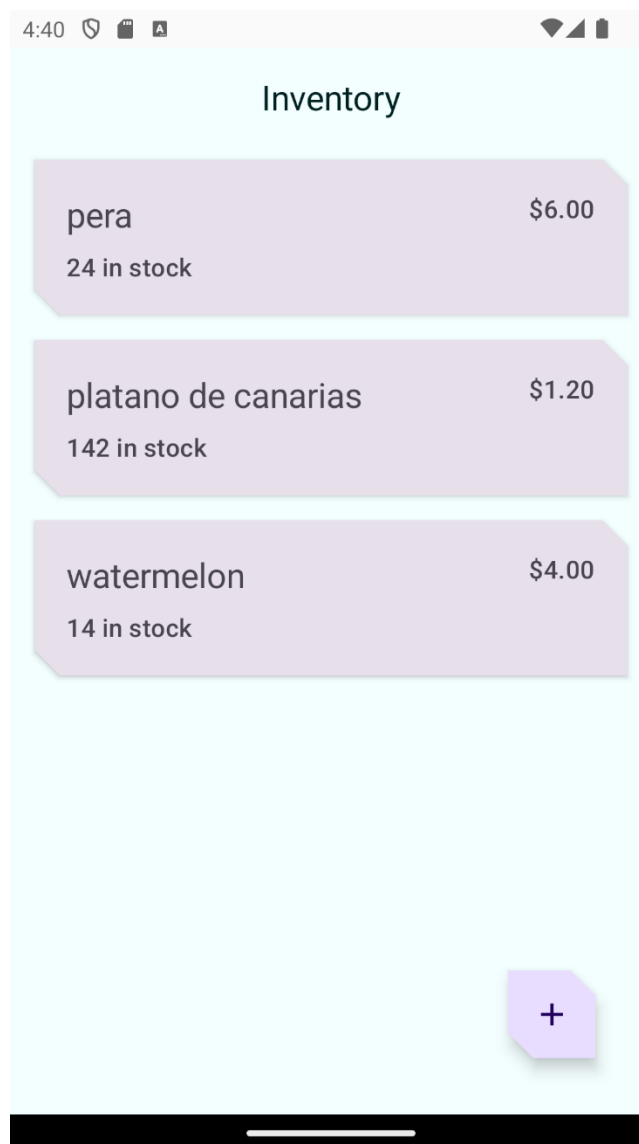


CODELAB 6: PERSISTENCIA DE DATOS

EJERCICIO ROOM

Esta aplicación se basa en un inventario en el cual puedes añadir los productos que quieras y quedaran registrados en la base de datos, además podremos vender los productos de forma individual o eliminarlos directamente si nos hemos quedado sin stock.

