

APLICACIONES MOVILES MULTIPLATAFORMA

LABORATORIO N° 13

Uso de modales – TextField – Plugin Shared Preferences



Alumno(s):		Nota	
Grupo:		Ciclo:V	

Laboratorio 13: Uso de modales – TextField – Shared Preferences

Objetivos:

Al finalizar el laboratorio el estudiante será capaz de:

- Utilizar modales en un proyecto Flutter
- Interactuar con TextField
- Utilizar plugins en Flutter

Seguridad:

- Ubicar maletines y/o mochilas en el gabinete del aula de Laboratorio.
- No ingresar con líquidos, ni comida al aula de Laboratorio.
- Al culminar la sesión de laboratorio apagar correctamente la computadora y la pantalla, y ordenar las sillas utilizadas.

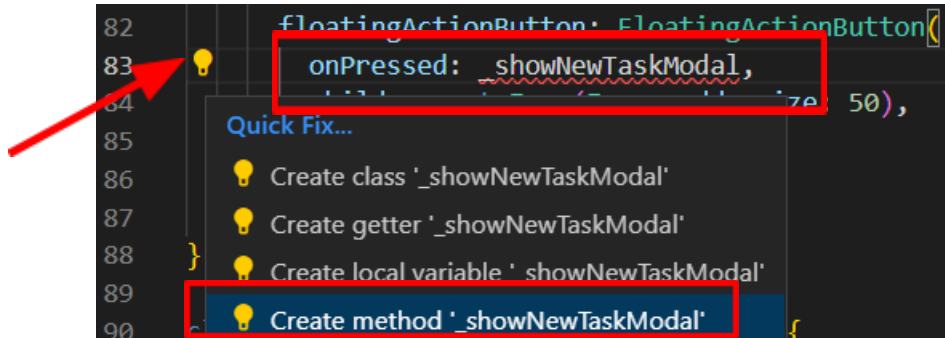
Equipos y Materiales:

- Una computadora con:
 - Windows 7 o superior
 - Conexión a la red del laboratorio
- Máquinas Virtuales
 - Windows 7 Pro 64bits Español – Plantilla

Procedimiento:

1. Agregando modales al proyecto:

- 1.1. En base al proyecto base del laboratorio anterior, vamos a modificar el contenido del archivo 'task_list_page.dart', para que muestre un modal al momento de presionar el botón principal:



- 1.2. Dentro del método generado indicamos lo siguiente:

```
89 void _showNewTaskModal() {
90   showModalBottomSheet(context: context, builder: builder)
91 }
```

```
76 void _showNewTaskModal() {
77   showModalBottomSheet(context: context, builder: builder)
78 }
79 }
```

- 1.3. Generamos una función anónima que internamente ejecuta la función que queremos ejecutar:

```
89 void _showNewTaskModal(BuildContext context) {
90   showModalBottomSheet(
91     context: context, builder: (_) => const _NewTaskModal();
92   )
93 }
```

```
95 class _NewTaskModal extends StatelessWidget {
96   const _NewTaskModal({super.key});
97
98   @override
99   Widget build(BuildContext context) {
100     return Container(
101       child: Column(
102         children: [
103           H1('Nueva tarea'),
104           TextField(),
105           ElevatedButton(
106             onPressed: () {},
107             child: Text('Guardar'),
108           ) // ElevatedButton
109         ],
110       ) // Column
111     ); // Container
112   }
113 }
```

```

5
6 void _showNewTaskModal(BuildContext context) {
7   showModalBottomSheet(
8     context: context, builder: (_) => const _NewTaskModal());
9   }
10 }
11
12 class _NewTaskModal extends StatelessWidget {
13   const _NewTaskModal({super.key}); A value for optional parameter 'key'
14
15   @override
16   Widget build(BuildContext context) {
17     return Container(
18       child: Column(
19         children: [
20           H1('Nueva tarea'), Use 'const' with the constructor to improve
21           TextField(), Use 'const' with the constructor to improve perfo
22           ElevatedButton(
23             onPressed: () {},
24             child: Text('Guardar'), Use 'const' with the constructor to
25           ) // ElevatedButton
26         ],
27       ), // Column
28     ); // Container
29   }
30 }

```

- 1.4. Queremos que el modal se adapte al tamaño de los objetos que tiene dentro. Para ello, aplicamos lo siguiente:

```

89 void _showNewTaskModal(BuildContext context) {
90   showModalBottomSheet(
91     context: context,
92     isScrollControlled: true,
93     builder: (_) => const _NewTaskModal());
94   }
95 }
96
97 class _NewTaskModal extends StatelessWidget {
98   const _NewTaskModal({super.key});
99
100   @override
101   Widget build(BuildContext context) {
102     return Container(
103       child: Column(
104         mainAxisAlignment: MainAxisAlignment.min,

