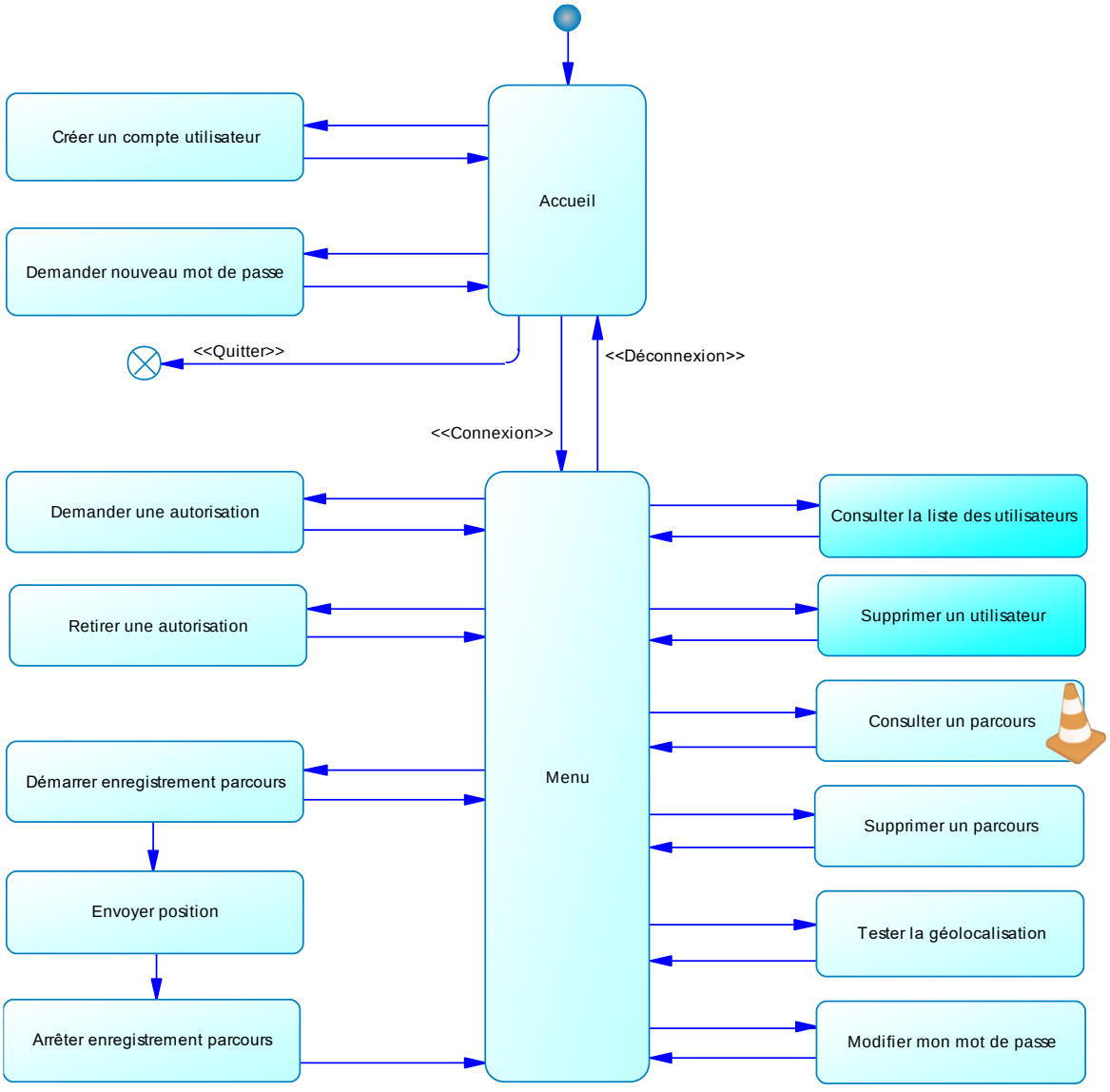
5- Développement de l'application mobile Android

5-10 ConsulterParcours (consulter un parcours)

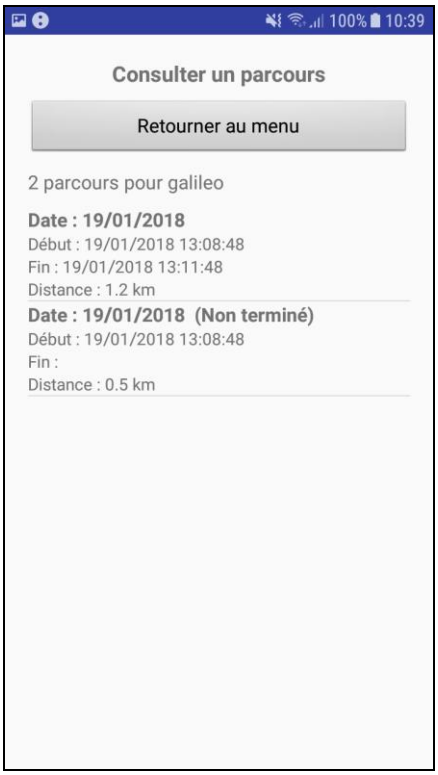
1- Situation de l'activité dans la structure de l'application
2- Modification du fichier strings.xml
3- Création d'une activité Google Maps
5- Modification de la programmation Java de MenuGeneral.java
6- Création de l'interface graphique
7- Programmation Java de l'activité ConsulterParcours.java

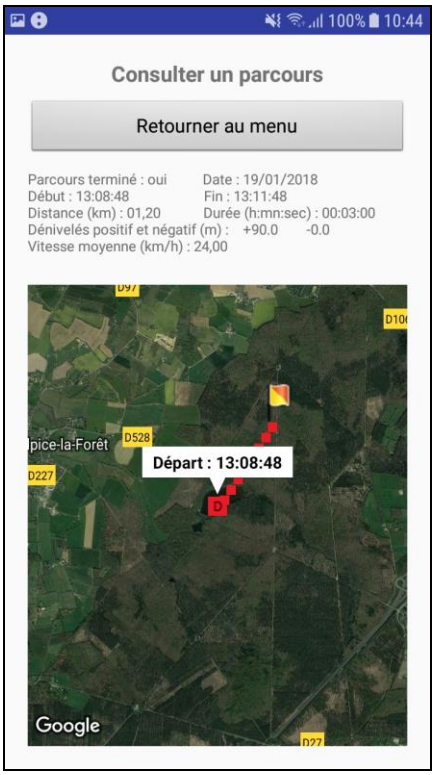
1- Situation de l'activité dans la structure de l'application



2- Modification du fichier strings.xml

L'interface graphique à créer :



