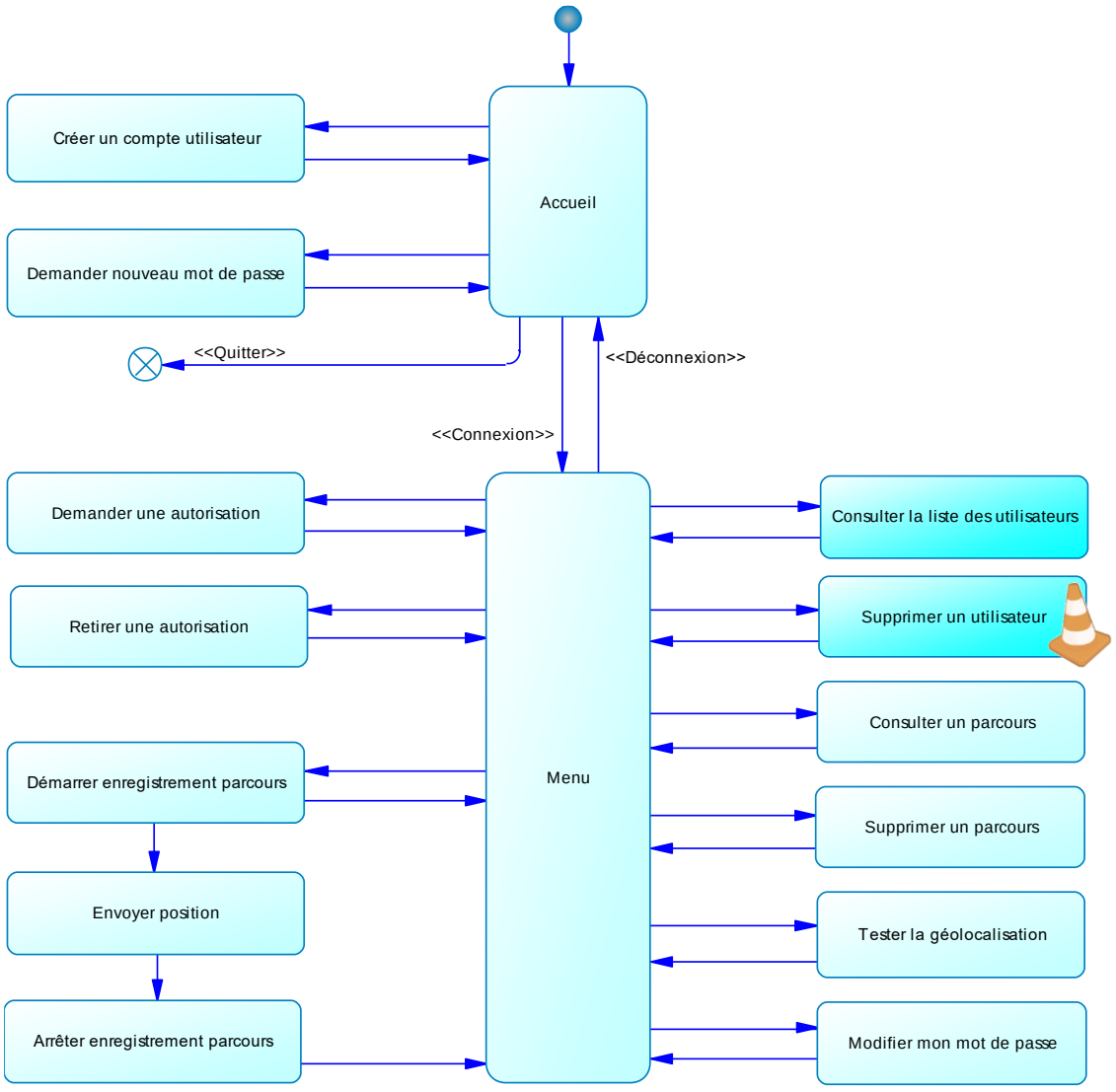
5- Développement de l'application mobile Android

