Laboratorio 7

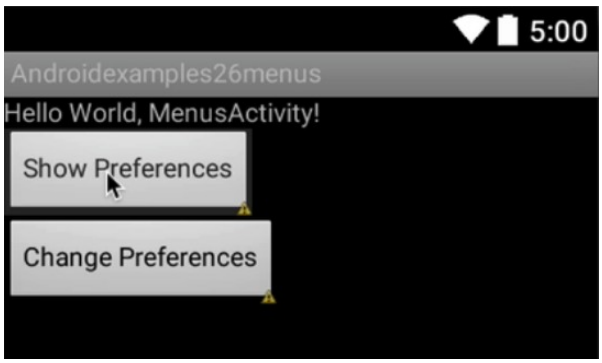
Apellidos: Moreno Vera
Nombres: Felipe Adrian
Código: 20120354I
Asignatura: Programación en Dispositivos Móviles (CC481)

2016 - I

Indice

Actividad 1

1. **En nuestra aplicación vamos a tener creadas dos *Activities*. Crearlas en el fichero *Manifest*.**
    1. **Una Activity principal llamada "MenusActivity".**
    2. **Una segunda llamada "MisPreferencias".**
2. **En nuestra *Activity* principal creamos nuestra vista como expone la imagen. En ella:**
    1. **Show preference: Muestra en pantalla mis preferencias**
    2. **Change Preferences: Cambia nuestras preferencias.**



3. **Volvemos al código Java de "MenusActivity". En ella vamos a declaramos el objeto *SharedPreferences* como atributo.**
   SharedPreferences preferences;
4. **A continuación nos centramos en el método onCreate de nuestra Activity principal.**
    1. **En él inicializamos el sistema de preferencias, como expone a continuación. Explique lo que expone.**

```
Button bShowPreferences =
(Button) findViewById(R.id.button1);
preferences = PreferenceManager
.getDefaultSharedPreferences(this);
```

    2. **Con los conceptos vistos en teoría añadimos funcionalidad a los *Button*. Para ello creamos los eventos *onClick***
        1. **El button "Show Preferences" que tendrá dos campos el de username y pass.**

```
bShowPreferences.setOnClickListener( new OnClickListener() {
public void onClick(View v) {
        String username =
        preferences.getString("username", "n/a");
        String password = preferences.getString(
"password", "n/a");
        showPrefs(username, password);
```

```
        }
});
```

2. **Para el Button Change Preference el evento que crearemos será el de llamada al método *"actualizarValorPrefrerencias()"***

```
Button buttonChangePreferences =
        (Button) findViewById(R.id.button2);
buttonChangePreferences.setOnClickListener(
new OnClickListener() {
public void onClick(View v) {
        updatePreferenceValue();
}
});
```

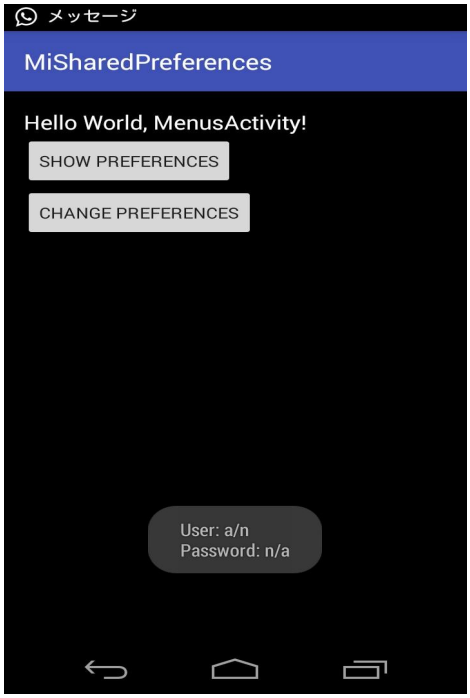
3. **Cree el método *showPreferences* que lo que hará es mostrar en un *TOAST* el username y el password pasado.**
4. **El método "*actualizarValorPreferencias()*" tendrá el siguiente código. Explique todo su contenido.**

