TECNOLOGICO JOSE MARIO MOLINA PASQUEL Y ENRRIQUEZ UNIDAD ACADEMICA ZAPOTLANEJO

MANUAL DE PROGRAMADOR ADMINISTRACION Y ORGANIZCION DE DATOS

MARCOS ALFREDO RAZO MARTINEZ

Contenido

Programa 1

Mostrar.dart

pagina1.dart | main.dart | mostrar.dart | basedatoshelper.dart

lib > widget > mostrar.dart > Clase > _modificar

```dart
import 'package:flutter/material.dart';
import 'package:practica1_u4/widget/basedatos/basedatoshelper.dart';

class Mostrar extends StatefulWidget {
  const Mostrar({super.key});

  @override
  State<StatefulWidget> createState() {
    return Clase();
  }
}

class Clase extends State<Mostrar> {
  List<Map<String, dynamic>> usuarios = [];
  String nombre = "";
  int color1 = 0;
  int color2 = 0;
  int color3 = 0;

  @override
  void initState() {
    super.initState();
    cargarDatos();
  }

  void cargarDatos() async {
    List<Map<String, dynamic>> datos = await Basedatoshelper().obtenerUsuario();
    setState(() {
      usuarios = datos;
    });
    print("Usuarios cargados: ${usuarios}");
```

```
void cargarDatos() async {
    usuarios = datos;
  });
  print("Usuarios cargados: ${usuarios}");
}

void eliminar(int id) async {
  await Basedatoshelper().eliminar(id);
  cargarDatos();
}

void _eliminar(int id) {
  eliminar(id);
}

void _modificar(int id, String usuario, String password) {
  TextEditingController usuarionuevo = TextEditingController();
  TextEditingController passwordnuevo = TextEditingController();

  showDialog(
    context: context,
    builder: (context) {
      return AlertDialog(
        title: Text('Editar Usuario'),
        content: SingleChildScrollView(
          child: Column(
            mainAxisSize: MainAxisSize.min,
            children: [
              TextField(
                controller: usuarionuevo,
                decoration: InputDecoration(labelText: '${usuario}'),
```

```
            TextField(
              controller: passwordnuevo,
              decoration: InputDecoration(labelText: 'Escribe el password'),
              obscureText: true,
            ), // TextField
          ],
        ), // Column
      ), // SingleChildScrollView
      actions: [
        TextButton(
          onPressed: () async {
            await Basedatoshelper().modificar(
              id,
              usuarionuevo.text,
              passwordnuevo.text,
            );
            cargarDatos();
            Navigator.pop(context);
          },
          child: Text('Guardar'),
        ), // TextButton
        TextButton(
          onPressed: () => Navigator.pop(context),
          child: Text('Cancelar'),
```

```
            child: Text( Cancelar ),
          ), // TextButton
        ],
      ); // AlertDialog
    },
  );
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Mostrar datos'),
      backgroundColor: Colors.limeAccent,
    ), // AppBar
    body: usuarios.isEmpty
        ? Center(child: Text('Base de datos vacía'))
        : ListView.builder(
            itemCount: usuarios.length,
            itemBuilder: (context, index) {
              nombre = usuarios[index]['usuario'][0];
              return ListTile(
                leading: CircleAvatar(
                  //backgroundColor:  Color.fromARGB(0, 0, 0, 0),
                  child: Text(
                    nombre.toUpperCase(),
                    style: TextStyle(color: Colors.black, fontSize: 24),
                  ), // Text
```

