HAI811I - Programmation mobile
COMTPE RENDU DU TP 1 - Les bases d'android

AMAH G. Richard

21 Février 2023

Table des matières

Table des figures

1 Introduction

Ce TP est le premier TP du module HAI811I - Développement et programmation pour supports mobiles, dont l'objectif est de présenter les bases permattemant de comprendre le développement logiciel sur des platformes mobiles de façon générale et particulièrement sur Android.
Il comporte une série d'exercices allant de l'installation de l'environnement de développement intégré utilisé aux applications avec des composants plus complexes en passant par les intents.
Le lien du dépôt du code source de ce TP se trouve dans la section ANNEXES et Réferences

2 Découverte de l'environnement de développement

L'objectif de cet exercice est de decouvrir lenvironnement de developement integre Android Studio utilise pour developper les applications pour la plateforme ou systeme d'exploitation ANDROID. L'IDE(Integrated Development Environment en Anglais) est directement telechargeable depuis le lien http ://developer.android.com. Il est utilisable sous Microsoft Windows, Mac OS, Linux et meme Chrome OS. Suivant le systeme d'exploitation la procedure d'installation differe. Au demarrage, on obtient l'interface suivante :

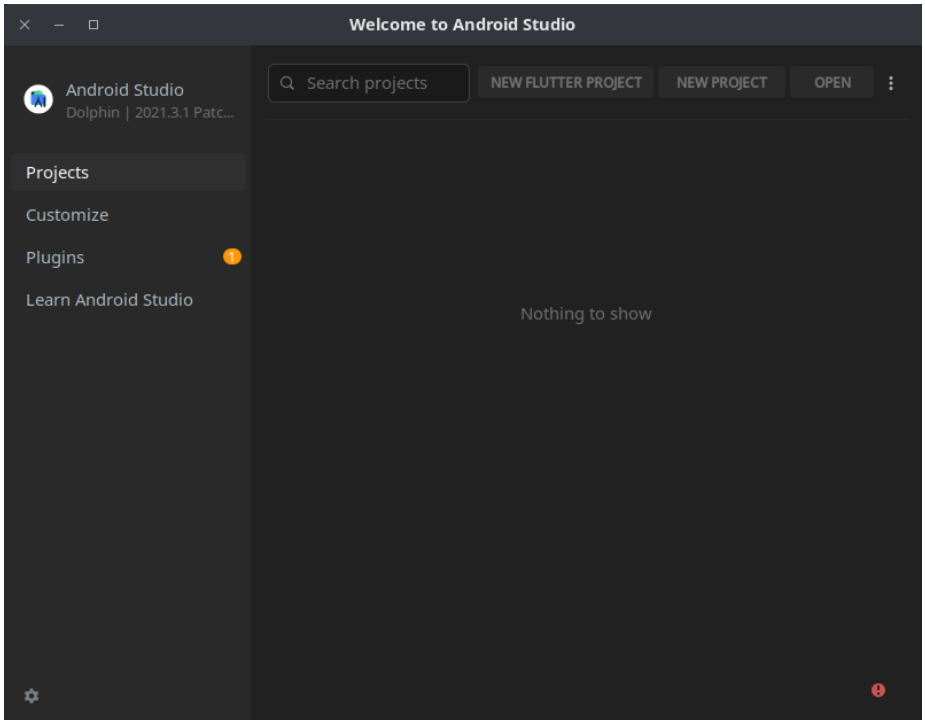

FIGURE 1 – Demarrage d'Android Studio

Pour achever l'installation de l'environnement, rendez-vous dans les paramètres accessibles par les trois points verticaux (en haut a droite), et installer les SDK souhaités via l'option SDK Manager, puis créer les émulateurs souhaités via l'option AVD Manager.

Lors du développement d'une application, la Figure 3 est un exemple d'interface d'un projet ouvert dans Android studio :

