Your changes have been saved in version 5.

Last modified 0 seconds ago

AllShoppings iOS API

La integración a la app, como se aprecia en el proyecto de ejemplo, es sencilla. La parte más compleja que se encuentra, es la generación de certificados digitales para APNS, los cuales son obligatoriamente necesarios para cada App en particular.

Hay una documentación muy buena, rápida y práctica en: http://quickblox.com/developers/How_to_create_APNS_certificates. Otra recomendada (sobre todo para usar un método simple de testing) es : http://www.raywenderlich.com/32960/apple-push-notification-services-in-ios-6-tutorial-part-1

Esta API puede enlazarse de dos maneras en un proyecto de XCode:

1) Puede ser compilada y enlazada como biblioteca, 2) Puede ser directamente asociada al proyecto.

La función del API es encargarse de dos cosas en específico:

1) Mantener la geolocalización constante del dispositivo sin tener una influencia grande en la batería (activa el GPS solamente 10 segundos cada 2 minutos), ya que el footprint que tiene es menor a 100K de RAM y el consumo de CPU no llega al 1% en una hora. 2) Recibir mensajería Push del servidor de AllShoppings para dos propósitos: Recibir las ofertas y cupones por un lado, y recibir mensajes de activación que sirven para enviar la localización aún cuando el teléfono ha sido reiniciado y no han abierto aún la app.

Para hacer funcionar el módulo será necesario un appId y un token de allshoppings que identifique a la aplicación mobile que se construya, adicionalmente al servidor o ambiente donde se desee apuntar. Por ejemplo:

appId: televisa_mx
token: B8C7BC1F60DD840345975E847F24D89A3817930E6B20A42F6B063E8BF251B327
serverUrl: http://api.allshoppings.mobi

Así mismo, es conveniente agregar un parámetro NSString appVersion que represente a la versión de la aplicación mobile construida, a modo de tracking de respuestas y errores.

Asimismo, la configuración del API permite (opcionalmente) agregar datos de registro del usuario. La información que se puede agregar es:

- email
- Nombre
- Apellido
- Género (M para masculino, F para femenino)
- Fecha de Nacimiento (En formato yyyy-mm-dd)

Estos datos de usuario no son obligatorios, pero son recomendables ya que con esto es posible generar una interacción mucho más rica para los objetivos de la aplicación a construir.

Hay dos capabilities que hay que activar al proyecto de XCode para que funcione la integración:

- Background Modes
  - Location Updates
  - Remote Notifications

Se integra, como se muestra en el proyecto de testing, directamente en el AppDelegate.m:

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

    NSLog(@"application didFinishLaunchingWithOptions");

    // Initialize AllShoppings
    NSMutableDictionary * parameters = [NSMutableDictionary dictionaryWithCapacity:4];
    [parameters setObject:@"B8C7BC1F60DD840345975E847F24D89A3817930E6B20A42F6B063E8BF251B327" forKey:@"appToken"];
    [parameters setObject:@"http://api.allshoppings.mobi" forKey:@"serverURL"];
    [parameters setObject:@"televisa_mx" forKey:@"appId"];
    [parameters setObject:@"10000" forKey:@"appVersion"];

    ASUser * user = [[ASUser alloc]init];
    user.email = @"12345@televisa";
    user.firstname = @"Matias";
    user.lastname = @"Hapanowicz";
    user.gender = @"M";
    user.birthdate = @"1981-10-13";

    [parameters setValue:user forKey:@"user"];

    AllShoppingsIntegrationAPI *api = [AllShoppingsIntegrationAPI sharedInstance];
    [api initializeWithProperties:parameters];
    [api setMessageAction:^(NSDictionary *message) {
        NSError *error;
        NSData *jsonData = [NSJSONSerialization dataWithJSONObject:message options:NSJSONWritingPrettyPrinted // Pass 0 if you don't care
        NSString *jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8StringEncoding];
        UIAlertView *m = [[UIAlertView alloc] initWithTitle:@"Hello World!"
                                            message:jsonString
                                            delegate:nil
                                            cancelButtonTitle:@"OK"
                                            otherButtonTitles:nil];
        [m show];
    }];

    [api startUp];

    // Override point for customization after application launch.
    return YES;
}
```

Por último, es necesario implementar los siguientes selectors para poder dar servicio a la mensajería push:

```
- (void)application:(UIApplication *)application didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken {
    [[AllShoppingsIntegrationAPI sharedInstance].pushHelper didRegisterForRemoteNotificationsWithDeviceToken:deviceToken];
}

- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo {
    [[AllShoppingsIntegrationAPI sharedInstance].pushHelper application:application didReceiveRemoteNotification:userInfo];
}

- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo fetchCompletionHandler:(void (^)(U
{
    [[AllShoppingsIntegrationAPI sharedInstance].pushHelper application:application didReceiveRemoteNotification:userInfo fetchCompletion
}

- (void)applicationDidBecomeActive:(UIApplication *)application {
    [[AllShoppingsIntegrationAPI sharedInstance].pushHelper applicationDidBecomeActive:application];
}
```

El sistema de mensajería push trabaja pasando el mensaje primero por la infraestructura del API (para determinar si el mensaje es para levantar o bajar algún servicio, por ejemplo el de geolocalización), y después lo delega al control que sea necesario según el flow de la aplicación. Para esto, se pasa directamente el puntero a un selector que atrape el mensaje push luego de que AllShoppings lo interprete. Esto tiene que configurarse a momento de inicialización del API. Un ejemplo:

```
    [api setMessageAction:^(NSDictionary *message) {
        NSError *error;
        NSData *jsonData = [NSJSONSerialization dataWithJSONObject:message options:NSJSONWritingPrettyPrinted // Pass 0 if you don't care
        NSString *jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8StringEncoding];
        UIAlertView *m = [[UIAlertView alloc] initWithTitle:@"Hello World!"
                                                    message:jsonString
                                                    delegate:nil
                                                    cancelButtonTitle:@"OK"
                                                    otherButtonTitles:nil];
        [m show];
    }];
```

Lo que hace este ejemplo, es que todo mensaje push interactivo que llegue sea mostrado en formato JSON en un UIAlertView.

Entonces para resumir la integración:

- Generar los certificados digitales según el método expuesto en la documentación adjunta
- Enlazar la librería .a o el proyecto del API al proyecto de la aplicación custom.
- Agregar las capabilities de Background Modes
  - Location Updates
  - Remote Notifications
- Implementar en AppDelegate los siguientes métodos:

```
- (void)application:(UIApplication *)application didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken
- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo
- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo fetchCompletionHandler:(void
- (void)applicationDidBecomeActive:(UIApplication *)application
```

- Agregar la inicialización en application:didFinishLaunchingWithOptions
- Primero inicializar el API con un NSDictionary que le provea los atributos según el ejemplo,
- Luego inicializar el callback para los mensajes push
- Luejo ejecutar el selector startUp del API

- El servicio para obtener la lista de cupones activos está en http://api.allshoppings.mobi/app/coupons. Como parámetros recibe el authToken de la app, el identificador del dispositivo desde donde se está llamando (Ese identificador se genera al inicializar la biblioteca de AllShoppings), y un flag que sirve para determinar si debe responder solamente con las promociones activas o con las vencidas también.

  Ejemplo de la llamada:

  http://api.allshoppings.mobi/app/coupons?authToken=2A2C192578F597CEE6BC3EE8A26B9F7DB0DBA4B0CD8930FDC3D2BB0E7D1B8CC9&deviceUUID=c254507c3690abd9&activeOnly=true

  En este caso los campos son:

  authToken: Token de autorización de la app
  deviceUUID: Identificador del telefono, tal como fue generado por la biblioteca de allshoppings
  activeOnly: Para filtrar solo por promociones activas. Si este parámetro no se establece, entonces trae no solo las promociones activas sino también las vencidas

  La respuesta es un mensaje JSON con la lista de promociones:

```
{
    "systemDateTime": "2015-04-29T14:20:25+00:00",
    "data": [
        {
            "ageTo": 99,
            "financialEntityId": "",
            "limitDateTime": "2015-04-29T15:05:06+00:00",
            "campaignId": "",
```

