Compte rendu :Internationalisation Android

Enseignant : Maddouri Faouzi

Réalisé par :

Guesmi Ikram
Farhaoui Eya

Année universitaire : 2024/2025

Établissement : ISET Rades

Contents

1 Création de l'interface utilisateur (UI)

1.1 Fichier XML (`activity_main.xml`)

Le fichier XML `activity_main.xml` a été conçu pour fournir une interface utilisateur claire et intuitive. Voici les éléments principaux inclus :

- **TextView :**affichent des informations clés (latitude, longitude, altitude, signal et batterie), qui sont mises à jour dynamiquement par l'application.
- **Button :** permettent à l'utilisateur de changer la langue de l'interface et de sauvegarder les informations dans un fichier CSV.

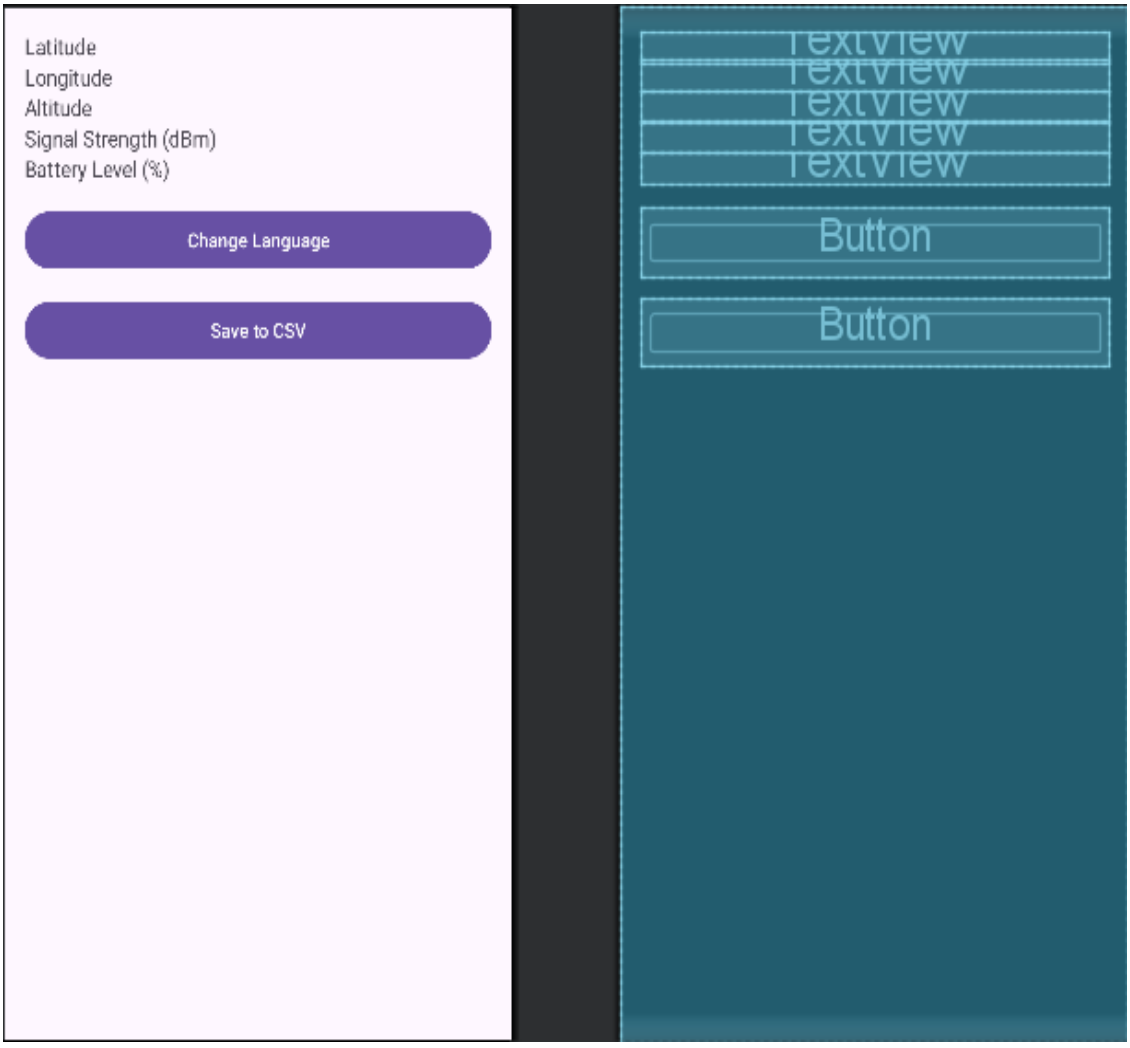

Figure 1: Layout de l'activité principale

1.2 Code XML pour l'interface utilisateur

Voici le code XML de la mise en page de l'interface utilisateur :

```xml
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:id="@+id/tvLatitude"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/latitude"
        android:textSize="16sp" />

    <TextView
        android:id="@+id/tvLongitude"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/longitude"
        android:textSize="16sp" />

    <TextView
        android:id="@+id/tvAltitude"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/altitude"
        android:textSize="16sp" />

    <TextView
        android:id="@+id/tvSignalStrength"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/signal_strength"
        android:textSize="16sp" />

    <TextView
        android:id="@+id/tvBatteryLevel"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/battery_level"
        android:textSize="16sp" />

    <Button
        android:id="@+id/btnChangeLanguage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/change_language"
        android:layout_marginTop="16dp" />

    <Button
        android:id="@+id/btnSaveToCSV"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/save_to_csv"
        android:layout_marginTop="16dp" />
</LinearLayout>
```

2 Ajout de permission au niveau de fichier AndroidManifest.xml:

Dans le fichier AndroidManifest.xml, nous avons ajouté une permission essentielle pour assurer le bon fonctionnement de l'application.

- **ACCESS_FINE_LOCATION** : Cette permission permet à l'application d'obtenir la position précise de l'utilisateur via les services de géolocalisation GPS. Elle est nécessaire pour récupérer des données de latitude, de longitude et d'altitude avec une haute précision.
