

TP1 : Les Bases d'Android

Programmation mobile

Mohamad Satea Almallouhi - Tony Nguyen
M1 Génie Logiciel
Faculté des Sciences
Université de Montpellier.

27 Février 2024



Table des matières

1	Hello world	3
2	Simple formulaire	3
2.1	Internationalisation	4
2.2	Événements	5
2.3	Intent explicite	6
2.4	Intent implicite	6
3	Consultation des horaires de trains	7
4	Simple d'agenda	7

Introduction

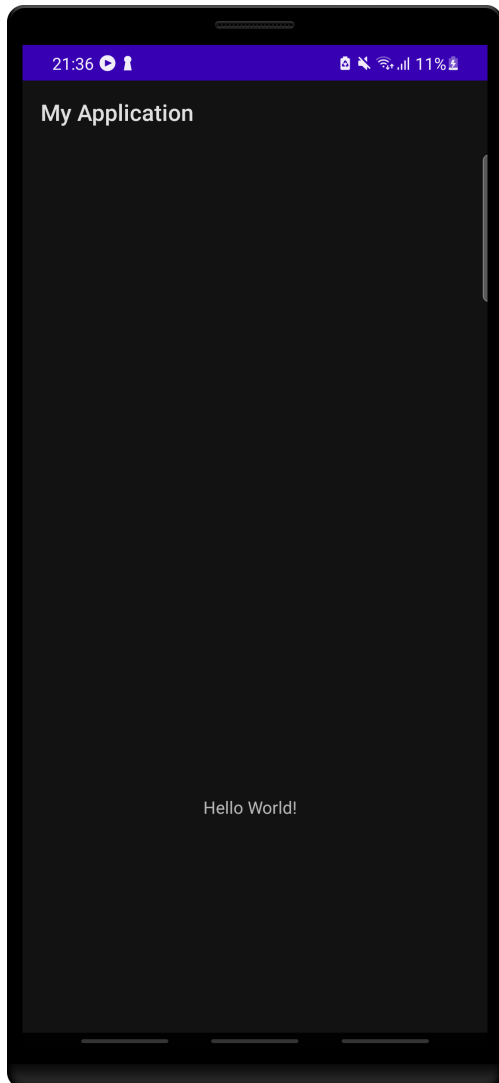
Dans ce TP, nous allons voir les bases de la programmation mobile pour Android en Java.

Nous allons voir comment afficher du texte mais aussi en saisir. De plus nous avons vue d'autres concept comme les ressources, les Intents et les activités.

Les sections de rapport suit les exercices.

1 Hello world

Nous allons voir comment afficher du texte à l'écran dans une activité sa vue associé.



```
<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_con
    android:layout_height="wrap_co
    android:layout_marginStart="16
    android:layout_marginTop="160d
    android:layout_marginEnd="32dp
    android:layout_marginBottom="1
    android:text="Hello World!"
    app:layout_constraintBottom_to
```

Afin de l'afficher à l'écran on va utiliser ce **layout** dans une activité à l'aide de la fonction **setContentview()**.

Il est également nécessaire d'indiquer à l'application l'activité à lancer comme point d'entrée. Pour cela, dans le manifest, nous ajoutons la balise **<intent-filter>**, voir figure 1 (page 4) pour plus de détail.

```
public class MainActivity extends AppCompatActivity {
    SateaMail
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

2 Simple formulaire

Tout d'abord, dans un fichier .xml dans res/layout, nous déclarons une balise **<TextView>** avec un attribut **android:text** qui a pour valeur "Hello World!".

Nous allons voir comment faire une application android réalisant un formulaire où l'on demandera des informations de différentes nature à l'utilisateur.

FIGURE 1 – Manifest pour l'application HelloWorld

