

TECNOLOGICO JOSE MARIO MOLINA
PASQUEL Y ENRIQUEZ UNIDAD
ACADEMICA ZAPOTLANEJO

MANUAL DE PROGRAMADOR
ADMINISTRACION Y ORGANIZACION DE
DATOS

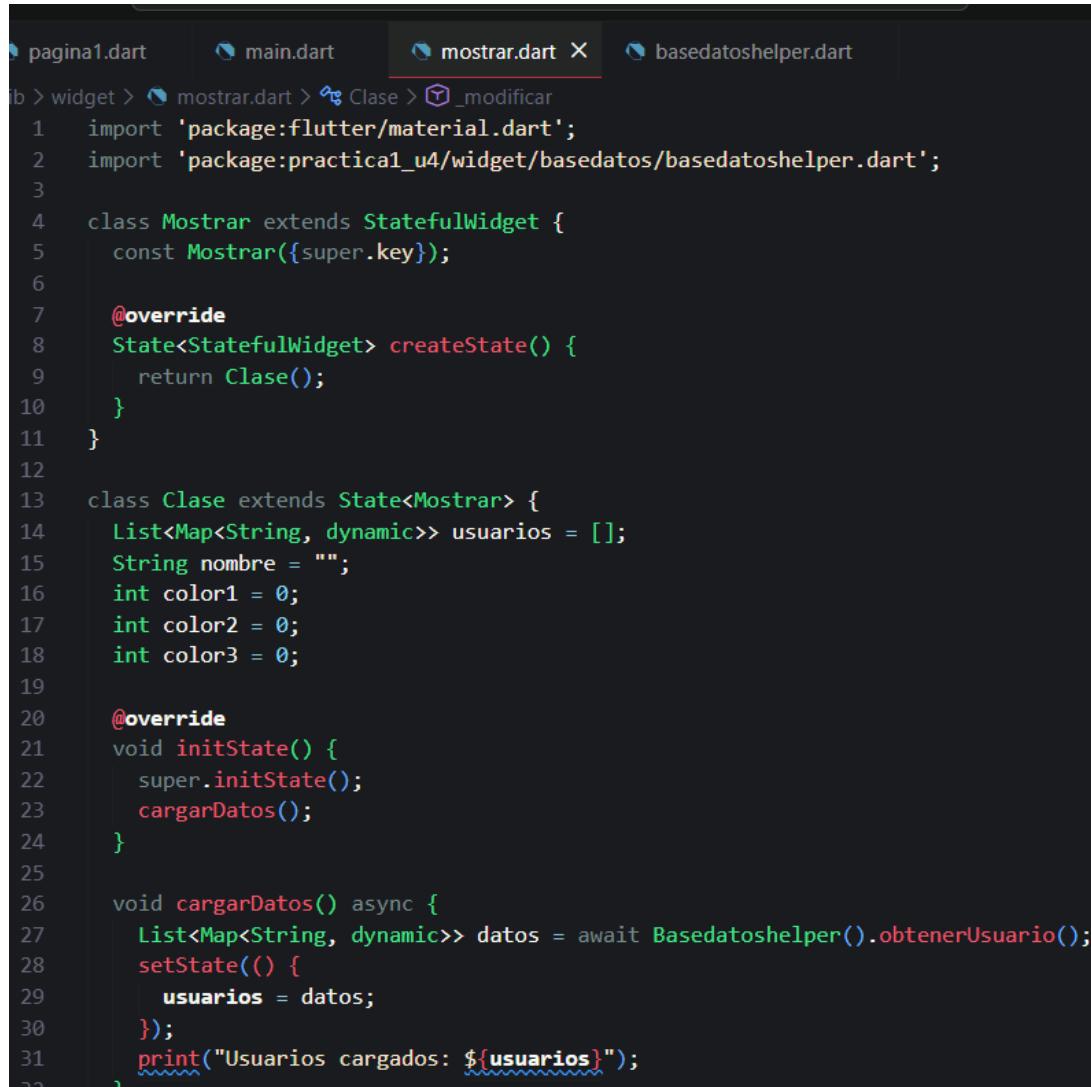
MARCOS ALFREDO RAZO MARTINEZ

Contenido

Programa 1	3
Mostrar.dart.....	3
Basedatosheelper.dart.....	8
Resultado	10

Programa 1

Mostrar.dart



```
pagina1.dart    main.dart    mostrar.dart X    basedatoshelper.dart
ib > widget > mostrar.dart > Clase > _modificar
1 import 'package:flutter/material.dart';
2 import 'package:practica1_u4/widget/basedatos/basedatoshelper.dart';
3
4 class Mostrar extends StatefulWidget {
5   const Mostrar({super.key});
6
7   @override
8   State<StatefulWidget> createState() {
9     return Clase();
10   }
11 }
12
13 class Clase extends State<Mostrar> {
14   List<Map<String, dynamic>> usuarios = [];
15   String nombre = "";
16   int color1 = 0;
17   int color2 = 0;
18   int color3 = 0;
19
20   @override
21   void initState() {
22     super.initState();
23     cargarDatos();
24   }
25
26   void cargarDatos() async {
27     List<Map<String, dynamic>> datos = await Basedatoshelper().obtenerUsuario();
28     setState(() {
29       usuarios = datos;
30     });
31     print("Usuarios cargados: ${usuarios}");
32 }
```

```
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,
                usuariónuevo.text,
                passwordnuevo.text,
            );
            cargarDatos();
            Navigator.pop(context);
        },
        child: Text('Guardar'),
    ), // TextButton
    TextButton(
        onPressed: () => Navigator.pop(context),
        child: Text('Cancelar'),
    ),
]
```

```
        child: Text('Cancelar'),
    ),
],
);
},
);
}

@Override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
            title: Text('Mostrar datos'),
            backgroundColor: Colors.limeAccent,
        ),
        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),
                            ),
                        ),
                    );
                },
            ),
    );
}
```

```
        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(Type: int
                    usuarios[index]['id'],
                    usuarios[index]['usuario'],
                    usuarios[index]['password'],
                );
            },
            icon: Icon(Icons.update),
        ),
        ],
    ), // Wrap
); // ListTile
},
), // ListView.builder
); // Scaffold
}
}
```

