

**Framework Tarjeta No Presente IOS**

**Índice**

Glosario 3

Introducción 4

Requisitos de Implementación 5

Implementación del framework 5

1 - Configurar versión swift 2.3 5

2 - Importar frameworks 6

3 - Reconstruir el proyecto 7

4 - Agregar las librerías 8

5 - Importar el framework dentro de una clase swift 9

6 - Agregar permisos al info.plist 10

Uso del Framework 11

Lista de métodos de la clase BPCodes 11

Lista de métodos de la clase RequestBPC 13

Ejemplo de transacción completa en consola. 16

Código ejemplo swfit xcode 17

Salida del ejemplo 17

# Glosario

**Framework:** Conjunto de herramientas que permiten a un desarrollador hacer uso de métodos y funciones previamente creadas.

**BPTNPFramework:** Librería desarrollada por Blumon Pay la cual permitirá realizar las transacciones de pago.

**SWIFT:** Lenguaje para desarrollo de aplicaciones ios.

**Legacy:** Propiedad que soporta versiones anteriores de swift

# Introducción

Esta guía le mostrara como hacer uso del **BPTNPFramework** ios, de tal forma que usted podrá implementar en su aplicación transacciones ( compras electrónicas con tarjetas de crédito o debito )

Blumon Pay elaboro un framework bajo el lenguaje Swift, que permitirá implementar en cualquier aplicación la posibilidad de pagar con tarjeta de crédito o debito no presente, teniendo la seguridad de que la transacción esta siendo procesada con todos los iso y estándar de seguridad que son validados por prosa; El framework realiza una conexión a servidores seguros donde la petición se procesa por múltiples switch lo cual asegura la compra o pago.

La implementación del framework es extremadamente sencilla por lo cual es una de las mejores opciones para quienes desean integrar pagos seguros dentro de sus aplicaciones.

# Requisitos de Implementación

La librería tiene algunos requisitos importantes para su funcionamiento a continuación se describirá cuales.

* Usar Swift 2.3 ( la siguiente actualización será para 3.0)
* Importar CryptoSwift Framework
* Implementar SAppTransportSecurity ( véalo en el apartado implementación del framework )

# Implementación del framework

Siga los pasos indicados en el orden mencionado para hacer uso de la librería de pagos de Blumon Pay, se recomienda leer la guía completa antes de comenzar a desarrollar para evitar posibles fallos mencionados dentro de este documento.

La implementación esta presentada en pasos de acuerdo al orden que deberá seguir para su correcto uso.

**IMPORTANTE**

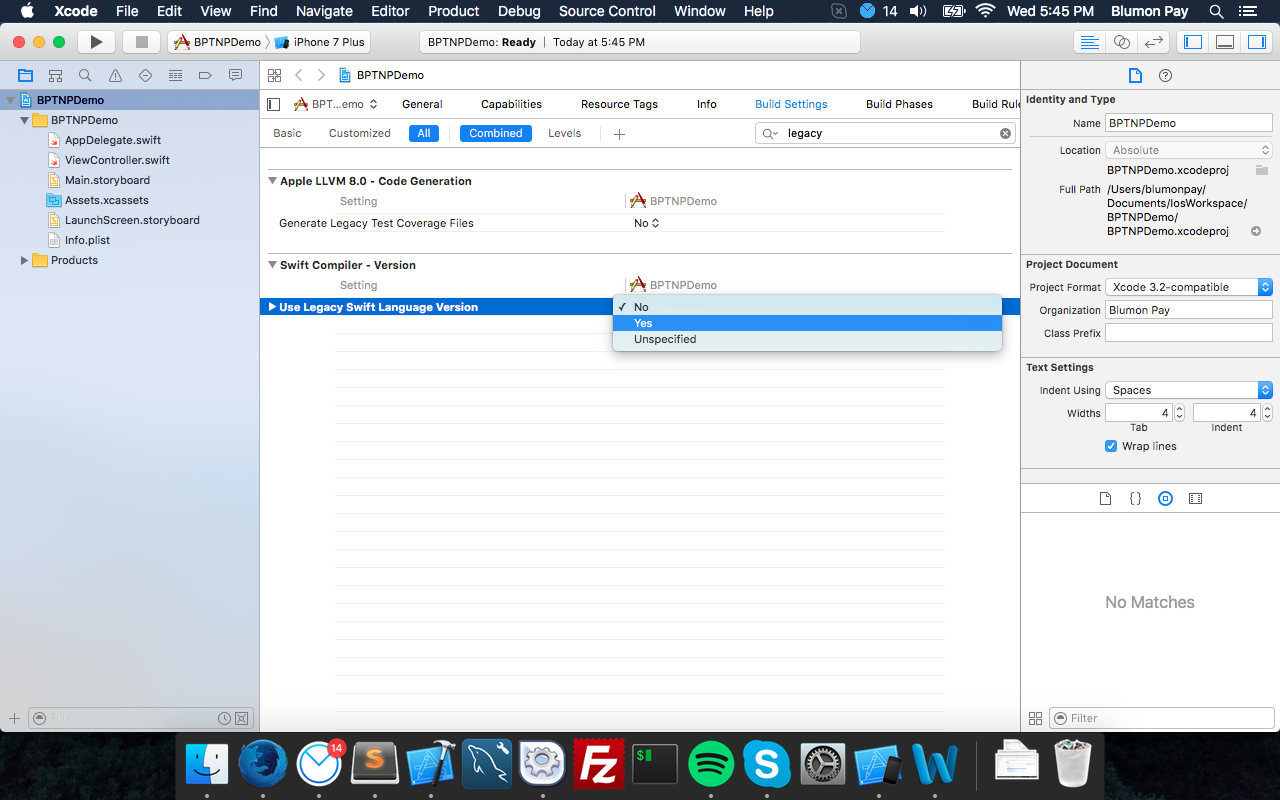
Usted podrá descargar la versión de pruebas únicamente, si desea el framework para producción, deberá solicitarlo junto con su usuario a Blumon Pay.

