Universidad Don Bosco

Desarrollo de Software para Moviles

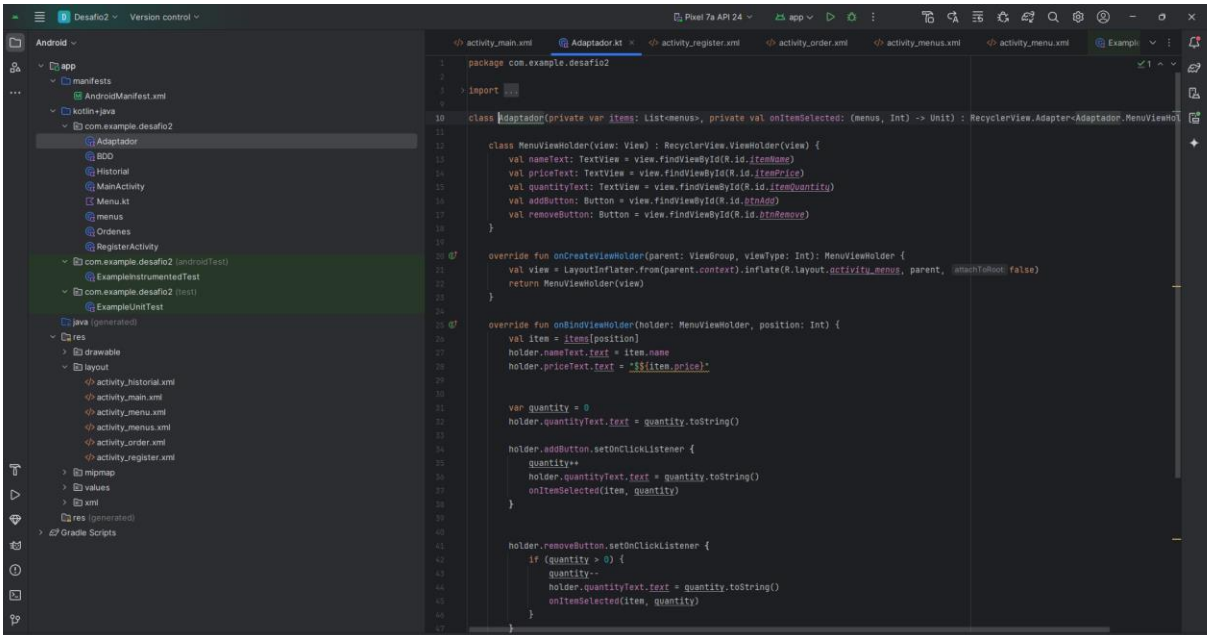
Alumno:

Anderson Alessandro Pablo Beltran PB230838

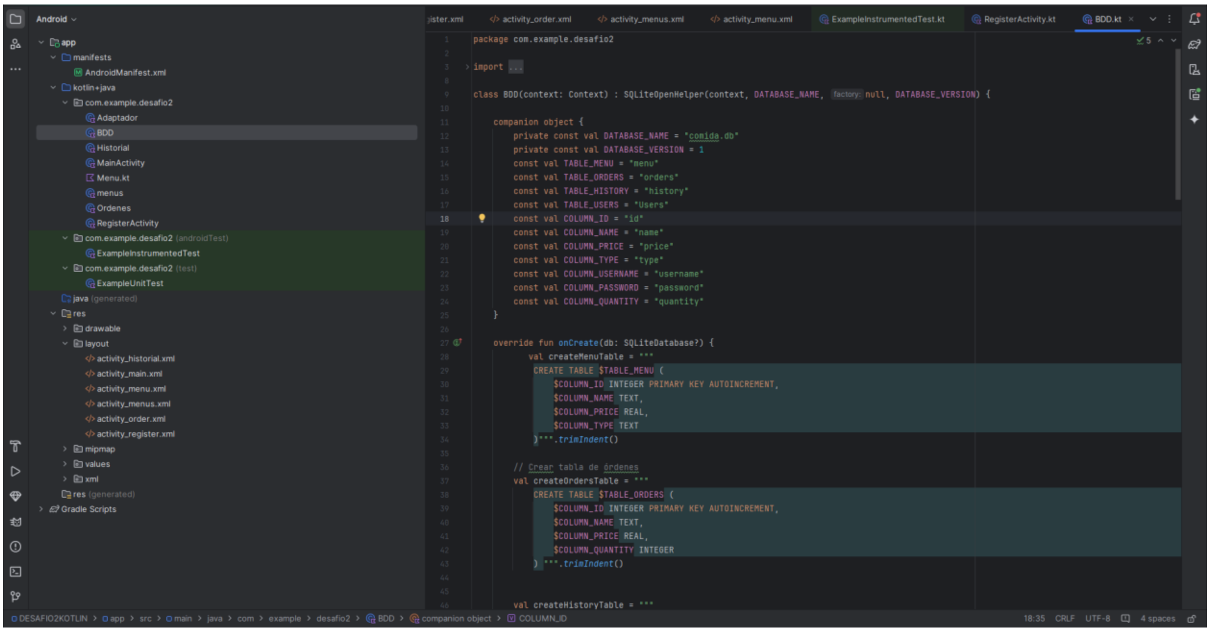
Actividad:

