

Taller Flutter básico

Nombre y apellidos:	José Teodosio Lorente Vallecillos
Rol (director de proyecto/arquitecto/experto/programador):	Director de proyecto

Objetivos

- Conocer cómo desarrollar aplicaciones Flutter usando el IDE Android Studio o Visual Studio
- Aprender las características básicas del framework Flutter para crear interfaces de usuario multiplataforma:
 - Widgets sin estado: MaterialApp, ThemeData, Text, Center, Column, Row, Scaffold, FloatingActionButton
 - Widgets con estado: StatefulWidget
 - Clases State

NOTA: Antes de empezar el taller debes completar los pasos previos, haciendo uso de las páginas web que se refieren y de cualesquiera otras que puedan ayudar a resolver cualquier problema de instalación y/o creación de un proyecto Flutter.

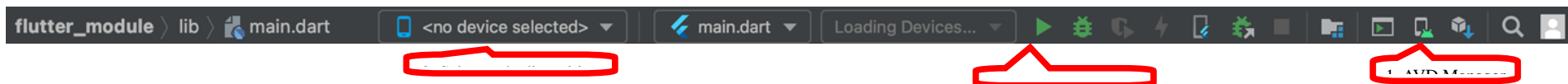
PASOS PREVIOS**1. SDK FLUTTER**


- Si no lo tienes instalado, instala el SDK de Flutter siguiendo <https://docs.flutter.dev/get-started/install>

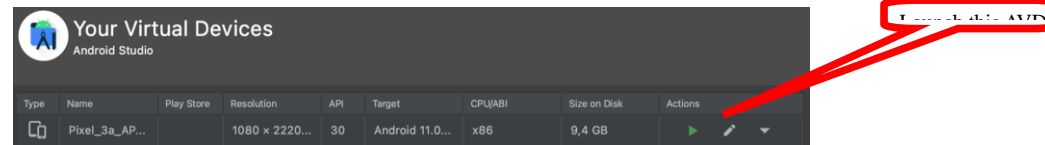
2 (opción 1). ANDROID STUDIO¹ Y EXTENSIÓN PARA FLUTTER

¹. Como alternativa a usar un IDE se puede usar Visual Studio Code como editor enriquecido de texto para este taller, por ser más ligero que Android Studio. Para los pasos previos de instalación y creación de un proyecto Flutter, se puede seguir lo indicado en el siguiente apartado, o bien consultar <https://flutter.dev/docs/get-started/editor?tab=vscode>.


- **Instalación de Android Studio (AS).**- Instala Android Studio y los plugins para Dart y Flutter, según las indicaciones en <https://flutter.dev/docs/get-started/editor#androidstudio>
- **Comprueba la instalación.**- Selecciona **Tools** → **Flutter** → **Flutter Doctor**² y comprueba que no haya errores ((si da algún error, sigue las instrucciones para resolverlo)
- **Creación proyecto Flutter.**- Crea una carpeta *flutter_taller* en tu directorio de trabajo. Abre Android Studio y crea un proyecto Flutter de tipo aplicación dentro de esa carpeta, llámale *flutter_taller*
- **Preparar el dispositivo móvil y ejecutar la app.**- Para ello seguiremos tres pasos, que pueden hacerse los tres en la barra de botones, como se indica a continuación:



1. **Creación de un dispositivo móvil virtual.**- Pincha en el botón AVD Manager y crea un dispositivo virtual para poder probar la app con el emulador de móviles que incluye Android Studio.
 - **Uso del móvil para probar una app (opcional).**- Puedes probar la app en el móvil con un cable USB. En ese caso, debes habilitar en el móvil la "Depuración USB" (en ajustes/configuración → Sistema → ajustes avanzados)
2. **Selección de un dispositivo.**- Selecciona el dispositivo (real o emulado) a usar en la parte central-izquierda de la barra de botones (ver la indicación en la figura anterior). Si es un dispositivo emulado, también se puede seleccionar pinchando la acción  ("Launch this AVD in the emulator") como aparece a continuación:



y espera, si es emulado, a que el emulador muestre el móvil encendido y operativo (puede tardar unos minutos).

3. **Ejecutar la app.**- Lanza la app usando el botón  (*Run*) de la barra de botones o ejecutando *flutter run* desde la consola estando dentro de la carpeta del proyecto.

