



# HAI811I - PROGRAMMATION MOBILE

TP1



16 FEVRIER 2025  
NADIM DOUHANE  
Université de Montpellier

# TP1 : Les Bases d'Android



**UNIVERSITÉ DE  
MONTPELLIER**

Ce TP est un projet réalisé dans le cadre d'un cours de programmation Mobile sur le logiciel Android Studios. Ce projet contient 3 applications suivant les 9 exercices du TP.



## Table des matières

Application 1 (exercices 3,4,5,6,7) .....	4
Application 2 (exercice 8) .....	7
Application 3 (exercice 9) .....	12
Lancement des applications : .....	18
Liens de vidéos de test d'application .....	19

Université de Montpellier



## HA1811I - Programmation Mobile

### TP1 : Les Bases d'Android

#### Exercice 1 : Découverte de l'environnement de développement

---

- Développement Android
  - o <http://developer.android.com/>
- Android Studio, choix des cibles AVD (Virtual device), SDK
  - o <https://developer.android.com/studio/intro>

#### Exercice 2 : hello world

---

- Exécuter (ou développer) l'application « hello world »
- Inspecter les différents éléments de cette application : les composants, le fichier manifest, les ressources, le répertoire bin, etc.

#### Exercice 3 : Une première application- Interface simple

---

- Créer une application demandant à son utilisateur de remplir les champs nom, prénom, âge, domaine de compétences et numéro de téléphone et de valider ces informations en appuyant sur un bouton.
- Créer cette application en créant entièrement la vue (interface) en XML.
- Créer cette application en créant entièrement la vue (interface) dans le code java.
- Enrichir cette application en mentionnant avant chaque champ de saisie le label de ce champ : nom, prénom, ...

#### Exercice 4 : Internationalisation des interfaces

---

- Reprendre l'exercice précédent pour créer une version en Anglais (en plus de celle en français déjà existante).

#### Exercice 5 : Événements associés aux objets graphiques d'une vue

---

- Reprendre l'application développée dans les deux exercices précédents pour associer au « bouton de validation » un événement qui permet d'ouvrir une fenêtre de dialogue invitant l'utilisateur à confirmer ou à annuler la validation (ou/et faire autre chose telle que changer la couleur du fond des zones d'édition.).

**Exercice 6 : Intent explicite**

---

- Reprendre l'application précédente pour :
  - o Créer un intent permettant de récupérer toutes les informations des champs saisis et de lancer une nouvelle activité via cet Intent.
  - o Créer une activité qui est lancée par l'activité principale via l'Intent précédent. Cette activité récupère les données saisies, les affiche et affiche deux boutons. Le premier bouton « OK » lance une troisième activité dont vous êtes libre de définir son contenu (par exemple écran vide). Le deuxième bouton « Retour » permet de revenir à l'activité précédente.

**Exercice 7 : Intent implicite**

---

- Reprendre l'application précédente pour :
  - o Ajouter à la dernière activité créée (interface vide) une image de téléphone, le numéro de téléphone saisi et un bouton « Appeler ».
  - o L'activation de ce bouton permet de lancer un appel téléphonique vers ce numéro.

**Exercice 8 : Application simple pour consulter les horaires de trains**

---

- Développer uniquement l'interface graphique d'une application permettant à un utilisateur de saisir un itinéraire (villes Départ et Arrivée) et de visualiser tous les horaires de trains pour cet itinéraire sous forme de liste.

**Exercice 9 : Application simple d'agenda**

---

- Développer uniquement l'interface graphique d'une application permettant de réaliser un agenda (afficher les dates avec leurs événements associés, ajouter des événements à associer à une date, ...).

## Application 1 (exercices 3,4,5,6,7)

La première application représente une application d'Enregistrement de personnes contenant trois pages :

- La 1ère page est la page principale par laquelle l'utilisateur est accueilli, sur cette page se trouve un formulaire de création de personne.
- La 2ème page contient les informations renseignées dans la 1ère page par l'utilisateur ainsi que 2 boutons, un bouton Retour et un bouton Enregistrer qui va permettre d'envoyer l'utilisateur vers la dernière page.
- La 3ème page contient une icône de téléphone avec un numéro et un bouton permettant d'ouvrir l'application d'appel avec le téléphone.