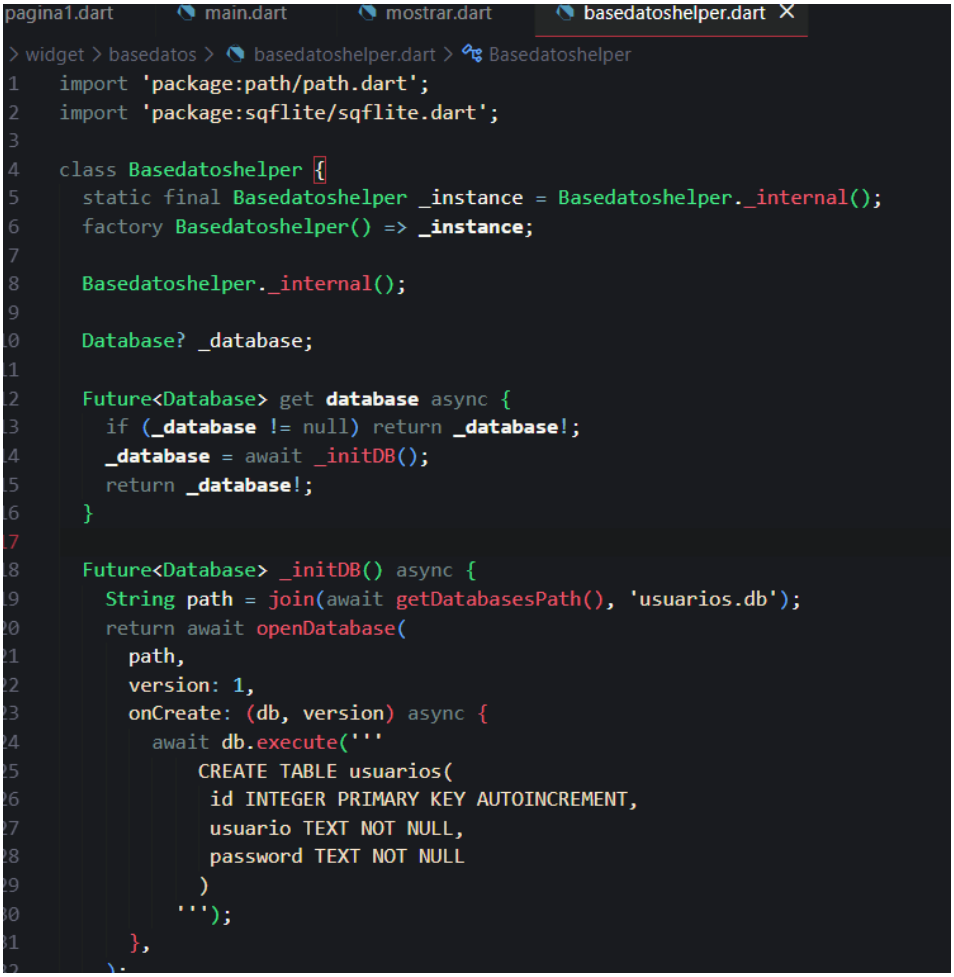
```
                style: TextStyle(color: Colors.black, fontSize: 24),
              ), // Text
            ), // CircleAvatar
            title: Text(usuarios[index]['usuario']),
            subtitle: Text(usuarios[index]['password']),
            trailing: Wrap(
              direction: Axis.horizontal,
              children: [
                IconButton(
                  onPressed: () {
                    _eliminar(usuarios[index]['id']);
                  },
                  icon: Icon(Icons.delete),
                ), // IconButton
                IconButton(
                  onPressed: ()
                    _modificar(
                      usuarios[index]['id'],
                      usuarios[index]['usuario'],
                      usuarios[index]['password'],
                    );
                  },
                  icon: Icon(Icons.update),
```

int index

Type: int

```
                  },
                  icon: Icon(Icons.update),
                ), // IconButton
              ],
            ), // Wrap
          ); // ListTile
        },
      ), // ListView.builder
    ); // Scaffold
  }
}
```