```
MyApplication4 > app > src > main > AndroidManifest.xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:tools="http://schemas.android.com/tools">
4
5      <application
6          android:allowBackup="true"
7          android:dataExtractionRules="@xml/data_extraction_rules"
8          android:fullBackupContent="@xml/backup_rules"
9          android:icon="@mipmap/ic_launcher"
10         android:label="@string/app_name"
11         android:roundIcon="@mipmap/ic_launcher_round"
12         android:supportRtl="true"
13         android:theme="@style/Theme.MyApplication"
14         tools:targetApi="31">
15
16         <activity
17             android:name=".MainActivity"
18             android:exported="true">
19             <intent-filter>
20                 <action android:name="android.intent.action.MAIN"/>
21                 <category android:name="android.intent.category.LAUNCHER" /> </intent-filter>
22             </activity>
23         </application>
24
25     </manifest>
```

21:33 51%

Formulaire

Prénom:
Tony

Nom:
Nguyen

Âge:
23

Profession:
Equipier

Portable:
Entrez Portable

Email:
Entez Email

SOUMETTRE

Afin d'implémenter chaque question de notre formulaire, nous utilisons dans notre layout les balises `<editText>` pour créer un champ de saisie et `<TextView>` pour l'étiquette associé.

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/button3"
    android:layout_marginTop="30dp"
    android:text="@string/first_name"
    android:textAppearance="?android:attr/textAppearanceLarge" />

<EditText
    android:maxLines="1"
    android:lines="1"
    android:singleLine="true"
    android:id="@+id/editTextName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/enter_first_name" />
```

2.1 Internationalisation

Le système Android étant largement répandu dans le monde, il est préférable de prévoir plusieurs

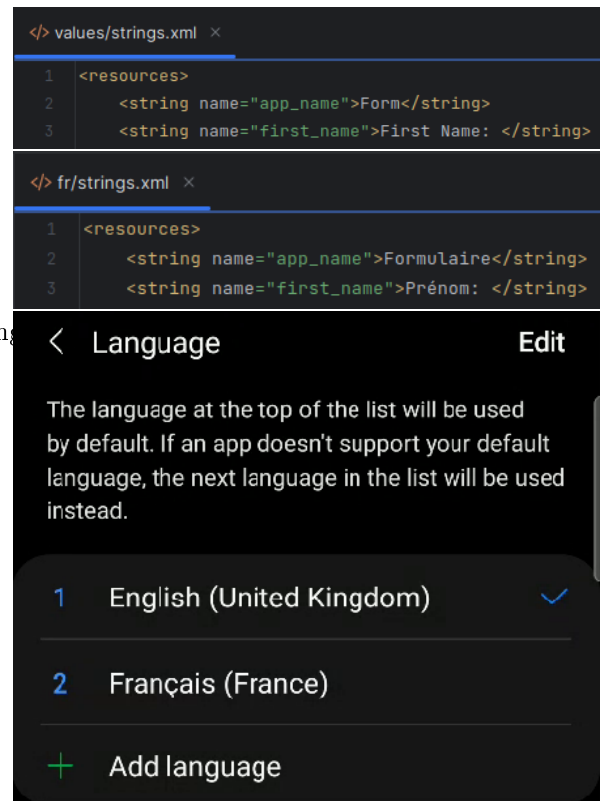
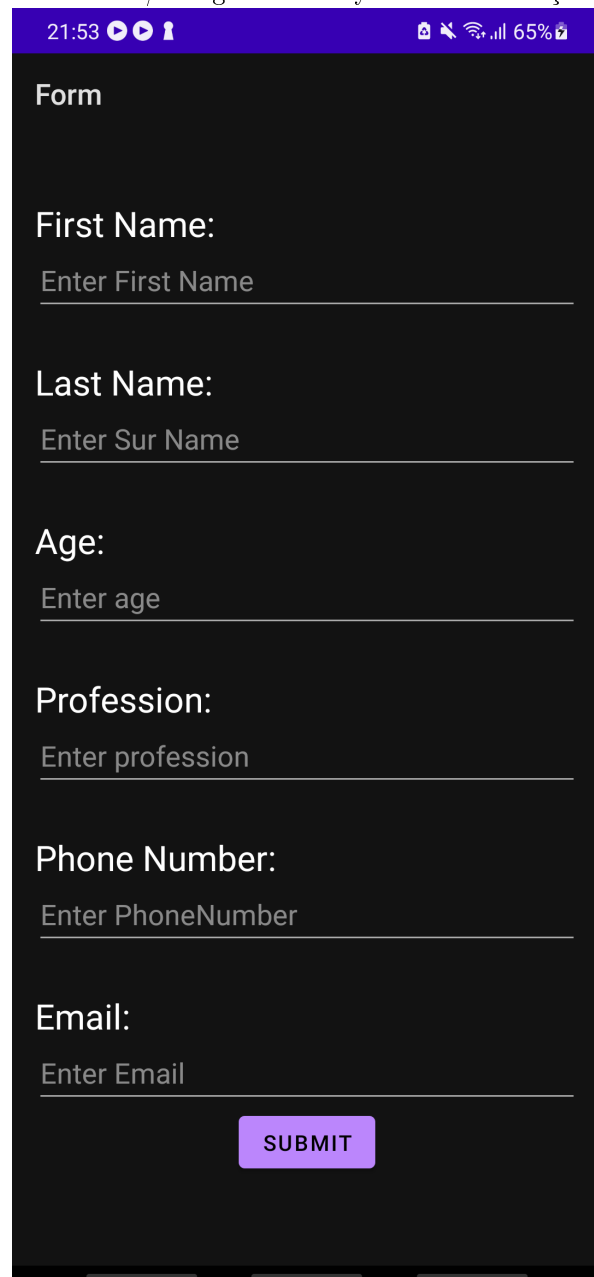
traduction pour des utilisateurs à l'international. Nous allons voir comment.

