INSTITUTO FED. DE EDUCAÇÃO, CIÊNC. E TEC. DE PERNAMBUCO
CURSO: TEC. EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
DISCIPLINA: PROGRAMAÇÃO PARA DISPOSITIVOS MÓVEIS
PROFESSOR: RAMIDE DANTAS
ASSUNTO: ALARMES, NOTIFICAÇÕES E BROADCAST RECEIVER

Aula Prática 09

Parte 1: Criando a atividade principal

Passo 1: Crie um novo projeto contendo uma atividade em branco:

Application Name:	Pratica09
Domain name:	pdm.tads.ifpe.edu.br
Min. SDK:	**Api 19: Android 4.4 (KitKat)**
Template:	Empty Activity
Activity:	MainActivity

Passo 2: Monte a tela principal da aplicação como exibido na imagem abaixo:

Passo 3: Faça com que o clique do botão “Agendar” acione o método `scheduleAlarm()` na classe MainActivity.

Passo 4: Implemente o método `scheduleAlarm()` no MainActivity.java com o código a seguir:

```
public void scheduleAlarm(View V) {
    // Tempo atual mais 10 segundos (10 * 1000 milissegundos)
    Long time = new GregorianCalendar().getTimeInMillis()+10*1000;

    // Cria Intent do alarm, que será recebido pela classe AlarmReceiver
    Intent intentAlarm = new Intent(this, AlarmReceiver.class);

    // Cria um PendingIntent, usado para fazer o broadcast do alarme
    PendingIntent pendingAlarmIntent = PendingIntent.getBroadcast(
             this, 1, intentAlarm, PendingIntent.FLAG_UPDATE_CURRENT);

    //Obtem o gerenciador de alarmes do sistema
    AlarmManager alarmManager =
             (AlarmManager) this.getSystemService(Context.ALARM_SERVICE);

    //Configura um alarme lançará o intent em 10 segundos
    alarmManager.set(AlarmManager.RTC_WAKEUP, time, pendingAlarmIntent);

    Toast.makeText(this, "Alarme agendado.", Toast.LENGTH_LONG).show();

    // Finaliza a atividade
    this.finish();
}
```

Nesse método é agendado um evento de alarme para 10 segundos depois que o botão for clicado. O alarme é configurado através do serviço do sistema AlarmManager. Quando o alarme é disparado, um Intent pré-configurado é lançado no sistema. Nesse exemplo, é um Intent explícito destinado a acionar uma classe da nossa aplicação chamada AlarmReceiver.

Parte 2: Recebendo o alarme e gerando uma notificação para o usuário.

Passo 1: Crie uma classe chamada AlarmReceiver que estende BroadcastReceiver.

```
public class AlarmReceiver extends BroadcastReceiver {
       ...
}
```

Um BroadcastReceiver é um componente da aplicação que recebe Intents lançados em broadcast (difusão) pelo sistema ou outras aplicações. Através desse componente é possível tratar ações do usuário (e.g., enviar e-mail) ou eventos do sistema (e.g., o sistema acabou de dar o boot).

Passo 2: Implemente o método `onReceive` da classe AlarmReceiver:

```
    @Override
    public void onReceive(Context context, Intent intent)
    {

        // Crio o Intent que será associado com o toque na notificação
        // esse Intent irá lançar a aplicação novamente.
        Intent newIntent = new Intent(context, MainActivity.class);
        newIntent.addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP |
                           Intent.FLAG_ACTIVITY_CLEAR_TOP);
        PendingIntent pendingNotificationIntent = PendingIntent.getActivity(
                  context, 0, newIntent, PendingIntent.FLAG_UPDATE_CURRENT);

        // Contrói a notificação que será exibida ao usuário
        Notification.Builder builder = new Notification.Builder(context);
        builder.setSmallIcon(R.mipmap.ic_launcher);
        builder.setContentText("Alarme disparou! Toque para reagendar.");
        builder.setContentIntent(pendingNotificationIntent);
        builder.setAutoCancel(true);
        Notification notification = builder.build();

        // Obter o gerenciador de notificações do sistema e exibe a notificação
        NotificationManager notificationManager =
           (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
        notificationManager.notify(0, notification);

    }
```

Esse método será invocado pelo sistema Android no tempo configurado para o alarme. Ele recebe como parâmetro o Intent que foi configurado no método `scheduleAlarm()`. Nesse método, é criado um novo Intent, o qual é associado a uma notificação que quando clicada lança novamente a nossa aplicação. O processo de criação da notificação faz uso do padrão de projetos Builder (ver classe `Notification.Builder`), no qual a notificação é construída gradualmente e gerada ao final pelo método `builder.build()`. Para lançar a notificação para o usuário é usado outro serviço do sistema, o NotificationManager.

Passo 3: Adicione o `AlarmReceiver` ao manifesto da aplicação, para que ele possa receber aos Intents lançados pelo alarme.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="br.edu.ifpe.tads.pdm.pratica08" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <receiver android:name=".AlarmReceiver"/>

    </application>
</manifest>
```

Parte 3: Melhorando a aplicação.

Passo 1: Modifique a atividade principal para adicionar um campo de texto no qual o usuário pode digitar quanto tempo esperar para receber o alarme.

Passo 2: Modifique o código do método `onReceive` da classe AlarmReceiver para lançar o alarme no tempo configurado pelo usuário.

Passo 3: Modifique a notificação lançada pela aplicação para, por exemplo, fazer um som ou piscar o LED do celular.