# Presentación

# Android Estudio V3.0.17/Flutter

Objetivo:Conocer la presentación de Flutter

F.Creación: 1-4-2018

Programa:

Fuente:[tutorial dart 7 pasos](https://flutter.io/get-started/codelab/)

[Curso dart](https://www.youtube.com/watch?v=capEylVXVi4)

https://www.youtube.com/watch?v=0Jn6EtIiC1Q

## El tradicional hola mundo

**import 'package:flutter/material.dart'**;  
  
**void** main() => runApp(**new** MyApp());  
  
**class** MyApp **extends** StatelessWidget {  
 @override  
 Widget build(BuildContext context) {  
 **return new** MaterialApp(  
 home: **new** Scaffold(  
 appBar: **new** AppBar(  
 title: **new** Text(**'Welcome to Flutter'**),  
 ),  
 body: **new** Center(  
 child: **new** Text(**'Hello World'**),  
 ),  
 ),  
 );  
 }  
}

Stateles🡪Sin estado,son inmutables no pueden cambiar , son valores finales.

[Material](https://material.io/guidelines/) es un lenguaje de diseño visual que es estandar sobre movil y la web

El método principal especifica la notación de flecha fat (=>), que es una abreviatura utilizada para funciones o métodos de una línea.

un ***widget***  es una pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor de *widgets* o *Widget Engine*. Entre sus objetivos están dar fácil acceso a funciones frecuentemente usadas y proveer de información visual. Aunque no es condición indispensable, los widgets suelen ser utilizados para ser "empotrados" en otra página web, copiando el código que el mismo widget pone a disposición del usuario.

La app extiende StatelessWidget lo cual significa que la app en si misma es un Widget.En flutter casi cada cosa es un widget, incluyendo alineamiento, padding y layout.

El widget Scaffold, de la Biblioteca de materiales, proporciona una barra de aplicaciones predeterminada, un título y una propiedad del cuerpo que contiene el árbol de widgets para la pantalla de inicio. El subárbol de widgets puede ser bastante complejo.

El trabajo principal de un widget es proporcionar un método build () que describa cómo mostrar el widget en términos de otros widgets de nivel inferior.

El árbol de widget para este ejemplo consiste de un widget Center conteniendo un texto hijo. El widget Center alinea al subarbol para centrar

## Paso 2: Usando un paquete externo

Se agregan las dependencias en pubspec.yamlm

**name:** startup\_namer  
**description:** startup\_namer   
  
**dependencies:  
 flutter:  
 sdk:** flutter  
  
 *# The following adds the Cupertino Icons font to your application.  
 # Use with the CupertinoIcons class for iOS style icons.* **cupertino\_icons:** ^0.1.0  
  
 **english\_words:** ^3.1.0 🡪 se agrego esta línea, con packaget get se

incorpora  
  
**dev\_dependencies:  
 flutter\_test:  
 sdk:** flutter

en el programa principal se agrega la librería

**import 'package:flutter/material.dart'**;  
**import 'package:english\_words/english\_words.dart'**;🡪 Esta

## Agregar un stateful widget

Su estado puede cambiar durante el tiempo de vida del Widget, implementar un state ful requiere al menos de dos clases:1)Una clase statefulWidget que crea una instancia de 2) Una clase estado, la primera es inmutable,pero la clase estado persiste sobre el tiempo de vida del widget

## Crear un infinito scrolling ListView

## Navigate to a new screen(route)

En Flutter el navegador administra un stack conteniendo las diferentes rutas de la aplicación

**import 'package:flutter/material.dart'**;  
**import 'package:english\_words/english\_words.dart'**;  
  
**void** main() => runApp(**new** MyApp());  
  
**class** MyApp **extends** StatelessWidget {  
 @override  
 Widget build(BuildContext context) {  
 **return new** MaterialApp(  
 title: **'Startup Name Generator'**,  
 theme: **new** ThemeData(  
 primaryColor: Colors.*white*,  
 ),  
 home: **new** RandomWords(),  
 );  
 }  
}  
  
**class** RandomWords **extends** StatefulWidget {  
 @override  
 createState() => **new** RandomWordsState();  
}  
  
**class** RandomWordsState **extends** State<RandomWords> {  
 **final \_suggestions** = <WordPair>[];  
  
 **final \_saved** = **new** Set<WordPair>();  
  
 **final \_biggerFont** = **const** TextStyle(fontSize: 18.0);  
  
 @override  
 Widget build(BuildContext context) {  
 **return new** Scaffold(  
 appBar: **new** AppBar(  
 title: **new** Text(**'Startup Name Generator'**),  
 actions: <Widget>[  
 **new** IconButton(icon: **new** Icon(Icons.*list*), onPressed: \_pushSaved)  
 ],  
 ),  
 body: \_buildSuggestions(),  
 );  
 }  
  
 Widget \_buildSuggestions() {  
 **return new** ListView.builder(  
 padding: **const** EdgeInsets.all(16.0),  
 itemBuilder: (context, i) {  
 **if** (i.**isOdd**) **return new** Divider();  
  
 **final** index = i ~/ 2;  
 **if** (index >= **\_suggestions**.**length**) {  
 **\_suggestions**.addAll(generateWordPairs().take(10));  
 }  
 **return** \_buildRow(**\_suggestions**[index]);  
 },  
 );  
 }  
  
 Widget \_buildRow(WordPair pair) {  
 **final** alreadySaved = **\_saved**.contains(pair);  
 **return new** ListTile(  
 title: **new** Text(  
 pair.**asPascalCase**,  
 style: **\_biggerFont**,  
 ),  
 trailing: **new** Icon(  
 alreadySaved ? Icons.*favorite* : Icons.*favorite\_border*,  
 color: alreadySaved ? Colors.*red* : **null**,  
 ),  
 onTap: () {  
 setState(  
 () {  
 **if** (alreadySaved) {  
 **\_saved**.remove(pair);  
 } **else** {  
 **\_saved**.add(pair);  
 }  
 },  
 );  
 },  
 );  
 }  
  
 **void** \_pushSaved() {  
 Navigator.*of*(**context**).push(  
 **new** MaterialPageRoute(  
 builder: (context) {  
 **final** tiles = **\_saved**.map(  
 (pair) {  
 **return new** ListTile(  
 title: **new** Text(  
 pair.**asPascalCase**,  
 style: **\_biggerFont**,  
 ),  
 );  
 },  
 );  
 **final** divided = ListTile  
 .*divideTiles*(  
 context: context,  
 tiles: tiles,  
 )  
 .toList();  
  
 **return new** Scaffold(  
 appBar: **new** AppBar(  
 title: **new** Text(**'Saved Suggestions'**),  
 ),  
 body: **new** ListView(children: divided),  
 );  
 },  
 ),  
 );  
 }  
}