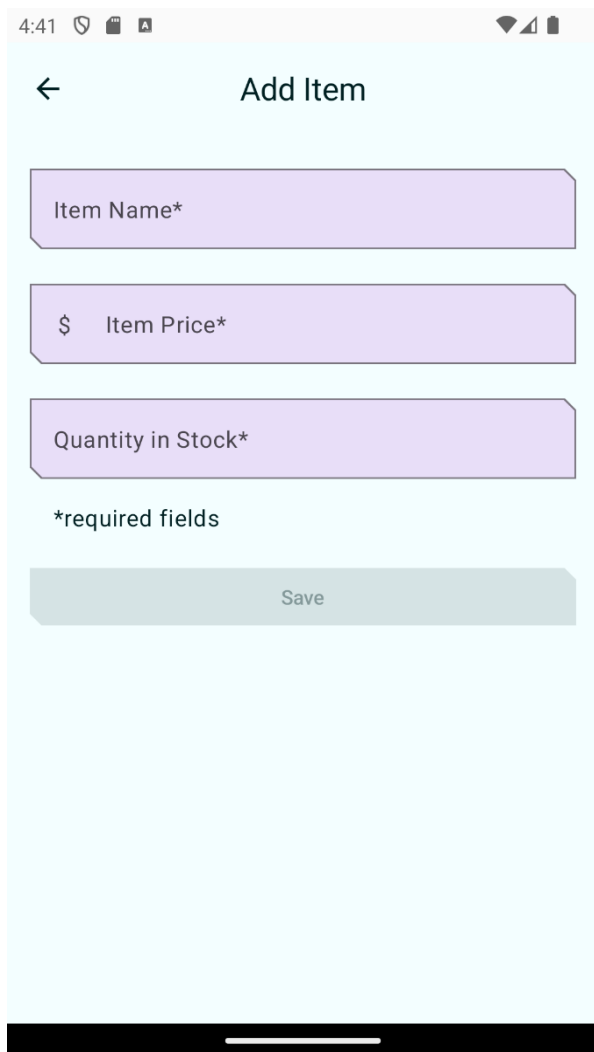
PANTALLA PRINCIPAL



Database Inspector					
Databases					
item_database					
items					
room_master_table					
items					
Live updates					
id					
name					
price					
quantity					
1	1	watermelon	4.0	14	
2	2	pera	6.0	24	
3	3	platano de canarias	1.2	142	

PANTALLA PARA AÑADIR ITEM

Tras pulsar el + de la pantalla principal se nos abrirá la pantalla de añadir ítems que nos permitirá añadir los nuevos productos que tengamos pudiendo introducir el nombre, precio y cantidad y que se quede registrado en nuestra base de datos.



4:41

← Add Item

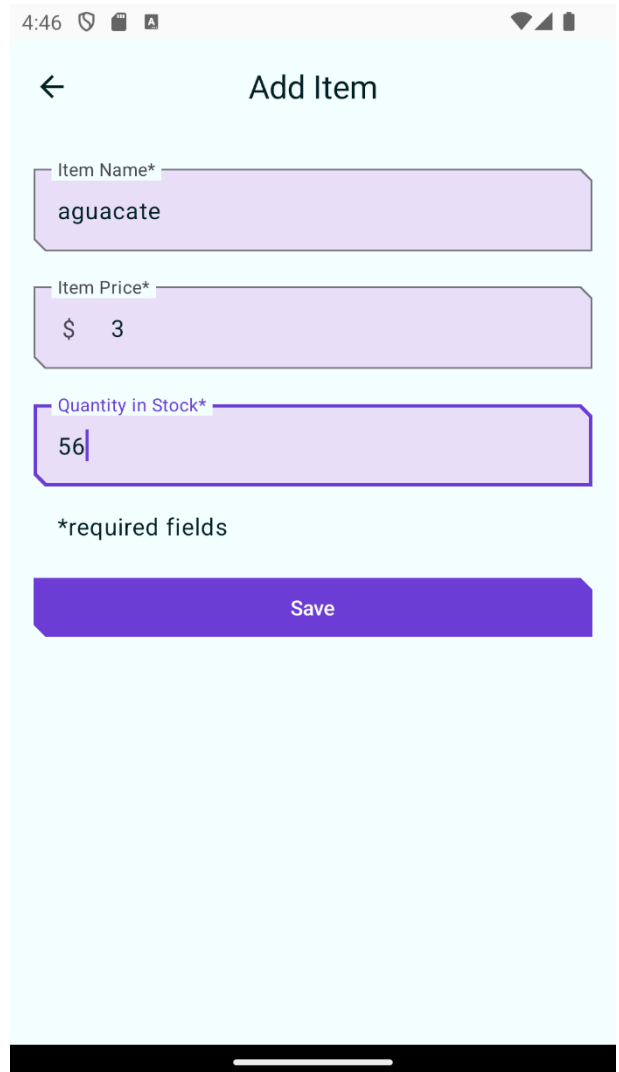
Item Name*

\$ Item Price*

Quantity in Stock*

*required fields

Save



4:46

← Add Item

Item Name*
aguacate

Item Price*
\$ 3

Quantity in Stock*
56

*required fields

Save

Como podemos observar se ha añadido correctamente aguacate a nuestro inventario y se ha actualizado nuestra base de datos.