En tal caso deberá remplaza el framework de pruebas con el de producción, no deberá hacer ningún cambio extra.

## 1 - Configurar versión swift 2.3

Tu proyecto deberá trabajar sobre swift 2.3 ya que todos los framework requeridos fueron compilados sobre esta versión, si la aplicación donde se quiere implementar hace uso de swfit 3.0 deberá cambiar a modo legacy, si las funciones de su aplicación tiene conflicto deberá cambiarlas a swift 2.3

En XCODE, deberás seleccionar tu proyecto, en la pestaña build settings, usar el filtro de búsqueda y escribir legacy, cambiar por yes en la opción use legacy swift language versión. Ver figura 1.0



**Figura 1.0** Cambiar a modo swift legacy

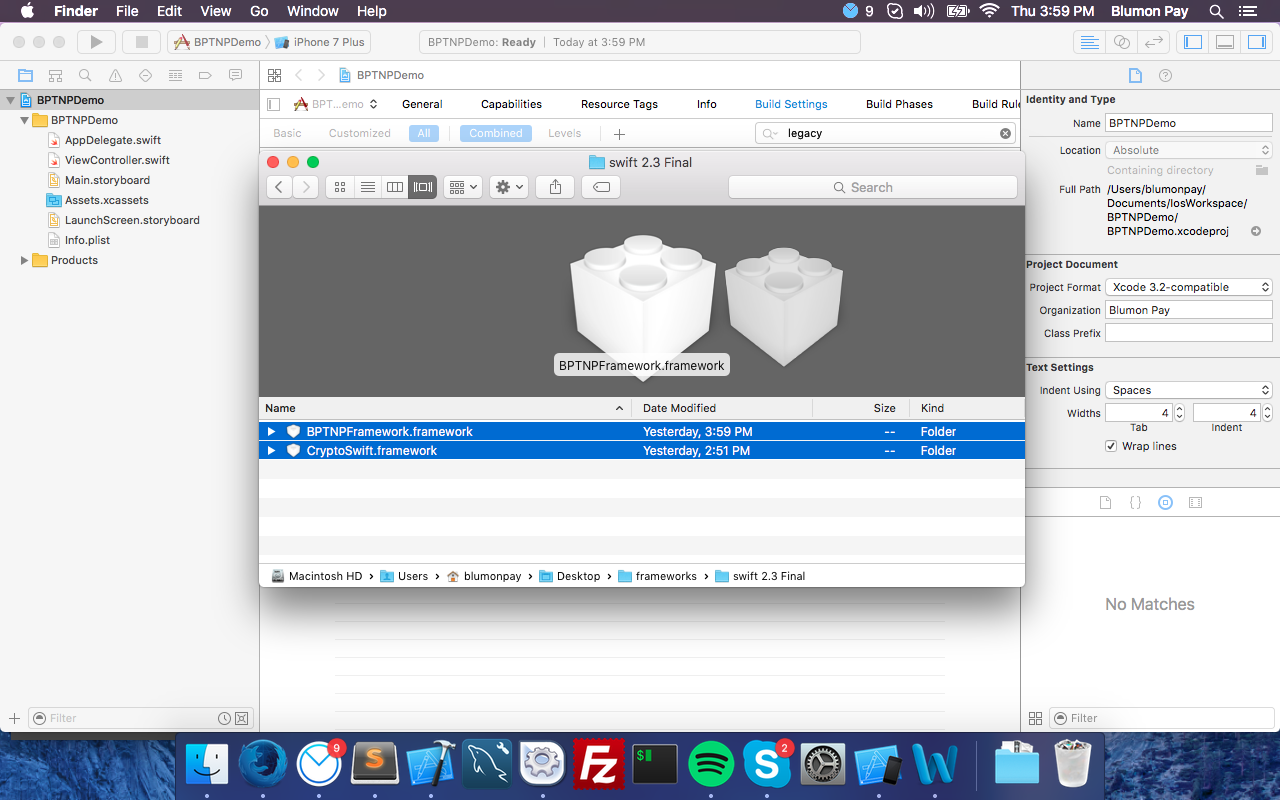
## 2 - Importar frameworks

Es necesario importar al proyecto no solo la librería de Blumon Pay, si no también unas librerías que nos proporcionan herramientas de seguridad que usa el modulo.