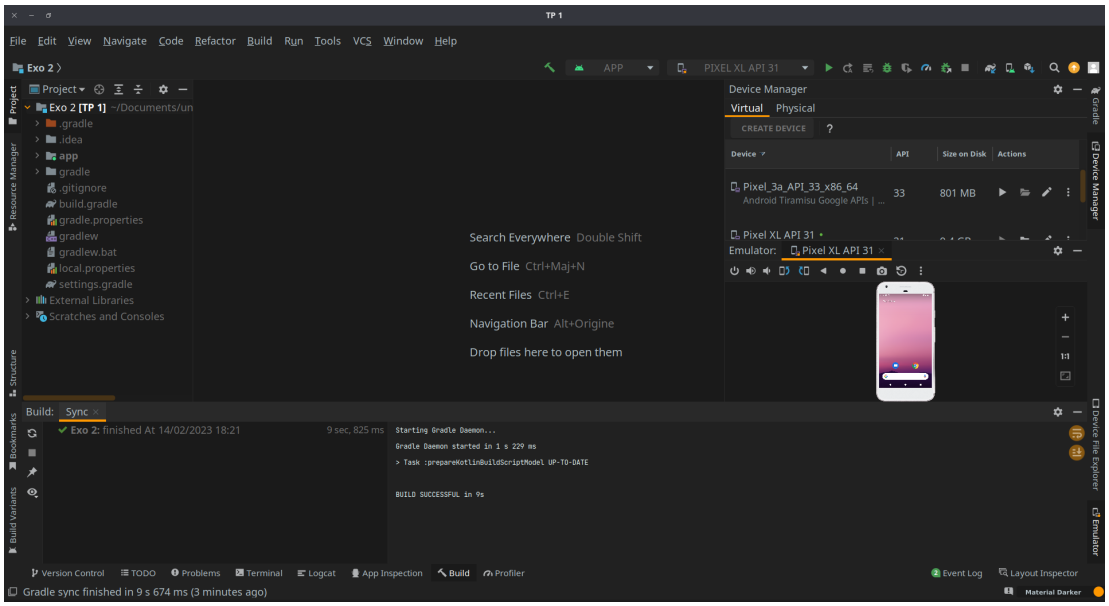

FIGURE 2 – Exemple d'un projet

Cette interface est constituée :

— D'une barre de navigation
— D'un panneau explorateur de projet (Panneau Project) permettant de naviguer dans le projet
— D'un gestionnaire d'appareil (Panneau Device Manager) dans le panneu droit de la fenêtre
— Et d'un espace d'édition de code (au milieu)

Notons que les autres parties de la fenêtre toute fois accessibles en navigant dans la barre du Menu d'Android studio.

3 Hello World

L'objectif de cet exercice est de créer et exécuter l'application basique ⟨⟨Hello world ⟩⟩, et d'inspecter ses differents elements.

Pour créer une telle application basique, depuis l'écran d'accueil d'Android Studio, On fait :
Nouveau Projet ⟩Activité vide ⟩Nom de l'application ⟩Finir.

A l'exéuction de cette application voici qu'on obtient :

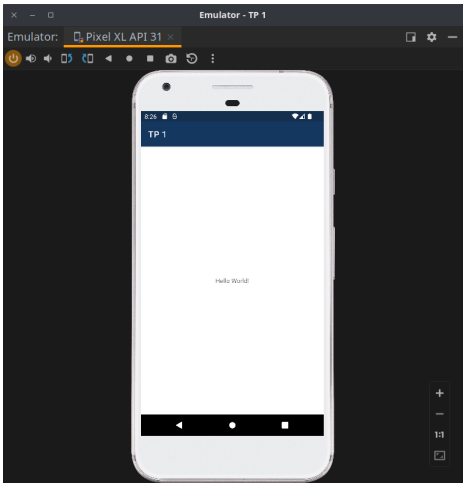

FIGURE 3 – Application Hello World

La présente application ne contient qu'une seule activité, qui à sa création ne renvoit qu'une vue contenant un texte centré disant ⟨⟨Hello world⟩⟩.

Les applications android natives sont constituées de plusieurs éléments à savoir un fichier de configuration constituant le point d'entrée de l'application, un répertoire des resources utilisées dans l'application telles que les chaînes de caractères, les gabarits (mise en pages), les couleurs etc ; un répertoire des fichiers source ; etc ...