Desafío 2 – Laboratorio

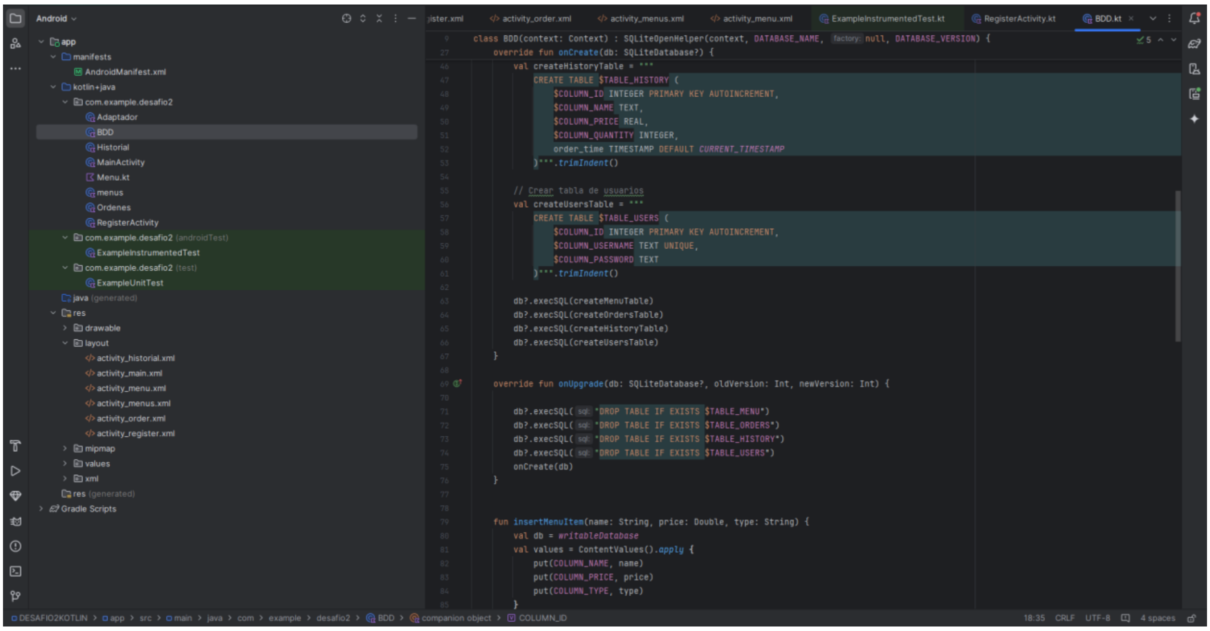
Capturas de Pantalla



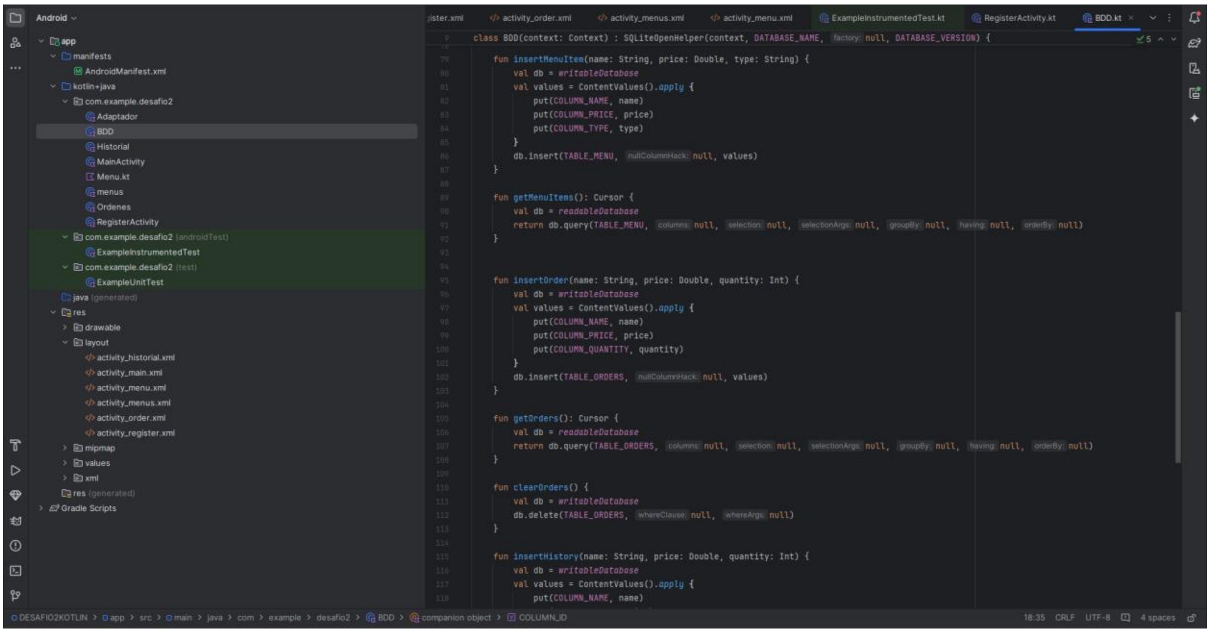
```kotlin
package com.example.desafio2

import ...

class Adaptador(private var items: List<menus>, private val onItemSelected: (menus, Int) -> Unit) : RecyclerView.Adapter<Adaptador.MenuViewHol

    class MenuViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        val nameText: TextView = view.findViewById(R.id.itemName)
        val priceText: TextView = view.findViewById(R.id.itemPrice)
        val quantityText: TextView = view.findViewById(R.id.itemQuantity)
        val addButton: Button = view.findViewById(R.id.btnAdd)
        val removeButton: Button = view.findViewById(R.id.btnRemove)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MenuViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.activity_menus, parent, attachToRoot: false)
        return MenuViewHolder(view)
    }

    override fun onBindViewHolder(holder: MenuViewHolder, position: Int) {
        val item = items[position]
        holder.nameText.text = item.name
        holder.priceText.text = "$${item.price}"


        var quantity = 0
        holder.quantityText.text = quantity.toString()

        holder.addButton.setOnClickListener {
            quantity++
            holder.quantityText.text = quantity.toString()
            onItemSelected(item, quantity)
        }


        holder.removeButton.setOnClickListener {
            if (quantity > 0) {
                quantity--
                holder.quantityText.text = quantity.toString()
                onItemSelected(item, quantity)
            }
        }
```



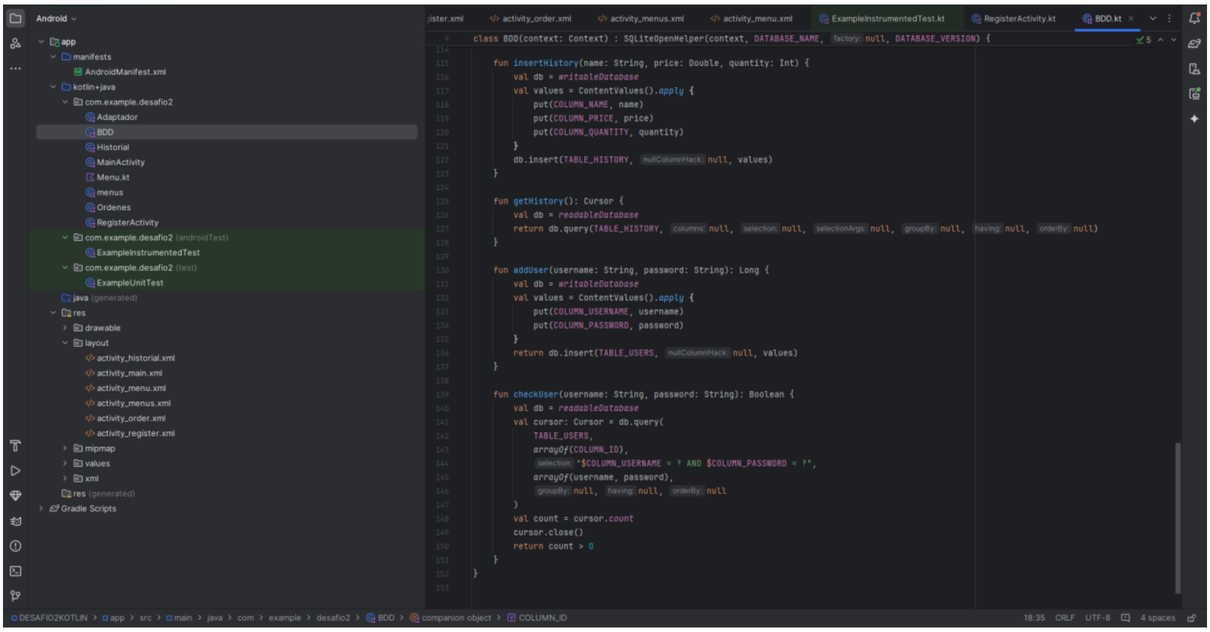
```kotlin
package com.example.desafio2

import ...

class BDD(context: Context) : SQLiteOpenHelper(context, DATABASE_NAME, factory: null, DATABASE_VERSION) {

    companion object {
        private const val DATABASE_NAME = "comida.db"
        private const val DATABASE_VERSION = 1
        const val TABLE_MENU = "menu"
        const val TABLE_ORDERS = "orders"
        const val TABLE_HISTORY = "history"
        const val TABLE_USERS = "Users"
        const val COLUMN_ID = "id"
        const val COLUMN_NAME = "name"
        const val COLUMN_PRICE = "price"
        const val COLUMN_TYPE = "type"
        const val COLUMN_USERNAME = "username"
        const val COLUMN_PASSWORD = "password"
        const val COLUMN_QUANTITY = "quantity"
    }

    override fun onCreate(db: SQLiteDatabase?) {
            val createMenuTable = """
             CREATE TABLE $TABLE_MENU (
                 $COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT,
                 $COLUMN_NAME TEXT,
                 $COLUMN_PRICE REAL,
                 $COLUMN_TYPE TEXT
             )""".trimIndent()

        // Crear tabla de órdenes
        val createOrdersTable = """
             CREATE TABLE $TABLE_ORDERS (
                 $COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT,
                 $COLUMN_NAME TEXT,
                 $COLUMN_PRICE REAL,
                 $COLUMN_QUANTITY INTEGER
             ) """.trimIndent()


        val createHistoryTable = """
```



