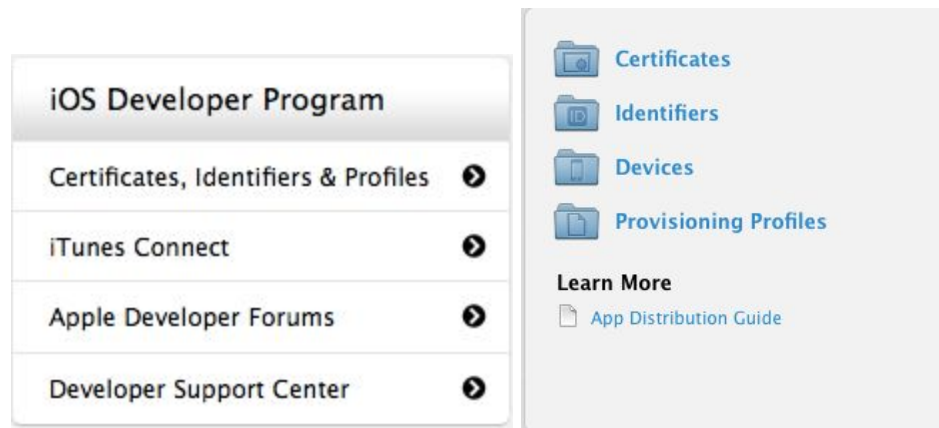


Manual Generación de certificados APN's (Apple Push Notifications)

Dentro del desarrollo de aplicaciones móviles, las Notificaciones Push (Push Notifications) toman un rol principal si lo que planea el desarrollador es notificar al usuario sobre algún evento importante dentro de la aplicación, por ejemplo, hubo un cambio importante dentro de los servicios de la aplicación. Con la problemática anterior se requiere implementar una solución y esta será la implementación de las Notificaciones Push. A continuación se muestra el procedimiento para poder incluir dichas notificaciones en nuestra aplicación.

Creando el AppID en el portal de desarrollo de Apple (<https://developer.apple.com/devcenter/ios/index.action>).

Tras entrar al centro de desarrolladores, al lado derecho aparece un menú como el que aparece en la imagen, seleccione la opción “Certificates, Identifiers & Profiles”.



Aparecerá un nuevo menú (vea la figura), seleccione la opción de “Identifiers”, en la nueva pantalla, seleccione el botón “+”, la siguiente pantalla solicita que le proporcionemos cierta información.

Lo primero que le proporcionamos es la descripción de nuestra aplicación.

El siguiente campo a llenar es el de llenar el identificador de nuestra aplicación. Seleccionamos la opción “Explicit App ID” y en la sección de “Bundle ID” le escribimos el mismo Bundle ID que tenemos en nuestro proyecto (en el archivo NombreProyecto-Info.plist).

Finalmente en la sección de App Services seleccionamos la opción de “Push Notifications” Una vez hecho esto daremos click en el botón “Continue”.



Registering an App ID

The App ID string contains two parts separated by a period (.)—an App ID Prefix that is defined as your Team ID by default and an App ID Suffix that is defined as a Bundle ID search string. Each part of an App ID has different and important uses for your app. [Learn More](#)

App ID Description

Name:

You cannot use special characters such as @, &, *, ", "

App ID Prefix

Value:

App ID Suffix

• Explicit App ID

If you plan to incorporate app services such as Game Center, In-App Purchase, Data Protection, and iCloud, or want a provisioning profile unique to a single app, you must register an explicit App ID for your app.

To create an explicit App ID, enter a unique string in the Bundle ID field. This string should match the Bundle ID of your app.

Bundle ID:

App Services

Select the services you would like to enable in your app. You can edit your choices after this App ID has been registered.

- Enable Services:
- ☐ App Groups
 - ☐ Associated Domains
 - ☐ Data Protection
 - ☐ Complete Protection
 - ☐ Protected Unless Open
 - ☐ Protected Until First User Authentication
 - ☒ Game Center
 - ☐ HealthKit
 - ☐ HomeKit
 - ☐ Wireless Accessory Configuration
 - ☐ Apple Pay
 - ☐ iCloud
 - ☐ Compatible with Xcode 5
 - ☐ Include CloudKit support (requires Xcode 6)
 - ☒ In-App Purchase
 - ☐ Inter-App Audio
 - ☐ Passbook
 - ☐ Push Notifications
 - ☐ VPN Configuration & Control