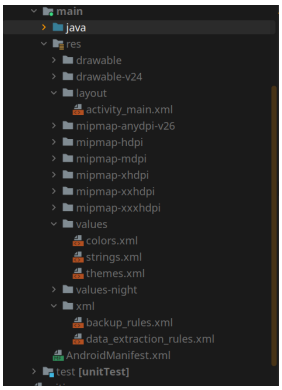

FIGURE 4 – Quelques éléments constitutifs de l'application Hello World

Comme mentionné, plus haut une seule activité est contenue dans cette application, MainActivity :

```java
// MainActivity.java
/** Class implementing interface */

package com.example.tp1;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

Listing 1 – MainActivity.java

Ici, le cycle de vie de cette activité ne se resume qu'à sa création. A la création, une resource gabarit en xml est appliquée à sa vue.

Enfin, voici la resource xml en question :

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- CECI EST UN COMMENT -->
<androidx.constraintlayout.widget.ConstraintLayout
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:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Listing 2 – applicationContext.xml

Par defaut, le gabarit contient un ConstraintLayout, changeable (et/ou adaptable) selon par d'autre type selon nos besoins. Il contient une Vue ⟨⟨TextView⟩⟩occupant des dimensions optimales selon la longueur du texte qu'elle contient.

4 Une première application - Interface simple

Dans cette question, il est demandé de créer une interface demandant à son utilisateur de remplir un formulaire contenant des champs Nom, Prénom, âge, Domaine de compétence, numéro de téléphone et enfin un boutton de validation. Cette interface peut être enrichie en donnant des indices de valeurs possibles attendues dans chaque champ correspondants ainsi que leurs noms à gauche. Cela peut se faire en définissant la vue complete du formulaire en xml ou directement dans le code java de l'activité concernée.