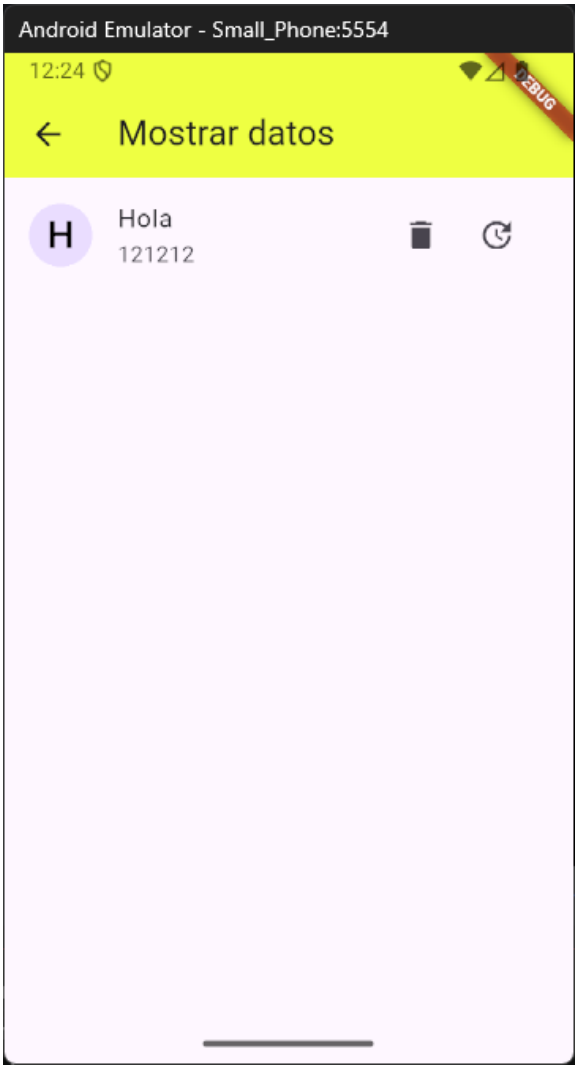
Basedatoshealper.dart



```
            password TEXT NOT NULL
          )
        ''');
      },
    );
  }

  Future<int> insertar(String usuario, String password) async {
    final db = await database;
    return await db.insert('usuarios', {
      'usuario': usuario,
      'password': password,
    });
  }

  Future<List<Map<String, dynamic>>> obtenerUsuario() async {
    final db = await database;
    return await db.query('usuarios');
  }

  Future eliminar(int id) async {
    final db = await database;
    return await db.delete('usuarios', where: 'id = ?', whereArgs: [id]);
  }

  Future<int> modificar(int nid, String nusuario, String npassword) async {
    final db = await database;
    return await db.update(
```

```
    final db = await database;
    return await db.update(
      'usuarios',
      {'usuario': nusuario, 'password': npassword},
      where: 'id = ?',
      whereArgs: [nid],
    );
  }
}
```

