Transferuj iOS Mobile Library

Krajowy Integrator Płatności Spółka Akcyjna z siedzibą w Poznaniu, przy ul. Św. Marcin 73/6, wpisana do rejestru przedsiębiorców Krajowego Rejestru Sądowego prowadzonego przez Sąd Rejonowy Poznań – Nowe Miasto i Wilda w Poznaniu, VIII Wydział Gospodarczy Krajowego Rejestru Sądowego pod numerem KRS 0000412357, numer NIP 7773061579, REGON 300878437, kapitał zakładowy 4 798 500,00 PLN (wpłacony w całości).

Transferuj iOS Mobile Library

1 Konfiguracja projektu

1. Biblioteka wspiera iOS w wersji 8.0 lub nowszej.
2. W środowisku Xcode 6 należy dołączyć do projektu framework TransferujSDK.framework dołączony do SDK.
    - Można to zrobić poprzez przeciągniecie pliku.
    - Dokumentacja Apple https://developer.apple.com/library/ios/recipes/xcode_help-project_editor/Articles/AddingaLibrarytoaTarget.html
3. Biblioteka zależy od frameworków UIKit.framework oraz Foundation.framework, które są dołączane wraz z TransferujSDK.framework.

2 Budowa frameworka

1. Biblioteka wspiera iOS w wersji 8.0 lub nowszej.
2. W środowisku Xcode 6 należy dołączyć do projektu framework TransferujSDK.framework dołączony do SDK.
    - Można to zrobić poprzez przeciągniecie pliku.
    - Dokumentacja Apple https://developer.apple.com/library/ios/recipes/xcode_help-project_editor/Articles/AddingaLibrarytoaTarget.html
3. Biblioteka zależy od frameworków UIKit.framework oraz Foundation.framework, które są dołączane wraz z TransferujSDK.framework.

3 Sposób użycia biblioteki w projekcie

Poniżej opisano możliwy sposób implementacji w projekcie.

1. Po poprawnym skonfigurowaniu projektu utwórz w *Storyboard*zie pusty *ViewController* nazwijmy go *PaymentViewController*.
2. Dodaj do niego *Cantainer View*. Nazwij (*Identifier*) segue do wewnętrznego kontrolera np. *PLTransferujPaymentInnerSegue*.
3. W utworzonym wewnętrznym *ViewController*ze ustaw *Custom Class* na *PLTransferujViewController*.
4. Następnie w pliku *PLTransferujViewController.h* dołącz nagłówek:

```
#import <TransferujSDK/PLTransferujViewController.h>
```

5. Pozostając w tym samym pliku zadeklaruj implementację protokołu

PLTransferujPaymentDelegate w kontrolerze *PLTransferujViewController*

```
@interface PaymentViewController : UIViewController <PLTransferujPaymentDelegate>
```

6. Zadeklaruj płatność oraz metody delegata.

```
@property (nonatomic, strong) PLTransferujPayment *payment;

...

- (void)transferujDidSucceedWithPayment:(PLTransferujPayment *)payment;

- (void)transferujDidFailedWithPayment:(PLTransferujPayment *)payment;
```

7. Przejdź do implementacji kontrolera *PaymentViewController.m*.

8. Utwórz w nim płatność i ustaw wymagane parametry zgodnie z dokumentacją na stronie transferuj.pl.

```
_payment = [[PLTransferujPayment alloc] init];


[_payment setMId:@"twoje_id"];

[_payment setMAmount:@"kwota_transakcji"];

[_payment setMDescription:@"opis_transakcji"];

[_payment setMClientEmail:@"email_klienta"];

[_payment setMClientName:@"imie_nazwisko_klienta"];

[_payment setMCrc:@"crc"];

[_payment setMSecurityCode:@"twój_kod"];
```

Można również ustawić gotowy, wygenerowany wcześniej link i wtedy konfiguracja obiektu reprezentującego płatność wygląda jak poniżej:

```
[_payment setMPaymentLink:@"wygenerowany_link_płatności"];
```

9. Aby rozpocząć proces płatności trzeba z dowolnego miejsca aplikacji wywołać *PaymentViewController*. W chwili wywołania wykonywany jest segue, który trzeba nazwać w *Storyboard*zie i przechwycić, w nim należy przekazać naszą płatność do wewnętrznego kontrolera oraz ustawić delegata dla zdarzeń.

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {

    NSString *segueName = segue.identifier;
```

```
    if ([segueName isEqualToString: @"PLTransferujPaymentInnerSegue"]) {
        PLTransferujViewController *childViewController = (PLTransferujViewController *) [segue destinationViewController];
        [childViewController setPayment:_payment];
        [childViewController setDelegate:self];
    }
}
```

Należy się upewnić, że została ona wcześniej odpowiednio utworzona.

10. Należy też dodać metody informujące o przebiegu transakcji.

```
- (void)transferujDidSucceedWithPayment:(PLTransferujPayment *)payment {
    UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Płatność" message:@"Płatność udana!" delegate:self cancelButtonTitle:@"OK" otherButtonTitles:nil];
    [alert show];
}

- (void)transferujDidFailedWithPayment:(PLTransferujPayment *)payment {
    UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Płatność" message:@"Płatność nie udana!" delegate:self cancelButtonTitle:@"OK" otherButtonTitles:nil];
    [alert show];
}
```

11. Jeżeli używamy metod zachowywania stanu aplikacji. Należy pamiętać o implementacji odpowiednich metod. Obiekt PLTransferujPayment może być kodowany i dekodowany za pomocą klasy *NSCoder*.

```
- (void)encodeRestorableStateWithCoder:(NSCoder *)coder {
    [coder encodeObject:_payment forKey:kExtraPayment];
    [super encodeRestorableStateWithCoder:coder];
}

- (void)decodeRestorableStateWithCoder:(NSCoder *)coder {
    _payment = [coder decodeObjectForKey:kExtraPayment];
```

```
    [super decodeRestorableStateWithCoder:coder];

}
```

Istnieje możliwość implementacji bez użycia *Storyboard*a, używając bezpośrednio widoku *PLTransferujViewController*, należy tylko pamiętać aby przekazać do *PLTransferujViewController* płatność oraz delegata, który obsłuży zdarzenia związane z płatnością.

4 Uwagi

Universal Framework

Przed zbudowaniem wersji uniwersalnej należy zbudować cel *FrameworkSDK* odpowiednio dla symulatora (np. iPhone 6 Plus) oraz dla urządzenia.

W celu zbudowania wersji uniwersalnej frameworka do SDK został dodany cel budowania *TransferujSDK-Universal*. Wersji uniwersalnej można używać w symulatorze x86_64.

Wynik budowania umieszczony jest w katalogu *Output*.

5 Historia zmian

Wersja 1.0 (Czerwiec 2015):

- Pierwsza wersja dokumentu.