ARTIGO: 176
IOS - Push Notification generate certify

De forma simples vou explicar os passos para gerar um certificado para um app IOS que precisa enviar notificações via push

1. Primeiramente você precisa ter uma conta na apple configurada corretamente com certificado de desenvolvedor
2. Precisa ter um dispositivo para testar. Não testei, mas pelo que li não funciona no emulador
3. Crie uma nova chave no aplicativo Acesso as Chaves, opção Assistente de Certifcado > Solicitar um Certificado de uma autoridade blá blá blá (*solicite uma para desenvolvimento e uma para produção*)
4. Dê um nome para o certificado, tipo MeuAppKey e seleciona a opção salvar em disco.
5. Agora no menu Chaves, a esquerda, você tem duas novas entradas para o certificado MeuApp, uma privada e uma pública (pem e .p12). Quando for salvar, salve uma versão da chave informando a senha e uma sem informar senha, com isso podemos testar.
6. Com o botão direito, salve as chave publica com o nome MeuApp**[Dev/Prod]**Key.p12 e a privada como MeuApp**[Dev/Prod]**Key.pem. (***[Dev/Prod]*** *-> uma para produção e uma para desenvolvimento*)
7. Agora, no painel da sua conta na Apple Store, adicione um novo App ID.
8. Marque o chekbox Push Notification e finalize
9. Após criado, selecione o App ID criado e vá em*edit*. Será exibida uma opção logo abaixo do*PushNotification*, que não está configurado, para criar um certificado.
10. Faça upload do arquivo .certSigningRequest gerado anteriormente (*você deve ter um para desenvolvimento e um para produção*)
11. Finalize o processo de criação do certificado e faça o download do arquivo aps_development.cer e aps.cer, uma para desenvolvimento e outra para produção respectivamente. Guarde a chave no mesmo local onde você guardou as chaves anteriores
12. Agora execute: (*faça o mesmo processo para desenvolvimento e produção*)

    No final das contas você tem que ter os seguintes arquivos: (*para não se perder use esse mesmo padrão de nomenclatura*)

```
MeuAppProd.certSigningRequest -> Solicita o certificado para produção
MeyAppDev.certSigningRequest -> Solicita o certificado para desenvolvimento

MeuAppDevKey.pem -> chave publica de produção gerado e salvo, não pede senha.
MeuAppProdKeySenha.pem -> chave publica produção gerado e salvo, não pede senha.. mas salve um com esse nome pra não se perder

MeuAppDevKey.p12 -> chave privada para desenvolvimento
MeuAppProdKeySenha.p12 -> chave privada para produção com a senha

aps.cer -> certificado para produção gerado pela apple
aps_development.cer -> certificado para desenvolvimento gerado pela apple
```

```
// use esse para criar o arquivo MeuAppDevCert.pem
openssl x509 -in aps_development.cer -inform der -out MeuAppDevCert.pem

// use os certificados salvos com senha
openssl pkcs12 -nocerts -out MeuAppDevKeySenha.pem -in MeuDevKeySenha.p12

// use os certificados salvos sem senha
openssl pkcs12 -out MeuAppDevKey.pem -in MeuAppDevKey.p12 -nodes
```

13. Teste o certificado.

```
// deve ser usado com o certificado salvo usando
openssl s_client -connect gateway.sandbox.push.apple.com:2195  -cert MeuAppDevCert.pem -key MeuAppDevKeySenha.pem
```

14. Voltando no seu painel no site da Apple, você deve criar um novo*Provisioning Profile* (um para desenvolvimento e um para produção) para seu aplicativo MeuApp. Isso é importante porque as notificações são enviadas via certificado.

15. Após criado, configure seu aplicativo com os perfis de desenvolvimento e produção criados recém criados

16. No seu aplicativo, no AppDelegate.m adicione o seguinte conteúdo:

```
// esse método deve apenas ser complementado, pois ele já existe

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

#ifdef __IPHONE_8_0
    //Right, that is the point
```

```
    UIUserNotificationSettings *settings = [UIUserNotificationSettings settingsForTypes:(UIUserNotificationTypeAlert
        | UIUserNotificationTypeBadge | UIUserNotificationTypeSound) categories:nil];
    [[UIApplication sharedApplication] registerUserNotificationSettings:settings];
#else
    //register to receive notifications
    [[UIApplication sharedApplication] registerForRemoteNotifications:
     (UIUserNotificationTypeBadge | UIUserNotificationTypeSound | UIUserNotificationTypeAlert)];
#endif
    return YES;
}

#ifdef __IPHONE_8_0

- (void)application:(UIApplication *)application didRegisterUserNotificationSettings:(UIUserNotificationSettings *)notificationSettings{
    //register to receive notifications
    [application registerForRemoteNotifications];
}

- (void)application:(UIApplication *)application handleActionWithIdentifier:(NSString *)identifier forRemoteNotification:(NSDictionary
*)userInfo completionHandler:(void(^)())completionHandler{
    //handle the actions
    if ([identifier isEqualToString:@"declineAction"]){
    }else if ([identifier isEqualToString:@"answerAction"]){
    }
}

#endif

- (void)application:(UIApplication *)application didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken{
   NSLog(@"## device token: %@", deviceToken);

}
-(void)application:(UIApplication *)application didFailToRegisterForRemoteNotificationsWithError:(NSError *)error{
   NSLog(@"## push notification error");
}
```

17. Colete o identificador do aplicativo: Compile, execute o app e confirme a mensagem de recebimento de notificações. Nos logs, copie o conteúdo do deviceToken. Remove os espaços desse identificador.

18. Usando o identificador coletado. Execute:

```
// use o certificado que foi salvo usando senha
cat MeuAppDevCert.pem MeuAppDevKeySenha.pem > ck.pem
```

19. Salve o código abaixo em um .php no mesmo diretório das chaves e execute para enviar uma mensagem de teste.

```
<?php
// Put your device token here (without spaces):
$deviceToken = 'seu identificador!!!!!!!'; // MeuApp
// Put your private key's passphrase here:
$passphrase = 'a senha usada para salvar a chave';
// Put your alert message here:
$message = 'My first push notification!';
////////////////////////////////////////////////////////////////////////////////
$ctx = stream_context_create();
stream_context_set_option($ctx, 'ssl', 'local_cert', 'ck.pem');
stream_context_set_option($ctx, 'ssl', 'passphrase', $passphrase);
// Open a connection to the APNS server
$fp = stream_socket_client(
  'ssl://gateway.sandbox.push.apple.com:2195', $err,
  $errstr, 60, STREAM_CLIENT_CONNECT|STREAM_CLIENT_PERSISTENT, $ctx);
if (!$fp)
  exit("Failed to connect: $err $errstr" . PHP_EOL);
echo 'Connected to APNS' . PHP_EOL;
// Create the payload body
$body['aps'] = array('alert' => $message, 'sound' => 'default');
// Encode the payload as JSON
$payload = json_encode($body);
// Build the binary notification
$msg = chr(0) . pack('n', 32) . pack('H*', $deviceToken) . pack('n', strlen($payload)) . $payload;
// Send it to the server
$result = fwrite($fp, $msg, strlen($msg));
if (!$result)
  echo 'Message not delivered' . PHP_EOL;
else
  echo 'Message successfully delivered' . PHP_EOL;
// Close the connection to the server
```

```
fclose($fp);
```

20. Esses são os cansativos passos para enviar uma simples notificação via*push*. Obrigado Apple, por facilitar tanto nossa nada complicada vida de programador!!

Fonte e ajuda: http://www.raywenderlich.com/32960/apple-push-notification-services-in-ios-6-tutorial-part-1