P6 : Compléments

1. Téléphonie
2. Lecture audio et vidéo
3. Afficher du contenu web (html)
4. Les API de géolocalisation et Google Map
5. Notification push

1 – Téléphonie

Avec l'aide du Telephony Manager, vous pouvez obtenir de nombreuses informations comme l'identifiant unique (IMEI ou MEID), le numéro de série de la carte SIM, le nom de l'opérateur téléphonique ...

```
TelephonyManager telephonyManager = (TelephonyManager)
        getSystemService(Context.TELEPHONY_SERVICE);

telephonyManager.getDeviceId();
telephonyManager.getPhoneType();
telephonyManager.getSimOperatorName();
telephonyManager.getNetworkType();
```

1 – Téléphonie

Il est également possible de lancer un appel téléphonique ou d'envoyer un SMS.

```
// lancer un appel
Intent intent = new Intent (Intent.ACTION_CALL);
intent.setData(Uri.parse("tel://0628323814"));
startActivity(intent);

// envoyer un SMS (deprecié)
SmsManager smsManager = SmsManager.getDefault();
smsManager.sendTextMessage("0628323814", null, "votre message", null, null);

// envoyer un SMS
Intent sendIntent = new Intent(Intent.ACTION_VIEW);
sendIntent.putExtra("sms_body", "default content");
sendIntent.setType("vnd.android-dir/mms-sms");
startActivity(sendIntent);

// ajouter la permission dans le manifest
// <uses-permission android:name="android.permission.SEND_SMS" />
```

2 – Lecture audio et vidéo

Android comprend un Media Player complet pour simplifier la lecture de fichier audio ou vidéo.

Différents formats sont supportés en fonction des versions Android (developer.android.com/guide/appendix/media-formats.html).

Lecture audio :

```
// Create a new media player and set the listeners
mMediaPlayer = new MediaPlayer();
mMediaPlayer.setDataSource(path);

mMediaPlayer.prepare();
mMediaPlayer.setOnBufferingUpdateListener(this);
mMediaPlayer.setOnCompletionListener(this);
mMediaPlayer.setOnPreparedListener(this);

mMediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
```

2 – Lecture audio et vidéo

Lecture vidéo :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <VideoView
        android:id="@+id/videoView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
    </VideoView>

</LinearLayout>
```

```
Uri videoPath = Uri.fromFile(new File("/sdcard/video.mp4"));

VideoView videoView = (VideoView) findViewById(R.id.videoView);
videoView.setKeepScreenOn(true);
videoView.setVideoURI(videoPath);
videoView.start();

MediaController mc = new MediaController(this);
videoView.setMediaController(mc);
```

Exemple en streaming : http://devimages.apple.com/iphone/samples/bipbop/bipbopall.m3u8

3 – Afficher du contenu web (html)

Le composant WebView permet d'afficher du code HTML / CSS / JavaScript .

La permission INTERNET est nécessaire.

```
<uses-permission android:name="android.permission.INTERNET" />
```

Depuis Android 4.4 (API 19), la webview est basée sur Chromium et supporte les derniers standard HTML 5.

Attention : Si votre application supporte une ancienne version d'Android, votre code HTML doit être rétro compatible afin qu'il s'affiche correctement sur WebKit et Chromium.

3 – Afficher du contenu web (html)

Afficher du code html à partir du composant webview.

Layout :

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <WebView
        android:id="@+id/webView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

Java :

```
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.loadUrl("http://www.example.com");
```

3 – Afficher du contenu web (html)

De nombreux paramètres peuvent être ajoutés à la webview.

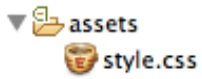
```
webView = (WebView) findViewById(R.id.webview);

webView.getSettings().setDefaultFontSize(webView.getSettings().getDefaultFontSize() + 2);
webView.getSettings().setGeolocationEnabled(true);
webView.getSettings().setSupportZoom(true);
```

Il est aussi possible de récupérer les paramètres de la webview.