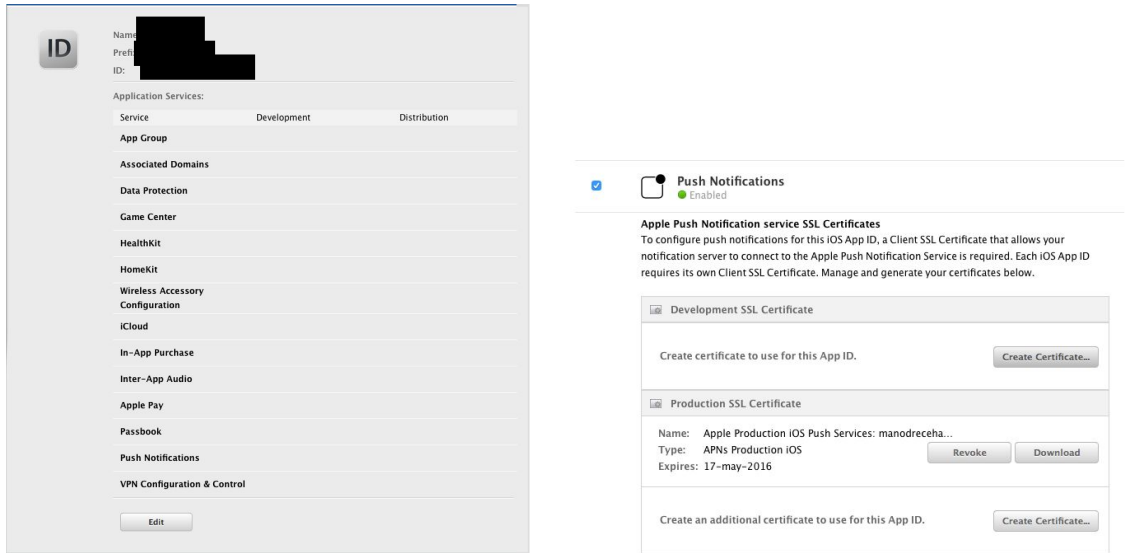
Cancel

Continue

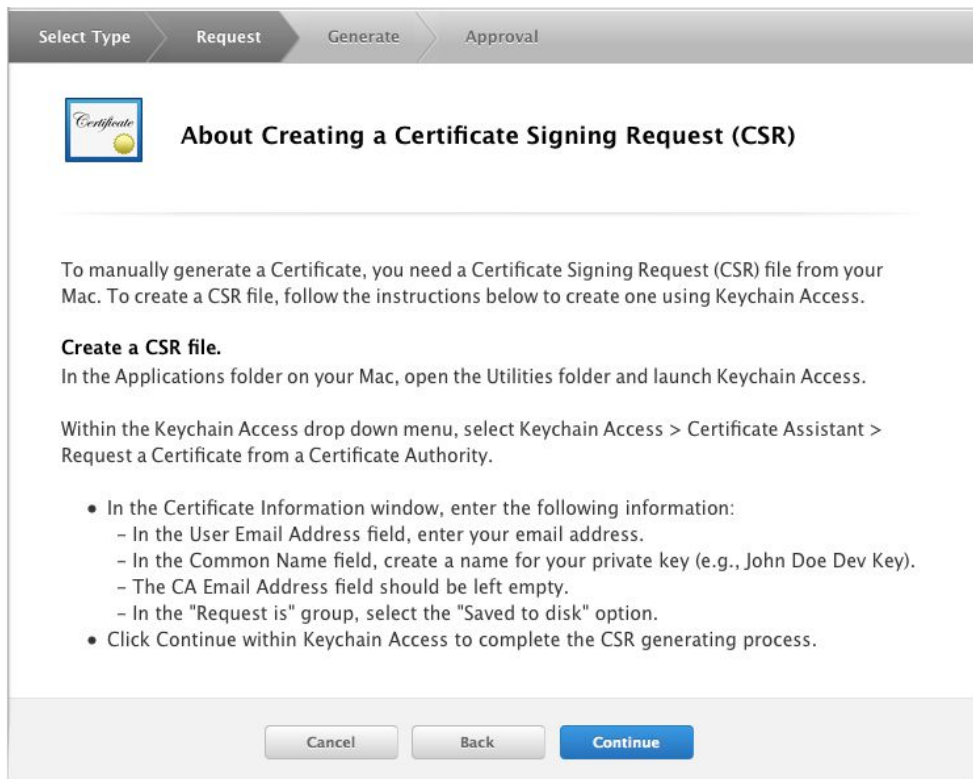
Una vez generado nuestro identificador lo buscaremos en la lista de identificadores y daremos click sobre él.