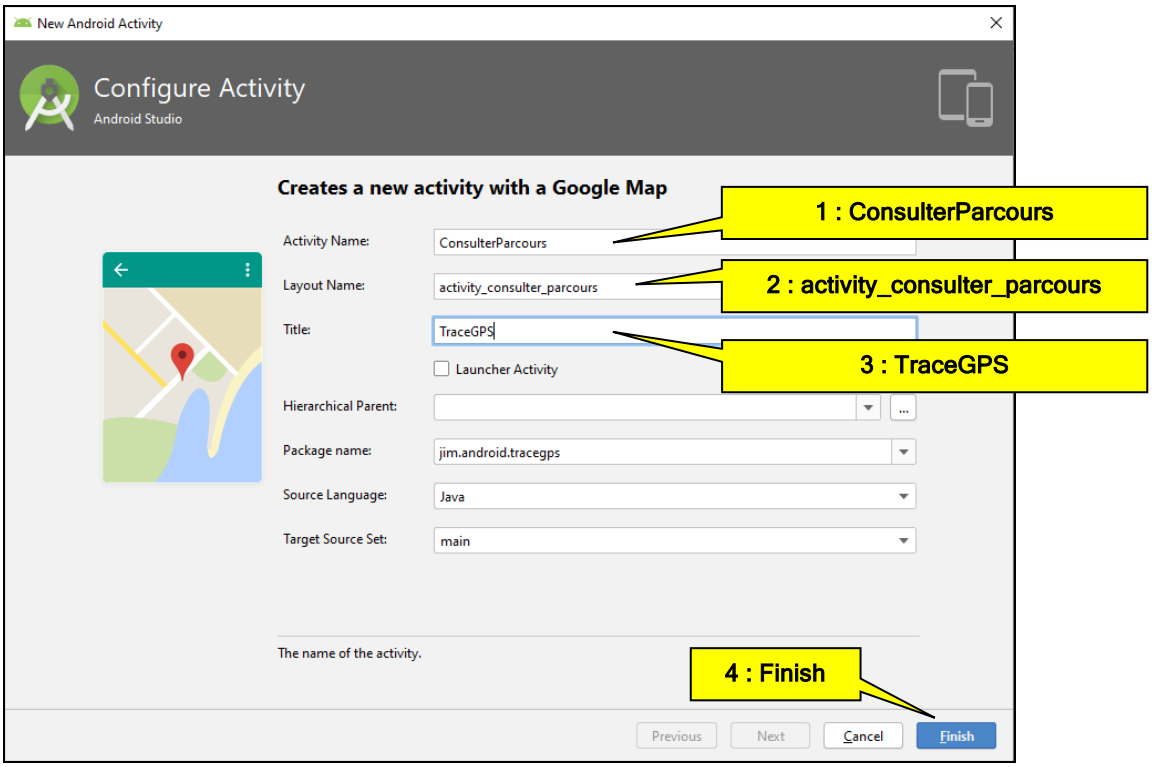


Dans le dossier **res/values**, complétez le fichier **strings.xml** avec le code suivant :

```
<!-- Les textes de la page de consultation de parcours -->
<string name="consulter_parcours_titre1">Consulter un parcours</string>
<string name="consulter_parcours_bouton_retourner">Retourner au menu</string>
```

3- Création d'une activité Google Maps

Créer une nouvelle activité en faisant un clic droit sur la racine **app** du projet et en choisissant la commande **New / Google / Google Maps Activity** :



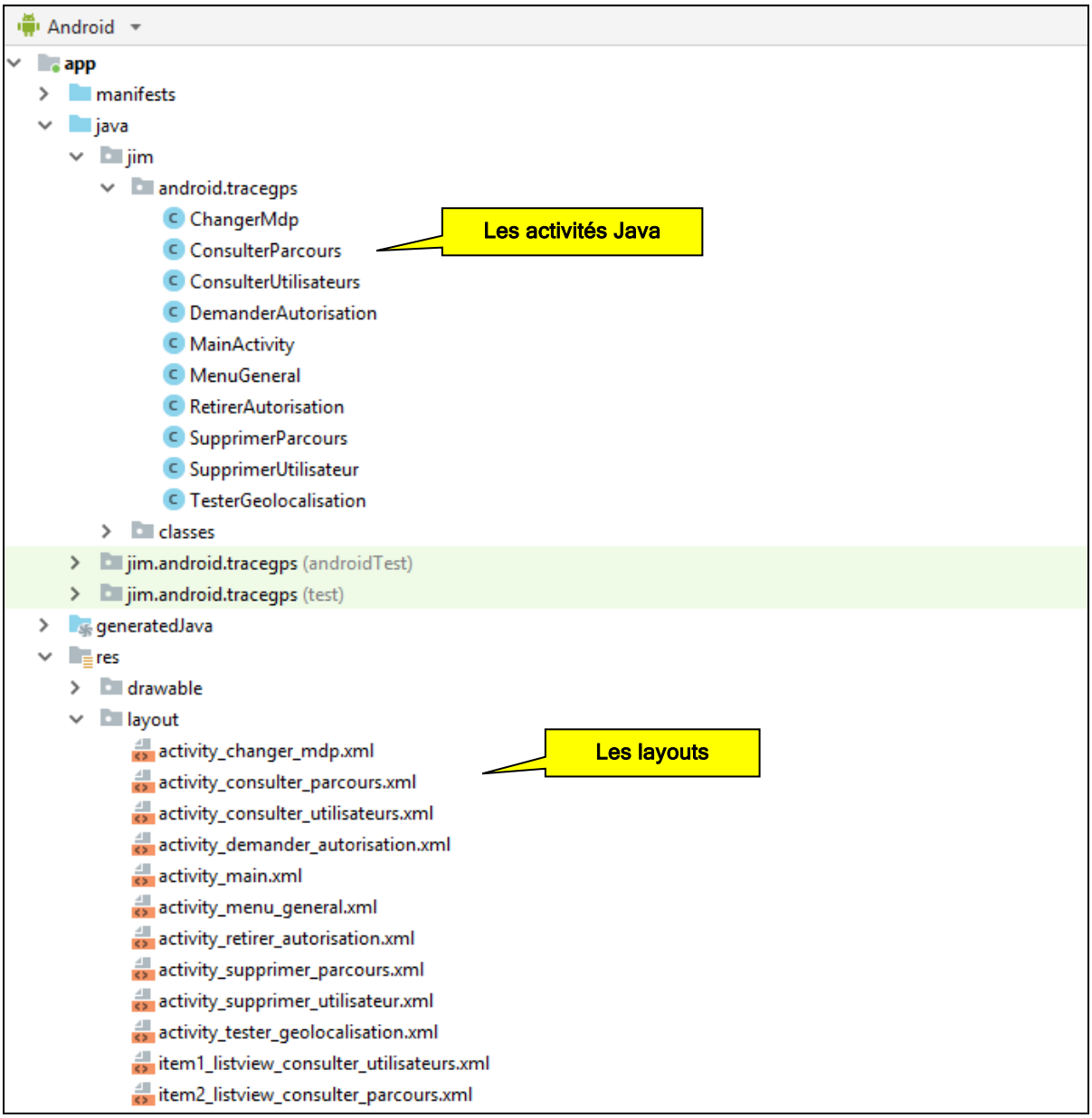
Une fois l'activité créee, Android Studio ouvre les fichiers **ConsulterParcours.java** et **google_maps_api.xml**.

