PDM Avanzado Xeolocalización

Sumario

Introdución

A xeolocalización e un servizo que nos proporciona Android e vai permitir saber cal é a nosa posición en forma de coordenadas (latitude e lonxitude).

Temos diferentes formas de obter a localización:

- Localización por GPS.
- A través das antenas de telefonía móbil.
- Mediante puntos de acceso WIFI cercanos.

Cada unha destas formas leva consigo consumo de baterías e precesión diferentes.

Veremos máis adiante que podemos solicitar información ó noso dispositivo para que nos indique cal é o provedor que se adapta a algún criterio indicado por nos (como precisión alta, por exemplo).

Envío de coordenadas GPS ó dispositivo

Na práctica que imos desenvolver nesta unidade imos ter que enviarlle ó dispositivo información GPS falsa para simular que nos estamos movendo.

Para facelo, teremos que ter en conta o tipo de dispositivo:

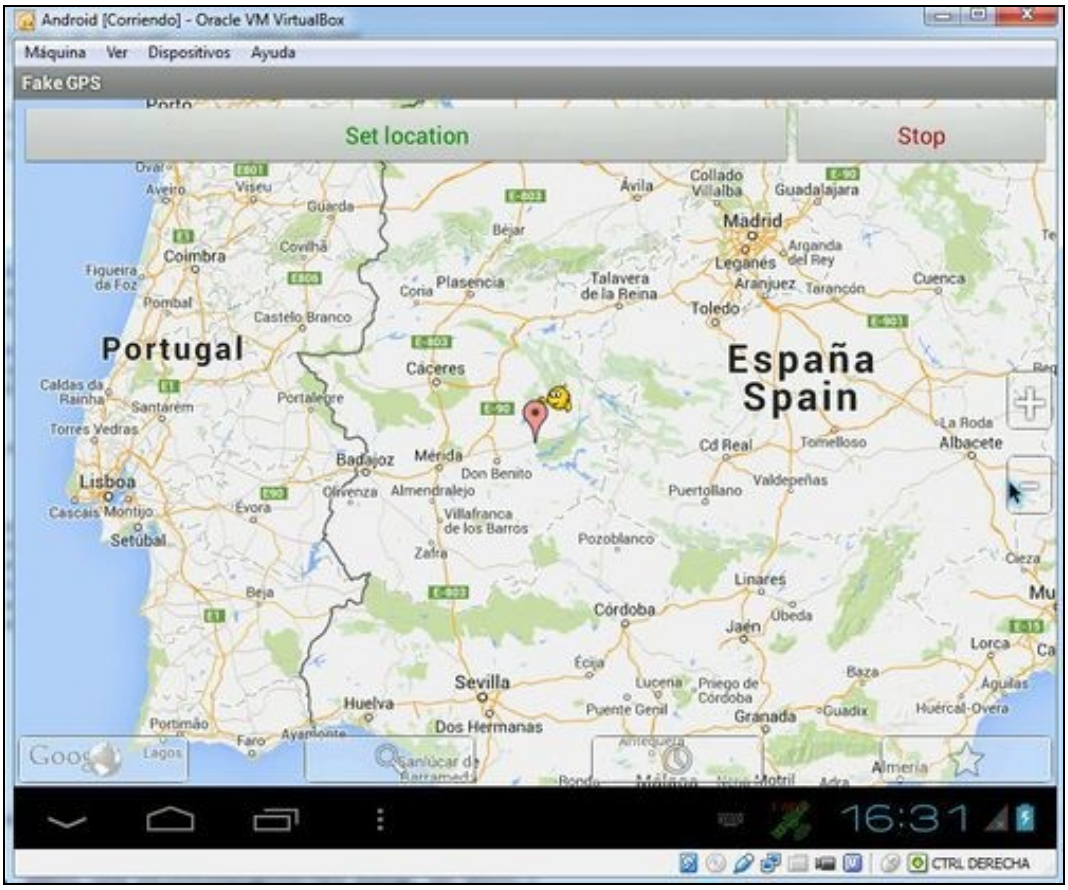
Dispositivo real ou máquina de VirtuaBox