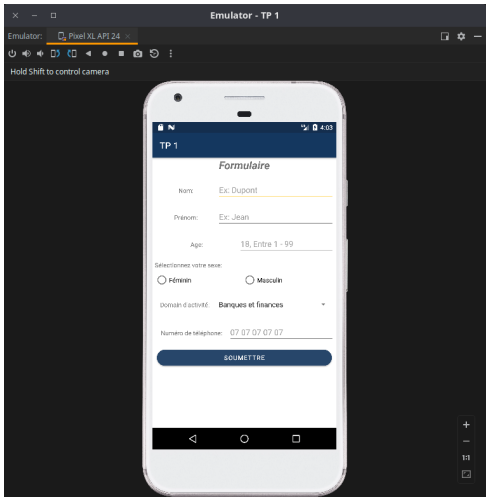

FIGURE 5 – Vue en XML

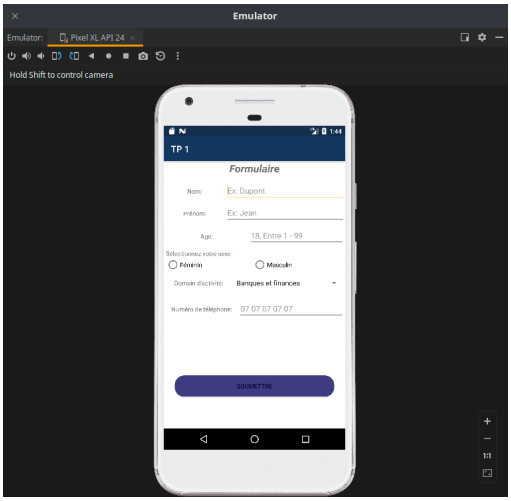

FIGURE 6 – Vue en Java

Ci-dessus, nous avons sur l'image à gauche, le rendu de la vue définie en xml et à droite celui de la vue directement définie dans le code java.

5 Internationalisation des interfaces

Cette question traite de moyen de localiser notre application selon la region de l'utilisateur en internalisant l'interface utilisateur de cette applicaition. Ceci se fait en externalisant les resources telles ques les chaînes de caractères, les images, les couleurs, les styles, les vues des écrans (etc...) c'est à dire utiliser de façon adaptive ces resources dans l'application selon des critères comme la langue du système de l'utilisateur, la région de cet utilisateur, ou quelqu'autre critère que ce soit.

Ci-dessous, on a un éditeur des ressources "chaînes de caractères" par rapport à la langue du systeme.

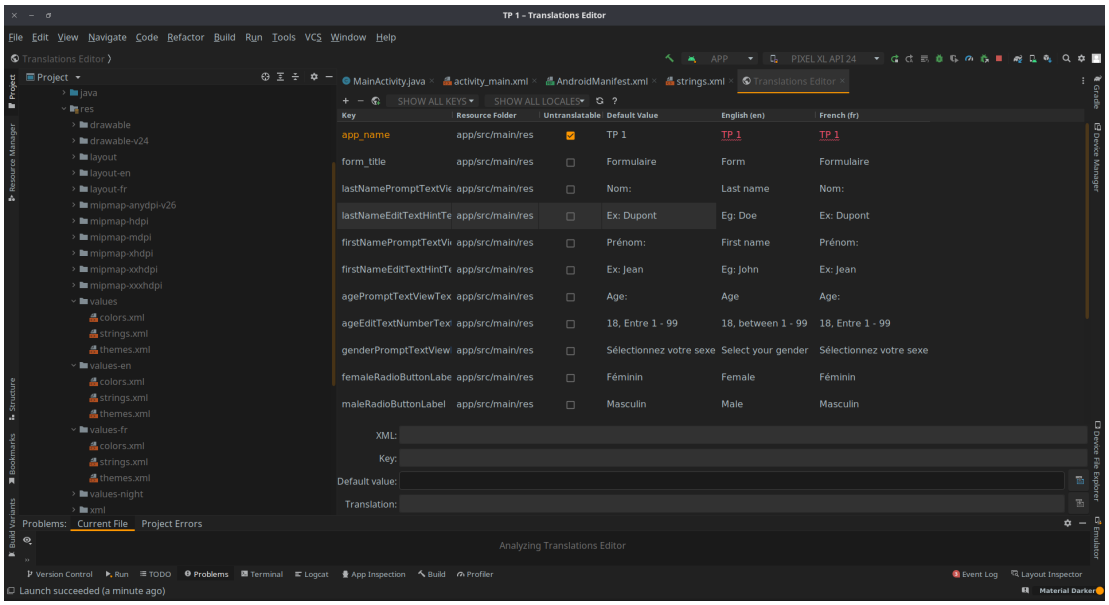

FIGURE 7 – Internalisation des valeurs string

En adaptant aussi le thème des vues selon la langue, nous obtenons ces différentes vues :

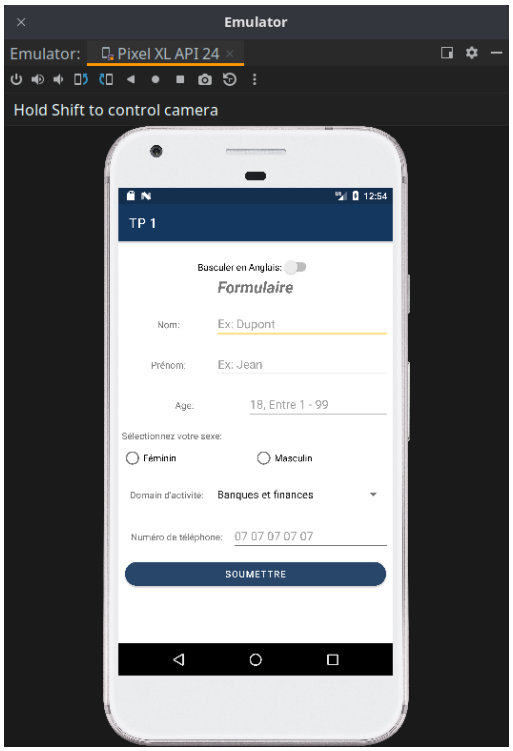

FIGURE 8 – Formulaire en francais

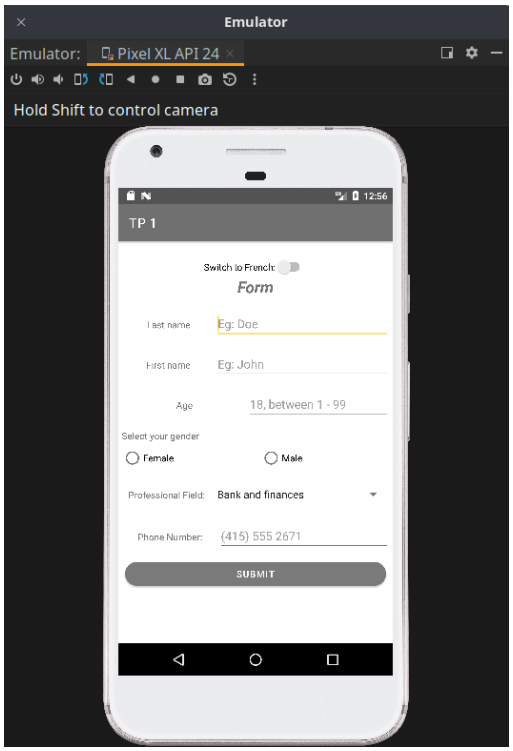

FIGURE 9 – Formulaire en Anglais

