SDK Dito 2.0

Objetivo

Este documento tem como objetivo auxiliar na configuração/utilização do novo SDK da Dito.

Instalação

Para instalar o SDK DITO em seu projeto arraste o arquivo DitoSDK.framework para seu projeto.

Configuração

Caso o projeto dê erro ao compilar após a cópia do SDK alegando que não encontrou suporte a arquitetura x86_64. Basta remover a arquitetura “arm64" do projeto conforme esta imagem.

Métodos disponíveis

O SDK conta com uma série de métodos para realizar as integrações disponíveis na DITO, estão descritos na tabela abaixo os métodos e suas respectivas funcionalidades.

Método	Função
configureEnviroment	Alterna entre os ambientes de desenvolvimento e produção.
configure	Realiza as configurações inicias e obrigatórias para realização das demais chamadas do SDK.
identify	Responsável por realizar o cadastro/login de um usuário na plataforma da DITO.
track	Responsável por realizar o tracking de eventos gerados pelo usuário, estando ele “online" ou “offline".
alias	Realiza à associação entre uma conta existente na DITO com outra(s) conta(s) quaisquer de redes sociais diversas.
unalias	Realiza a desassociação entre contas.
registerDevice	Responsável por habilitar um determinado device a receber notificações da DITO.
unregisterDevice	Responsável por desabilitar um determinado device a receber notificações da DITO.
request	Realiza chamadas diversas na API da DITO utilizando-se dos métodos disponíveis no protocolo HTTP (GET, PUT, DELETE, POST) com os parâmetros fornecidos pelo usuário.
notificationRead	Método para enviar a DITO que um determinado push notification foi aberto pelo usuário.

Descritivo dos métodos

Essa seção irá mostrar com é feita a utilização de cada método mencionado na seção anterior, tal como a passagem dos parâmetros e interações.

IMPORTANTE:

Todos os dados referentes a identificadores (ids) e access token das redes sociais contidos neste documento e no Sample de utilização do SDK são fictícios, ficando a cargo do desenvolvedor que irá implementar a SDK obte-los através das respectivas integrações com as redes sociais.

- ConfigureEnviroment

+(void)configureEnvironment:(EnvironmentType)environment;

Onde usar:
Este método, caso seja de interesse do desenvolvedor modificar o ambiente para realização de testes, deve ser chamado na classe AppDelegate do projeto.

NOTA: Caso este método não seja chamado **TODAS** as informações serão enviadas ao servidor de **PRODUÇÃO.**

Parâmetros:

Parâmetro	Obrigatório	Tipo	Valores Aceitos
environment	SIM	Enum (EnvironmentType)	PRODUCTION, DEVELOPMENT

Exemplo:
No arquivo AppDelegate.m adicione a seguinte chamada:

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:
(NSDictionary *)launchOptions {
        …
        [DitoAPI configureEnvironment:DEVELOPMENT];
        …

    return YES;
}
```

Adicione também o seguinte import:

```
#import <DitoSDK/DitoAPI.h>
```

- Configure

+(void)configure:(NSString *)apiKey secret:(NSString *)secret;

Onde usar:
Este método configura as credenciais para as integrações com a DITO e deve ser chamado na classe AppDelegate do projeto.

Parâmetros:

Parâmetro	Obrigatório	Tipo
apiKey	SIM	NSString
secret	SIM	NSString

Ambos os valores fornecidos pela DITO.

Exemplo:
No arquivo AppDelegate.m adicione a seguinte chamada:

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:
(NSDictionary *)launchOptions {
    …

    NSString *const kApiKey =
@“MjAxNS0wMy0zMSAxMToxOToyNCAtMDMwME1vYmlsZSBTREtzMTI4”;

    NSString *const kSecret = @“vutpDmAIUdMfVQgcGiGanCun4opBVRUJoqIIzyGi”;

    [DitoAPI configure:kApiKey secret:kSecret];

    …

  return YES;
}
```

Adicione também o seguinte import:

```
#import <DitoSDK/DitoAPI.h>
```

- Identify

+(void)identify:(DitoCredentials *)credentials accessToken:(NSString *)accessToken data:
(NSDictionary *)data completion:(void(^)(id response, NSError *error))block;

Parâmetros:

Parâmetro	Obrigatório	Tipo
credentials	SIM	DitoCredentials
accessToken	SIM (Somente não é obrigatório caso a credencial fornecida seja um “id" de um usuário do portal DITO).	NSString
data	NÃO	NSDictionary
completion	NÃO	Bloco

Exemplo:

