

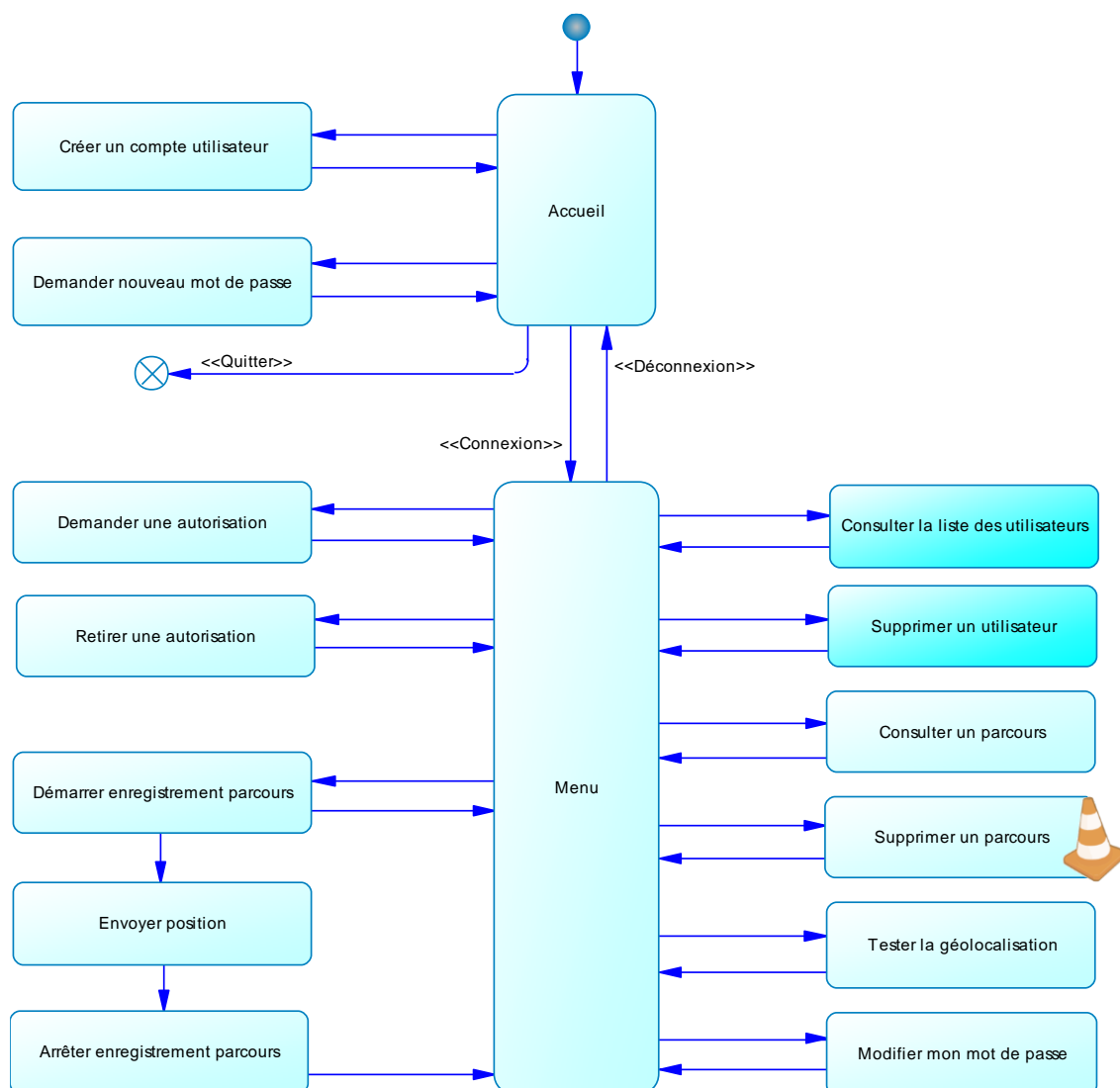


5- Développement de l'application mobile Android

5-8 SupprimerParcours (supprimer un parcours)

