# Segunda

# Android Estudio V3.0.17/Flutter

Objetivo:Avanzar en el estudio de Flutter

F.Creación: 5-4-2018

Programa:

Fuente: [segunda presentacion](https://flutter.io/tutorials/layout/#step-0)

[Filas y columnas](https://flutterdoc.com/understanding-layouts-in-flutter-73a797e7cd3b)

[Tratamiento de imagenes assets](https://flutter.io/assets-and-images/)

[Container Class](https://docs.flutter.io/flutter/widgets/Container-class.html)

1. Se [incorpora fotografía al proyecto](https://github.com/flutter/website/blob/master/_includes/code/layout/lakes/images/lake.jpg), en la raiz del directorio se crea images

flutter:

uses-material-design: true

assets:

* images/lake.jpg

Posteriormente se referencia con

laImagen(){  
 **new** Image.asset(  
 **'images/lake.jpg'**,  
 width: 600.0,  
 height: 240.0,  
 fit: BoxFit.**cover**,  
 )

[análisis de BoxFit](https://docs.flutter.io/flutter/painting/BoxFit-class.htmlhttps:/docs.flutter.io/flutter/painting/BoxFit-class.html)

import 'dart:async' show Future;

import 'package:flutter/services.dart' show rootBundle;

Future<String> loadAsset() async {

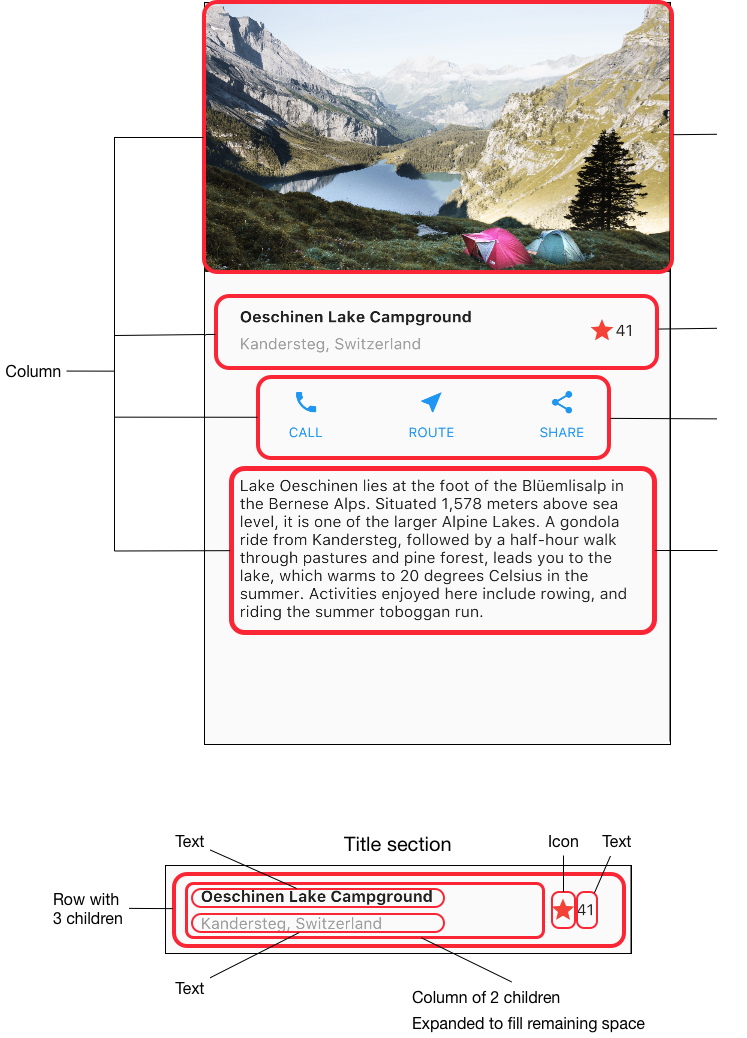
return await rootBundle.loadString('assets/config.json');

}

1. El primer paso es diagramar el layout

* Identificar filas y columnas
* Existe sobreposición de estos elementos?
* Se necesitan tab?
* Observe areas que requieren alineamiento, padding o bordes

1. Primero, identifica los elementos más grandes. En este ejemplo, cuatro elementos se organizan en una columna: una imagen, dos filas y un bloque de texto.



