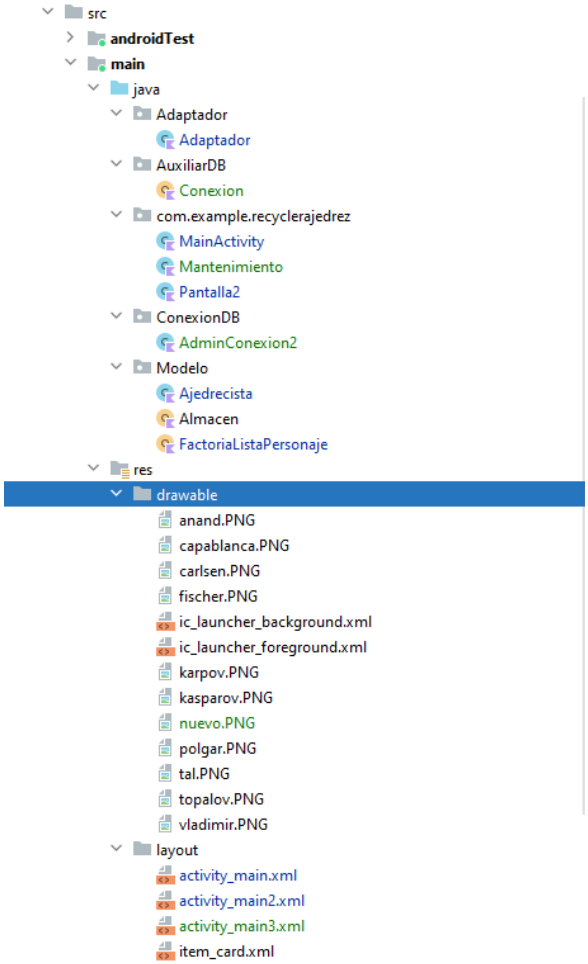



```
object Conexion {

    private  var DATABASE_NAME = "administracion4.db3"
    private  var DATABASE_VERSION = 1


    new *
    fun cambiarBD(nacionalidadBD:String){
        this.DATABASE_NAME = nacionalidadBD
    }

    new *
    fun addAjedrecista(contexto: AppCompatActivity, p: Ajedrecista):Long{
        val admin = AdminConexion2(contexto, this.DATABASE_NAME,  factory: null, DATABASE_VERSION)
        val bd = admin.writableDatabase //habilito la BBDD para escribir en ella, también deja leer.
        val registro = ContentValues() //objeto de kotlin, contenido de valores, un Map. Si haceis ctrl+clic lo veis.
        registro.put("nombre", p.nombre)
        registro.put("elo",p.elo)
        registro.put("nacionalidad", p.nacionalidad)
        val codigo = bd.insert( table: "personas",  nullColumnHack: null, registro)
        bd.close()
        return codigo
    }

    new *
    fun delAjedrecista(contexto: AppCompatActivity, nombre: String):Int{
        val admin = AdminConexion2(contexto, this.DATABASE_NAME,  factory: null, DATABASE_VERSION)
        val bd = admin.writableDatabase

        val cant = bd.delete( table: "personas",  whereClause: "nombre=?", arrayOf(nombre))
        bd.close()
        return cant
    }
```

```
new *
fun modAjedrecista(contexto: AppCompatActivity, nombre:String, p:Ajedrecista):Int {
    val admin = AdminConexion2(contexto, this.DATABASE_NAME,  factory: null, DATABASE_VERSION)
    val bd = admin.writableDatabase
    val registro = ContentValues()
    registro.put("elo",p.elo.toString())
    registro.put("nacionalidad", p.nacionalidad)


    val cant = bd.update( table: "personas", registro,  whereClause: "nombre=?", arrayOf(nombre.toString()))

    bd.close()
    return cant
}

new *
fun buscarAjedrecista(contexto: AppCompatActivity, nombre:String):Ajedrecista? {
    var p:Ajedrecista? = null //si no encuentra ninguno vendrá null, por eso la ? y también en la devolución de la función.