Le fichier **res/values/google_maps_api.xml** contient des instructions sur le moyen d'obtenir une clé **d'API Google Maps** afin de pouvoir exécuter l'application (**en mode Debug**) :

```
<resources>
    <!--
    TODO: Before you run your application, you need a Google Maps API key.

    To get one, follow this link, follow the directions and press "Create" at the end:
    https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID
    &r=7C:F1:B8:3D:B0:45:ED:84:55:4A:F0:3C:AC:AD:24:5A:E5:C2:3D:8C%3Bjim.android.tracegps

    You can also add your credentials to an existing key, using these values:

    Package name:
    7C:F1:B8:3D:B0:45:ED:84:55:4A:F0:3C:AC:AD:24:5A:E5:C2:3D:8C

    SHA-1 certificate fingerprint:
    7C:F1:B8:3D:B0:45:ED:84:55:4A:F0:3C:AC:AD:24:5A:E5:C2:3D:8C

    Alternatively, follow the directions here:
    https://developers.google.com/maps/documentation/android/start#get-key

    Once you have your key (it starts with "AIza"), replace the "google_maps_key"
    string in this file.
    -->
    <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">
        AIzaSyAsci4xqPSalNrMMKmwPRIEU9qd2axTBv8
    </string>
</resources>
```

La clé API est connue car elle a déjà été indiquée lors du développement de l'activité TesterGeolocalisation

L'activité **ConsulterParcours.java** et le layout **activity_consulter_parcours.xml** sont alors créés :



5- Modification de la programmation Java de MenuGeneral.java

Dans l'activité **MenuGeneral.java**, complétez l'écouteur d'événement associé au bouton **buttonConsulterParcours** :

```
/** classe interne pour gérer le clic sur le bouton buttonConsulterParcours. */
private class buttonConsulterParcoursClickListener implements View.OnClickListener {
    public void onClick(View v) {
        // crée une Intent pour lancer l'activité
        Intent uneIntent = new Intent(MenuGeneral.this, ConsulterParcours.class);
        // passe nom, mdp et typeUtilisateur à l'Intent
        uneIntent.putExtra(EXTRA_PSEUDO, pseudo);
        uneIntent.putExtra(EXTRA_MDP, mdp);
        uneIntent.putExtra(EXTRA_TYPE_UTILISATEUR, typeUtilisateur);
        // démarre l'activité à partir de l'Intent
        startActivity(uneIntent);
    }
}
```

Testez cette étape sur un mobile réel et corrigez les erreurs si besoin.

Le bouton **Consulter un parcours** doit activer l'activité **ConsulterParcours** ; vous obtiendrez une carte centrée sur la ville de **Sydney** en Australie (Google n'a pas choisi la ville de Rennes...) :

6- Création de l'interface graphique

6-1 L'interface de base proposée par Android Studio

Le fichier **res/layout / activity_consulter_parcours.xml** contient la description de l'interface graphique :

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="jim.android.tracegps.ConsulterParcours" />
```

Il comporte un **<fragment>** occupant tout l'espace et dont l'identifiant est **map**.

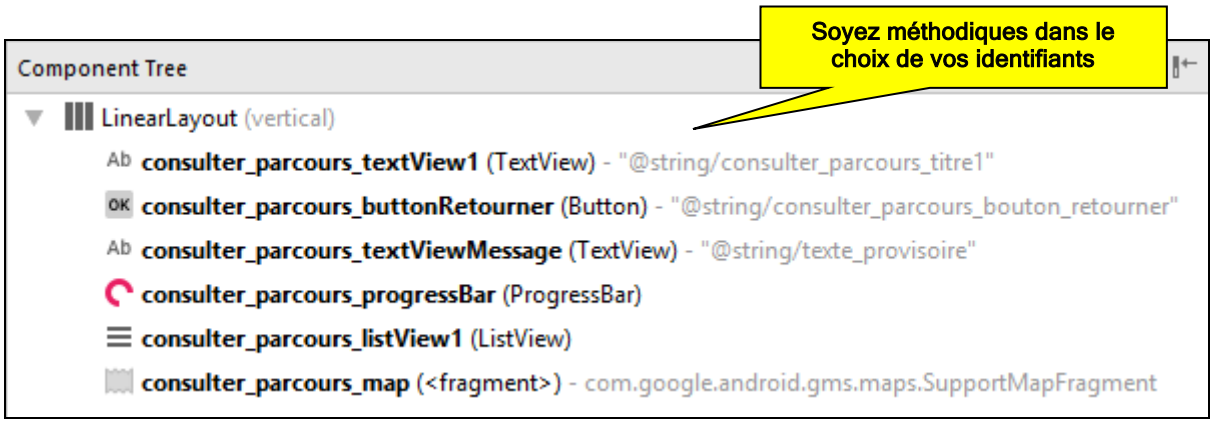
6-2 Modification de l'interface graphique

Il faut commencer par ajouter un **LinearLayout vertical** qui contiendra le bouton et le **Fragment**.
Cette modification ne peut pas se faire en mode **Design**. Passez en mode **Text** et modifiez le code :

```
<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"
android:orientation="vertical"
android:padding="@dimen/tailleMarges"
tools:context="jim.android.tracegps.ConsulterParcours">

    <fragment xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/consulter_parcours_map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

</LinearLayout>
```

Modifier l'id du fragment

Revenez maintenant en mode **Design** et placez les différents composants en suivant la structure suivante et en utilisant bien sûr les chaines du fichier **strings.xml** :



Le code XML du 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"
    android:orientation="vertical"
    android:padding="@dimen/tailleMarges"
    tools:context="jim.android.tracegps.ConsulterParcours">

    <TextView
        android:id="@+id/consulter_parcours_textView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="10dp"
        android:text="@string/consulter_parcours_titre1"
        android:textAlignment="center"
        android:textSize="18sp"
        android:textStyle="bold" />

    <Button
        android:id="@+id/consulter_parcours_buttonRetourner"
        style="@android:style/Widget.Button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/consulter_parcours_bouton_retourner"
        android:textSize="16sp" />

    <TextView
        android:id="@+id/consulter_parcours_textViewMessage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="10dp"
        android:paddingTop="10dp"
        android:text="@string/texte_provisoire"
        android:textSize="16sp" />

    <ProgressBar
        android:id="@+id/consulter_parcours_progressBar"
        style="?android:attr/progressBarStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal" />

    <ListView
        android:id="@+id/consulter_parcours_listView1"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <fragment xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/consulter_parcours_map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

7- Programmation Java de l'activité ConsulterParcours.java

7-1 Description du code Java initial

Le code initial du fichier **MapsActivity.java** est le suivant :

```
package jim.android.tracegps;

import android.support.v4.app.FragmentActivity;
import android.os.Bundle;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

public class ConsulterParcours extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap mMap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_consulter_parcours);
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