A gauche, nous avons l'exécution de la même application lorsque la langue du système est en français, et à droite lorsque la langue du système est changée en Anglais. Par défaut, sur le système android, on a au minimum une langue installée. Pour ajouter une nouvelle langue au systeme, dans les paramètres du système :
Paramètres supplémentaires (ou Langues et saisie) ⟩Langue et Region (Langue) ⟩Ajouter une langue.

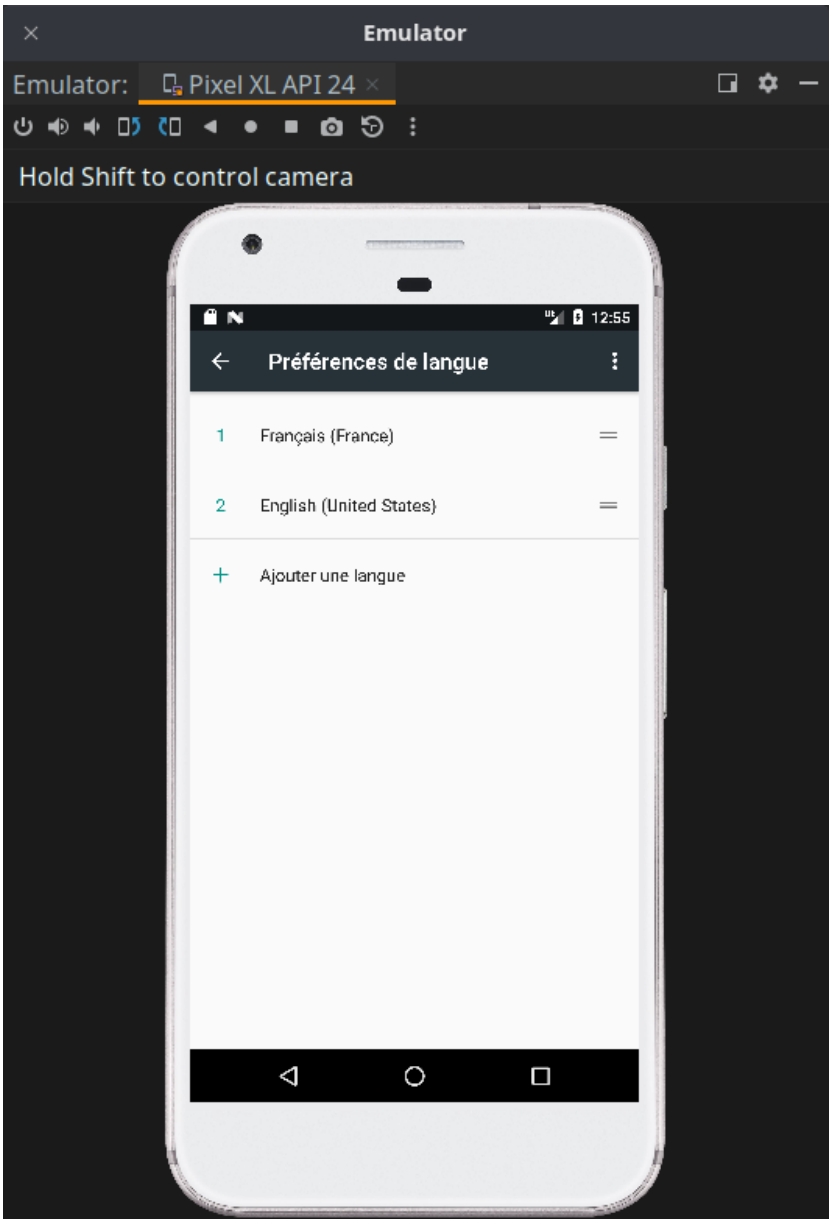

FIGURE 10 – Langues ajoutées du système

6 Événements associés aux objets graphiques d'une vue

En reprenant une copie de l'application realisée jusqu'à la question précédente, on peut ajouter des écouteurs d'évènements à des éléments de la vue pour produire d'autres actions dans le cycle de vie de notre application. Ici, un écouteur des évènements de clics est associé au boutton de soumission. Lorsque, l'utilisateur de l'application clique sur ce boutton, une boîte de dialogue superposée à la vue du formulaire s'ouvre, lui demandant de confirmer ou non les informations. Sur confirmation, la couleur des champs de saisie est changée comme inddiqué sur Fig : Couleur changee des EditText sur options YES .

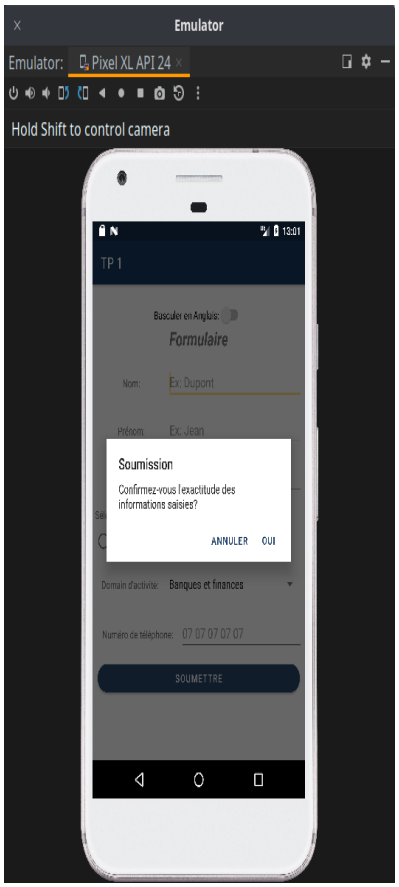

FIGURE 11 – Boîte de dialogue Alerte

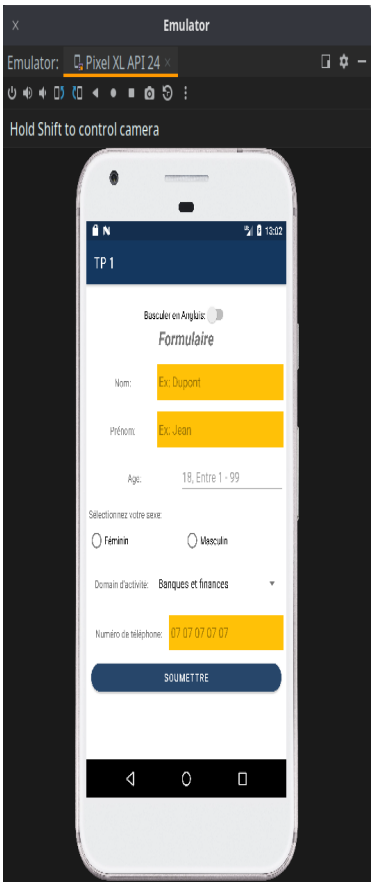

FIGURE 12 – Couleur changee des EditText sur options YES

7 Intent explicite