5-7 SupprimerUtilisateur (supprimer un utilisateur)

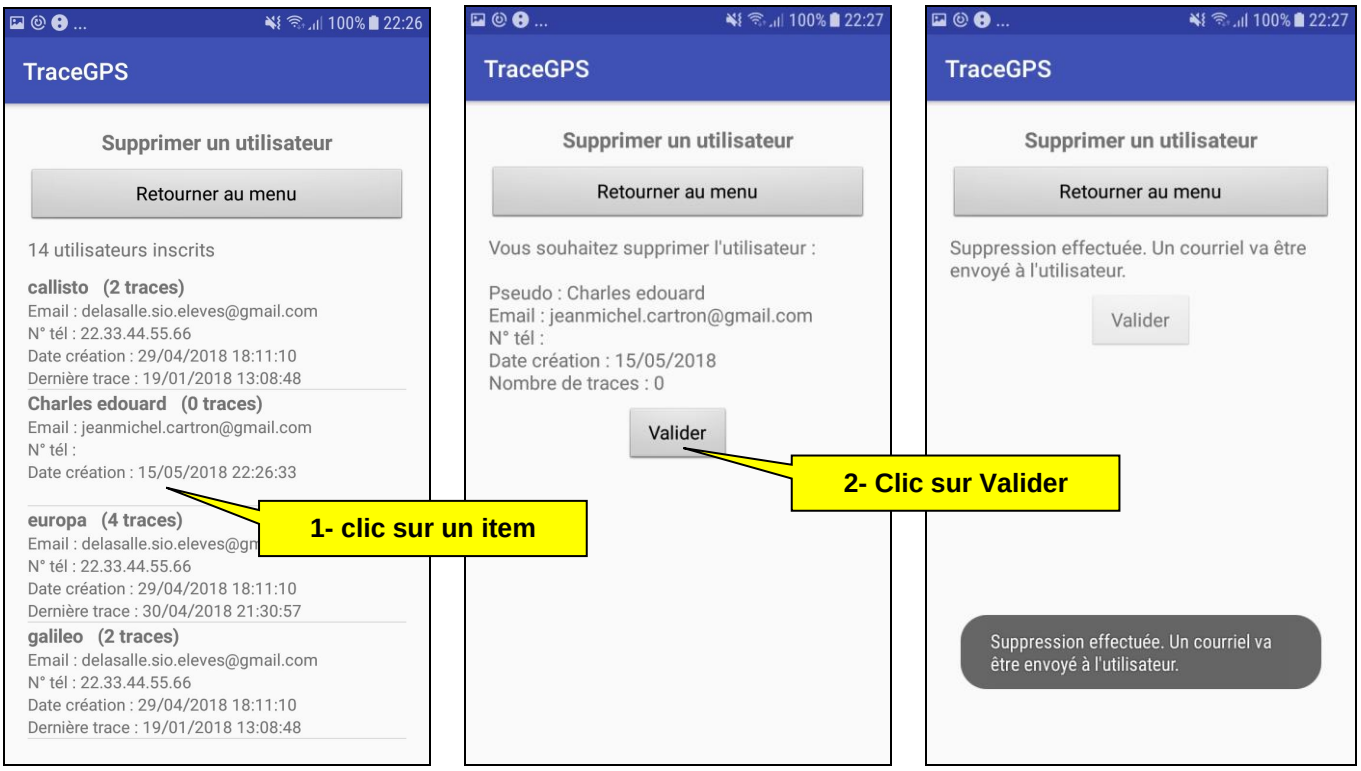
1- Situation de l'activité dans la structure de l'application
2- Modification du fichier strings.xml
3- Création de l'activité
4- Création de l'interface graphique
5- Modification de la programmation Java de MenuGeneral.java
6- Programmation Java de l'activité SupprimerUtilisateur.java