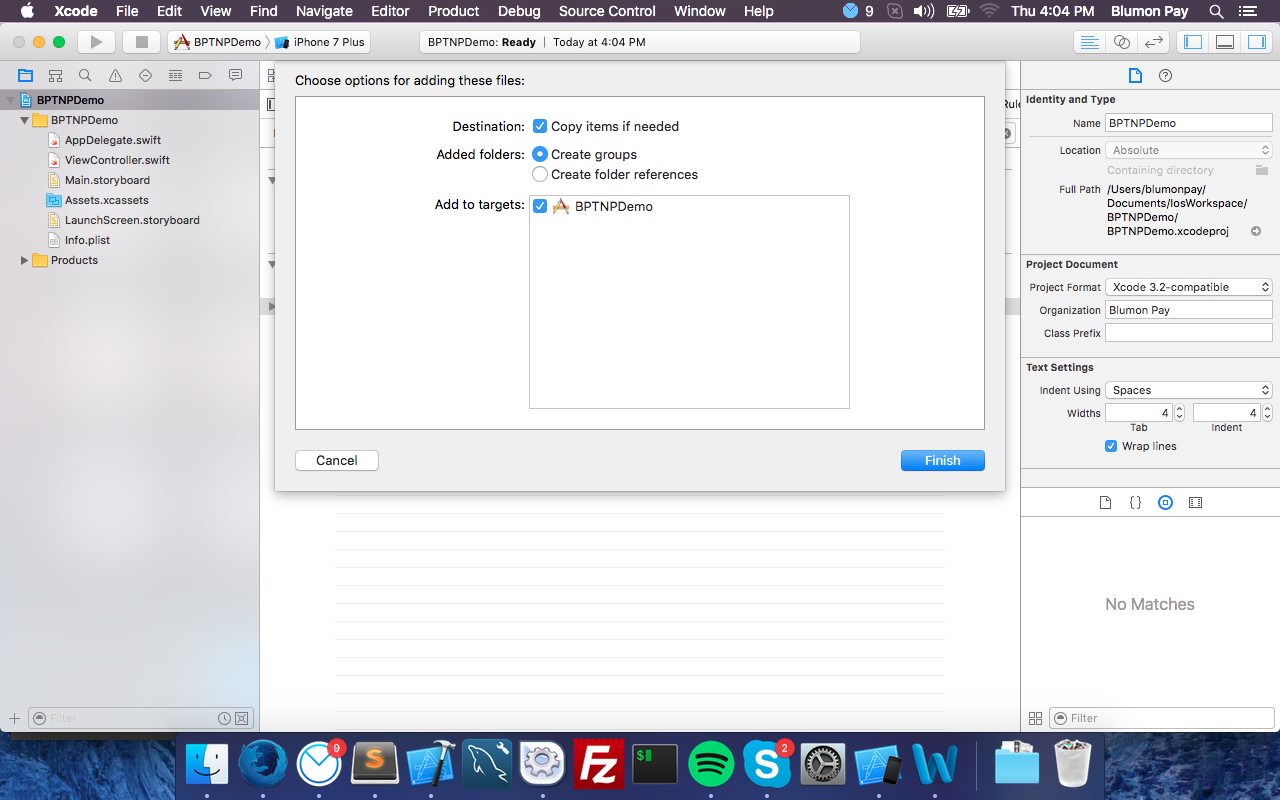
Abra finder, en la carpeta donde tenga los framework descargados, tendrá que poner la ventana de finder sobre XCODE, para poder arrastrar BPTNPFramework y CryptoSwift.

1. Seleccione ambos framework, y arrástrelos a cualquier parte dentro de su proyecto en XCODE, ver figura 1.1.
2. Aparecerá una ventana emergente de XCODE, donde deberá marcar las siguientes opciones, ver figura 1.2.

* **Destination:** Copy ítems if needed -> Seleccionar
* **Added folders:** Created groups -> Seleccionar
* **Add to targets:** Tu App -> Seleccionar



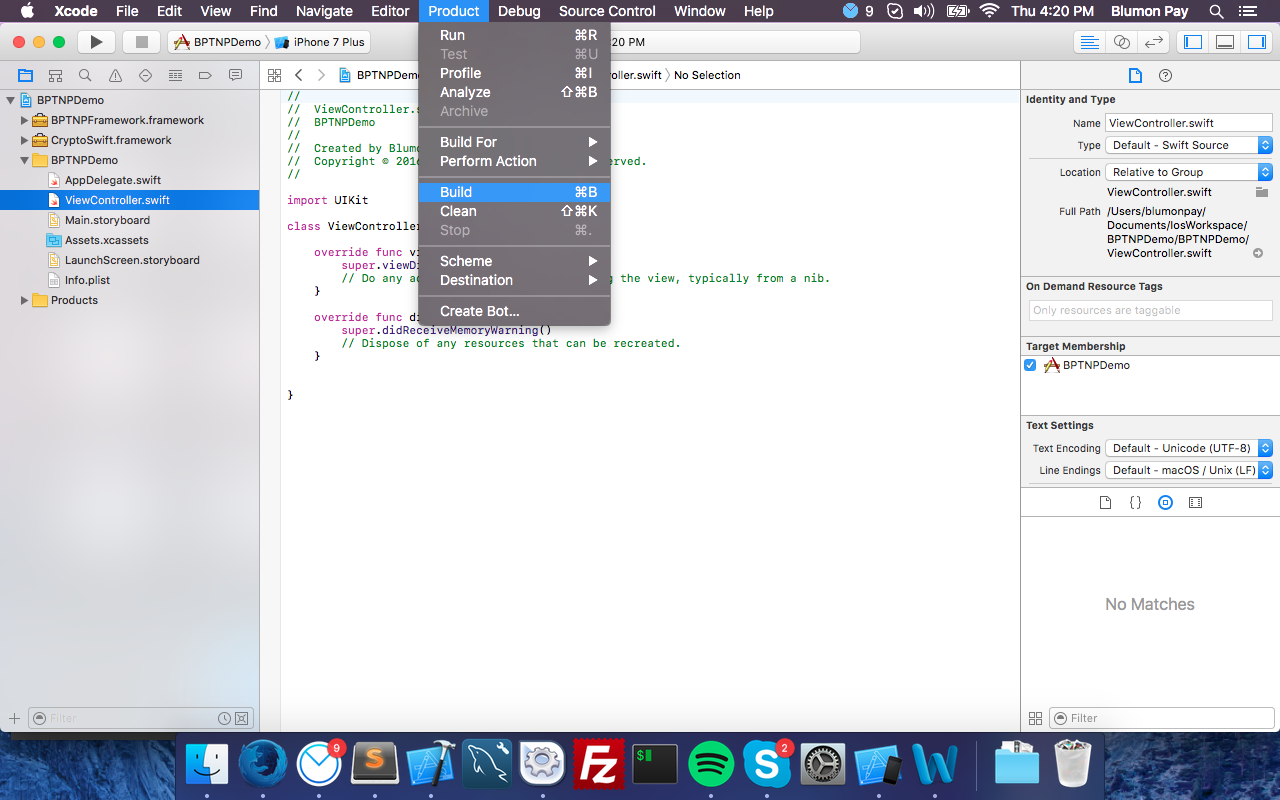
**Figura 1.1** Arrastrar frameworks al proyecto