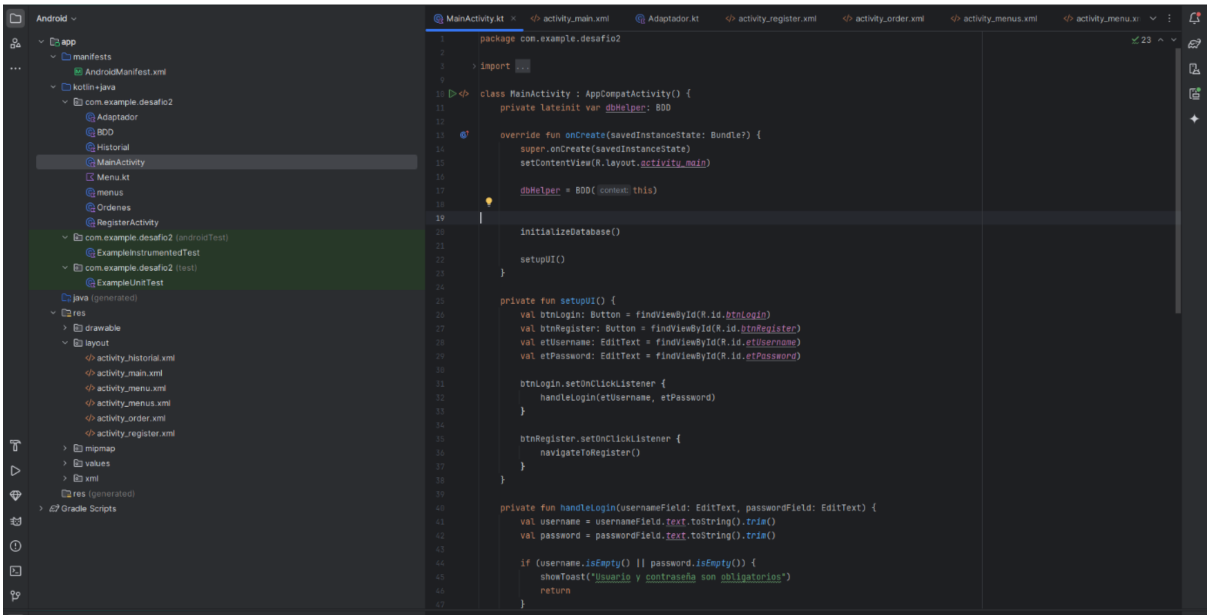
```kotlin
class BDD(context: Context) : SQLiteOpenHelper(context, DATABASE_NAME,  factory: null, DATABASE_VERSION) {
    override fun onCreate(db: SQLiteDatabase?) {
        val createHistoryTable = """
            CREATE TABLE $TABLE_HISTORY (
                $COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT,
                $COLUMN_NAME TEXT,
                $COLUMN_PRICE REAL,
                $COLUMN_QUANTITY INTEGER,
                order_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP
            )""".trimIndent()

        // Crear tabla de usuarios
        val createUsersTable = """
            CREATE TABLE $TABLE_USERS (
                $COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT,
                $COLUMN_USERNAME TEXT UNIQUE,
                $COLUMN_PASSWORD TEXT
            )""".trimIndent()

        db?.execSQL(createMenuTable)
        db?.execSQL(createOrdersTable)
        db?.execSQL(createHistoryTable)
        db?.execSQL(createUsersTable)
    }

    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {

        db?.execSQL( sql: "DROP TABLE IF EXISTS $TABLE_MENU")
        db?.execSQL( sql: "DROP TABLE IF EXISTS $TABLE_ORDERS")
        db?.execSQL( sql: "DROP TABLE IF EXISTS $TABLE_HISTORY")
        db?.execSQL( sql: "DROP TABLE IF EXISTS $TABLE_USERS")
        onCreate(db)
    }


    fun insertMenuItem(name: String, price: Double, type: String) {
        val db = writableDatabase
        val values = ContentValues().apply {
            put(COLUMN_NAME, name)
            put(COLUMN_PRICE, price)
            put(COLUMN_TYPE, type)
        }
```



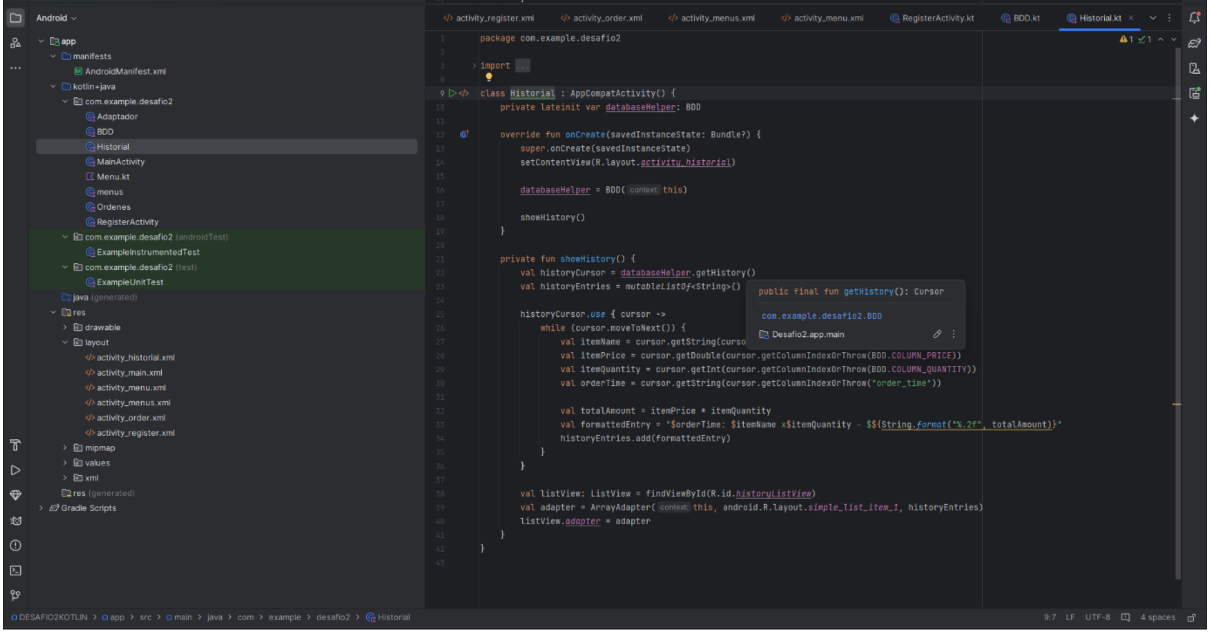
```kotlin
class BDD(context: Context) : SQLiteOpenHelper(context, DATABASE_NAME,  factory: null, DATABASE_VERSION) {
    fun insertMenuItem(name: String, price: Double, type: String) {
        val db = writableDatabase
        val values = ContentValues().apply {
            put(COLUMN_NAME, name)
            put(COLUMN_PRICE, price)
            put(COLUMN_TYPE, type)
        }
        db.insert(TABLE_MENU,  nullColumnHack: null, values)
    }

    fun getMenuItems(): Cursor {
        val db = readableDatabase
        return db.query(TABLE_MENU,  columns: null,  selection: null,  selectionArgs: null,  groupBy: null,  having: null,  orderBy: null)
    }


    fun insertOrder(name: String, price: Double, quantity: Int) {
        val db = writableDatabase
        val values = ContentValues().apply {
            put(COLUMN_NAME, name)
            put(COLUMN_PRICE, price)
            put(COLUMN_QUANTITY, quantity)
        }
        db.insert(TABLE_ORDERS,  nullColumnHack: null, values)
    }

    fun getOrders(): Cursor {
        val db = readableDatabase
        return db.query(TABLE_ORDERS,  columns: null,  selection: null,  selectionArgs: null,  groupBy: null,  having: null,  orderBy: null)
    }

    fun clearOrders() {
        val db = writableDatabase
        db.delete(TABLE_ORDERS,  whereClause: null,  whereArgs: null)
    }

    fun insertHistory(name: String, price: Double, quantity: Int) {
        val db = writableDatabase
        val values = ContentValues().apply {
            put(COLUMN_NAME, name)
```



```
class BDD(context: Context) : SQLiteOpenHelper(context, DATABASE_NAME, factory: null, DATABASE_VERSION) {

    fun insertHistory(name: String, price: Double, quantity: Int) {
        val db = writableDatabase
        val values = ContentValues().apply {
            put(COLUMN_NAME, name)
            put(COLUMN_PRICE, price)
            put(COLUMN_QUANTITY, quantity)
        }
        db.insert(TABLE_HISTORY, nullColumnHack: null, values)
    }

    fun getHistory(): Cursor {
        val db = readableDatabase
        return db.query(TABLE_HISTORY, columns: null, selection: null, selectionArgs: null, groupBy: null, having: null, orderBy: null)
    }

    fun addUser(username: String, password: String): Long {
        val db = writableDatabase
        val values = ContentValues().apply {
            put(COLUMN_USERNAME, username)
            put(COLUMN_PASSWORD, password)
        }
        return db.insert(TABLE_USERS, nullColumnHack: null, values)
    }

    fun checkUser(username: String, password: String): Boolean {
        val db = readableDatabase
        val cursor: Cursor = db.query(
            TABLE_USERS,
            arrayOf(COLUMN_ID),
            selection: "$COLUMN_USERNAME = ? AND $COLUMN_PASSWORD = ?",
            arrayOf(username, password),
            groupBy: null, having: null, orderBy: null
        )
        val count = cursor.count
        cursor.close()
        return count > 0
    }
}
```