Al hacerlo se desplegará un menú dentro de este daremos click en el botón “Edit”. En la siguiente pantalla buscaremos la sección de Push Notifications. En la sección de “Development SSL Certificate” seleccionaremos “Create Certificate”

Aparecerá una pantalla de como generar un CSR File, a continuación se describe como hacerlo.



The top part of the image shows two screenshots from the Apple Developer portal. The left screenshot shows the 'Edit' button for an App ID. The right screenshot shows the 'Push Notifications' section, which is enabled. It lists 'Development SSL Certificate' and 'Production SSL Certificate' with 'Create Certificate...' buttons.



The bottom screenshot shows the 'About Creating a Certificate Signing Request (CSR)' window. It provides instructions for manually generating a CSR file using Keychain Access. The window has a progress bar at the top with steps: Select Type, Request, Generate, and Approval. The 'Request' step is currently active.

About Creating a Certificate Signing Request (CSR)

To manually generate a Certificate, you need a Certificate Signing Request (CSR) file from your Mac. To create a CSR file, follow the instructions below to create one using Keychain Access.

Create a CSR file.

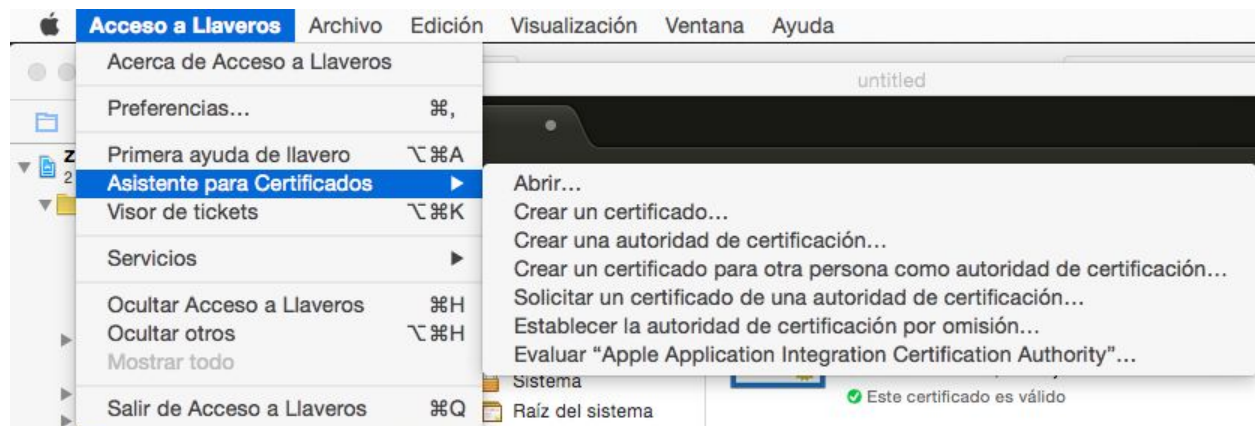
In the Applications folder on your Mac, open the Utilities folder and launch Keychain Access.

Within the Keychain Access drop down menu, select Keychain Access > Certificate Assistant > Request a Certificate from a Certificate Authority.

- In the Certificate Information window, enter the following information:
 - In the User Email Address field, enter your email address.
 - In the Common Name field, create a name for your private key (e.g., John Doe Dev Key).
 - The CA Email Address field should be left empty.
 - In the "Request is" group, select the "Saved to disk" option.
- Click Continue within Keychain Access to complete the CSR generating process.

At the bottom of the window, there are three buttons: Cancel, Back, and Continue.

Abra la aplicación “Acceso a llaveros”, posteriormente en la barra superior (la que tiene la fecha y hora seleccione la opción Acceso a Llaveros->Asistente de certificados->Solicitar un certificado a una autoridad de certificación...