- **ACCESS_COARSE_LOCATION** : Cette permission permet d'obtenir la position approximative de l'utilisateur à partir du réseau cellulaire ou des réseaux Wi-Fi. Bien que moins précise que `ACCESS_FINE_LOCATION`, elle est utile pour une estimation de localisation basée sur les tours cellulaires.
- **ACCESS_NETWORK_STATE** : Cette permission permet à l'application d'accéder aux informations sur l'état du réseau (Wi-Fi, données mobiles, etc.). Elle est nécessaire pour vérifier si le téléphone est connecté à Internet et obtenir des informations sur le réseau actuel.
- **ACCESS_WIFI_STATE** : Permet à l'application d'obtenir l'état du réseau Wi-Fi, y compris les informations sur les connexions Wi-Fi actives et la force du signal Wi-Fi.
- **READ_PHONE_STATE** : Cette permission permet à l'application d'accéder à certaines informations sur l'état du téléphone, y compris le niveau du signal mobile, le numéro de téléphone.
- **WRITE_EXTERNAL_STORAGE** : Cette permission permet à l'application d'écrire dans le stockage externe (par exemple, sur la carte SD ou dans un répertoire spécifique de l'appareil). Elle est nécessaire pour enregistrer le fichier `LogTracking.csv` dans lequel l'application stocke les données de géolocalisation et de signal.
- **READ_EXTERNAL_STORAGE** : Permet à l'application de lire les fichiers du stockage externe.
- **BATTERY_STATS** : Cette permission permet à l'application de collecter des statistiques sur la batterie, comme le niveau de charge et d'autres informations sur la consommation d'énergie du dispositif.

2.1 Code du fichier AndroidManifest.xml

Voici le code XML du fichier 'AndroidManifest.xml' qui inclut les permissions nécessaires :

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.BATTERY_STATS"
        tools:ignore="ProtectedPermissions" />
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
```

```
        android:theme="@style/Theme.GPSLogger"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

3 Configuration de l'activité principale (MainActivity) :

Cette section explique chaque fonction utilisée dans l'activité principale de l'application, qui permet de collecter et afficher des informations relatives à la géolocalisation, l'état du signal mobile, l'état de la batterie et de changer la langue de l'application.

3.1 Méthode `onCreate()`

La méthode `onCreate()`est utilisée pour initialiser l'interface utilisateur et configurer les services nécessaires, tels que la gestion de la géolocalisation et du signal mobile.