    val admin = AdminConexion2(contexto, this.DATABASE_NAME,  factory: null, DATABASE_VERSION)
    val bd = admin.readableDatabase

    val fila =bd.rawQuery(
         sql: "SELECT elo,nacionalidad FROM personas WHERE nombre=?",
        arrayOf(nombre.toString())
    )
    //en fila viene un CURSOR, que está justo antes del primero por eso lo ponemos en el primero en la siguiente linea
    if (fila.moveToFirst()) {//si no hay datos el moveToFirst, devuelve false, si hay devuelve true.
        p = Ajedrecista(nombre,fila.getString( columnIndex: 0),nombre,fila.getString( columnIndex: 1),  mundiales: 0,  rankingFIFE: 0,  estilo: "0",  rivalidades: "",
    }
    bd.close()
    return p
}
```

```
new *
fun obtenerAjedrecistas(contexto: AppCompatActivity):ArrayList<Ajedrecista>{
    var Ajedrecistas:ArrayList<Ajedrecista> = ArrayList( initialCapacity: 1)

    val admin = AdminConexion2(contexto, this.DATABASE_NAME,  factory: null, DATABASE_VERSION)

    val bd = admin.readableDatabase
    val fila = bd.rawQuery( sql: "select nombre,elo,nacionalidad from personas",  selectionArgs: null)
    while (fila.moveToNext()) {

        var imagen="@drawable/"+fila.getString( columnIndex: 0).toString().toLowerCase()

        var p:Ajedrecista = Ajedrecista(fila.getString( columnIndex: 0),fila.getString( columnIndex: 1),imagen,fila.getString( columnIndex: 2),
        Ajedrecistas.add(p)
    }
    bd.close()
    return Ajedrecistas //este arrayList lo puedo poner en un adapter de un RecyclerView por ejemplo.
}
```

```
override fun onCreate(savedInstanceState: Bundle?) {

    try {

        contextoPrincipal=this

        super.onCreate(savedInstanceState)

        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        Almacen.ajedrecistas = Conexion.obtenerAjedrecistas( contexto: this)
```

```
binding.bMantenimiento.setOnClickListener(){ it: View!

    for (a in Conexion.obtenerAjedrecistas( contexto: this)){
        Conexion.delAjedrecista( contexto: this, a.nombre)
    }

    for (a in Almacen.ajedrecistas){
        Conexion.addAjedrecista( contexto: this,a)
    }

    var inte:Intent = Intent(contextoPrincipal,Mantenimiento::class.java)
    ContextCompat.startActivity(contextoPrincipal,inte, options: null)
}

binding.bReiniciar.setOnClickListener(){ it: View!

    Almacen.ajedrecistas=FactoriaListaPersonaje.generaLista( cant: 12)

    for (a in Almacen.ajedrecistas){
        Conexion.addAjedrecista( contexto: this,a)
    }

    miRecyclerView = binding.listaRecycler
    miRecyclerView.setHasFixedSize(true)