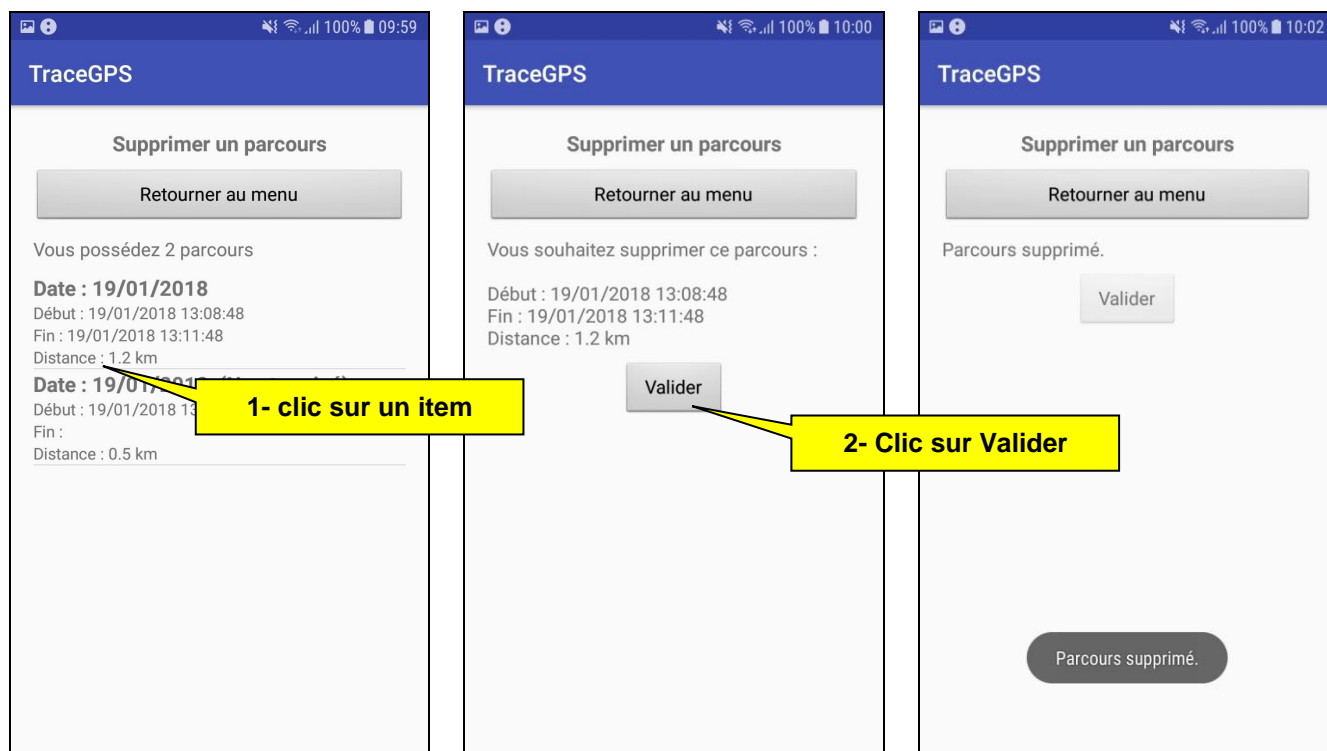
- 1- Situation de l'activité dans la structure de l'application
- 2- Modification du fichier strings.xml
- 3- Création de l'activité
- 4- Création de l'interface graphique
- 5- Modification de la programmation Java de MenuGeneral.java
- 6- Programmation Java de l'activité SupprimerParcours.java

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



2- Modification du fichier strings.xml

L'interface graphique à créer :