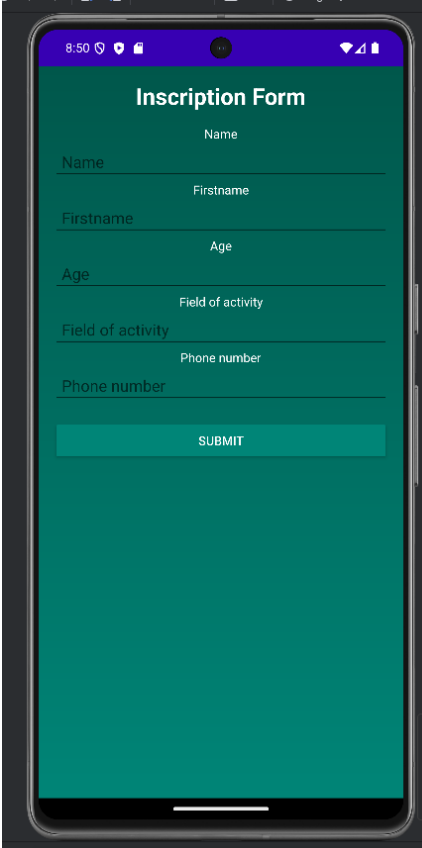
Lorsque l'utilisateur ouvre l'application, l'utilisateur se trouve sur la page 1 (voir [figure 1](#)) il a la possibilité de renseigner des informations, une fois cette étape réalisée, il a la possibilité de cliquer un bouton Enregistrer de là, une fenêtre de dialogue apparaît grâce à laquelle il a la possibilité de confirmer l'enregistrement des données. Si l'utilisateur clique sur le bouton annuler, il reste sur la première page et la fenêtre de dialogue disparaît. Dans le cas où il cliquerait sur confirmer, il est envoyé sur la deuxième page (voir [figure 3](#)) où il peut cliquer sur un bouton retour via lequel il sera renvoyé sur la première page. L'utilisateur peut aussi cliquer un bouton "Ok" qui enverra vers la 3ème page (voir [figure 4](#)) dans laquelle il aura la possibilité de cliquer sur un bouton Appelle qui ouvrira l'application appel avec le numéro de la personne inscrit pour appel. Comme demandé en consigne, les textes de l'application sont disponibles en langue anglaise (voir [figure 2](#)).

The image shows a smartphone screen displaying a registration form titled "Formulaire d'inscription". The form has a teal background and white text. It contains the following fields and values:

- Nom: Macron
- Prénom: Emmanuel
- Âge: 45
- Domaine de compétences: Presidence
- Numéros de téléphones: 06 78 96 64 10

At the bottom of the form is a button labeled "VALIDER". The smartphone's status bar at the top shows the time 20:47 and various icons. The bottom of the screen shows a navigation bar with a back arrow and a home indicator.