    miRecyclerView.layoutManager = LinearLayoutManager( context: this)

    var miAdapter = Adaptador(Almacen.ajedrecistas,  contexto: this)

    miRecyclerView.adapter = miAdapter


}
```

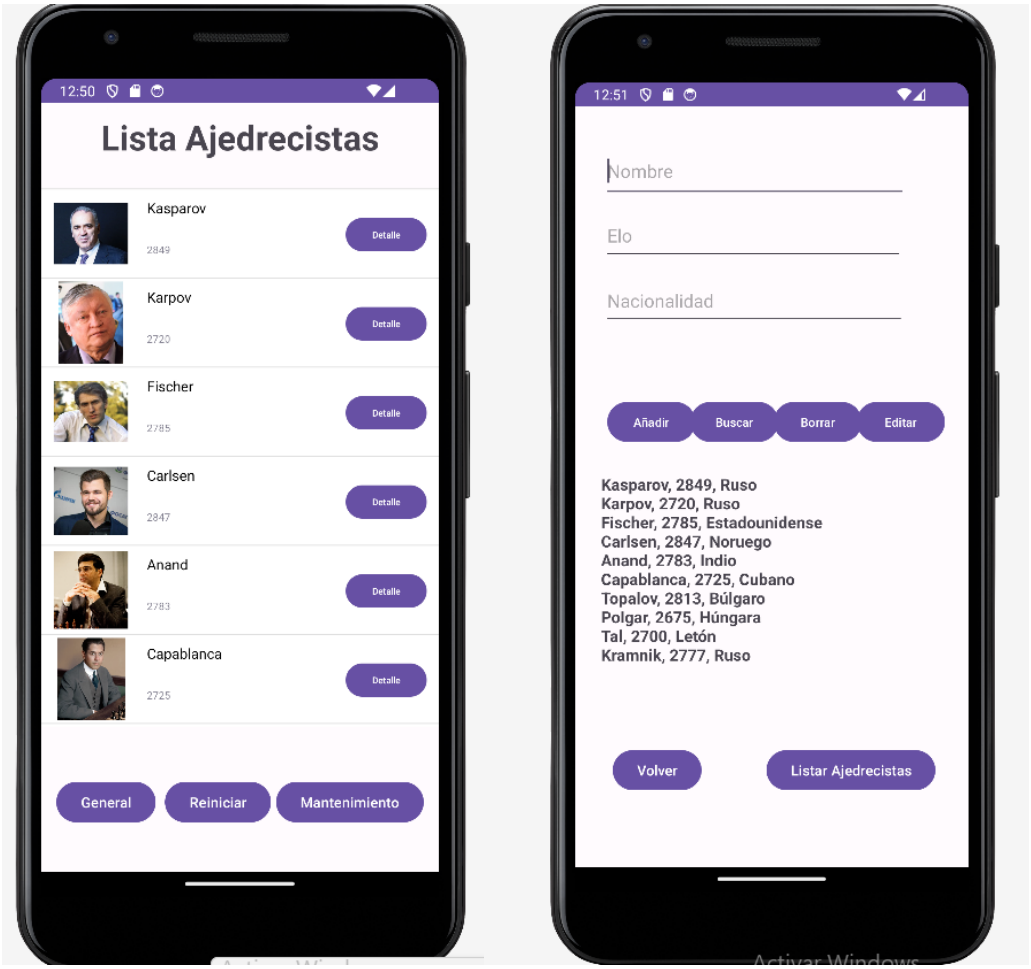
```
new *
class Mantenimiento : AppCompatActivity() {
    lateinit var edNombre: EditText
    lateinit var edElo: EditText
    lateinit var botonAdd: Button
    lateinit var botonBuscar: Button
    lateinit var botonBorrar: Button
    lateinit var botonEditar: Button
    lateinit var txtListdo: TextView
    lateinit var edNacionalidad:EditText
    lateinit var botonVolver:Button


    new *
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        Log.e( tag: "ACSCO", msg: "Paso por OnCreate ")
        setContentView(R.layout.activity_main3)
        edNombre = findViewById(R.id.tb_nombre)
        edNombre.requestFocus()
        edElo = findViewById(R.id.tb_elo)
        edNacionalidad=findViewById(R.id.tb_nacionalidad)
        botonAdd = findViewById(R.id.b_anadir)
        botonBuscar = findViewById(R.id.b_buscar)
        botonBorrar = findViewById(R.id.b_borrar)
        botonEditar = findViewById(R.id.b_editar)
        txtListdo = findViewById(R.id.tv_listado)
        botonVolver =findViewById(R.id.b_volverMantenimiento)


    fun addAjedrecista(view: View) {
        if (edNombre.text.toString().trim().isEmpty() || edElo.text.toString().trim().isEmpty()
            || edNacionalidad.text.toString().trim().isEmpty()){
            val builder = AlertDialog.Builder( context: this)
            with(builder)
            { this: AlertDialog.Builder
                setTitle("Faltan campos")
                setMessage("Rellena todos los campos antes de agregar")
                setPositiveButton( text: "Aceptar", DialogInterface.OnClickListener(function = positiveButtonClick))
                show() ^with
            }
        }
        else {

            var imagen="@drawable/"+"nuevo"

            var pers: Ajedrecista = Ajedrecista(
                edNombre.getText().toString(),
                edElo.getText().toString(),
                imagen,
                edNacionalidad.getText().toString(),
                 mundiales: 0, rankingFIFE: 0, estilo: "null", rivalidades: "null", cusiosidades: "null", porcentajeVictorias: 0
            )
            var codigo= Conexion.addAjedrecista( contexto: this, pers)
            edNacionalidad.setText("")
            edNombre.setText("")
            edElo.setText("")
            edNombre.requestFocus()
            //la L es por ser un Long lo que trae codigo.