Basedatoshelper.dart

```
pagina1.dart    main.dart    mostrar.dart    basedatoshelper.dart X
> widget > basedatos > basedatoshelper.dart > Basedatoshelper
1   import 'package:path/path.dart';
2   import 'package:sqflite/sqflite.dart';
3
4   class Basedatoshelper {
5       static final Basedatoshelper _instance = Basedatoshelper._internal();
6       factory Basedatoshelper() => _instance;
7
8       Basedatoshelper._internal();
9
10      Database? _database;
11
12      Future<Database> get database async {
13          if (_database != null) return _database!;
14          _database = await _initDB();
15          return _database!;
16      }
17
18      Future<Database> _initDB() async {
19          String path = join(await getDatabasesPath(), 'usuarios.db');
20          return await openDatabase(
21              path,
22              version: 1,
23              onCreate: (db, version) async {
24                  await db.execute('''
25                      CREATE TABLE usuarios(
26                          id INTEGER PRIMARY KEY AUTOINCREMENT,
27                          usuario TEXT NOT NULL,
28                          password TEXT NOT NULL
29                      )
30                      ''');
31              },
32          );
33      }
34  }
```

```
        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');
}

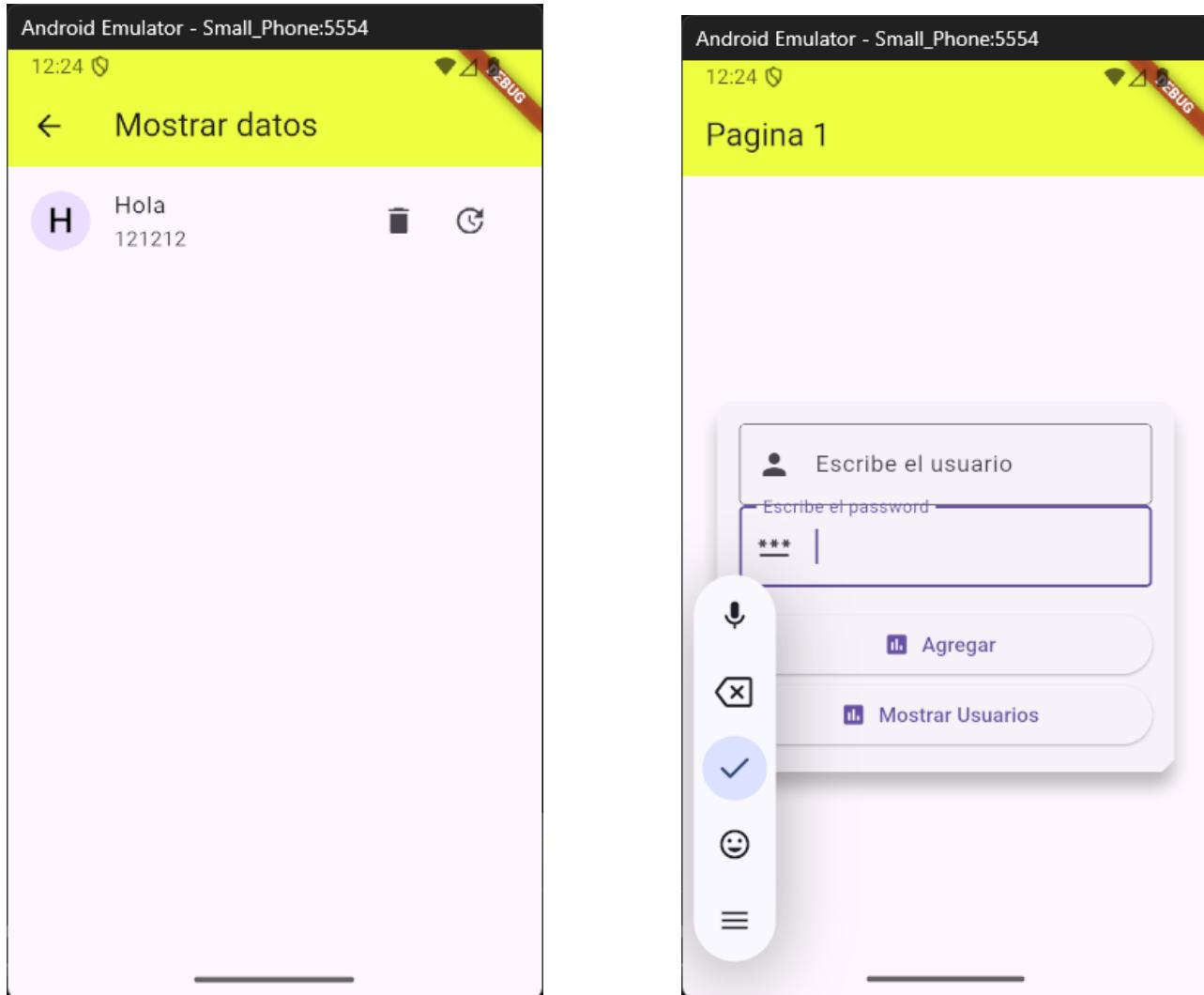
Future eliminar(int id) async {
    final db = await database;
    return await db.delete('usuarios', where: 'id = ?', whereArgs: [id]);
}

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



Qrsqlite

Pagina1.dart