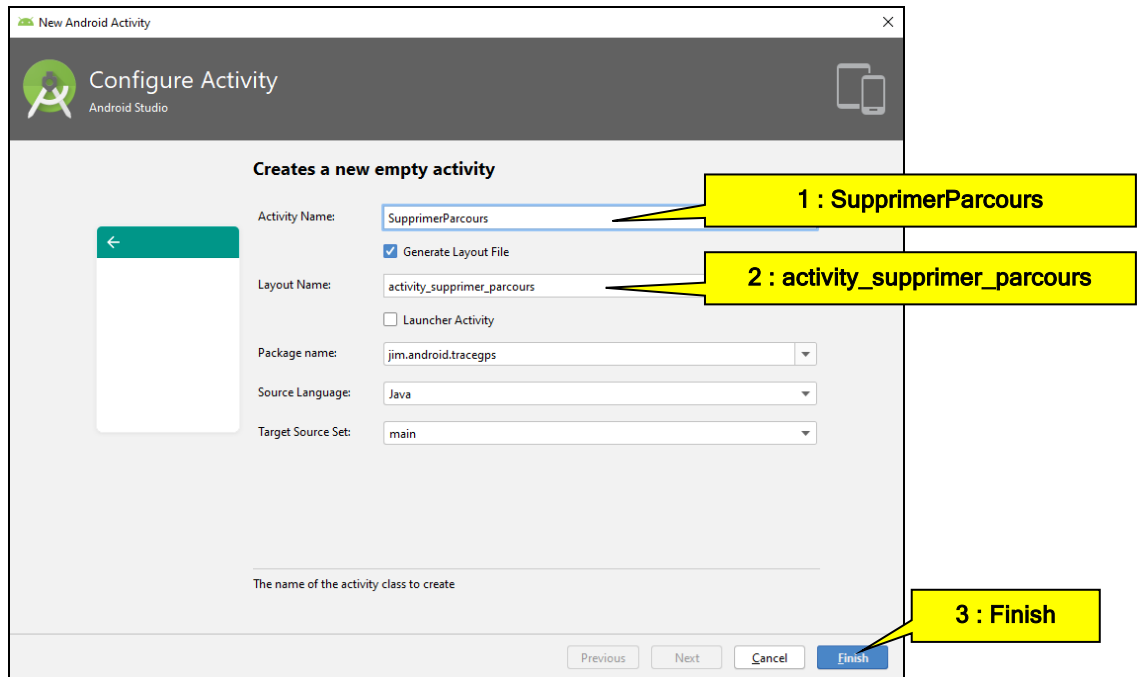


Dans le dossier **res/values**, complétez le fichier **strings.xml** avec le code suivant :