2 (opción 2). VISUAL STUDIO CODE Y EXTENSIÓN PARA FLUTTER

- **Instalación de Visual Studio Code (VSC).**- Instala Visual Studio Code: <https://code.visualstudio.com/>

². También puedes escribir *flutter doctor* desde un terminal (si solo funciona estando en la carpeta donde está instalado en SDK de flutter, añade el camino a flutter a la variable PATH, en el fichero correspondiente según tu sistema operativo y tu shell de línea de comandos).

- **Adición de los plugins Flutter y Dart.**- Añade los plugins para Flutter y Dart³:
 - Abre Visual Studio
 - Selecciona **View** → **Command Palette** ...
 - Escribe "install" y selecciona '**Extensions: Install Extensions**'
 - En el campo de búsqueda (arriba a la izquierda) introduce *flutter*, selecciona **Flutter** de la lista y pulsa en *Install*
- **Comprueba la instalación.**- Asegúrate de que está todo bien instalado, para ello selecciona: **View** → **Command Palette...** → **Flutter: Run Flutter Doctor**⁴
 - Si da algún error, sigue las instrucciones para resolverlo
- **Creación proyecto Flutter**
 - Selecciona **View** → **Command Palette...**, teclea "flutter", selecciona **Flutter:New Project** y usa la plantilla **Application**
 - Ponle como nombre *flutter_taller*
- **Preparar el dispositivo móvil y ejecutar la app.**- Para ello usaremos la barra de estado de VS Code (barra azul abajo de la ventana), como se indica a continuación:



- **Selección de un dispositivo.**- Selecciona el dispositivo (real o emulado) a usar en la parte derecha de la barra de botones (ver la indicación 1 en la figura anterior). Puedes crear otro dispositivo móvil virtual eligiéndolo de la lista de opciones o elegir tu móvil si está conectado, para ello:
 - **Uso del móvil para probar una app (opcional).**- Puedes probar la app en el móvil con un cable USB. En ese caso, debes habilitar en el móvil la "Depuración USB" (en ajustes/configuración → Sistema → ajustes avanzados)

y espera, si es emulado, a que el emulador muestre el móvil encendido y operativo (puede tardar unos minutos).

- **Ejecutar la app.**- Lanza la app con depuración seleccionando **Debug my code** de la barra de estado o ejecutando **Run** → **Start Debugging** o pulsando F5. O ejecuta sin depuración seleccionando **Run** → **Run Without Debugging**.


DESCRIPCIÓN DEL TALLER

ENTENDIENDO FLUTTER DESDE EL PRINCIPIO (trabajo en parejas)

Vamos a construir una pequeña demo con una página/pantalla/ruta (fichero *main.dart*) de inicio con un botón que, cuando es pulsado 10 veces por el usuario, lo lleva a una segunda página/pantalla/ruta (fichero *paginaPremio.dart*) y le "entrega un premio". Usa los ficheros *flutter.pdf* y *flutter_basics.pdf* como ayuda.

³. El plugin de Dart se instalará automáticamente al instalar el plugin de flutter.

⁴. También puedes escribir *flutter doctor* desde un terminal (si solo funciona estando en la carpeta donde está instalado en SDK de flutter, añade el camino a flutter a la variable PATH, en el fichero correspondiente según tu sistema operativo y tu shell de línea de comandos).

- **Empezar de cero.**- Los ficheros `.dart` se encuentran en la carpeta `lib`. Por ahora solo hay uno: `main.dart`. En vez de borrarlo, renómbralo (por ejemplo `mainOriginal.dart`) con *Cambiar Nombre* (Visual Studio Code) o *Refactor*→*Rename* (Android Studio) en el menú contextual por si quieres consultarlo. Después crea un nuevo fichero Dart de nombre `main.dart` (con *New*→*Dart file* en el menú contextual). Asegúrate de situarte en la carpeta `lib`.
- **Crear la clase principal** (fichero `main.dart`).- Sigue cada uno de los pasos de la siguiente tabla, copiando el código de la primera columna, si lo hay, en el fichero `main.dart`. Tendrás además que seguir las indicaciones del IDE para resolver todos los errores sintácticos, como importar los ficheros que hagan falta). Asegúrate la primera vez que ejecutes el nuevo fichero `main.dart` de que sea ése el que se ejecute y no el antiguo (elige la primera vez *Run 'main.dart'* en el menú contextual estando sobre la ventana de código). Ejecuta la app después de cada paso para ver los cambios, respondiendo en la misma tabla a cada pregunta. Para ahorrar tiempo, ejecuta la app usando la *recarga en caliente* – *hot reload* –. Para ello pincha en el icono  que aparece, en Android Studio en la barra de botones superior o abajo en la barra de botones de la consola y en Visual Studio Code en la barra de botones de depuración arriba en el centro).