Les intents permettent de faire communiquer les différents composants d'une application entre eux, ou des applications entre elles. En ameliorant l'application précédante, 2 nouvelles activités ont été ajoutées avec leurs vues correspondantes.

Dans la première activité, lorsque les informations saisies ont été bien saisies et soumises par l'utilisateur, on passe à une activité intermediaire qui demande de confirmer le justesse des informations entrées. Sur validation, l'utilisateur passe à la seconde activité, et sur annulation, il revient à la vue de la première activité.

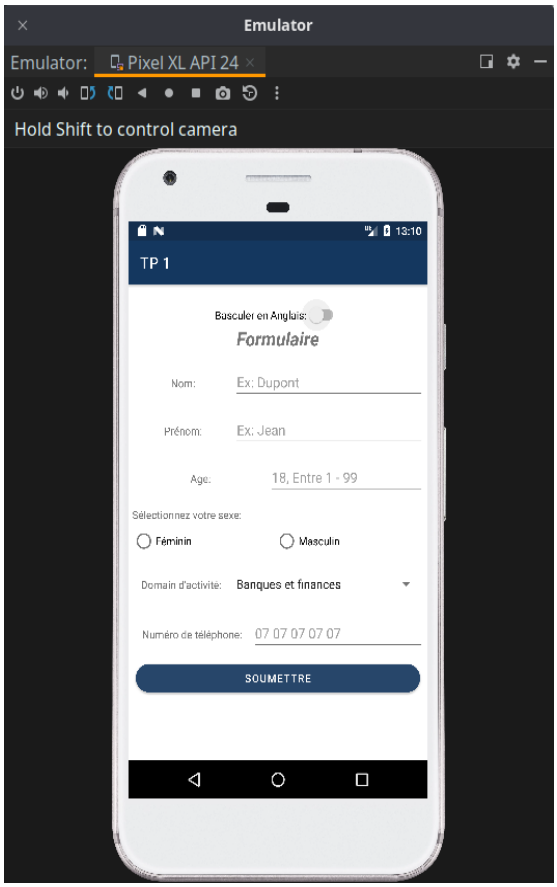

FIGURE 13 – Activité 1

Cette activité intermediaire remplace la boîte de dialogue présente dans l'exercice précédente.

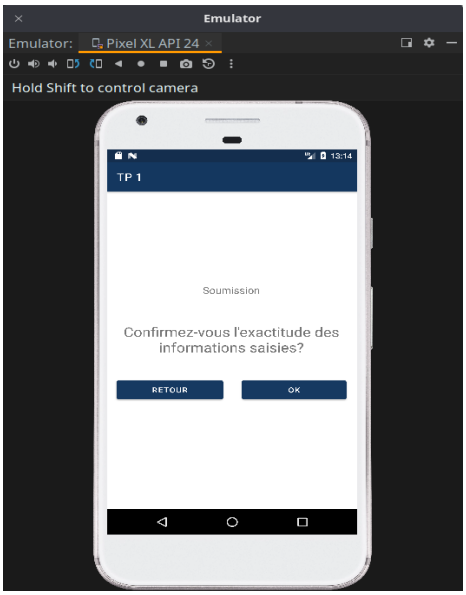

FIGURE 14 – Boîte de dialogue plein-écran personnalisée

Dans cette derniere activité, on récupère les informations saisies et les affiche.

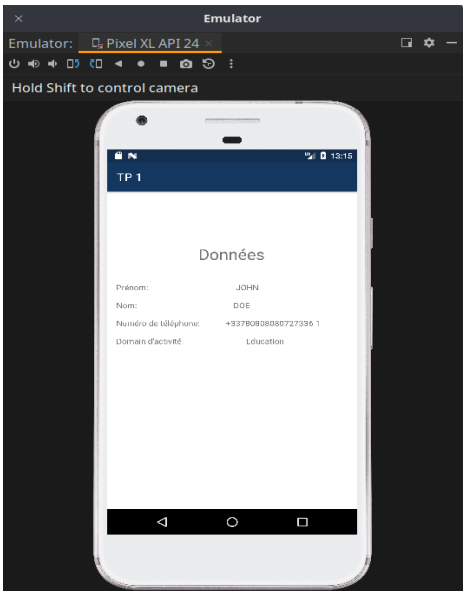

FIGURE 15 – Activité 2

8 Intents implicites

Dans cette partie, la vue de la dernière est ameliorée en y une image de téléphone et boutton permettant de passer un appel téléphonique, en laissant au système de choisir l'application la plus adaptée pour effectuer les appels téléphoniques et ceci via une intent implicite.

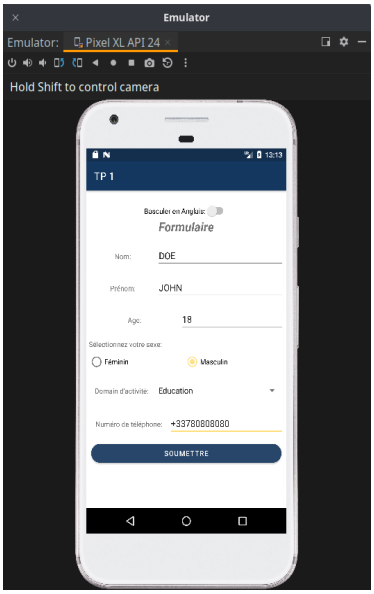

FIGURE 16 – Ecran 1

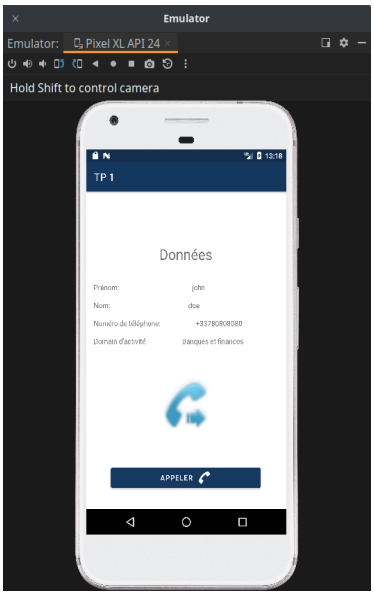

FIGURE 17 – Ajout d'image de téléphone

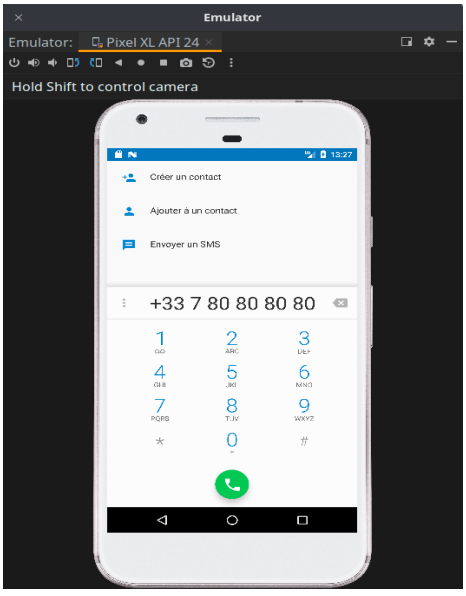

FIGURE 18 – Application Téléphone externe

