**T.P. 3: Interactions avec les composants logicielles et matériels**

# Préambule : Projet AMIO

L'objectif de ce mini-projet et de vous faire finaliser une application Android exploitant des données issues d'un réseau de capteurs et exposées à travers un web service (IoTLab de TELECOM Nancy). Le but est de détecter les lumières laissées actives dans les bureaux en soirée. L'application doit permettre les actions suivantes:

- lister dans l'activité principale les capteurs actifs et les valeurs de luminosité qu'ils relèvent, en mettant en évidence ceux qui indiquent la présence d'une lumière active
- émettre une notification si une nouvelle lumière vient d'être allumée en semaine entre 19h et 23h, en spécifiant le capteur impliqué
- envoyer un email si cet événement survient le week-end entre 19h et 23h ou en semaine entre 23h et 6h.
- permettre la configuration des plages horaires et de l'adresse email dans un menu dédié

Le projet est à réaliser en binôme et à rendre pour le 05/12/22. Vous devez fournir sur Arche une archive contenant les sources, le .apk et un fichier texte décrivant les fonctionnalités de votre application. Les critères d'évaluation tiendront compte du respect des fonctionnalités attendues, de l'ergonomie de l'application, de la propreté du code, ou encore d'autres fonctionnalités que vous auriez implantées.

Outre la prise en main du SDK Android et de l'IDE Android Studio, nous avons déjà vu, durant les précédentes séances, un certain nombre d'éléments de l'API qui doivent vous aider dans la conception de cette application : la réalisation d'une **Activity** avec des éléments d'UI et l'exécution d'une tâche asynchrone **AsyncTask**, la réalisation d'un **Service** avec exécution périodique **TimerTask**, la consultation d'un web service avec **HttpURLConnection** et la lecture des réponses avec **JsonReader**, la persistance des données avec **SharedPreferences**, la création d'un **BroadcastReceiver** pour gérer les événements système, l'édition du **Manifest** pour déclarer des **permissions** et des **intent-filter**, et enfin, l'envoi de **Notification**.

Ce sujet se veut plus court que les précédents TP afin de vous laisser le temps de les terminer. Nous allons introduire les derniers éléments nécessaires à l'application à savoir, l'envoi d'email par composition de service et la réalisation d'un menu pour éditer les préférences.

# Exercice 1 : Envoi d'email par composition de service

Cet exercice a pour but de démontrer, à travers l'envoi d'email, comment utiliser d'autres applications ou composants logiciels d'Android pour réaliser une action spécifique.

- Question 1 – Communication IPC (Intent)
  - lorsqu'un changement a été détecté, créer une Intent dont l'action doit être ACTION_SEND
  - note: pour éviter toute ambiguïté, il est bon de spécifier la nature et le type de données qui vont être prises en charges via l'Intent créé :

```
emailIntent.setData(Uri.parse("mailto:"));
emailIntent.setType("text/plain");
```

  - rajouter à cet Intent les paramètres additionnels nécessaires afin de spécifier l'objet, le corps et le destinataire du mail (choisir par exemple une adresse personnelle) .
    - note : ces paramètres sont écrits en dur dans le code afin de simplifier l'exercice ; il est toutefois possible de les rentrer manuellement avant chaque envoi d'email

- une fois que l'Intent est correctement créé et renseignée, l'envoyer en Broadcast système quand un changement est détecté et observer ce qui se passe:

```
startActivity(Intent.createChooser(emailIntent,"Send mail..."));
```

## Exercice 2 : Menu des préférences

Les applications ont souvent besoin d'être paramétrées suivant les préférences de l'utilisateur. Android permet de réaliser facilement cette configuration via des classes spécifiques : http://developer.android.com/guide/topics/ui/settings.html

- Question 1 – Hériter de PreferenceActivity
  - le plus simple (bien que déprécié ;-( ) consiste à créer une nouvelle activité blanche « Settings » qui hérite de PreferenceActivity
  - ajouter un appel à addPreferencesFromResource(R.xml.preferences); dans onCreate()

- Question 2 – Preferences XML
  - ajouter un répertoire « xml » au répertoire « res », puis un nouveau fichier nommé « preferences.xml » dans /res/xml .

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
    <EditTextPreference android:title="Your Name"
        android:key="username"
        android:summary="Please provide your username"></EditTextPreference>
</PreferenceScreen>
```

- Question 3 – Lier l'activité « Settings » à l'activité principale
  - dans l'action Bar, lancer l'activité des Préférences quand le bouton « action_settings » est appuyé
    - créer un Intent à destination de Settings.class et appeler startActivity(Intent)

## Exercice 3 (facultatif): Action sur composant matériel

Cet exercice introduit brièvement les interactions avec les composants cette fois matériel du smartphone (ex : camera, vibreur, GPS, etc.).

- Question 1 – Permission de vibrer
  - ajouter dans AndroidManifest.xml la permission nécessaire pour que l'application puisse utiliser le vibreur :

```
<uses-permission android:name="android.permission.VIBRATE" />
```

- Question 2 – Récupération de l'instance « Vibreur »
  - de la même façon qu'il est parfois nécessaire de retrouver les composants graphiques d'une activité avant de pouvoir s'en servir, il est impératif de récupérer l'instance de type « Vibreur » qui est propre au système ;

```
Vibrator v = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
```

- Question 3 – Vibrations
  - rajouter maintenant la ou les lignes de codes permettant de faire vibrer le téléphone de façon constante pendant une durée déterminée (e.g. 500 millisecondes) quand une nouvelle lumière est détectée