- **Initialisation des vues :** Les vues `TextView` et `Button` sont initialisées pour afficher la latitude, la longitude, l'altitude, le niveau du signal et de la batterie, et pour gérer les événements de changement de langue et de sauvegarde des données dans un fichier CSV.
- **Configuration des services :** La méthode récupère les services `LocationManager` et `TelephonyManager` nécessaires pour la géolocalisation et l'état du réseau mobile.
- **Vérification des permissions :** Avant d'accéder aux fonctionnalités requérant des permissions (comme la géolocalisation), la méthode `checkPermissions()` est appelée pour vérifier si l'application a les permissions nécessaires. Si elles sont accordées, la méthode `requestLocationUpdates()` est appelée pour récupérer les mises à jour de la position de l'utilisateur.
- **Gestion des boutons :** Les boutons `btnChangeLanguage` et `btnSaveToCSV` sont configurés pour changer la langue et sauvegarder les données dans un fichier CSV respectivement.

```

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

        tvLatitude = findViewById(R.id.tvLatitude);
        tvLongitude = findViewById(R.id.tvLongitude);
        tvAltitude = findViewById(R.id.tvAltitude);
        tvSignalStrength = findViewById(R.id.tvSignalStrength);
        tvBatteryLevel = findViewById(R.id.tvBatteryLevel);

        Button btnChangeLanguage = findViewById(R.id.btnChangeLanguage);
        Button btnSaveToCSV = findViewById(R.id.btnSaveToCSV);
```

```
        locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
        telephonyManager = (TelephonyManager) getSystemService(TELEPHONY_SERVICE);
        logFile = new File(getExternalFilesDir(null), "LogTracking.csv");

        btnChangeLanguage.setOnClickListener(v -> changeLanguage());
        btnSaveToCSV.setOnClickListener(v -> saveToCSV());

        // Vérifiez les permissions
        if (checkPermissions()) {
            requestLocationUpdates();
        } else {
            requestPermissions();
        }
    }

```

3.2 Méthode `requestLocationUpdates()`

La méthode `requestLocationUpdates()` est utilisée pour démarrer la collecte des mises à jour de localisation toutes les secondes (`1000` millisecondes) avec une distance minimale de `0` mètres entre les mises à jour. Elle utilise le fournisseur GPS (`LocationManager.GPS_PROVIDER`).

- **Gestion des exceptions :** La méthode est entourée d'un bloc `try-catch` pour capturer une `SecurityException` au cas où les permissions nécessaires n'ont pas été accordées.

```

 private void requestLocationUpdates() {
        try {
            locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 1000, 0,
            ↪  locationListener);
        } catch (SecurityException e) {
            e.printStackTrace();
            Toast.makeText(this, "Permissions not granted for location access",
            ↪  Toast.LENGTH_SHORT).show();
        }
    }
```

3.3 Méthode `checkPermissions()`

La méthode `checkPermissions()` vérifie si l'application a les permissions nécessaires pour accéder à la géolocalisation précise (`ACCESS_FINE_LOCATION`), à la géolocalisation approximative (`ACCESS_COARSE_LOCATION`) et à l'état du téléphone (`READ_PHONE_STATE`).

- **Retour booléen :** Elle renvoie `true` si toutes les permissions sont accordées, sinon elle renvoie `false`.

```
private boolean checkPermissions() {
        return ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
        ↪  == PackageManager.PERMISSION_GRANTED &&
                ActivityCompat.checkSelfPermission(this,
                ↪  Manifest.permission.ACCESS_COARSE_LOCATION) ==
                ↪  PackageManager.PERMISSION_GRANTED &&
```

```
                ActivityCompat.checkSelfPermission(this, Manifest.permission.READ_PHONE_STATE) ==
                ↪  PackageManager.PERMISSION_GRANTED;

```

3.4 Méthode `requestPermissions()`

La méthode `requestPermissions()` est appelée si les permissions ne sont pas accordées. Elle demande explicitement à l'utilisateur de fournir les permissions nécessaires pour accéder aux fonctionnalités de géolocalisation et d'état du téléphone.