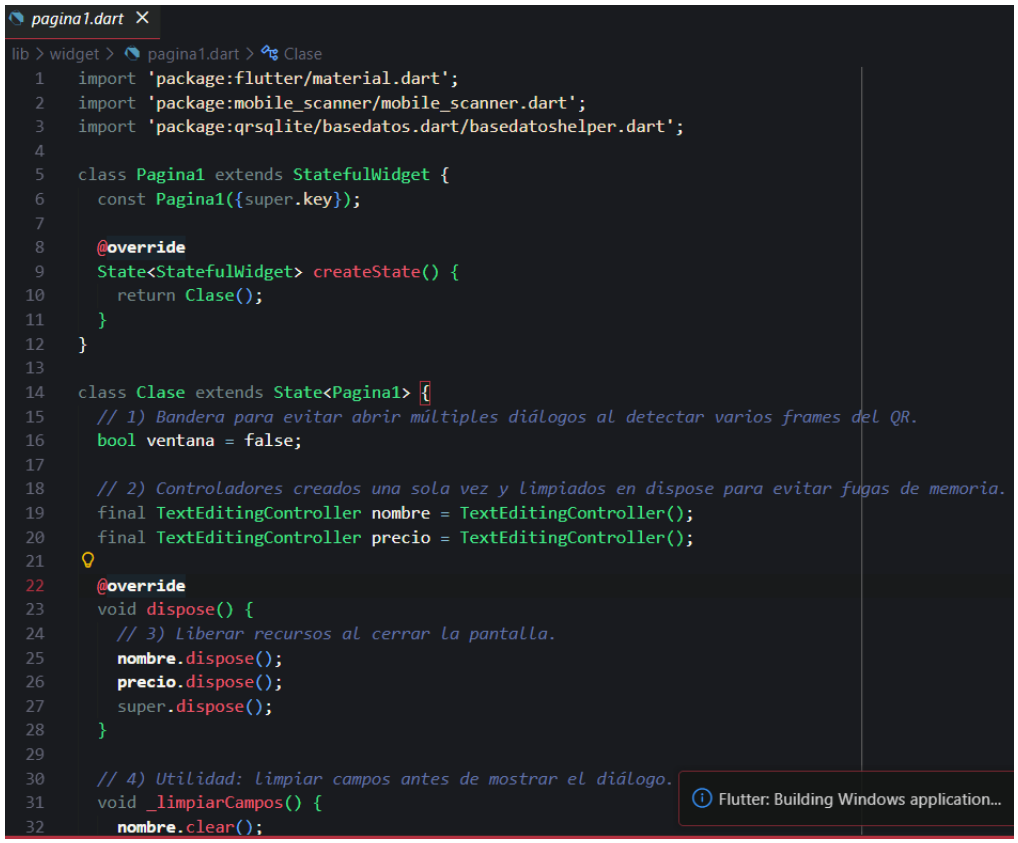
Resultado


Android Emulator - Small_Phone:5554
12:24
DEBUG
Mostrar datos
H
Hola
121212


Android Emulator - Small_Phone:5554
12:24
DEBUG
Pagina 1
Escribe el usuario
Escribe el password
***
Agregar
Mostrar Usuarios

Qrsqlite

Pagina1.dart

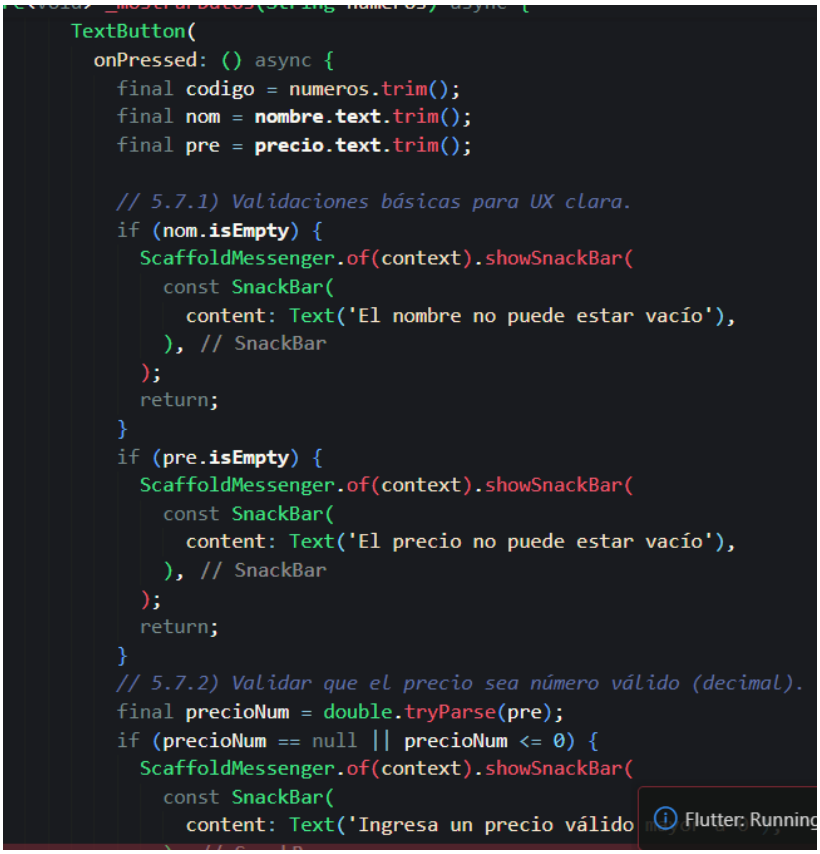


```dart
class Clase extends State<Pagina1> {
  void _limpiarCampos() {
    nombre.clear();
    precio.clear();
  }

  // 5) Diálogo para capturar datos; recibe el código QR como 'numeros'.
  Future<void> _mostrarDatos(String numeros) async {
    _limpiarCampos(); // 5.1) Asegurar campos en blanco al abrir.

    // 5.2) showDialog devuelve un Future; esperamos (await) para saber cuándo cerrar y poder resetear 'v
    await showDialog(
      context: context,
      barrierDismissible:
          false, // 5.3) Evitar cierres accidentales al tocar fuera.
      builder: (context) {
        return AlertDialog(
          title: const Text('Agregar producto'),
          content: SingleChildScrollView(
            child: Column(
              mainAxisSize: MainAxisSize.min,
              children: [
                // 5.4) Mostrar el código leído.
                Text(
                  'El código: $numeros',
                  style: const TextStyle(fontSize: 15),
                ), // Text
                const SizedBox(height: 12),
                // 5.5) Campo nombre (texto normal).
                TextField(
                  controller: nombre,
```