Pour cela nous allons utiliser le concept de **resources**.

Remarquons que dans le fichier de layout, dans les balises, les attributs text et hint n'ont pas de concrète mais plutôt **une référence (@type/name)**.

Dans res/values/strings.xml et res/values/fr/strings.xml nous déclarons les différents valeurs, mais avec le même attribut name.

En changeant la langue du système globale, l'application sera choisir entre le fichier strings.xml par défaut ou fr/strings.xml si le système est en français.

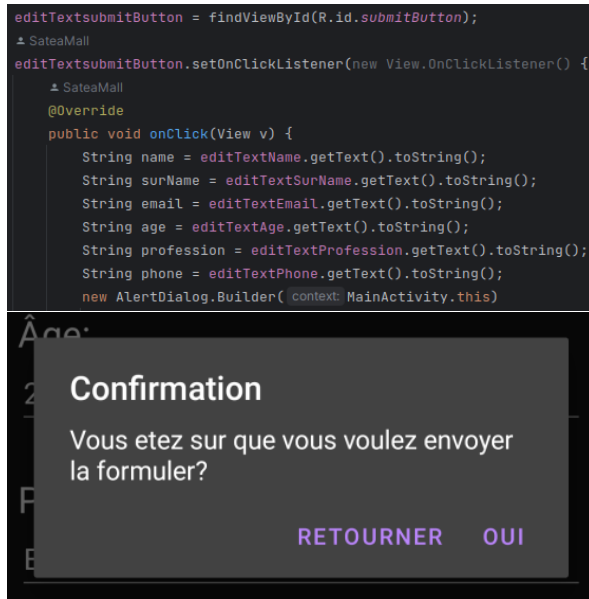


2.2 Événements

À présent, nous allons découvrir de quel façon réagir lors de l'action utilisateur "appuyer sur un bouton".

Avant tout, il est nécessaire de récupérer l'objet représentant le bouton. Une fois récupérer, nous allons pouvoir lui ajouter **un listener** afin qu'il réagisse à un appuie avec la fonction **setOnClickListener**. Nous lui donnons en argument une sous classe de **OnClickListener** en créant une classe anonyme héritant de **OnClickListener** dans laquelle nous **redéfinissons onClick()**. C'est dans cette fonction que nous pouvons décider quelles actions prendre lors de cet évènements.