            if(codigo!=-1L) {
                val builder = AlertDialog.Builder( context: this)
                with(builder)
                { this: AlertDialog.Builder
```

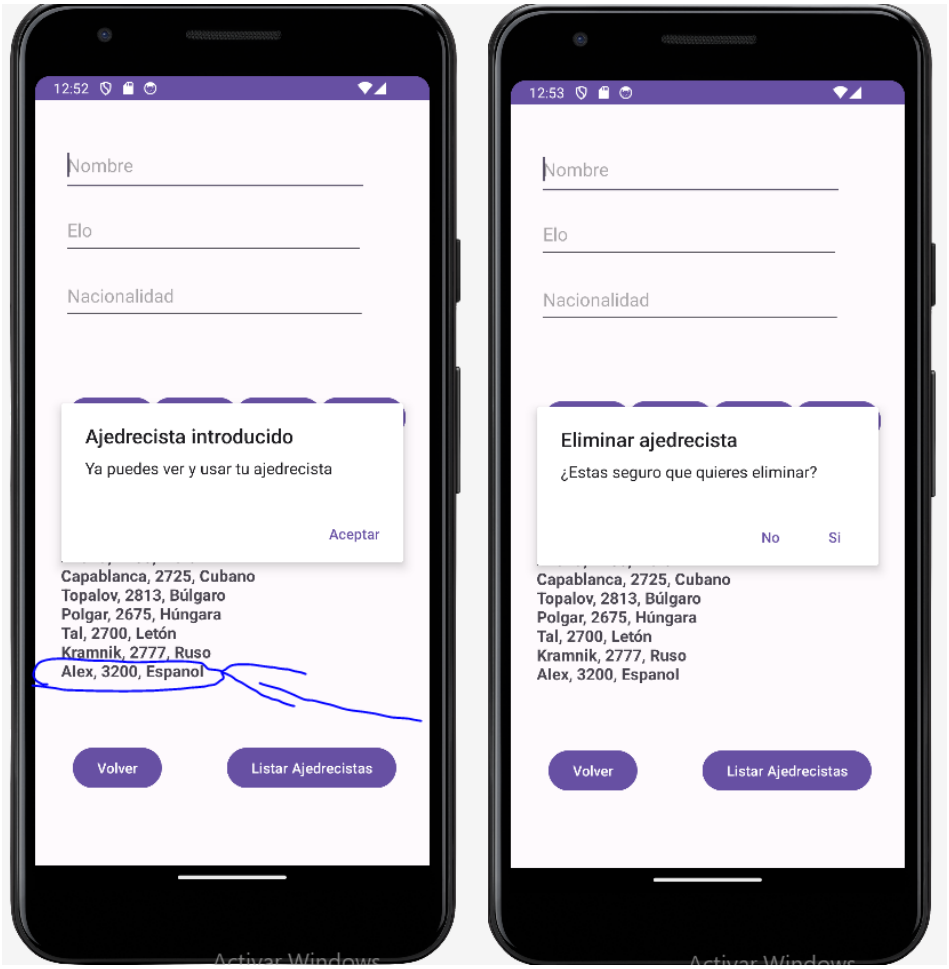
```
fun delAjedrecista(view: View) {

    val contexto=this

    var cant = Conexion.buscarAjedrecista( contexto: this, edNombre.text.toString())
    edNombre.setText("")
    edNacionalidad.setText("")
    edElo.setText("")
    if (cant != null) {
        val builder = AlertDialog.Builder( context: this)

        with(builder)
        { this: AlertDialog.Builder
            setTitle("Eliminar ajedrecista")
            setMessage("¿Estas seguro que quieres eliminar?")