- Se o dispositivo é real ou estades a utilizar a máquina virtual suministrada (neste caso o software xa está instalado):

  ◊ Deberedes instalar a aplicación **Fake GPS**: https://play.google.com/store/apps/details?id=com.lexa.fakegps&hl=es

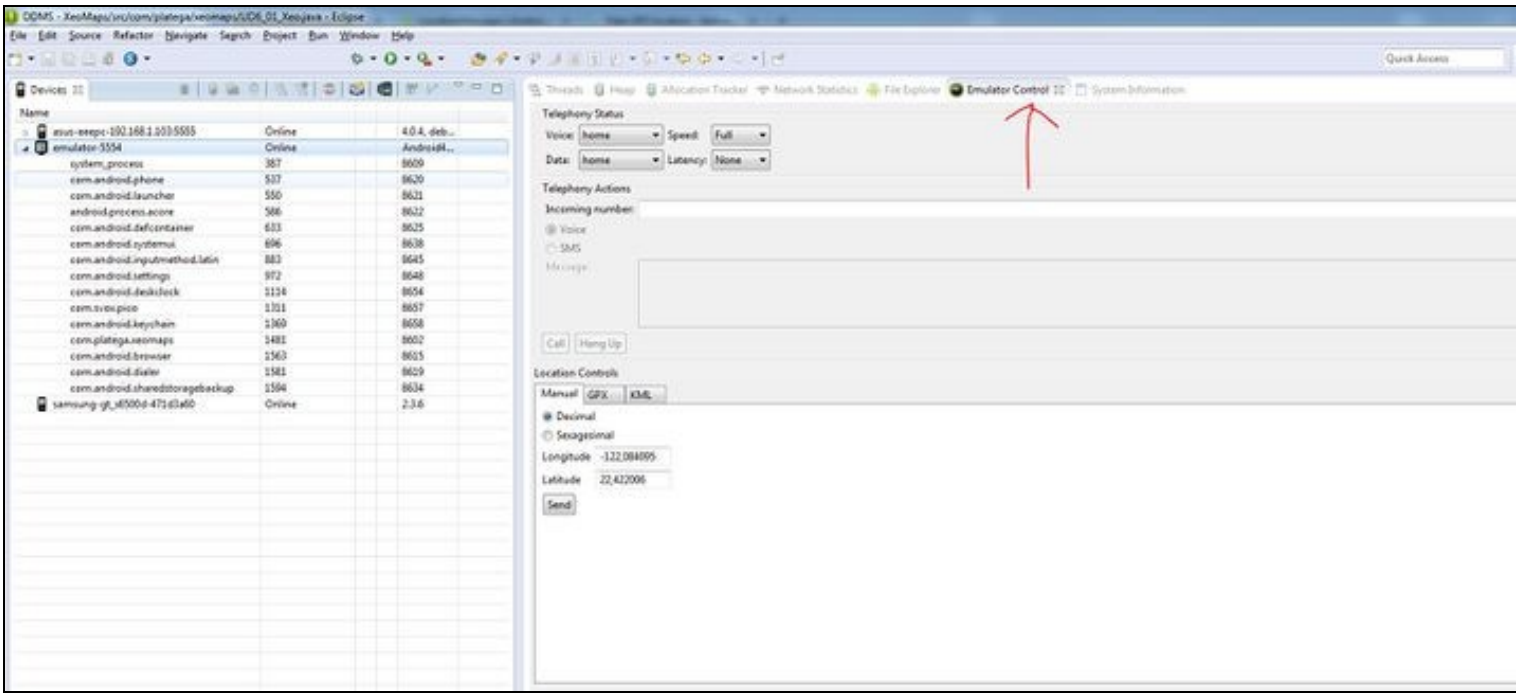
  O funcionamento desta aplicación é moi sinxelo.

  - Primeiro deberemos abrir dita aplicación, antes de executar a aplicación que queiramos enganar.
  - Unha vez aberta debemos mover o mapa (co rato, arrastrando) e cando esteamos situados no punto que nos interesa premer o botón **Set Location**.
  - Ó facelo a aplicación se minimiza e aparece unha icona (parte inferior dereita). Podemos volver a abrila dende alí.
  - Se queremos saír da aplicación debemos premer o botón de **Stop**.