**Figura 1.2** Copiando y creando referencias del framework

## 3 - Reconstruir el proyecto

Es necesario usar la opción de build para que el proyecto pueda detectar las librerías que acabamos de copiar, ver imagen 1.3

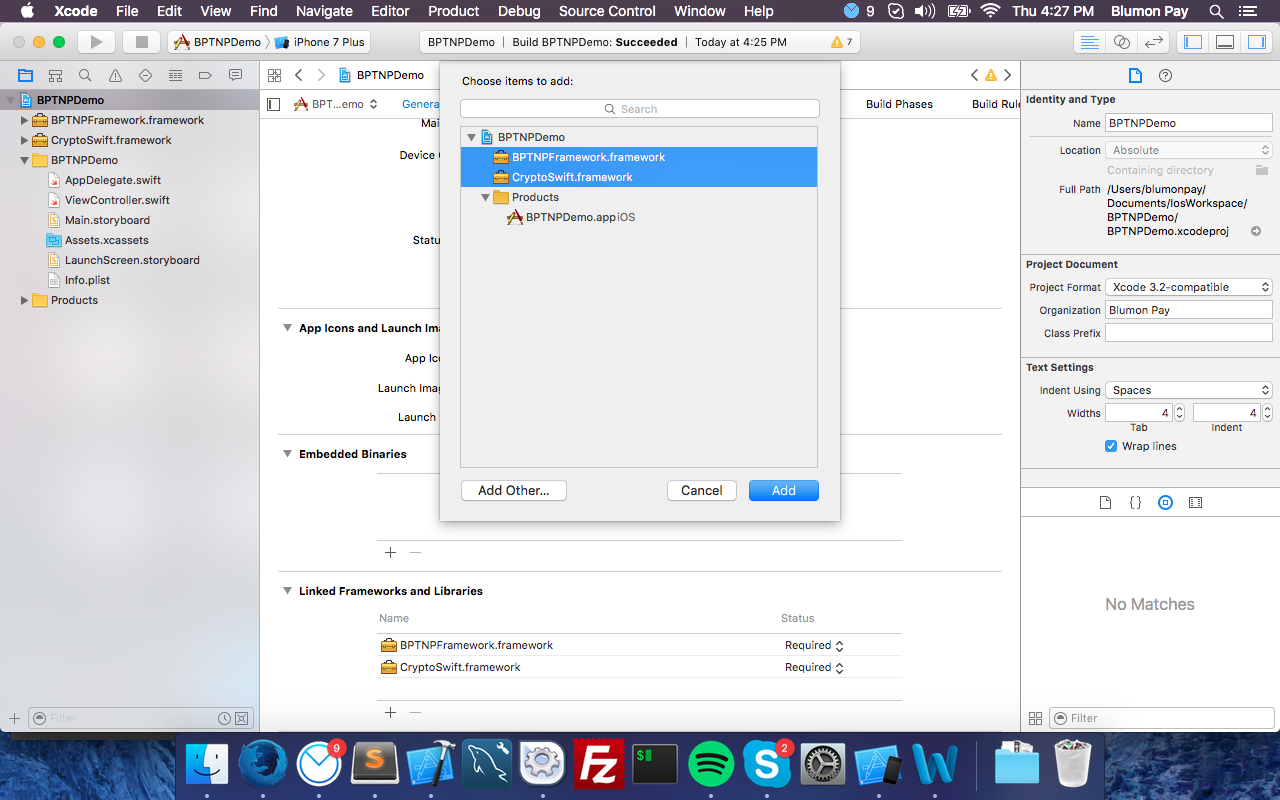


**Figura 1.3** Construyendo el proyecto

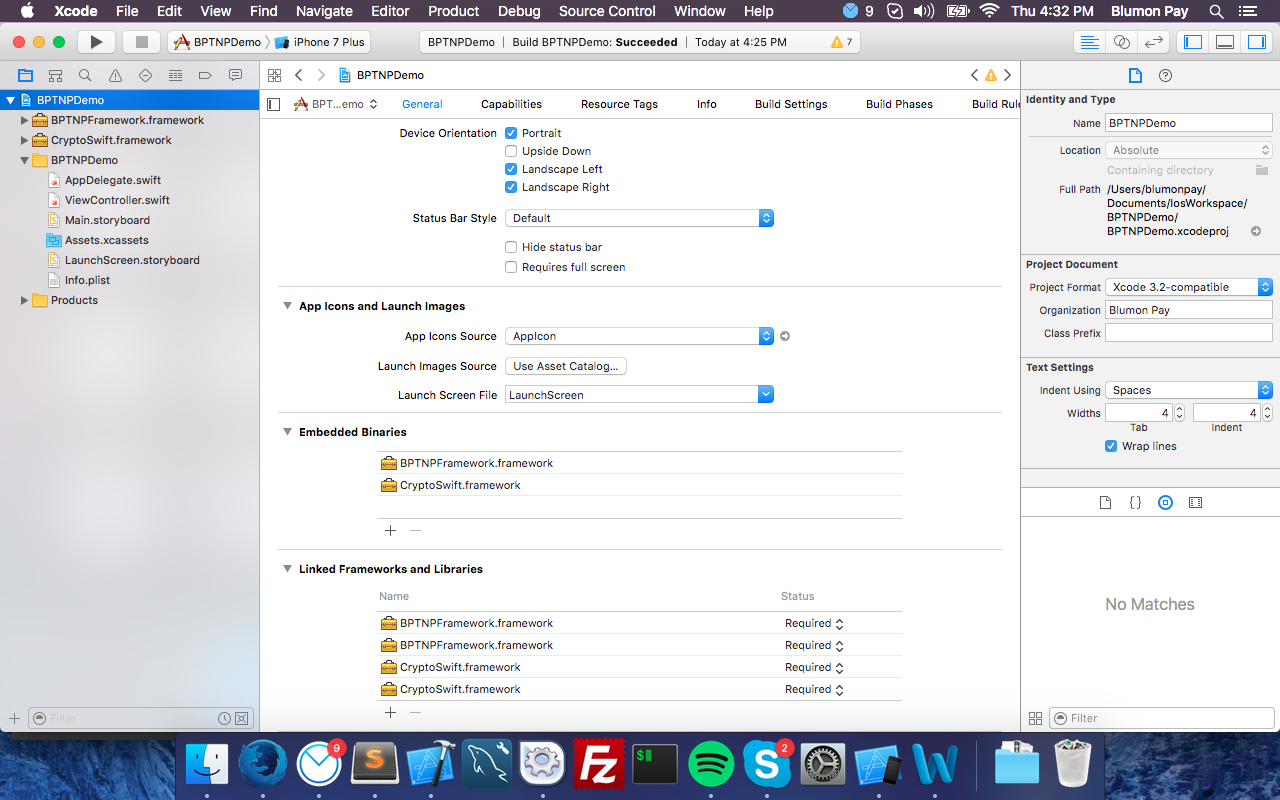
## 4 - Agregar las librerías

Es necesario agregar la referencia de la librería dentro del proyecto para que puedan