Figure 1- HomePage\_FR



**Inscription Form**

Name

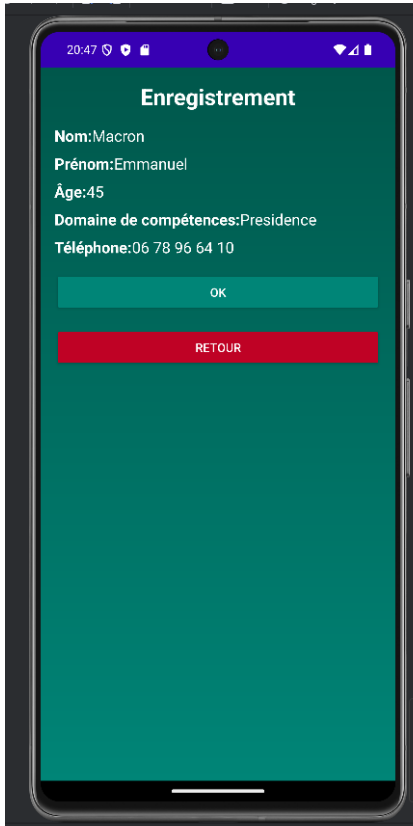
Firstname

Age

Field of activity

Phone number

SUBMIT

*Figure 2 - HomePage\_EN*

**Enregistrement**

Nom:Macron

Prénom:Emmanuel

Âge:45

Domaine de compétences:Presidence

Téléphone:06 78 96 64 10

OK

RETOUR

*Figure 3 - RegisterPage*

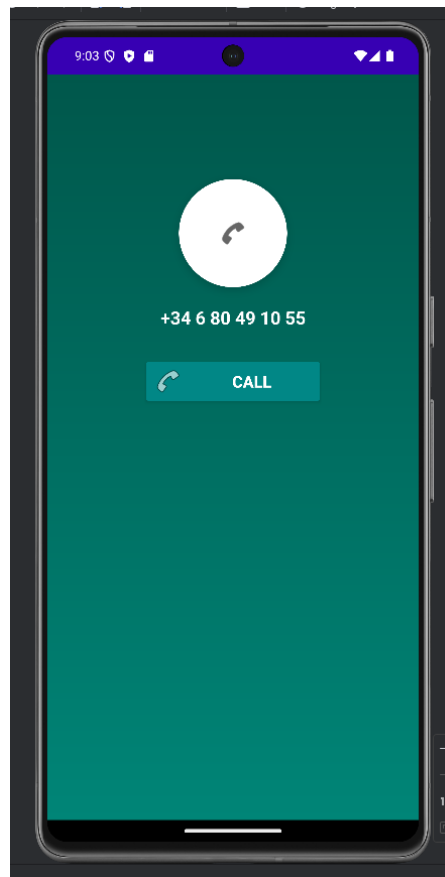


Figure 4 – CallPage

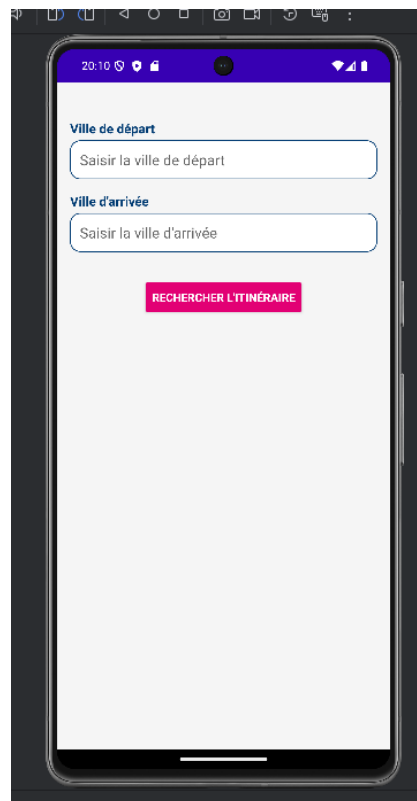
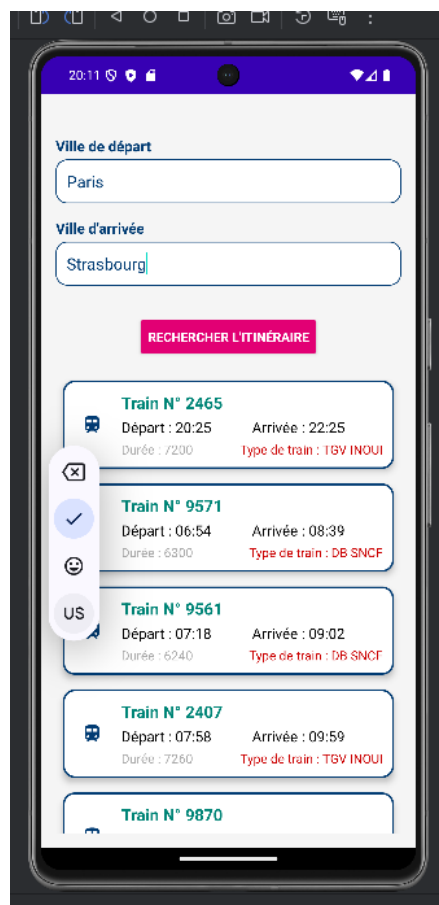
## Application 2 (exercice 8)