Emulador

- Se estades a utilizar o emulador, deberedes ir á vista DDMS e premer sobre a lapela **Emulator Control**:



Nesta pantalla podedes escribir as coordenadas GPS e enviarllas ó dispositivo premendo o botón de 'Send'. Existen arquivos GPX e KML que se poden xerar ou descargar de Internet e son un conxunto de coordenadas GPS que se envían cada certo tempo de forma automática.

Xeolocalización

Permisos nececesarios

O primeiro de todo será engadir ó arquivo AndroidManifiest.xml os permisos necesarios.

Estes son:

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

Formas de localización: Provedores

Os provedores de localización (Location Providers) vannos servir para obter a información sobre a localización do dispositivo.

Os máis coñecidos son os de GPS e localización mediante telefonía, pero podemos ter outros.

A clase que imos utilizar para obter os provedores é a clase LocaltionManager.

Para obtela teremos que poñer:

```
LocationManager locManager = (LocationManager)getSystemService(LOCATION_SERVICE);
```

e para obter os provedores:

```
List<String> listaProviders = locManager.getAllProviders();
```

Isto devolve todos os nomes (String) dos provedores do dispositivo. Temos que ter en conta que nesta lista aparecerán provedores ós que poda que non teñamos acceso (por non ter os permisos postos no AndroidManifiest) e provedores que pode que non estean activados.

Unha vez temos os nomes podemos obter as características específicas de cada un.

Lembrar que comentamos antes que cada provedor pode ter características de precisión, consumo de enerxía.

```
LocationProvider locProv = locManager.getProvider(LocationManager.GPS_PROVIDER);
```

Neste caso estamos a utilizar unha constante definida na clase LocationManager que ten de valor 'gps'. Esta cadea sería unha das que teríamos na lista de provedores do paso anterior.

Agora sobre dito provedor podemos:

- Obter a súa precisión: Método getAccuracy():

  Devolve un valor que pode ser comparado con:

  ◊ Criteria.ACCURACY_FINE: Precisión alta.
  ◊ Criteria.ACCURACY_COARSE: Precisión media.
  ◊ Criteria.ACCURACY_LOW: Precisión baixa.

- Saber se se pode obter a altitude: Método supportsAltitude(): Devolve un boolean.
- Saber se se pode obter información da velocidade: Método supportsSpeed(): Devolve un boolean.

.... Nota: Veremos máis adiante que a información da velocidade e altitude as podemos obter do obxecto pertencente á clase Location.

- Obter o seu nivel de consumo de batería: Método getPowerRequirement():

  Devolve un valor que pode ser comparado con:

  ◊ Criteria.POWER_HIGH: Alto consumo de enerxía.
  ◊ Criteria.POWER_MEDIUM: Medio consumo de enerxía.
  ◊ Criteria.POWER_LOW: Baixo consumo de enerxía.

Obter o mellor provedor: Criterios de selección

Máis información en: http://developer.android.com/reference/android/location/Criteria.html

Existe una forma de, en base a unha serie de criterios (consumo, precisión...), obter un provedor que se adapte a eles.

Por exemplo:

```
Criteria filtro = new Criteria();
filtro.setAccuracy(Criteria.ACCURACY_FINE);
filtro.setAltitudeRequired(true);
```

Agora podemos chamar ós métodos que nos van devolver os nomes dos provedores que cumpran os criterios anteriores:

```
String provConFiltro= locManager.getBestProvider(filtro, false);
```

ou ben:

```
List<String> listaProvConFiltro = locManager.getProviders(filtro, false);
```

Nota: A segunda forma pode devolver máis dun.

O segundo parámetro é un boolean que indica se queremos que teña en conta aqueles provedores que se atopan desactivados.

E por que vou querer obter un provedor desactivado ?

Porque vou a informar ó usuario desa circunstancia e darlle a opción de activalo (por exemplo o uso do GPS).