Nous choisissons comme comportement du bouton, une simple demande de confirmation ou d'annulation. Cela est présenter sous la forme d'une fenêtre de dialogue. Android nous permet d'en créer une comme ceci : `"new AlertDialog.Builder().show();"`



2.3 Intent explicite

Qu'est ce qu'un "Intent" ?

Un Intent est un objet représentant la communication entre composant sur Android. Il représente une demande d'opération à exécuter. Il permet notamment de démarer des d'autres composants/application.

Dans un Intent **explicite**, l'activité à démarer est **explicitement nommé**.

Tout d'abord voyons ensemble les intents explicite.

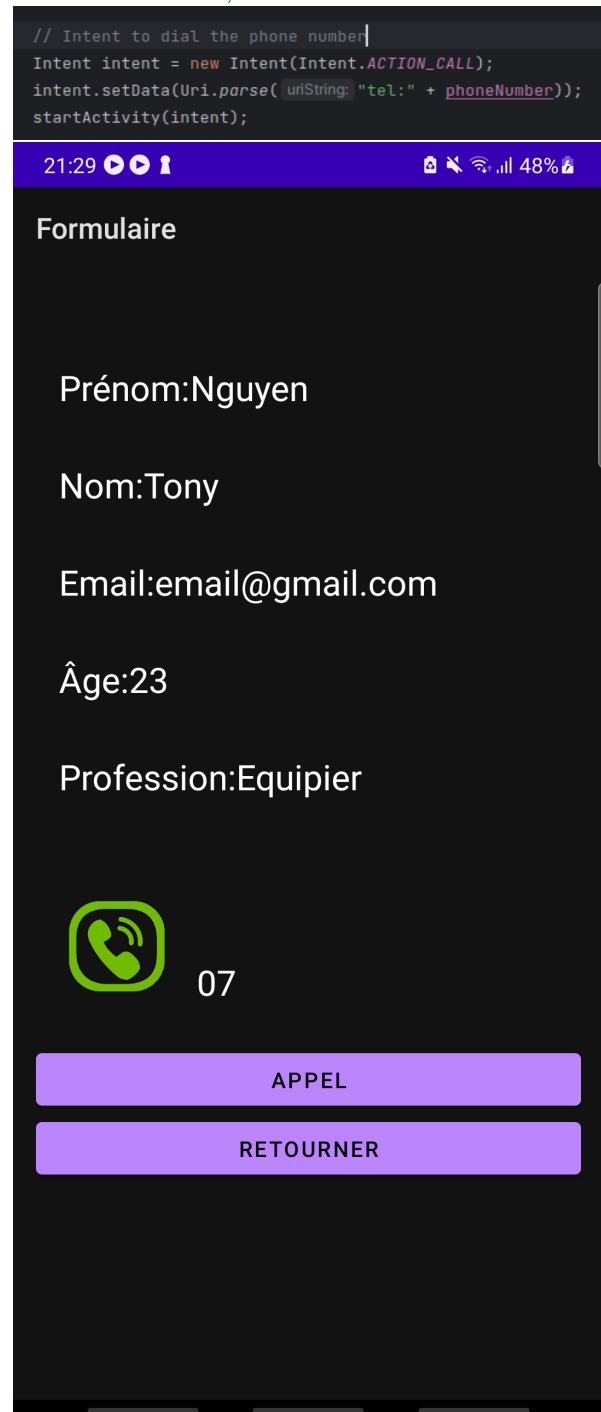
En effet, comme nous pouvons le remarquer, dans le constructeur de l'Intent, nous mettons en second paramètre la classe de l'activité à démarer. La méthode **putExtra** nous permet va nous permettre de transmettre des informations à l'activité à démarer. Pour effectivement lancer cette nouvelle activité, il nous suffit de faire **startActivity(uneIntent)**.