    /**
     * Manipulates the map once available.
     * This callback is triggered when the map is ready to be used.
     * This is where we can add markers or lines, add listeners or move the camera. In this case,
     * we just add a marker near Sydney, Australia.
     * If Google Play services is not installed on the device, the user will be prompted to install
     * it inside the SupportMapFragment. This method will only be triggered once the user has
     * installed Google Play services and returned to the app.
     */
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        // Add a marker in Sydney and move the camera
        LatLng sydney = new LatLng(-34, 151);
        mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney"));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
    }
}
```

L'objet GoogleMap (qui sera renommé)

onCreate s'exécute lors de la création de l'activité et appelle le service Google Maps

L'identifiant du fragment

L'appel du service Google Maps en mode asynchrone (correspond à l'appel d'une AsyncTask)

onMapReady s'exécute dès que la carte est prête (correspond au onPostExecute d'une AsyncTask)

Remarquez que le **traitement asynchrone des tâches** est déjà intégré dans **l'API Google Maps**, et évite au développeur d'applications de mettre en oeuvre des objets **AsyncTask** :

- l'instruction **mapFragment.getMapAsync(this)** est équivalente au déclenchement d'une tâche asynchrone
- l'événement **onMapReady** est équivalent à l'événement **onPostExecute** de la classe **AsyncTask**

7-2 Correction du code Java initial

Dans le code initialement généré par Android Studio, l'objet **GoogleMap** représentant la carte est nommé **mMap** ; pour améliorer la lisibilité du code, vous allez le renommer **laCarte.**

Supprimez également les lignes barrées :

```
package jim.android.tracegps;

import android.support.v4.app.FragmentActivity;
import android.os.Bundle;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

public class ConsulterParcours extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap mMap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_consulter_parcours);
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

    /**
     * Manipulates the map once available.
     * This callback is triggered when the map is ready to be used.
     * This is where we can add markers or lines, add listeners or move the camera. In this case,
     * we just add a marker near Sydney, Australia.
     * If Google Play services is not installed on the device, the user will be prompted to install
     * it inside the SupportMapFragment. This method will only be triggered once the user has
     * installed Google Play services and returned to the app.
     */
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        // Add a marker in Sydney and move the camera
        LatLng sydney = new LatLng(-34, 151);
        mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney"));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
    }
}
```

Supprimez les lignes barrées

Changez mMap en laCarte

Changez map en consulter_parcours_map

Changez mMap en laCarte

7-3 Déclarations diverses et initialisation des objets

Dans le fichier **ConsulterParcours.java**, ajoutez le code indiqué en gras :

```
package jim.android.tracegps;

import android.support.v4.app.FragmentActivity;
import android.os.Bundle;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

import android.widget.Button;
import android.widget.TextView;
import android.widget.ListView;
import android.widget.ProgressBar;
import android.widget.AdapterView;
import android.view.View;
import android.content.Intent;

public class ConsulterParcours extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap laCarte;

    // les objets du layout
    private TextView textViewMessage;       // le TextView pour afficher le message
    private Button buttonRetourner;         // le Button pour retourner au menu
    private ProgressBar progressBar;        // le ProgressBar pour afficher le cercle de chargement
    private ListView laListView;            // le ListView pour afficher les parcours

    // le passage des données entre activités se fait au moyen des "extras" qui sont portés par les Intent.
    // un extra est une couple de clé/valeur
    // nous en utiliserons 2 ici, dont voici les 2 clés et les 2 variables associées :
    private final String EXTRA_PSEUDO = "pseudo";
    private final String EXTRA_MDP = "mdp";
    private String pseudo;
    private String mdp;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_consulter_parcours);
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.consulter_parcours_map);
        mapFragment.getMapAsync(this);

        // récupération du nom, et du mot de passe passés par l'activité précédente
        Intent uneIntent = getIntent();
        pseudo = uneIntent.getStringExtra(EXTRA_PSEUDO);
        mdp = uneIntent.getStringExtra(EXTRA_MDP);

        // récupération des objets du layout grâce à leur ID
        textViewMessage = (TextView) findViewById(R.id.consulter_parcours_textViewMessage);
        buttonRetourner = (Button) findViewById(R.id.consulter_parcours_buttonRetourner);
        progressBar = (ProgressBar) findViewById(R.id.consulter_parcours_progressBar);
        laListView = (ListView) findViewById(R.id.consulter_parcours_listView1);

        // arrête le cercle de chargement
        progressBar.setVisibility(View.GONE);

        // association d'un écouteur d'évenement aux boutons
        buttonRetourner.setOnClickListener ( new buttonRetournerClickListener());

        // association d'un écouteur d'évenement à l'événement OnItemClic du ListView
        laListView.setOnItemClickListener ( new laListViewOnItemClickListener());
    }
```

```
    @Override
    public void onMapReady(GoogleMap googleMap) {
        laCarte = googleMap;

    }

    /** classe interne pour gérer le clic sur le bouton buttonRetourner. */
    private class buttonRetournerClickListener implements View.OnClickListener {
        public void onClick(View v) {
            finish();
        }
    }

    /** classe interne pour gérer le clic sur un item du ListView. */
    private class laListViewOnItemClickListener implements AdapterView.OnItemClickListener{
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            // A COMPLETER PLUS TARD
        }
    }

} // fin de l'activité
```

Testez l'application et le bon fonctionnement du bouton **Retourner au menu** :

7-4 Afficher la liste des utilisateurs qui m'autorisent avec une tâche asynchrone

L'affichage des utilisateurs qui m'autorisent nécessite d'appeler le service web :

- **GetLesUtilisateursQuiMautorisent** : pour obtenir la liste des utilisateurs qui m'autorisent à suivre leurs parcours

A la suite des **import** existants, ajoutez les **import** suivants :

```
import jim.classes.*;
import java.util.ArrayList;
import android.os.AsyncTask;
```

A la suite des déclarations existantes, ajoutez la déclaration suivante :

```
private ArrayList<Utilisateur> lesUtilisateursQuiMautorisent;     // les utilisateurs qui m'autorisent
```

A la fin de l'activité, ajoutez la tâche asynchrone **TacheGetLesUtilisateursQuiMautorisent** (vous pouvez vous inspirer du document "***5-4 (0) Projet TraceGPS - Dév appli android - ConsulterUtilisateurs***") :

```
// ----------------------------------------------------------------------------------------------------------------------
// ---------------------- tâche asynchrone pour PasserelleServicesWebXML.getLesUtilisateursQuiMautorisent ----------------------
// ----------------------------------------------------------------------------------------------------------------------

private class TacheGetLesUtilisateursQuiMautorisent extends AsyncTask<ArrayList<Utilisateur>, Void, String> {

    La fonction onPreExecute doit démarrer l'affichage de l'objet progressBar.

    La fonction doInBackground doit appeler le service web GetLesUtilisateursQuiMautorisent en utilisant une
    des méthodes statiques de la classe PasserelleServicesWebXML et en lui passant les paramètres
    nécessaires. Cette méthode devra remplir la collection lesUtilisateursQuiMautorisent.