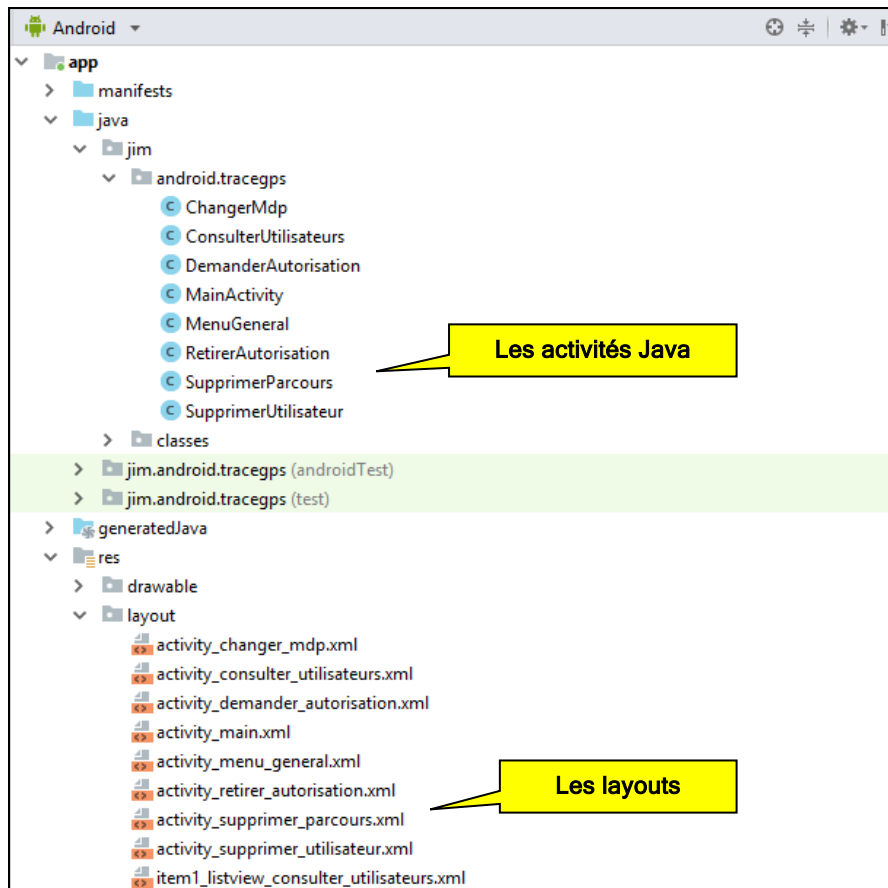
```
<!-- Les textes de la page de suppression d'un parcours -->
<string name="supprimer_parcours_titre1">Supprimer un parcours</string>
<string name="supprimer_parcours_bouton_retourner">Retourner au menu</string>
<string name="supprimer_parcours_bouton_valider">Valider</string>
```

3- Création de l'activité

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



L'activité **SupprimerParcours.java** et le layout **activity_supprimer_parcours.xml** sont alors créés :



4- Création de l'interface graphique

4-1 Création du layout de l'activité

A la création d'une nouvelle activité, l'interface comporte automatiquement un **ConstraintLayout** vide.

Comme d'habitude, nous allons commencer par remplacer le **ConstraintLayout** proposé par un **LinearLayout (vertical)** qui est beaucoup plus souple pour positionner les objets graphiques.

Le **ConstraintLayout** ne pouvant être ni modifié ni supprimé en mode **Design**, on va donc le modifier en mode **Text** :

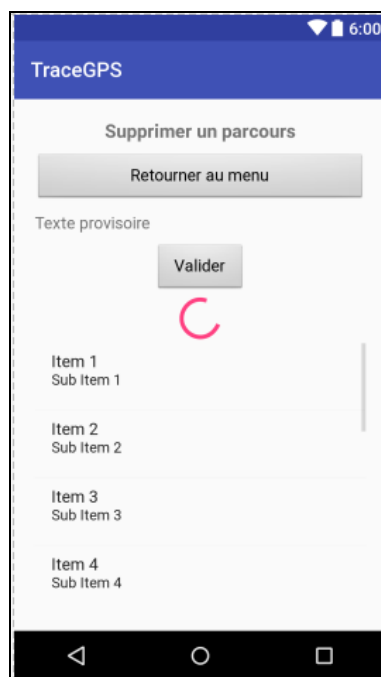
The screenshot shows the XML code for the activity layout in the 'Text' mode. The code is as follows:

```
<?xml version="1.0" encoding="utf-8"
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:padding="@dimen/tailleMarges"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="jim.android.tracegps.SuppresserParcours">
</LinearLayout>
```

Annotations:

- Remplacez ConstraintLayout par LinearLayout**: Points to the `<LinearLayout` tag.
- Supprimez cette ligne**: Points to the `xmlns:app="http://schemas.android.com/apk/res-auto"` line.
- Ajoutez ces 2 lignes**: Points to the `android:padding="@dimen/tailleMarges"` and `tools:context="jim.android.tracegps.SuppresserParcours">` lines.