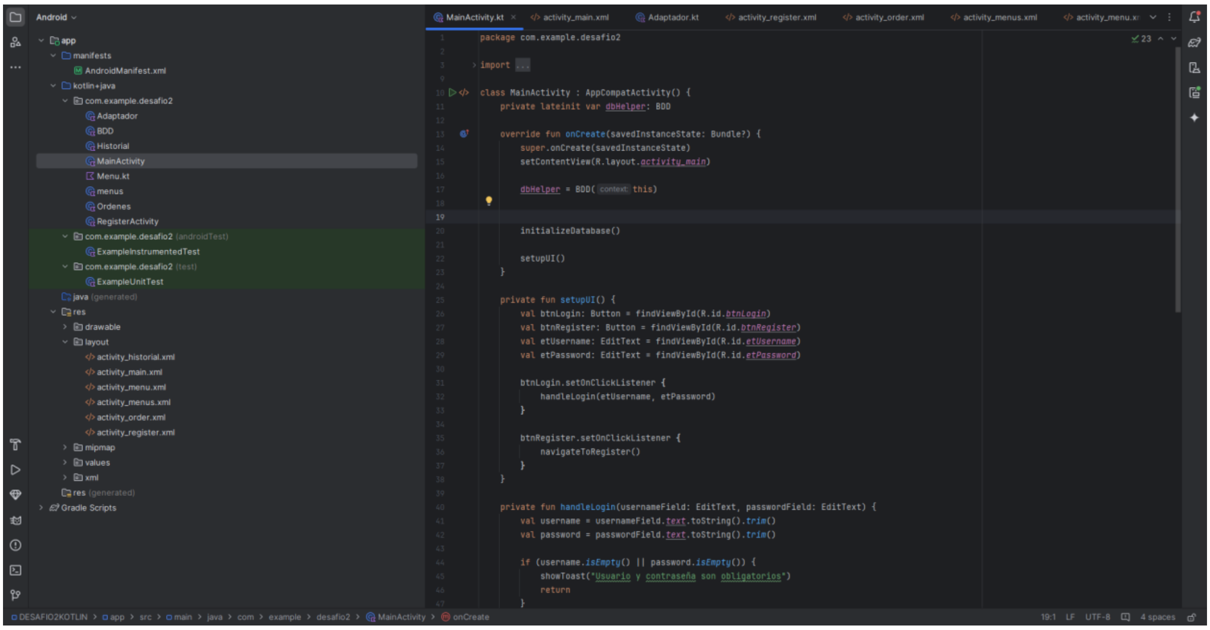


```
package com.example.desafio2

import ...

class MainActivity : AppCompatActivity() {
    private lateinit var dbHelper: BDD

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        dbHelper = BDD( context: this)


        initializeDatabase()

        setupUI()
    }

    private fun setupUI() {
        val btnLogin: Button = findViewById(R.id.btnLogin)
        val btnRegister: Button = findViewById(R.id.btnRegister)
        val etUsername: EditText = findViewById(R.id.etUsername)
        val etPassword: EditText = findViewById(R.id.etPassword)

        btnLogin.setOnClickListener {
            handleLogin(etUsername, etPassword)
        }

        btnRegister.setOnClickListener {
            navigateToRegister()
        }
    }

    private fun handleLogin(usernameField: EditText, passwordField: EditText) {
        val username = usernameField.text.toString().trim()
        val password = passwordField.text.toString().trim()

        if (username.isEmpty() || password.isEmpty()) {
            showToast("Usuario y contraseña son obligatorios")
            return
        }
```

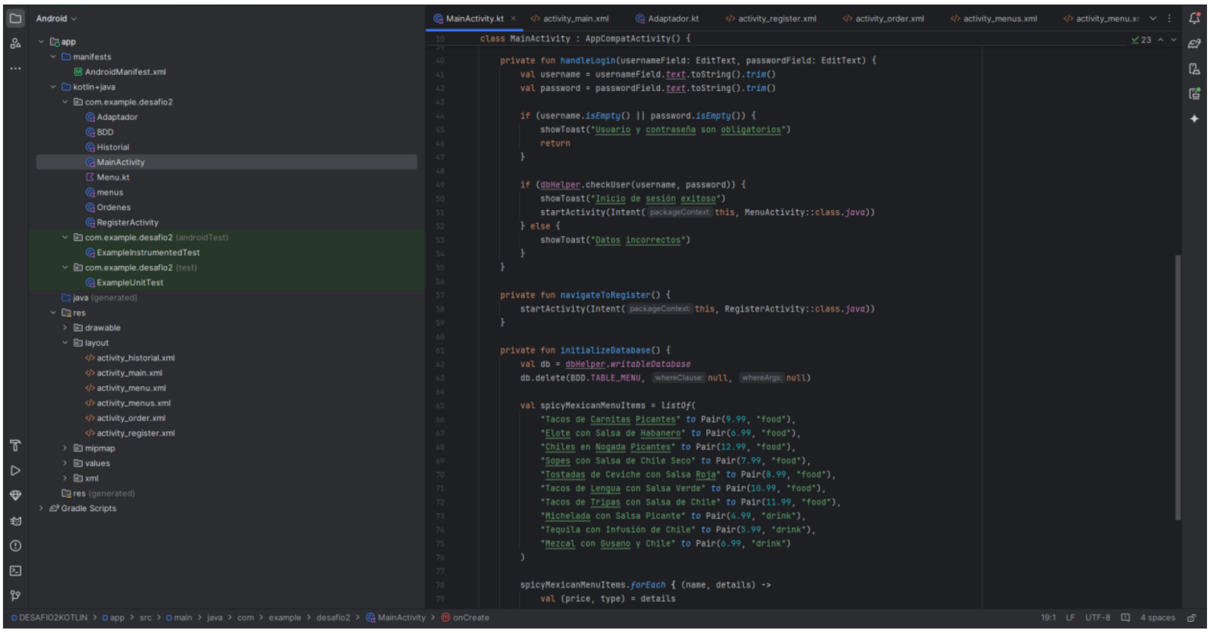


```
package com.example.desafio2

import ...

class Historial : AppCompatActivity() {
    private lateinit var databaseHelper: BDD

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_historial)

        databaseHelper = BDD( context: this)

        showHistory()
    }

    private fun showHistory() {
        val historyCursor = databaseHelper.getHistory()
        val historyEntries = mutableListOf<String>()

        historyCursor.use { cursor ->
            while (cursor.moveToNext()) {
                val itemName = cursor.getString(curso
                val itemPrice = cursor.getDouble(cursor.getColumnIndexOrThrow(BDD.COLUMN_PRICE))
                val itemQuantity = cursor.getInt(cursor.getColumnIndexOrThrow(BDD.COLUMN_QUANTITY))
                val orderTime = cursor.getString(cursor.getColumnIndexOrThrow("order_time"))

                val totalAmount = itemPrice * itemQuantity
                val formattedEntry = "$orderTime: $itemName x$itemQuantity - $${String.format("%.2f", totalAmount)}"
                historyEntries.add(formattedEntry)
            }
        }

        val listView: ListView = findViewById(R.id.historyListView)
        val adapter = ArrayAdapter( context: this, android.R.layout.simple_list_item_1, historyEntries)
        listView.adapter = adapter
    }
}
```

