Your changes have been saved in version 9.

Last modified 0 seconds ago

AllShoppings Android API

La integración a la app, como se aprecia en el proyecto de ejemplo, es sencilla.
Un requisito previo antes de iniciar con la integración, es necesariamente obtener un GCM ID para el proyecto, para que sea identificable por el servidor de AllShoppings y le permita enviar un mensaje push. Una documentación rápida y sencilla al respecto se encuentra en:
http://dev.tapjoy.com/faq/how-to-find-sender-id-and-api-key-for-gcm/

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. Por ejemplo:

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

Así mismo, es conveniente agregar un parámetro String 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.

El archivo AndroidManifest.xml de la aplicación a construir debe contener lo siguiente (reemplazando {package} por el nombre de paquete correspondiente a la app):

```
    <permission
       android:name="{package}.permission.C2D_MESSAGE"
       android:protectionLevel="signature" />

  <uses-permission android:name="{package}.permission.C2D_MESSAGE" />

  <uses-permission android:name="android.permission.INTERNET" />
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
  <uses-permission android:name="android.permission.WAKE_LOCK" />
  <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
  <uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
  <uses-permission android:name="android.permission.BROADCAST_STICKY" />
  <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
  <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
  <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
  <uses-permission android:name="android.permission.READ_PHONE_STATE" />

  <application

      ...
      <!-- AllShoppings API -->
      <activity android:name="mobi.allshoppings.gcm.PushHandlerActivity" />

      <receiver
          android:name="mobi.allshoppings.gcm.ASGCMBroadcastReceiver"
          android:permission="com.google.android.c2dm.permission.SEND" >
          <intent-filter>
              <action android:name="com.google.android.c2dm.intent.RECEIVE" />
              <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
              <category android:name="mobi.allshoppings.integrationapisample" />
          </intent-filter>
      </receiver>

      <service android:name="mobi.allshoppings.gcm.GCMIntentService" />

      <receiver
          android:name="mobi.allshoppings.api.ASOnBootReceiver"
          android:enabled="true"
          android:exported="false" >
          <intent-filter>
              <action android:name="android.intent.action.BOOT_COMPLETED" />
          </intent-filter>
      </receiver>

      <service
          android:name="mobi.allshoppings.api.ASGeoLocationService"
          android:icon="@drawable/ic_launcher"
          android:label="@string/app_name" />
```

```
    <receiver
        android:name="mobi.allshoppings.api.ASGeofenceTransitionReceiver"
        android:exported="false" >
        <intent-filter>
            <action android:name="mobi.allshoppings.api.ACTION_RECEIVE_GEOFENCE" />
        </intent-filter>
    </receiver>

    <meta-data
        android:name="com.google.android.gms.version"
        android:value="@integer/google_play_services_version" />
</application>
```

Para inicializar el API, basta con hacer los siguientes llamados a la biblioteca desde el onCreate() del Activity principal:

```
AllShoppingsIntegrationAPI api = AllShoppingsIntegrationAPI.getInstance(this);

Map<String, Object> properties = new HashMap<String, Object>();
properties.put(AllShoppingsIntegrationAPI.CONTEXT, this);
properties.put(AllShoppingsIntegrationAPI.GCM_SENDER_ID, "818571021175");do p
properties.put(AllShoppingsIntegrationAPI.APP_TOKEN, "B8C7BC1F60DD840345975E847F24D89A3817930E6B20A42F6B063E8BF251B327");
properties.put(AllShoppingsIntegrationAPI.APP_ID, "televisa_mx");
properties.put(AllShoppingsIntegrationAPI.SERVICE_URL, "http://api.allshoppings.mobi");
properties.put(AllShoppingsIntegrationAPI.APP_VERSION, "10000");
properties.put(AllShoppingsIntegrationAPI.PUSH_MESSAGE_OBSERVER, new PushMessageObserver() {
    @Override
    public void processMessage(Context context, Bundle extras) {
        AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
        builder.setMessage(extras.toString())
               .setTitle("Message Received");
        AlertDialog dialog = builder.create();
        dialog.show();
    }

    @Override
    public boolean overrideDefault() {
        return false;
    }

    @Override
    public void onMessage(Context context, Bundle extras) {
        Log.d(TAG, "onMessage");
    }
});

ASUser user = new ASUser();
user.setEmail("1234@televisa");
user.setFirstname("Matias");
user.setLastname("Hapanowicz");
user.setGender("M");
user.setBirthdate("1981-10-13");
properties.put(AllShoppingsIntegrationAPI.USER, user);

api.initializeWithProperties(properties);
api.startup();
```

Entonces para resumir la integración:

- Generar el GCMSenderID según el método expuesto en la documentación adjunta
- Enlazar el jar del API al proyecto de la aplicación custom.
- Modificar el AndroidManifest.xml según lo expuesto anteriormente.
- Inicializar el API con un Map que le provea los atributos según el ejemplo,
- Ejecutar el método 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.