La deuxième application représente une application de visualisation d'Horaire de train entre deux destinations, cette application contient une page avec :

- Un édit text pour renseigner une ville de départ
- Un édit text pour renseigner une ville d'arrivé
- Une liste d'horaires de train

Lorsque l'utilisateur arrive sur l'application, il a la possibilité de renseigner des villes de départ et d'arrivé (voir [figure 5](#)), s'il ne renseigne pas l'une des villes, il est informé via un toast qu'il manque une ville. De plus, l'application nécessite une connexion internet, car afin de récupérer les informations de train, il y a une requête auprès de l'API SNCF. Une fois avoir renseigné les deux villes et avoir cliqué sur le bouton la liste des horaires s'affichent en dessous du bouton (voir [figure 6](#)). L'API est accessible via un le fichier **SNCFClient.java**



*Figure 5 - HorairePage**Figure 6 - HorairePage avec liste*

Code java d'accès et de manipulation à l'API SNCF :

```
package com.example.tp1_ex8;
import okhttp3.*;
import org.json.JSONObject;
import java.io.IOException;
import android.util.Log;

public class SNCFClient {
    private static final String API_KEY = "0503def7-c65a-425b-be42-261595735c89";
    private static final String BASE_URL =
        "https://api.sncf.com/v1/coverage/sncf/journeys";

    public interface ApiResponseListener {
        void onSuccess(JSONObject response);
        void onError(String error);
    }

    public static void rechercherItineraire(String depart, String arrivee,
        ApiResponseListener listener) {
        OkHttpClient client = new OkHttpClient();

        rechercherCodeVille(depart, new ApiResponseListener() {
            @Override
            public void onSuccess(JSONObject response) {
                String departCode = extraireCodeVille(response);

                rechercherCodeVille(arrivee, new ApiResponseListener() {
                    @Override
                    public void onSuccess(JSONObject response) {
                        String arriveeCode = extraireCodeVille(response);

                        HttpUrl url = HttpUrl.parse(BASE_URL).newBuilder()
                            .addQueryParameter("from", departCode)
                            .addQueryParameter("to", arriveeCode)
                            .addQueryParameter("count", "10")
                            .build();

                        Log.d("SNCFClient", "URL générée : " + url.toString());

                        Request request = new Request.Builder()
                            .url(url)
```

```
.addHeader("Authorization", API_KEY)
.build();

client.newCall(request).enqueue(new Callback() {
    @Override
    public void onFailure(Call call, IOException e) {
        listener.onError("Erreur réseau : " + e.getMessage());
    }

    @Override
    public void onResponse(Call call, Response response) throws
IOException {
        if (response.isSuccessful()) {
            try {
                JSONObject jsonResponse = new
JSONObject(response.body().string());
                listener.onSuccess(jsonResponse);
            } catch (Exception e) {
                listener.onError("Erreur de parsing JSON : " + e.getMessage());
            }
        } else {
            String errorMessage = "Erreur API : " + response.message();
            listener.onError(errorMessage);
        }
    }
});

@Override
public void onError(String error) {
    listener.onError("Erreur pour la ville d'arrivée : " + error);
}

@Override
public void onError(String error) {
    listener.onError("Erreur pour la ville de départ : " + error);
}
});
}
```

```
private static String extraireCodeVille(JSONObject response) {
    return response.optJSONArray("places").optJSONObject(0).optString("id");
}

public static void rechercherCodeVille(String ville, ApiResponseListener listener) {
    OkHttpClient client = new OkHttpClient();

    // URL de recherche de villes
    HttpUrl url =
    HttpUrl.parse("https://api.sncf.com/v1/coverage/sncf/places").newBuilder()
        .addQueryParameter("q", ville) // Recherche par nom de la ville
        .build();

    Request request = new Request.Builder()
        .url(url)
        .addHeader("Authorization", API_KEY)
        .build();

    // Exécution de la requête
    client.newCall(request).enqueue(new Callback() {
        @Override
        public void onFailure(Call call, IOException e) {
            listener.onError("Erreur réseau : " + e.getMessage());
        }

        @Override
        public void onResponse(Call call, Response response) throws IOException {
            if (response.isSuccessful()) {
                try {
                    JSONObject jsonResponse = new JSONObject(response.body().string());
                    listener.onSuccess(jsonResponse);
                } catch (Exception e) {
                    listener.onError("Erreur de parsing JSON : " + e.getMessage());
                }
            } else {
                listener.onError("Erreur API : " + response.message());
            }
        }
    });
}
```

```
}
```