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



2- Modification du fichier strings.xml

L'interface graphique à créer :

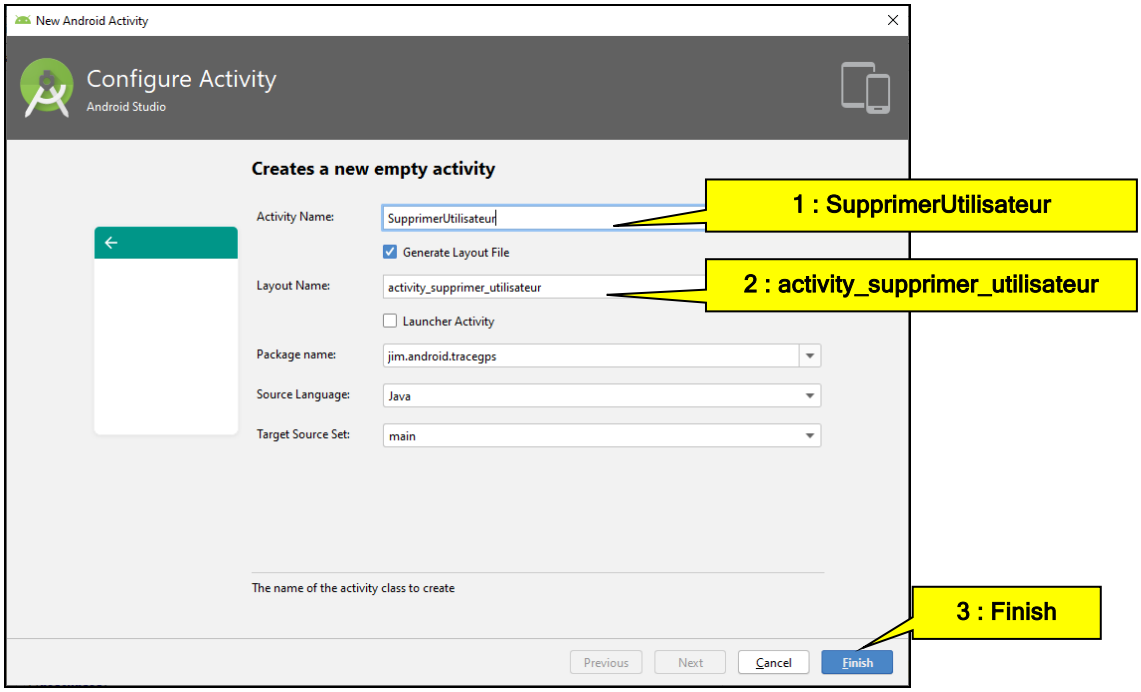


Dans le dossier **res/values**, complétez le fichier **strings.xml** avec le code suivant :