```
webView.getSettings().getDefaultFontSize();
webView.getSettings().getUserAgentString();
webView.getSettings().getJavaScriptEnabled();
```

3 – Afficher du contenu web (html)

Il est possible d'inclure un style CSS stocké dans le projet dans le répertoire « assets ».



Java :

```
String html = "<html>" +
        "<head>" +
        "<link href=\"file:///android_asset/style.css\" rel=\"stylesheet\"
        "</head>" +
        "<body>Hello World.</body>" +
        "</html>";

myWebView.loadData(html, "text/html", null);
```

3 – Afficher du contenu web (html)

Il est également possible de charger une url à partir du dossier « assets » ou à partir d'une url « http ».

Code html :

```
<html>
<head>
    <title>Titre de la page</title>

</head>
<body>

Contenu HTML

</body>
</html>
```

Java :

```
webView = (WebView) mActivity.findViewById(R.id.webView);

webView.loadUrl("file:///android_asset/page.html");

webView.loadUrl("http://www.google.fr");
```

3 – Afficher du contenu web (html)

Les liens peuvent être interceptés pour modifier le comportement du client de la Webview.

Code html :

```
<html>
<head><title>Titre de la page</title></head>
<body>

<a href="liens.html">Mon lien</a>

</body>
</html>
```

Java :

```
private class MyWebViewClient extends WebViewClient {
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        if (url.startsWith("liens.html")) {
            return false;
        }
        return true;
    }
}

webView.setWebViewClient(new MyWebViewClient());
```

3 – Afficher du contenu web (html)

Les interfaces JavaScript permettent d'exécuter du code natif à partir d'une webview.

```
<html>
<head><title>Titre de la page</title></head>
<body>

<a href="javascript:NATIVE.showToast('message de test');">Affiche un Toast</a>

</body>
</html>
```

Java :

```
webView = (WebView) mActivity.findViewById(R.id.webView);

webView.getSettings().setJavaScriptEnabled(true);

JavascriptInterfaceNative mJavaScriptInterfaceNative = new JavascriptInterfaceNative(mActivity);
webView.addJavascriptInterface(mJavaScriptInterfaceNative, "NATIVE");

if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT) {
    WebView.setWebContentsDebuggingEnabled(true);
}

webView.loadUrl("file:///android_asset/page.html");
```

3 – Afficher du contenu web (html)

La méthode « showToast » est définie dans la classe « JavaScriptInterfaceNative ».

L'annotation « @JavascriptInterface » permet de rendre accessible la méthode publique « showToast ».

```
class JavascriptInterfaceNative {

    Activity c;

    public JavascriptInterfaceNative(Activity c) {
        this.c = c;
    }

    @JavascriptInterface
    public String showToast(String toast) {

        Toast.makeText(c, toast, Toast.LENGTH_SHORT).show();

        return toast;
    }
}
```

3 – Afficher du contenu web (html)

Il est également possible d'exécuter des méthodes JavaScript à partir du code Java.

```
<script type="text/javascript">
    function redirection() {
        window.location.href='http://www.google.fr';
    }
</script>
```

Java :

```
@JavascriptInterface
public boolean showToast(String toast) {

    Toast.makeText(c, toast, Toast.LENGTH_SHORT).show();

    c.runOnUiThread(new Runnable() {

        @Override
        public void run() {

            webView.loadUrl("javascript:redirection();");
        }
    });

    return true;
}
```

4 – Les API de géolocalisation et Google Map

Selon le modèle du téléphone, il est possible d'utiliser les fonctionnalités d'android pour pouvoir effectuer une localisation sous plusieurs technologies :

- Position Satellite GPS
- Triangulation selon l'emplacement des bornes gsm
- Positionnement selon l'emplacement des bornes wifi

Ces différents provider peuvent aussi renvoyer des renseignements complémentaires (metrics) : latitude, la vitesse de déplacement, latitude/longitude.

Des permissions devront être ajoutées au manifest.

4 – Les API de géolocalisation et Google Map

Exemple :