Modification des dépendances pour du **build.gradle (Module : app)** pour l'accès au requête internet :

```
dependencies {  
  
    implementation libs.appcompat  
    implementation libs.material  
    testImplementation libs.junit  
    androidTestImplementation libs.ext.junit  
    androidTestImplementation libs.espresso.core  
    //implementation 'com.squareup.okhttp3:okhttp:4.9.3'  
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'  
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'  
    implementation 'com.squareup.okhttp3:okhttp:4.9.1'  
  
}
```

### Application 3 (exercice 9)

La troisième application représente une application Agenda dans laquelle on peut créer des événements (voir **figure 7**). Cette application contient une page avec :

- Un CalendarView via lequel on peut parcourir les dates
- Un bouton pour ajouter un événement
- Une liste d'évènement

Lorsque l'utilisateur arrive sur l'application, il a la possibilité de créer un événement en cliquant sur un bouton dès lors un popup s'ouvre dans lequel l'utilisateur renseigne les informations sur les événements (voir **figure 8**). Une fois créé l'évènement s'affiche en dessous du bouton si l'évènement a été créé au jour cliqué sur le calendrier (voir **figure 9**). L'utilisateur peut voir les infos de l'évènement en cliquant sur l'évènement de la liste

(voir **figure 10**). De plus l'utilisateur a la possibilité de supprimer l'évènement en effectuant un glissement vers la droite (*Swipe And Delete*).

Code Java pour le Swipe-And-Delete :

```
ItemTouchHelper itemTouchHelper = new ItemTouchHelper(new
ItemTouchHelper.SimpleCallback(0, ItemTouchHelper.RIGHT) {
    @Override
    public boolean onMove(RecyclerView recyclerView, RecyclerView.ViewHolder
viewHolder, RecyclerView.ViewHolder target) {
        return false;
    }

    @Override
    public void onSwiped(RecyclerView.ViewHolder viewHolder, int direction) {
        int position = viewHolder.getAdapterPosition();
        Evenement eventToRemove = evenementDuJourList.get(position);
        evenementList.remove(eventToRemove);
        evenementDuJourList.remove(position);
        evenementAdapter.notifyItemRemoved(position);
    }
});

itemTouchHelper.attachToRecyclerView(recyclerView);
```

Méthode pour sauvegarder les évènements :

```
private void saveEventsToSharedPreferences(List<Evenement> evenementList) {
    SharedPreferences sharedPreferences = getSharedPreferences("AgendaPrefs",
MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    Gson gson = new Gson();
    String json = gson.toJson(evenementList);
    editor.putString("events", json);
    editor.apply();
}
```

```

private List<Evenement> getEventsFromSharedPreferences() {
    SharedPreferences sharedPreferences = getSharedPreferences("AgendaPrefs",
MODE_PRIVATE);
    Gson gson = new Gson();
    String json = sharedPreferences.getString("events", null);
    if (json == null) {
        return new ArrayList<>();
    }
    Type type = new TypeToken<List<Evenement>>().getType();
    return gson.fromJson(json, type);
}

```



Figure 7 - HomePage\_Calendrier

Appels d'urgen... 96 % 23:39

< Février 2025 >

Faire le TP de dev mobile

15/2/2025

08:00 09:00

Avoir une bonne note.

