UNIVERSITÉ DE MONTPELLIER

module : App mobile

TP 2 Developpement Mobile avec Android

Réalisé par :

M lle MEKHNACHE Toudherth – GL

Référent :

M r SERIAI ABDELHEK Djamal

M r BACHAR Rima

Promotion 2022/2023

0.1 Introduction

Dans le but de se familiariser avec java pour android. on a implementé un nombre d'applications mobile demandé durant le TP1. Premiere application est sur un formulaire avec Internationalisation de l'interface et l'ancement d'appel. La deuxieme application pour consulter les horaires de trains, et finalement une application simple d'agenda.

0.2 Android Studio

Android Studio est un environnement de développement intégré (IDE) utilisé pour créer des applications Android en utilisant le langage de programmation Java ou Kotlin. Cet IDE fournit un ensemble complet d'outils de développement, notamment un éditeur de code, un débogueur, un émulateur Android pour tester les applications, ainsi qu'un large éventail de bibliothèques et d'API Android pour faciliter le développement d'applications mobiles.

0.3 JAVA pour Android

Java est un langage de programmation de haut niveau orienté objet utilisé dans le développement d'applications Android. Il est connu pour sa portabilité, sa sécurité et sa facilité d'utilisation. La syntaxe de Java est similaire à celle des autres langages de programmation orientés objet tels que C++, ce qui facilite l'apprentissage pour les développeurs débutants. Java est utilisé pour développer des applications Android depuis la version initiale de la plate-forme Android, et il reste aujourd'hui l'un des langages de programmation les plus couramment utilisés pour le développement d'applications Android.

0.4 Les capteurs dans Android

Les capteurs sont des composants matériels des appareils mobiles qui peuvent détecter des mouvements et des conditions environnementales. Les smartphones Android sont équipés de plusieurs capteurs intégrés, tels que l'accéléromètre, le gyroscope, le magnétomètre, le capteur de proximité, le capteur de lumière ambiante, le capteur de pression, le capteur de température, le capteur de battements cardiaques et bien d'autres.

Dans Android Studio, les développeurs peuvent accéder à ces capteurs en utilisant les API de capteurs Android. Les développeurs peuvent utiliser les données des capteurs pour créer des applications qui peuvent suivre les mouvements de l'utilisateur, mesurer la vitesse de déplacement, l'orientation, la distance parcourue, la luminosité de l'environnement, etc. L'utilisation des capteurs peut être très utile dans une variété de contextes d'application, tels que les jeux, la santé et le bien-être, les applications de fitness, les applications de réalité augmentée, les applications de sécurité et bien d'autres. Les capteurs sont une fonctionnalité importante pour les développeurs Android, car ils offrent une multitude de possibilités pour améliorer l'expérience utilisateur dans les applications mobiles.

0.5 Realisation de l'application sur les capteurs en Android

0.5.1 Accueil de l'application capteur

L'interface utilisateur Accueil de l'application. cette interface contient plusieurs buttons pour différents capteurs utiliser dans un smartphone Android :