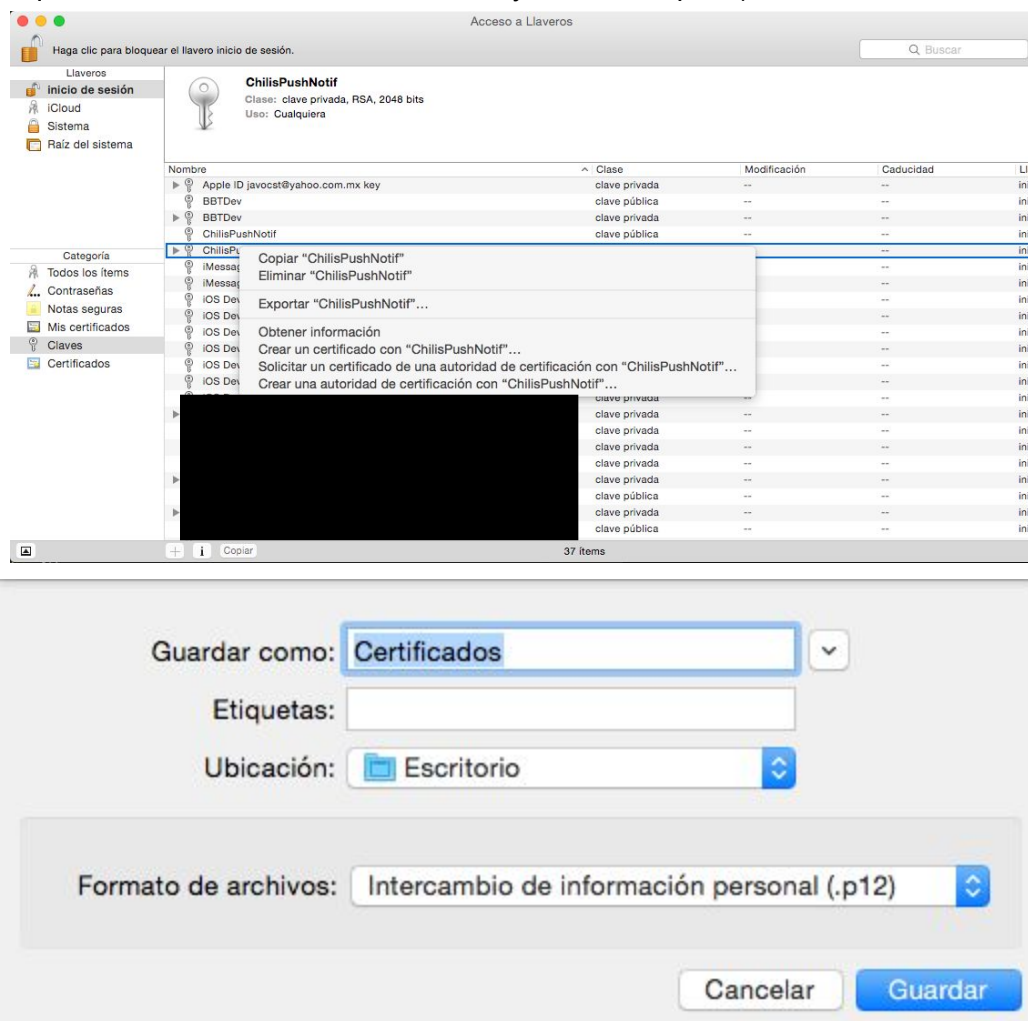
Aparecerá una pantalla como la de la figura, deberá llenar la dirección de correo del usuario, asignar un nombre común (que sea distintivo) y seleccionar la opción Se guarda en el disco. De click en continuar y guarde el archivo (Por default se guarda en el escritorio no cierre Acceso a llaveros aún).

De regreso al portal de desarrollo daremos click en el botón “Continue” y nos llevará a una nueva página donde daremos click en el boton “Choose File” y posteriormente en el botón “Generate”, finalmente de click en Download para bajar el certificado (guarde este archivo en la misma carpeta en la que guardó el archivo con el que generó el certificado).

Generando el archivo .p12

En Acceso a llaveros, al generar el archivo CSR (el que se usa para generar el certificado) genera un par de llaves en los llaveros, buscaremos por nombre común la llave privada. Daremos click secundario en la llave privada y seleccionaremos la opción “Exportar NombreLlave”

Al aparecer un menú desplegable, nos aseguraremos de que en la opción “Formato de archivos” esté seleccionado Intercambio de información personal (.p12) y daremos click en guardar dándole el nombre llave. Nos pedirá primero que asignemos una contraseña para el archivo .p12 después nos pedirá la contraseña de nuestra mac (guardaremos en el mismo directorio que tenemos nuestros archivos CSR y certificado push).