```
public class MyLocationManager implements LocationListener {

    @Override
    public void onLocationChanged(Location location) {
        if(location != null) {
            double lat = location.getLatitude();
            double lon = location.getLongitude();
        }

    }
```

```
<!-- Réseau mobile / Wifi -->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

<!-- GPS -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

4 – Les API de géolocalisation et Google Map

Exemple :

```
LocationManager locationManager;
String svcName = Context.LOCATION_SERVICE;
locationManager = (LocationManager)getSystemService(svcName);

Criteria criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_FINE);
criteria.setPowerRequirement(Criteria.POWER_LOW);;
criteria.setAltitudeRequired(false);
criteria.setBearingRequired(false);
criteria.setSpeedRequired(false);
criteria.setCostAllowed(true);

String provider1 = locationManager.getBestProvider(criteria, true);
Location l = locationManager.getLastKnownLocation(provider1);

updateWithNewLocation(l);

locationManager.requestLocationUpdates(provider1, 2000, 10, locationListener);
```

4 – Les API de géolocalisation et Google Map

```
private void updateWithNewLocation(Location l) {
    TextView myLocationText;
    myLocationText = (TextView)findViewById(R.id.myLocation);

    String latLongString = "Aucun Emplacement trouvé";
    String adresseString = "Aucune adresse trouvée";

    if(l != null)
    {
        double lat = l.getLatitude();
        double lng = l.getLongitude();
        double alt = l.getAltitude();

        latLongString = "Latitude:" + lat + "\nLongitude:" +lng + "\nAltitude:" + alt;

        Geocoder geocoder = new Geocoder(this, Locale.getDefault());

        try{
            List<Address> adresses = geocoder.getFromLocation(lat, lng, 1);
            StringBuilder stringBuilder = new StringBuilder();
            if(adresses.size() > 0)
            {
                Address adresse = adresses.get(0);

                for(int i=0; i<adresse.getMaxAddressLineIndex(); ++i)
                    stringBuilder.append(adresse.getAddressLine(i)).append("\n");

                stringBuilder.append(adresse.getLocality()).append("\n");
                stringBuilder.append(adresse.getPostalCode()).append("\n");
                stringBuilder.append(adresse.getCountryName());
            }
            adresseString = stringBuilder.toString();
        }catch(IOException e){}
    }
    myLocationText.setText("Position :\n" + latLongString + "\nAdresse :\n" + adresseString);
}
```

4 – Les API de géolocalisation et Google Map

L'API Google Map permet au travers d'une MapView de représenter des données géographiques.

Pour utiliser une MapView, il faut ajouter les API Google et obtenir une clé d'API sur le site développeur Android.

La procédure complète est décrite sur ce site :
https://developers.google.com/maps/documentation/android/

4 – Les API de géolocalisation et Google Map

1) Installer la librairie « Google Play Service »
2) Ajouter au projet les librairie « Google » et « Play Service »
3) Modifier le manifest :

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

<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="API_KEY" />
```

4) Créer un projet sur le site de Google « Google API Console »
5) Récupérer la clé SH1 et «API_KEY» depuis le site « Google »
6) Ajouter la clé dans le manifest
7) Ajouter les permissions et feature au manifest
8) Ajouter une carte sur un layout

4 – Les API de géolocalisation et Google Map

MapView :

```
<fragment
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.MapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

```
// Récupération de la carte
GoogleMap map = ((MapFragment) getFragmentManager().findFragmentById(R.id.map)).getMap();

// affichage de la position de l'utilisateur
map.setMyLocationEnabled(true);

// déplacement vers la position du monument Notre Dame de Paris
map.moveCamera(CameraUpdateFactory.newLatLngZoom(new LatLng(48.853, 2.35), 13));

// ajout d'un marqueur
map.addMarker(new MarkerOptions()
.title("Paris, Notre Dame")
.snippet("La cathédrale Notre-Dame de Paris."));
```