#	Código	Pregunta	Respuesta
1. Hello, world!	<pre>void main() { runApp(Center(child: Text('Hello, world!', textDirection: TextDirection.ltr,)),) ;}</pre>	<ul style="list-style-type: none"> • ¿De qué tipo es el argumento del método <code>runApp</code>? • Prueba el código 	<p>Es una función de tipo <code>void</code>, y tiene como parámetro un <code>Widget</code>. Hizo falta importar los paquetes, <code>flutter/material.dart</code> y <code>flutter_test/flutter_test.dart</code>. El código funciona</p>
2. <i>MyApp</i> : una sola clase sin estado	<p>Copia aquí tu código:</p> <pre>void main() { runApp(MyApp()); } class MyApp extends StatelessWidget { @override Widget build(BuildContext context) { return Container(child: Center(child: Text(</pre>	<ul style="list-style-type: none"> • Crea una clase <i>MyApp</i> que sea un widget sin estado (<code>StatelessWidget</code>). Esta clase debe implementar el método <i>build</i> con la siguiente signatura: <i>Widget build (BuildContext context)</i> para construir los widgets de la vista. Mueve el widget <code>Center</code> que se ha puesto como argumento <code>runApp</code> para que sea el contenido que devuelve el método <i>build</i>. • Cambia el argumento del método <code>runApp</code> para crear un objeto de la clase <i>MyApp</i>. • Prueba el código 	<p>El código funciona, visualmente no se aprecia el cambio.</p>

	<pre> 'Hello, world!', textDirection: TextDirection.ltr,),),); } } </pre>		
3. Clase <i>MaterialApp</i>	<p>Copia aquí tu código:</p> <pre> class MyApp extends StatelessWidget { @override Widget build(BuildContext context) { return MaterialApp(title: "Taller Flutter DS Jose Teodosio", theme: ThemeData(primaryS watch: Colors.green), home: Text('Hello, world!', textDirection: TextDirection.ltr,),); } } </pre>	<p>Vamos a añadir ahora la clase <i>MaterialApp</i> para definir el título de la app y el tema (estilo de diseño) de la app. Para ellos, cambiaremos el widget que crea el método (<i>build</i>) por uno más complejo, de tipo <i>MaterialApp</i>, que requiere tres argumentos: <i>title</i> (un string con el título a la app), <i>theme</i> (un widget <i>ThemeData</i> con el tema), y <i>home</i>, un widget para definir el contenido de la página.</p> <ul style="list-style-type: none"> • Pon como título "Taller Flutter DS<mi nombre> • Elige el tema por defecto con el atributo <i>primarySwatch</i> en color verde (<i>Colors.green</i>) • Pon como contenido de la página el widget <i>Text</i> que creaba la anterior versión del método <i>build</i> • Prueba el código • ¿Puedes ver el título y el tema en verde? ¿Qué crees que puede faltar con respecto a la estructura básica de una app Flutter? 	<p>No puedo ver el título, puedo ver el texto de home, el cual es rojo y está subrayado por una doble línea continua amarilla, y no se puede ver tampoco el tema verde pero se puede apreciar en el fondo del icono de la aplicación, el cual es el por defecto de flutter.</p> <p>Faltan las páginas con un layout para diseñar las y si hemos usado el <i>StatelessWidget</i> nos falta el <i>StatefulWidget</i> junto con <i>State</i>.</p>
4. Andamios (Scaffold)	<p>Copia aquí tu código:</p> <pre> class MyApp extends StatelessWidget { const MyApp({super.key}); </pre>	<p>Pues sí, para dar contenido a la primera página de la app necesitamos que el widget pasado como argumento <i>home</i>: de <i>MaterialApp</i> sea un <i>Scaffold</i>. Se trata de un widget que divide la pantalla en tres partes, sus tres argumentos básicos: <i>appBar</i>: una barra arriba donde va el título (widget <i>AppBar</i>), <i>body</i>: el cuerpo (cualquier widget de diseño, por ejemplo <i>Center</i>, <i>Column</i>, <i>Row</i>, ... que se pueden a su vez anidar) y <i>floatingActionButton</i>: un widget <i>FloatingActionButton</i> para alguna acción básica por parte del usuario (por ejemplo, el icono + <i>-Icons.add-</i> para añadir)</p>	<p>Si, ahora si puedo ver el título, el cuerpo y el botón flotante claramente distribuidos en sus zonas.</p> <p>Se podría usar el <i>onPressed</i>, para tener</p>