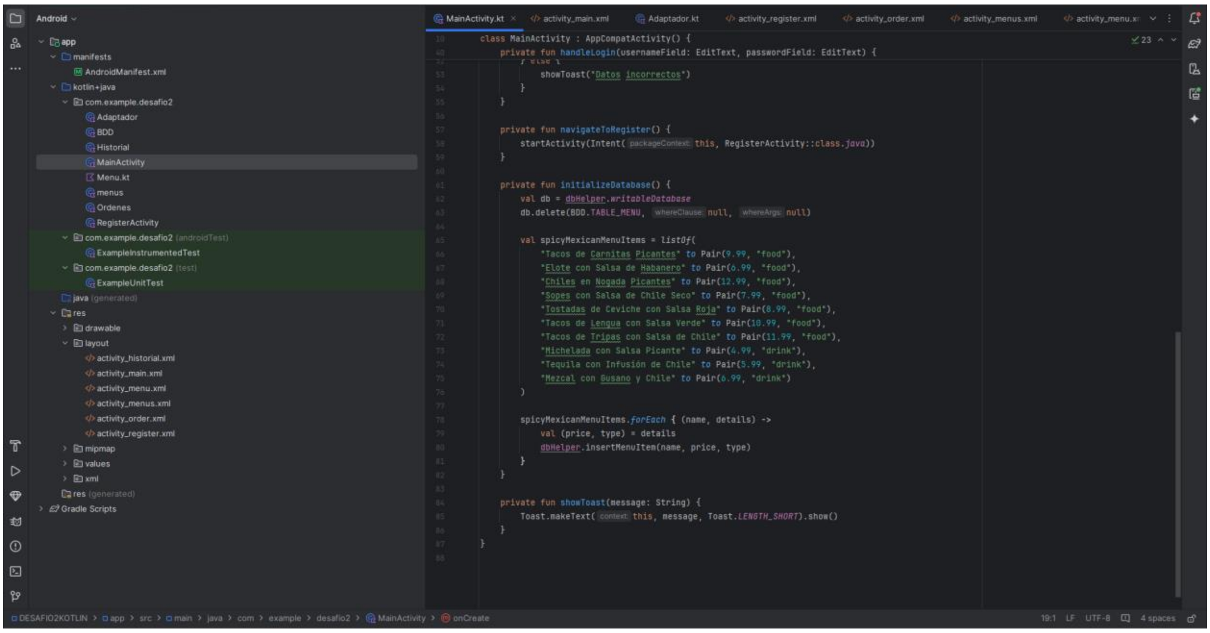
```
public final fun getHistory(): Cursor
com.example.desafio2.BDD
Desafio2.app.main
```



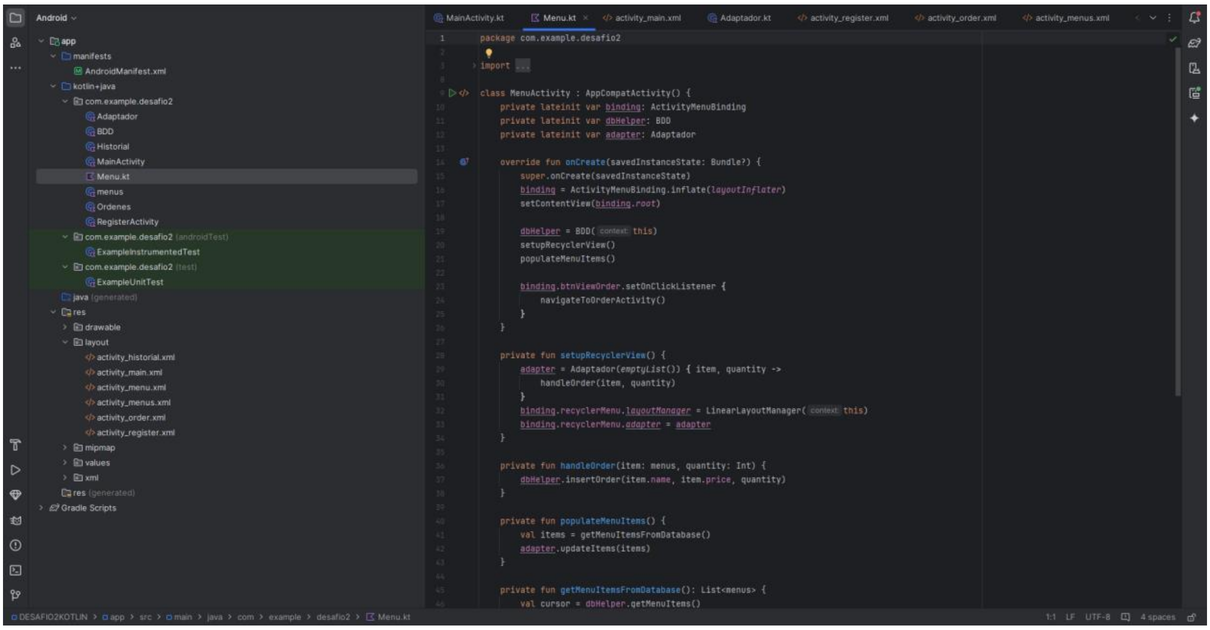
```kotlin
package com.example.desafio2

import ...

class MainActivity : AppCompatActivity() {
    private lateinit var dbHelper: BDD

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        dbHelper = BDD( context: this)


        initializeDatabase()

        setupUI()
    }

    private fun setupUI() {
        val btnLogin: Button = findViewById(R.id.btnLogin)
        val btnRegister: Button = findViewById(R.id.btnRegister)
        val etUsername: EditText = findViewById(R.id.etUsername)
        val etPassword: EditText = findViewById(R.id.etPassword)

        btnLogin.setOnClickListener {
            handleLogin(etUsername, etPassword)
        }

        btnRegister.setOnClickListener {
            navigateToRegister()
        }
    }

    private fun handleLogin(usernameField: EditText, passwordField: EditText) {
        val username = usernameField.text.toString().trim()
        val password = passwordField.text.toString().trim()

        if (username.isEmpty() || password.isEmpty()) {
            showToast("Usuario y contraseña son obligatorios")
            return
        }
```



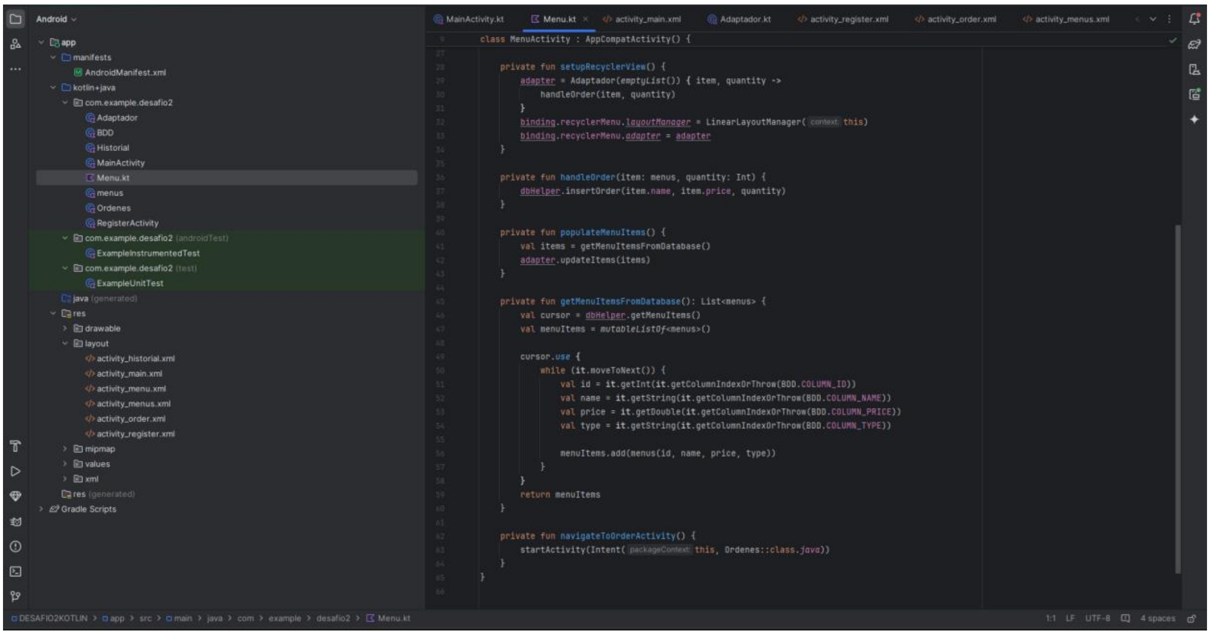
```kotlin
class MainActivity : AppCompatActivity() {
    private fun handleLogin(usernameField: EditText, passwordField: EditText) {
        val username = usernameField.text.toString().trim()
        val password = passwordField.text.toString().trim()

        if (username.isEmpty() || password.isEmpty()) {
            showToast("Usuario y contraseña son obligatorios")
            return
        }

        if (dbHelper.checkUser(username, password)) {
            showToast("Inicio de sesión exitoso")
            startActivity(Intent( packageContext: this, MenuActivity::class.java))
        } else {
            showToast("Datos incorrectos")
        }
    }

    private fun navigateToRegister() {
        startActivity(Intent( packageContext: this, RegisterActivity::class.java))
    }

    private fun initializeDatabase() {
        val db = dbHelper.writableDatabase
        db.delete(BDD.TABLE_MENU,  whereClause: null,  whereArgs: null)

        val spicyMexicanMenuItems = listOf(
            "Tacos de Carnitas Picantes" to Pair(9.99, "food"),
            "Elote con Salsa de Habanero" to Pair(6.99, "food"),
            "Chiles en Nogada Picantes" to Pair(12.99, "food"),
            "Sopes con Salsa de Chile Seco" to Pair(7.99, "food"),
            "Tostadas de Ceviche con Salsa Roja" to Pair(8.99, "food"),
            "Tacos de Lengua con Salsa Verde" to Pair(10.99, "food"),
            "Tacos de Tripas con Salsa de Chile" to Pair(11.99, "food"),
            "Michelada con Salsa Picante" to Pair(4.99, "drink"),
            "Tequila con Infusión de Chile" to Pair(5.99, "drink"),
            "Mezcal con Gusano y Chile" to Pair(6.99, "drink")
        )

        spicyMexicanMenuItems.forEach { (name, details) ->
            val (price, type) = details
```