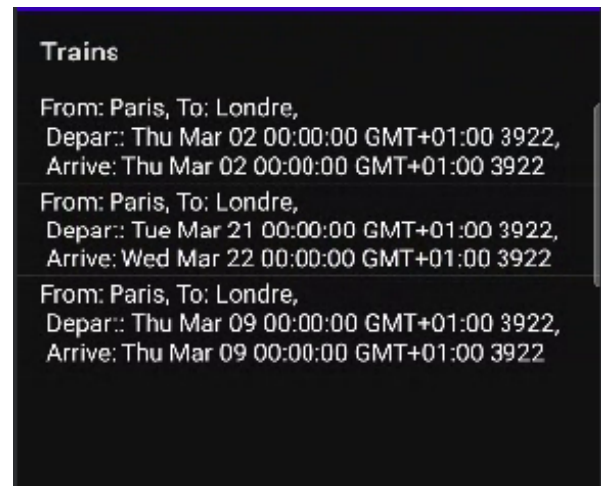


2.4 Intent implicite

Nous allons maintenant voir les intents implicite qui eux ne nécessite pas de nommer l'activité à démarer.

Comme on peut le voir sur l'image ci-dessous, nous n'indiquons pas le nom de l'activité à démarer, seulement l'action à réaliser.





3 Consultation des horaires de trains

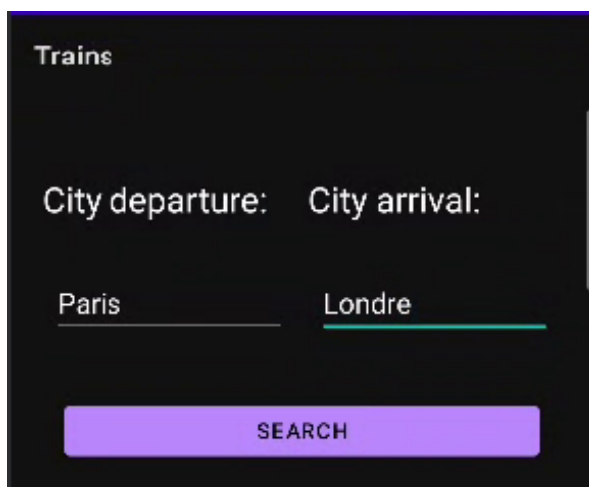
Nous allons faire une application pour consulter des horaires de trains.

La classe `Cherche` est l'activité d'entrée, la page d'accueil. Avec des objets `EditText`, on récupère les villes de départ et d'arrivée saisies.

Ensuite avec un `OnClickListener` et un `Intent`, on transmet les données saisies lors d'une pression sur le bouton "Search" à l'activité `Consultation`.

La classe `Trajet` sert seulement à représenter un Trajet.

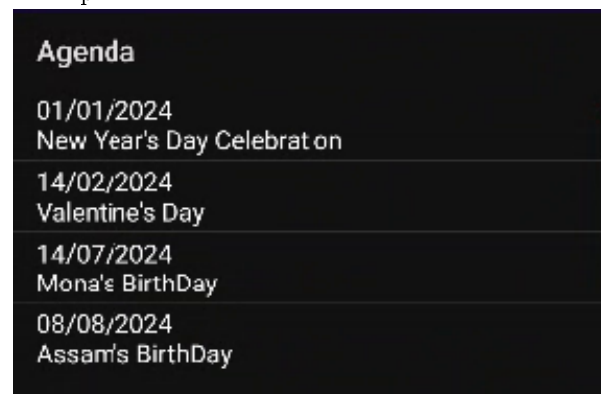
Nous pouvons voir le résultat sur les images suivantes avec le diagramme de classe de l'application sur la figure 2 (page 8).



4 Simple d'agenda

Nous allons réaliser une application d'agenda.

Voici la page de base de notre agenda `minimallist`. On représente différents événements avec leurs descriptions et leurs dates.



Nous avons la possibilité d'ajouter des événements.