```

```
class _NewTaskModal extends StatelessWidget {
  const _NewTaskModal({super.key}); // A value for c

  @override
  Widget build(BuildContext context) {
    return Container( // Unnecessary instance of 'Co
      child: Column(
        mainAxisAlignment: MainAxisAlignment.min,
        children: [
          H1('Nueva tarea'), // Use 'const' with the
          TextField(), // Use 'const' with the const
          ElevatedButton(
            onPressed: () {},
            child: Text('Guardar'), // Use 'const' w
          ) // ElevatedButton
        ],
      ), // Column
    );
  }
}
```

1.5. Generamos espacio en el aplicativo de acuerdo a lo siguiente:

```
100  @override
101  Widget build(BuildContext context) {
102    return Container(
103      padding: const EdgeInsets.symmetric(
104        horizontal: 33,
105        vertical: 23
106      ), // EdgeInsets.symmetric
107      child: Column(
108        crossAxisAlignment: CrossAxisAlignment.start,
109        mainAxisAlignment: MainAxisAlignment.min,
110        children: [
111          H1('Nueva tarea'),
112          const SizedBox(height: 26),
113          TextField(),
114          const SizedBox(height: 26),
115          ElevatedButton(
116            onPressed: () {},
117            child: Text('Guardar'),
118          ) // ElevatedButton
119        ],
120      ),
121    );
122  }
```

1.6. Cambiamos el borde del contenedor a redondeado. Para tal, realizamos lo siguiente.

```
101  Widget build(BuildContext context) {
102    return Container(
103      padding: const EdgeInsets.symmetric(
104        horizontal: 33,
105        vertical: 23
106      ), // EdgeInsets.symmetric
107      decoration: const BoxDecoration(
108        borderRadius: BorderRadius.vertical(top: Radius.circular(21)),
109        color: Colors.white,
110      ), // BoxDecoration
111    );
112  }
```

1.7. Nos dirigimos al archivo 'app.dart' y modificamos Theme de la siguiente manera:

```

20      textTheme: Theme.of(context).textTheme.apply(
21        fontFamily: 'Poppins',
22        bodyColor: textColor,
23        displayColor: textColor,
24      ),
25      bottomSheetTheme: const BottomSheetThemeData(
26        backgroundColor: Colors.transparent,
27      ), // BottomSheetThemeData
28      useMaterial3: true,

```

```

23      ),
24      bottomSheetTheme: const BottomSheetThemeData(
25        backgroundColor: Colors.transparent,
26      ), // BottomSheetThemeData
27      useMaterial3: true

```

1.8. Modificamos también el Textfield de la siguiente manera (en el archivo **'task_list_page.dart'**):

```

115      H1('Nueva tarea'),
116      const SizedBox(height: 26),
117      TextField(
118        decoration: InputDecoration(
119          filled: true,
120          fillColor: Colors.white,
121          border: OutlineInputBorder(
122            borderRadius: BorderRadius.circular(16)
123          ), // OutlineInputBorder
124          hintText: 'Descripción de la tarea'
125        ), // InputDecoration
126      ), // TextField

```

```

103      const SizedBox(height: 26),
104      TextField(
105        decoration: InputDecoration(
106          filled: true,
107          fillColor: Colors.white,
108          border: OutlineInputBorder(
109            borderRadius: BorderRadius.circular(16)
110          ), // OutlineInputBorder
111          hintText: 'Descripción de la tarea'
112        ), // InputDecoration
113      ), // TextField

```

1.9. Modificamos también el estilo del botón de la siguiente manera, de acuerdo al Theme, para que aplique a todos los botones. Nos ubicamos dentro del archivo **'app.dart'** nuevamente y modificamos lo siguiente:

```

25     bottomSheetTheme: const BottomSheetThemeData(
26       backgroundColor: Colors.transparent,
27     ), // BottomSheetThemeData
28     elevatedButtonTheme: ElevatedButtonThemeData(
29       style: ElevatedButton.styleFrom(
30         minimumSize: const Size(
31           double.infinity,
32           54,), // Size
33         shape: RoundedRectangleBorder(
34           borderRadius: BorderRadius.circular(10)
35         ), // RoundedRectangleBorder
36         textStyle: Theme.of(context).textTheme.bodyMedium!.copyWith(
37           fontSize: 18,
38           fontWeight: FontWeight.w700,
39         )
40       )
41     ), // ElevatedButtonThemeData

```

```

27     elevatedButtonTheme: ElevatedButtonThemeData(
28       style: ElevatedButton.styleFrom(
29         minimumSize: const Size(
30           double.infinity,
31           54,), // Size
32         shape: RoundedRectangleBorder(
33           borderRadius: BorderRadius.circular(10)
34         ), // RoundedRectangleBorder
35         textStyle: Theme.of(context).textTheme.bodyMedium!.copyWith(
36           fontSize: 18,
37           fontWeight: FontWeight.w700,
38         )
39       )
40     ), // ElevatedButtonThemeData

```

1.10. Guardamos los cambios, comprobamos y mostramos los resultados obtenidos a continuación.



2. Interacción con Textfield:

- 2.1. A continuación, queremos ingresar un mensaje y que se pueda registrar en el aplicativo. Para ello, realizamos las siguientes modificaciones en el archivo **'task_list_page.dart'**:

```

89   void _showNewTaskModal(BuildContext context) {
90     showModalBottomSheet(
91       context: context,
92       isScrollControlled: true,
93       builder: ( ) => NewTaskModal());
94   }
95 }
96
97 class _NewTaskModal extends StatelessWidget {
98   NewTaskModal({super.key});
99
100   final _controller = TextEditingController();
101
102   @override