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



The screenshot shows the 'Component Tree' in the IDE. The root is 'LinearLayout (vertical)'. It contains the following components:

- `supprimer_parcours_textView1 (TextView)` - "@string/supprimer_parcours_titre1"
- `supprimer_parcours_buttonRetourner (Button)` - "@string/supprimer_parcours_bouton_retourner"
- `supprimer_parcours_textViewMessage (TextView)` - "@string/texte_provisoire"
- `supprimer_parcours_buttonValider (Button)` - "@string/supprimer_parcours_bouton_valider"
- `supprimer_parcours_progressBar (ProgressBar)`
- `supprimer_parcours_listView1 (ListView)`

Annotation: **Soyez méthodiques dans le choix de vos identifiants** points to the component names in the tree.

Le code XML du layout :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:padding="@dimen/tailleMarges"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="jim.android.tracegps.SupprimerParcours">

    <TextView
        android:id="@+id/supprimer_parcours_textView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="10dp"
        android:text="@string/supprimer_parcours_titre1"
        android:textAlignment="center"
        android:textSize="18sp"
        android:textStyle="bold" />

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

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

    <Button
        android:id="@+id/supprimer_parcours_buttonValider"
        style="@android:style/Widget.Button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="@string/supprimer_parcours_bouton_valider"
        android:textSize="16sp" />

    <ProgressBar
        android:id="@+id/supprimer_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/supprimer_parcours_listView1"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