            //Otra forma es definir directamente aquí lo que se hace cuando se pulse.
            setPositiveButton( text: "Si", DialogInterface.OnClickListener(function = { dialog: DialogInterface, which: Int ->
                Conexion.delAjedrecista(contexto,cant.nombre)
            }))
            setNegativeButton( text: "No", ({ dialog: DialogInterface, which: Int ->
                Toast.makeText(applicationContext,
                    text: "Has pulsado no", Toast.LENGTH_SHORT).show()
            }))
            show() ^with //builder.show()
        }
    }
    else
        Toast.makeText( context: this,  text: "No existe ese ajedrecista", Toast.LENGTH_SHORT).show()

}
```

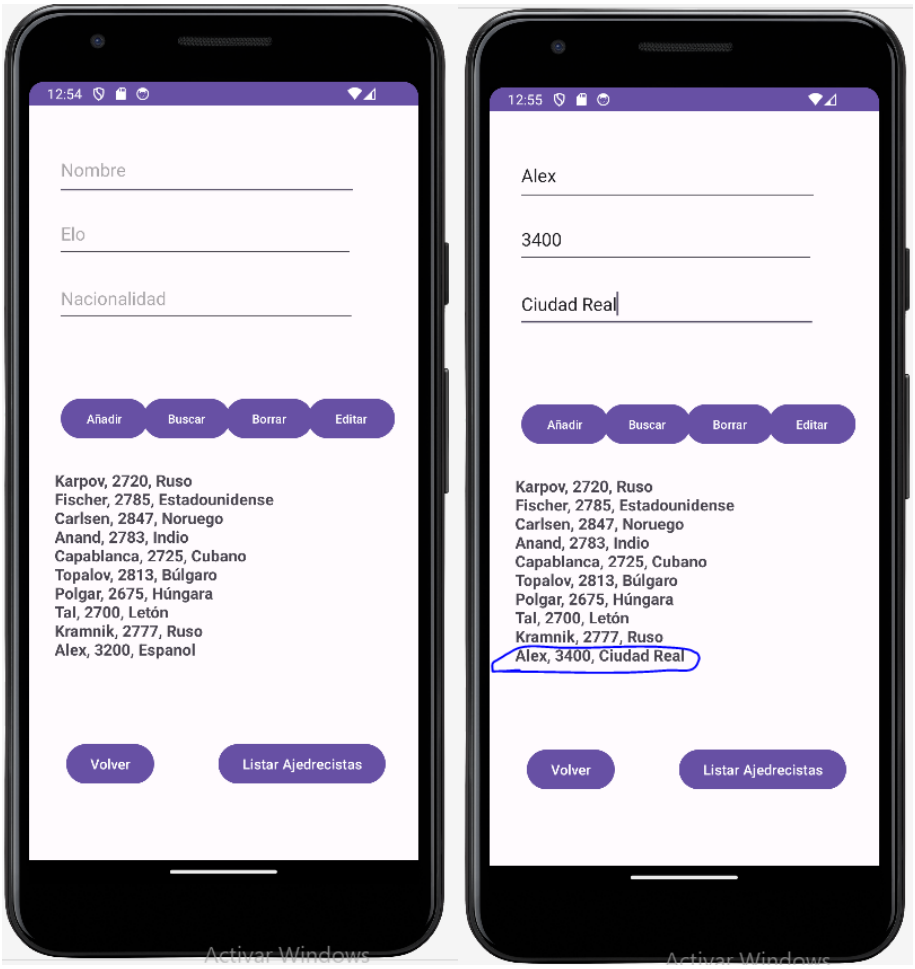
```
new *
fun modAjedrecista(view: View) {
    if (edNacionalidad.text.toString().trim().isEmpty()|| edNombre.text.toString().trim().isEmpty()
        || edElo.text.toString().trim().isEmpty()){
        val builder = AlertDialog.Builder( context: this)
        with(builder)
        { this: AlertDialog.Builder
            setTitle("Faltan campos")
            setMessage("Rellena todos los campos antes de agregar")
            setPositiveButton( text: "Aceptar", DialogInterface.OnClickListener(function = positiveButtonClick))
            show() ^with
        }
    }
    else {
        var pers: Ajedrecista = Ajedrecista(
            edNombre.getText().toString(),
            edElo.getText().toString(),
            edNombre.getText().toString(),
            edNacionalidad.getText().toString(),