- **Demande de permissions :** Les permissions `ACCESS_FINE_LOCATION`, `ACCESS_COARSE_LOCATION` et `READ_PHONE_STATE` sont demandées avec un code de demande `PERMISSION_REQUEST_CODE`.

```

private void requestPermissions() {
        ActivityCompat.requestPermissions(this,
                new String[]{
                        Manifest.permission.ACCESS_FINE_LOCATION,
                        Manifest.permission.ACCESS_COARSE_LOCATION,
                        Manifest.permission.READ_PHONE_STATE
                },
                PERMISSION_REQUEST_CODE);
    }
```

3.5 Méthode `onRequestPermissionsResult()`

Cette méthode est appelée après que l'utilisateur ait répondu à la demande de permissions. Elle vérifie si les permissions ont été accordées.

- **Réponse de l'utilisateur :** Si la permission est accordée, elle appelle `requestLocationUpdates()` pour démarrer la collecte des données de localisation. Sinon, elle affiche un message indiquant que les permissions sont nécessaires pour que l'application fonctionne correctement.

```

@Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
    ↪  @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (requestCode == PERMISSION_REQUEST_CODE) {
            if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                requestLocationUpdates();
            } else {
                Toast.makeText(this, "Permissions denied. App cannot function properly.",
                ↪  Toast.LENGTH_SHORT).show();
            }
        }
    }
```

3.6 Méthode `changeLanguage()`

La méthode `changeLanguage()` permet de changer la langue de l'application en fonction de la langue actuelle. Elle vérifie la langue actuelle et bascule entre l'anglais, le français et l'arabe.

- **Changement de locale :** La locale de l'application est mise à jour et l'activité est redémarrée pour appliquer la nouvelle langue.

```
1
2  private void changeLanguage() {
3         Locale current = getResources().getConfiguration().locale;
4         String newLanguage = current.getLanguage().equals("en") ? "fr" :
          ↪  current.getLanguage().equals("fr") ? "ar" : "en";
5         Locale locale = new Locale(newLanguage);
6         Locale.setDefault(locale);
7
8         Configuration config = new Configuration();
9         config.setLocale(locale);
10        getResources().updateConfiguration(config, getResources().getDisplayMetrics());
11
12        // Restart activity to apply changes
13        Intent intent = getIntent();
14        finish();
15        startActivity(intent);
16    }
```

3.7 Méthode `saveToCSV()`

La méthode `saveToCSV()` permet de sauvegarder les informations de géolocalisation et du signal mobile dans un fichier CSV. Elle écrit les données récupérées dans un fichier situé dans le répertoire de stockage externe de l'application.

- **Écriture dans un fichier CSV :** Elle utilise un `FileWriter` pour ajouter une ligne avec la latitude, la longitude, l'altitude, le niveau du signal et la charge de la batterie dans le fichier `LogTracking.csv`.
- **Gestion des erreurs :** Si une erreur se produit lors de l'écriture dans le fichier, un message d'erreur est affiché à l'utilisateur.

```
1
2  private void saveToCSV() {
3         try (FileWriter writer = new FileWriter(logFile, true)) {
4             writer.append(String.format(Locale.getDefault(), "%s,%s,%s,%s,%s\n",
5                     tvLatitude.getText(),
6                     tvLongitude.getText(),
7                     tvAltitude.getText(),
8                     tvSignalStrength.getText(),
9                     tvBatteryLevel.getText()));
10            Toast.makeText(this, "Data saved to CSV", Toast.LENGTH_SHORT).show();
11        } catch (IOException e) {
12            e.printStackTrace();
13            Toast.makeText(this, "Failed to save data", Toast.LENGTH_SHORT).show();
14        }
15    }
```

3.8 Méthode `onLocationChanged()`

La méthode `onLocationChanged()` est appelée chaque fois que la localisation de l'utilisateur change. Elle récupère les informations de latitude, longitude, altitude, signal et batterie et les affiche dans les `TextView` correspondants.

- **Mise à jour des vues :** Les informations de localisation sont affichées dans les vues `tvLatitude`, `tvLongitude`, `tvAltitude`, `tvSignalStrength` et `tvBatteryLevel`.

```
1
2   private final LocationListener locationListener = new LocationListener() {
3       @Override
4       public void onLocationChanged(@NonNull Location location) {
5           double latitude = location.getLatitude();
6           double longitude = location.getLongitude();
7           double altitude = location.getAltitude();
8           int signalStrength = getSignalStrength();
9           int batteryLevel = getBatteryLevel();
10
11          tvLatitude.setText(getString(R.string.latitude) + ": " + latitude);
12          tvLongitude.setText(getString(R.string.longitude) + ": " + longitude);
13          tvAltitude.setText(getString(R.string.altitude) + ": " + altitude);
14          tvSignalStrength.setText(getString(R.string.signal_strength) + ": " + signalStrength);
15          tvBatteryLevel.setText(getString(R.string.battery_level) + ": " + batteryLevel);
16      }
17  };
```