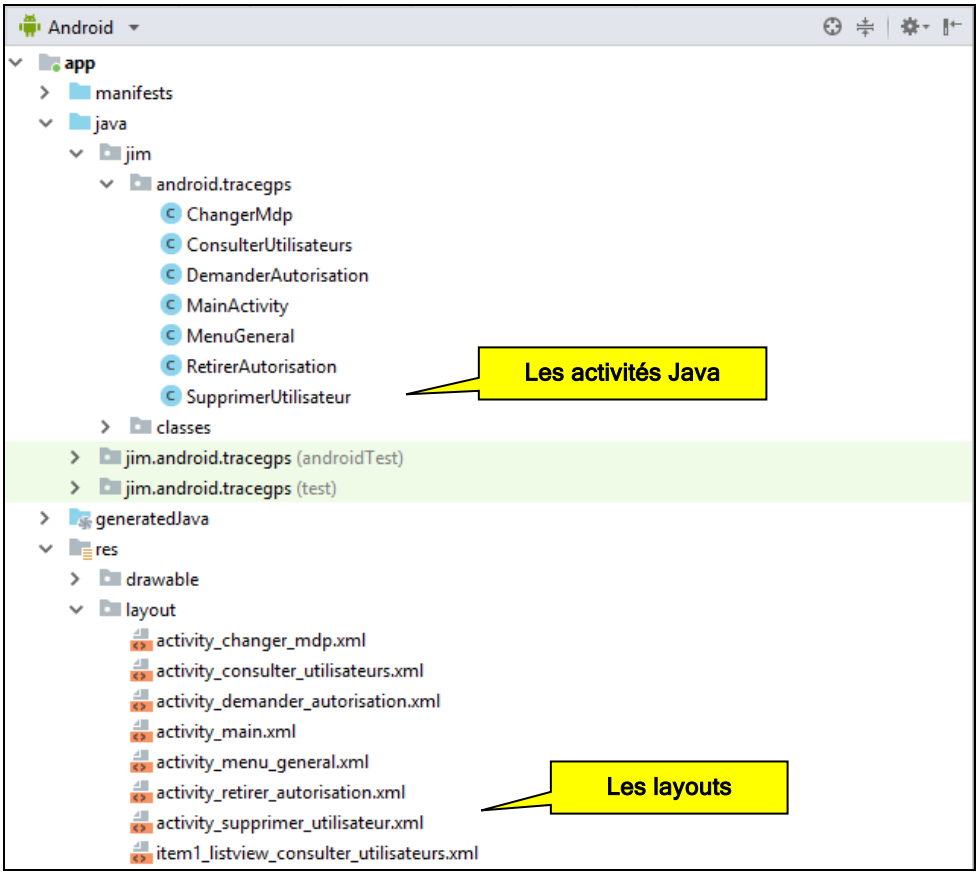
```
<!-- Les textes de la page de suppression d'un utilisateur -->
<string name="supprimer_utilisateur_titre1">Supprimer un utilisateur</string>
<string name="supprimer_utilisateur_bouton_retourner">Retourner au menu</string>
<string name="supprimer_utilisateur_bouton_valider">Valider</string>
```

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 **SupprimerUtilisateur.java** et le layout **activity_supprimer_utilisateur.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** :

Remplacez ConstraintLayout par LinearLayout

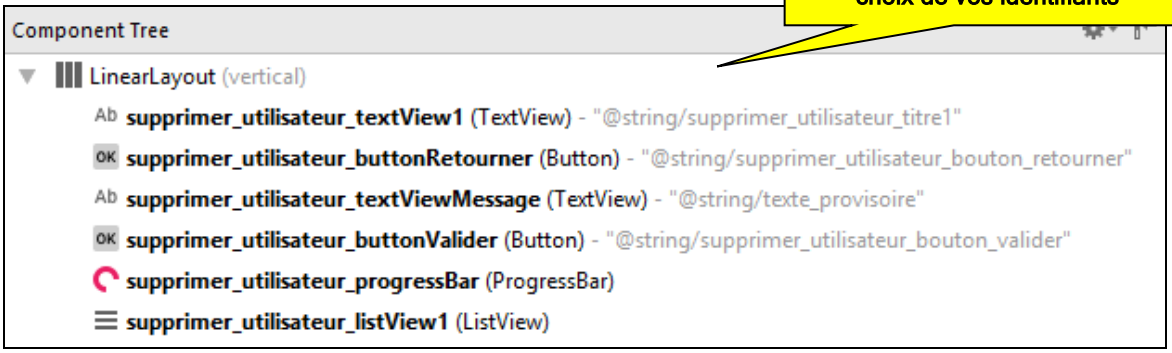
```
<?xml version="1.0" encoding
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:padding="@dimen/tailleMarges"
    android:layout_width="match_parent"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

</LinearLayout>
```

Supprimez cette ligne

Ajoutez ces 2 lignes

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** :