Como activar un provedor

Unha vez que sabemos cal provedor queremos utilizar teremos que saber se dito provedor (partimos que base que enviamos false como segundo parámetro ó método getProvider, como vimos antes) está activo e informar ó usuario de que o active.

Para isto dispoñemos do método isProviderEnabled(String provedor).

Algúns dos nomes que podemos utilizar (se queremos buscar por un provedor concreto):

- LocationManager.NETWORK_PROVIDER. Localización pola rede de telefonía.
- LocationManager.GPS_PROVIDER. Localización por GPS.

Por exemplo:

```
if (!locManager.isProviderEnabled(LocationManager.GPS_PROVIDER)){
    Toast.makeText(getApplicationContext(), "O GPS non está activo", Toast.LENGTH_LONG).show();
}
```

No caso de non estar activo podemos dar ó usuario a opción de activalo chamando á activity:

```
startActivity(new Intent(android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS));
```

Obtendo a localización: Interface LocationListener

Subscrición a un provedor

En Android,a forma de obter a localización a través dun provedor escollido no paso anterior, consiste en subscribirse ás notificacións de cambio de posición.

Unha vez subscrito automaticamente chamará a certo método en base a unha serie de criterios establecidos previamente (cambio de posición, cada certo tempo,...)

Non sabemos cando teremos a primeira información xa que dependeremos dos criterios elixidos e do tempo que tarde o dispositivo en empezar a recibir datos (por exemplo, cando activamos o GPS ten que esperar a localizar as sinais dos satélites,...)

Unha forma de obter unha localización inicial é chamando ó método getLastKnownLocation(String provider) o cal devolve a última localización rexistrada do provedor indicado:

```
Location location = locManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
```

Este método non devolve a posición actual, non solicita unha nova posición ó provedor de localización.

Este método se limita a devolver a última posición que se obtivo a través do provedor que se lle indica como parámetro.

Temos que ter coidado con dito método xa que pode devolver null se nunca obtivemos unha localización có provedor indicado.

Para subscribirnos ó servizo de notificacións do provedor debemos facer uso do método requestLocationUpdates() da clase LocationManager.

Este método está sobrecargado e temos varias posibilidades. Nos imos ver unha delas.

- public void requestLocationUpdates (String provider, long minTime, float minDistance, LocationListener listener)

  Este método leva 4 parámetros:

  - ◊ String provider: Nome do provedor.
  - ◊ long minTime: Tempo en mseg que pasará ata enviar novos datos de localización. Se se pon 0 non terá en conta este criterio. O fai aproximadamente (non é exacto).
  - ◊ float minDistance: Distancia en metros que ten haber dende a última posición enviada. Se se pon 0 non terá en conta este criterio. O fai aproximadamente (non é exacto).
  - ◊ LocationListener listener: Clase que implementa dita interface. Dentro desta interface teremos diferentes métodos.

Xa vimos no curso de iniciación as diferentes formas de implementar unha interface.

Por exemplo:

```
LocationManager locManager = (LocationManager)getSystemService(LOCATION_SERVICE);
locManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,0,100,this);
```

Neste caso estamos a subscribirnos ó provedor GPS para que envío notificacións cada vez que nos movemos máis de 100 metros con respecto a unha localización anterior.

A activity onde se atopa o código anterior ten que implementar a interface LocationListener.

Interface LocationListener

Tal como comentamos antes, cando chamamos ó método requestLocationUpdates debemos indicar a clase que implementa a interface LocationListener.

Ó facelo, obriga a interface a implementar os seguintes métodos:

- onLocationChanged(location). Chamado cada vez que se recibe unha actualización da posición.
- onProviderDisabled(provider). Chamado cando o provedor se deshabilita.
- onProviderEnabled(provider). Chamado cando el provedor se habilita.
- onStatusChanged(provider, status, extras). Chamado cada vez que o provedor cambia o seu estado, que pode variar entre OUT_OF_SERVICE, TEMPORARILY_UNAVAILABLE,AVAILABLE.