    La fonction onPostExecute doit arrêter l'affichage de l'objet progressBar et tester le message retourné par le
    service web :

        • Si le message retourné par la méthode commence par le mot "Erreur", il faut afficher dans l'objet
          textViewMessage le message retourné par la méthode

        • Sinon, il faut exécuter la fonction afficherLesUtilisateurs dont le code provisoire est donné plus loin


}
```

A la fin de l'activité, ajoutez la fonction provisoire **afficherLesUtilisateurs** :

```
// afficher la liste des utilisateurs
public void afficherLesUtilisateurs() {
    // on affiche le nombre d'utilisateurs (affichage provisoire)
    textViewMessage.setText(lesUtilisateursQuiMautorisent.size() + " utilisateur(s) m'autoris(ent)");
} // fin de la fonction afficherLesUtilisateurs
```

Complétez la fonction **onCreate** pour appeler la tâche **TacheGetLesUtilisateursQuiMautorisent** :

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_consulter_parcours);
    // Obtain the SupportMapFragment and get notified when the map is ready to be used.
    SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.consulter_parcours_map);
    mapFragment.getMapAsync(this);

    // récupération du nom, et du mot de passe passés par l'activité précédente
    Intent uneIntent = getIntent();
    pseudo = uneIntent.getStringExtra(EXTRA_PSEUDO);
    mdp = uneIntent.getStringExtra(EXTRA_MDP);

    // récupération des objets du layout grâce à leur ID
    textViewMessage = (TextView) findViewById(R.id.consulter_parcours_textViewMessage);
    buttonRetourner = (Button) findViewById(R.id.consulter_parcours_buttonRetourner);
    progressBar = (ProgressBar) findViewById(R.id.consulter_parcours_progressBar);
    laListView = (ListView) findViewById(R.id.consulter_parcours_listView1);

    // arrête le cercle de chargement
    progressBar.setVisibility(View.GONE);

    // association d'un écouteur d'évenement aux boutons
    buttonRetourner.setOnClickListener ( new buttonRetournerClickListener());

    // association d'un écouteur d'évenement à l'événement OnItemClic du ListView
    laListView.setOnItemClickListener ( new laListViewOnItemClickListener());

    Instancier la collection lesUtilisateursQuiMautorisent.

    Lancer l'exécution de la tâche asynchrone TacheGetLesUtilisateursQuiMautorisent.

}
```

Testez l'application ; vous devez obtenir un affichage provisoire similaire à celui-ci :

7-5 Gestion de l'affichage des utilisateurs qui m'autorisent dans la ListView

A la suite des **import** existants, ajoutez l'**import** suivant :

```
import android.widget.ArrayAdapter;
```

A la suite des déclarations existantes, ajoutez la déclaration suivante :

```
private ArrayList<String> listeChaines;      // les libellés à placer dans le ListView
```

Modifiez la fonction définitive **afficherLesUtilisateurs** (vous pouvez vous inspirer du document "***5-6 (2) Projet TraceGPS - Dév appli android - RetirerAutorisation***") :

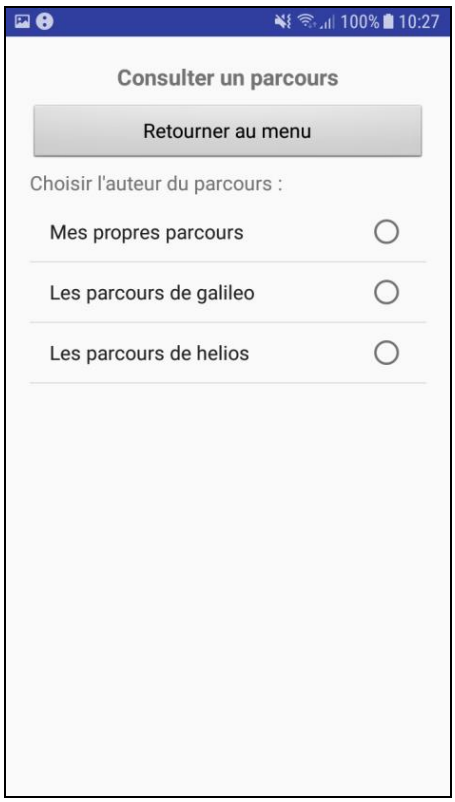
```
// afficher la liste des utilisateurs
public void afficherLesUtilisateurs() {
    textViewMessage.setText("Choisir l'auteur du parcours :");

    // vidage de la liste pour afficher les utilisateurs
    listeChaines = new ArrayList<String>();
    listeChaines.add("Mes propres parcours");
    // parcours de l'ensemble des utilisateurs contenus dans lesUtilisateursQuiMautorisent
    for (int i = 0; i < lesUtilisateursQuiMautorisent.size(); i++)
    {   Utilisateur unUtilisateur = lesUtilisateursQuiMautorisent.get(i);
        // ajout des utilisateurs à la liste
        listeChaines.add("Les parcours de " + unUtilisateur.getPseudo());
    }
```

Créer un ArrayAdapter nommé **monAdapter** avec le contexte, le style des items, et les données à afficher.

Afficher l'objet **laListView** à partir de l'objet **monAdapter**.

```
} // fin de la fonction afficherLesUtilisateurs
```

Testez l'application ; vous devez obtenir un affichage définitif similaire à celui-ci :



7-6 Gestion du clic sur un item du ListView

On va maintenant gérer le clic sur un item du ListView en affichant la liste des parcours de l'utilisateur choisi.

L'affichage des parcours de l'utilisateur choisi nécessite d'appeler le service web :

- **GetLesParcoursDunUtilisateur** : pour obtenir la liste des parcours d'un utilisateur

A la suite des déclarations existantes, ajoutez les déclarations suivantes :

```
private String pseudoUtilisateurChoisi;     // le pseudo de l'utilisateur choisi
private ArrayList<Trace> lesTraces;         // contient la collection des traces de l'utilisateur choisi
```

A la fin de l'activité, ajoutez la tâche asynchrone **TacheGetLesTracesDeLUtilisateur** (vous pouvez vous inspirer du document "***5-4 (0) Projet TraceGPS - Dév appli android - ConsulterUtilisateurs***") :

```
// ------------------------------------------------------------------------------------------------------
// ------------------------- tâche asynchrone pour PasserelleServicesWebXML.getLesParcoursDunUtilisateur ----------------------
// ------------------------------------------------------------------------------------------------------

private class TacheGetLesTracesDeLUtilisateur extends AsyncTask<ArrayList<Trace>, Void, String> {

    La fonction onPreExecute doit démarrer l'affichage de l'objet progressBar.

    La fonction doInBackground doit appeler le service web GetLesParcoursDunUtilisateur en utilisant une des
    méthodes statiques de la classe PasserelleServicesWebXML et en lui passant les paramètres nécessaires.
    Cette méthode devra remplir la collection lesTraces.

    La fonction onPostExecute doit arrêter l'affichage de l'objet progressBar et tester le message retourné par le
    service web :

        • Si le message retourné par la méthode commence par le mot "Erreur", il faut afficher dans l'objet
          textViewMessage le message retourné par la méthode

        • Sinon, il faut exécuter la fonction afficherLesTraces dont le code provisoire est donné plus loin

}
```

A la fin de l'activité, ajoutez la fonction provisoire **afficherLesTraces** :