ser usados los métodos de las mismas, no importa que ya se haya importado el framework es necesario agregar la librería.

Deberá seleccionar el proyecto, en la pestaña de general encontrara la opción embedded binaries, deberá presionar sobre el signo + para agregar las librerías, busque los 2 framework que acabamos de copiar y selecciónelos, ver figura 1.4, 1.5



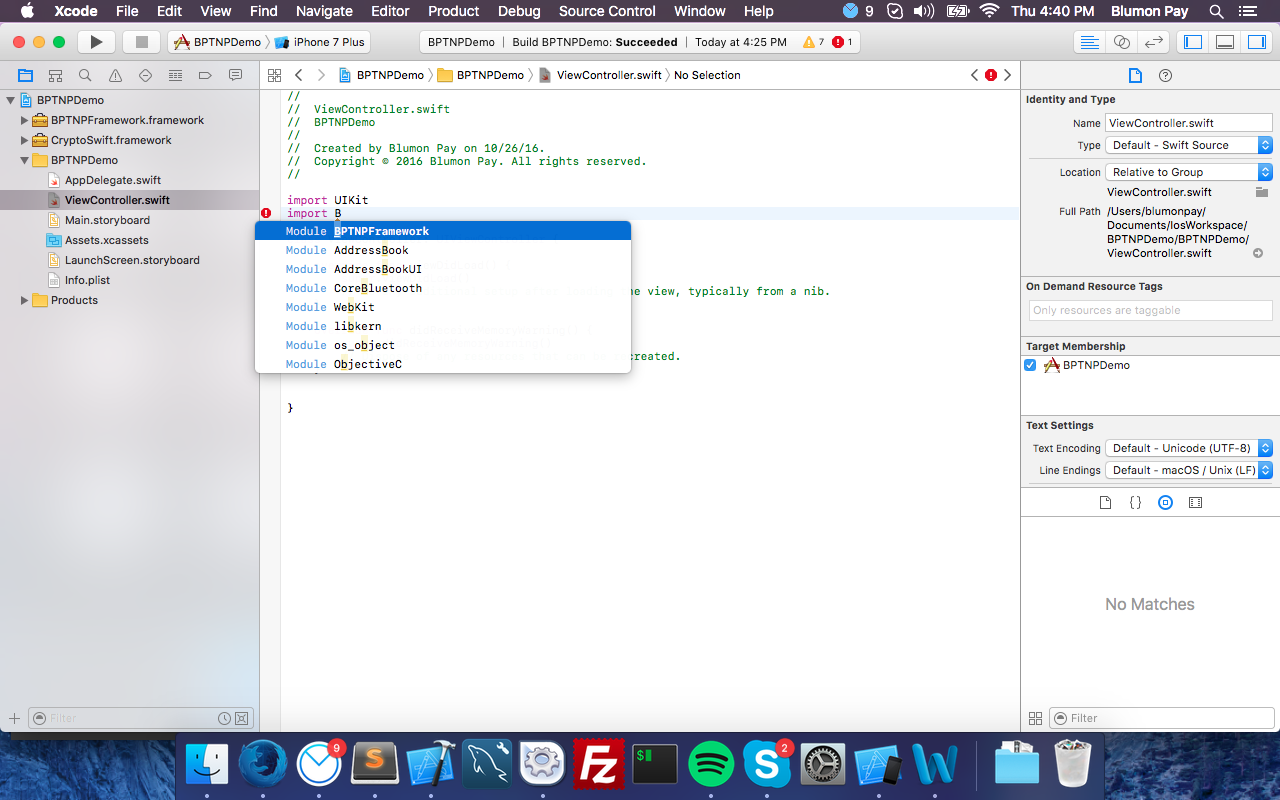
**Figura 1.4** Importando librerías al proyecto