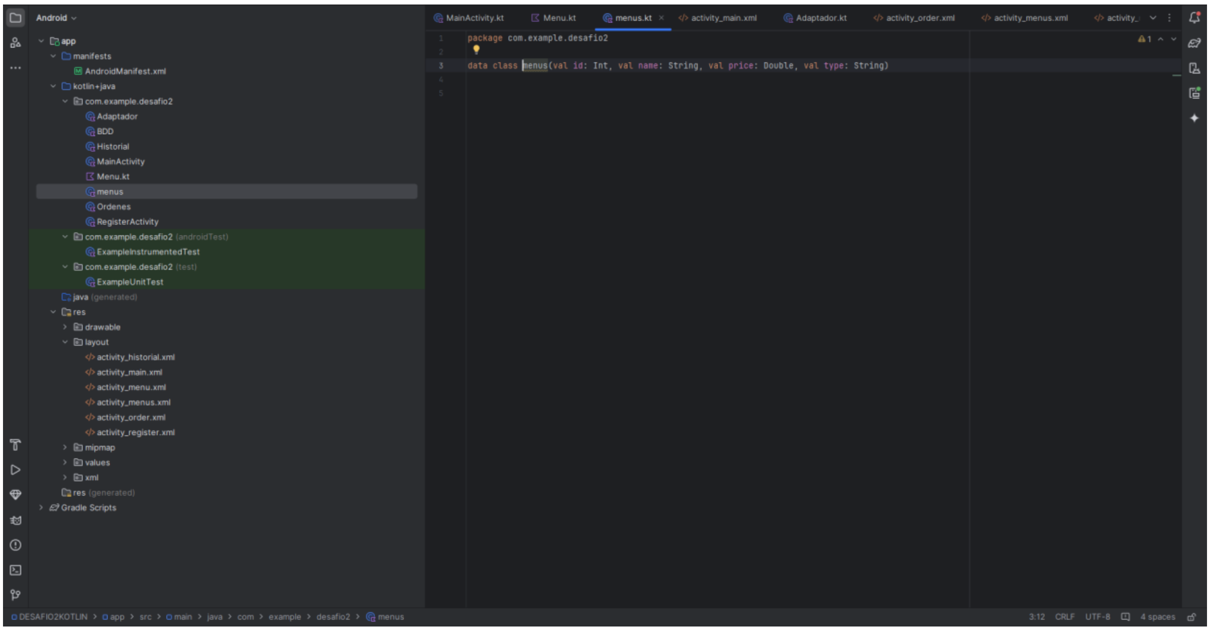
```
class MainActivity : AppCompatActivity() {
    private fun handleLogin(usernameField: EditText, passwordField: EditText) {
        } else {
            showToast("Datos incorrectos")
        }
    }

    private fun navigateToRegister() {
        startActivity(Intent( packageContext: this, RegisterActivity::class.java))
    }

    private fun initializeDatabase() {
        val db = dbHelper.writableDatabase
        db.delete(BDD.TABLE_MENU,  whereClause: null,  whereArgs: null)

        val spicyMexicanMenuItems = listOf(
            "Tacos de Carnitas Picantes" to Pair(9.99, "food"),
            "Elote con Salsa de Habanero" to Pair(6.99, "food"),
            "Chiles en Nogada Picantes" to Pair(12.99, "food"),
            "Sopes con Salsa de Chile Seco" to Pair(7.99, "food"),
            "Tostadas de Ceviche con Salsa Roja" to Pair(8.99, "food"),
            "Tacos de Lengua con Salsa Verde" to Pair(10.99, "food"),
            "Tacos de Tripas con Salsa de Chile" to Pair(11.99, "food"),
            "Michelada con Salsa Picante" to Pair(4.99, "drink"),
            "Tequila con Infusión de Chile" to Pair(5.99, "drink"),
            "Mezcal con Gusano y Chile" to Pair(6.99, "drink")
        )

        spicyMexicanMenuItems.forEach { (name, details) ->
            val (price, type) = details
            dbHelper.insertMenuItem(name, price, type)
        }
    }

    private fun showToast(message: String) {
        Toast.makeText( context: this, message, Toast.LENGTH_SHORT).show()
    }
}
```



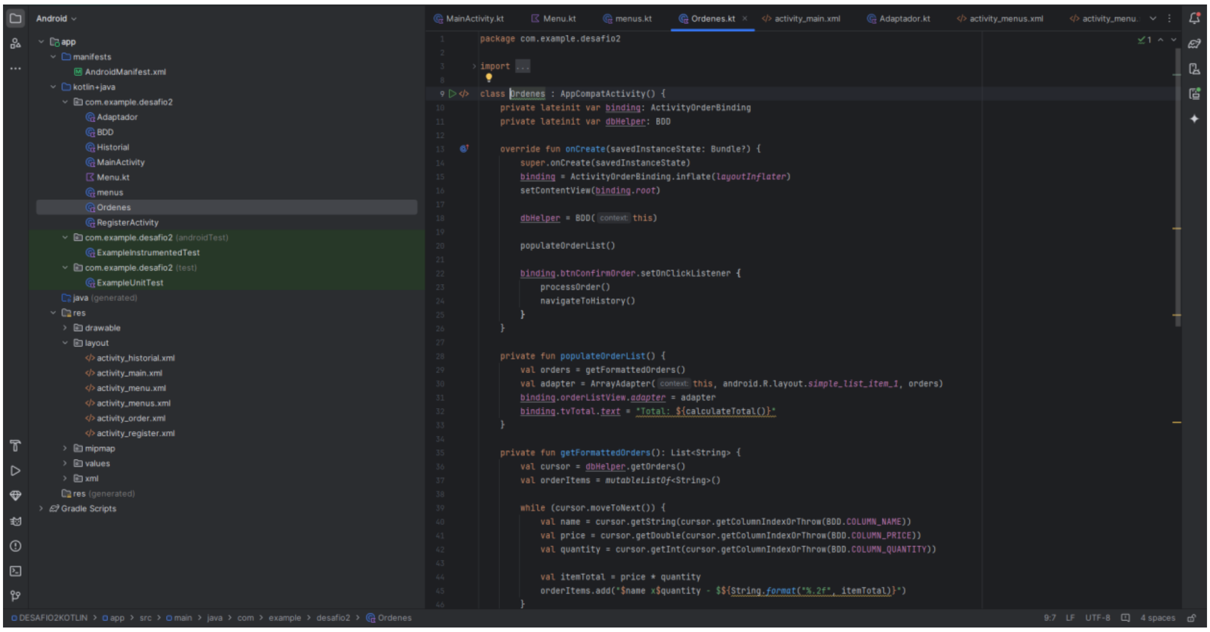
```
package com.example.desafio2

import ...

class MenuActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMenuBinding
    private lateinit var dbHelper: BDD
    private lateinit var adapter: Adaptador

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMenuBinding.inflate(layoutInflater)
        setContentView(binding.root)

        dbHelper = BDD( context: this)
        setupRecyclerView()
        populateMenuItems()

        binding.btnViewOrder.setOnClickListener {
            navigateToOrderActivity()
        }
    }

    private fun setupRecyclerView() {
        adapter = Adaptador(emptyList()) { item, quantity ->
            handleOrder(item, quantity)
        }
        binding.recyclerMenu.layoutManager = LinearLayoutManager( context: this)
        binding.recyclerMenu.adapter = adapter
    }

    private fun handleOrder(item: menus, quantity: Int) {
        dbHelper.insertOrder(item.name, item.price, quantity)
    }

    private fun populateMenuItems() {
        val items = getMenuItemsFromDatabase()
        adapter.updateItems(items)
    }

    private fun getMenuItemsFromDatabase(): List<menus> {
        val cursor = dbHelper.getMenuItems()
```