```

```

75
76   void _showNewTaskModal() {
77     showModalBottomSheet(
78       context: context,
79       isScrollControlled: true,
80       builder: ( ) => _NewTaskModal(), Use 'const'
81     );
82   }
83 }
84
85 class _NewTaskModal extends StatelessWidget {
86   const _NewTaskModal({Key? key}); A value for opti
87
88   @override
89   Widget build(BuildContext context) {
90     return Container(
91       padding: const EdgeInsets.symmetric(
92         horizontal: 22,

```

```

118       const SizedBox(height: 26),
119       TextField(
120         controller: _controller,
121         decoration: InputDecoration(
122           filled: true,

```

```

131     ElevatedButton(
132       onPressed: ( ) {
133         if (_controller.text.isNotEmpty) {
134           final task = Task(_controller.text);
135           Navigator.of(context).pop();
136         }
137       },
138       child: Text('Guardar'),
139     ) // ElevatedButton

```

- 2.2. Se puede hacer un callback para informar al listado que hay una nueva tarea. Para ello, modificamos lo siguiente


```

97 class _NewTaskModal extends StatelessWidget {
98   NewTaskModal({super.key});
99
100   final _controller = TextEditingController();
101   final void Function(Task task) onTaskCreated;

```

```

97 class _NewTaskModal extends StatelessWidget {
98   NewTaskModal({super.key});
99
100   Quick Fix...
101   Add final field formal parameters

```

```

89 void _showNewTaskModal(BuildContext context) {
90   showModalBottomSheet(
91     context: context,
92     isScrollControlled: true,
93     builder: (_) => _NewTaskModal(onTaskCreated: (Task task) { },)),
94   )
95 }
96
97 class _NewTaskModal extends StatelessWidget {
98   _NewTaskModal({super.key, required this.onTaskCreated});

```

- 2.3. Luego, antes de cerrar el widget del botón, se quiere llamar al callback generado, modificando lo siguiente:

```

133 const SizedBox(height: 26),
134 ElevatedButton(
135   onPressed: () {
136     if (_controller.text.isNotEmpty) {
137       final task = Task(_controller.text);
138       onTaskCreated(task);
139       Navigator.of(context).pop();
140     }
141   },

```

- 2.4. A continuación, hacemos la gestión de estado para que se añada la nueva tarea en la lista de tareas, de acuerdo a lo siguiente:

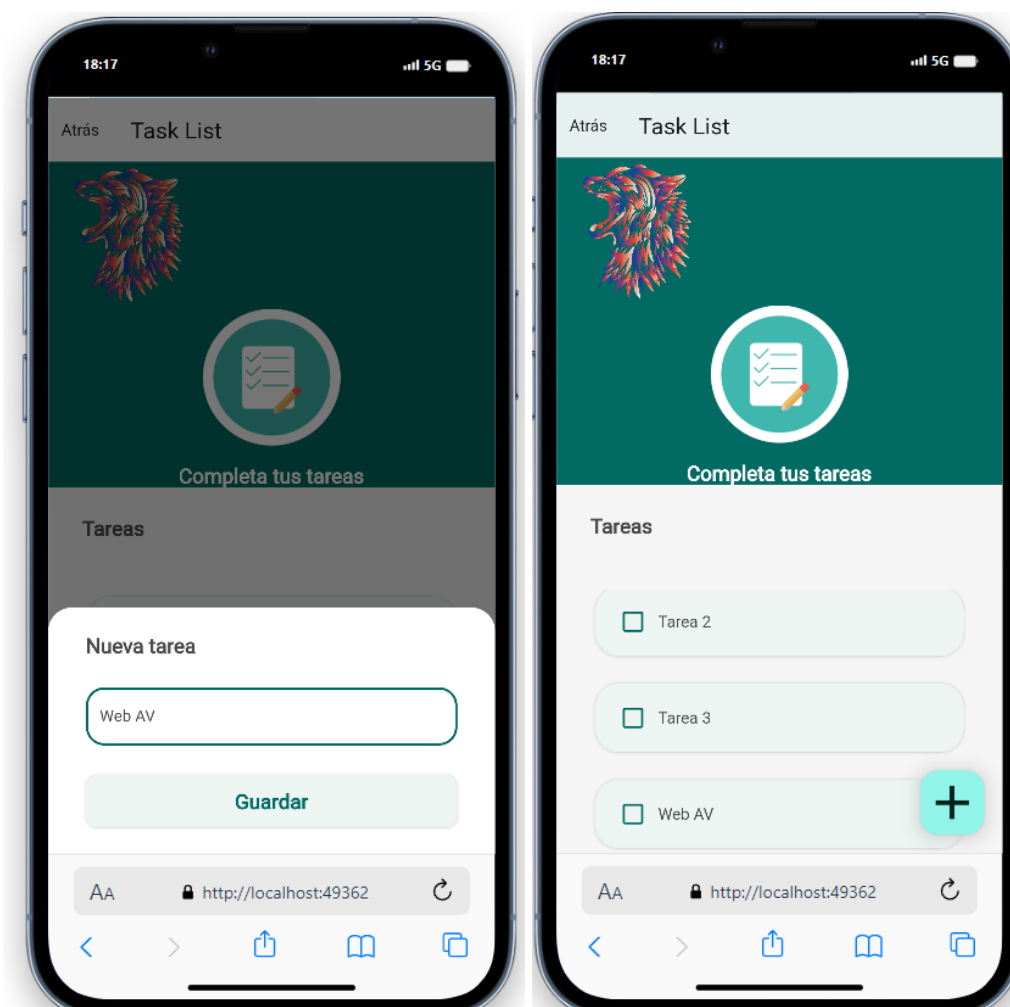
```

89 void _showNewTaskModal(BuildContext context) {
90   showModalBottomSheet(
91     context: context,
92     isScrollControlled: true,
93     builder: (_) => NewTaskModal(onTaskCreated: (Task task) {
94       setState(() {
95         taskList.add(task);
96       });
97     },));