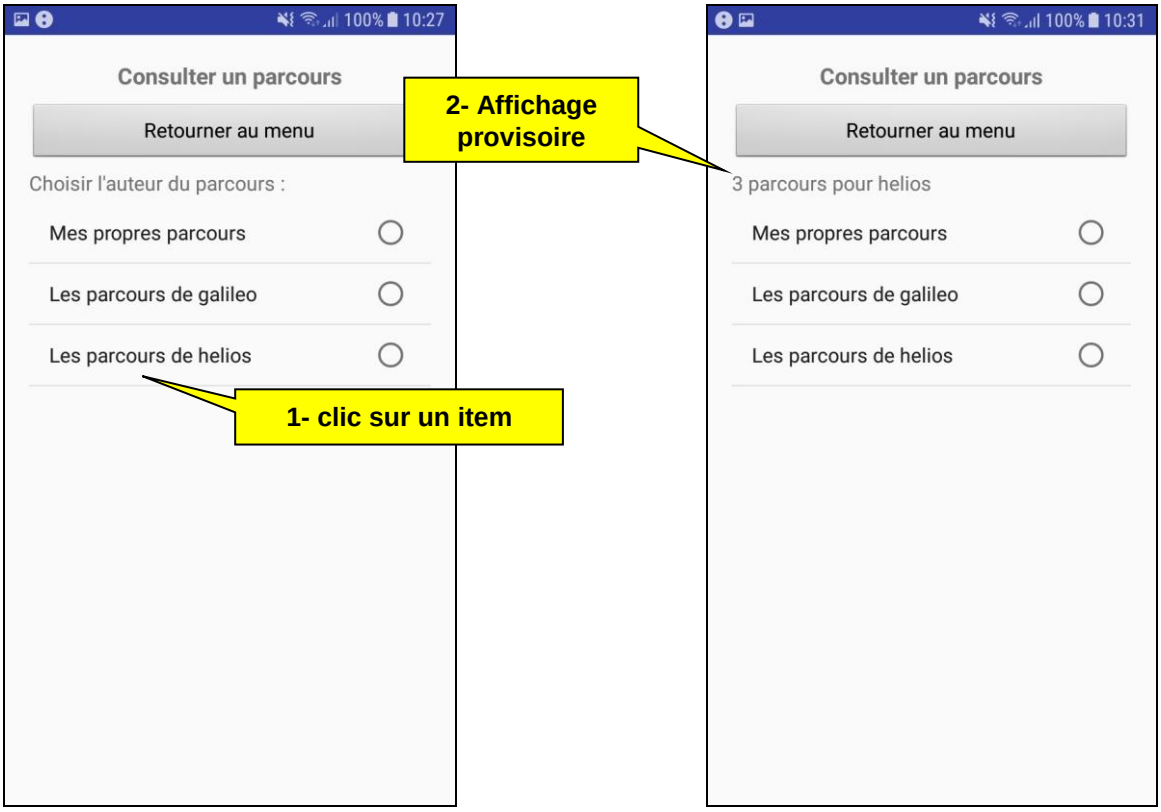
```
// afficher la liste des traces
public void afficherLesTraces() {
    // on affiche le nombre de traces (affichage provisoire)
    textViewMessage.setText(lesTraces.size() + " parcours pour " + pseudoUtilisateurChoisi);
} // fin de la fonction afficherLesTraces
```

Complétez l'écouteur d'événement **laListViewOnItemClickListener** :

```
/** classe interne pour gérer le clic sur un item du ListView. */
private class laListViewOnItemClickListener implements AdapterView.OnItemClickListener{
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        if (textViewMessage.getText().toString().equals("Choisir l'auteur du parcours :")) {
            // recherche de l'utilisateur choisi à partir de la position de l'item choisi
            if (position == 0)
                pseudoUtilisateurChoisi = pseudo;
            else {
                String texte = listeChaines.get(position);
                pseudoUtilisateurChoisi = texte.substring(16);
            }
            // chargement de la liste des utilisateurs à partir du service web à l'aide d'une tâche asynchrone
            lesTraces = new ArrayList<Trace>();
            new TacheGetLesTracesDeLUtilisateur().execute(lesTraces);
        }
        else {
            // recherche du parcours choisi à partir de la position de l'item choisi
            // A COMPLETER PLUS TARD
        }
    }
}
```

position indique le numéro de l'item ayant reçu le clic

Exécutez et testez.



7-7 Gestion de l'affichage des parcours dans la ListView

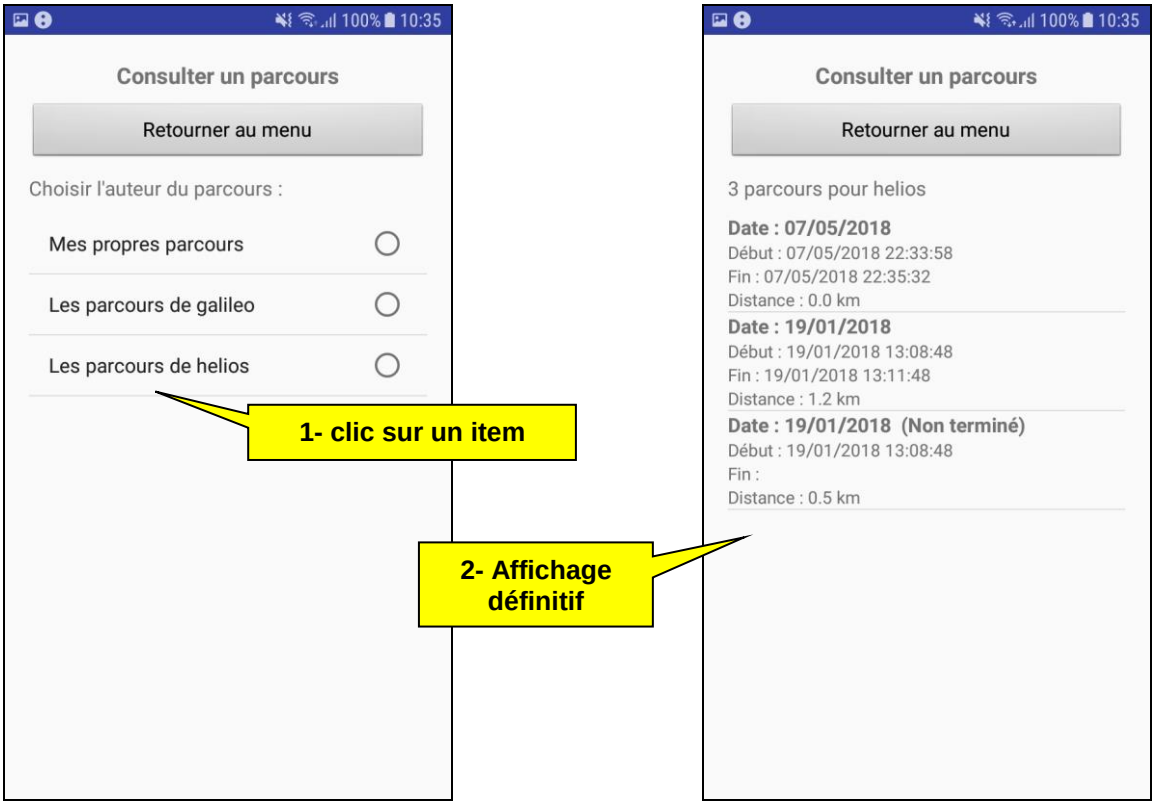
A la suite des **import** existants, ajoutez les **import** suivants :

```
import java.util.HashMap;
import android.widget.SimpleAdapter;
```