Soyez méthodiques dans le choix de vos identifiants



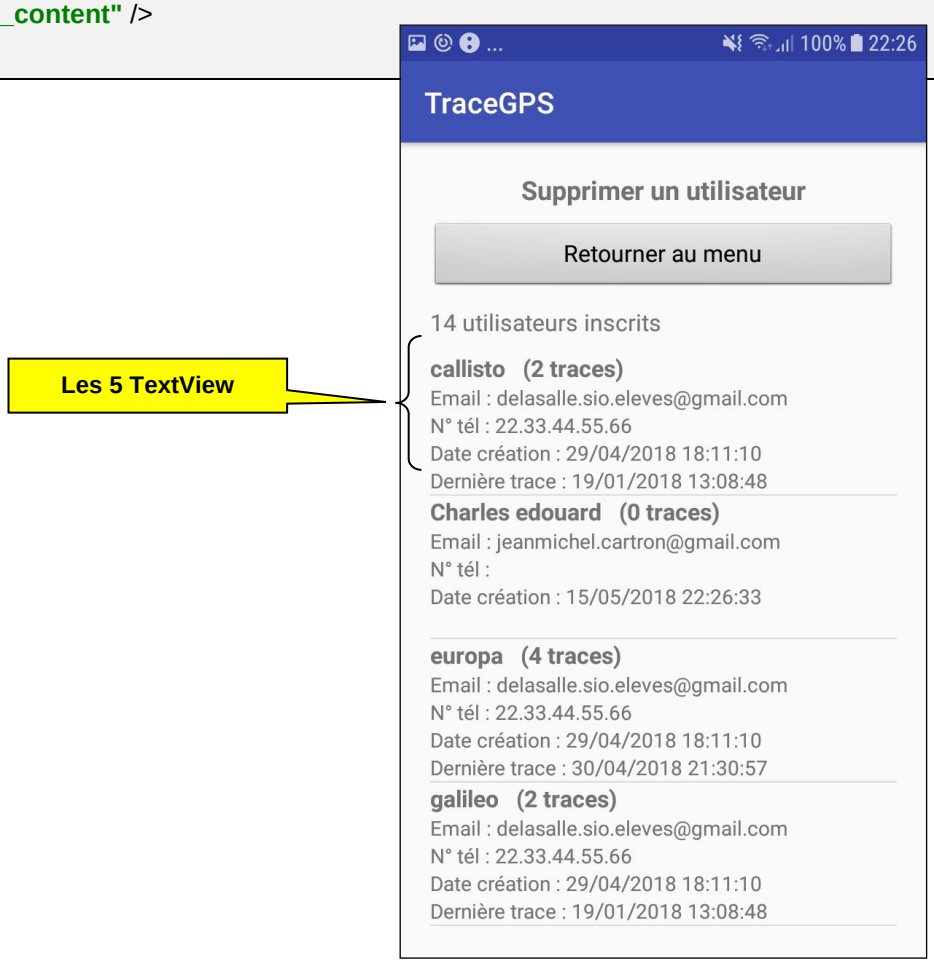
Le code XML du layout :

```
<?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.SupprimerUtilisateur">

   <TextView
      android:id="@+id/supprimer_utilisateur_textView1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:paddingBottom="10dp"
      android:text="@string/supprimer_utilisateur_titre1"
      android:textAlignment="center"
      android:textSize="18sp"
      android:textStyle="bold" />

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

   <TextView
      android:id="@+id/supprimer_utilisateur_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_utilisateur_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_utilisateur_bouton_valider"
      android:textSize="16sp" />

   <ProgressBar
      android:id="@+id/supprimer_utilisateur_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_utilisateur_listView1"
      android:layout_width="match_parent"
      android:layout_height="match_parent" />

</LinearLayout>
```

4-2 Utilisation du layout déjà créé pour l'affichage des données de chaque utilisateur