**Figura 1.5** Ebedded binaries importados

## 5 - Importar el framework dentro de una clase swift

Es necesario hacer un import al framework para decirle que queremos usarlo en esa clase, ver figura 1.6

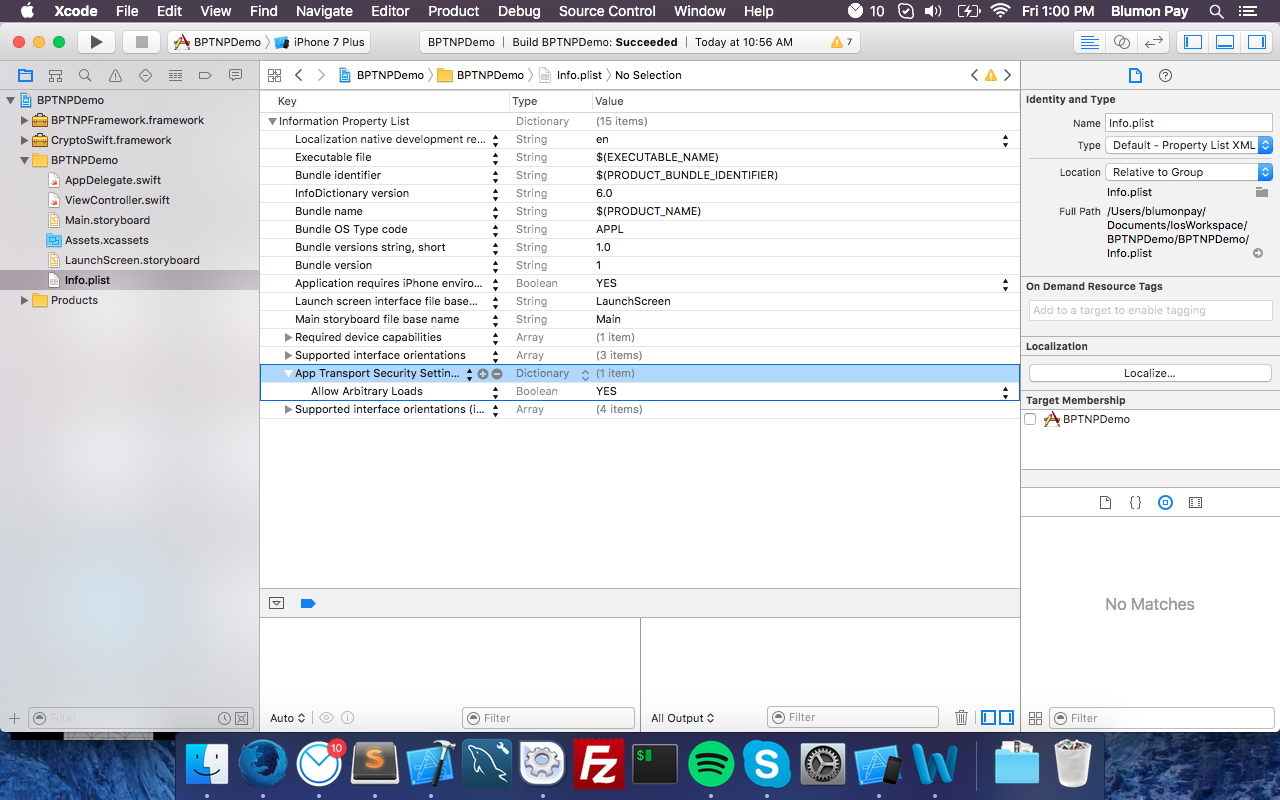


**Figura 1.6** Importar el framework en una clase swift

## 6 - Agregar permisos al info.plist

Es necesario implementar algunos permisos para realizar peticiones al exterior del dispositivo.

Seleccione el archivo info.plist, debajo de todas las opciones presione sus 2 dedos en el track pad para mostrar las opciones y seleccione add row, ingrese el siguiente texto ***NSAppTransportSecurity***, se creara un diccionario, expandir la opción y presionar sobre el + para agregar un valor con el siguiente texto ***Allow Arbitrary Loads*** que es un booleano, deberá estar en **yes**, ver figura 1.7



**Figura 1.7** Agregando permisos en info.plist

# Uso del Framework

Las funciones, procesos y usos del framework serán explicados a continuación, recuerde leer el documento completo muchas de las dudas podrían ser resueltas antes de que desarrolle cualquier cosa.