             mundiales: 0, rankingFIFE: 0, estilo: "null", rivalidades: "null", cusiosidades: "null", porcentajeVictorias: 0
        )
        var cant = Conexion.modAjedrecista( contexto: this, edNombre.text.toString(), pers)
        if (cant == 1)
            Toast.makeText( context: this,  text: "Se modificaron los datos", Toast.LENGTH_SHORT).show()
        else
            Toast.makeText( context: this,  text: "No existe una Ajedrecista con ese DNI", Toast.LENGTH_SHORT).show()
    }
    listarAjedrecistas(view)
}
```

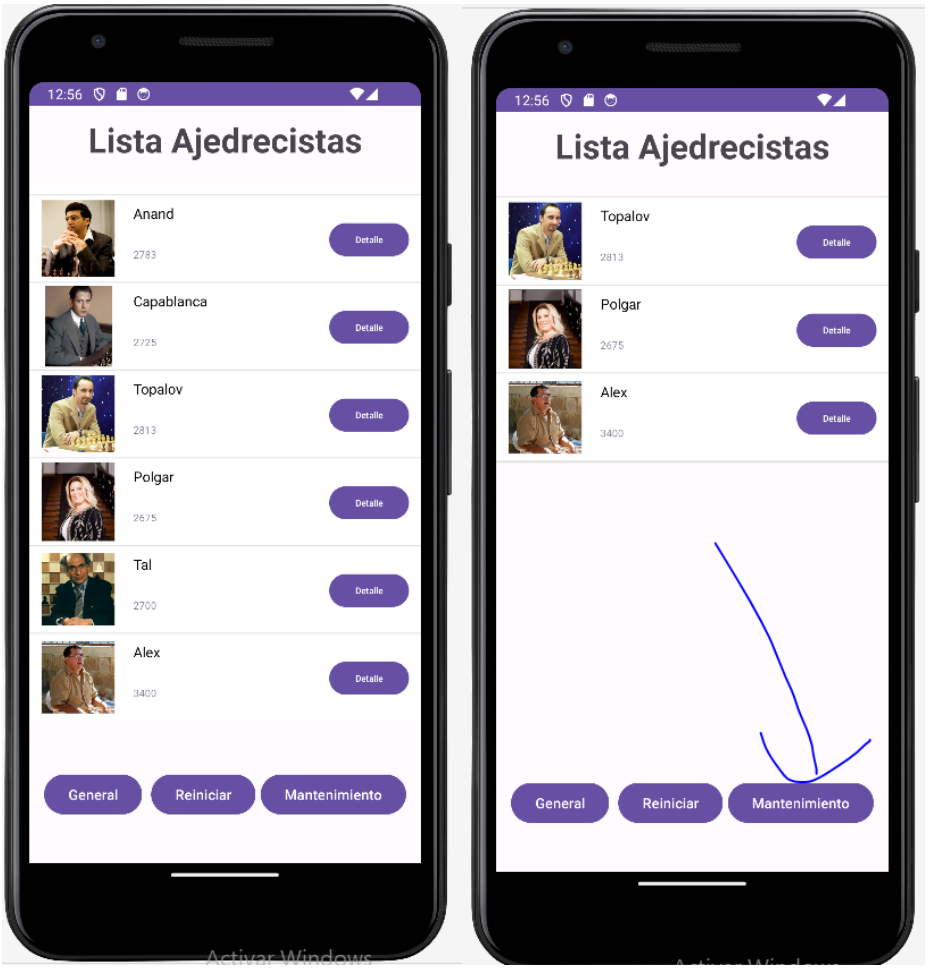
```
new *
fun buscarAjedrecista(view: View) {
    var p:Ajedrecista? = null
    p = Conexion.buscarAjedrecista( contexto: this, edNombre.text.toString())
    if (p!=null) {
        edNombre.setText(p.nombre)
        edElo.setText(p.elo)
        edNacionalidad.setText(p.nacionalidad)
    } else {
        Toast.makeText( context: this,  text: "No existe ese ajedrecista", Toast.LENGTH_SHORT).show()
    }

}

new *
fun listarAjedrecistas(view: View) {
    var listado:ArrayList<Ajedrecista> = Conexion.obtenerAjedrecistas( contexto: this)

    txtListdo.setText("")

    if (listado.size==0) {
        Toast.makeText( context: this,  text: "No existen datos en la tabla", Toast.LENGTH_SHORT).show()
    }
    else {
        for(p in listado){
            var cadena = p.nombre + ", " + p.elo + ", " + p.nacionalidad + "\r\n"
            txtListdo.append(cadena)
        }
    }
}
```

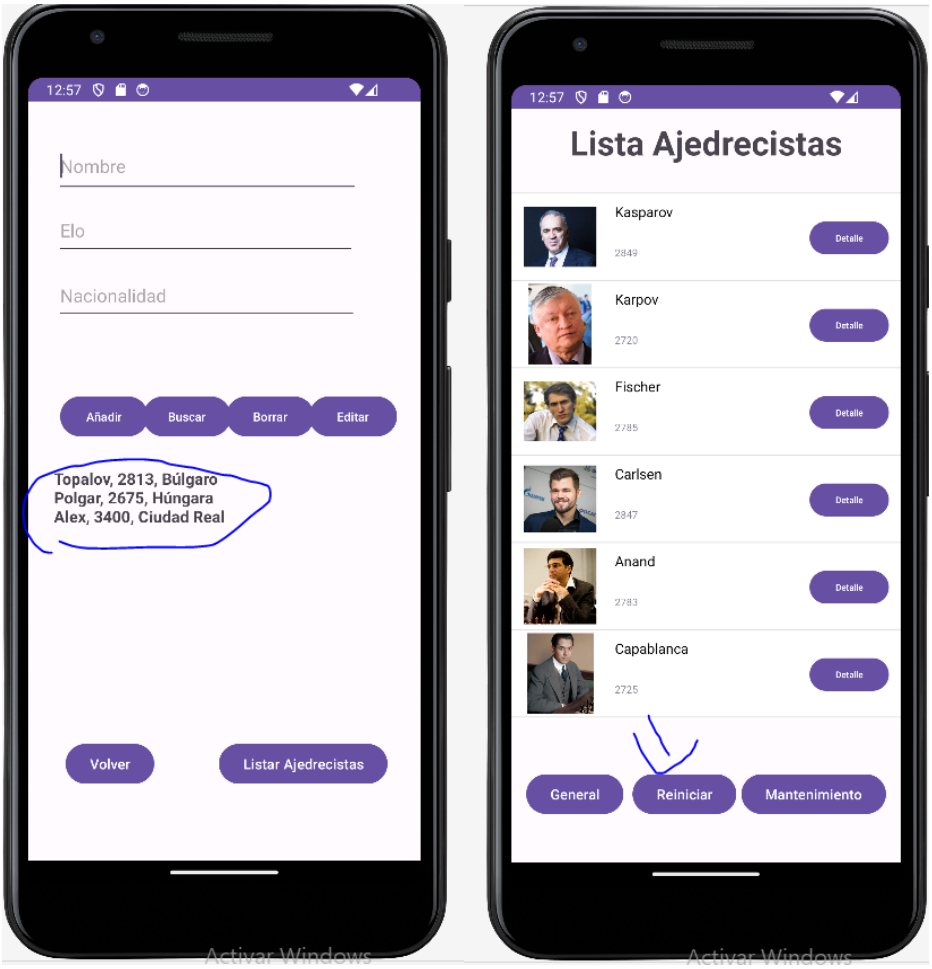
```
new *