5 – Push GCM

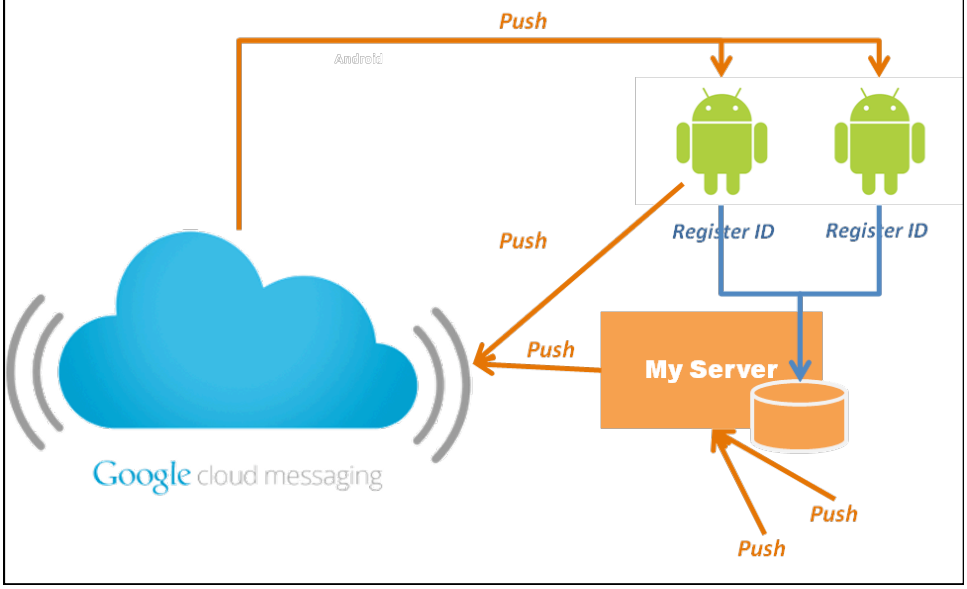
Les notifications Push:

- « Google Cloud Messaging » (GCM)
- Anciennement « Cloud to Device Messaging » (C2DM)

Les Notifications Push sont architecturées autour de 3 points :

- le serveur de notification GCM (Google)
- le serveur applicatif d'où seront envoyés les messages (notre serveur).
- le périphérique Android qui contient l'application (client)

5 – Push GCM



5 – Push GCM

1) Créer un compte Google puis créer un « API Project » et activer le « Cloud Messaging » pour Android.
   https://console.developers.google.com
1) Obtenir un identifiant « Project ID » (client) et la clé « API Key » (notre serveur).
2) Ajouter les librairie « Google Cloud Messaging for Android Library » (Android SDK Manager > Extras).

Google Cloud Messaging for Android Library 3 Installed

3) Utiliser la cible « Google API »

Android 4.4.2 Android Open Source Project
Google APIs Google Inc.

4) Associer le projet « Google Play Service lib »

../google-play-services_lib google-play-services_lib

5 – Push GCM

Les permissions nécessaires :

```
<!-- Permissions specifiques à notre application -->
<permission
    android:name="com.example.messagepush.permission.C2D_MESSAGE"
    android:protectionLevel="signature" />

<uses-permission android:name="com.example.messagepush.permission.C2D_MESSAGE" />

<!-- Wake lock -->
<uses-permission android:name="android.permission.WAKE_LOCK" />

<!-- App receives GCM messages. -->
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />

<!-- GCM connects to Google Services. -->
<uses-permission android:name="android.permission.INTERNET" />

<!-- GCM requires a Google account. -->
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
```

5 – Push GCM

Dans la partie « applications » du manifest :

Ajouter le broadcastReceiver qui gère l'intent envoyé par « GCM » et le service appelé depuis GCMBroadcastReceiver.

```
<receiver
    android:name="com.sfr.crowdservice.GCMBroadcastReceiver"
    android:permission="com.google.android.c2dm.permission.SEND" >
    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.RECEIVE" />

        <category android:name="com.sfr.crowdservice" />
    </intent-filter>
</receiver>

<service android:name="com.sfr.crowdservice.GCMIntentService" />
```