Importar una base de datos SQLite

Funcionamiento

Se dispone de la aplicación Android ***d_almacenamiento_f_sqllite_ej,*** esta aplicación contiene una base de datos de nombres de personas.
Inicialmente la app importa la base de datos desde un fichero sqlite en la carpeta raw.
Interesa pues, que cada vez que se actualice la base de datos también se actualice los datos de la aplicación, para esto debemos de tener en cuenta los siguientes ficheros:

- **AndroidManifest.xml:** dentro de este fichero debemos aumentar en uno el código de la versión (versionCode) de la aplicación cada vez que se realice una actualización de la aplicación. A continuación se muestra como el valor de versionCode ha pasado de tener el valor 1 a tener el valor 2.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.pablomonteserin.almacenamiento.sqllite.ej"
    android:versionCode="2" android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="19"/>
```

- **invitado_db:** Es el fichero sqlite que contiene los datos de la aplicación, este fichero sólo se carga la primera vez que se ejecuta la aplicación o en una actualización. Por eso, cada vez que se quiera actualizar la base de datos, se debe modificar el código de la versión en el fichero **AndroidManifest.xml** mencionado anteriormente.

Finalmente una vez actualizado el código de la aplicación, es decir, tanto el contenido de **invitado_db** como el código de la versión en **AndroidManifest.xml,** se procederá a ejecutar la aplicación. Y así podremos ver que los datos que se cargan están actualizados.

¿Cómo funciona?

Haciendo uso de la clase **SharedPreferences** de Android guardaremos en un fichero una configuración que nos permita saber si se ha ejecuta la aplicación al menos una vez y si el código de la versión ha cambiado.

Creamos 2 nuevas variables en la clase MyAPP:
La primera variable indica el nombre del fichero de preferencias.
La segunda variable indicará si se ha detectado una actualización.

```
private final String PREFERENCES_FILE = "Ajustes";
private boolean hayActualizacion;
```

Ahora procedemos a recuperar la última configuración guardada:

```
SharedPreferences prefs = this.getSharedPreferences(PREFERENCES_FILE, Context.MODE_PRIVATE);

boolean flag = prefs.getBoolean("flag",false);
int version = prefs.getInt("version", -1);
```

El valor de **flag** nos indicará si la aplicación se ha ejecutado al menos una vez.
El valor de **version** nos indicará la ultima versión que la aplicación tuvo la última vez que se ejecutó.

Recuperando la versión de la aplicación

El siguiente trozo de código almacena en la variable **pInfo** toda la información de configuración de la aplicación. De esta manera en la variable **posible_nueva_version** se almacena la versión actual de la aplicación. Luego, se procede a comparar si el valor de la variable **version** es inferior a la versión actual de la aplicación, si lo es, se detecta que hay actualización y se indica así asignando **true** a la variable **hayActualizacion,** inmediatamente después se almacena la nueva versión en nuestro fichero de configuración.

```
PackageInfo pInfo;
try {
    pInfo = getPackageManager().getPackageInfo(getPackageName(), 0);
    int posible_nueva_version = pInfo.versionCode;

    if(version<posible_nueva_version){
        hayActualizacion = true;
        prefs = this.getApplicationContext().getSharedPreferences(PREFERENCES_FILE, Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = prefs.edit();
        editor.putInt("version", posible_nueva_version);
        editor.commit();
    }
    else{
        hayActualizacion = false;
    }

} catch (NameNotFoundException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}
```

Finalmente, para determinar si debemos sobreescribir la base de datos de la aplicación con lo que tenemos en el fichero **invitado_db** debemos hacer la siguiente comprobación:

```
if(!flag || hayActualizacion){

    prefs = this.getApplicationContext().getSharedPreferences(PREFERENCES_FILE, Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = prefs.edit();
    editor.putBoolean("flag", true);
    editor.commit();

    //Cargar base de datos...
```

Tanto si el flag está desactivado porque nunca se ha accedido a la aplicación o si hay una actualización detectada se procede a cargar todos los datos que hay en el fichero **invitado_db.**

Se puede apreciar también que se guarda la variable flag en el fichero de configuración, de esta manera guardamos la configuración de que el usuario ya ha accedido a la aplicación al menos una vez haciendo que la variable **flag** tenga el valor **true**.