class AdminConexion2(context: Context, name: String, factory: SQLiteDatabase.CursorFactory?, version: Int) : SQLiteOpenHelper(context, name,
    new *
    override fun onCreate(db: SQLiteDatabase) {
        Log.e( tag: "ACSCO",  msg: "Paso por OnCreate del AdminSQLLite")
        db.execSQL( sql: "DROP TABLE IF EXISTS personas")
        db.execSQL( sql: "create table personas(nombre text primary key, elo text, nacionalidad text)")
    }

    new *
    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
        Log.e( tag: "ACSCO",  msg: "Paso por OnUpgrade del AdminSQLLite")

        // Realiza cambios en la estructura de la tabla si es necesario
        db.execSQL( sql: "DROP TABLE IF EXISTS personas")
        db.execSQL( sql: "create table personas(nombre text primary key, elo text, nacionalidad text)")
    }
}
```


12:50
Lista Ajedrecistas
Kasparov
2849
Detalle
Karpov
2720
Detalle
Fischer
2785
Detalle
Carlsen
2847
Detalle
Anand
2783
Detalle
Capablanca
2725
Detalle
General
Reiniciar
Mantenimiento
12:51
Nombre
Elo
Nacionalidad
Añadir
Buscar
Borrar
Editar
Kasparov, 2849, Ruso
Karpov, 2720, Ruso
Fischer, 2785, Estadounidense
Carlsen, 2847, Noruego
Anand, 2783, Indio
Capablanca, 2725, Cubano
Topalov, 2813, Búlgaro
Polgar, 2675, Húngara
Tal, 2700, Letón
Kramnik, 2777, Ruso
Volver
Listar Ajedrecistas


12:52
Nombre
Elo
Nacionalidad
Ajedrecista introducido
Ya puedes ver y usar tu ajedrecista
Aceptar
Capablanca, 2725, Cubano
Topalov, 2813, Búlgaro
Polgar, 2675, Húngara
Tal, 2700, Letón
Kramnik, 2777, Ruso
Alex, 3200, Espanol