Remplacez le contenu de la fonction définitive **afficherLesTraces** (vous pouvez vous inspirer du document "***5-4 (0) Projet TraceGPS - Dév appli android - ConsulterUtilisateurs***") :

```
// afficher la liste des traces
public void afficherLesTraces() {
    // on affiche le nombre de traces
    textViewMessage.setText(lesTraces.size() + " parcours pour " + pseudoUtilisateurChoisi);

    // création de la ArrayList qui permettra de remplir la listView
    ArrayList<HashMap<String, String>> lesElementsDuListView = new ArrayList<HashMap<String, String>>();

    for (int i = 0; i < lesTraces.size(); i++)
    {   Trace uneTrace = lesTraces.get(i);
        // création d'une HashMap pour insérer les informations d'une trace
        HashMap<String, String> element = new HashMap<String, String>();
        if (uneTrace.getTerminee())
            element.put("date", "Date : " + Outils.formaterDate(uneTrace.getDateHeureDebut()));
        else
            element.put("date", "Date : " + Outils.formaterDate(uneTrace.getDateHeureDebut()) + "  (Non terminé)");
        element.put("heure_debut", "Début : " + Outils.formaterDateHeureFR(uneTrace.getDateHeureDebut()));
        if (uneTrace.getTerminee())
            element.put("heure_fin", "Fin : " + Outils.formaterDateHeureFR(uneTrace.getDateHeureFin()));
        else
            element.put("heure_fin", "Fin : ");
        element.put("distance", "Distance : " + uneTrace.getDistanceTotale() + " km");

        // ajoute le HashMap dans le ArrayList
        lesElementsDuListView.add(element);
    }

    Créer un SimpleAdapter pour mettre les items de la liste lesElementsDuListView dans la vue
    item2_listview_consulter_parcours.

    Attribuer au listView laListView le SimpleAdapter monAdapter que l'on vient de créer.

} // fin de la fonction afficherLesTraces
```

Testez l'application ; vous devez obtenir un affichage définitif similaire à celui-ci :



7-8 Gestion du clic sur un item du ListView et affichage du parcours choisi

L'affichage détaillé du parcours choisi nécessite d'appeler le service web :

- **GetUnParcoursEtSesPoints** : pour obtenir le parcours avec l'ensemble de ses points

A la suite des **import** existants, ajoutez l'**import** suivant :

```
import android.util.TypedValue;
```

A la suite des déclarations existantes, ajoutez les déclarations suivantes :

```
private int idParcoursAConsulter;                // l'id du parcours à consulter
private Trace leParcoursAConsulter;              // le parcours à consulter
private ArrayList<PointDeTrace> lesPoints;       // les points du parcours
```

A la fin de l'activité, ajoutez la tâche asynchrone **TacheGetUnParcoursEtSesPoints** (vous pouvez vous inspirer du document "***5-4 (0) Projet TraceGPS - Dév appli android - ConsulterUtilisateurs***") :

```
// ------------------------------------------------------------------------------------------------------
// ------------------------- tâche asynchrone pour PasserelleServicesWebXML.getUnParcoursEtSesPoints ----------
// ------------------------------------------------------------------------------------------------------

private class TacheGetUnParcoursEtSesPoints extends AsyncTask<Trace, Void, String> {

    La fonction onPreExecute doit démarrer l'affichage de l'objet progressBar.

    La fonction doInBackground doit appeler le service web GetUnParcoursEtSesPoints en utilisant une des
    méthodes statiques de la classe PasserelleServicesWebXML et en lui passant les paramètres nécessaires.
    Cette méthode devra remplir l'objet leParcoursAConsulter.

    La fonction onPostExecute doit arrêter l'affichage de l'objet progressBar et tester le message retourné par le
    service web :

        • Si le message retourné par la méthode commence par le mot "Erreur", il faut afficher dans l'objet
          textViewMessage le message retourné par la méthode
        • Sinon, il faut exécuter les 2 fonctions afficherLaTrace et afficherLesMarqueurs dont le code provisoire
          est donné plus loin

}
```

A la fin de l'activité, ajoutez la fonction provisoire **afficherLesMarqueurs** :

```
    // afficher les marqueurs sur la carte
    public void afficherLesMarqueurs() {
        // A COMPLETER
    } // fin de la fonction afficherLesMarqueurs
```

A la fin de l'activité, ajoutez la fonction **afficherLaTrace** :