```

- 2.5. Guardamos los cambios, ejecutamos el proyecto, comprobamos y adjuntamos los resultados obtenidos. Realizamos comentarios diversos sobre el funcionamiento del aplicativo hasta el momento.

- 2.6. Eliminamos la lista de tareas que se genera mediante código, para que pueda ingresarse mediante el modal generado. Adjuntamos resultados obtenidos en la ejecución del aplicativo.



3. Uso de plugin Shared Preferences:

- 3.1. Ingresamos a la página <https://pub.dev/>, para tener alcance de los paquetes y plugins de Flutter.
- 3.2. Para empezar, un package es una librería externa que nos provee código Dart (p. ej. un conjunto de widgets), la cual podemos incluir en nuestro proyecto y utilizar ello. Un plugin es una librería que nos permite acceder a la parte nativa de la plataforma a la cual vamos a desarrollar (p. ej. un SDK de mapas de Android, iOS o Mac). En nuestro caso vamos a utilizar un plugin llamado Shared Preferences.
- 3.3. Buscamos este plugin mediante la página compartida. Ingresamos a la documentación del siguiente plugin:

shared_preferences

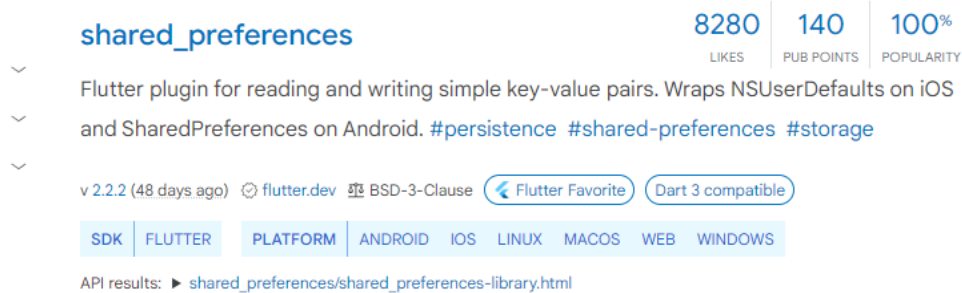
7569 140 100%
 LIKES PUB POINTS POPULARITY

Flutter plugin for reading and writing simple key-value pairs. Wraps UserDefaults on iOS and SharedPreferences on Android.

v 2.2.0 (4 days ago) flutter.dev BSD-3-Clause Flutter Favorite Dart 3 compatible

SDK FLUTTER PLATFORM ANDROID IOS LINUX MACOS WEB WINDOWS

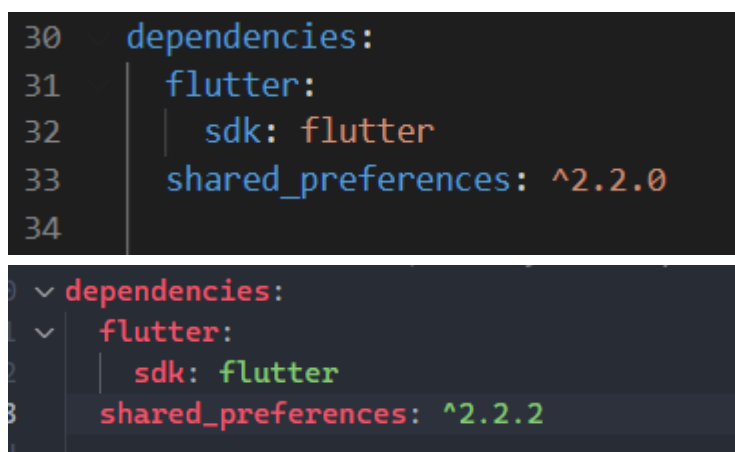
API results: ▶ shared_preferences/shared_preferences-library.html