```
pagina1.dart X
lib > widget > pagina1.dart > Clase
1 import 'package:flutter/material.dart';
2 import 'package:mobile_scanner/mobile_scanner.dart';
3 import 'package:qrsqlite/basedatos.dart/basedatoshelper.dart';
4
5 class Pagina1 extends StatefulWidget {
6   const Pagina1({super.key});
7
8   @override
9   State< StatefulWidget > createState() {
10    return Clase();
11  }
12}
13
14 class Clase extends State< Pagina1 > {
15   // 1) Bandera para evitar abrir múltiples diálogos al detectar varios frames del QR.
16   bool ventana = false;
17
18   // 2) Controladores creados una sola vez y limpiados en dispose para evitar fugas de memoria.
19   final TextEditingController nombre = TextEditingController();
20   final TextEditingController precio = TextEditingController();
21
22   @override
23   void dispose() {
24     // 3) Liberar recursos al cerrar la pantalla.
25     nombre.dispose();
26     precio.dispose();
27     super.dispose();
28   }
29
30   // 4) Utilidad: limpiar campos antes de mostrar el diálogo.
31   void _limpiarCampos() {
32     nombre.clear();
33   }
34
35   void _mostrarQR() {
36     Navigator.push(
37       context,
38       MaterialPageRoute(
39         builder: (context) => QRScannerScreen(),
40       ),
41     );
42   }
43
44   void _mostrarQR() {
45     Navigator.push(
46       context,
47       MaterialPageRoute(
48         builder: (context) => QRScannerScreen(),
49       ),
50     );
51   }
52
53   void _mostrarQR() {
54     Navigator.push(
55       context,
56       MaterialPageRoute(
57         builder: (context) => QRScannerScreen(),
58       ),
59     );
60   }
61
62   void _mostrarQR() {
63     Navigator.push(
64       context,
65       MaterialPageRoute(
66         builder: (context) => QRScannerScreen(),
67       ),
68     );
69   }
70
71   void _mostrarQR() {
72     Navigator.push(
73       context,
74       MaterialPageRoute(
75         builder: (context) => QRScannerScreen(),
76       ),
77     );
78   }
79
80   void _mostrarQR() {
81     Navigator.push(
82       context,
83       MaterialPageRoute(
84         builder: (context) => QRScannerScreen(),
85       ),
86     );
87   }
88
89   void _mostrarQR() {
90     Navigator.push(
91       context,
92       MaterialPageRoute(
93         builder: (context) => QRScannerScreen(),
94       ),
95     );
96   }
97
98   void _mostrarQR() {
99     Navigator.push(
100      context,
101      MaterialPageRoute(
102        builder: (context) => QRScannerScreen(),
103      ),
104    );
105  }
106
107  void _mostrarQR() {
108    Navigator.push(
109      context,
110      MaterialPageRoute(
111        builder: (context) => QRScannerScreen(),
112      ),
113    );
114  }
115
116  void _mostrarQR() {
117    Navigator.push(
118      context,
119      MaterialPageRoute(
120        builder: (context) => QRScannerScreen(),
121      ),
122    );
123  }
124
125  void _mostrarQR() {
126    Navigator.push(
127      context,
128      MaterialPageRoute(
129        builder: (context) => QRScannerScreen(),
130      ),
131    );
132  }
133
134  void _mostrarQR() {
135    Navigator.push(
136      context,
137      MaterialPageRoute(
138        builder: (context) => QRScannerScreen(),
139      ),
140    );
141  }
142
143  void _mostrarQR() {
144    Navigator.push(
145      context,
146      MaterialPageRoute(
147        builder: (context) => QRScannerScreen(),
148      ),
149    );
150  }
151
152  void _mostrarQR() {
153    Navigator.push(
154      context,
155      MaterialPageRoute(
156        builder: (context) => QRScannerScreen(),
157      ),
158    );
159  }
160
161  void _mostrarQR() {
162    Navigator.push(
163      context,
164      MaterialPageRoute(
165        builder: (context) => QRScannerScreen(),
166      ),
167    );
168  }
169
170  void _mostrarQR() {
171    Navigator.push(
172      context,
173      MaterialPageRoute(
174        builder: (context) => QRScannerScreen(),
175      ),
176    );
177  }
178
179  void _mostrarQR() {
180    Navigator.push(
181      context,
182      MaterialPageRoute(
183        builder: (context) => QRScannerScreen(),
184      ),
185    );
186  }
187
188  void _mostrarQR() {
189    Navigator.push(
190      context,
191      MaterialPageRoute(
192        builder: (context) => QRScannerScreen(),
193      ),
194    );
195  }
196
197  void _mostrarQR() {
198    Navigator.push(
199      context,
200      MaterialPageRoute(
201        builder: (context) => QRScannerScreen(),
202      ),
203    );
204  }
205
206  void _mostrarQR() {
207    Navigator.push(
208      context,
209      MaterialPageRoute(
210        builder: (context) => QRScannerScreen(),
211      ),
212    );
213  }
214
215  void _mostrarQR() {
216    Navigator.push(
217      context,
218      MaterialPageRoute(
219        builder: (context) => QRScannerScreen(),
220      ),
221    );
222  }
223
224  void _mostrarQR() {
225    Navigator.push(
226      context,
227      MaterialPageRoute(
228        builder: (context) => QRScannerScreen(),
229      ),
230    );
231  }
232
233  void _mostrarQR() {
234    Navigator.push(
235      context,
236      MaterialPageRoute(
237        builder: (context) => QRScannerScreen(),
238      ),
239    );
240  }
241
242  void _mostrarQR() {
243    Navigator.push(
244      context,
245      MaterialPageRoute(
246        builder: (context) => QRScannerScreen(),
247      ),
248    );
249  }
250
251  void _mostrarQR() {
252    Navigator.push(
253      context,
254      MaterialPageRoute(
255        builder: (context) => QRScannerScreen(),
256      ),
257    );
258  }
259
260  void _mostrarQR() {
261    Navigator.push(
262      context,
263      MaterialPageRoute(
264        builder: (context) => QRScannerScreen(),
265      ),
266    );
267  }
268
269  void _mostrarQR() {
270    Navigator.push(
271      context,
272      MaterialPageRoute(
273        builder: (context) => QRScannerScreen(),
274      ),
275    );
276  }
277
278  void _mostrarQR() {
279    Navigator.push(
280      context,
281      MaterialPageRoute(
282        builder: (context) => QRScannerScreen(),
283      ),
284    );
285  }
286
287  void _mostrarQR() {