3.9 Méthode `getSignalStrength()`

La méthode `getSignalStrength()` récupère le niveau du signal cellulaire actuel en utilisant `TelephonyManager`. Elle vérifie d'abord si la permission `READ_PHONE_STATE` a été accordée avant de récupérer les informations sur la cellule et d'extraire le niveau du signal (en dBm).

- **Récupération du signal :** Si la permission est accordée, elle obtient la force du signal de la première cellule disponible dans la liste `cellInfos`.

```
1
2   private int getSignalStrength() {
3       if (ActivityCompat.checkSelfPermission(this, Manifest.permission.READ_PHONE_STATE) !=
        ↪  PackageManager.PERMISSION_GRANTED) {
4           return 0;
5       }
6       List<CellInfo> cellInfos = telephonyManager.getAllCellInfo();
7       if (cellInfos != null && !cellInfos.isEmpty()) {
8           CellSignalStrength cellSignalStrength = cellInfos.get(0).getCellSignalStrength();
9           return cellSignalStrength.getDbm();
10      }
11      return 0;
12  }
```

3.10 Méthode `getBatteryLevel()`

La méthode `getBatteryLevel()` récupère le niveau de la batterie en utilisant un `IntentFilter` pour l'action `ACTION_BATTERY_CHANGED`. Elle extrait le niveau de la batterie et le renvoie sous forme de pourcentage.

- **Récupération de l'état de la batterie :** Elle calcule le niveau de la batterie en fonction des valeurs de niveau et d'échelle récupérées depuis `Intent`.

```
 private int getBatteryLevel() {
        IntentFilter intentFilter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
        Intent batteryStatus = registerReceiver(null, intentFilter);
        if (batteryStatus != null) {
            int level = batteryStatus.getIntExtra(BatteryManager.EXTRA_LEVEL, -1);
            int scale = batteryStatus.getIntExtra(BatteryManager.EXTRA_SCALE, -1);
            return (int) ((level / (float) scale) * 100);
        }
        return -1;
    }
```