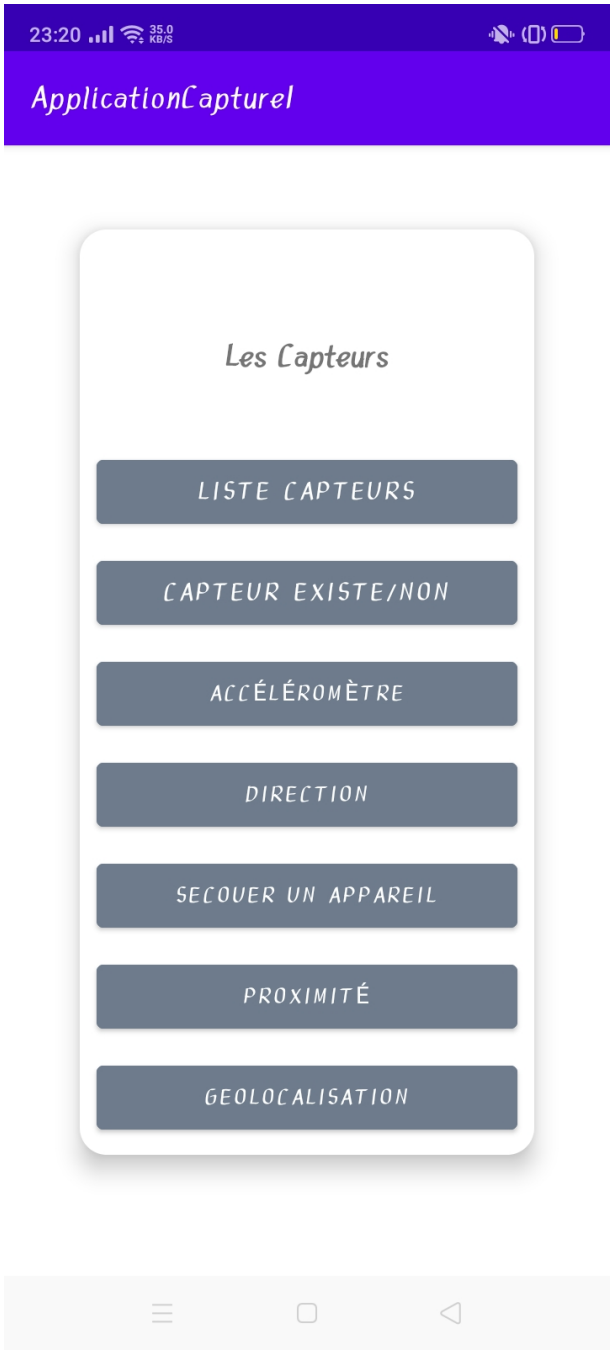

FIGURE 1 – Interface Accueil

0.5.2 Liste de capteurs

Cette interface returne la liste des capteurs existe dans un tels smartphone Android en details. Le code suivant permet de presenter comment recuperer la liste des capteurs dans un smartphone.

```
sensorManager= (SensorManager)  getSystemService(Context.
   SENSOR_SERVICE);

 //la presence  d'un capture
Sensor defaultProximitySensor = sensorManager.getDefaultSensor(
   Sensor.TYPE_PROXIMITY);

// Fonction pour la recuperation des capteurs avec leurs
   informations :
   private void listSensor() {
        List<Sensor> sensors = sensorManager.getSensorList(Sensor
           .TYPE_ALL);

        StringBuffer sensorDesc = new StringBuffer();
        for (Sensor sensor : sensors) {
            sensorDesc.append("New sensor detected : \r\n");
            sensorDesc.append("\tName: " + sensor.getName() + "\r
               \n");
            sensorDesc.append("\tType: "+ sensor.getType() +"/r/n
               ");
            sensorDesc.append("Version: " + sensor.getVersion() +
                "\r\n");
            sensorDesc.append("Resolution (in the sensor unit): "
                + sensor.getResolution() + "\r\n");
            sensorDesc.append("Power in mA used by this sensor
               while in use" + sensor.getPower() +"\r\n");
            sensorDesc.append("Vendor: " + sensor.getVendor() + "
               \r\n");
            sensorDesc.append("Maximum range of the sensor in the
                sensor's unit." + sensor.getMaximumRange() + "\r\n
               ");
            sensorDesc.append("Minimum delay allowed between two
               events in microsecond    +or zero if this sensor
               only returns a value when the data it's measuring
               changes    "+ sensor.getMinDelay() + "\r\n");
        }
        Toast.makeText(this, sensorDesc.toString(), Toast.
           LENGTH_LONG).show();
    }
```

Voici l'interface qui returne les capteurs existe dans un smartphone Oppo.