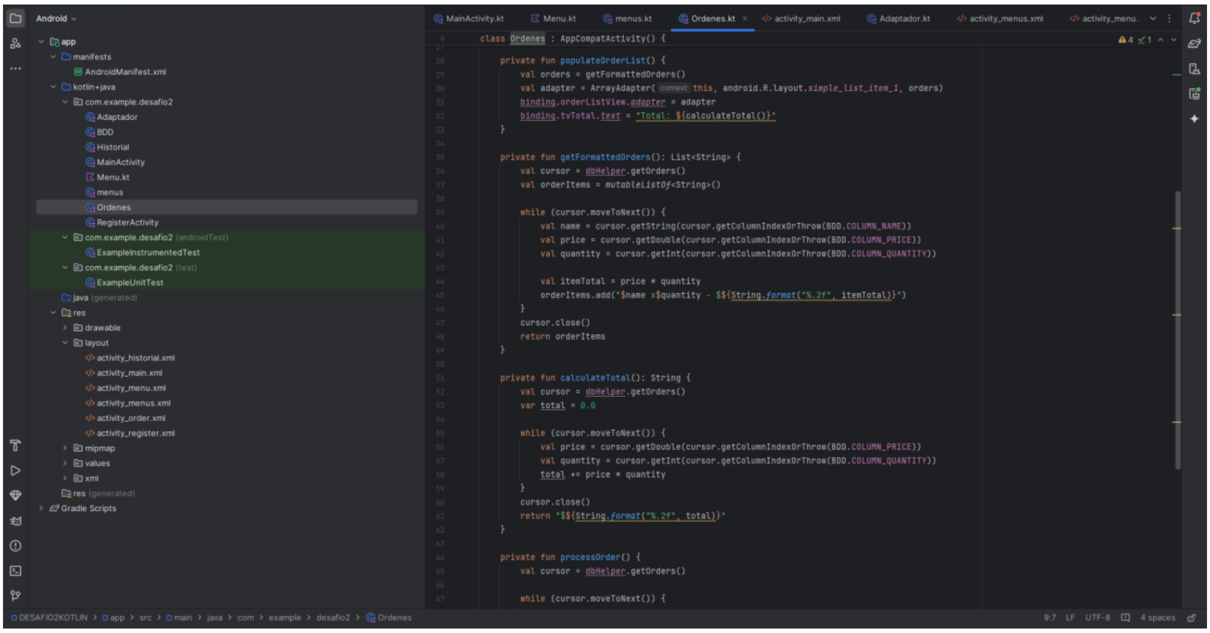
```kotlin
class MenuActivity : AppCompatActivity() {

    private fun setupRecyclerView() {
        adapter = Adaptador(emptyList()) { item, quantity ->
            handleOrder(item, quantity)
        }
        binding.recyclerMenu.layoutManager = LinearLayoutManager(this)
        binding.recyclerMenu.adapter = adapter
    }

    private fun handleOrder(item: menus, quantity: Int) {
        dbHelper.insertOrder(item.name, item.price, quantity)
    }

    private fun populateMenuItems() {
        val items = getMenuItemsFromDatabase()
        adapter.updateItems(items)
    }

    private fun getMenuItemsFromDatabase(): List<menus> {
        val cursor = dbHelper.getMenuItems()
        val menuItems = mutableListOf<menus>()

        cursor.use {
            while (it.moveToNext()) {
                val id = it.getInt(it.getColumnIndexOrThrow(BDD.COLUMN_ID))
                val name = it.getString(it.getColumnIndexOrThrow(BDD.COLUMN_NAME))
                val price = it.getDouble(it.getColumnIndexOrThrow(BDD.COLUMN_PRICE))
                val type = it.getString(it.getColumnIndexOrThrow(BDD.COLUMN_TYPE))

                menuItems.add(menus(id, name, price, type))
            }
        }
        return menuItems
    }

    private fun navigateToOrderActivity() {
        startActivity(Intent(this, Ordenes::class.java))
    }
}
```



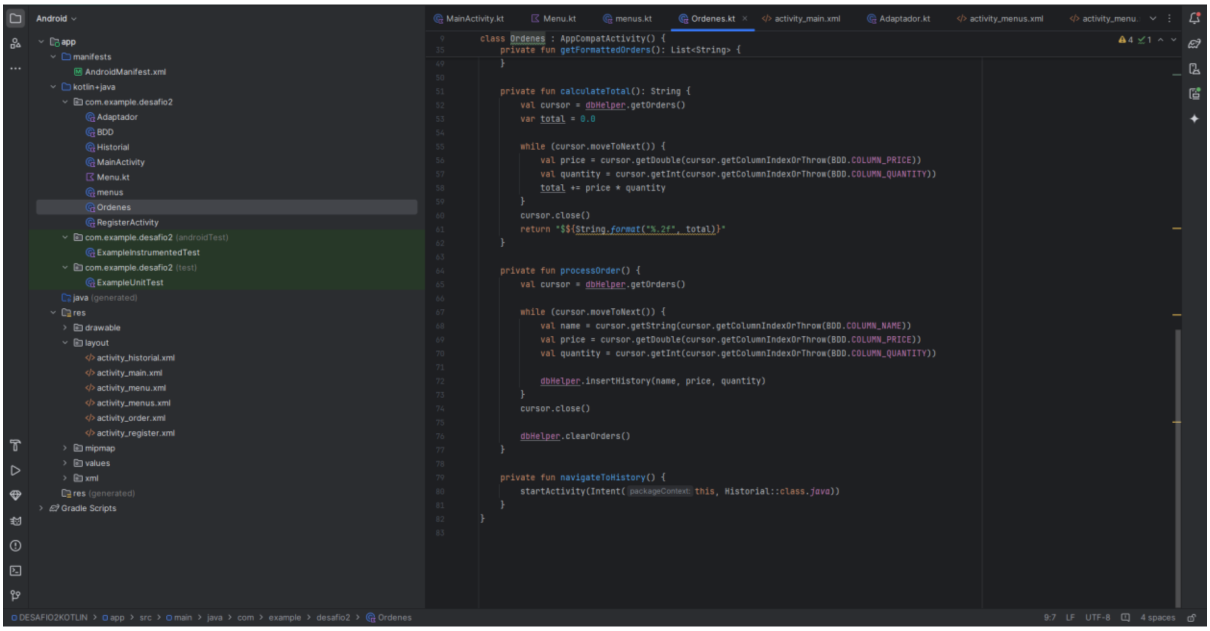
```kotlin
package com.example.desafio2

data class menus(val id: Int, val name: String, val price: Double, val type: String)
```



