

## ENCUESTA SQLITE

Solo subiré a este pdf las capturas que tengan que ver con sqlite

### AdminSQLiteConexion

estás en aulasciclos2425

VO Víctor Oliver Donoso

```
1 package adaptador
2
3 import android.app.AlertDialog
4 import android.content.Context
5 import android.view.LayoutInflater
6 import android.view.ViewGroup
7 import androidx.recyclerview.widget.RecyclerView
8 import com.example.encuesta.R
9 import Auxiliar.Conexion // Asegúrate de importar tu clase Conexion
10 import android.widget.Toast
11
12 class AdaptadorPersonas(
13     private val encuestas: MutableList<String>,
14     private val context: Context // Pasamos el contexto para poder mostrar el AlertDialog
15 ) : RecyclerView.Adapter<ResumenPersonaViewHolder>() {
16
17     override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ResumenPersonaViewHolder {
18         val view = LayoutInflater.from(parent.context).inflate(R.layout.item_Lista, parent, attachToRoot: false)
19         return ResumenPersonaViewHolder(view)
20     }
21
22     override fun getItemCount(): Int {
23         return encuestas.size
24     }
25
26     override fun onBindViewHolder(holder: ResumenPersonaViewHolder, position: Int) {
27         val encuesta = encuestas[position]
28         holder.tvPersona.text = encuesta
29
30         // Configuración de un long click listener para mostrar el AlertDialog
```

```
estás en aulasciclos2425

Pixel 7 Pro API 35  app  72 ^ v

Translations Editor  AdaptadorPersonas.kt  Conexion.kt  MainActivity.kt  AdminSQLiteConexion.kt

12  class AdaptadorPersonas(
26      override fun onBindViewHolder(holder: ResumenPersonaViewHolder, position: Int) {
30          // Configuración de un long click listener para mostrar el AlertDialog
31          holder.itemView.setOnLongClickListener {
32              // Extraer el ID de la encuesta (suponiendo que el formato es "Encuesta #id")
33              val encuestaId = obtenerIdDeEncuesta(encuesta)
34
35              // Crear y mostrar el diálogo de confirmación
36              AlertDialog.Builder(context).apply {
37                  setTitle("Confirmación")
38                  setMessage("¿Seguro que desea eliminar esta encuesta?")
39                  setPositiveButton(text: "Si") { _, _ ->
40                      // Intentar eliminar de la base de datos
41                      try {
42                          val exito = Conexion.eliminarEncuesta(context, encuestaId)
43
44                          if (exito > 0) {
45                              // Si la eliminación fue exitosa, eliminar de la lista
46                              encuestas.removeAt(position)
47                              notifyItemRemoved(position)
48                              notifyItemRangeChanged(position, encuestas.size)
49
50                              // Mostrar un mensaje de éxito
51                              Toast.makeText(context, text: "Encuesta eliminada", Toast.LENGTH_SHORT).show()
52                          } else {
53                              // Mostrar mensaje de error si no se pudo eliminar de la base de datos
54                              Toast.makeText(context, text: "Error al eliminar de la base de datos", Toast.LENGTH_SHORT).show()
55                          }
56                      } catch (e: Exception) {
57                          // Handle database or other exceptions
58                      }
59                  }
60              }
61          }
62      }
63  }
```

```
Translations Editor  AdaptadorPersonas.kt  Conexion.kt  MainActivity.kt  AdminSQLiteConexion.kt  72 ^ v
```

```
12 class AdaptadorPersonas(  
26     override fun onBindViewHolder(holder: ResumenPersonaViewHolder, position: Int) {  
31         holder.itemView.setOnLongClickListener {  
36             AlertDialog.Builder(context).apply {  
39                 setPositiveButton(text: "Si") { _, _ ->  
56                     } catch (e: Exception) {  
57                         // Handle database or other exceptions  
58                         Toast.makeText(context, text: "Error: ${e.message}", Toast.LENGTH_SHORT).show()  
59                     }  
60                 }  
61                 setNegativeButton(text: "Cancelar", listener: null)  
62             }.show()  
63             true // Indica que el evento ha sido manejado  
64         }  
65     }  
66  
67     // Método para obtener el ID de la encuesta  
68     private fun obtenerIdDeEncuesta(encuesta: String): Int {  
69         // Aquí suponemos que el ID está incluido en el texto de la encuesta  
70         // Ejemplo: "Encuesta #123", entonces extraemos el número  
71         return encuesta.substringAfter(delimiter: "#").toIntOrNull() ?: -1 // Devuelve -1 si no puede extraer  
72     }  
73 }  
74
```