Pixel 2 API 34.2 > com.example.inventory

Database Inspector Network Inspector Background Task Inspector

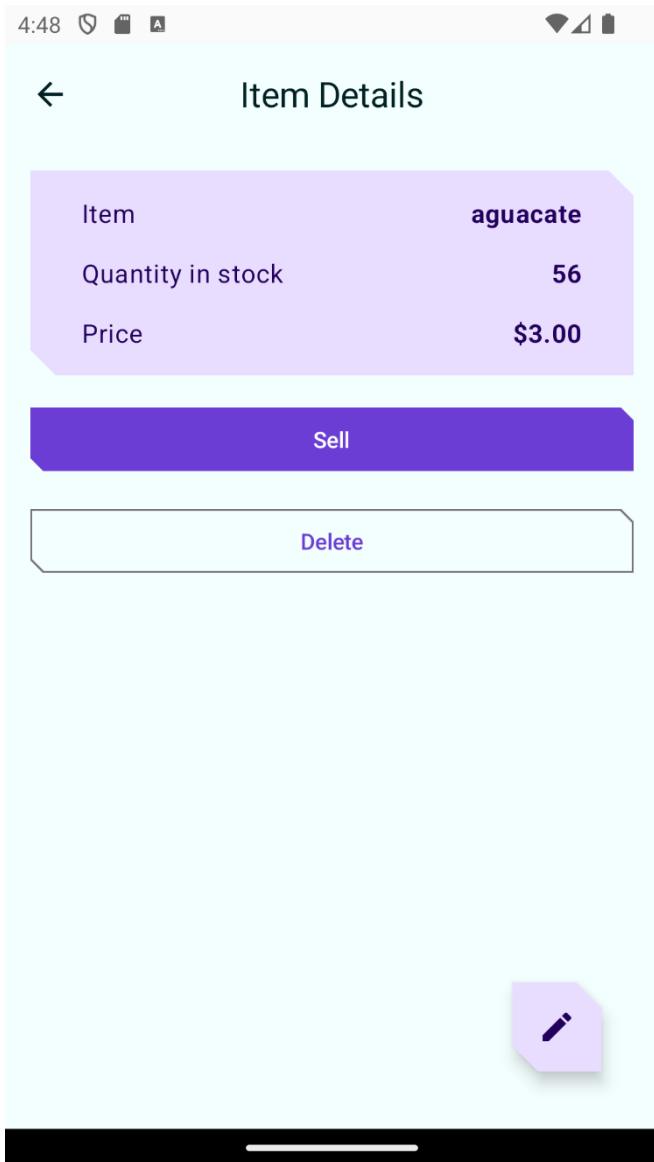
Databases **Items** room_master_table

Live updates

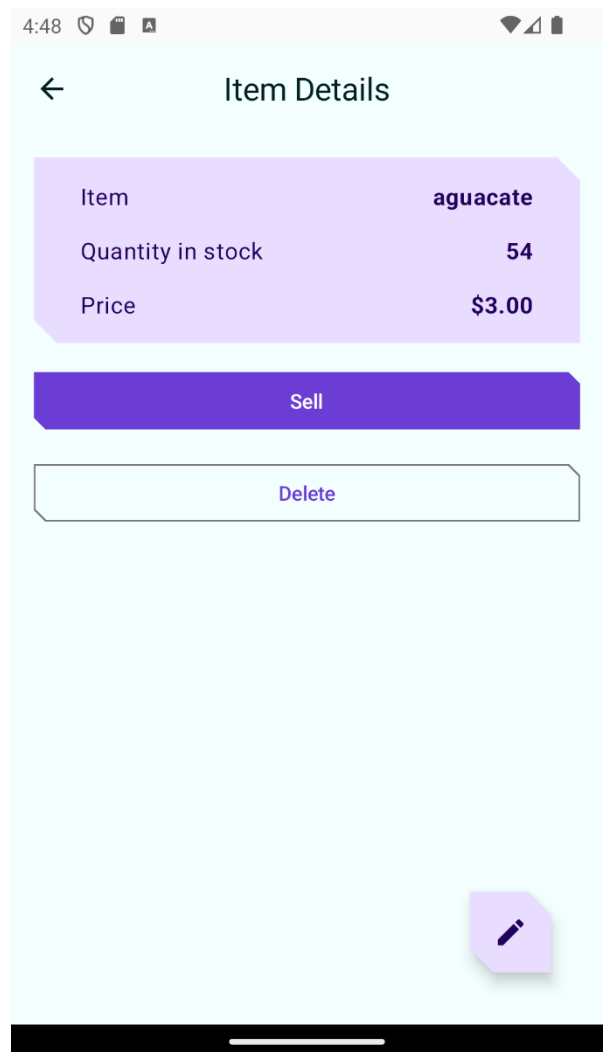
	id	name	price	quantity
1	1	watermelon	4.0	14
2	2	pera	6.0	24
3	3	platano de canarias	1.2	142
4	4	aguacate	3.0	56