Flutter: Running Gradle task 'assembleDebug'...

```dart
Future<void> _mostrarDatos(String numeros) async {
                // 5.5) Campo nombre (texto normal).
                TextField(
                  controller: nombre,
                  decoration: const InputDecoration(
                    labelText: 'Escribe el nombre del producto',
                    border: OutlineInputBorder(),
                  ), // InputDecoration
                  textInputAction: TextInputAction.next,
                ), // TextField
                const SizedBox(height: 12),
                // 5.6) Campo precio (numérico). Quitar obscureText: no es contraseña.
                TextField(
                  controller: precio,
                  decoration: const InputDecoration(
                    labelText: 'Escribe el precio',
                    border: OutlineInputBorder(),
                  ), // InputDecoration
                  keyboardType: TextInputType.numberWithOptions(decimal: true),
                ), // TextField
              ],
            ), // Column
          ), // SingleChildScrollView
          actions: [
            // 5.7) Guardar: validar mínimo y luego insertar.
            TextButton(
              onPressed: () async {
                final codigo = numeros.trim();
                final nom = nombre.text.trim();
                final pre = precio.text.trim();
```

Flutter: Running Gradle task '



```dart
TextButton(
  onPressed: () async {
    final codigo = numeros.trim();
    final nom = nombre.text.trim();
    final pre = precio.text.trim();

    // 5.7.1) Validaciones básicas para UX clara.
    if (nom.isEmpty) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
          content: Text('El nombre no puede estar vacío'),
        ), // SnackBar
      );
      return;
    }
    if (pre.isEmpty) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
          content: Text('El precio no puede estar vacío'),
        ), // SnackBar
      );
      return;
    }
    // 5.7.2) Validar que el precio sea número válido (decimal).
    final precioNum = double.tryParse(pre);
    if (precioNum == null || precioNum <= 0) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
          content: Text('Ingresa un precio válido
```