- 3.4. Hacemos clic sobre su nombre para tener alcance de su documentación. A continuación, alcancemos los puntos principales de la documentación.
- 3.5. De acuerdo a la documentación, nos ubicamos a la altura del título hacemos clic en el icono del costado derecho, para copiar el nombre del plugin:

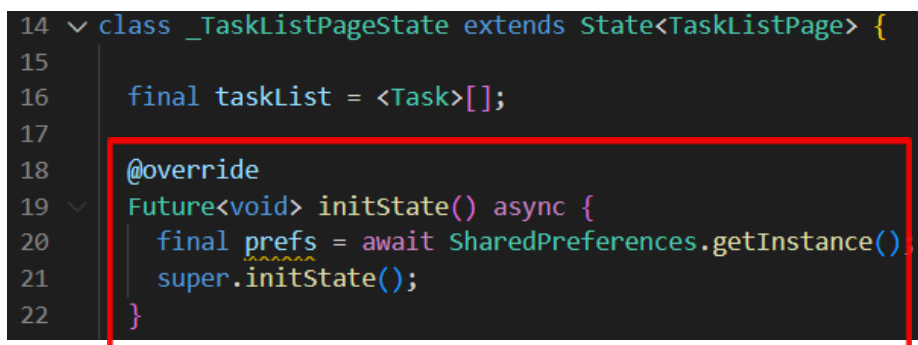


- 3.6. Luego, regresamos a nuestro proyecto en VS Code y abrimos el archivo 'pubsec.yaml'. Modificamos según lo siguiente:



Nota: Colocar el nombre del plugin al mismo nivel de "flutter".


- 3.7. Guardar los cambios realizados. Debe empezar una instalación mediante 'flutter pub get'. Esperar a que culmine el proceso para continuar.
- 3.8. Luego, nos dirigimos al archivo 'task_list_page.dart' y modificamos según lo siguiente:



- 3.9. Vamos a continuar guardando nuestro listado de tareas mediante el plugin instalado. Para ello, seguimos los siguientes pasos:

3.9.1. Notemos que ya se ha añadido el plugin al comienzo del archivo:

```
4 import 'package:flutter/material.dart';  
5 import 'package:shared_preferences/shared_preferences.dart';
```



3.9.2. Modificamos lo siguiente para guardar el listado en el plugin. El plugin tiene diversos métodos Set (SetString, SetStringList, SetBool, etc.) Para ello, no se puede guardar un objeto Task directamente ya que debemos **serializar** primero ese objeto. Esto significa que vamos a convertir ese objeto a un array de bytes. En nuestro caso vamos a convertir el objeto Task a una cadena de texto. La forma más práctica de hacer ello es pasar el objeto a una estructura JSON, para luego transformar el mapa que se indique para que se obtenga una cadena de texto. Para ello, aplicamos lo siguiente:

```
94 void _showNewTaskModal(BuildContext context) {  
95   showModalBottomSheet(  
96     context: context,  
97     isScrollControlled: true,  
98     builder: (_) => _NewTaskModal(onTaskCreated: (Task task) {  
99       setState(() async {  
100         taskList.add(task);  
101         final prefs = await SharedPreferences.getInstance();  
102         final map = {  
103           'title': task.title,  
104           'done': task.done  
105         };  
106         prefs.setStringList('tasks', [  
107           jsonEncode(map),  
108         ]);  
109       });  
110     });  
111 }
```

3.9.3. Vamos a crear dos métodos dentro de la clase 'Task', para convertir el objeto Task a una estructura de tipo mapa, que es lo requerido para la serialización mediante JSON.

```
lib > app > model > task.dart > ...  
1 class Task {  
2  
3   Task(this.title, {this.done = false});  
4  
5   final String title;  
6   bool done;  
7  
8   Map<String, dynamic> toJson() {  
9     return {  
10       'title': title,  
11       'done': done  
12     };  
13   }  
14  
15 }
```

```

task_list_page.dart 1 task.dart x app.c
lib > app > view > model > task.dart > Task > toJson
1 class Task {
2   Task(this.title, {this.done = false});
3   final String title;
4   bool done;
5
6   Map<String, dynamic> toJson() {
7     return {'title': title, 'done': done};
8   }
9 }
10

```

3.9.4. Modificamos el contenido del archivo **'task_list_page.dart'**:

```

94 void _showNewTaskModal(BuildContext context) {
95   showModalBottomSheet(
96     context: context,
97     isScrollControlled: true,
98     builder: (_) => _NewTaskModal(onTaskCreated: (Task task) {
99       setState(() async {
100         taskList.add(task);
101         final prefs = await SharedPreferences.getInstance();
102         prefs.setStringList('tasks', [
103           jsonEncode(task.toJson()),
104         ]);
105       });
106     }
107   );
108 }

```

```

86 void _showNewTaskModal() {
87   showModalBottomSheet(
88     context: context,
89     isScrollControlled: true,
90     builder: (_) => _NewTaskModal(
91       onTaskCreated: (Task task) {
92         setState(() async {
93           taskList.add(task);
94           final prefs = await SharedPreferences.getInstance();
95           prefs.setStringList('tasks', [
96             jsonEncode(task.toJson()),
97           ]);
98         });
99       },
100     ),
101   );
102 }
103 }