Generación de los archivos .pem

Hasta ahora, en el mismo directorio tenemos 3 archivos (csr,llave.p12 y aps_development.cer).
Abriremos una terminal.

Nos situaremos en la carpeta en la que tenemos nuestros 3 archivos, para efectos demostrativos la carpeta de ejemplo se llama “chilis-push” entonces en el terminal escribiremos:

```
cd Desktop/chilis-push/
```

Daremos enter y nos llevará a la carpeta, en caso de que le demos el comando `ls` nos mostrará los archivos dentro del directorio.

Primero convertiremos el certificado aps_development.cer en un archivo .pem tecleando en la terminal lo siguiente:

```
$ openssl x509 -in aps_development.cer -inform der -out PushCert.pem
```

Convertiremos la llave privada (archivo .p12) en un archivo .pem tecleando lo siguiente en la terminal:

```
$ openssl pkcs12 -nocerts -out PushKey.pem -in llave.p12
```

O en caso de que se requiera, la llave puede ser importada a un archivo .pem sin passphrase con el siguiente comando:

```
$ openssl pkcs12 -in <key>.p12 -out <key>.pem -nocerts -nodes
```

La instrucción anterior te pedirá primero la contraseña del archivo .p12 (véase más arriba) y posteriormente que se le asigne una contraseña nueva al archivo .pem (no olvide esta contraseña).

Finalmente combinaremos ambos archivos en uno solo tecleando lo siguiente en la terminal:

```
$ cat PushCert.pem PushKey.pem > ck.pem
```

Ahora hay que probar que nuestro certificado así como nuestra llave funcionen tecleamos lo siguiente en la terminal:

```
$ telnet gateway.sandbox.push.apple.com 2195
```

Esto hará una conexión, sin encriptar al APNS server. Si la respuesta es como la siguiente:

Trying 17.172.232.226...

Connected to gateway.sandbox.push-apple.com.akadns.net.

Escape character is '^['.

Entonces tu mac puede hacer Push Notifications, de lo contrario, entonces deberás revisar la configuración de tu firewall.

Ahora nos conectaremos haciendo uso de nuestros certificado y llave SSL tecleando lo siguiente en la terminal:

```
$ openssl s_client -connect gateway.sandbox.push.apple.com:2195  
-cert PushCert.pem -key PushKey.pem
```

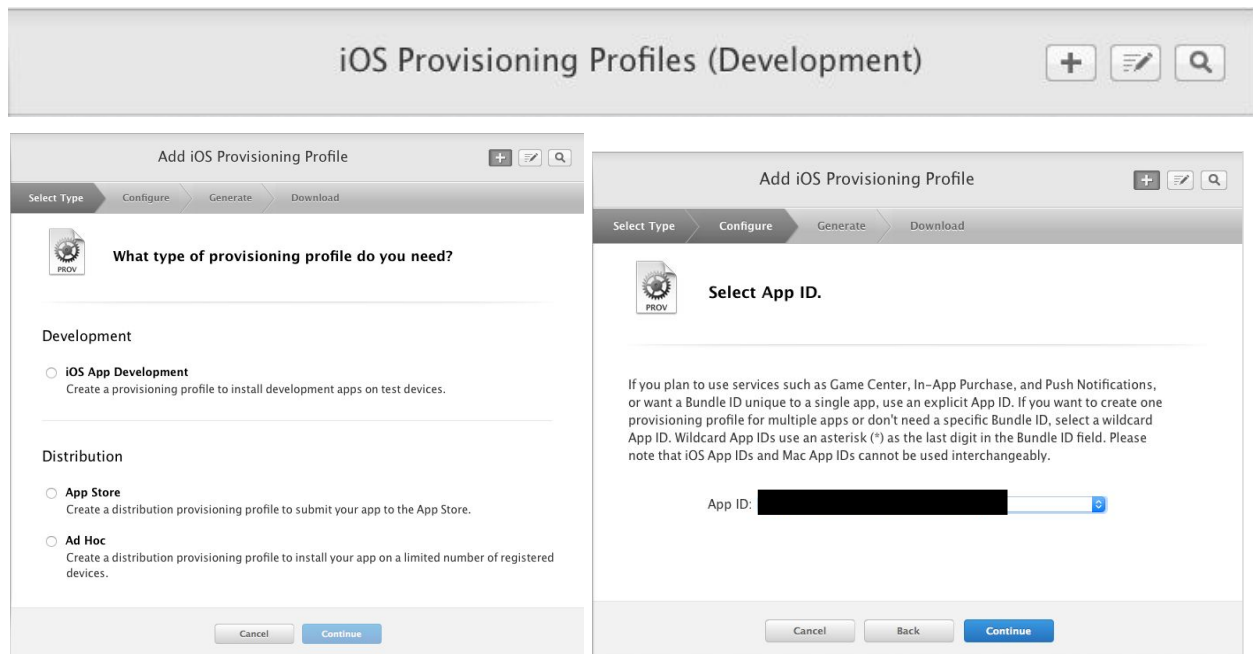
Te arrojará una respuesta bastante grande, esta respuesta te indica cómo trabaja openssl bajo el telón. Si la conexión resulta exitosa, la terminal te dejará teclear algunos caracteres y cuando presiones Enter, el servidor se desconectará de forma automática.