288    Navigator.push(
289      context,
290      MaterialPageRoute(
291        builder: (context) => QRScannerScreen(),
292      ),
293    );
294  }
295
296  void _mostrarQR() {
297    Navigator.push(
298      context,
299      MaterialPageRoute(
300        builder: (context) => QRScannerScreen(),
301      ),
302    );
303  }
304
305  void _mostrarQR() {
306    Navigator.push(
307      context,
308      MaterialPageRoute(
309        builder: (context) => QRScannerScreen(),
310      ),
311    );
312  }
313
314  void _mostrarQR() {
315    Navigator.push(
316      context,
317      MaterialPageRoute(
318        builder: (context) => QRScannerScreen(),
319      ),
320    );
321  }
322
323  void _mostrarQR() {
324    Navigator.push(
325      context,
326      MaterialPageRoute(
327        builder: (context) => QRScannerScreen(),
328      ),
329    );
330  }
331
332  void _mostrarQR() {
333    Navigator.push(
334      context,
335      MaterialPageRoute(
336        builder: (context) => QRScannerScreen(),
337      ),
338    );
339  }
340
341  void _mostrarQR() {
342    Navigator.push(
343      context,
344      MaterialPageRoute(
345        builder: (context) => QRScannerScreen(),
346      ),
347    );
348  }
349
350  void _mostrarQR() {
351    Navigator.push(
352      context,
353      MaterialPageRoute(
354        builder: (context) => QRScannerScreen(),
355      ),
356    );
357  }
358
359  void _mostrarQR() {
360    Navigator.push(
361      context,
362      MaterialPageRoute(
363        builder: (context) => QRScannerScreen(),
364      ),
365    );
366  }
367
368  void _mostrarQR() {
369    Navigator.push(
370      context,
371      MaterialPageRoute(
372        builder: (context) => QRScannerScreen(),
373      ),
374    );
375  }
376
377  void _mostrarQR() {
378    Navigator.push(
379      context,
380      MaterialPageRoute(
381        builder: (context) => QRScannerScreen(),
382      ),
383    );
384  }
385
386  void _mostrarQR() {
387    Navigator.push(
388      context,
389      MaterialPageRoute(
390        builder: (context) => QRScannerScreen(),
391      ),
392    );
393  }
394
395  void _mostrarQR() {
396    Navigator.push(
397      context,
398      MaterialPageRoute(
399        builder: (context) => QRScannerScreen(),
400      ),
401    );
402  }
403
404  void _mostrarQR() {
405    Navigator.push(
406      context,
407      MaterialPageRoute(
408        builder: (context) => QRScannerScreen(),
409      ),
410    );
411  }
412
413  void _mostrarQR() {
414    Navigator.push(
415      context,
416      MaterialPageRoute(
417        builder: (context) => QRScannerScreen(),
418      ),
419    );
420  }
421
422  void _mostrarQR() {
423    Navigator.push(
424      context,
425      MaterialPageRoute(
426        builder: (context) => QRScannerScreen(),
427      ),
428    );
429  }
430
431  void _mostrarQR() {
432    Navigator.push(
433      context,
434      MaterialPageRoute(
435        builder: (context) => QRScannerScreen(),
436      ),
437    );
438  }
439
440  void _mostrarQR() {
441    Navigator.push(
442      context,
443      MaterialPageRoute(
444        builder: (context) => QRScannerScreen(),
445      ),
446    );
447  }
448
449  void _mostrarQR() {
450    Navigator.push(
451      context,
452      MaterialPageRoute(
453        builder: (context) => QRScannerScreen(),
454      ),
455    );
456  }
457
458  void _mostrarQR() {
459    Navigator.push(
460      context,
461      MaterialPageRoute(
462        builder: (context) => QRScannerScreen(),
463      ),
464    );
465  }
466
467  void _mostrarQR() {
468    Navigator.push(
469      context,
470      MaterialPageRoute(
471        builder: (context) => QRScannerScreen(),
472      ),
473    );
474  }
475
476  void _mostrarQR() {
477    Navigator.push(
478      context,
479      MaterialPageRoute(
480        builder: (context) => QRScannerScreen(),
481      ),
482    );
483  }
484
485  void _mostrarQR() {
486    Navigator.push(
487      context,
488      MaterialPageRoute(
489        builder: (context) => QRScannerScreen(),
490      ),
491    );
492  }
493
494  void _mostrarQR() {
495    Navigator.push(
496      context,
497      MaterialPageRoute(
498        builder: (context) => QRScannerScreen(),
499      ),
500    );
501  }
502
503  void _mostrarQR() {
504    Navigator.push(
505      context,
506      MaterialPageRoute(
507        builder: (context) => QRScannerScreen(),
508      ),
509    );
510  }
511
512  void _mostrarQR() {
513    Navigator.push(
514      context,
515      MaterialPageRoute(
516        builder: (context) => QRScannerScreen(),
517      ),
518    );
519  }
520
521  void _mostrarQR() {
522    Navigator.push(
523      context,
524      MaterialPageRoute(
525        builder: (context) => QRScannerScreen(),
526      ),
527    );
528  }
529
530  void _mostrarQR() {
531    Navigator.push(
532      context,
533      MaterialPageRoute(
534        builder: (context) => QRScannerScreen(),
535      ),
536    );
537  }
538
539  void _mostrarQR() {
540    Navigator.push(
541      context,
542      MaterialPageRoute(
543        builder: (context) => QRScannerScreen(),
544      ),
545    );
546  }
547
548  void _mostrarQR() {
549    Navigator.push(
550      context,
551      MaterialPageRoute(
552        builder: (context) => QRScannerScreen(),
553      ),
554    );
555  }
556
557  void _mostrarQR() {
558    Navigator.push(
559      context,
560      MaterialPageRoute(
561        builder: (context) => QRScannerScreen(),
562      ),
563    );
564  }
565
566  void _mostrarQR() {
567    Navigator.push(
568      context,
569      MaterialPageRoute(
570        builder: (context) => QRScannerScreen(),
571      ),
572    );
573  }
574
575  void _mostrarQR() {
576    Navigator.push(
577      context,
578      MaterialPageRoute(
579        builder: (context) => QRScannerScreen(),
580      ),
581    );
582  }
583
584  void _mostrarQR() {
585    Navigator.push(
586      context,
587      MaterialPageRoute(
588        builder: (context) => QRScannerScreen(),
589      ),
590    );
591  }
592
593  void _mostrarQR() {
594    Navigator.push(
595      context,