## Clase Persona

estás en aulasciclos2425

Pixel 7 Pro API 35 app

AdaptadorPersonas.kt Persona.kt Conexion.kt MainActivity.kt AdminSQLiteConexion.kt

```
1 package Modelo
2
3
4
5 data class Persona(
6
7     var nombre: String,
8     var sistemaOperativo: String,
9     var especializacion: ArrayList<String>,
10    var horasEstudio: Int
11 ) {
12     override fun toString(): String {
13         return "Persona(Nombre='$nombre', sistemaOperativo='$sistemaOperativo', " +
14             "especializacion=$especializacion, horasEstudio=$horasEstudio)"
15     }
16 }
17
18
19
20
21
22
23
24
25
```

5:12 CRLF UTF-8 4 spaces

## Conexión

VO Víctor Oliver Donoso

```
estás en aulasciclos2425

Pixel 7 Pro API 35
app
Conexion.kt
MainActivity.kt
AdminSQLiteConexion.kt

package Auxiliar

import ...

object Conexion {

    private const val DATABASE_NAME = "Encuesta.db3"
    private const val DATABASE_VERSION = 1

    // Método para añadir una persona a la base de datos
    fun addPersona(context: Context, p: Persona): Long {
        // Abrir la base de datos en modo escritura
        val admin = AdminSQLiteConexion(context, DATABASE_NAME, factory: null, DATABASE_VERSION)
        val bd = admin.writableDatabase

        // Crear un objeto ContentValues para insertar los datos
        val registro = ContentValues()
        registro.put("nombre", p.nombre)
        registro.put("sistemaOperativo", p.sistemaOperativo)
        registro.put("especializacion", p.especializacion.joinToString(separator: ", ")) // Unir la lista de
        registro.put("horasEstudio", p.horasEstudio)

        // Insertar los datos en la tabla "personas" y devolver el ID insertado
        val codigo = bd.insert(table: "personas", nullColumnHack: null, registro)

        // Cerrar la base de datos
        bd.close()

        return codigo
    }
}
```

harEncuesta: Int 68:6 CRLF UTF-8 4 spaces



VO Víctor Oliver Donoso ▾

estás en aulasciclos2425

Pixel 7 Pro API 35 ▾

app ▾



AdaptadorPersonas.kt

Persona.kt

Conexion.kt ×

MainActivity.kt

AdminSQLiteConexion.kt ▾



```
object Conexion {
    fun addPersona(context: Context, p: Persona): Long {
    }

    // Método para obtener todas las personas desde la base de datos
    @SuppressWarnings("Range")
    fun obtenerPersonas(context: Context): List<Persona> {
        val admin = AdminSQLiteConexion(context, DATABASE_NAME, factory: null, DATABASE_VERSION)
        val bd = admin.readableDatabase

        val cursor = bd.rawQuery(sql: "SELECT * FROM personas", selectionArgs: null)
        val personas = mutableListOf<Persona>()

        if (cursor.moveToFirst()) {
            do {
                val nombre = cursor.getString(cursor.getColumnIndex("nombre"))
                val sistemaOperativo = cursor.getString(cursor.getColumnIndex("sistemaOperativo"))
                val especializacion = cursor.getString(cursor.getColumnIndex("especializacion")).split(" ")
                val horasEstudio = cursor.getInt(cursor.getColumnIndex("horasEstudio"))

                personas.add(Persona(nombre, sistemaOperativo, ArrayList(especializacion), horasEstudio))
            } while (cursor.moveToNext())
        }

        cursor.close()
        bd.close()

        return personas
    }
}
```



