I.E.S. EL CAÑAVERAL

CryptoHub

2º DAM B

PROGRAMACIÓN MULTIMEDIA

Y

DISPOSITIVOS MÓVILES

EVAL 1º

Apellidos y Nombre del Alumnos/as

Mandar Gil, Victor Aaron

Mandar Gil,
Victor Aaron

Indice

Mandar Gil,
Victor Aaron

Descripción:

La aplicación se especializa en la compra y venta de criptomonedas, destacando por su capacidad de registro de usuarios y almacenamiento de sus datos. La clase Main permite el inicio de sesión de la aplicación La clase Registro gestiona el ingreso de nuevos usuarios, garantizando la exclusividad de nombres de usuario. La persistencia de datos se logra mediante la clase SQLAd que establece y actualiza la estructura de la base de datos SQLite. La clase Fondos facilita la gestión de saldos, permitiendo a los usuarios añadir fondos de manera sencilla. Por ultima la clase Inicio permite Visualizar los datos de los usuarios así como comprar y vender criptomonedas.

Elementos:

- Main:
  - LinearLayout
  - Etiquetas
  - Campos de entrada de texto
  - Botones
  - Etiqueta con link dentro del proyecto
  - Consulta de datos
- Inicio
  - TableLayout
  - Etiquetas
  - Botones
  - Imágenes
  - Etiquetas con valores dinámicos
  - Consulta de datos
  -
- Registro:
  - Etiquetas
  - Campos de entrada de texto
  - Botón
  - Consulta de base de datos
  - Inserción de datos
  - Actualización de datos
  - Expresión when
- Fondos:
  - TableLayout
  - Etiqueta
  - Campo de entrada de texto
  - Botones
  - Actualización de datos

Dificultades:

A lo largo del desarrollo he tenido problemas al actualizar la estructura de la base de datos. Siempre que cambiaba algo, como las tablas el nombre de algun campo o la cantidad de campos tuve que forzar el borrado de la base de datos. Improvise el siguiente codigo para hacer el apaño:

```
fun deleteDatabase(context: Context): Boolean{
   return context.deleteDatabase("Cryptos")
}
```

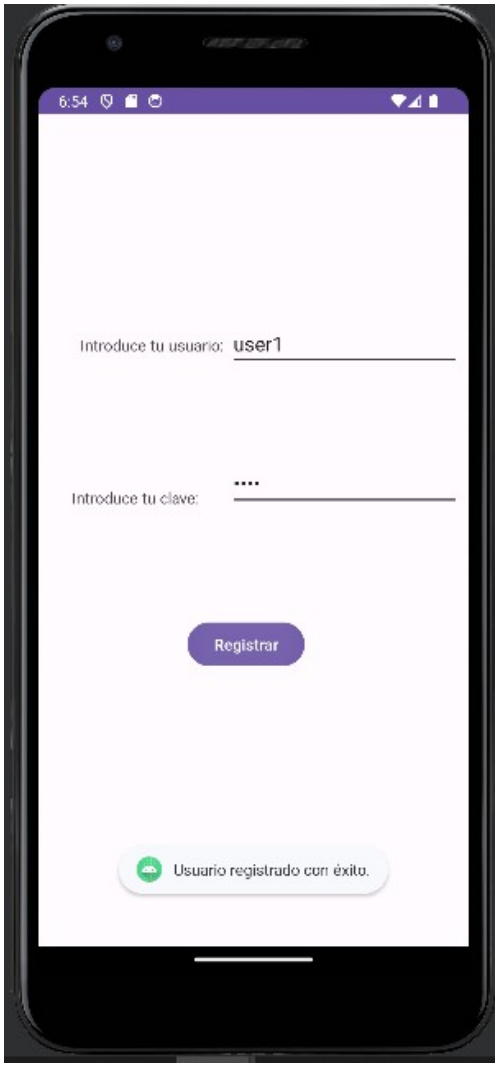
Esta funcion la incorporaba en la clase SQLAd y la ejecutaba como primera orden en el Main para que purgase la base de datos.

Fuera de esto no he encontrado mayores dificultades.