9 Application simple pour consulter les horaires de trains

L'objectif de cette est de développer une application mobile android permettant à l'utilisateur de saisir le nom d'une ville de départ et celui d'une ville d'arrivée pour consulter les prochains horaires disponibles de train. Dans cette, perspective, pour obtenir les

données horaires des trains, la solution se repose sur l'utilisation de l'API (Application Programming Interface) de la SNCF. La gestion de certains execeptions à l'execution ne sont pas gérer dans cette solution, comme par exemple un un nom de ville nom reconnue par l'API, vu que la validation des donnees n'est pas considérée dans cette solution. Ceci peut eventuellement amener l'application a se planter. Du fait que la

gestion des exceptions ne soit l'objectif de cette question, on suppose que les donnée saisies sont toujours correctes.

Ci-dessous, on a la vue du mini formulaire de renseignement d'itinéraire.

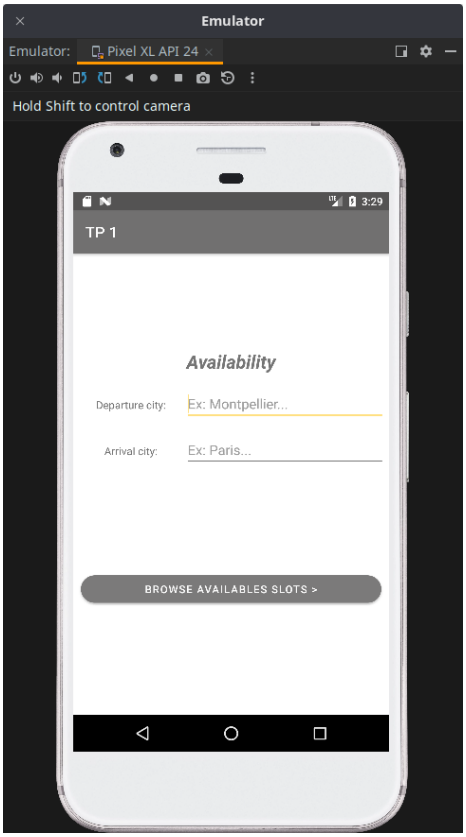

FIGURE 19 – Ecran de recherche d'horaire

Après que l'utilisateur aie renseigné son itinéraire et aie soumis sa demande, une

vue de traitement permet de notifier è l'utilisateur que sa demande est en cours de traitement. La tâche de requettage des données via l'API a été déléguée à un composant

de travail s'executant en arrière plan de la présentation des vues. Ce concept n'ayant pas été abordé dans le cours précédant, nous ferons abstration de son fonctionnement.