596      MaterialPageRoute(
597        builder: (context) => QRScannerScreen(),
598      ),
599    );
600  }
601
602  void _mostrarQR() {
603    Navigator.push(
604      context,
605      MaterialPageRoute(
606        builder: (context) => QRScannerScreen(),
607      ),
608    );
609  }
610
611  void _mostrarQR() {
612    Navigator.push(
613      context,
614      MaterialPageRoute(
615        builder: (context) => QRScannerScreen(),
616      ),
617    );
618  }
619
620  void _mostrarQR() {
621    Navigator.push(
622      context,
623      MaterialPageRoute(
624        builder: (context) => QRScannerScreen(),
625      ),
626    );
627  }
628
629  void _mostrarQR() {
630    Navigator.push(
631      context,
632      MaterialPageRoute(
633        builder: (context) => QRScannerScreen(),
634      ),
635    );
636  }
637
638  void _mostrarQR() {
639    Navigator.push(
640      context,
641      MaterialPageRoute(
642        builder: (context) => QRScannerScreen(),
643      ),
644    );
645  }
646
647  void _mostrarQR() {
648    Navigator.push(
649      context,
650      MaterialPageRoute(
651        builder: (context) => QRScannerScreen(),
652      ),
653    );
654  }
655
656  void _mostrarQR() {
657    Navigator.push(
658      context,
659      MaterialPageRoute(
660        builder: (context) => QRScannerScreen(),
661      ),
662    );
663  }
664
665  void _mostrarQR() {
666    Navigator.push(
667      context,
668      MaterialPageRoute(
669        builder: (context) => QRScannerScreen(),
670      ),
671    );
672  }
673
674  void _mostrarQR() {
675    Navigator.push(
676      context,
677      MaterialPageRoute(
678        builder: (context) => QRScannerScreen(),
679      ),
680    );
681  }
682
683  void _mostrarQR() {
684    Navigator.push(
685      context,
686      MaterialPageRoute(
687        builder: (context) => QRScannerScreen(),
688      ),
689    );
690  }
691
692  void _mostrarQR() {
693    Navigator.push(
694      context,
695      MaterialPageRoute(
696        builder: (context) => QRScannerScreen(),
697      ),
698    );
699  }
700
701  void _mostrarQR() {
702    Navigator.push(
703      context,
704      MaterialPageRoute(
705        builder: (context) => QRScannerScreen(),
706      ),
707    );
708  }
709
710  void _mostrarQR() {
711    Navigator.push(
712      context,
713      MaterialPageRoute(
714        builder: (context) => QRScannerScreen(),
715      ),
716    );
717  }
718
719  void _mostrarQR() {
720    Navigator.push(
721      context,
722      MaterialPageRoute(
723        builder: (context) => QRScannerScreen(),
724      ),
725    );
726  }
727
728  void _mostrarQR() {
729    Navigator.push(
730      context,
731      MaterialPageRoute(
732        builder: (context) => QRScannerScreen(),
733      ),
734    );
735  }
736
737  void _mostrarQR() {
738    Navigator.push(
739      context,
740      MaterialPageRoute(
741        builder: (context) => QRScannerScreen(),
742      ),
743    );
744  }
745
746  void _mostrarQR() {
747    Navigator.push(
748      context,
749      MaterialPageRoute(
750        builder: (context) => QRScannerScreen(),
751      ),
752    );
753  }
754
755  void _mostrarQR() {
756    Navigator.push(
757      context,
758      MaterialPageRoute(
759        builder: (context) => QRScannerScreen(),
760      ),
761    );
762  }
763
764  void _mostrarQR() {
765    Navigator.push(
766      context,
767      MaterialPageRoute(
768        builder: (context) => QRScannerScreen(),
769      ),
770    );
771  }
772
773  void _mostrarQR() {
774    Navigator.push(
775      context,
776      MaterialPageRoute(
777        builder: (context) => QRScannerScreen(),
778      ),
779    );
780  }
781
782  void _mostrarQR() {
783    Navigator.push(
784      context,
785      MaterialPageRoute(
786        builder: (context) => QRScannerScreen(),
787      ),
788    );
789  }
790
791  void _mostrarQR() {
792    Navigator.push(
793      context,
794      MaterialPageRoute(
795        builder: (context) => QRScannerScreen(),
796      ),
797    );
798  }
799
800  void _mostrarQR() {
801    Navigator.push(
802      context,
803      MaterialPageRoute(
804        builder: (context) => QRScannerScreen(),
805      ),
806    );
807  }
808
809  void _mostrarQR() {
810    Navigator.push(
811      context,
812      MaterialPageRoute(
813        builder: (context) => QRScannerScreen(),
814      ),
815    );
816  }
817
818  void _mostrarQR() {
819    Navigator.push(
820      context,
821      MaterialPageRoute(
822        builder: (context) => QRScannerScreen(),
823      ),
824    );
825  }
826
827  void _mostrarQR() {
828    Navigator.push(
829      context,
830      MaterialPageRoute(
831        builder: (context) => QRScannerScreen(),
832      ),
833    );
834  }
835
836  void _mostrarQR() {
837    Navigator.push(
838      context,
839      MaterialPageRoute(
840        builder: (context) => QRScannerScreen(),
841      ),
842    );
843  }
844
845  void _mostrarQR() {
846    Navigator.push(
847      context,
848      MaterialPageRoute(
849        builder: (context) => QRScannerScreen(),
850      ),
851    );
852  }
853
854  void _mostrarQR() {
855    Navigator.push(
856      context,
857      MaterialPageRoute(
858        builder: (context) => QRScannerScreen(),
859      ),
860    );
861  }
862
863  void _mostrarQR() {
864    Navigator.push(
865      context,
866      MaterialPageRoute(
867        builder: (context) => QRScannerScreen(),
868      ),
869    );
870  }
871
872  void _mostrarQR() {
873    Navigator.push(
874      context,
875      MaterialPageRoute(
876        builder: (context) => QRScannerScreen(),
877      ),
878    );
879  }
880
881  void _mostrarQR() {
882    Navigator.push(
883      context,
884      MaterialPageRoute(
885        builder: (context) => QRScannerScreen(),
886      ),
887    );
888  }
889
890  void _mostrarQR() {
891    Navigator.push(
892      context,
893      MaterialPageRoute(
894        builder: (context) => QRScannerScreen(),
895      ),
896    );
897  }
898
899  void _mostrarQR() {
900    Navigator.push(
901      context,
902      MaterialPageRoute(