```

3.9.5. Pero no queremos que se realice este proceso por cada objeto, sino de acuerdo a la lista de tareas. Siendo así, utilizamos la función **'map'**, mediante la cual se convierte un listado de objetos de **Task** a un objeto de cadenas de texto, el cual se va a serializar. Para ello, aplicamos lo siguiente:

```

98 builder: (_) => _NewTaskModal(onTaskCreated: (Task task) {
99   setState(() async {
100     taskList.add(task);
101     final prefs = await SharedPreferences.getInstance();
102     prefs.setStringList(
103       'tasks', taskList.map((e) => jsonEncode(e.toJson())).toList());
104   });
105 }

```

```

1  void _showNewTaskModal() {
2    showModalBottomSheet(
3      context: context,
4      isScrollControlled: true,
5      builder: (_) => _NewTaskModal(
6        onTaskCreated: (Task task) {
7          setState(() async {
8            taskList.add(task);
9            final prefs = await SharedPreferences.getInstance();
10           prefs.setStringList(
11             'tasks', taskList.map((e) => jsonEncode(e.toJson())).toList());
12           });
13        },
14      ),
15    );
16  }
17 }
18

```

3.9.6. Para la deserialización se debe generar un constructor en la clase **'Task'**, como se define a continuación:

```

lib > app > model > task.dart > Task
1  class Task {
2
3    Task(this.title, {this.done = false});
4
5    Task.fromJson(Map<String, dynamic> json){
6      title = json['title'];
7      done = json[done];
8    }
9    late final String title;
10   late bool done;
11

```

```

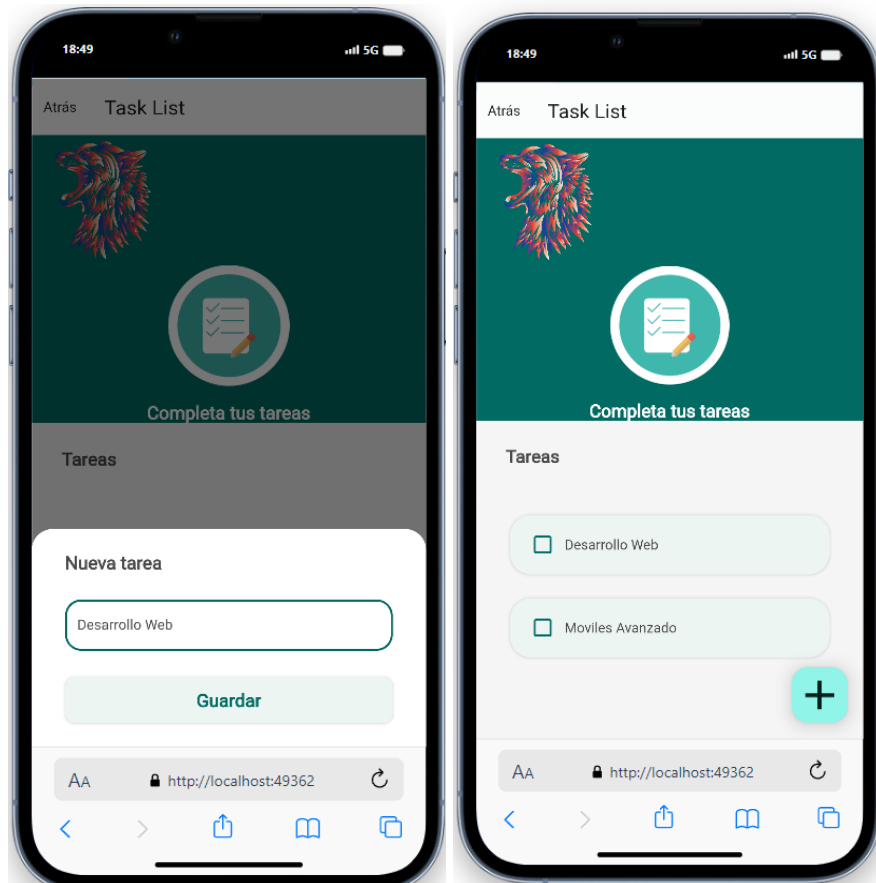
1  class Task {
2    Task(this.title, {this.done = false});
3
4    Task.fromJson(Map<String, dynamic> json) {
5      title = json['title'];
6      done = json[done];
7    }
8    late final String title;
9    late bool done;
10
11   Map<String, dynamic> toJson() {
12     return {'title': title, 'done': done};
13   }
14 }

```

3.9.7. Finalmente, modificamos el archivo **'task_list_page.dart'**

```
102     prefs.setStringList(  
103         'tasks', taskList.map((e) => jsonEncode(e.toJson())).toList());  
104  
105     final taskStrings = prefs.getStringList('tasks');  
106     final newTaskList = taskStrings?.map((e) => Task(jsonDecode(e))).toList();  
107  
108     });
```

3.9.8. Guardamos los cambios, comprobamos y adjuntamos los resultados obtenidos. Realizamos comentarios sobre el funcionamiento del aplicativo.