## Lista de métodos de la clase BPCodes

Esta es una de las clases publicas del framework que contiene los métodos y usos de esta clase; Son los siguientes.

**Sintaxis:**

BPCodes().getCodeDescription(code: intValue);

**Uso:**

Esta clase implemente el manejo de mensajes, ya sea que se realizo correctamente una operación o que hubo algún fallo, únicamente es necesario pasar un parámetro de tipo Int.

**Impórtate:**

Esta clase únicamente regresa un string dependiendo del código enviado, estos códigos son los que usa la librería, si usted decide implementar sus propios mensajes solo será necesario que revise que numero pertenece al mensaje devuelto de tal manera que usted sepa que numero refiere a una operación exitosa o de error.

Continúe leyendo, para cada función será mencionado su código si la operación fue exitosa.

**Valores posibles:**

000 : "Err 000: Unkwno",

001 : "Err 001: Cant Connect to Server BPC",

002 : "Err 002: Conection Success, but Response Data is NIL",

003 : "Err 003: Conection Success, but Response Cant Be Read",

004 : "Cod 004: Conection Successfull, Downloading Data to the Device",

005 : "Cod 005: Login Error, Incorrect Access Data",

006 : "Err 006: Login correct, but could not download data to device ( Fail to execute sqlite insert )",

007 : "Cod 007: Conection Successfull",

008 : "Cod 008: Cleaned Login Session",

009 : "Err 009: Could not Delete the Session Table (sqlite error)",

010 : "Err 010: Cant Read Data from firstLogin Table (sqlite error)",

011 : "Err 011: Cant Assign Values To PretransactionResponseDataSingleton",

012 : "Err 012: PreTransaction Denied",

013 : "Err 013: PreTransaction Success, but Response Cant Be Read",

014 : "Err 014: Login Sucess and Loadead, but Cant Assign Values to LoginSingleton",

015 : "Err 015: Transaction Success, but Response Cant Be Read",

016 : "Cod 016: Transaction Successfull",

017 : "Err 017: Transaction Denied, Response:,

018 : "Err 018: Transaction Sucess, but Cant Assign Values to TransactionResponseDataSingleton"

**Ejemplo de uso :**

import UIKit

import BPTNPFramework

class ViewController: UIViewController {

override func viewDidLoad() {

super.viewDidLoad()

print(BPCodes().getCodeDescription(001));

}

}

**Salida:**

Err 001: Cant Connect to Server BPC

## Lista de métodos de la clase RequestBPC

Esta clase contiene los métodos principales para realizar los pagos electrónicos; Los métodos y usos de esta clase son los siguientes.

**Sintaxis:**

RequestBPC().requestKnowIfSessionLoginIsEmpty();

**Uso:**

Cuando se use el método de iniciar sesión creara datos, por esto surge la necesidad de saber si ya existe este inicio y mejorar la velocidad de los procesos, con el método escrito anteriormente usted podrá conocer si ya se ah iniciado sesión alguna vez este regresara un valor:

* True -> si la sesión es vacía.
* False -> si ya existe una sesión iniciada.

**Sintaxis:**

RequestBPC().requestLogin("pruebasIOSTNP", password: "pruebasIOSTNP", serial: "777-555-333");

**Uso:**

Este método será usado para generar una sesión que permite establecer una conexión con el centro de pagos y así saber quien quiere acceder al servicio.

**Datos para framework de pruebas:**

Usar los mismos que en la descripción de la sintaxis.

**User:** pruebasIOSTNP

**Password:** pruebasIOSTNP

**Serial:** 777-555-333

**Importante:**

Este método debe ser lo primero que usted use antes de realizar una transacción, o de lo contrario fallara la operación.

El usuario, password y serial de producción, deberán ser solicitados a Blumon Pay.

**Códigos de Éxito:**

Si el inicio de sesión es correcto, deberá recibir un int con alguno de los siguientes valores. 004 o 007

Si usted recibe cualquier otro será un fallo, y debería imprimirlo así como denegar el inicio de una transacción.

**Sintaxis:**

RequestBPC().requestTransaction(flagIsTest: String, cardNumber: String, cardMonth: String, cardYear: String, cardMember: String, currency: String, concepts: String, address: String, amount: Float, phone: String, zipCode: String, CVV: String);

**Uso:**

Este método será usado para generar una transacción o pago, realizara el envió de datos al centro de pagos quien se encargara de realizar el cobro.

**Valores requeridos:**

**flagIsTest:** “1” o “0” **->** 1 pruebas, 0 producción ( su framework actual es pruebas)

**cardNumber:** “1234567891234567” **->** Numero de la tarjeta

**cardMonth:** “10” **->** Mes de expiración de la tarjeta

**cardYear:** “16” **->** Últimos 2 dígitos del año de expiración

**cardMember:** “Sergio Mejía R.” **->** Nombre del Propietario

**currency:** “MXN” -> Moneda

**concepts:** “Computadora ” **->** Concepto de la compra

**address:** “Avenida 123 Portales” **->** Dirección del propietario

**amount:** “9.99” **->** Monto de la compra

**phone:** “5533443344” **->** Teléfono

**zipCode:** “02410” **->** Código Postal

**CVV:** “123” **->** Código al reverso de la tarjeta

**Importante:**

Su aplicación deberá validar todos estos campos, para que sus transacciones sean exitosas, por ejemplo el tamaño de los números de tarjeta, el campo de mes y año sea solo dos dígitos etc; De tal forma que pueda obtenerse algo como los parámetros de valores requeridos mostrados anteriormente.