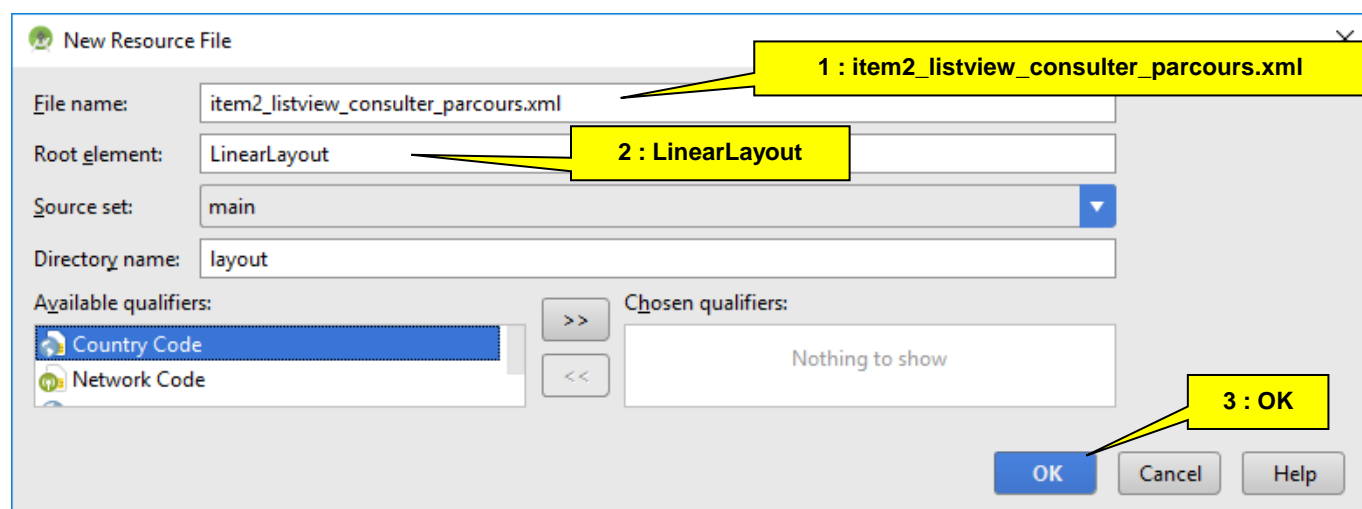
</LinearLayout>
```

4-2 Création d'un layout pour l'affichage des données de chaque parcours

Créez un nouveau layout pour afficher les données de chaque item du ListView, en visualisant pour chaque item :

- la date (en gras) et l'indication "non terminé" si le parcours n'est pas terminé
- l'heure de début
- l'heure de fin (si le parcours est terminé)
- la distance parcourue en km

Créer ce nouveau layout en faisant un clic droit sur le dossier **res / layout** du projet **app** et en choisissant la commande **New / Layout resource file** :



Ce layout contiendra 5 contrôles **TextView** ; vous pouvez le créer directement en mode **Text** :

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

    <TextView android:id="@+id/item2_textView_date"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="16sp"
        android:textStyle="bold" />

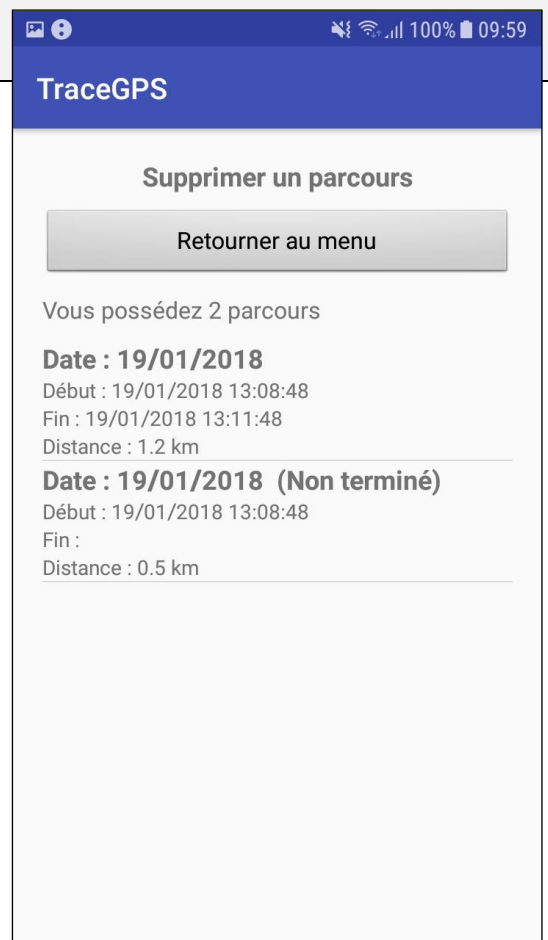
    <TextView android:id="@+id/item2_textView_heure_debut"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView android:id="@+id/item2_textView_heure_fin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView android:id="@+id/item2_textView_distance"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Les 4 TextView



5- Modification de la programmation Java de MenuGeneral.java

Complétez l'écouteur d'événement associé à **buttonSupprimerParcours** :

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

Testez cette étape sur un mobile réel **en vous connectant avec un compte utilisateur** et corrigez les erreurs si besoin.

Le bouton **Supprimer un parcours** doit activer l'activité **SupprimerParcours** :



6- Programmation Java de l'activité SupprimerParcours.java

6-1 Déclarations diverses et initialisation des objets

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

```
package jim.android.tracegps;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

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 SupprimerParcours extends AppCompatActivity {

    // les objets du layout
    private TextView textViewMessage;           // le TextView pour afficher le message
    private Button buttonRetourner;             // le Button pour retourner au menu
    private Button buttonValider;               // le Button pour valider la suppression
    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_supprimer_parcours);

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

        // récupération des objets du layout grâce à leur ID
        textViewMessage = (TextView) findViewById(R.id.supprimer_parcours_textViewMessage);
        buttonRetourner = (Button) findViewById(R.id.supprimer_parcours_buttonRetourner);
        buttonValider = (Button) findViewById(R.id.supprimer_parcours_buttonValider);
        progressBar = (ProgressBar) findViewById(R.id.supprimer_parcours_progressBar);
        laListView = (ListView) findViewById(R.id.supprimer_parcours_listView1);