	<pre> @override Widget build(BuildContext context) { return Scaffold(appBar: AppBar(title: const Text('Ir rapido o no ir rapido', textDirection: TextDirection.ltr,),), body: const Center(child: Text('Hello World!', textDirection: TextDirection.ltr,),), floatingActionButton: FloatingActionButton(// ignore: avoid_returning_null_f or_void onPressed: (() => null), child: const Icon(Icons.add)),); } } </pre>	<ul style="list-style-type: none"> • Modifica el widget pasado como argumento <i>home</i>: en la versión anterior del widget <i>MaterialApp</i> creado de forma que: • sea un <i>Scaffold</i> • contenga como argumento <i>appBar</i> un widget <i>Text</i> con un título ocurrente • contenga como argumento <i>body</i>: todo el contenido anterior (el widget <i>Center</i>) • contenga como <i>floatingActionButton</i> un widget que solo ponga un icono + como hijo: <i>child: Icon(Icons.add)</i> y en el atributo <i>onPressed</i> ponga una función que no haga nada por ahora: <code>onPressed: (()=>null)</code> • Ejecuta la app • ¿Puedes ver ahora el título, el cuerpo y el botón flotante? • ¿Cómo podemos incluir un contador para nuestra app? 	<p>un contador y cada vez que se pulse el botón, se suma uno al contador, y mostrar el contador en el body.</p>
5. Clases para las páginas	<p>Copia aquí tu código:</p> <pre> class MyHomePage </pre>	<p>Para incluir variables y estructurar el código podemos usar más clases. Lo más básico es definir un clase para la página principal que tenga un contador.</p>	<p>La app se ejecuta correctamente, ahora bien no es capaz de</p>

	<pre> extends StatelessWidget { final String title; int counter; MyHomePage({required this.title}) : counter = 0; void _incrementCounter() { counter++; print(counter); } @override Widget build(BuildContext context) { return Scaffold(appBar: AppBar(title: Text(title, textDirection: TextDirection.ltr,),), body: Center(child: Text('Número de veces pulsaciones +: \$counter. ¡Ya queda menos!',),),); } </pre>	<ul style="list-style-type: none"> • Crea una clase <code>MyHomePage</code> (por ahora sin estado) (<i>StatelessWidget</i>), con un atributo constante para el título (<i>title</i>) y otro <i>counter</i>, entero e inicializado a 0. • Define un constructor que acepte el título como argumento y lo asigne al atributo ¿Recuerdas cómo se hace en Dart de forma muy simple?: <code>MyHomePage({this.title});</code> • Define un método privado <code>_incrementCounter()</code> que incremente en uno el atributo <i>counter</i> • Mueve la creación del <i>Scaffold</i> del paso anterior (el argumento <i>home</i>: de <i>MaterialApp</i>) al método <i>build</i> de <i>MyHomePage</i> y asegúrate de pasar al argumento <i>home</i>: una instancia de <i>MyHomePage</i> con el título que antes habías usado • Haz los siguientes cambios en el <i>Scaffold</i> de <i>MyHomePage</i>: <ul style="list-style-type: none"> ◦ Modifica el widget pasado en el argumento <i>floatingActionButton</i> de forma que también llame al método <code>_incrementCounter</code> cuando se presione <code>onPressed: _incrementCounter()</code> ◦ Cambia el widget <i>Text</i> del cuerpo (<i>body</i>.) de la página para que, en vez de poner “Hola mundo”, escriba ‘Número de veces pulsaciones +: \$counter. ¡Ya queda menos!’ • Ejecuta la app • ¿Es capaz de actualizar el contador en la página? ¿Qué puede faltar? 	<p>actualizarel contador en la pagina, falta un <code>setState</code></p>
--	---	---	---