903        builder: (context) => QRScannerScreen(),
904      ),
905    );
906  }
907
908  void _mostrarQR() {
909    Navigator.push(
910      context,
911      MaterialPageRoute(
912        builder: (context) => QRScannerScreen(),
913      ),
914    );
915  }
916
917  void _mostrarQR() {
918    Navigator.push(
919      context,
920      MaterialPageRoute(
921        builder: (context) => QRScannerScreen(),
922      ),
923    );
924  }
925
926  void _mostrarQR() {
927    Navigator.push(
928      context,
929      MaterialPageRoute(
930        builder: (context) => QRScannerScreen(),
931      ),
932    );
933  }
934
935  void _mostrarQR() {
936    Navigator.push(
937      context,
938      MaterialPageRoute(
939        builder: (context) => QRScannerScreen(),
940      ),
941    );
942  }
943
944  void _mostrarQR() {
945    Navigator.push(
946      context,
947      MaterialPageRoute(
948        builder: (context) => QRScannerScreen(),
949      ),
950    );
951  }
952
953  void _mostrarQR() {
954    Navigator.push(
955      context,
956      MaterialPageRoute(
957        builder: (context) => QRScannerScreen(),
958      ),
959    );
960  }
961
962  void _mostrarQR() {
963    Navigator.push(
964      context,
965      MaterialPageRoute(
966        builder: (context) => QRScannerScreen(),
967      ),
968    );
969  }
970
971  void _mostrarQR() {
972    Navigator.push(
973      context,
974      MaterialPageRoute(
975        builder: (context) => QRScannerScreen(),
976      ),
977    );
978  }
979
980  void _mostrarQR() {
981    Navigator.push(
982      context,
983      MaterialPageRoute(
984        builder: (context) => QRScannerScreen(),
985      ),
986    );
987  }
988
989  void _mostrarQR() {
990    Navigator.push(
991      context,
992      MaterialPageRoute(
993        builder: (context) => QRScannerScreen(),
994      ),
995    );
996  }
997
998  void _mostrarQR() {
999    Navigator.push(
1000      context,
1001      MaterialPageRoute(
1002        builder: (context) => QRScannerScreen(),
1003      ),
1004    );
1005  }
1006
1007  void _mostrarQR() {
1008    Navigator.push(
1009      context,
1010      MaterialPageRoute(
1011        builder: (context) => QRScannerScreen(),
1012      ),
1013    );
1014  }
1015
1016  void _mostrarQR() {
1017    Navigator.push(
1018      context,
1019      MaterialPageRoute(
1020        builder: (context) => QRScannerScreen(),
1021      ),
1022    );
1023  }
1024
1025  void _mostrarQR() {
1026    Navigator.push(
1027      context,
1028      MaterialPageRoute(
1029        builder: (context) => QRScannerScreen(),
1030      ),
1031    );
1032  }
1033
1034  void _mostrarQR() {
1035    Navigator.push(
1036      context,
1037      MaterialPageRoute(
1038        builder: (context) => QRScannerScreen(),
1039      ),
1040    );
1041  }
1042
1043  void _mostrarQR() {
1044    Navigator.push(
1045      context,
1046      MaterialPageRoute(
1047        builder: (context) => QRScannerScreen(),
1048      ),
1049    );
1050  }
1051
1052  void _mostrarQR() {
1053    Navigator.push(
1054      context,
1055      MaterialPageRoute(
1056        builder: (context) => QRScannerScreen(),
1057      ),
1058    );
1059  }
1060
1061  void _mostrarQR() {
1062    Navigator.push(
1063      context,
1064      MaterialPageRoute(
1065        builder: (context) => QRScannerScreen(),
1066      ),
1067    );
1068  }
1069
1070  void _mostrarQR() {
1071    Navigator.push(
1072      context,
1073      MaterialPageRoute(
1074        builder: (context) => QRScannerScreen(),
1075      ),
1076    );
1077  }
1078
1079  void _mostrarQR() {
1080    Navigator.push(
1081      context,
1082      MaterialPageRoute(
1083        builder: (context) => QRScannerScreen(),
1084      ),
1085    );
1086  }
1087
1088  void _mostrarQR() {
1089    Navigator.push(
1090      context,
1091      MaterialPageRoute(
1092        builder: (context) => QRScannerScreen(),
1093      ),
1094    );
1095  }
1096
1097  void _mostrarQR() {
1098    Navigator.push(
1099      context,
1100      MaterialPageRoute(
1101        builder: (context) => QRScannerScreen(),
1102      ),
1103    );
1104  }
1105
1106  void _mostrarQR() {
1107    Navigator.push(
1108      context,
1109      MaterialPageRoute(
1110        builder: (context) => QRScannerScreen(),
1111      ),
1112    );
1113  }
1114
1115  void _mostrarQR() {
1116    Navigator.push(
1117      context,
1118      MaterialPageRoute(
1119        builder: (context) => QRScannerScreen(),
1120      ),
1121    );
1122  }
1123
1124  void _mostrarQR() {
1125    Navigator.push(
1126      context,
1127      MaterialPageRoute(
1128        builder: (context) => QRScannerScreen(),
1129      ),
1130    );
1131  }
1132
1133  void _mostrarQR() {
1134    Navigator.push(
1135      context,
1136      MaterialPageRoute(
1137        builder: (context) => QRScannerScreen(),
1138      ),
1139    );
1140  }
1141
1142  void _mostrarQR() {
1143    Navigator.push(
1144      context,
1145      MaterialPageRoute(
1146        builder: (context) => QRScannerScreen(),
1147      ),
1148    );
1149  }
1150
1151  void _mostrarQR() {
1152    Navigator.push(
1153      context,
1154      MaterialPageRoute(
1155        builder: (context) => QRScannerScreen(),
1156      ),
1157    );
1158  }
1159
1160  void _mostrarQR() {
1161    Navigator.push(
1162      context,
1163      MaterialPageRoute(
1164        builder: (context) => QRScannerScreen(),
1165      ),
1166    );
1167  }
1168
1169  void _mostrarQR() {
1170    Navigator.push(
1171      context,
1172      MaterialPageRoute(
1173        builder: (context) => QRScannerScreen(),
1174      ),
1175    );
1176  }
1177
1178  void _mostrarQR() {
1179    Navigator.push(
1180      context,
1181      MaterialPageRoute(
1182        builder: (context) => QRScannerScreen(),
1183      ),
1184    );
1185  }
1186
1187  void _mostrarQR() {
1188    Navigator.push(
1189      context,
1190      MaterialPageRoute(
1191        builder: (context) => QRScannerScreen(),
1192      ),
1193    );
1194  }
1195
1196  void _mostrarQR() {
1197    Navigator.push(
1198      context,
1199      MaterialPageRoute(