Le code de `MainActivity.java` est le suivant :

```
package com.example.gpslogger;

import android.Manifest;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.pm.PackageManager;
import android.content.res.Configuration;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.BatteryManager;
import android.os.Bundle;
import android.telephony.CellSignalStrength;
import android.telephony.CellInfo;
import android.telephony.TelephonyManager;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.List;
import java.util.Locale;

public class MainActivity extends AppCompatActivity {

    private static final int PERMISSION_REQUEST_CODE = 1;

    private TextView tvLatitude, tvLongitude, tvAltitude, tvSignalStrength, tvBatteryLevel;
    private LocationManager locationManager;
    private TelephonyManager telephonyManager;
    private File logFile;

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

        tvLatitude = findViewById(R.id.tvLatitude);
```

```
        tvLongitude = findViewById(R.id.tvLongitude);
        tvAltitude = findViewById(R.id.tvAltitude);
        tvSignalStrength = findViewById(R.id.tvSignalStrength);
        tvBatteryLevel = findViewById(R.id.tvBatteryLevel);

        Button btnChangeLanguage = findViewById(R.id.btnChangeLanguage);
        Button btnSaveToCSV = findViewById(R.id.btnSaveToCSV);

        locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
        telephonyManager = (TelephonyManager) getSystemService(TELEPHONY_SERVICE);
        logFile = new File(getExternalFilesDir(null), "LogTracking.csv");

        btnChangeLanguage.setOnClickListener(v -> changeLanguage());
        btnSaveToCSV.setOnClickListener(v -> saveToCSV());

        // Vérifiez les permissions
        if (checkPermissions()) {
            requestLocationUpdates();
        } else {
            requestPermissions();
        }
    }

    private void requestLocationUpdates() {
        try {
            locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 1000, 0,
            ↪  locationListener);
        } catch (SecurityException e) {
            e.printStackTrace();
            Toast.makeText(this, "Permissions not granted for location access",
            ↪  Toast.LENGTH_SHORT).show();
        }
    }

    private boolean checkPermissions() {
        return ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
        ↪  == PackageManager.PERMISSION_GRANTED &&
                ActivityCompat.checkSelfPermission(this,
                ↪  Manifest.permission.ACCESS_COARSE_LOCATION) ==
                ↪  PackageManager.PERMISSION_GRANTED &&
                ActivityCompat.checkSelfPermission(this, Manifest.permission.READ_PHONE_STATE) ==
                ↪  PackageManager.PERMISSION_GRANTED;
    }

    private void requestPermissions() {
        ActivityCompat.requestPermissions(this,
                new String[]{
                        Manifest.permission.ACCESS_FINE_LOCATION,
                        Manifest.permission.ACCESS_COARSE_LOCATION,
                        Manifest.permission.READ_PHONE_STATE
                },
                PERMISSION_REQUEST_CODE);
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
    ↪  @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
```

```
        if (requestCode == PERMISSION_REQUEST_CODE) {
            if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                requestLocationUpdates();
            } else {
                Toast.makeText(this, "Permissions denied. App cannot function properly.",
                ↪  Toast.LENGTH_SHORT).show();
            }
        }
    }

    private void changeLanguage() {
        Locale current = getResources().getConfiguration().locale;
        String newLanguage = current.getLanguage().equals("en") ? "fr" :
        ↪  current.getLanguage().equals("fr") ? "ar" : "en";
        Locale locale = new Locale(newLanguage);
        Locale.setDefault(locale);

        Configuration config = new Configuration();
        config.setLocale(locale);
        getResources().updateConfiguration(config, getResources().getDisplayMetrics());

        // Restart activity to apply changes
        Intent intent = getIntent();
        finish();
        startActivity(intent);
    }

    private void saveToCSV() {
        try (FileWriter writer = new FileWriter(logFile, true)) {
            writer.append(String.format(Locale.getDefault(), "%s,%s,%s,%s,%s\n",
                    tvLatitude.getText(),
                    tvLongitude.getText(),
                    tvAltitude.getText(),
                    tvSignalStrength.getText(),
                    tvBatteryLevel.getText()));
            Toast.makeText(this, "Data saved to CSV", Toast.LENGTH_SHORT).show();
        } catch (IOException e) {
            e.printStackTrace();
            Toast.makeText(this, "Failed to save data", Toast.LENGTH_SHORT).show();
        }
    }

    private final LocationListener locationListener = new LocationListener() {
        @Override
        public void onLocationChanged(@NonNull Location location) {
            double latitude = location.getLatitude();
            double longitude = location.getLongitude();
            double altitude = location.getAltitude();
            int signalStrength = getSignalStrength();
            int batteryLevel = getBatteryLevel();

            tvLatitude.setText(getString(R.string.latitude) + ": " + latitude);
            tvLongitude.setText(getString(R.string.longitude) + ": " + longitude);
            tvAltitude.setText(getString(R.string.altitude) + ": " + altitude);
            tvSignalStrength.setText(getString(R.string.signal_strength) + ": " + signalStrength);
            tvBatteryLevel.setText(getString(R.string.battery_level) + ": " + batteryLevel);
        }
    };
```

```
152
153     private int getSignalStrength() {
154         if (ActivityCompat.checkSelfPermission(this, Manifest.permission.READ_PHONE_STATE) !=
        ↪  PackageManager.PERMISSION_GRANTED) {
155             return 0;
156         }
157         List<CellInfo> cellInfos = telephonyManager.getAllCellInfo();
158         if (cellInfos != null && !cellInfos.isEmpty()) {
159             CellSignalStrength cellSignalStrength = cellInfos.get(0).getCellSignalStrength();
160             return cellSignalStrength.getDbm();
161         }
162         return 0;
163     }
164
165     private int getBatteryLevel() {
166         IntentFilter intentFilter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
167         Intent batteryStatus = registerReceiver(null, intentFilter);
168         if (batteryStatus != null) {
169             int level = batteryStatus.getIntExtra(BatteryManager.EXTRA_LEVEL, -1);
170             int scale = batteryStatus.getIntExtra(BatteryManager.EXTRA_SCALE, -1);
171             return (int) ((level / (float) scale) * 100);
172         }
173         return -1;
174     }
175 }
176
```

4 Description des fichiers de ressources

L'application utilise des fichiers de ressources `strings.xml` pour définir les chaînes de texte de l'interface utilisateur dans trois langues : l'anglais, le français et l'arabe. Chaque fichier contient des chaînes de texte pour les différents éléments de l'interface, tels que le nom de l'application, les labels des informations géographiques, la force du signal et le niveau de la batterie.