VO Víctor Oliver Donoso

estás en aulasciclos2425

Pixel 7 Pro API 35

app



AdaptadorPersonas.kt

Persona.kt

Conexion.kt

MainActivity.kt

AdminSQLiteConexion.kt

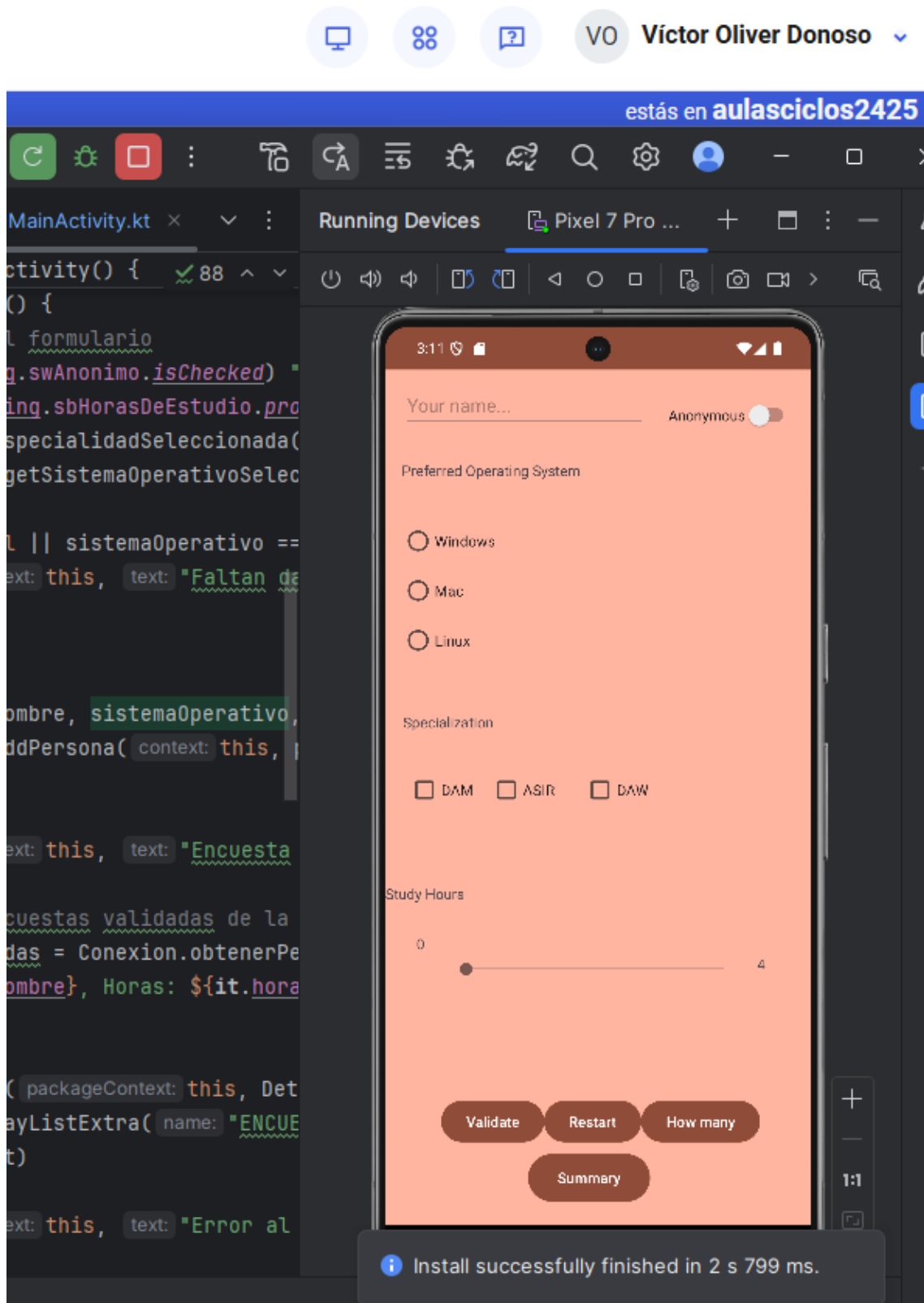
```
object Conexion {
    fun obtenerPersonas(context: Context): List<Persona> {
        val horasEstudio = cursor.getInt(cursor.getColumnIndex("horasEstudio"))
        personas.add(Persona(nombre, sistemaOperativo, ArrayList(especializacion), horasEstudio))
    } while (cursor.moveToNext())
}

cursor.close()
bd.close()

return personas
}

fun eliminarEncuesta(context: Context, encuestaId: Int): Int {
    val admin = AdminSQLiteConexion(context, DATABASE_NAME, factory: null, DATABASE_VERSION)
    val db = admin.writableDatabase
    val rowsDeleted = db.delete(table: "personas", whereClause: "Id=?", arrayOf(encuestaId.toString()))
    db.close()
    return rowsDeleted
}
```

Funcionamiento, sin introducir ninguna encuesta, solo con las que se guardaron







VO Víctor Oliver Donoso ▾

estás en **aulasciclos2425**