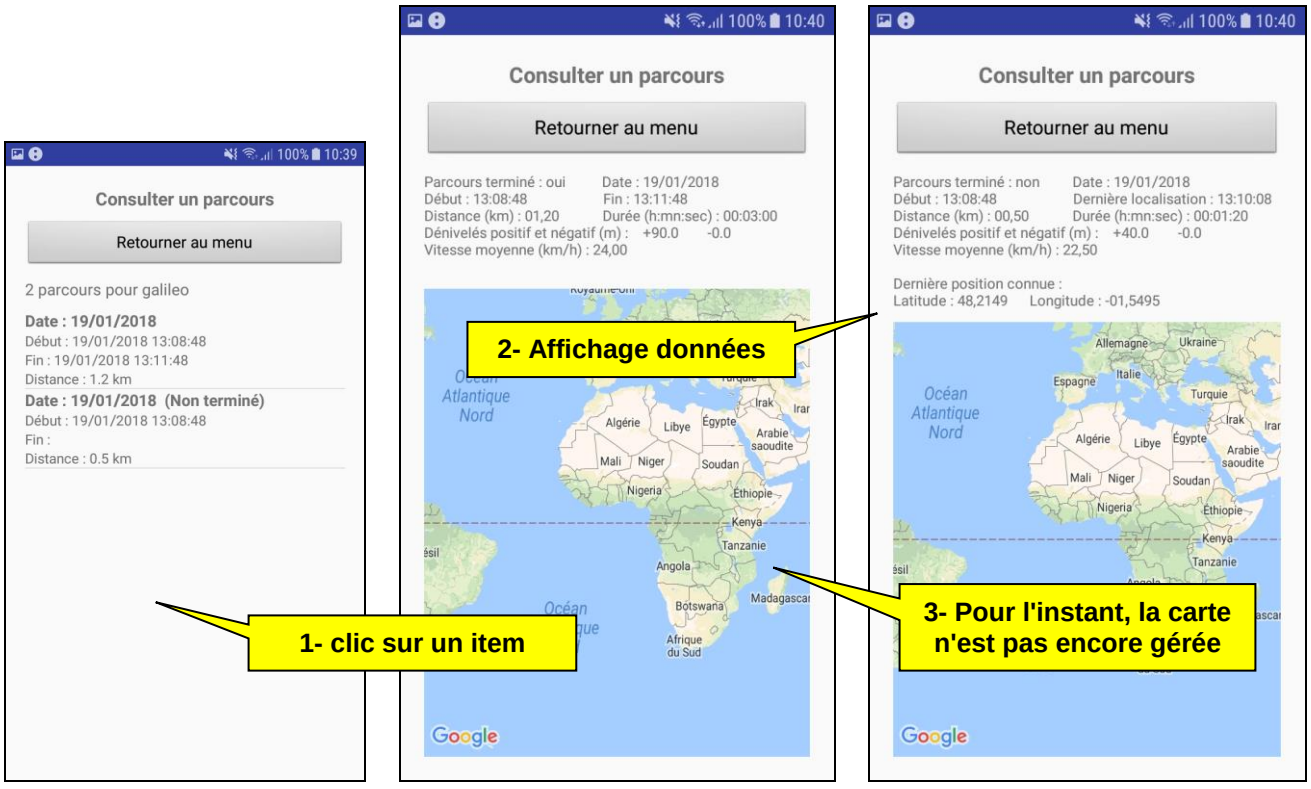
```
// afficher les données de la trace
public void afficherLaTrace() {
    // affichage des données dans le TextView
    String msg = "";
    lesPoints = leParcoursAConsulter.getLesPointsDeTrace();
    PointDeTrace leDernierPoint = null;
    if (lesPoints.size() > 0) {
        leDernierPoint = lesPoints.get(lesPoints.size() - 1);
    }
    if (leParcoursAConsulter.getTerminee()) {
        msg = "Parcours terminé : oui          ";
        msg += "Date : " + Outils.formaterDate(leParcoursAConsulter.getDateHeureDebut()) + "\n";
        msg += "Début : " + Outils.formaterHeureFR(leParcoursAConsulter.getDateHeureDebut()) + "                    ";
        msg += "Fin : " + Outils.formaterHeureFR(leParcoursAConsulter.getDateHeureFin()) + "\n";
        msg += "Distance (km) : " + Outils.formaterNombre(leParcoursAConsulter.getDistanceTotale(), "0.00") + "              ";
        msg += "Durée (h:m:s) : " + leParcoursAConsulter.getDureeTotale() + "\n";
        msg += "Dénivelés positif et négatif (m) :    +" + Outils.formaterNombre(leParcoursAConsulter.getDenivelePositif(), "0.0");
        msg += "       -" + Outils.formaterNombre(leParcoursAConsulter.getDeniveleNegatif(), "0.0")  + "\n";

        msg += "Vitesse moyenne (km/h) : " + Outils.formaterNombre(leParcoursAConsulter.getVitesseMoyenne(), "0.00") + "\n";
    }
    else {
        msg = "Parcours terminé : non         ";
        msg += "Date : " + Outils.formaterDate(leParcoursAConsulter.getDateHeureDebut()) + "\n";
        msg += "Début : " + Outils.formaterHeureFR(leParcoursAConsulter.getDateHeureDebut()) + "                    ";
        if (leDernierPoint != null)
            msg += "Dernière localisation : " + Outils.formaterHeureFR(leDernierPoint.getDateHeure()) + "\n";
        else
            msg += "\n";
        msg += "Distance (km) : " + Outils.formaterNombre(leParcoursAConsulter.getDistanceTotale(), "0.00") + "              ";
        msg += "Durée (h:m:s) : " + leParcoursAConsulter.getDureeTotale() + "\n";
        msg += "Dénivelés positif et négatif (m) :    +" + Outils.formaterNombre(leParcoursAConsulter.getDenivelePositif(), "0.0");
        msg += "       -" + Outils.formaterNombre(leParcoursAConsulter.getDeniveleNegatif(), "0.0")  + "\n";
        msg += "Vitesse moyenne (km/h) : " + Outils.formaterNombre(leParcoursAConsulter.getVitesseMoyenne(), "0.00") + "\n";
        if (leDernierPoint != null) {
            msg += "\nDernière position connue : \n";
            msg += "Latitude : " + Outils.formaterNombre(leDernierPoint.getLatitude(), "0.0000");
            msg += "      Longitude : " + Outils.formaterNombre(leDernierPoint.getLongitude(), "0.0000");
        }
    }
    textViewMessage.setTextSize(TypedValue.COMPLEX_UNIT_SP, 12);
    textViewMessage.setText(msg);
} // fin de la fonction afficherLaTrace
```

Complétez l'écouteur d'événement **laListViewOnItemClickListener** :

```
/** classe interne pour gérer le clic sur un item du ListView. */
private class laListViewOnItemClickListener implements AdapterView.OnItemClickListener{
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        if (textViewMessage.getText().toString().equals("Choisir l'auteur du parcours :")) {
            // recherche de l'utilisateur choisi à partir de la position de l'item choisi
            if (position == 0)
                pseudoUtilisateurChoisi = pseudo;
            else {
                String texte = listeChaines.get(position);
                pseudoUtilisateurChoisi = texte.substring(16);
            }
            // chargement de la liste des utilisateurs à partir du service web à l'aide d'une tâche asynchrone
            lesTraces = new ArrayList<Trace>();
            new TacheGetLesTracesDeLUtilisateur().execute(lesTraces);
        }
        else {
            // recherche du parcours choisi à partir de la position de l'item choisi
            leParcoursAConsulter = lesTraces.get(position);
            idParcoursAConsulter  = leParcoursAConsulter.getId();
            laListView.setVisibility(View.GONE);
            new TacheGetUnParcoursEtSesPoints().execute(leParcoursAConsulter);
        }
    }
}
```

position indique le numéro de l'item ayant reçu le clic

Exécutez et testez (avec un parcours terminé et un parcours non terminé) :



7-9 Affichage du parcours sur la carte

Pour afficher le parcours, 3 images vous sont fournies :


carre_rouge.png

Pour les points intermédiaires

carre_rouge_depart.png

Pour le point de départ


drapeau.png

Pour le point d'arrivée

Avec l'explorateur Windows, recopiez ces 3 fichiers dans le dossier suivant de votre application :

<votre workspace>\TraceGPS\app\src\main\res\drawable

A la suite des **import** existants, ajoutez l'**import** suivant :

```
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
```

Complétez la fonction **afficherLesMarqueurs** :

```
    // afficher les marqueurs sur la carte
    public void afficherLesMarqueurs() {
        // placer la caméra au centre du parcours
        int zoom = 13;     // zoom entre 2 et 21
        LatLng leCentreDuParcours = new LatLng(leParcoursAConsulter.getCentre().getLatitude(),
                leParcoursAConsulter.getCentre().getLongitude());
        laCarte.moveCamera(CameraUpdateFactory.newLatLngZoom(leCentreDuParcours, zoom));
        laCarte.setMapType(GoogleMap.MAP_TYPE_HYBRID);
        // ajoute des marqueurs sur la carte
        laCarte.clear();      // supprime les marqueurs existants

        for (int i = 0 ; i < lesPoints.size() ; i++)
        {   PointDeTrace lePoint = lesPoints.get(i);
            MarkerOptions leMarqueur = new MarkerOptions();
            leMarqueur.position(new LatLng(lePoint.getLatitude(), lePoint.getLongitude()));
            if (i == 0) {
                leMarqueur.icon(BitmapDescriptorFactory.fromResource(R.drawable.carre_rouge_depart));
                leMarqueur.title("Départ : " + Outils.formaterHeureFR(lePoint.getDateHeure()));
            }
            else if (i == lesPoints.size() - 1) {
                leMarqueur.icon(BitmapDescriptorFactory.fromResource(R.drawable.drapeau));
                leMarqueur.title("Arrivée : " + Outils.formaterHeureFR(lePoint.getDateHeure()));
            }
            else {
                leMarqueur.icon(BitmapDescriptorFactory.fromResource(R.drawable.carre_rouge));
                leMarqueur.title(Outils.formaterHeureFR(lePoint.getDateHeure()));
            }
            laCarte.addMarker(leMarqueur);
        }
    } // fin de la fonction afficherLesMarqueurs
```

Exécutez et testez (avec un parcours terminé et un parcours non terminé) :