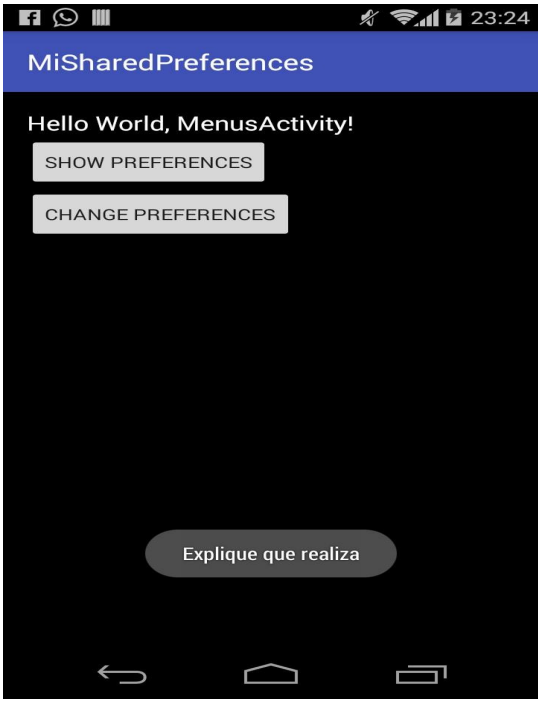
```
private void updatePreferenceValue(){
Editor edit = preferences.edit();
String username = preferences.getString("username", "n/a");
StringBuffer buffer = new StringBuffer();
for (int i = username.length() - 1;  i >= 0; i--) {
        buffer.append(username.charAt(i));
}
edit.putString("username", buffer.toString());
edit.commit();
Toast.makeText(this,  "Explique que realiza",
        Toast.LENGTH_LONG).show();
}
```

toma un string de un edit y lo almacena en un buffer, que luego muestra, con junto a username.

5. **Antes de ejecutar la aplicación y vemos que se ejecuta correctamente.**

Aquí mostramos una solución hasta el momento





6. **En nuestro código *onOptionsItemSelected()* creamos nuestra opción del menú. Esta opción tendrá el siguiente código, explíquelo.**

```
public boolean onOptionsItemSelected(MenuItem item) {
switch (item.getItemId()) {
        case R.id.preferences:
                Intent i = new Intent(this, MisPreferencias.class);
                 startActivity(i);
                 Toast.makeText(this,  "Introduce nombre/pass",
                         Toast.LENGTH_LONG).show();
                 break;
}
return true;
}
```

   si se da click en preferences, nos mandara a la clase preferencias junto con un Toast al terminar.

7. **En el código JAVA de la Activity "MisPreferencias".**
   1. **Heredará de *PreferenceActivity*.**
   2. **Deberemos añadir las siguiente línea en el método onCreate().**

   addPreferencesFromResource(R.xml.preferences);
8. **Definimos el fichero de la carpeta XML con el nombre "preferences.xml" (si no existe una carpeta xml crealá)**

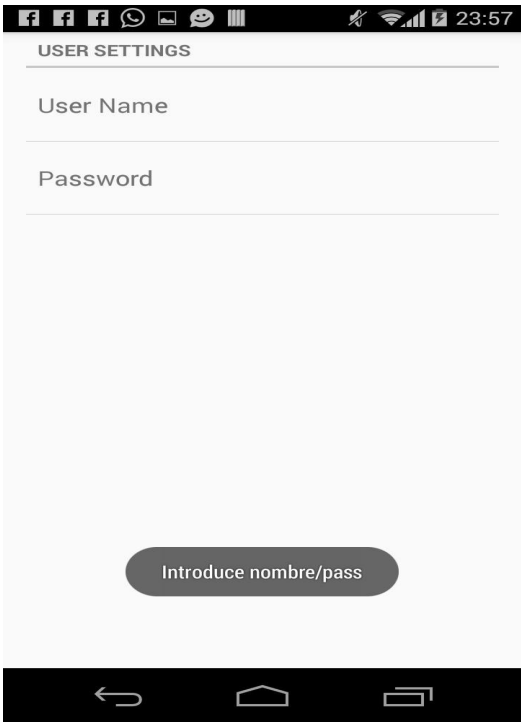
1. **Explique el siguiente código:**

```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >
   <PreferenceCategory android:title="User Settings">
      <EditTextPreference
         android:key="username"
         android:title="User Name"/>
      <EditTextPreference
         android:key="password"
         android:title="Password"/>
   </PreferenceCategory>
                  </PreferenceScreen>
```

9. **Ejecute de nuevo la aplicación presionando “Preferences” en el menú.**

El Método usado aqui para la pestaña preference es antiguo, pues a partir de la API 11, hay otra manera de crearla, por ejemplo, creamos una clase estática llamada ClaseEjemploPreference, que extienda PreferenceFragment. Y esta clase si tiene el método addPreferencesFromResource.
Y luego finalmente Creamos otra actividad y esat si hereda de AppCompatActivity con el método getFragmentManager().beginTransaction().replace(android.R.id.content_mispreferenceas, new ClaseEjemploPreference()).commit() y tendremos la vista que buscamos.