Generando el Provisioning Profile

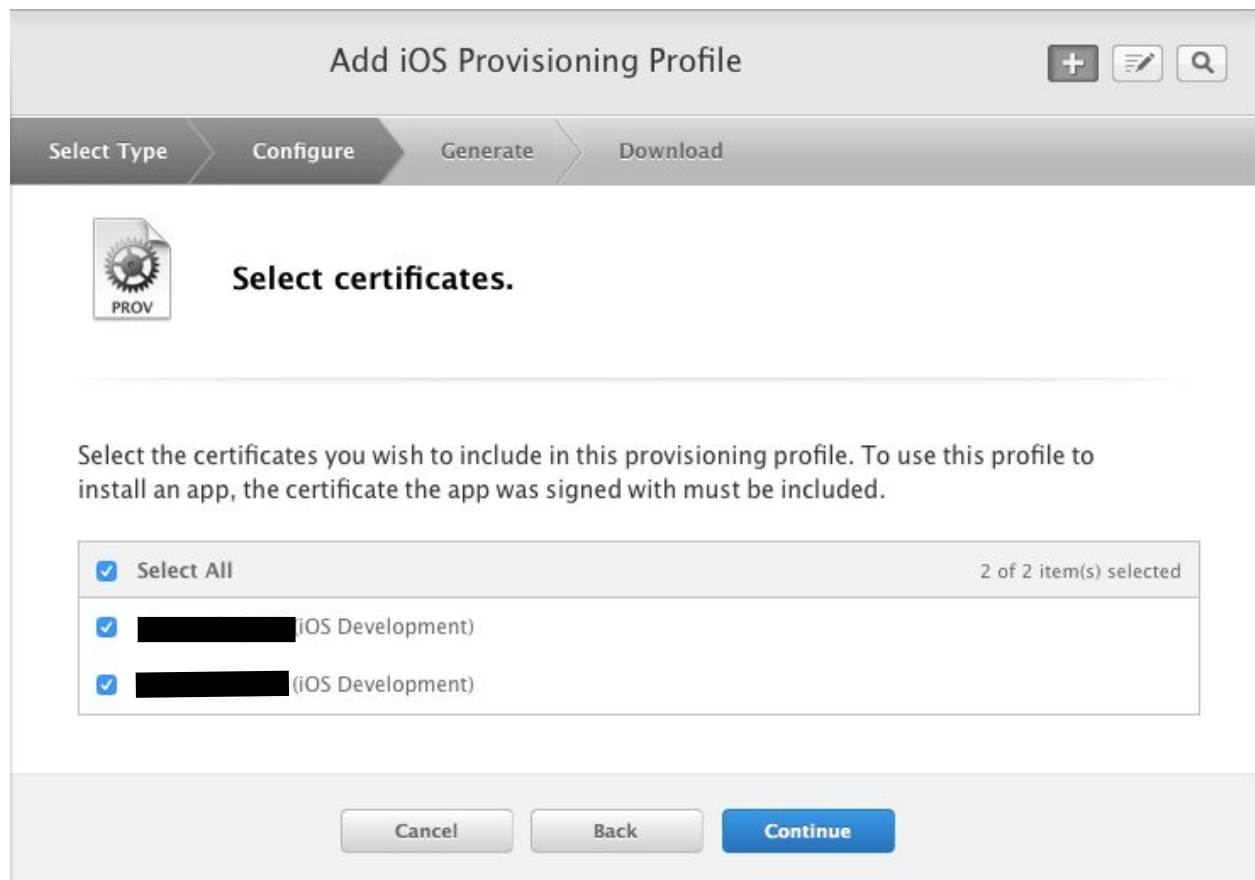
De nuevo en el portal de desarrollo de Apple, Haremos click en la sección de Provisioning Profiles y nos mostrará los perfiles activos así como los inválidos.