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

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

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

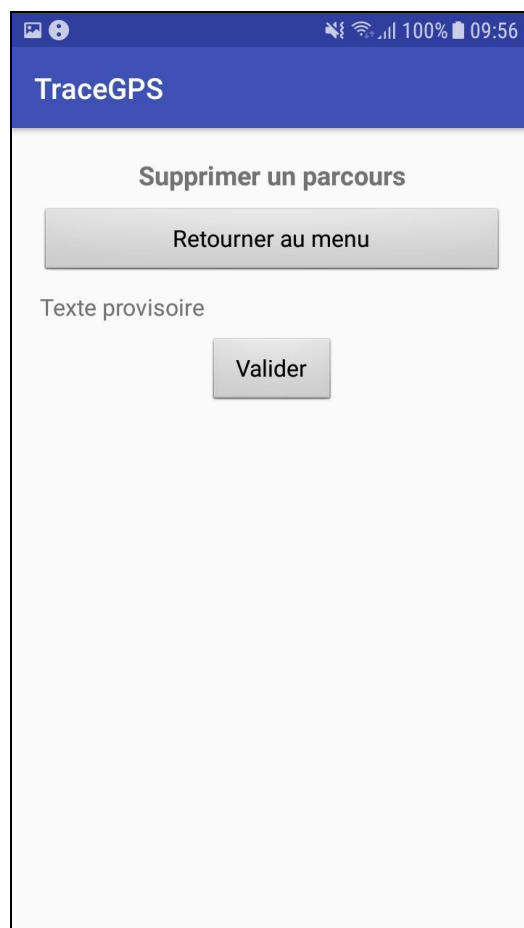
```
/** classe interne pour gérer le clic sur le bouton buttonRetourner. */
private class buttonRetournerClickListener implements View.OnClickListener {
    public void onClick(View v) {
        finish();
    }
}

/** classe interne pour gérer le clic sur le bouton buttonValider. */
private class buttonValiderClickListener implements View.OnClickListener {
    public void onClick(View v) {
        // A COMPLETER PLUS TARD
    }
}

/** 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** :



6-2 Mise en place d'une tâche asynchrone

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

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

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<Trace> lesTraces; // contient la collection des traces de l'utilisateur
```

A la fin de l'activité, ajoutez la tâche asynchrone **TacheGetLesTracesDeLUtilisateur** et la fonction provisoire **afficherLesTraces** (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

```
}
```

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

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

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_supprimer_parcours);

    // 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.supprimer_parcours_textViewMessage);
    buttonRetourner = (Button) findViewById(R.id.supprimer_parcours_buttonRetourner);
    buttonValider = (Button) findViewById(R.id.supprimer_parcours_buttonValider);
    progressBar = (ProgressBar) findViewById(R.id.supprimer_parcours_progressBar);
    laListView = (ListView) findViewById(R.id.supprimer_parcours_listView1);

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

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

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

Masquer le bouton **buttonValider**.

Instancier la collection **lesTraces**.

Lancer l'exécution de la tâche asynchrone **TacheGetLesTracesDeLUtilisateur**.

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



6-3 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;
```

Complétez 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("Vous possédez " + lesTraces.size() + " parcours");

    // 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");

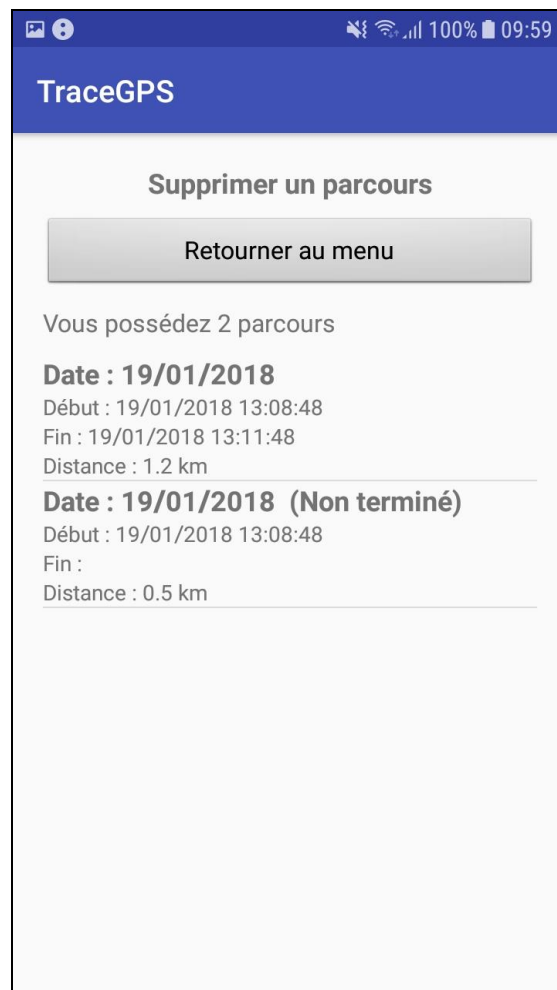
        // ajoute le HashMap dans le ArrayList
        lesElementsDuListView.add(element);
    }
}
```