Flutter: Running

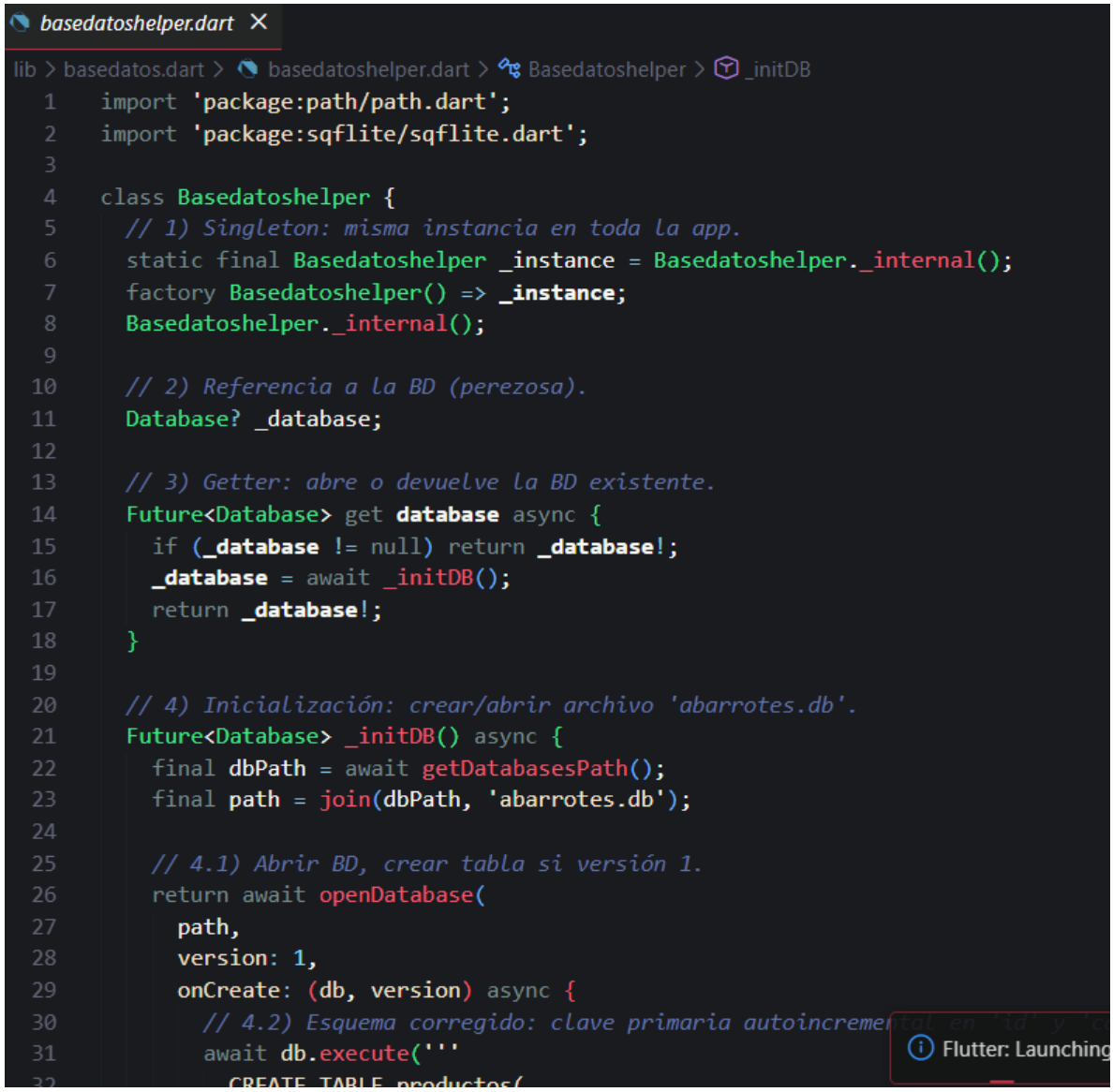
```
  Future<void> _mostrarDatos(String numeros) async {
                ScaffoldMessenger.of(context).showSnackBar(
                  const SnackBar(
                    content: Text('Ingresa un precio válido mayor a 0'),
                  ), // SnackBar
                );
                return;
              }

              try {
                // 5.7.3) Insertar en BD; guardar precio como texto (consistente
                await Basedatoshelper().insertar(codigo, nom, pre);

                // 5.7.4) Feedback de éxito opcional.
                ScaffoldMessenger.of(context).showSnackBar(
                  const SnackBar(content: Text('Producto guardado')),
                );

                // 5.7.5) Cerrar el diálogo tras guardar.
                Navigator.pop(context);
              } catch (e) {
                // 5.7.6) Manejo de error en inserción.
                ScaffoldMessenger.of(context).showSnackBar(
                  SnackBar(content: Text('Error al guardar: $e')),
                );
              }
            },
            child: const Text('Guardar'),
          ), // TextButton
          // 5.8) Cancelar: cerrar sin acciones.
```