Le daremos click al botón “+” y nos aparecerá la siguiente pantalla:

Seleccionaremos la opción iOS App Development y daremos click en el botón “continue” y nos llevará a una nueva pantalla.



Aquí elegiremos nuestra App ID que creamos con anterioridad. Le daremos click en “Continue” la cual nos llevará a una pantalla nueva la cual nos pedirá que elijamos un con que identidad de desarrollo deseamos firmar nuestro código.

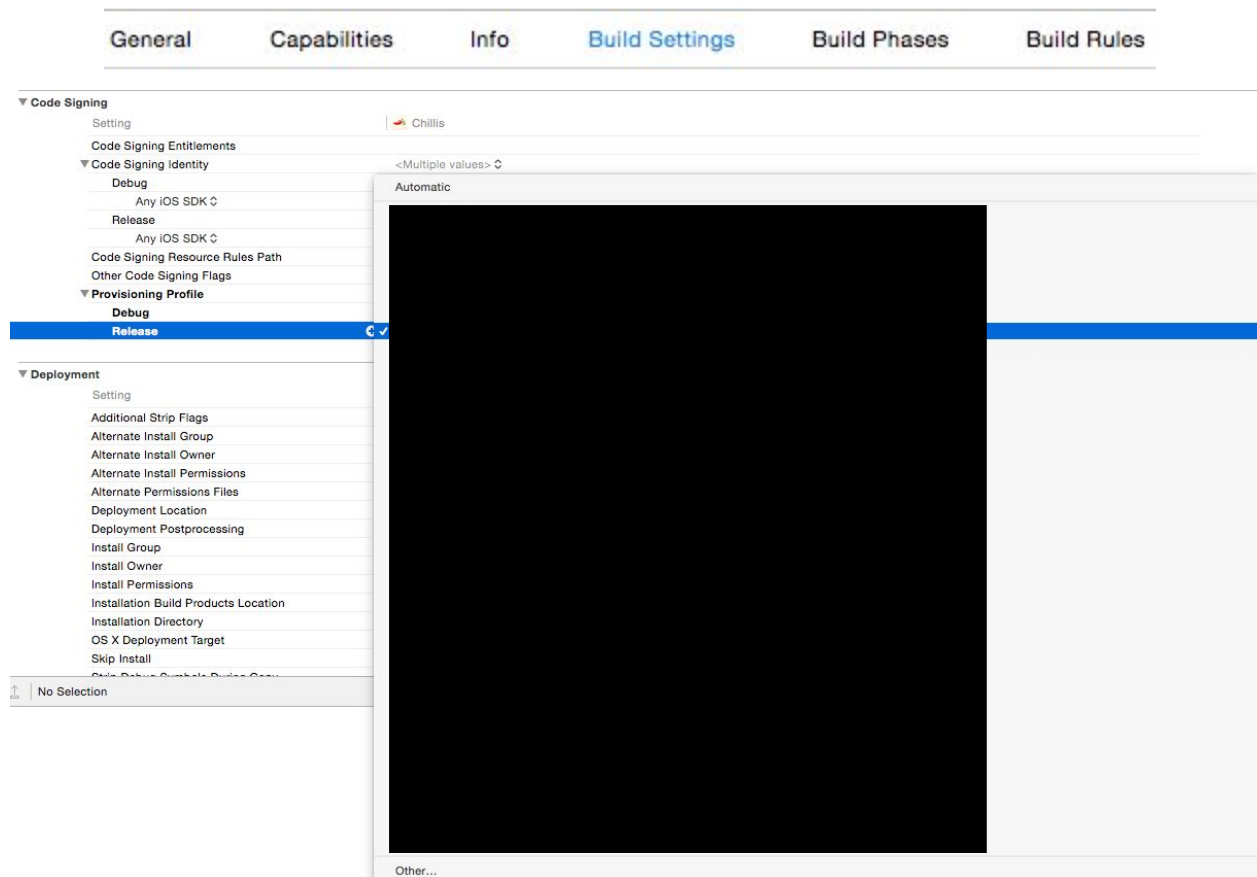


Seleccionaremos nuestra identidad y daremos click en “Continue”, en caso de que nuestro provisioning profile sea de desarrollo nos preguntará con qué dispositivos deseamos probar (podemos agregar hasta 100 dispositivos).