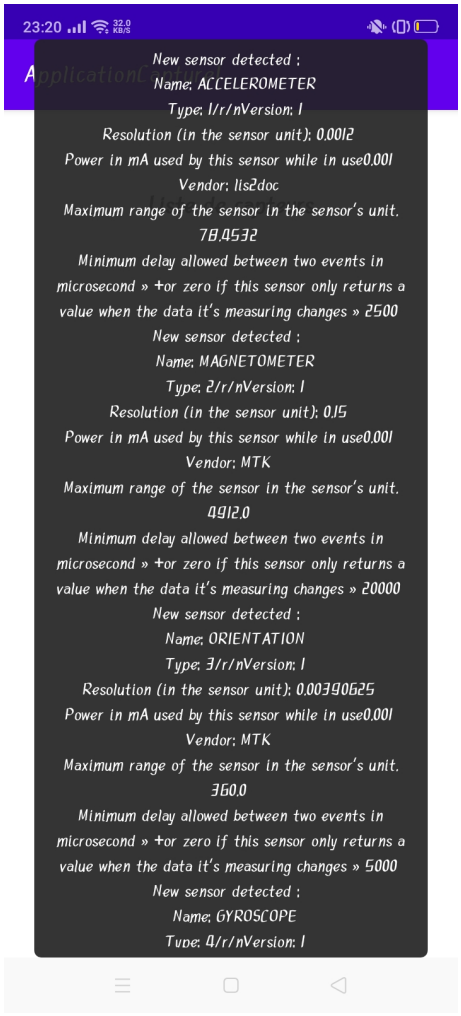

FIGURE 2 – Liste de capteurs

0.5.3 Détection de présence/absence de capteur de luminosité

La détection de présence ou d'absence de capteurs sur un dispositif peut être réalisée en utilisant l'API SensorManager dans le développement d'applications Android. L'API SensorManager fournit des méthodes pour accéder aux capteurs disponibles sur un dispositif, pour enregistrer des écouteurs de capteurs et pour gérer la fréquence d'échantillonnage des données de capteurs.

Voici un exemple de code pour détecter la présence d'un capteur de luminosité sur un dispositif :

```
    // Obtention d'une reference au gestionnaire de capteurs
      SensorManager sensorManager = (SensorManager)
         getSystemService(Context.SENSOR_SERVICE);

       // Verification de la disponibilite du capteur de
          luminosite
       if (sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT) !=
          null) {
```

```
            // Le capteur est disponible, on peut l'utiliser
            Sensor lightSensor = sensorManager.getDefaultSensor(
                Sensor.TYPE_LIGHT);
            Toast.makeText(this, "Capteur de luminosite
                disponible", Toast.LENGTH_SHORT).show();
        } else {
            // Le capteur n'est pas disponible, on informe l'
                utilisateur
            Toast.makeText(this, "Capteur de luminosite
                indisponible", Toast.LENGTH_SHORT).show();
        }
```

Voici l'interface utilisateur qui a détecté la présence d'un capteur de luminosité sur un dispositif Oppo :

FIGURE 3 – Presence de capteur de luminosité

0.5.4 Accéléromètre

L'accéléromètre est un capteur de mouvement qui permet de mesurer l'accélération linéaire d'un objet. Il est souvent utilisé dans les appareils mobiles pour détecter l'orientation et le mouvement de l'appareil. L'accéléromètre mesure les variations de l'accélération dans les trois axes de l'espace, généralement notés X, Y et Z. Ces valeurs sont mesurées en mètres par seconde au carré (m/s^2) ou en unités de gravité (g).

Voici un exemple de code qui represente comment utiliser le capteur accéléromètre.

```
@Override
    public void onSensorChanged(SensorEvent event) {

        float x = event.values[0];
        float y = event.values[1];
        float z = event.values[2];

        if ( x <= 0.15 ){
            rl.setBackgroundColor(Color.parseColor("#47b550"));
        }
        if(x > 0.15 && x <= 1 ){
            rl.setBackgroundColor(Color.parseColor("#000000"));
        }
        if ( x > 1 ){
            rl.setBackgroundColor(Color.parseColor("#b53333"));
        }


    }
```

Voici le resultat d'utilisation de cette methode ;