1200        builder: (context) => QRScannerScreen(),
1201      ),
1202    );
1203  }
1204
1205  void _mostrarQR() {
1206    Navigator.push(
1207      context,
1208      MaterialPageRoute(
1209        builder: (context) => QRScannerScreen(),
1210      ),
1211    );
1212  }
1213
1214  void _mostrarQR() {
1215    Navigator.push(
1216      context,
1217      MaterialPageRoute(
1218        builder: (context) => QRScannerScreen(),
1219      ),
1220    );
1221  }
1222
1223  void _mostrarQR() {
1224    Navigator.push(
1225      context,
1226      MaterialPageRoute(
1227        builder: (context) => QRScannerScreen(),
1228      ),
1229    );
1230  }
1231
1232  void _mostrarQR() {
1233    Navigator.push(
1234      context,
1235      MaterialPageRoute(
1236        builder: (context) => QRScannerScreen(),
1237      ),
1238    );
1239  }
1240
1241  void _mostrarQR() {
1242    Navigator.push(
1243      context,
1244      MaterialPageRoute(
1245        builder: (context) => QRScannerScreen(),
1246      ),
1247    );
1248  }
1249
1250  void _mostrarQR() {
1251    Navigator.push(
1252      context,
1253      MaterialPageRoute(
1254        builder: (context) => QRScannerScreen(),
1255      ),
1256    );
1257  }
1258
1259  void _mostrarQR() {
1260    Navigator.push(
1261      context,
1262      MaterialPageRoute(
1263        builder: (context) => QRScannerScreen(),
1264      ),
1265    );
1266  }
1267
1268  void _mostrarQR() {
1269    Navigator.push(
1270      context,
1271      MaterialPageRoute(
1272        builder: (context) => QRScannerScreen(),
1273      ),
1274    );
1275  }
1276
1277  void _mostrarQR() {
1278    Navigator.push(
1279      context,
1280      MaterialPageRoute(
1281        builder: (context) => QRScannerScreen(),
1282      ),
1283    );
1284  }
1285
1286  void _mostrarQR() {
1287    Navigator.push(
1288      context,
1289      MaterialPageRoute(
1290        builder: (context) => QRScannerScreen(),
1291      ),
1292    );
1293  }
1294
1295  void _mostrarQR() {
1296    Navigator.push(
1297      context,
1298      MaterialPageRoute(
1299        builder: (context) => QRScannerScreen(),
1300      ),
1301    );
1302  }
1303
1304  void _mostrarQR() {
1305    Navigator.push(
1306      context,
1307      MaterialPageRoute(
1308        builder: (context) => QRScannerScreen(),
1309      ),
1310    );
1311  }
1312
1313  void _mostrarQR() {
1314    Navigator.push(
1315      context,
1316      MaterialPageRoute(
1317        builder: (context) => QRScannerScreen(),
1318      ),
1319    );
1320  }
1321
1322  void _mostrarQR() {
1323    Navigator.push(
1324      context,
1325      MaterialPageRoute(
1326        builder: (context) => QRScannerScreen(),
1327      ),
1328    );
1329  }
1330
1331  void _mostrarQR() {
1332    Navigator.push(
1333      context,
1334      MaterialPageRoute(
1335        builder: (context) => QRScannerScreen(),
1336      ),
1337    );
1338  }
1339
1340  void _mostrarQR() {
1341    Navigator.push(
1342      context,
1343      MaterialPageRoute(
1344        builder: (context) => QRScannerScreen(),
1345      ),
1346    );
1347  }
1348
1349  void _mostrarQR() {
1350    Navigator.push(
1351      context,
1352      MaterialPageRoute(
1353        builder: (context) => QRScannerScreen(),
1354      ),
1355    );
1356  }
1357
1358  void _mostrarQR() {
1359    Navigator.push(
1360      context,
1361      MaterialPageRoute(
1362        builder: (context) => QRScannerScreen(),
1363      ),
1364    );
1365  }
1366
1367  void _mostrarQR() {
1368    Navigator.push(
1369      context,
1370      MaterialPageRoute(
1371        builder: (context) => QRScannerScreen(),
1372      ),
1373    );
1374  }
1375
1376  void _mostrarQR() {
1377    Navigator.push(
1378      context,
1379      MaterialPageRoute(
1380        builder: (context) => QRScannerScreen(),
1381      ),
1382    );
1383  }
1384
1385  void _mostrarQR() {
1386    Navigator.push(
1387      context,
1388      MaterialPageRoute(
1389        builder: (context) => QRScannerScreen(),
1390      ),
1391    );
1392  }
1393
1394  void _mostrarQR() {
1395    Navigator.push(
1396      context,
1397      MaterialPageRoute(
1398        builder: (context) => QRScannerScreen(),
1399      ),
1400    );
1401  }
1402
1403  void _mostrarQR() {
1404    Navigator.push(
1405      context,
1406      MaterialPageRoute(
1407        builder: (context) => QRScannerScreen(),
1408      ),
1409    );
1410  }
1411
1412  void _mostrarQR() {
1413    Navigator.push(
1414      context,
1415      MaterialPageRoute(
1416        builder: (context) => QRScannerScreen(),
1417      ),
1418    );
1419  }
1420
1421  void _mostrarQR() {
1422    Navigator.push(
1423      context,
1424      MaterialPageRoute(
1425        builder: (context) => QRScannerScreen(),
1426      ),
1427    );
1428  }
1429
1430  void _mostrarQR() {
1431    Navigator.push(
1432      context,
1433      MaterialPageRoute(
1434        builder: (context) => QRScannerScreen(),
1435      ),
1436    );
1437  }
1438
1439  void _mostrarQR() {
1440    Navigator.push(
1441      context,
1442      MaterialPageRoute(
1443        builder: (context) => QRScannerScreen(),
1444      ),
1445    );
1446  }
1447
1448  void _mostrarQR() {
1449    Navigator.push(
1450      context,
1451      MaterialPageRoute(
1452        builder: (context) => QRScannerScreen(),
1453      ),
1454    );
1455  }
1456
1457  void _mostrarQR() {
1458    Navigator.push(
1459      context,
1460      MaterialPageRoute(
1461        builder: (context) => QRScannerScreen(),
1462      ),
1463    );
1464  }
1465
1466  void _mostrarQR() {
1467    Navigator.push(
1468      context,
1469      MaterialPageRoute(
1470        builder: (context) => QRScannerScreen(),
1471      ),
1472    );
1473  }
1474
1475  void _mostrarQR() {
1476    Navigator.push(
1477      context,
1478      MaterialPageRoute(
1479        builder: (context) => QRScannerScreen(),
1480      ),
1481    );
1482  }
1483
1484  void _mostrarQR() {
1485    Navigator.push(
1486      context,
1487      MaterialPageRoute(
1488        builder: (context) => QRScannerScreen(),
1489      ),
1490    );
1491  }
1492
1493  void _mostrar
```