Daremos click en continuar y nos pedirá que le demos un nombre a nuestro provisioning profile y le daremos click en “Generate” para generar el perfil y posteriormente descargarlo. Lo descargamos e instalamos dando doble click sobre él.

Configurando el proyecto de Xcode.

Abriremos nuestro proyecto al cual deseamos agregarle las Push Notifications, daremos click en la hoja azul en la barra lateral izquierda, y cambiará la parte central daremos click en donde dice “Build Settings”, seleccionaremos la letra “A” que está formada por un lápiz un pincel y una regla, seleccionaremos donde dice “All” y buscaremos la sección de nombre “Code Signing”. Desplegamos el menú “Code Signing Identity” y firmaremos según sea el caso, para desarrollo o para producción en Debug y Release con el provisioning profile que recién generamos (no te preocupes debe ser el único con el que te dejará firmar aparte de la identidad genérica).



Configuración del código

Finalmente entraremos en código de Objective-C para indicarle a nuestra aplicación que podrá recibir notificaciones.

Primero en el archivo AppDelegate.m en el método

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

escribiremos lo siguiente:

```
[[UIApplication sharedApplication]  
registerForRemoteNotificationTypes: (UIRemoteNotificationTypeBadge |  
UIRemoteNotificationTypeSound | UIRemoteNotificationTypeAlert)];
```

De igual forma en el archivo AppDelegate.m escribiremos los siguientes métodos.

```
- (void)application: (UIApplication*) application  
didRegisterForRemoteNotificationsWithDeviceToken: (NSData*) deviceToken  
{  
    NSLog(@"My token is: %@", deviceToken);  
}  
  
- (void)application: (UIApplication*) application  
didFailToRegisterForRemoteNotificationsWithError: (NSError*) error  
{  
    NSLog(@"Failed to get token, error: %@", error);  
}
```

Al ejecutar la app (cmd+r) la aplicación deberá mandar una alerta de que esa aplicación desea mandar Push Notifications que si deseas permitir las push notifications. Al dar la autorización, en la consola de debug de Xcode deberá salir algo como:

My token is: <740f4707 bebcf74f 9b7c25d4 8e335894 5f6aa01d a5ddb387 462c7eaf 61bb78ad>

Si ejecutas la app en el simulador, la aplicación te lanzará un error indicando que no puede tener Push Notifications en el simulador.

Mandando nuestra primera notificación.

A continuación listamos un pequeño script que manda las push notifications.

```
<?php
// Put your device token here (without spaces):
$deviceToken =
'0f744707bebcf74f9b7c25d48e3358945f6aa01da5ddb387462c7eaf61bbad78';
// Put your private key's passphrase here:
$passphrase = 'pushchat';
// 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', $passphrase);
stream_context_set_option($ctx, 'ssl', 'cafile',
'entrust_2048_ca.cer');
// 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);
?>
```

Lo único que debemos de cambiar son los siguientes tres campos:

```
// Put your device token here (without spaces):
$deviceToken =
'0f744707bebcf74f9b7c25d48e3358945f6aa01da5ddb387462c7eaf61bbad78';

// Put your private key's passphrase here:
$passphrase = 'pushchat';

// Put your alert message here:
$message = 'My first push notification!';
```

En la variable `$deviceToken` quitamos lo que está entre comillas, y ponemos el que nos arroja la consola de Xcode (sin espacios) en la variable `$passphrase` cambiamos lo que está entre comillas y le escribimos nuestra contraseña que usamos para generar el archivo `PushKey.pem` y en él la variable `$message` quitamos lo que está entre comillas y ponemos un mensaje cualquiera.

Ahora copiamos el archivo **simplepush.php** al mismo directorio donde generamos el archivo `ck.pem` y escribimos en la terminal lo siguiente:

```
$ php simplepush.php
```

si todo va bien, el script debe arrojar lo siguiente:

```
Connected to APNS
Message successfully delivered
```

Y la notificación llegará en breve.