Le layout **item1_listview_consulter_utilisateurs.xml** (déjà créé) contient 5 contrôles **TextView** :

```
<?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="wrap_content"
    android:orientation="vertical">

    <TextView android:id="@+id/item1_textView_pseudo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="16sp"
        android:textStyle="bold" />

    <TextView android:id="@+id/item1_textView_adrmail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView android:id="@+id/item1_textView_numtel"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView android:id="@+id/item1_textView_date_creation"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView android:id="@+id/item1_textView_date_derniere_trace"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```



5- Modification de la programmation Java de MenuGeneral.java

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

```
/** classe interne pour gérer le clic sur le bouton buttonSupprimerUtilisateur. */
private class buttonSupprimerUtilisateurClickListener implements View.OnClickListener{
    public void onClick(View v) {
        // crée une Intent pour lancer l'activité
        Intent uneIntent = new Intent(MenuGeneral.this, SupprimerUtilisateur.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 **en vous connectant avec un compte administrateur**, et corrigez les erreurs si besoin.

Le bouton **Supprimer un utilisateur** doit activer l'activité **SupprimerUtilisateur** :

6- Programmation Java de l'activité SupprimerUtilisateur.java

6-1 Déclarations diverses et initialisation des objets

Dans le fichier **SupprimerUtilisateur.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 SupprimerUtilisateur 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 utilisateurs

    // 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_utilisateur);

        // 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_utilisateur_textViewMessage);
        buttonRetourner = (Button) findViewById(R.id.supprimer_utilisateur_buttonRetourner);
        buttonValider = (Button) findViewById(R.id.supprimer_utilisateur_buttonValider);
        progressBar = (ProgressBar) findViewById(R.id.supprimer_utilisateur_progressBar);
        laListView = (ListView) findViewById(R.id.supprimer_utilisateur_listView1);
        // arrête le cercle de chargement
        progressBar.setVisibility(View.GONE);
```

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

        // association d'un écouteur d'évenement à l'événement OnItemClic 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 utilisateurs nécessite d'appeler le service web :

- **GetTousLesUtilisateurs** : pour obtenir la liste de tous les utilisateurs

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> lesUtilisateurs;          // contient la collection des utilisateurs
```

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

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

private class TacheGetTousLesUtilisateurs 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 GetTousLesUtilisateurs en utilisant une des méthodes
    statiques de la classe PasserelleServicesWebXML et en lui passant les paramètres nécessaires. Cette
    méthode devra remplir la collection lesUtilisateurs.

    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

}
```

```
// afficher la liste des utilisateurs
public void afficherLesUtilisateurs() {
    // on affiche le nombre d'utilisateurs
    textViewMessage.setText(lesUtilisateurs.size() + " utilisateurs inscrits");
} // fin de la fonction afficherLesUtilisateurs
```

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

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

    // 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_utilisateur_textViewMessage);
    buttonRetourner = (Button) findViewById(R.id.supprimer_utilisateur_buttonRetourner);
    buttonValider = (Button) findViewById(R.id.supprimer_utilisateur_buttonValider);
    progressBar = (ProgressBar) findViewById(R.id.supprimer_utilisateur_progressBar);
    laListView = (ListView) findViewById(R.id.supprimer_utilisateur_listView1);
    // arrête le cercle de chargement
    progressBar.setVisibility(View.GONE);

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

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

    Masquer le bouton buttonValider.

    Instancier la collection lesUtilisateurs.

    Lancer l'exécution de la tâche asynchrone TacheGetTousLesUtilisateurs.

}
```