ANNULER VALIDER

Dormir  
15/2/2025

?123 , 😊 . ↩

Figure 8 - Popup \_ création évènement





Figure 9 - HomePage avec évènement créé



Figure 10 - Popup avec info de l'évènement

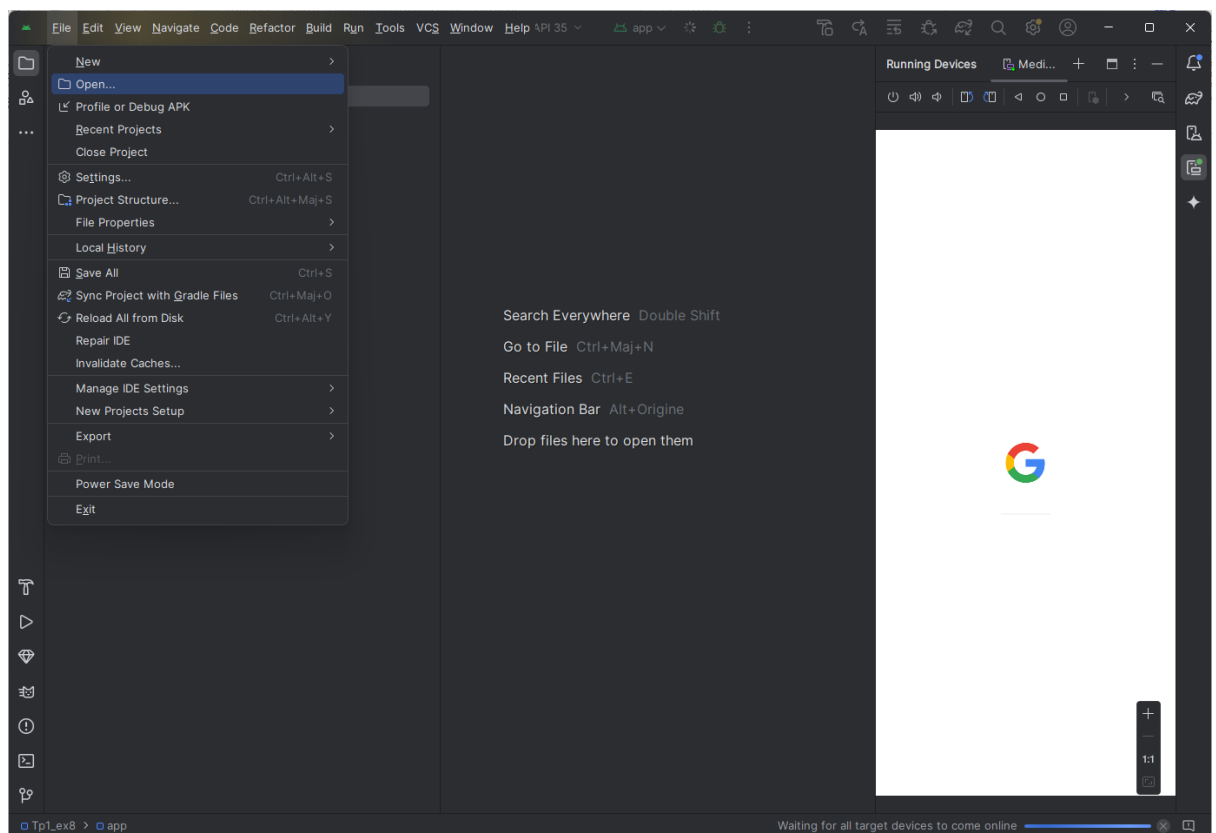
Modification des dépendances pour du **build.gradle (Module : app)** pour sauvegarder les évènements même si l'utilisateur quitte l'application via SharedReference :

```
dependencies {

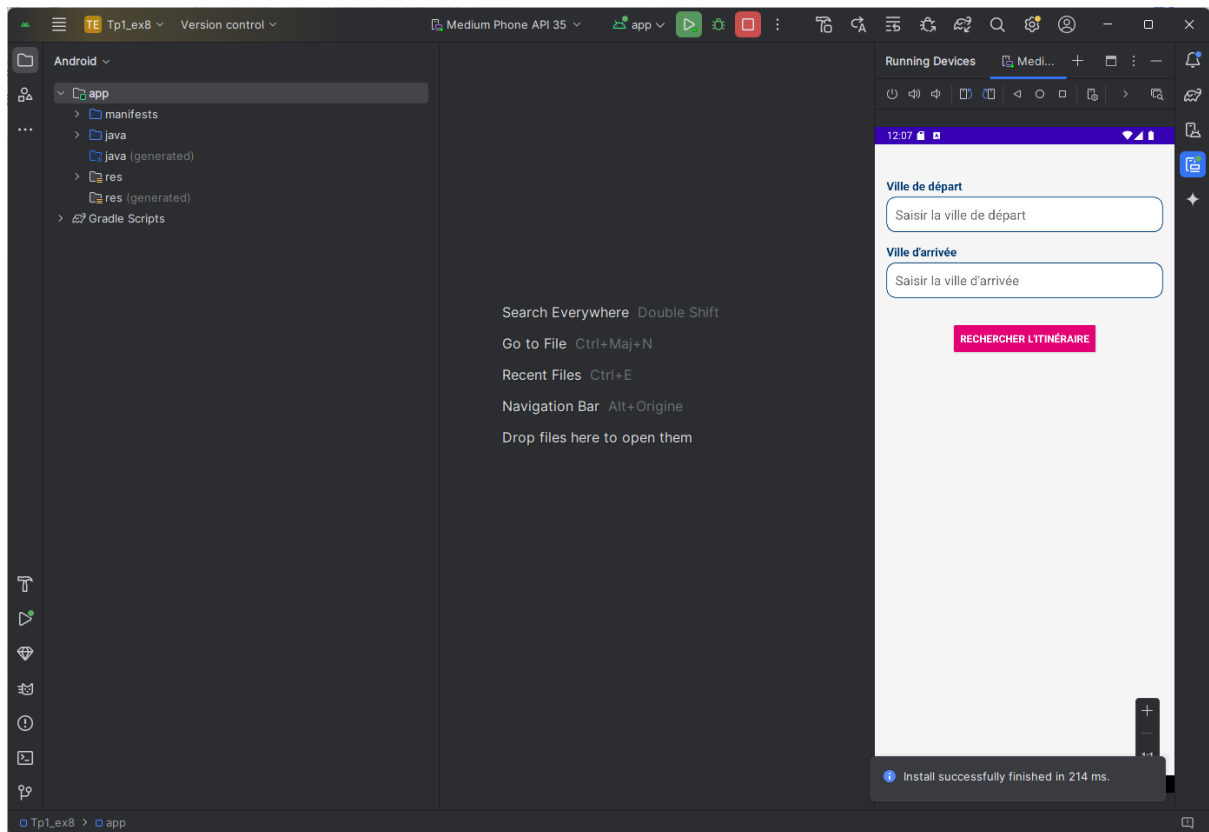
    implementation libs.appcompat
    implementation libs.material
    testImplementation libs.junit
    androidTestImplementation libs.ext.junit
    androidTestImplementation libs.espresso.core
    implementation 'com.google.code.gson:gson:2.8.8'
}
```

## Lancement des applications :

1. Télécharger les applications Zippé (***Tp1\_prog\_mobile\_app.zip***, ***Tp1\_ex8.zip***, ***Tp\_ex9.zip***)
2. Dézipper les applications
3. Lancer l'application ***Android Studios***
4. Ouvrir les applications (voir photo ci-dessous) :



5. Choisir l'application que vous voulez lancer
6. Cliquer sur le bouton ***Run 'app'***
7. Tester l'application sur le téléphone virtuel ou bien sur votre propre téléphone Android en mode développeur.



## Liens de vidéos de test d'application

Vidéo test application 1 :

<https://youtube.com/shorts/x6gAZWfPv-E>

Vidéo test application 2 :

<https://youtube.com/shorts/zp78x-yWHzU>

Vidéo test application 3 :

<https://youtu.be/2qrapHz1qPl>