Volver
Listar Ajedrecistas
12:53
Nombre
Elo
Nacionalidad
Eliminar ajedrecista
¿Estas seguro que quieres eliminar?
No
Si
Capablanca, 2725, Cubano
Topalov, 2813, Búlgaro
Polgar, 2675, Húngara
Tal, 2700, Letón
Kramnik, 2777, Ruso
Alex, 3200, Espanol
Volver
Listar Ajedrecistas


12:54
Nombre
Elo
Nacionalidad
Añadir
Buscar
Borrar
Editar
Karpov, 2720, Ruso
Fischer, 2785, Estadounidense
Carlsen, 2847, Noruego
Anand, 2783, Indio
Capablanca, 2725, Cubano
Topalov, 2813, Búlgaro
Polgar, 2675, Húngara
Tal, 2700, Letón
Kramnik, 2777, Ruso
Alex, 3200, Espanol
Volver
Listar Ajedrecistas
12:55
Alex
3400
Ciudad Real
Añadir
Buscar
Borrar
Editar
Karpov, 2720, Ruso
Fischer, 2785, Estadounidense
Carlsen, 2847, Noruego
Anand, 2783, Indio
Capablanca, 2725, Cubano
Topalov, 2813, Búlgaro
Polgar, 2675, Húngara
Tal, 2700, Letón
Kramnik, 2777, Ruso
Alex, 3400, Ciudad Real
Volver
Listar Ajedrecistas


12:56
Lista Ajedrecistas
Anand
2783
Detalle
Capablanca
2725
Detalle
Topalov
2813
Detalle
Polgar
2675
Detalle
Tal
2700
Detalle
Alex
3400
Detalle
General
Reiniciar
Mantenimiento
12:56
Lista Ajedrecistas
Topalov
2813
Detalle
Polgar
2675
Detalle
Alex
3400
Detalle
General
Reiniciar
Mantenimiento


12:57
Nombre
Elo
Nacionalidad
Añadir
Buscar
Borrar
Editar
Topalov, 2813, Búlgaro
Polgar, 2675, Húngara
Alex, 3400, Ciudad Real
Volver
Listar Ajedrecistas
12:57
Lista Ajedrecistas
Kasparov
2849
Detalle
Karpov
2720
Detalle
Fischer
2785
Detalle
Carlsen
2847
Detalle
Anand
2783
Detalle
Capablanca
2725
Detalle
General
Reiniciar
Mantenimiento