Para utilizar o identify antes de mais nada é necessário criar um objeto da classe DitoCredentials, conforme mostrado abaixo, lembrando mais uma vez que as credenciais fornecidas abaixo tem apenas efeito de exemplificar a chamada:

```
NSString *const kFacebookID = @“10205082116146301”;

NSString *const kAccessToken =
@“CAAV9fG4JUfcBAH6KfAL3cbJx5Sq0CuKLWXaZCo3gLxneGPd3OZC5wxpvaLN2hL4reuRfLfkji4EzSYB8XxXHaXae5I5ZAH6BHZBAZB7VtyqPTJLRQ2yQoTDXgCbPHTvLZCWYYOPDgSjKBs4wssenZB8P8aqBFwAZAKlilfaj4y6zHDO2tdcWafkirQibT7ajMGrx3S2tJEOzTa2XhOq5mybS02xptJ0W8cPolov4XBVOtauTfGTlovMKss4SOjJdGtUBxBTAPG61NAZDZD”;

NSDictionary *kUser = @{@"name" : @“Nome do Usuário", @"email" :
@"email@do.usuario.br", @"data_nascimento" : @“1900-01-01"};

DitoCredentials *credentials = [[DitoCredentials alloc] initWithID:
facebookID:kFacebookID googlePlusID:nil twitterID:nil reference:nil];
```

E em seguida realizar a chamada ao método:

```
@try {
    [DitoAPI identify:credentials accessToken:kAccessToken data:kUser completion:^(id
    response, NSError *error) {
        // Tratar retorno aqui caso desejado
    }];
} @catch(NSException *exception) {
        NSLog(@“Erro: %@“, exception);
} @finally {
}
```

Adicione também o seguinte import:

```
#import <DitoSDK/DitoAPI.h>
```

- Track

+(void)track:(DitoCredentials *)credentials event:(NSDictionary *)event completion:(void(^)(id response, NSError *error))block;

Parâmetros:

Parâmetro	Obrigatório	Tipo
credentials	SIM	DitoCredentials
event	SIM	NSDictionary
completion	NÃO	Bloco

Exemplo:

Para o track também é necessário um objeto credentials com a mesma estrutura criada no método identify:

```
NSDictionary *kEvent = @{@"action" : @“Nome do Evento", @"revenue" : @“0.00", @"data" : @{@"name" : @“Nome”}};

@try {
    [DitoAPI track:credentials event:kEvent completion:^(id response, NSError *error) {
        // Tratar retorno aqui caso desejado
    }];
} @catch(NSException *exception) {
        NSLog(@“Erro: %@“, exception);
} @finally {
}
```

Adicione também o seguinte import:

```
#import <DitoSDK/DitoAPI.h>
```

- Alias

+(void)alias:(DitoCredentials *)credentials accounts:(NSArray *)accounts completion:(void(^)(id response, NSError *error))block;

Parâmetros:

Parâmetro	Obrigatório	Tipo
credentials	SIM	DitoCredentials
accounts	SIM	NSArray (Contendo objetos da classe DitoAccount)
completion	NÃO	Bloco

Exemplo:

Para o alias também é necessário um objeto credentials com a mesma estrutura criada no método identify. Além disso, as contas que irão ser associadas devem ser enviadas no parâmetro “accounts" em forma de um array:

```
DitoAccount *account = [[DitoAccount alloc] initWithID:@"1574730112774792" accessToken:@"CAAWC8vPa5uEBAFs59ouBWCkkA0cZCy8i8yaVFPZBni77N818DJhRTW8HLGBVGGqlXRsIigQldtbI2EAMGgPp5TjUg0AkIeNCyZCkC8mVAQnynUjWlktHIK2v9BrDjS4fIDASIMZBjCCdjucAtfJkSCxvjHhIa0AQ5Xqx2kjhK4ZBRpHQI2r8BZBmpZC4WnZCZA7YFn1zmI6VB2JRmuya8cg77Vi9xUhPIomZCxsu0nvQL3ygZDZD" type:FACEBOOK];
```

```
NSArray *kAccounts = @[account];

@try {
    [DitoAPI alias:credentials accounts:kAccounts completion:^(id response, NSError
    *error) {
        // Tratar retorno aqui caso desejado
    }];
} @catch(NSException *exception) {
        NSLog(@“Erro: %@“, exception);
} @finally {
}
```

Adicione também o seguinte import:

```
#import <DitoSDK/DitoAPI.h>
```

- Unalias

+(void)unalias:(DitoCredentials *)credentials accounts:(NSArray *)accounts completion:(void(^)(id response, NSError *error))block;

Parâmetros:

Parâmetro	Obrigatório	Tipo
credentials	SIM	DitoCredentials
accounts	SIM	NSArray (Contendo objetos da classe DitoAccount)
completion	NÃO	Bloco

Exemplo:

Para o unalias também é necessário um objeto credentials com a mesma estrutura criada no método identify. Além disso, as contas que irão ser associadas devem ser enviadas no parâmetro “accounts" em forma de um array:

```
DitoAccount *account = [[DitoAccount alloc] initWithID:@"1574730112774792"
type:FACEBOOK];

NSArray *kAccounts = @[account];

@try {
    [DitoAPI unalias:credentials accounts:kAccounts completion:^(id response, NSError
    *error) {
        // Tratar retorno aqui caso desejado
    }];
} @catch(NSException *exception) {
        NSLog(@“Erro: %@“, exception);
} @finally {
}
```

Adicione também o seguinte import:

```
#import <DitoSDK/DitoAPI.h>
```

IMPORTANTE:

Para os métodos de alias e unalias o objeto “DitoAccount" possui um atributo “type” que referencia o enum AccountsType, este enum indica a qual rede social a conta pertence, e seus valores podem ser:

FACEBOOK, TWITTER e GOOGLE_PLUS

- RegisterDevice

+(void)registerDevice:(DitoCredentials *)credentials deviceToken:(NSString *)deviceToken completion:(void(^)(id response, NSError *error))block;

Parâmetros:

Parâmetro	Obrigatório	Tipo
credentials	SIM	DitoCredentials
deviceToken	SIM	NSString
completion	NÃO	Bloco

Exemplo:

Para o registerDevice também é necessário um objeto credentials com a mesma estrutura criada no método identify.O device token pode ser obtido através do processo de “registerToPushNotification" e em seguida deve ser enviado a DITO para que aquele dispositivo esteja habilitado a receber notificações enviadas pela plataforma.

```
NSString *kDeviceToken = @"1234567890";

@try {
    [DitoAPI registerDevice:credentials deviceToken:kDeviceToken completion:^(id
    response, NSError *error) {
        // Tratar retorno aqui caso desejado
    }];
} @catch(NSException *exception) {
        NSLog(@“Erro: %@“, exception);
} @finally {
}
```

Adicione também o seguinte import:

```
#import <DitoSDK/DitoAPI.h>
```

- UnregisterDevice

+(void)unregisterDevice:(DitoCredentials *)credentials deviceToken:(NSString *)deviceToken completion:(void(^)(id response, NSError *error))block;

Parâmetros:

Parâmetro	Obrigatório	Tipo
credentials	SIM	DitoCredentials
deviceToken	SIM	NSString
completion	NÃO	Bloco

Exemplo:

Para o unregisterDevice também é necessário um objeto credentials com a mesma estrutura criada no método identify.O device token pode ser obtido através do processo de “registerToPushNotification" e em seguida deve ser enviado a DITO para que aquele dispositivo tenha o recebimento de notificações pela plataforma desabilitado.

```
NSString *kDeviceToken = @"1234567890";

@try {
    [DitoAPI unregisterDevice:credentials deviceToken:kDeviceToken completion:^(id
    response, NSError *error) {
        // Tratar retorno aqui caso desejado
    }];
} @catch(NSException *exception) {
        NSLog(@“Erro: %@“, exception);
} @finally {
}
```

Adicione também o seguinte import:

```
#import <DitoSDK/DitoAPI.h>
```

- Request

+(void)request:(NSString *)module path:(NSString *)path params:(NSMutableDictionary *)params requestType:(HttpTypes)requestType completion:(void(^)(id response, NSError *error))block;

Parâmetros:

Parâmetro	Obrigatório	Tipo	Valores Aceitos
module	SIM	NSString	
path	NÃO	NSString	
params	NÃO	NSMutableDictionary	
requestType	SIM	HttpTypes	POST GET PUT DELETE
completion	NÃO	Bloco	

Exemplo:

```
NSString *kModule = @“login";
NSString *kPath = @“/app”;
HttpTypes kMethod = GET;

[DitoAPI request:kModule path:kPath params:nil requestType:kMethod completion:^(id response, NSError *error) {
	// Tratar retorno aqui caso desejado
}];
```

Adicione também o seguinte import:

```
#import <DitoSDK/DitoAPI.h>
```

- NotificationRead

+(void)notificationRead:(NSString *)message completion:(void(^)(id response, NSError *error))block;

Parâmetros:

Parâmetro	Obrigatório	Tipo	Valores Aceitos
message	SIM	NSString	JSON que é enviado ao device por meio da notificação disparada a partir da plataforma da DITO.
completion	NÃO	Bloco	

Exemplo:

O JSON utilizado abaixo é apenas um exemplo de como o JSON será enviado pela plataforma da DITO.

```
NSString *kMessage = @"{\"notification\" : \”43847187\”, \”link\” : null}”

[DitoAPI notificationRead:kMessage completion:^(id response, NSError *error) {
        // Tratar retorno aqui caso desejado
}];
```

Adicione também o seguinte import:

```
#import <DitoSDK/DitoAPI.h>
```