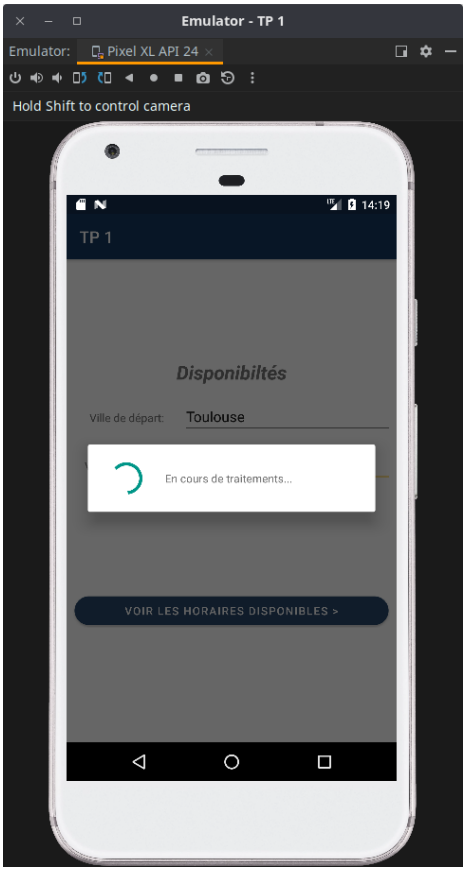

FIGURE 20 – Ecran d'attente

Après le traitement de la requête et récupération des données, les résultats sont affichés dans une vue ListView.

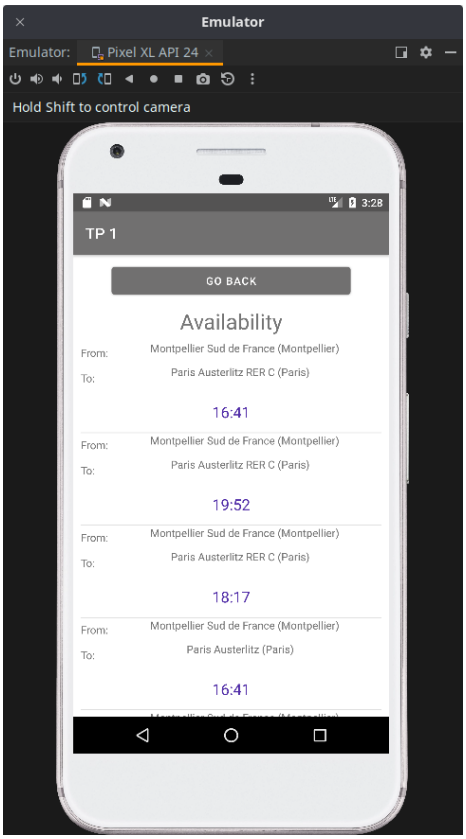

FIGURE 21 – Liste des disponibilités

L'utilisateur dispose d'un boutton retour lui permettant de revenir a l'écran du min-formulaire.

10 Application simple d'agenda

Dans cette dernièrre partie de ce TP, il nous est demandé de créer une application permettant de visualiser les évènments associés a une date, et d'ajouter de évènements. L'interface graphique est constituée d'une vue du calendrier et d'une listes des dates des évènements crées par l'utilisateur.

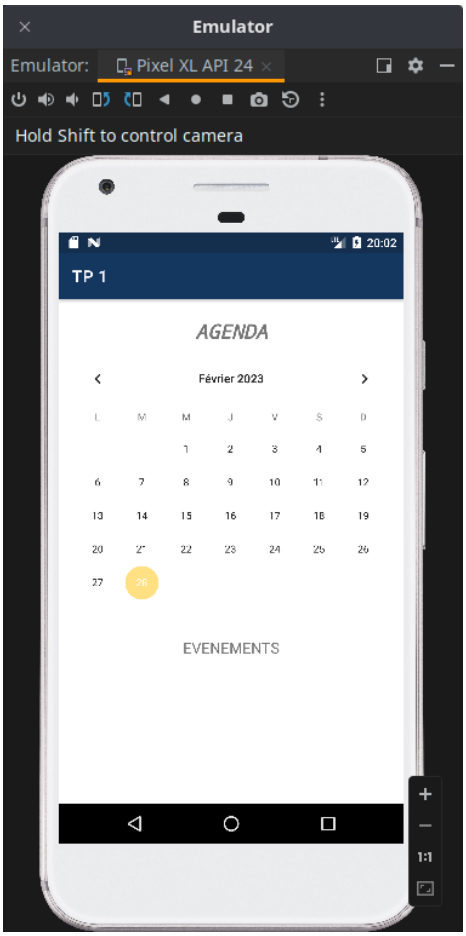

FIGURE 22 – Liste vide d'évènements

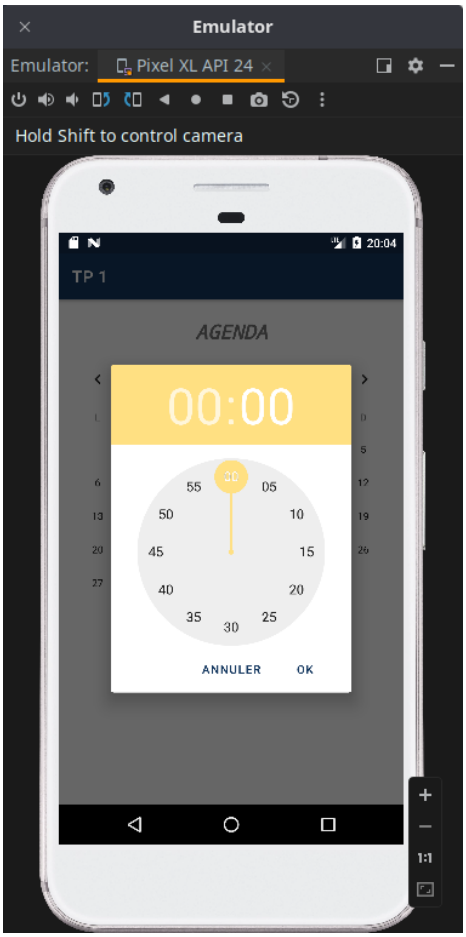

FIGURE 23 – Choix d'heure

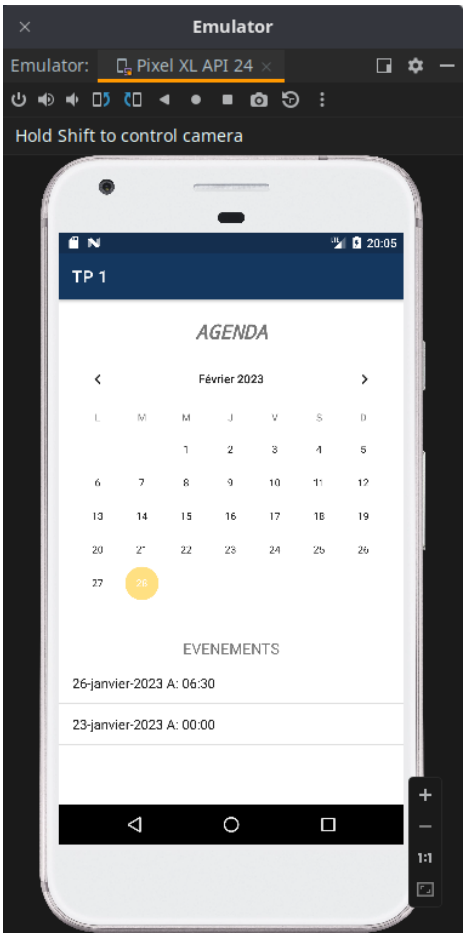

FIGURE 24 – Quelques évènements

11 ANNEXES et Réferences

1. Lien du Repo : https://github.com/007riche/master-submission/tree/main/SEM%202/HAI811/TP%201
2. developer.android.com/guide/topics/resources/localization?hl=fr#resource-switching
3. developer.android.com/training/basics/supporting-devices/languages?hl=fr#java
4. code.tutsplus.com/tutorials/android-from-scratch-using-rest-apis--cms-27117
5. doc.navitia.io/
6. https://playground.navitia.io/index
7. https://developer.android.com/reference/android/widget/TimePicker