Exemplo de código:

```
@Override
public void onLocationChanged(Location location) {
// TODO Auto-generated method stub
// location.getLatitude(): obtemos a nova latitude
                // location.getLongitude(): obtemos a nova lonxitude
}


@Override
public void onProviderDisabled(String provider) {
// TODO Auto-generated method stub
Toast.makeText(getApplicationContext(), "O provedor " + provider + " xa non está activo", Toast.LENGTH_LONG).show();
}


@Override
public void onProviderEnabled(String provider) {
// TODO Auto-generated method stub
Toast.makeText(getApplicationContext(), "O provedor " + provider + " está activo", Toast.LENGTH_LONG).show();

}


@Override
public void onStatusChanged(String provider, int status, Bundle extras) {
// TODO Auto-generated method stub
Toast.makeText(getApplicationContext(), "O provedor " + provider + " cambió  de estado:" + String.valueOf(status), Toast.LENGTH_LONG)

}
```

Dentro deste código cabe sinalar o método:

- public void onLocationChanged(Location location)

Como comentamos, o obxecto da clase Location vai ter a información sobre a nosa localización.

Eliminado a subscrición a un provedor

Para parar de 'escoitar' notificacións dun provedor debemos chamar ó método removeUpdates(String provedor).

Por exemplo:

```
locManager.removeUpdates(LocationManager.GPS_PROVIDER);
```

Loxicamente teremos que facer isto antes de saír da aplicación para liberar os recursos.

Caso práctico

Nesta práctica imos a recoller nunha lista os cambios de posición GPS cando nos movamos máis de 300 metros.

Teremos un botón para empezar a recoller datos (subscribirnos) e outro botón para o contrario (eliminar subscrición).

Tal como comentamos antes:

- Teremos que ter instalar e executada a aplicación 'Fake GPS' ou ben utilizar un emulador.
- Teremos que engadir no arquivo AndroidManifiest.xml os permisos necesarios.

Creamos a activity

- Nome do proxecto: **UD6_01_Xeo**
- Nome da activity: **UD6_01_Xeo.java**

Código do layout xml

```
<RelativeLayout 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"
    tools:context="${relativePackage}.${activityClass}" >

    <Button
        android:id="@+id/UD6_01_btnRexistrarGPS"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="REXISTRAR GPS" />

    <Button
        android:id="@+id/UD6_01_btnPararRexistro"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/UD6_01_btnRexistrarGPS"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="16dp"
        android:text="PARAR REXISTRO GPS" />

        <ListView
        android:id="@+id/UD6_01_lstListaCoordGPS"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@android:color/darker_gray"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="10dp"
        android:layout_below="@+id/UD6_01_btnPararRexistro" >

    </ListView>

</RelativeLayout>
```