Tarea:

1. De acuerdo a la página: <https://pub.dev/> (o las referencias que se vea por conveniente), generar un nuevo proyecto Flutter que considere el uso de un plugin, diferente al visto en el ejemplo anterior. Adjuntar evidencias de desarrollo y resultados obtenidos.

```
1 import 'package:flutter/material.dart';
2 import 'package:mi_tarea04_flutter/app/view/gallery.dart';
3 import 'package:lottie/lottie.dart';
4 import 'package:animations/animations.dart';
5
6 class SplashPage extends StatelessWidget {
7   @override
8   Widget build(BuildContext context) {
9     return Scaffold(
10      body: Center(
11        child: Column(
12          mainAxisAlignment: MainAxisAlignment.center,
13          children: [
14            FadeScaleTransition(
15              animation: AlwaysStoppedAnimation(
16                1), // Puedes ajustar el valor según tus necesidades
17              child: Lottie.asset(
18                'assets/animations/hello.json',
19                width: 300,
20                height: 300,
21                repeat: true,
22              ),
23            ),
24            SizedBox(height: 20),
25            FadeScaleTransition(
26              animation: AlwaysStoppedAnimation(1),
27              child: Text(
28                '¡Bienvenido!',
29                style: TextStyle(
30                  fontFamily: 'CascadiaCode',
31                  fontSize: 40,
32                ),
33            ),
34            ),
35            SizedBox(height: 50),
36            ElevatedButton(
37              onPressed: () {
38                // Redirección a la página Galería
39                Navigator.of(context).pushReplacement(
40                  MaterialPageRoute(
41                    builder: (context) => GaleriaPage(),
42                  ),
43            );
44          },
45          style: ElevatedButton.styleFrom(
46            primary: Colors.blue,
47            onPrimary: Colors.white,
48            elevation: 5,
49            padding: EdgeInsets.symmetric(horizontal: 20, vertical: 10),
50            shape: RoundedRectangleBorder(
51              borderRadius: BorderRadius.circular(10),
52            ),
53          ),
54          child: Text(
55            'Ir a Galería',
56            style: TextStyle(
57              fontFamily: 'CascadiaCode',
58              fontSize: 20,
59            ),
60          ),
61        ),
62      ],
63    ),
64  );
65 }
66
67 }
```



```
1 import 'package:flutter/material.dart';
2 import 'package:mi_tarea04_flutter/app/view/splash.dart';
3 import 'package:share/share.dart';
4 import 'package:animations/animations.dart';
5
6 class GaleriaPage extends StatelessWidget {
7   final Map<String, int> likeCount = {
8     'Arboleda': 0,
9     'Bosque': 0,
10    'Naturaleza': 0,
11    'Riachuelo': 0,
12  };
13
14   final List<String> imageNames = [
15     'Arboleda',
16     'Bosque',
17     'Naturaleza',
18     'Riachuelo',
19     'Arboleda',
20     'Bosque',
21     'Naturaleza',
22     'Riachuelo',
23   ];
```

```
1 @override
2   Widget build(BuildContext context) {
3     return Scaffold(
4       appBar: AppBar(
5         title: Text('Galería'),
6         leading: IconButton(
7           icon: Icon(Icons.arrow_back),
8           onPressed: () {
9             Navigator.of(context).pushReplacement(
10               MaterialPageRoute(
11                 builder: (context) => SplashPage(),
12               ),
13             );
14           },
15         ),
16       ),
17       body: GridView.builder(
18         gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
19           crossAxisCount: 2,
20           crossAxisSpacing: 8.0,
21           mainAxisSpacing: 8.0,
22         ),
23         itemCount: imageNames.length,
24         itemBuilder: (context, index) {
25           return GestureDetector(
26             onTap: () {
27               Navigator.of(context).push(
28                 MaterialPageRoute(
29                   builder: (context) => DetallesImagenPage(
30                     imageName: imageNames[index],
31                     likeCount: likeCount[imageNames[index]] ?? 0,
32                   ),
33                 ),
34             );
35           },
36         ),
37       ),
38     );
39   }
```