Para realizar esta acción usted debió haber realizado un inicio de sesión anterior a esta operación, de lo contrario fallara la ejecución de esta transacción.

**Códigos de Éxito:**

Si la transacción fue procesada correctamente regresara un int con valor: 016

Si usted esta usando la clase BPCodes, este sabrá a que refiere el código y si falla también.

**Sintaxis:**

RequestBPC().requestClearSessionLogin();

**Uso:**

Este método será usado para borrar los datos guardados de un inicio de sesión previo.

**Importante:**

Si su aplicación será usada por múltiples dispositivos, usted deberá agregar este borrado de sesión antes de cada operación ya que en el dispositivo se guarda un ID de inicio, pero en el centro de pagos también por seguridad; Ambos inicios deben coincidir para permitir realizar transacciones.

**Si usted inicia en 1 dispositivo tendría**

ID 1, ID BPC 1.  
**Si se inicia en otro dispositivo tendría**

ID 2, ID BPC 2.

Por lo cual la sesión en el dispositivo 1 seguiría iniciada es decir en 1, pero el ID del centro de pagos seria 2 la actual y cuando se validara el protocolo de seguridad usted estaría realizando una petición.

ID 1, ID BPC 2 , por lo cual fallaría la operación y no podría realizar el cobro o pago.

Entonces, si se pretende usar la aplicación para distribución o uso de múltiples dispositivos usted debe limpiar los datos de sesión antes de cada operación.

Si se pretendiera únicamente usar un dispositivo maestro para cobro, la operación seria ampliamente mas rápida por los datos de sesión iniciados en el dispositivo y no seria necesario borrar la sesión.

**Sintaxis:**

RequestBPC().requestLoginSessionData();

**Uso:**

Este método será usado para obtener los datos que nos responde el inicio de sesión, no es necesario obtenerlos pero si usted quiere acceder a ellos basta con realizar un print(RequestBPC().requestLoginSessionData());

Le dirá la posición de cada valor para poder llamarlos,

RequestBPC().requestLoginSessionData()[0];

Esto le devolvería el primer valor del arreglo.

**Importante:**

Para que este devuelva valores, se tendría que haber llamado anteriormente el método RequestBPC().requestLogin("user", password: "password", serial: "xxx-xxx-xxx"); de lo contrario la petición de datos no podrá ser llamada.

**Sintaxis:**

RequestBPC().requestTransaccionResponseData();

**Uso:**

Este método será usado para obtener los datos que nos responde la transacción una vez finalizada correctamente, podrán conocer el contenido realizando la siguiente instrucción print(RequestBPC().requestTransaccionResponseData());

Le dirá la posición de cada valor para poder llamarlos,

RequestBPC().requestTransaccionResponseData()[0];

Esto le devolvería el primer valor del arreglo.

**o** String(RequestBPC().requestTransaccionResponseData().valueForKey("transactionId") as! String;

Esto devolverá el valor de transaction id.

**Importante:**

Para que este devuelva valores, se tendría que haber llamado anteriormente el método RequestBPC().requestTransaction(parametros); de lo contrario la petición de datos no podrá ser llamada.

**Valores del arreglo:**

Aquí podrá obtener los siguientes valores relevantes, amount, reference, transaction id, entre otros valores.

# Ejemplo de transacción completa en consola.

En el mismo lugar que encontró este manual, podrá descargar este ejemplo y otro con entorno grafico. Usted deberá controlar los parámetros que envía a los métodos que realizan las transacciones para que este se procese correctamente.

## Código ejemplo swfit xcode

import UIKit

import BPTNPFramework

class ViewController: UIViewController {

override func viewDidLoad() {

super.viewDidLoad()

// Borrar sesiones en caso de que otro dispositivo use el mismo proceso con el mismo usuario.

RequestBPC().requestClearSessionLogin();

let requestLogin = RequestBPC().requestLogin("pruebasIOSTNP", password: "pruebasIOSTNP", serial: "777-555-333");

if requestLogin == 004 || requestLogin == 007 {

// Esto indica que el proceso de login fue realizado correctamente.

// Ahora podemos realizar la transaccion

let requestTransaction = RequestBPC().requestTransaction("1", cardNumber: "1234567891234567", cardMonth: "10", cardYear: "16", cardMember: "Sergi Mejia R.", currency: "MXN", concepts: "Compra de Laptop", address: "Haciendas 123", amount: 15.300, phone: "5566223344", zipCode: "02410", CVV: "123");

if requestTransaction == 016 {

// Esto indica que el proceso de transaccion fue realizado correctamente

// Usted podra mostrar mensaje de operacion correcta y otorgar valores de respuesta

print("Pago Exitoso");

print(RequestBPC().requestTransaccionResponseData().valueForKey("transactionId"));

print(RequestBPC().requestTransaccionResponseData().valueForKey("reference"));

}else{

// Fallo la transaccion e imprimimos el error

print(BPCodes().getCodeDescription(requestTransaction));

}

}else{

// Fallo el inicio de sesion e imprimimos el error.

print(BPCodes().getCodeDescription(requestLogin));

}

}

override func didReceiveMemoryWarning() {

super.didReceiveMemoryWarning()

// Dispose of any resources that can be recreated.

}

}

## Salida del ejemplo

Pago Exitoso

Optional(1220)

Optional(28102016122415)