4.1 Ressource en Anglais

```
1
2  <resources>
3      <string name="app_name">GPS Logger</string>
4      <string name="latitude">Latitude</string>
5      <string name="longitude">Longitude</string>
6      <string name="altitude">Altitude</string>
7      <string name="signal_strength">Signal Strength (dBm)</string>
8      <string name="battery_level">Battery Level (%)</string>
9      <string name="change_language">Change Language</string>
10     <string name="save_to_csv">Save to CSV</string>
11 </resources>
12
```

4.2 Ressource en Francais

```
<resources>
    <string name="app_name">Journal GPS</string>
    <string name="latitude">Latitude</string>
    <string name="longitude">Longitude</string>
    <string name="altitude">Altitude</string>
    <string name="signal_strength">Puissance du signal (dBm)</string>
    <string name="battery_level">Niveau de batterie (%)</string>
    <string name="change_language">Changer la langue</string>
    <string name="save_to_csv">Enregistrer en CSV</string>
</resources>
```

5 Résultats Observés

Figure 2: resultat en arabe

Figure 3: resultat en francais

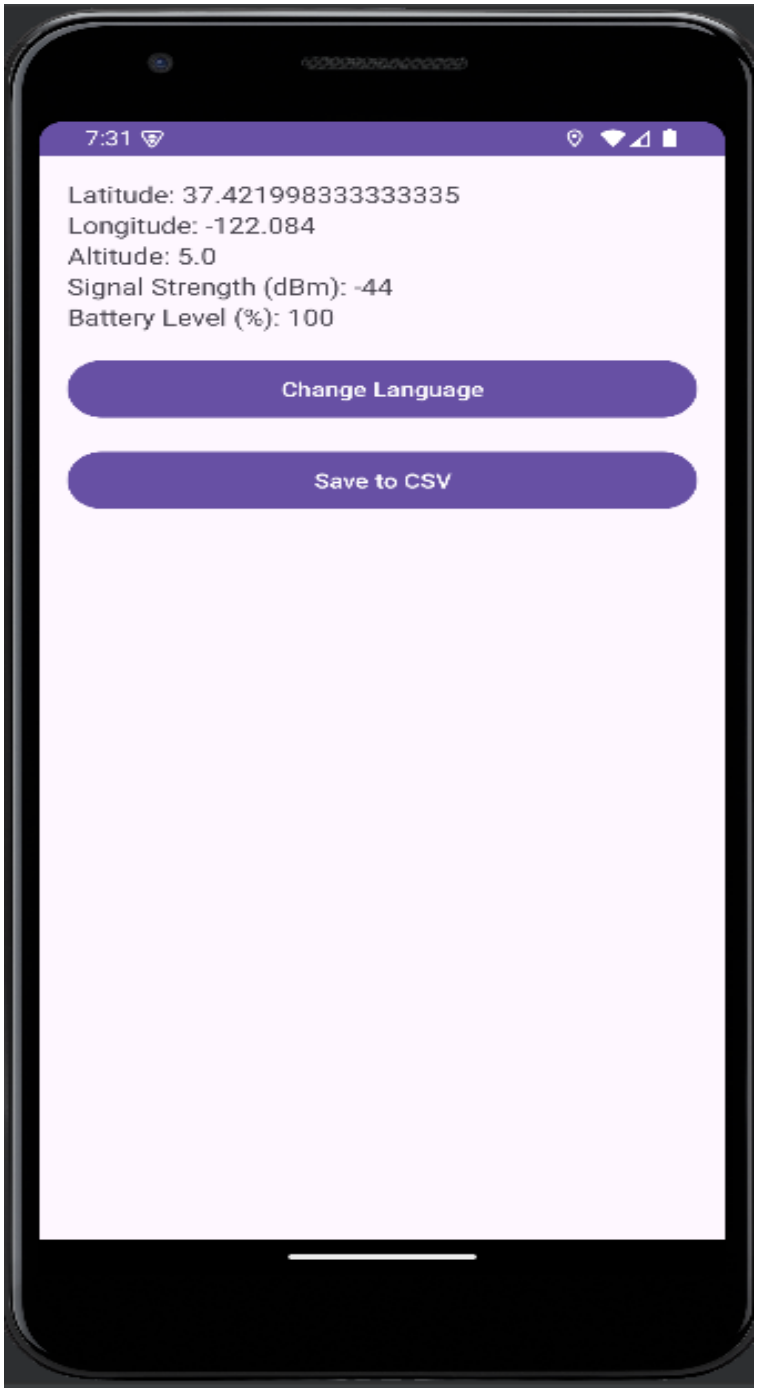

Figure 4: resultat en anglais

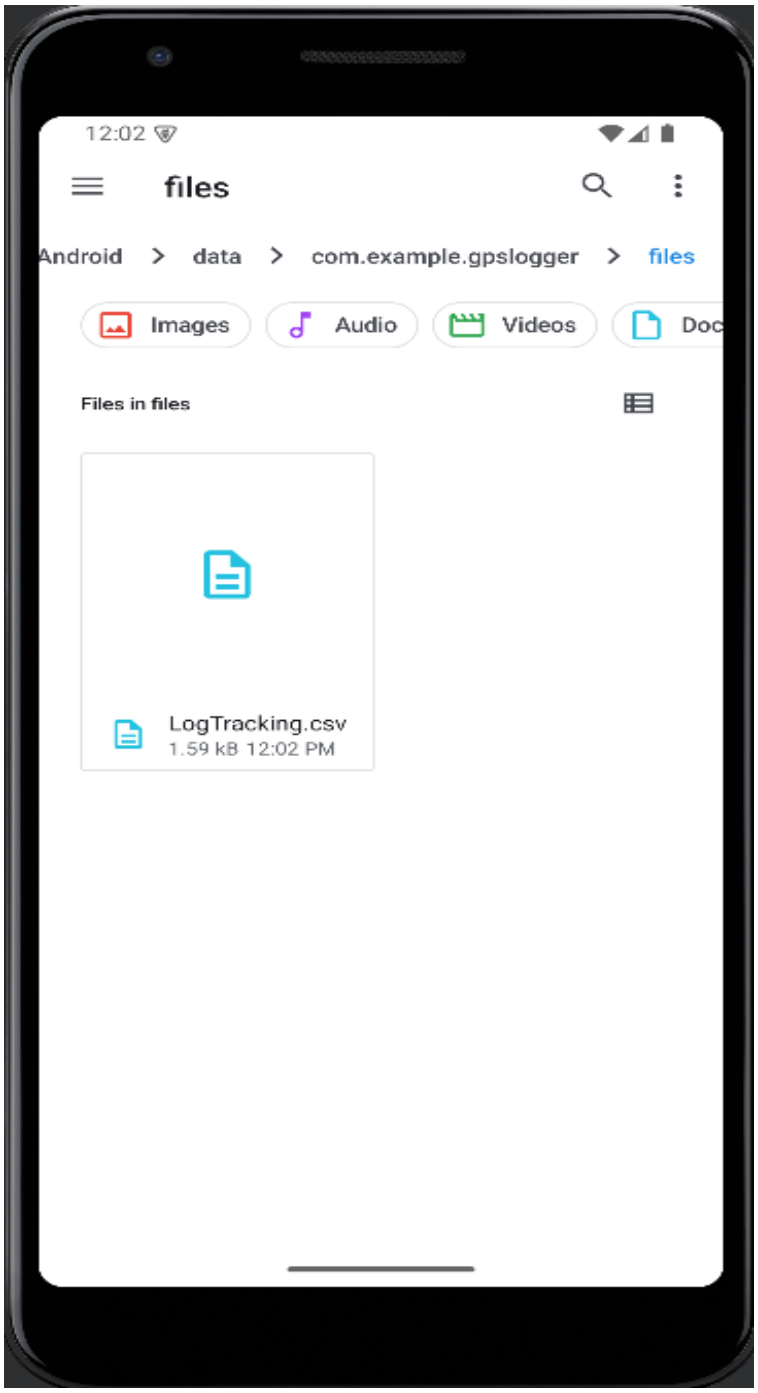

Figure 5: Enregistrement du fichier csv