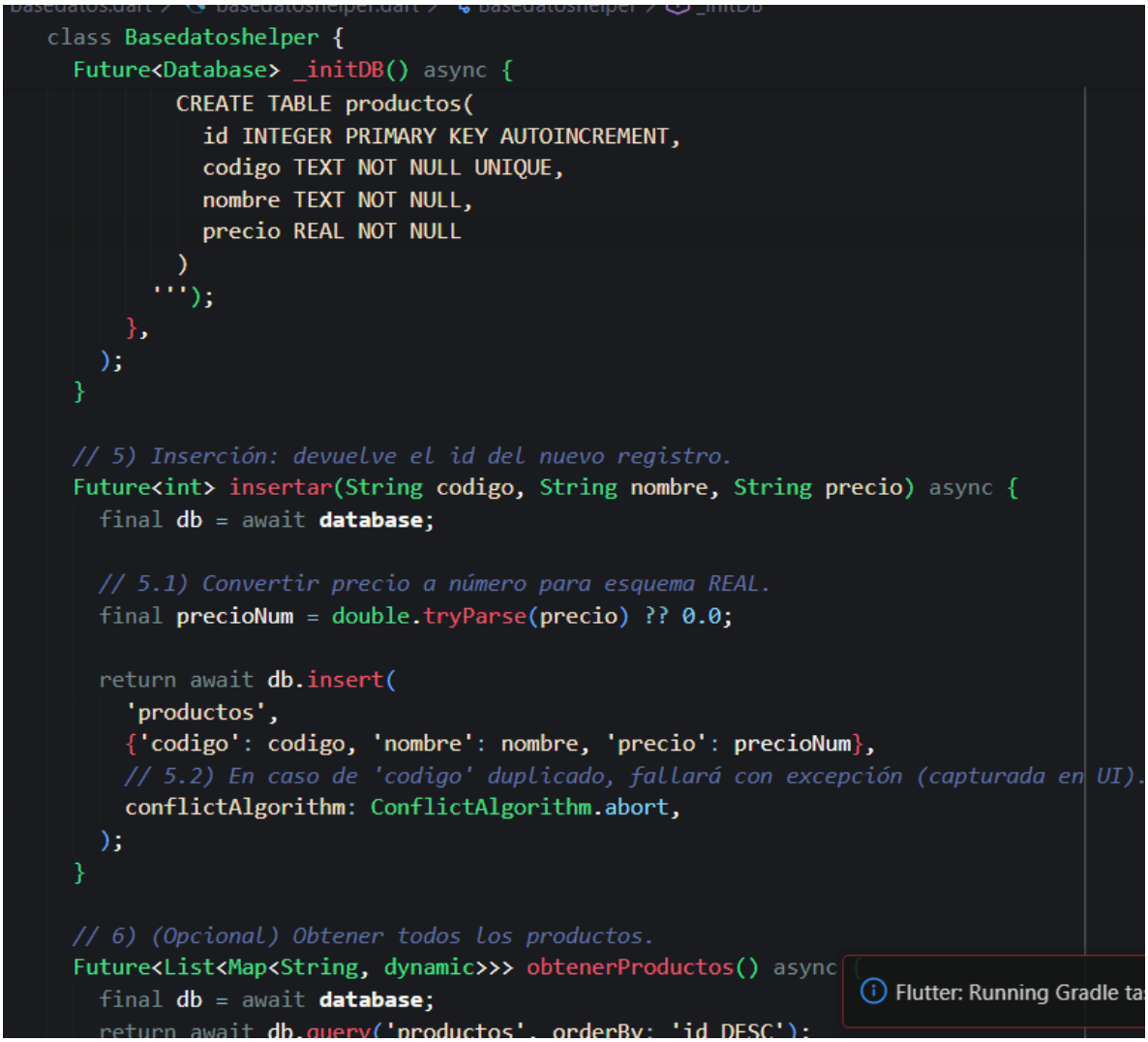
```
class Clase extends State<Pagina1> {
  Future<void> _mostrarDatos(String numeros) async {
          ), // TextButton
          // 5.8) Cancelar: cerrar sin acciones.
          TextButton(
            onPressed: () => Navigator.pop(context),
            child: const Text('Cancelar'),
          ), // TextButton
        ],
      ); // AlertDialog
    },
  );

  // 5.9) Al cerrar el diálogo (éxito o cancelación), permitir otro escaneo.
  setState(() {
    ventana = false;
  });
}
```

```dart
class Clase extends State<Pagina1> {
  @override
  Widget build(BuildContext context) {
    // 6) Estructura principal de la pantalla.
    return Scaffold(
      appBar: AppBar(
        title: const Text('QR con SQFlite'),
        backgroundColor: const Color.fromARGB(255, 5, 255, 17),
      ), // AppBar
      // 7) Lector de QR: 'onDetect' puede dispararse muchas veces por segundo.
      body: MobileScanner(
        onDetect: (capture) {
          // 7.1) Evitar múltiples diálogos simultáneos.
          if (ventana) return;

          // 7.2) Tomar el primer código detectado.
          final barcode = capture.barcodes.first;
          final numeros = barcode.rawValue ?? 'Sin codigo';

          // 7.3) Si hay código válido, abrir diálogo y bloquear nueva ventana.
          if (numeros != 'Sin codigo' && numeros.trim().isNotEmpty) {
            setState(() {
              ventana = true;
            });
            _mostrarDatos(numeros);
          }
        },
      ), // MobileScanner
    ); // Scaffold
  }
}
```