Código da clase UD6_01_Xeo

Obxectivo: Subscribirse ó provedor GPS e rexistrar as coordenadas GPS.

```
import java.util.ArrayList;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.location.Criteria;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.view.View;
```

```
import android.view.View.OnClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.Toast;

public class UD6_01_Xeo extends Activity implements LocationListener{

private ArrayList<String> localizacions;
private ArrayAdapter<String> adaptador;

private LocationManager locManager;
private String provedor;

private void obterprovedores(){
Criteria filtro = new Criteria();
filtro.setAccuracy(Criteria.ACCURACY_FINE);

locManager = (LocationManager)getSystemService(LOCATION_SERVICE);

provedor = locManager.getBestProvider(filtro, false);// Se non está activo o avisamos e chamamos a activity para activalo
//              provedor = LocationManager.NETWORK_PROVIDER;    => Exemplo concreto sen filtro

if (provedor==null){
Toast.makeText(getApplicationContext(), "Non existen provedores dispoñibles.", Toast.LENGTH_LONG).show();
finish();
}
if (!locManager.isProviderEnabled(provedor)){
Toast.makeText(getApplicationContext(), "O " + provedor + " non está activo", Toast.LENGTH_LONG).show();
dialogoAlertaNonGPS();
}
else{
Toast.makeText(getApplicationContext(), "provedor atopado:" + provedor, Toast.LENGTH_LONG).show();
}
}

private void dialogoAlertaNonGPS() {
final AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setMessage(
"O GPS parece desactivado, queres activalo ?")
.setCancelable(false)
.setPositiveButton("Si",
new DialogInterface.OnClickListener() {
@Override
public void onClick(DialogInterface arg0, int arg1) {
// TODO Auto-generated method stub
startActivity(new Intent(
android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS));

}
})
.setNegativeButton("Non", new DialogInterface.OnClickListener() {

@Override
public void onClick(DialogInterface dialog, int which) {
// TODO Auto-generated method stub
dialog.cancel();
UD6_01_Xeo.this.finish();

}
});
final AlertDialog alert = builder.create();
alert.show();
}

private void xestionarEventos(){

Button btnRexistrar = (Button)findViewById(R.id.UD6_01_btnRexistrarGPS);
btnRexistrar.setOnClickListener(new OnClickListener() {

@Override
public void onClick(View arg0) {
// TODO Auto-generated method stub
```

```
locManager.requestLocationUpdates(provedor,0,100,UD6_01_Xeo.this);
Toast.makeText(getApplicationContext(), "Comenzado a rexistrar...", Toast.LENGTH_SHORT).show();

Location last =locManager.getLastKnownLocation(provedor);
if (last!=null)
localizacions.add("ULTIMA COÑECIDA: LAT:" + String.valueOf(last.getLatitude()) + " - LONX:" + String.valueOf(last.getLongitude()));

}
});

Button btnPararRexistrar = (Button)findViewById(R.id.UD6_01_btnPararRexistro);
btnPararRexistrar.setOnClickListener(new OnClickListener() {

@Override
public void onClick(View v) {
// TODO Auto-generated method stub
locManager.removeUpdates(UD6_01_Xeo.this);
Toast.makeText(getApplicationContext(), "Parando de rexistrar...", Toast.LENGTH_SHORT).show();
}
});

}


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ud6_01__xeo);

        localizacions = new ArrayList<String>();
        adaptador = new ArrayAdapter<String>(getApplicationContext(), android.R.layout.simple_list_item_1, localizacions);
        ListView listaGPS = (ListView)findViewById(R.id.UD6_01_lstListaCoordGPS);
        listaGPS.setAdapter(adaptador);

        obterprovedores();
        xestionarEventos();
    }


@Override
public void onLocationChanged(Location location) {
// TODO Auto-generated method stub
localizacions.add("LATITUDE:" + String.valueOf(location.getLatitude()) + " - LONXITUDE:" + String.valueOf(location.getLongitude()));
adaptador.notifyDataSetChanged();
}


@Override
public void onProviderDisabled(String provider) {
// TODO Auto-generated method stub
Toast.makeText(getApplicationContext(), "O provedor " + provider + " xa non está activo", Toast.LENGTH_LONG).show();
}


@Override
public void onProviderEnabled(String provider) {
// TODO Auto-generated method stub
Toast.makeText(getApplicationContext(), "O provedor " + provider + " está activo", Toast.LENGTH_LONG).show();

}


@Override
public void onStatusChanged(String provider, int status, Bundle extras) {
// TODO Auto-generated method stub

}
}
```

- Liña 21: Array onde gardamos a lista de coordenadas GPS e que se van cargar na ListView.
- Liña 22: Adaptador asociado ó ListView.
- Liña 25: Nome do provedor a utilizar.

- Liñas 27-47: Obtemos un provedor en base ó criterio de precisión alta. Gardamos o nome do provedor atopado la propiedade **provedor**.
- Liñas 49-76: Caixa de diálogo que aparece en caso de que o provedor estea inactivo.
- Liña 86: Subscribimos ó provedor ó facer click no botón.
- Liña 102: Eliminamos a subscrición do provedor ó facer click no botón.
- Liña 128: Engadimos á lista de localizacións a nova localización.
- Liña 129: Notificamos ó adaptador que hai un cambio nos datos para que a lista se recargue.

-- Ángel D. Fernández González e Carlos Carrión Álvarez -- (2014).