Solución:



Actividad 2

1. **Crea un proyecto con la vista puesta en la imagen.**
2. **Añade los métodos vistos en teoría según corresponda a los botones que se ven en la imagen. Acuérdese de configurar la memoria externa.**

Para mi mala suerte, en mi celular no poseo Sdcard. Por lo que no puedo crear el fichero externo, así que usaré la máquina virtual. (modifiqué texto de prueba para diferenciar en in y ex)
Probando fichero interno.

Probando fichero externo.

Ya que mi celular no posee memoria externa, por lo que tuve que prestarme, pues mi máquina virtual tampoco pudo montar la Sdcard.
Pruébalo si es que tienes SDCard y verás que si funciona :D

Actividad 3

1. **Creamos un nuevo proyecto en blanco.**
2. **Creamos la di siguiente interfaz gráfica.**

3. **Antes de trabajar con nuestro código Java de la actividad principal vamos a crear la clase para manejar la BBDD en Android. Por tanto creamos una clase que se llame *AdminSQLite.java* que extenderá de *SQLiteOpenHelper*. Explique lo que realiza.**

```
// es el constructor
public class AdminSQLite extends SQLiteOpenHelper {
   public AdminSQLite(Context context, String nombre,
          CursorFactory factory, int version) {
      super(context, nombre, factory, version);
   }
// Crea la base de datos, mediante conexion pasandole la sentencia Create table
   @Override
   public void onCreate(SQLiteDatabase db) {
      db.execSQL("create table votantes(dni integer primary key,
         nombre text, colegio text, nromesa integer)");
   }
// Actualiza la base de datos, y si existe, la borra antes.
   @Override
   public void onUpgrade(SQLiteDatabase db, int versionAnte,
         int versionNue) {
      db.execSQL("drop table if exists votantes");
      db.execSQL("create table votantes(dni integer primary key,
                  nombre text, colegio text, nromesa integer)");
   }
}
```

4. **Una vez creado el Java de la BBDD y el XML de la interfaz vamos a desarrollar el Java de la actividad principal.**
    1. **Lo primero es crear los atributos y llamar al XML de layout**

```
private EditText et1, et2, et3, et4;
private Cursor fila;
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   et1 = (EditText) findViewById(R.id.et_dni);
   et2 = (EditText) findViewById(R.id.et_datos);
   et3 = (EditText) findViewById(R.id.et_colegio);
   et4 = (EditText) findViewById(R.id.et_mesa);
}
```

    2. **Realizamos el *Button* "alta". Explique su contenido y la clase *ContentValues.***

```
public void alta(View v) {
    AdminSQLite admin =   new AdminSQLite(this,          define al admin
       "administracion", null, 1);
    SQLiteDatabase bd = admin.getWritableDatabase();     le da permiso de escritura a la db
    String dni = et1.getText().toString();                    extrae datos del editText
    String nombre = et2.getText().toString();                 extrae datos del editText
    String colegio = et3.getText().toString();                extrae datos del editText
    String nromesa = et4.getText().toString();                extrae datos del editText
    Cursor fila = bd.rawQuery("select * from votantes where dni="    Crea un cursor
       + dni, null);
    if(!fila.moveToFirst()) {                        evalúa si existe info dni de un elemento
       ContentValues registro = new ContentValues();   La clase ContentValues, funciona
       registro.put("dni", dni);                       Como un registro, donde por cada
       registro.put("nombre", nombre);                 etiqueta, le puedes añadir un conjunt
       registro.put("colegio", colegio);               de valores similar a una tabla.
       registro.put("nromesa", nromesa);
       bd.insert("votantes", null, registro);            Inserta los valores incluidos
       bd.close();                                       con etiqueta votantes, es decir,
       et1.setText("");                                  crea la base de datos,
       et2.setText("");
       et3.setText("");                                  et*.setText(“”), vacía los editText
       et4.setText("");
       Toast.makeText(this, "Datos actualizados",      Lanza Toast de modificación
          Toast.LENGTH_SHORT).show();
    } else {                                  Si ya existe, solo lanza toast y cierra la db
       bd.close();
       Toast.makeText(this, "Contacto existente",
          Toast.LENGTH_SHORT).show();
}
```

3. **Realizamos el *Button* “baja”. Explique su contenido.**

```
public void baja(View v) {
    AdminSQLite admin =      new AdminSQLite(this,        define al admin
"administracion", null, 1);
    SQLiteDatabase bd = admin.getWritableDatabase();   le da permiso de escritura a la db
    String dni = et1.getText().toString();                    obtiene dni text del EditText
    int cant = bd.delete("votantes", "dni=" + dni, null);   sentencia delete en la db con filtro dni
      bd.close();                                                    cierra db
      et1.setText("");                                               vacía contenido
      et2.setText("");
      et3.setText("");
      et4.setText("");
      if (cant == 1)                                                  si retorna 1, se eliminó bien.
        Toast.makeText(this,  "Borrado",
                Toast.LENGTH_SHORT).show();
      else                                                          si retorna otro valor, significa que no existe
        Toast.makeText(this, "No existe",
                Toast.LENGTH_SHORT).show();
}
```

