TP N° 2 — Anatomie d'une application Android

*Implémentation du `BroadcastReceiver` : "**BatteryMonitorApp**"*

Objectif du TP

L'objectif de ce TP est de découvrir le rôle du `BroadcastReceiver` dans Android à travers un scénario pratique : surveiller l'état de la batterie pour adapter le comportement d'une application et réfléchir à l'optimisation énergétique.

Scénario motivant

Vous êtes développeur d'une application Android. Votre utilisateur se plaint qu'elle consomme trop d'énergie. Vous décidez donc de **surveiller l'état de charge de la batterie** : quand le téléphone est en charge, l'application suspend ses services énergivores.

Schéma explicatif du flux d'événements Android

Le système Android émet un `broadcast` lorsque la batterie change d'état. Le `BroadcastReceiver` intercepte cet événement et notifie l'application.

Figure : Flux d'événements — de la batterie vers l'application via le BroadcastReceiver.

1 Création du projet Android

Créez un projet **BatteryMonitorApp** dans Android Studio :
- API minimale : **21 (Android 5.0)**
- Langage : **Java** ou **Kotlin**

Remarque

Ce TP peut être réalisé en **Java** ou en **Kotlin**. Android Studio permet une conversion automatique :

`Code → Convert Java File to Kotlin File (Ctrl+Alt+Shift+K)`

Questions de préparation

- Qu'est-ce qu'un `BroadcastReceiver` ?
- Dans quels cas Android envoie-t-il des broadcasts ?
- Comment ce mécanisme peut-il aider à économiser la batterie ?

Discutez avec votre binôme avant de passer à la suite.

—

2 Création du BroadcastReceiver

Option 1 — Java

BatteryReceiver.java

```java
package com.example.batterymonitorapp;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.BatteryManager;
import android.widget.Toast;

public class BatteryReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        int status = intent.getIntExtra(BatteryManager.EXTRA_STATUS, -1);
        boolean isCharging = status == BatteryManager.BATTERY_STATUS_CHARGING;
        int level = intent.getIntExtra(BatteryManager.EXTRA_LEVEL, -1);

        if (isCharging) {
            Toast.makeText(context, "      Phone is charging",
                Toast.LENGTH_SHORT).show();
        } else if (level < 20) {
            Toast.makeText(context, "      Low battery!",
                Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(context, "      Battery level: " + level + "%",
                Toast.LENGTH_SHORT).show();
        }
    }
}
```

Option 2 — Kotlin

BatteryReceiver.kt

```kotlin
package com.example.batterymonitorapp

import android.content.BroadcastReceiver
import android.content.Context
import android.content.Intent
import android.os.BatteryManager
import android.widget.Toast

class BatteryReceiver : BroadcastReceiver() {
    override fun onReceive(context: Context?, intent: Intent?) {
        val status = intent?.getIntExtra(BatteryManager.EXTRA_STATUS, -1) ?: -1
        val isCharging = status == BatteryManager.BATTERY_STATUS_CHARGING
        val level = intent?.getIntExtra(BatteryManager.EXTRA_LEVEL, -1) ?: -1

        when {
            isCharging -> Toast.makeText(context, "      Phone is charging",
                Toast.LENGTH_SHORT).show()
            level < 20 -> Toast.makeText(context, "      Low battery!",
                Toast.LENGTH_SHORT).show()
            else -> Toast.makeText(context, "      Battery level: $level%",
                Toast.LENGTH_SHORT).show()
        }
    }
}
```

3 Déclaration dans le Manifest

AndroidManifest.xml (extrait)

```xml
<application
    android:allowBackup="true"
    android:label="@string/app_name"
    android:theme="@style/Theme.BatteryMonitorApp">

    <receiver android:name=".BatteryReceiver">
        <intent-filter>
            <action android:name="android.intent.action.BATTERY_CHANGED"/>
        </intent-filter>
    </receiver>

</application>
```

4 Enregistrement dynamique dans MainActivity

Option 1 — Java

MainActivity.java (extrait)

```java
package com.example.batterymonitorapp;

import androidx.appcompat.app.AppCompatActivity;
import android.content.IntentFilter;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    private BatteryReceiver batteryReceiver = new BatteryReceiver();

    @Override
    protected void onStart() {
        super.onStart();
        IntentFilter filter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
        registerReceiver(batteryReceiver, filter);
    }

    @Override
    protected void onStop() {
        super.onStop();
        unregisterReceiver(batteryReceiver);
    }
}
```

Option 2 — Kotlin

MainActivity.kt (extrait)

```kotlin
package com.example.batterymonitorapp

import android.content.IntentFilter
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {

    private val batteryReceiver = BatteryReceiver()

    override fun onStart() {
        super.onStart()
        registerReceiver(batteryReceiver,
            IntentFilter(Intent.ACTION_BATTERY_CHANGED))
    }

    override fun onStop() {
        super.onStop()
        unregisterReceiver(batteryReceiver)
    }
}
```

5 Test et amélioration

Défi en binôme

— Testez votre application : débranchez et rebranchez le chargeur.
— Ajoutez une vibration ou un son lorsque le téléphone est en charge.
— Comparez votre implémentation Java/Kotlin avec celle d'un autre binôme.

Question d'application avancée

Comment exploiter ce `BroadcastReceiver` pour améliorer l'expérience utilisateur ? Exemple : désactiver temporairement les services énergivores (GPS, synchronisation) quand le téléphone n'est pas en charge, puis les relancer automatiquement.

Réflexion finale

1. Quelle différence entre un Receiver statique et un Receiver dynamique ?
2. Pourquoi faut-il enregistrer/désenregistrer le Receiver correctement ?
3. Donnez un autre cas d'usage des `BroadcastReceiver` (connexion réseau, SMS, etc.).

Livrable

À remettre

Chaque binôme doit envoyer un fichier :

`TP2_Nom1_Nom2.zip`

Il doit contenir :
— Le projet complet `BatteryMonitorApp` ;
— Une capture d'écran du test (état de charge / batterie faible) ;
— Un court fichier `README.txt` décrivant le fonctionnement du `BroadcastReceiver`.