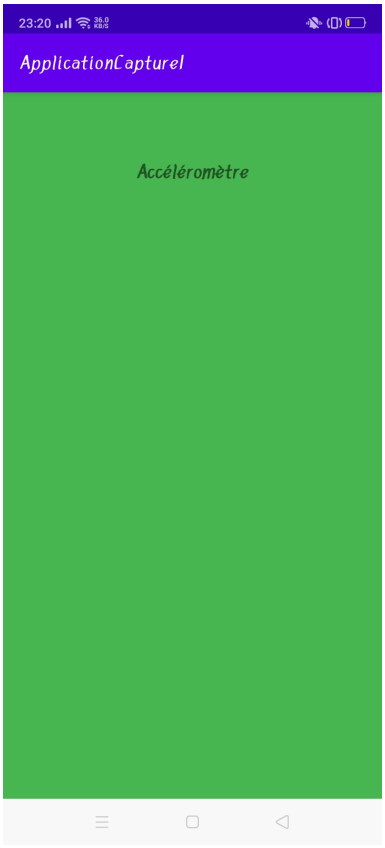

FIGURE 4 – valeur inférieure

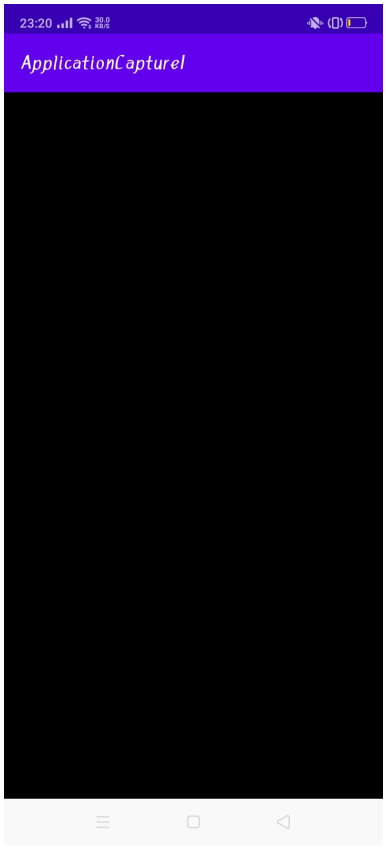

FIGURE 5 – valeur moyenne

FIGURE 6 – valeur supérieure

0.5.5 Direction du mouvement

```
    @Override
   public void onSensorChanged(SensorEvent event) {

       float xChange = history[0] - event.values[0];
       float yChange = history[1] - event.values[1];

       history[0] = event.values[0];
       history[1] = event.values[1];

       if (xChange > 2){
           direction[0] = "GAUCHE";
       }
       else if (xChange < -2){
           direction[0] = "DROITE";
       }

       if (yChange > 2){
           direction[1] = "BAS";
       }
       else if (yChange < -2){
           direction[1] = "HAUT";
       }

       String res = "DIRECTION : " + direction[0] + ",  " +
          direction[1];
       tv.setText(res);

   }
```

FIGURE 7 – Direction : droite, bas

FIGURE 8 – Direction : gouche, haut

0.5.6 Secouer un appareil

Pour détecter un mouvement de secousse sur un appareil Android, vous pouvez utiliser le capteur d'accéléromètre intégré dans l'appareil. Voici un exemple de code qui montre comment détecter un mouvement de secousse :

```
    @Override
   public void onSensorChanged(SensorEvent event) {


       if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER)
          {
           long curTime = System.currentTimeMillis();
           if ((curTime - mLastShakeTime) >
              MIN_TIME_BETWEEN_SHAKES_MILLISECS) {

               float x = event.values[0];
               float y = event.values[1];
               float z = event.values[2];

               double acceleration = Math.sqrt(Math.pow(x, 2) +
                       Math.pow(y, 2) +
                       Math.pow(z, 2)) - SensorManager.
                          GRAVITY_EARTH;
               if (acceleration > SHAKE_THRESHOLD) {
                   mLastShakeTime = curTime;
```

```
                    Boolean light = true;
                    if (light) {
                        try {
                            cameraManager.setTorchMode(cameraId,
                               true);
                            System.out.println("CAMERA ON CAMERA
                               ON CAMERA ON");
                            tv_title.setText("CAMERA ON");
                            Toast.makeText(this, "CAMERA ON",
                               Toast.LENGTH_LONG).show();
                        } catch (CameraAccessException e) {
                            e.printStackTrace();
                        }
                        light = false;
                    } else {
                        try {
                            cameraManager.setTorchMode(cameraId,
                               false);
                            System.out.println("CAMERA OFF CAMERA
                                OFF CAMERA OFF");
                            tv_title.setText("CAMERA OFF");
                            Toast.makeText(this, "CAMERA OFF",
                               Toast.LENGTH_LONG).show();
                        } catch (CameraAccessException e) {
                            e.printStackTrace();
                        }

                        light = true;
                    }
                }
            }
        }
    }
```

FIGURE 9 – Secouer un appareil, allume le flash en secouant l'appareil

FIGURE 10 – Captur indiquat que le flash est allumé en secouant l'appareil

0.5.7 Proximité

Le capteur de proximité est un capteur présent sur certains smartphones Android qui permet de détecter la présence d'un objet à proximité de l'écran du téléphone, généralement le visage de l'utilisateur lorsqu'il passe un appel téléphonique. Le capteur de proximité fonctionne en mesurant la réflexion d'un faisceau infrarouge émis par le capteur. Lorsqu'un objet est détecté à proximité, la lumière réfléchie est plus importante, ce qui permet de déterminer la présence de l'objet.

```
    @Override
       public void onSensorChanged(SensorEvent event) {
           // TODO Auto-generated method stub
          if (event.sensor.getType() == Sensor.TYPE_PROXIMITY) {
               if (event.values[0] == 0) {
                   data.setText("Near");
                   iv.setImageDrawable(getDrawable(R.drawable.
                      hautparle));
               } else {
                   data.setText("Away");
                   iv.setImageDrawable(getDrawable(R.drawable.
                      volume));
               }
           }
       }
```

Vous trouverez ici les différents interfaces graphique de l'application.