```
class Clase extends StatefulWidget {
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(
                        mainAxisAlignment: MainAxisAlignment.min,
                        children: [
                            // 5.4) Mostrar el código leido.
                            Text(
                                'El código: $numeros',
                                style: const TextStyle(fontSize: 15),
                            ), // Text
                            const SizedBox(height: 12),
                            // 5.5) Campo nombre (texto normal).
                            TextField(
                                controller: nombre,

```

① Flutter: Running Gradle task 'assembleDebug'...

```
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();

```

① Flutter: Running Gradle task 'assembleDebug'...

```
    _mostrarDatos(String numeros) async {
      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'),
              ), // SnackBar
            );
            return;
          }
        },
      );
    }
  }
}
```

```

class Clase extends StatefulWidget {
    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.
}

```

Flutter Running Gradle

```

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;
});
}

```

```
class Clase extends StatefulWidget {
  @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

```
lib > basedatos.dart > Basedatoshelper.dart > Basedatoshelper > _initDB
1 import 'package:path/path.dart';
2 import 'package:sqflite/sqflite.dart';
3
4 class Basedatoshelper {
5     // 1) Singleton: misma instancia en toda la app.
6     static final Basedatoshelper _instance = Basedatoshelper._internal();
7     factory Basedatoshelper() => _instance;
8     Basedatoshelper._internal();
9
10    // 2) Referencia a la BD (perezosa).
11    Database? _database;
12
13    // 3) Getter: abre o devuelve la BD existente.
14    Future<Database> get database async {
15        if (_database != null) return _database!;
16        _database = await _initDB();
17        return _database!;
18    }
19
20    // 4) Inicialización: crear/abrir archivo 'abarrotes.db'.
21    Future<Database> _initDB() async {
22        final dbPath = await getDatabasesPath();
23        final path = join(dbPath, 'abarrotes.db');
24
25        // 4.1) Abrir BD, crear tabla si versión 1.
26        return await openDatabase(
27            path,
28            version: 1,
29            onCreate: (db, version) async {
30                // 4.2) Esquema corregido: clave primaria autoincrem
31                await db.execute('''
32                    CREATE TABLE products(
33                        id INTEGER PRIMARY KEY AUTOINCREMENT,
34                        name TEXT NOT NULL,
35                        quantity INTEGER NOT NULL,
36                        price REAL NOT NULL
37                    );
38                ''');
39            }
40        );
41    }
42}
```

```
basedatos.dart > basedatosniciper.dart > basedatoshelper > _initDB
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;
    return await db.query('productos', orderBy: 'id DESC');
}
```

```
// 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