4. **Realizamos el Button “consulta”. Explique su contenido.**

```
public void consulta(View v) {
      AdminSQLite admin =   new AdminSQLite(this,             define al admin
        "administracion", null, 1);
      SQLiteDatabase bd = admin.getWritableDatabase(); le da permiso de escritura a la db
      String dni = et1.getText().toString();
      Cursor fila = bd.rawQuery( "select nombre,colegio,nromesa
        from votantes where dni=" + dni, null);  Selecciona nombre, cole, y nromesa según dni
      if (fila.moveToFirst()) {                    Si existe, muestra los datos en los EditText
        et2.setText(fila.getString(0));
        et3.setText(fila.getString(1));
        et4.setText(fila.getString(2));
      } else                                          Si no existe lanza toast
        Toast.makeText(this, "No existe persona",
              Toast.LENGTH_SHORT).show();
      bd.close();
}
```

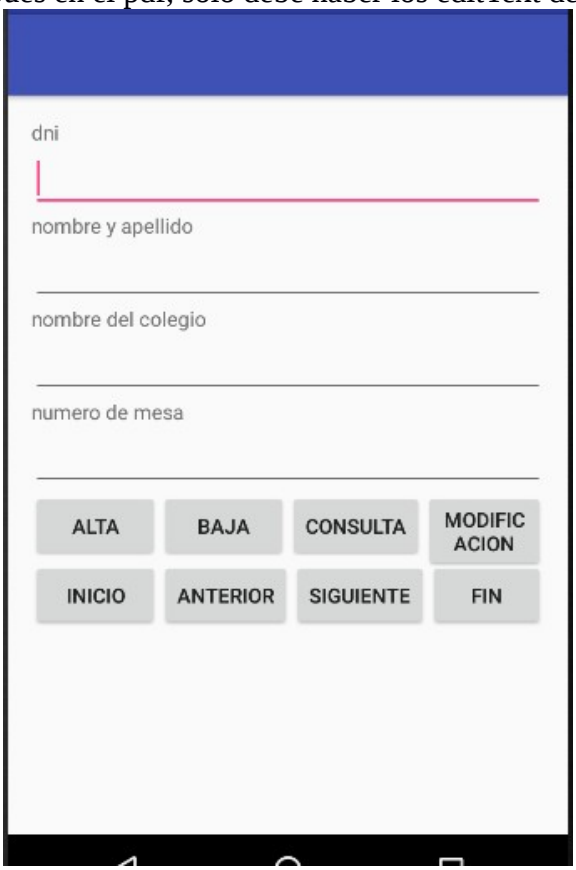
5. **Realizamos el *Button* “modificación”. Explique su contenido.**

```
public void modificacion(View v) {
      AdminSQLite admin =   new AdminSQLite(this,        define al admin
        "administracion", null, 1);
      SQLiteDatabase bd = admin.getWritableDatabase();  le da permiso de escritura a la db
      String dni = et1.getText().toString();    Obtiene info de los EditText
      String nombre = et2.getText().toString();
      String colegio = et3.getText().toString();
```

```
        String nromesa = et4.getText().toString();
        ContentValues registro = new ContentValues();  Crea unContentValues
        registro.put("nombre", nombre);        añade los valores de los String (que fueron tomado
        registro.put("colegio", colegio);      de los EditText)
        registro.put("nromesa", nromesa);
        int cant = bd.update("votantes", registro, "dni=" + dni, null);  actualiza la tabla votantes
        bd.close();          cierra la db                |                 del registro, según el dni
        if (cant == 1)              Si el resultado es 1, es correcto
           Toast.makeText(this, "Modificación realizada",        Lanza Toast
                   Toast.LENGTH_SHORT).show();
        else                        Si no lo es, retorna otro valor
           Toast.makeText(this, “No se encuentra",               Lanza Toast
                   Toast.LENGTH_SHORT).show();
}
```

Solución:

Vista, No tiene toolbar, pues en el pdf, solo debe haber los editText definidos.