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

6-3 Gestion de l'affichage des utilisateurs dans la ListView

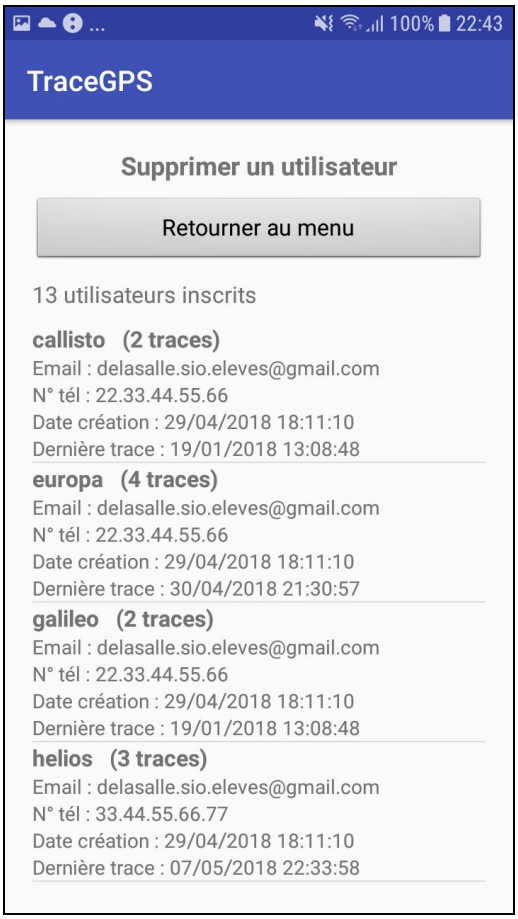
A la suite des **import** existants, ajoutez les **import** suivants :

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

Complétez la fonction définitive **afficherLesUtilisateurs** (vous pouvez vous inspirer du document "***5-4 (0) Projet TraceGPS - Dév appli android - ConsulterUtilisateurs***") :

```
// afficher la liste des utilisateurs
public void afficherLesUtilisateurs() {
    // on affiche le nombre d'utilisateurs
    textViewMessage.setText(lesUtilisateurs.size() + " utilisateurs inscrits");

    Créer l'ArrayList lesElementsDuListView qui permettra de remplir la listView.

    Créer un SimpleAdapter pour mettre les items de la liste lesElementsDuListView dans la vue
    item1_listview_consulter_utilisateurs.

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

} // fin de la fonction afficherLesUtilisateurs
```

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 String pseudoASupprimer;                    // le pseudo de l'utilisateur 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 pseudo choisi à partir de la position de l'item choisi
        Utilisateur utilisateurASupprimer  = lesUtilisateurs.get(position);
        pseudoASupprimer  = utilisateurASupprimer.getPseudo();
        laListView.setVisibility(View.GONE);
        buttonValider.setVisibility(View.VISIBLE);
        String msg = "Vous souhaitez supprimer l'utilisateur :\n\n";
        msg += "Pseudo : " + utilisateurASupprimer.getPseudo()  + "\n";
        msg += "Email : " + utilisateurASupprimer.getAdrMail()  + "\n";
        msg += "N° tél : " + utilisateurASupprimer.getNumTel()  + "\n";
        msg += "Date création : " + Outils.formaterDate(utilisateurASupprimer.getDateCreation())  + "\n";
        if (utilisateurASupprimer.getNbTraces() > 0)
            msg += "Date dernière trace : " + Outils.formaterDate(utilisateurASupprimer.getDateDerniereTrace())  + "\n";
        msg += "Nombre de traces : " + utilisateurASupprimer.getNbTraces();
        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 :