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 :



6-4 Gestion du clic sur un item du ListView

On va maintenant gérer le clic sur un item du ListView en masquant la ListView et en affichant le bouton `buttonValider` :



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

```
private int idParcoursASupprimer; // l'id du parcours à supprimer
```

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) {
        // recherche du parcours choisi à partir de la position de l'item choisi
        Trace leParcoursASupprimer = lesTraces.get(position);
        idParcoursASupprimer = leParcoursASupprimer.getId();
        laListView.setVisibility(View.GONE);
        buttonValider.setVisibility(View.VISIBLE);
        String msg = "Vous souhaitez supprimer ce parcours :\n\n";
        msg += "Début : " + Outils.formaterDateHeureFR(leParcoursASupprimer.getDateHeureDebut()) + "\n";
        if (leParcoursASupprimer.getTerminee())
            msg += "Fin : " + Outils.formaterDateHeureFR(leParcoursASupprimer.getDateHeureFin()) + "\n";
        else
            msg += "Fin : " + "\n";
        msg += "Distance : " + leParcoursASupprimer.getDistanceTotale() + " km";
        textViewMessage.setText(msg);
    }
}
```

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

Exécutez et testez.

6-5 Gestion du clic sur le bouton de validation

Cette action nécessite d'appeler le service web :

- **SupprimerUnParcours** : pour supprimer un parcours

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

```
import android.widget.Toast;
```

A la fin de l'activité, ajoutez la tâche asynchrone **TacheSupprimerParcours** (vous pouvez vous inspirer du document "**5-3 (1) Projet TraceGPS - Dév appli android - ChangerMdp**") :

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

private class TacheSupprimerParcours extends AsyncTask<Void, Void, String> {

    La fonction onPreExecute doit désactiver les boutons buttonRetourner et buttonValider et démarrer l'affichage de l'objet progressBar.

    La fonction doInBackground doit appeler le service web SupprimerUnParcours en utilisant une des méthodes statiques de la classe PasserelleServicesWebXML et en lui passant les paramètres nécessaires.

    La fonction onPostExecute doit réactiver le bouton buttonRetourner, arrêter l'affichage de l'objet progressBar et afficher dans l'objet textViewMessage et dans un toast fugitif le message retourné par la méthode.

}
```

Complétez l'écouteur **buttonValiderClickListener** pour appeler la tâche **TacheSupprimerParcours**:

```
/** classe interne pour gérer le clic sur le bouton buttonValider. */
private class buttonValiderClickListener implements View.OnClickListener {
    public void onClick(View v) {
        // appel du service web SupprimerUnParcours avec une tâche asynchrone
        new TacheSupprimerParcours().execute();
    }
}
```


Testez l'application ; vous devez obtenir des dialogues similaires à ceux-ci :



L'utilisateur peut vérifier la suppression en retournant sur la page de suppression et en constatant que le parcours ne figure plus dans la liste de ses parcours :