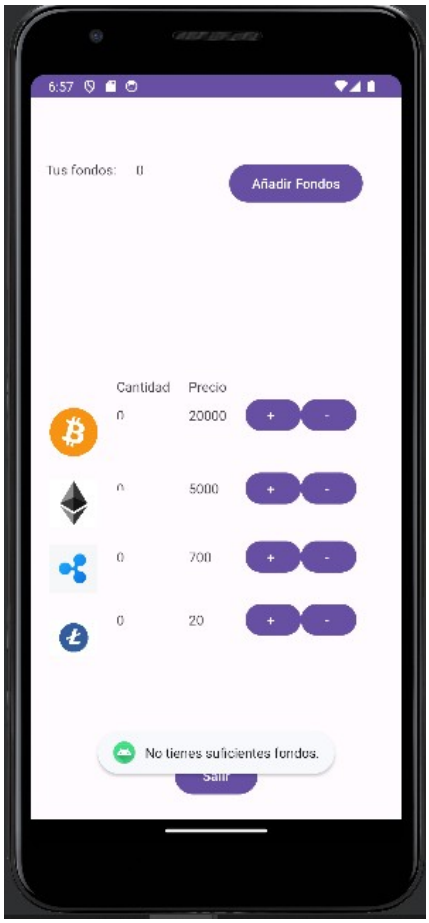
Pantallazos:

Intento de inicio de sesion:

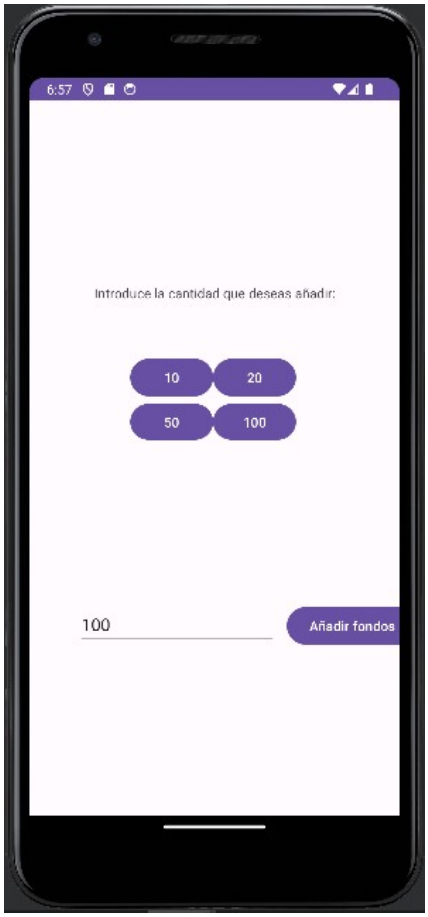
Registro de usuario:



Pantalla de inicio y mensaje de error de fondos:



Añadiendo fondos:



Codigo Relevante:

Main:

Evento del boton Entrar:

```
bttnEntrar.setOnClickListener { it: View!
    val admin = SQLAd( context: this,  name: "Cryptos",  factory: null,  version: 1)
    val bd = admin.readableDatabase
    val usuario = edtUser.text.toString()
    val clave = edtPass.text.toString()

    val cursor = bd.query( table: "usuario", arrayOf("user", "clave"),
        selection: "user = ? AND clave = ?", arrayOf(usuario, clave),
        groupBy: null,  having: null,  orderBy: null)

    if (cursor.moveToFirst()) {
        val intento = Intent( packageContext: this, Inicio::class.java)
        intento.putExtra( name: "user", usuario)
        startActivity(intento)
    } else {
        Toast.makeText(
            context: this,
            text: "El usuario o la contraseña no es correcto",
            Toast.LENGTH_SHORT
        ).show()
    }

    edtUser.text.clear()
    edtPass.text.clear()
    cursor.close()
    bd.close()
}
```

Registro:

Evento boton Registrar:

```
bttnRegistrar.setOnClickListener { it: View!
    try {
        val bd = admin.writableDatabase
        val usuario = edtUser.text.toString()
        val clave = edtPass.text.toString()

        val cursor = bd.query( table: "usuario", arrayOf("user"), selection: "user = ?",
            arrayOf(usuario), groupBy: null, having: null, orderBy: null)

        if (cursor.moveToFirst()) {
            Toast.makeText( context: this, text: "El usuario ya existe.", Toast.LENGTH_SHORT).show()
            cursor.close()
            bd.close()
        } else {
            val registro = ContentValues()
            registro.put("user", usuario)
            registro.put("clave", clave)
            registro.put("bitcoin", 0)
            registro.put("ethereum", 0)
            registro.put("ripple", 0)
            registro.put("litecoin", 0)
            registro.put("saldo", 0)

            val ins = bd.insert( table: "usuario", nullColumnHack: null, registro)
```

```
            if (ins != -1L) {
                Toast.makeText( context: this, text: "Usuario registrado con éxito.", Toast.LENGTH_SHORT)
                    .show()
            } else {
                Toast.makeText( context: this, text: "Error al registrar el usuario.", Toast.LENGTH_SHORT)
                    .show()
            }
            cursor.close()
            bd.close()
            finish()
        }
    } catch (e: Exception) {
        Toast.makeText( context: this, e.toString(), Toast.LENGTH_SHORT).show()
    }
}
```