PANTALLA DE DETALLES

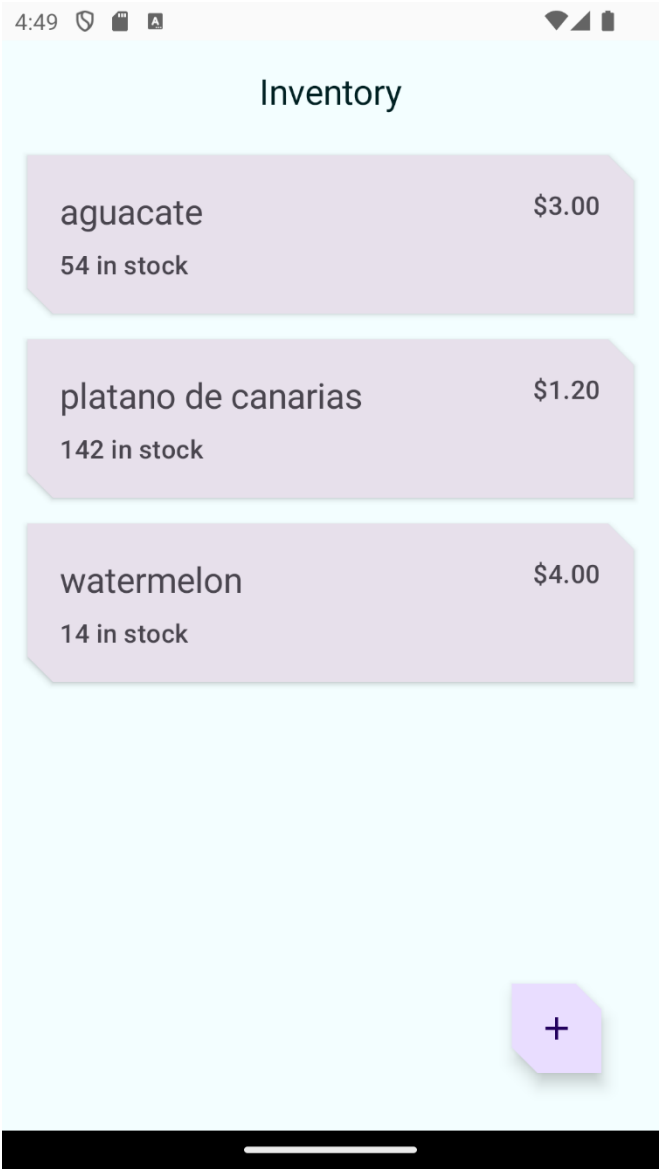
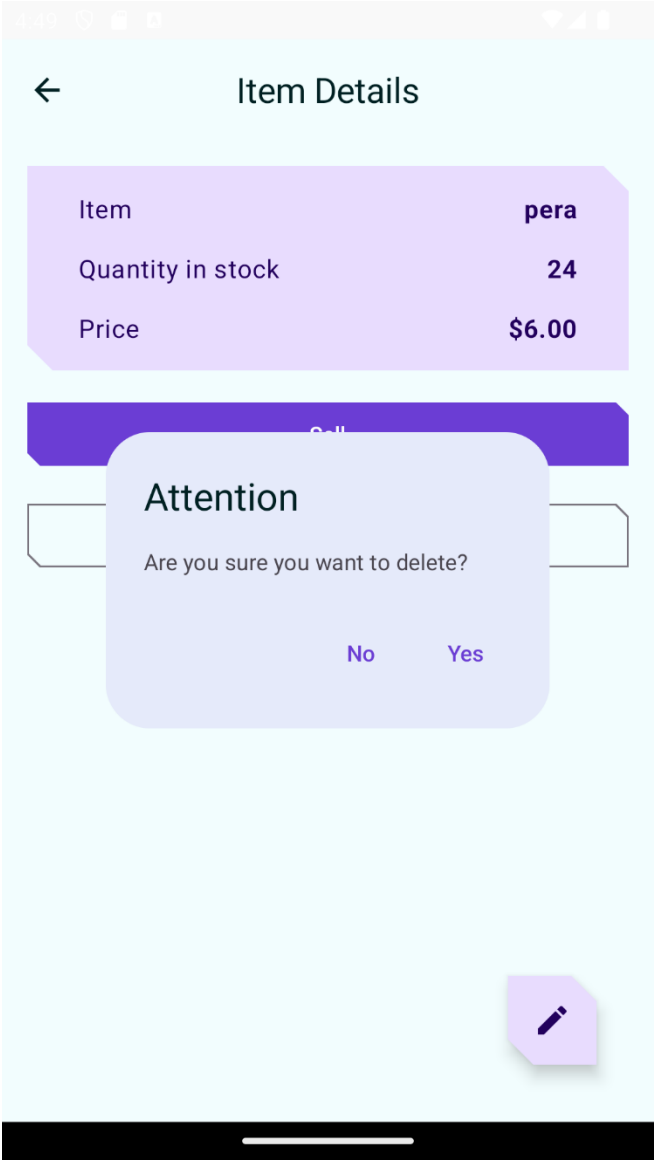
En esta pantalla se nos mostrarán los detalles de nuestros productos, pudiendo clicar en el botón de sell si hemos vendido alguno para actualizarlo o el botón de delete si deseamos eliminar nuestro producto de inventario