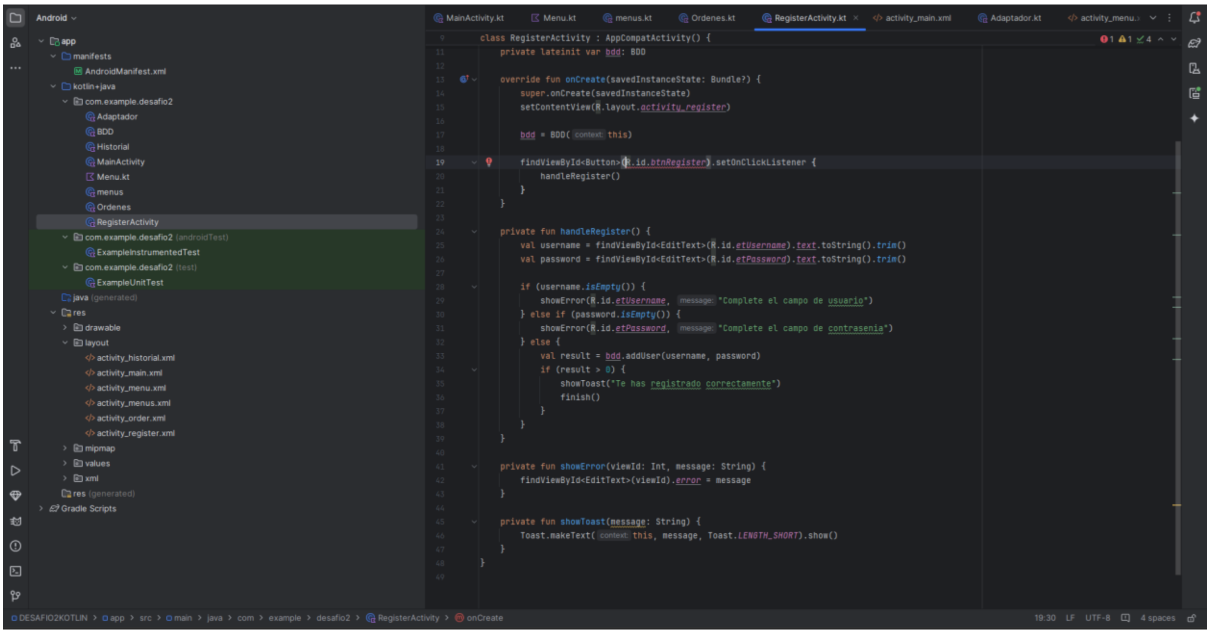
```kotlin
package com.example.desafio2

import ...

class Ordenes : AppCompatActivity() {
    private lateinit var binding: ActivityOrderBinding
    private lateinit var dbHelper: BDD

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityOrderBinding.inflate(layoutInflater)
        setContentView(binding.root)

        dbHelper = BDD( context: this)

        populateOrderList()

        binding.btnConfirmOrder.setOnClickListener {
            processOrder()
            navigateToHistory()
        }
    }

    private fun populateOrderList() {
        val orders = getFormattedOrders()
        val adapter = ArrayAdapter( context: this, android.R.layout.simple_list_item_1, orders)
        binding.orderListView.adapter = adapter
        binding.tvTotal.text = "Total: ${calculateTotal()}"
    }

    private fun getFormattedOrders(): List<String> {
        val cursor = dbHelper.getOrders()
        val orderItems = mutableListOf<String>()

        while (cursor.moveToNext()) {
            val name = cursor.getString(cursor.getColumnIndexOrThrow(BDD.COLUMN_NAME))
            val price = cursor.getDouble(cursor.getColumnIndexOrThrow(BDD.COLUMN_PRICE))
            val quantity = cursor.getInt(cursor.getColumnIndexOrThrow(BDD.COLUMN_QUANTITY))

            val itemTotal = price * quantity
            orderItems.add("$name x$quantity - $${String.format("%.2f", itemTotal)}")
        }
```



```kotlin
class Ordenes : AppCompatActivity() {
    private fun populateOrderList() {
        val orders = getFormattedOrders()
        val adapter = ArrayAdapter( context: this, android.R.layout.simple_list_item_1, orders)
        binding.orderListView.adapter = adapter
        binding.tvTotal.text = "Total: ${calculateTotal()}"
    }

    private fun getFormattedOrders(): List<String> {
        val cursor = dbHelper.getOrders()
        val orderItems = mutableListOf<String>()

        while (cursor.moveToNext()) {
            val name = cursor.getString(cursor.getColumnIndexOrThrow(BDD.COLUMN_NAME))
            val price = cursor.getDouble(cursor.getColumnIndexOrThrow(BDD.COLUMN_PRICE))
            val quantity = cursor.getInt(cursor.getColumnIndexOrThrow(BDD.COLUMN_QUANTITY))

            val itemTotal = price * quantity
            orderItems.add("$name x$quantity - $${String.format("%.2f", itemTotal)}")
        }
        cursor.close()
        return orderItems
    }

    private fun calculateTotal(): String {
        val cursor = dbHelper.getOrders()
        var total = 0.0

        while (cursor.moveToNext()) {
            val price = cursor.getDouble(cursor.getColumnIndexOrThrow(BDD.COLUMN_PRICE))
            val quantity = cursor.getInt(cursor.getColumnIndexOrThrow(BDD.COLUMN_QUANTITY))
            total += price * quantity
        }
        cursor.close()
        return "$${String.format("%.2f", total)}"
    }

    private fun processOrder() {
        val cursor = dbHelper.getOrders()

        while (cursor.moveToNext()) {
```

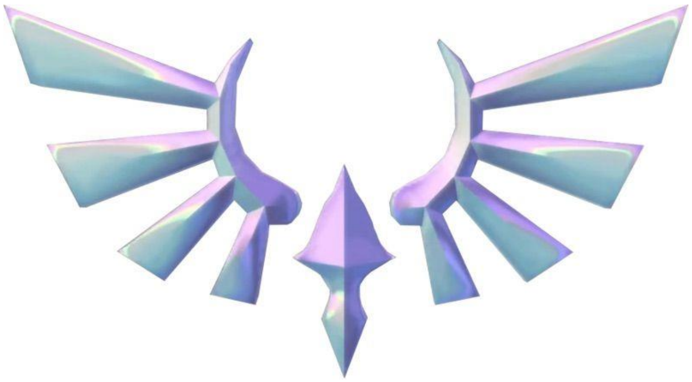


```kotlin
class Ordenes : AppCompatActivity() {
    private fun getFormattedOrders(): List<String> {
    }

    private fun calculateTotal(): String {
        val cursor = dbHelper.getOrders()
        var total = 0.0

        while (cursor.moveToNext()) {
            val price = cursor.getDouble(cursor.getColumnIndexOrThrow(BDD.COLUMN_PRICE))
            val quantity = cursor.getInt(cursor.getColumnIndexOrThrow(BDD.COLUMN_QUANTITY))
            total += price * quantity
        }
        cursor.close()
        return "$${String.format("%.2f", total)}"
    }

    private fun processOrder() {
        val cursor = dbHelper.getOrders()

        while (cursor.moveToNext()) {
            val name = cursor.getString(cursor.getColumnIndexOrThrow(BDD.COLUMN_NAME))
            val price = cursor.getDouble(cursor.getColumnIndexOrThrow(BDD.COLUMN_PRICE))
            val quantity = cursor.getInt(cursor.getColumnIndexOrThrow(BDD.COLUMN_QUANTITY))

            dbHelper.insertHistory(name, price, quantity)
        }
        cursor.close()

        dbHelper.clearOrders()
    }

    private fun navigateToHistory() {
        startActivity(Intent( packageContext: this, Historial::class.java))
    }
}
```



```kotlin
class RegisterActivity : AppCompatActivity() {
    private lateinit var bdd: BDD

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_register)

        bdd = BDD( context: this)

        findViewById<Button>(R.id.btnRegister).setOnClickListener {
            handleRegister()
        }
    }

    private fun handleRegister() {
        val username = findViewById<EditText>(R.id.etUsername).text.toString().trim()
        val password = findViewById<EditText>(R.id.etPassword).text.toString().trim()

        if (username.isEmpty()) {
            showError(R.id.etUsername,  message: "Complete el campo de usuario")
        } else if (password.isEmpty()) {
            showError(R.id.etPassword,  message: "Complete el campo de contrasenia")
        } else {
            val result = bdd.addUser(username, password)
            if (result > 0) {
                showToast("Te has registrado correctamente")
                finish()
            }
        }
    }

    private fun showError(viewId: Int, message: String) {
        findViewById<EditText>(viewId).error = message
    }

    private fun showToast(message: String) {
        Toast.makeText( context: this, message, Toast.LENGTH_SHORT).show()
    }
}
```

Icono y nombre



TaconTodo