Flutter: Running Grad

Basedatoshelper.dart




class Basedatoshelper {
Future<Database> _initDB() async {
CREATE TABLE productos(
id INTEGER PRIMARY KEY AUTOINCREMENT,
codigo TEXT NOT NULL UNIQUE,
nombre TEXT NOT NULL,
precio REAL NOT NULL
)
''');
},
);
}
// 5) Inserción: devuelve el id del nuevo registro.
Future<int> insertar(String codigo, String nombre, String precio) async {
final db = await database;
// 5.1) Convertir precio a número para esquema REAL.
final precioNum = double.tryParse(precio) ?? 0.0;
return await db.insert(
'productos',
{'codigo': codigo, 'nombre': nombre, 'precio': precioNum},
// 5.2) En caso de 'codigo' duplicado, fallará con excepción (capturada en UI).
conflictAlgorithm: ConflictAlgorithm.abort,
);
}
// 6) (Opcional) Obtener todos los productos.
Future<List<Map<String, dynamic>>> obtenerProductos() async
final db = await database;
Flutter: Running Gradle ta

```dart
  // 6) (Opcional) Obtener todos los productos.
  Future<List<Map<String, dynamic>>> obtenerProductos() async {
    final db = await database;
    return await db.query('productos', orderBy: 'id DESC');
  }

  // 7) (Opcional) Eliminar por id.
  Future<int> eliminar(int id) async {
    final db = await database;
    return await db.delete('productos', where: 'id = ?', whereArgs: [id]);
  }

  // 8) (Opcional) Actualizar nombre y precio por código.
  Future<int> modificarPorCodigo(
    String codigo,
    String nombre,
    double precio,
  ) async {
    final db = await database;
    return await db.update(
      'productos',
      {'nombre': nombre, 'precio': precio},
      where: 'codigo = ?',
      whereArgs: [codigo],
      conflictAlgorithm: ConflictAlgorithm.abort,
    );
  }
}
```

Resultado