- **SupprimerUnUtilisateur** : pour suprimer un utilisateur

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

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

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

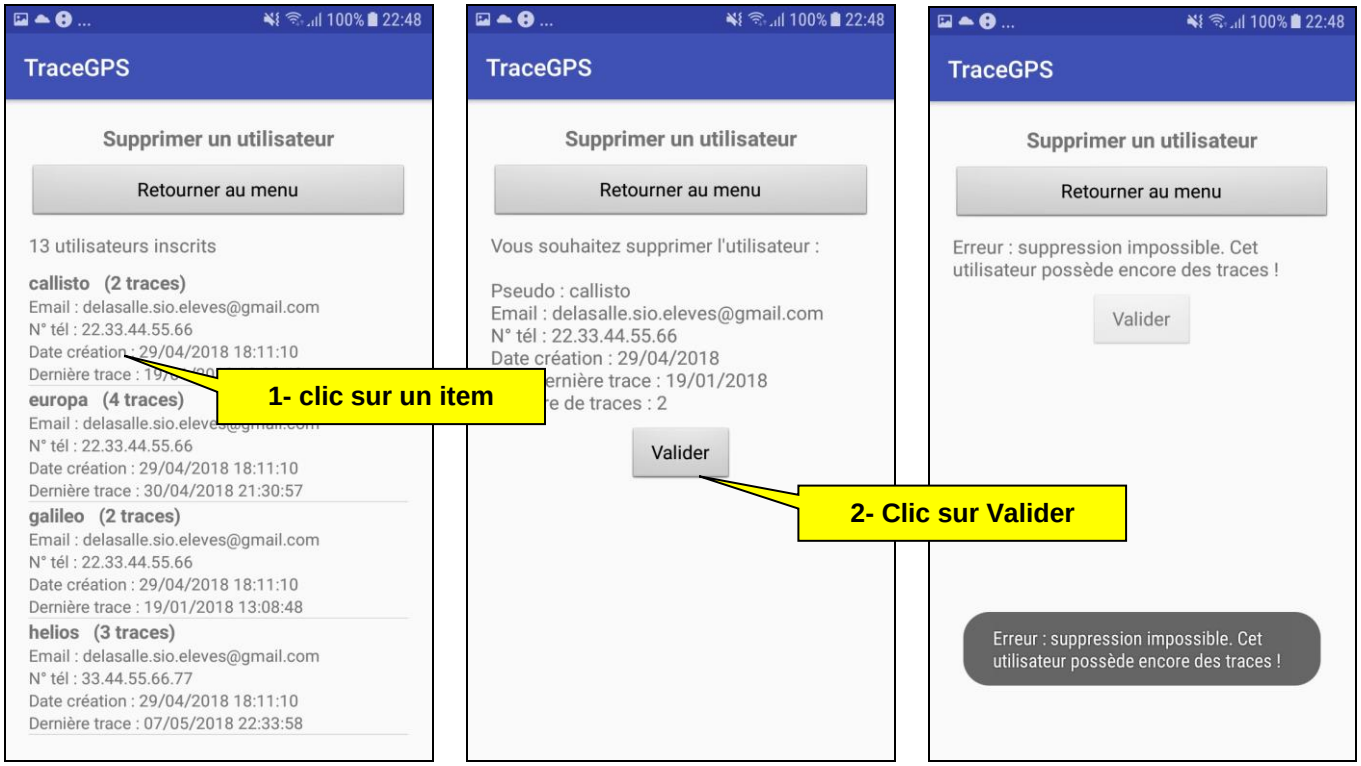
```
// ----------------------------------------------------------------------------------------------------------------------
// ---------- tâche asynchrone pour PasserelleServicesWebXML.supprimerUnUtilisateur ----------
// ----------------------------------------------------------------------------------------------------------------------

private class TacheSupprimerUnUtilisateur 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 SupprimerUnUtilisateur 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 **TacheSupprimerUnUtilisateur**:

```
/** 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 SupprimerUnUtilisateur avec une tâche asynchrone
        new TacheSupprimerUnUtilisateur().execute();
    }
}
```

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

En choisissant un utilisateur possédant des traces enregistrées :



En choisissant un utilisateur ne possédant pas de traces enregistrées :

L'utilisateur supprimé doit recevoir un mail similaire à celui-ci :

Suppression de votre compte dans le système TraceGPS Boîte de réception x

delasalle.sio.crib@gmail.com via 120-prod.mail-out.ovh.net
À moi

Bonjour jim123456

L'administrateur du service TraceGPS vient de supprimer votre compte utilisateur.

L'administrateur peut vérifier la suppression en retournant sur la page de suppression :