	<pre>floatingActionButton: FloatingActionButton(onPressed: _incrementCounter, tooltip: 'Increment', child: Icon(Icons.add),),); } }</pre>		
6. Clases con estado	<pre>void _incrementCounter() { setState(() { _counter++; }); }</pre> <p>Copia aquí tu código:</p> <pre>import 'package:flutter/material.dart'; void main() { runApp(MyApp()); } class MyApp extends StatelessWidget { @override Widget build(BuildContext context) {</pre>	<p>Para poder usar páginas dinámicas (con estados) debemos crear dos clases:</p> <ol style="list-style-type: none"> 1. Subclase [X] de <i>StatefulWidget</i> con un método para crear el estado (un objeto de <i>State<X></i>) <ul style="list-style-type: none"> • Cambia la clase <i>MyHomePage</i> de forma que: <ul style="list-style-type: none"> ◦ sea <i>StatefulWidget</i> ◦ sobrescriba el método <i>createState</i>, construyendo una instancia de la clase <i>_HomePageState</i> que se explica a continuación 2. Subclase de <i>State<X></i> con el modelo (atributos y métodos) y el método <i>build</i> para definir la vista del widget de la clase X. El widget de la clase X es accesible desde su objeto de la clase <i>State<X></i> por la propiedad <i>widget</i> y su contexto (<i>BuildContext</i>) por la propiedad <i>context</i>. <ul style="list-style-type: none"> • Crea esta clase y llámala <i>_HomePageState</i> • Mueve los métodos que tenías en la clase <i>MyHomePage</i> a <u><i>HomePageState</i></u> (<i>_incrementCounter</i>, y <i>build</i>) y el atributo <i>_counter</i> • ¿Cómo puedes referenciar al atributo <i>title</i> de la clase <i>MyHomePage</i> desde su clase estado? • Ejecuta la app • ¿Es capaz de actualizar el contador en la página? Cambia la implementación del método <i>_incrementCounter</i> por la que aparece a la izquierda. ¿Por qué ahora sí funciona? 	<p>Se puede referenciar el atributo 'title' desde la clase <i>_HomePageState</i> mediante el uso de la propiedad <i>widget</i>, de esta manera: <i>widget.title</i></p> <p>La app se ejecuta perfectamente. Y no es capaz de actualizar el contador correctamente en la página, ya que se debe añadir el <i>setState()</i> para notificar a Flutter que la interfaz de usuario necesita actualizarse.</p>


```
return const
MaterialApp(
  title: 'No se
muestra',
  home:
MyHomePage(
  title: 'Ir rapido o no
ir rapido',
  ),
);
}
}

class MyHomePage
extends StatefulWidget
{
  final String title;

  const
MyHomePage({super.k
ey, required this.title});

  @override
  // ignore:
library_private_types_i
n_public_api
  _HomePageState
createState() =>
  // ignore:
no_logic_in_create_st
ate
  _HomePageState();
}

// ignore:
```

```
must_be_immutable
class
  _HomePageState
  extends
  State<MyHomePage>
  {
    int _counter = 0;

    void
    _incrementCounter() {
      setState(() {
        _counter++;
      });
    }

    @override
    Widget
    build(BuildContext
    context) {
      return Scaffold(
        appBar: AppBar(
          title: Text(
            widget.title,
            textDirection:
            TextDirection.ltr,
          ),
        ),
        body: Center(
          child: Text(
            'Número de
            veces pulsaciones +:
            $_counter. ¡Ya queda
            menos!',
          ),
        ),
      ),
    ),
  },
);
```

```
floatingActionButton:  
FloatingActionButton(  
  onPressed:  
    _incrementCounter,  
  tooltip:  
    'Increment',  
  child:  
    Icon(Icons.add),  
),  
);  
}  
}
```

ENTREGA DE RESULTADOS

Sube a tu espacio personal de PRADO los siguientes dos ficheros:

- Este fichero, completado, en pdf
- Pantalla del móvil o el simulador con la ventana **Premio** cuando se donan todos