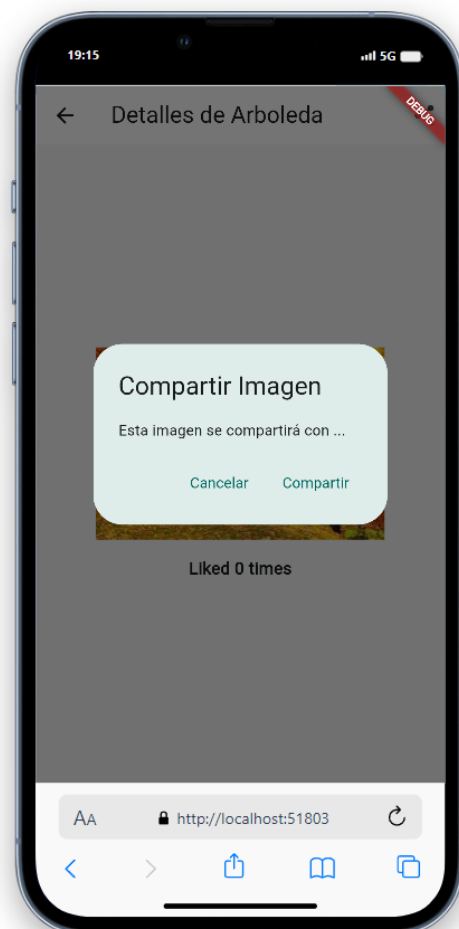


```

1  class DetallesImagenPage extends StatelessWidget {
2    final String imageName;
3    final int likeCount;
4
5    DetallesImagenPage({required this.imageName, required this.likeCount});
6
7    void _showShareDialog(BuildContext context) {
8      showDialog(
9        context: context,
10       builder: (BuildContext context) {
11         return AlertDialog(
12           title: Text('Compartir Imagen'),
13           content: Text('Esta imagen se compartirá con ...'),
14           actions: [
15             TextButton(
16               onPressed: () {
17                 Navigator.of(context).pop();
18               },
19               child: Text('Cancelar'),
20             ),
21             TextButton(
22               onPressed: () {
23                 Share.share('Check out this awesome image: $imageName');
24                 Navigator.of(context).pop();
25               },
26               child: Text('Compartir'),
27             ),
28           ],
29         );
30       },
31     );
32   }
33
34   @override
35   Widget build(BuildContext context) {
36     return Scaffold(
37       appBar: AppBar(
38         title: Text('Detalles de $imageName'),
39         leading: IconButton(
40           icon: Icon(Icons.arrow_back),
41           onPressed: () {
42             Navigator.of(context).pop();
43           },
44         ),
45       actions: [
46         IconButton(
47           icon: Icon(Icons.share),
48           onPressed: () {
49             _showShareDialog(context);
50           },
51         ),
52       ],
53     ),
54     body: Center(
55       child: Column(
56         mainAxisAlignment: MainAxisAlignment.center,
57         children: [
58           Hero(
59             tag: 'imageHero$imageName',
60             child: FittedBox(
61               fit: BoxFit.cover,
62               child: Image.asset(
63                 'assets/images/$imageName.png',
64               ),
65             ),
66           ),
67           SizedBox(height: 16.0),
68           Text(
69             'Liked $likeCount times',
70             style: TextStyle(
71               fontWeight: FontWeight.bold,
72               fontSize: 16.0,
73             ),
74           ),
75         ],
76       ),
77     );
78   }
79 }
80 }

```

```
splash.dart M gallery.dart M pubspec.yaml M X
pubspec.yaml
15 # Read more about iOS versioning at
16 # https://developer.apple.com/library/archive
17 # In Windows, build-name is used as the major
18 # of the product and file versions while build
19 version: 1.0.0+1
20
21 environment:
22   sdk: '≥3.1.4 <4.0.0'
23
24 # Dependencies specify other packages that your
25 # To automatically upgrade your package depend
26 # consider running 'flutter pub upgrade --major
27 # dependencies can be manually updated by cha
28 # the latest version available on pub.dev. To
29 # versions available, run 'flutter pub outdat
30 dependencies:
31   flutter:
32     sdk: flutter
33   flutter_staggered_grid_view: ^0.4.0
34   animations: ^2.0.8
35   share: ^2.0.4
36
```



PLUGINS UTILIZADOS	
animations: ^2.0.8	share: ^2.0.4
<ul style="list-style-type: none"> ● Propósito: Mejorar la estética y la experiencia visual de la aplicación. ● Características Clave: Facilitar la implementación de transiciones suaves, es altamente personalizable y añade un toque profesional al diseño. 	<ul style="list-style-type: none"> ● Propósito: Facilitar el intercambio de contenido desde la aplicación. ● Características Clave: Simplificar la implementación de funciones de compartir contenido, se integra con las capacidades de compartir nativas y fomenta la interacción y el compromiso del usuario.

Conclusiones:

- La incorporación de plugins en Flutter es una práctica común y efectiva para extender las funcionalidades de la aplicación. En este caso, utilizamos animations para mejorar la experiencia visual con transiciones suaves y share para facilitar el intercambio de contenido.
- La implementación de animaciones en la interfaz de usuario mediante el uso de animations contribuye significativamente a un diseño atractivo y moderno. Las transiciones suaves mejoran la estética general de la aplicación.
- La inclusión de funciones como compartir contenido mediante el paquete share mejora la interactividad de la aplicación. Permitir a los usuarios compartir fácilmente contenido fomenta la participación y la difusión de la aplicación.
- Las transiciones animadas y las funciones de compartir, contribuye a una experiencia del usuario optimizada. Estos elementos no solo son visualmente atractivos, sino que también mejoran la usabilidad y la satisfacción del usuario.
- La comunidad de Flutter proporciona una variedad de paquetes que simplifican tareas comunes y mejoran la eficiencia del desarrollo, permitiendo a los desarrolladores centrarse en la creación de experiencias de usuario excepcionales.