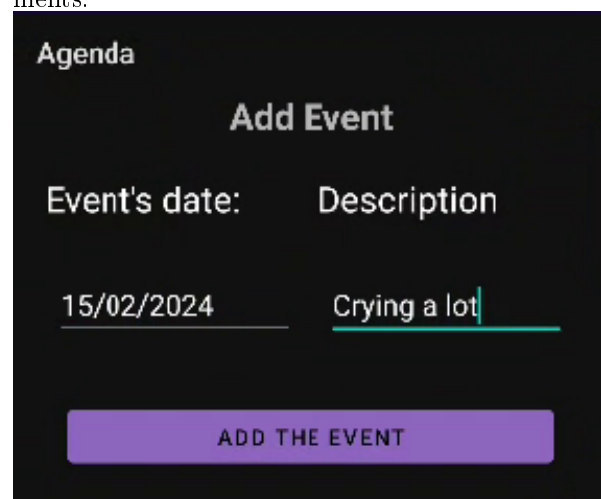
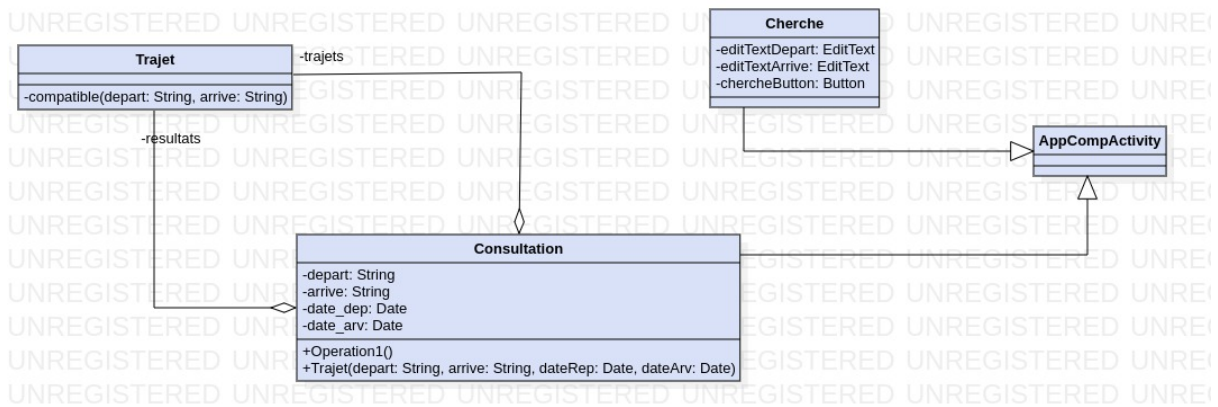
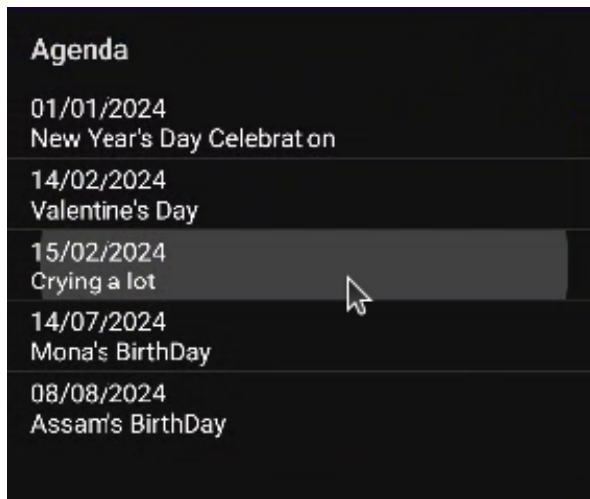


FIGURE 2 – Diagramme de classe de l'application Trains



Et effectivement, on le retrouve bien dans la liste d'événements.

Nous avons le diagramme de classe de l'application sur la figure 3 (page 9).



Démonstration

En ligne sur Youtube, à l'adresse URL <https://youtu.be/nQUkpSUjJlY> une démonstration vidéo de notre travail.

FIGURE 3 – Diagramme de classe de l'application Agenda