FIGURE 11 – L'objet est loin

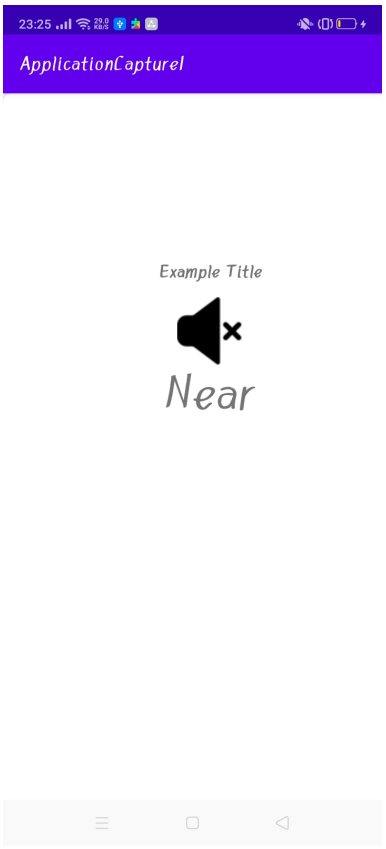

FIGURE 12 – L'objet est proche

0.5.8 Géolocalisation

Le capteur de géolocalisation sur un appareil Android est généralement un module GPS qui permet de déterminer la position de l'appareil à l'aide de satellites. Le GPS est un système de positionnement mondial qui utilise des signaux envoyés par des satellites en orbite autour de la Terre. L'appareil Android peut également utiliser d'autres méthodes de localisation telles que le Wi-Fi, le réseau mobile et le Bluetooth pour déterminer la position de l'appareil.

```
    // M thode appel e lorsqu'on demande la permission d'
       acc s    la position
@Override
public void onRequestPermissionsResult(int requestCode,
   String[] permissions, int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions
       , grantResults);

    if (requestCode == 1) {
        if (grantResults.length > 0 && grantResults[0] ==
           PackageManager.PERMISSION_GRANTED &&
                grantResults[1] == PackageManager.
                   PERMISSION_GRANTED) {
            // Si les autorisations sont accord es, demander
                la mise    jour de la position
            locationManager.requestLocationUpdates(
```

```
                LocationManager.GPS_PROVIDER, 0, 0, this);
        } else {
            // Si les autorisations ne sont pas accord es,
                afficher un message d'erreur
            Toast.makeText(this, "Veuillez accorder les
                permissions pour acc der    la position",
                Toast.LENGTH_SHORT).show();
        }
    }
}
```

FIGURE 13 – Activé autorisations d'accées

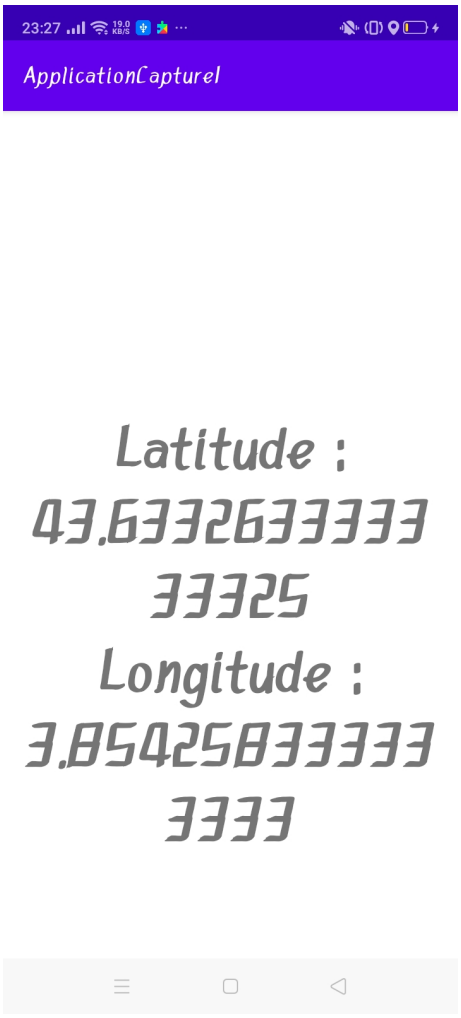

FIGURE 14 – Affichage de la localisation.

0.6 Le TP est sur ma page Git hub

voici le lien : https ://github.com/Toudherth/ApplicationCapture1

Clickez ici sur le lien vers Git hub.