```
            "avatarId": "5f1d5e6f-4854-4ab5-b39a-14ccae108a71.png",
            "lastUpdate": "2015-04-29T14:20:06+00:00",
            "genders": [
                "male",
                "female"
            ],
            "brandId": "cinepolis_mx",
            "brandName": "",
            "redeemDateTime": "",
            "validTo": "2015-08-31T00:00:00+00:00",
            "GMT": "",
            "systemDateTime": "2015-04-29T14:20:25+00:00",
            "description": "Avengers, La era de Ultrón. Entrada de último minuto.",
            "campaignSpecialId": "1430288511084",
            "name": "Avengers: La era de Ultron",
            "userId": "mhapanow@hotmail.com",
            "creationDateTime": "2015-04-29T14:20:06+00:00",
            "trigger": "",
            "storeName": "",
            "redeemStatus": 0,
            "offerTypeName": "Promoción",
            "couponCode": "",
            "checkinId": "",
            "shoppingId": "",
            "creationDate": "2015-04-29T00:00:00+00:00",
            "ageFrom": 0,
            "shoppingName": "",
            "validFrom": "2015-04-01T00:00:00+00:00",
            "identifier": "1430317206802",
            "storeId": "",
            "offerTypeId": "1379869942406",
            "statusChangeDateTime": "",
            "span": 2700000
        }
    ]
}
```


De aquí los campos importantes son:

identifier: ID del cupón. Este no es el número de folio, es un identificador interno para hacer referencia a esta entrada en particular.
avatarId: Es la imágen de la promoción. Aquí muestra solamente el nombre de archivo. Para acceder a la imágen, hay que irla a buscar a http://api.allshoppings.mobi/img/{id}. Por ejemplo para este caso, sería http://api.allshoppings.mobi/img/5f1d5e6f-4854-4ab5-b39a-14ccae108a71.png. Un detalle con las imágenes: pueden recibir un parámetro de tamaño para no transferir datos innecesariamente. Por ejemplo, con la llamada http://api.allshoppings.mobi/img/5f1d5e6f-4854-4ab5-b39a-14ccae108a71.png?width=50, Devuelve la misma imágen escalada a un width de 50px, con el consecuente ahorro de tasa de transferencia.
userId: Identificador de usuario al que está asignada la promoción
name: Nombre de la promoción.
description: Descripcion de la promoción.
span: Duración de la promoción (expresada en milisegundos)
creationDateTime: Cuando fue emitida la promoción.
limitDateTime: Fecha límite (hasta cuando se puede canjear)

Todas las fechas están expresadas en GMT (por eso terminan en +00:00). Lo que significa que para mostrar la hora de Mexico, hay que compensar la zona horaria (-5 en Verano, -6 en Invierno).

- Para evitar recibir una promoción frente a un disparador externo, hay un servicio que permite bloquear temporalmente las promociones al usuario del teléfono.

  http://api.allshoppings.mobi/app/deviceMessageLock?authToken=2A2C192578F597CEE6BC3EE8A26B9F7DB0DBA4B0CD8930FDC3D2BB0E7D1B8CC9

  Este servicio se invoca por el método POST, y recibe como parámetro un objeto JSON con el siguiente formato:


```
{
    "deviceUUID": "c254507c3690abd9",
    "scope": 1,
    "campaignActivityId": "1234567",
    "duration": 60000
}
```


  authToken: Token de autorización de la app. Hay que enviarlo como parte de la URL de Llamada
  deviceUUID: Identificador del telefono, tal como fue generado por la biblioteca de allshoppings
  scope: Alcance del bloqueo. Dejar fijo en 1
  campaignActivityId: Es el identificador de promoción que provocó el bloqueo. Este parámetro es opcional. Esto sirve en el caso de que al llegar un mensaje push con una promoción, la app detecte que el usuario ya compró entrada, por lo cual responde automáticamente invocando este servicio con el identificador de promoción reicibido en el mensaje push
  duration: Duración del bloqueo expresada en mili segundos (1000 ms = 1 segundo)

  Este servicio responde con un mensaje genérico, indicando la hora de emisión de la respuesta:


```
{
    "status": 1,
    "systemDateTime": "2015-04-29T14:37:15+00:00"
}
```


  Esta devolución es solamente utilizada para saber que el servicio fue invocado con éxito.

- Si se desea testear en un dispositivo orientado a un ambiente fuera de producción, existe un servicio que lo que hace es crear una promoción para el usuario tester y enviarsela. Aquí hay un ejemplo de un snippet de codigo shell script para llamar a este servicio:

```
# Script TestTicket

if [ $# -lt 1 ]; then
   echo -e "Usage:\n\t$0 deviceId \n"
   exit 1;
fi

_URL="http://api.allshoppings.mobi/app/testTicket?authToken=B8C7BC1F60DD840345975E847F24D89A3817930E6B20A42F6B063E8BF251B327"
_DATA="{"deviceUUID":"$1"}"

echo ""
echo "Calling:" ${_URL}
echo ""

_RES=`curl -k -X POST -H "Content-Type:application/json" -d "${_DATA}" ${_URL}`
echo ""
echo "RESULT: $_RES"
```

Después de guardarlo como script y darle permisos de ejecución, lo invocan de la siguiente manera:

```
./TestTicket c254507c3690abd9
```

De esta manera, estarían creando una promoción que se asocie al teléfono cuyo ID es c254507c3690abd9.

Esto va a ser de mucho uso para generar promociones en varios telefonos y simular el estímulo que enviará el server de AllShoppings cuando las condiciones trigger se cumplan.