Añadiendo información:(Mía)
Botón Alta

Si no existe muestra:

Si ya existe muestra:

Botón Baja:

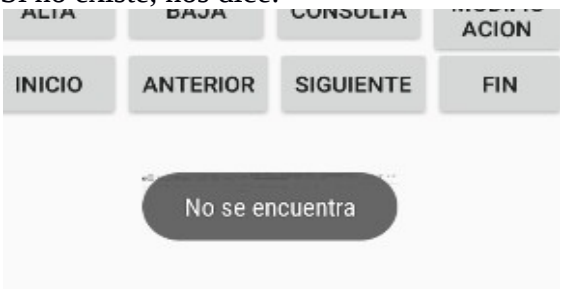
Si existe, es borrado:

Una vez borrado, o que ya no exista:

Botón modificación:

Si existe, se modifica.

Si no existe, nos dice:



Botón consulta:

Si no existe:

Si existe, nos rellena los demás campos.

Actividad 4

1. **En esta ampliación vamos a modifcar solamente nuestra clase Java principal.**
   1. **Realizamos el *Button* “inicio”. Explique su contenido.**

```
public void inicio(View view){
    AdminSQLite admin = new AdminSQLite(this,
        "administracion", null, 1);
    SQLiteDatabase bd = admin.getWritableDatabase();
    fila = bd.rawQuery("select * from votantes order by dni asc ",
        null);
    if (fila.moveToFirst()) {
       et1.setText(fila.getString(0));
       et2.setText(fila.getString(1));
       et3.setText(fila.getString(2));
       et4.setText(fila.getString(3));
    } else
       Toast.makeText(this, "No hay registrados" ,
            Toast.LENGTH_SHORT).show();
    bd.close();
}
```

Similar a los anteriores, pero este nos muestra los datos del primero de los datos.

   2. **Realizamos el *Button* “anterior”. Explique su contenido y el *try-catch*.**

```
public void anterior(View view){
    try {
       if (!fila.isFirst()) {
          fila.moveToPrevious();
          et1.setText(fila.getString(0));
          et2.setText(fila.getString(1));
          et3.setText(fila.getString(2));
          et4.setText(fila.getString(3));
       } else
          Toast.makeText(this, "Inicio de la tabla",
               Toast.LENGTH_SHORT).show();
    }catch (Exception e){
       e.printStackTrace();
    }
}
```

Similar al anterior, pero este muestra muestra el conjunto de datos anterior.

3. **Realizamos el *Button* “siguiente”. Explique su contenido.**

```
public void siguiente(View view){
    try {
       if (!fila.isLast()) {
          fila.moveToNext();
          et1.setText(fila.getString(0));
          et2.setText(fila.getString(1));
          et3.setText(fila.getString(2));
          et4.setText(fila.getString(3));
       } else
          Toast.makeText(this, "Llegó al final",
                Toast.LENGTH_SHORT).show();
    }catch (Exception e){
       e.printStackTrace();
    }
}
```

Similar al anterior, pero este muestra muestra el conjunto de datos siguiente.

4. **Realizamos el *Button* “fin”. Explique su contenido.**

```
public void fin(View view){
    AdminSQLite admin = new AdminSQLite(this,
          "administracion", null, 1);
    SQLiteDatabase bd = admin.getWritableDatabase();
    Cursor fila = bd.rawQuery(
          "select * from votantes order by dni asc ", null);
    if (fila.moveToLast()) {
       et1.setText(fila.getString(0));
       et2.setText(fila.getString(1));
       et3.setText(fila.getString(2));
       et4.setText(fila.getString(3));
    } else
       Toast.makeText(this, "No hay registros" ,
             Toast.LENGTH_SHORT).show();
       bd.close();
}
```

Similar a los demás, nos muerta el final.

5. **Realizamos el *Button* “onReset”. Explique su contenido.**

```
public void onReset(View view){
    et1.setText("");
    et2.setText("");
    et3.setText("");
    et4.setText("");
}
```

Simplemente, borra todo contenido de los EditText.

Ops, me olvidé de añadir el botón reset info, entonces la vista final con el botón y el método implementado es:

Solución:

Basta con crear 3 ejemplos, para verificar las funcionalidades.
Notar que al ordenar y aplicar las funcionalidades de anterior, siguiente, inicio y fin.
Los ordena por número de DNI.

Link del github con los códigos del laboratorio:

https://github.com/Jenazad/PDM/tree/master/Laboratorio_7

Referencias

http://www.sgoliver.net/blog/preferencias-en-android-ii-preferenceactivity/
http://developer.android.com/intl/es/reference/android/preference/PreferenceActivity.html
http://www.sgoliver.net/blog/ficheros-en-android-ii-memoria-externa-tarjeta-sd/
http://www.sgoliver.net/blog/ficheros-en-android-i-memoria-interna/