alloc] initWithCode: self.operationCurrencyCode.text
                                                                                                      symbol: self.operationCurrencySymbol.text];

    ModelMerchantDataOperation *modelMerchantDataOperation = [[ModelMerchantDataOperation alloc] initWithOperationNumber:self.operationNumber.text
                                                                                                                  amount:self.operationAmount.text
                                                                                                                currency:currency
                                                                                                     productDescription:self.operationProduct.text];

    ModelMerchantDataPerson *modelMerchantDataPerson = [[ModelMerchantDataPerson alloc] initWithFirstName:@"Miguel Angel"
                                                                                                 lastName:@"Tomairo"
                                                                                                    email:@"miguel.tomairo@alignet.com"
                                                                                                  address:@"casimiro ulloa 333"
                                                                                                      zip:@"000"
                                                                                                     city:@"Lima"
                                                                                                    state:@"Lima"
                                                                                                  country:@"Peru"
                                                                                                    phone:@"990380090"];
```

```
ModelMerchantData *modelMerchantData = [[ModelMerchantData alloc] initWithOperation: modelMerchantDataOperation
                                                                            shipping: modelMerchantDataPerson
                                                                             billing: modelMerchantDataPerson
                                                                            customer: modelMerchantDataPerson
                                                                           signature: @""];

ModelMerchantSettings *modelMerchantSettings = [[ModelMerchantSettings alloc] initWithLocale:self.settingLocale.text
                                                                                  identifier:self.settingIdentifier.text
                                                                                      brands:[self.settingBrands.text componentsSeparatedByString:@","]
                                                                               signatureKey:@"SYbpPEwdKxErVzt@66458456"
                                                                               responseType:ResponseTypeExtended];

ModelMerchantFeaturesWallet *modelMerchantFeaturesWallet = [[ModelMerchantFeaturesWallet alloc] initWithUserCommerce:self.featuresWalletUserCode.text];

ModelMerchantFeaturesReserved *reserved1 = [[ModelMerchantFeaturesReserved alloc] initWithName:@"reserved1"
                                                                                        value:@"1"];
ModelMerchantFeaturesReserved *reserved2 = [[ModelMerchantFeaturesReserved alloc] initWithName:@"reserved2"
                                                                                        value:@"2"];
ModelMerchantFeaturesReserved *reserved3 = [[ModelMerchantFeaturesReserved alloc] initWithName:@"reserved3"
                                                                                        value:@"3"];

NSArray *reservedArray = @[reserved1, reserved2, reserved3];

ModelMerchantFeatures *modelMerchantFeatures = [[ModelMerchantFeatures alloc] initWithWallet:modelMerchantFeaturesWallet
                                                                                   reserved:reservedArray
                                                                                  planQuota:false];


ModelMerchant *merchant = [[ModelMerchant alloc] initWithData: modelMerchantData settings: modelMerchantSettings features: modelMerchantFeatures];


return merchant;
```

4.3 Invocar la Libreria

Si hemos declarado correctamente los valores anteriores procedemos a invocar la librería. Para este propósito invocamos el storyboard perteneciente al SDK, declaramos el protocolo delegado para poder pasar los valores de configuración que hemos ido agregando.

4.3.1 Con Swift

```swift
    // MARK: - IBAction
    @IBAction func btnStartProcessPayment(_ sender: UIButton) {
        let sb = UIStoryboard (name: "MainPaymeApi", bundle: Bundle(for: PayController.self))
        let vcStartPayment = sb.instantiateViewController(withIdentifier: "PayController") as! PayController
        vcStartPayment.paymeProtocol = self
        //        self.present(vcStartPayment, animated: true, completion: nil)
        self.navigationController?.pushViewController(vcStartPayment, animated: true)
    }
```

4.3.2 Con Objective-c

```objc
#pragma mark - Action

- (IBAction)btnStartProcessPayment:(UIButton *)sender {

    UIStoryboard *sb = [UIStoryboard storyboardWithName:@"MainPaymeApi" bundle: [NSBundle bundleForClass:[PayController class]]];
    PayController *vc = [sb instantiateViewControllerWithIdentifier:@"PayController"];
    vc.paymeProtocol = self;
    [self.navigationController pushViewController:vc animated:YES];
}
```

Nota:
Para realizar pruebas con tarjetas en el ambiente de desarrollo (Development) se tendrán que cumplir con las siguientes indicaciones:

- Pago aceptado: Ingresar una tarjeta valida (Visa, Mastercard, Amex, Diners) e ingresar un monto menor a S/ 3000
- Pago Denegado: Ingresar una tarjeta valida e ingresar un monto superior a S/ 3000