Inicio:

Funcion verFondos:

```
@SuppressLint("Range")
private fun verFondos(usuario: String?, textFondos: TextView) {
    val admin = SQLAd( context: this,  name: "Cryptos",  factory: null,  version: 1)
    val bd = admin.readableDatabase

    try {
        val cursor = bd.query( table: "usuario", arrayOf("saldo"),
            selection: "user = ?", arrayOf(usuario),  groupBy: null,  having: null,  orderBy: null)
        var saldo = 0
        if (cursor.moveToFirst()) {
            saldo = cursor.getInt(cursor.getColumnIndex("saldo"))
        }
        cursor.close()
        bd.close()
        textFondos.text = saldo.toString()
    } catch (e: Exception) {
        bd.close()
        Toast.makeText( context: this, e.toString(), Toast.LENGTH_SHORT).show()
        Toast.makeText( context: this, e.toString(), Toast.LENGTH_SHORT).show()
        Toast.makeText( context: this, e.toString(), Toast.LENGTH_SHORT).show()
    }
}
```

Funcion onActivityResult:

Comprueba el resultado de la actividad para actualizar o no los fondos.

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (requestCode == 1 && resultCode == RESULT_OK) {
        val user = intent.getStringExtra( name: "user")
        val textFondos = findViewById<TextView>(R.id.txtvwFondos)
        verFondos(user, textFondos)
    }
}
```

Funcion comprar:

Consulta del saldo del usuario y la cantidad de la criptomoneda pasada como parametro.

```
@SuppressLint("Range")
fun comprar(usuario: String?, moneda: String, textFondos: TextView) {
    val admin = SQLAd( context: this,  name: "Cryptos",  factory: null,  version: 1)
    val bd = admin.readableDatabase
    val cursor = bd.query( table: "usuario", arrayOf(moneda, "saldo"),
        selection: "user = ?", arrayOf(usuario),  groupBy: null,  having: null,  orderBy: null)
    var cantidad = 0
    var saldo = 0
    if (cursor.moveToFirst()) {
        cantidad = cursor.getInt(cursor.getColumnIndex(moneda))
        saldo = cursor.getInt(cursor.getColumnIndex("saldo"))
    }
    cursor.close()
    bd.close()
```

Expresion when para determinar el coste de la compra.

```
val costo: Int = when (moneda) {
    "bitcoin" -> 20000
    "ethereum" -> 5000
    "ripple" -> 700
    "litecoin" -> 20
    else -> 0
}
```

Comprobacion de la viabilidad de la compra y actualizacion en base de datos y en interfaz grafica.

```
if (saldo >= costo) {
    val admin2 = SQLAd( context: this,  name: "Cryptos",  factory: null,  version: 1)
    val bd2 = admin2.readableDatabase
    val valores = ContentValues()
    valores.put("saldo", saldo - costo)
    bd2.update( table: "usuario", valores,  whereClause: "user = ?", arrayOf(usuario))
    bd2.close()

    val admin3 = SQLAd( context: this,  name: "Cryptos",  factory: null,  version: 1)
    val bd3 = admin3.readableDatabase
    val valores2 = ContentValues()
    valores2.put(moneda, cantidad + 1)
    bd3.update( table: "usuario", valores2,  whereClause: "user = ?", arrayOf(usuario))
    bd3.close()

    verFondos(usuario, textFondos)
} else {
    Toast.makeText( context: this,  text: "No tienes suficientes fondos.", Toast.LENGTH_SHORT).show()
}
```

Fondos:

Evento añadir:

```
añadir.setOnClickListener { it: View!
    val admin = SQLAd( context: this, name: "Cryptos", factory: null, version: 1)
    val bd = admin.readableDatabase
    val usu = intent.getStringExtra( name: "user")
    val cantidad = texto.text.toString().toInt()
    val valores = ContentValues()
    valores.put("saldo", fondos + cantidad)
    bd.update( table: "usuario", valores, whereClause: "user = ?", arrayOf(usu))
    bd.close()
    setResult(RESULT_OK)
    finish()
}
```

SQLAd:

```
class SQLAd (context: Context, name: String,
             factory: SQLiteDatabase.CursorFactory?,
             version: Int) : SQLiteOpenHelper(context, name, factory, version){

    Victor *
    override fun onCreate(db: SQLiteDatabase) {
        db.execSQL( sql: "create table usuario (user text primary key," +
                " clave text, bitcoin integer, ethereum integer, ripple integer," +
                " litecoin integer, saldo integer)")
    }

    Victor
    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
        db.execSQL( sql: "drop table if exists usuario")
        onCreate(db)
    }
}
```