Tras clicar 2 veces en sell observamos que ha disminuido la cantidad.



Si clicamos en la opción de delete observamos que nos aparece un panel de atención por si estamos seguros de que queremos eliminarlo, tran decirle que si este lo eliminara y se eliminara tanto del inventario como de la base de datos



Pixel 2 API 34 2 > com.example.inventory

Database Inspector

Select a process to connect to. Fund Task Inspector

Databases

- item_database
 - items
 - room_master_table

Live updates

	id	name	price	quantity
1	1	watermelon	4.0	14
2	3	platano de canarias	1.2	142
3	4	aguacate	3.0	54

PANTALLA EDICIÓN

Si clicamos en el lápiz que nos aparece abajo a la derecha en la pantalla se nos abrirá una nueva interfaz en la cual podremos editar todas las características de nuestro producto(nombre, precio, cantidad).

6:25

← Edit Item

Item Name*
platano de canarias

Item Price*
\$ 1.2

Quantity in Stock*
141

*required fields




Save




Práctica: Compila la app de Bus Schedule

La aplicación se encargara de ofrecernos informacion sobre la hora a la que el siguiente bus que pasa, además del horario especifico de las paradas de buses seleccionadas.

PANTALLA PRINCIPAL

5:18





Bus Schedule

Stop Name	Arrival Time
Main Street	3:00 PM
Park Street	3:12 PM
Maple Avenue	3:25 PM
Broadway Avenue	3:41 PM
Post Street	3:58 PM
Elm Street	4:09 PM
Oak Drive	4:20 PM
Middle Street	4:34 PM
Palm Avenue	4:51 PM
Winding Way	4:55 PM

DAO DATABASE

```
1 package com.example.busschedule.data
2
3 import ...
4
5
6
7 @Dao
8 interface BusScheduleDao {
9
10     @Query ("SELECT * FROM Schedule ORDER BY arrival_time ASC")
11     fun getAll(): Flow<List<BusSchedule>>
12
13     @Query ("SELECT * FROM Schedule WHERE stop_name = :stop_name ORDER BY arrival_time ASC")
14     fun getByStopName(stop_name: String): Flow<List<BusSchedule>>
15 }
```

ROOM DATABASE

```
package com.example.busschedule.data

import ...

@Database(entities = [BusSchedule::class], version = 1)
abstract class DatabaseSchedule: RoomDatabase() {

    abstract fun busScheduleDao(): BusScheduleDao

    companion object {
        @Volatile
        private var Instance: DatabaseSchedule? = null

        fun getDatabase(context: Context): DatabaseSchedule {
            return Instance ?: synchronized(lock: this) {
                Room.databaseBuilder(context, DatabaseSchedule::class.java, name: "database_schedule")
                    .createFromAsset(databaseFilePath: "database/bus_schedule.db")
                    .fallbackToDestructiveMigration()
                    .build() DatabaseSchedule
                    .also { Instance = it }
            }
        }
    }
}
```