1. Step 2: Implement the title row

Primero, construirás la columna de la izquierda en la sección del título. Al poner la Columna dentro de un widget expandido se extiende la columna para usar todo el espacio libre restante en la fila. Al establecer la propiedad crossAxisAlignment en CrossAxisAlignment.start, la columna se coloca al principio de la fila.

Poner la primera fila de texto dentro de un contenedor permite agregar relleno. El segundo hijo en la columna, también texto, se muestra en gris.

Los dos últimos elementos en la fila del título son un ícono de estrella, pintado de rojo, y el texto "41". Coloque toda la fila en un contenedor y rellene a lo largo de cada borde con 32 píxeles.

1. En principio el programa principal queda

**return new** MaterialApp(  
 title: **'Flutter Demo'**,  
 home: **new** Scaffold(  
 appBar: **new** AppBar(  
 title: **new** Text(**'Top Lakes'**),  
 ),  
 body: **new** ListView(  
 children: [  
 laImagen(),  
 titleSection,  
 buttonSection,  
 textSection,  
 ],  
 ),  
 ),  
);

Un ListView es una lista desplazable de widgets dispuestos linealmente.

ListView es el widget de desplazamiento más comúnmente utilizado. Muestra sus hijos uno tras otro en la dirección de desplazamiento.

Si no es nulo, el elemento itemExtent obliga a los hijos a tener la extensión dada en la dirección de desplazamiento. Especificar un elemento itemExtent es más eficiente que dejar que los hijos determinen su propia extensión porque la maquinaria de desplazamiento puede utilizar el conocimiento previo de la extensión de los niños para guardar el trabajo, por ejemplo, cuando la posición de desplazamiento cambia drásticamente.

Hay tres opciones para construir un ListView:

El constructor predeterminado toma una lista explícita de hijos. Este constructor es apropiado para vistas de lista con un número pequeño de hijos porque la construcción de la Lista requiere hacer un trabajo para cada niño que posiblemente podría mostrarse en la vista de lista en lugar de solo aquellos niños que son realmente visibles.

ListView.builder toma IndexedWidgetBuilder, que crea los elementos bajo demanda. Este constructor es apropiado para vistas de lista con un número grande (o infinito) de hijos porque el constructor se llama solo para aquellos niños que son realmente visibles.

ListView.custom toma un SliverChildDelegate, que proporciona la capacidad de personalizar aspectos adicionales del modelo hijo. Por ejemplo, un SliverChildDelegate puede controlar el algoritmo utilizado para estimar el tamaño de los niños que no son realmente visibles.

## TitleSection

Aquí está el código que implementa la fila del título.

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

Widget titleSection = new Container(

padding: const EdgeInsets.all(32.0),

child: new Row(

children: [

new Expanded(

child: new Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

new Container(

padding: const EdgeInsets.only(bottom: 8.0),

child: new Text(

'Oeschinen Lake Campground',

style: new TextStyle(

fontWeight: FontWeight.bold,

),

),

),

new Text(

'Kandersteg, Switzerland',

style: new TextStyle(

color: Colors.grey[500],

),

),

],

),

),

new Icon(

Icons.star,

color: Colors.red[500],

),

new Text('41'),

],

),

);

//...

}

Al crear diseños para nuestras aplicaciones de Flutter, es importante comprender cómo el marco se encarga de la creación de estos diseños.

Dentro de Flutter, los componentes visuales que usted declara son widgets, así como los elementos de diseño también. The Flutter Framework construye su diseño mediante la composición de widgets, todo lo que usted construye programáticamente es un widget y estos se compilan juntos para crear la interfaz de usuario. .

frase1(String linea){  
 **return new** Container(  
 padding: **const** EdgeInsets.only(bottom: 8.0),  
 child: **new** Text(  
 linea,  
 style: **new** TextStyle(  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 );  
}

Un widget de conveniencia que combina widgets comunes de pintura, posicionamiento y tamaño.

Un container primero rodea al hijo con padding (inflado por cualquier borde presente en la decoración) y luego aplica restricciones adicionales (incorporando el ancho y la altura como restricciones, si cualquiera de ellas es no nulo). El contenedor está rodeado por un espacio vacío adicional descrito desde el margen.

Durante la pintura, el contenedor primero aplica la transformación dada, luego pinta la decoración para llenar la extensión acolchada, luego pinta al niño y finalmente pinta la Decoración de primer plano, que también llena la extensión acolchada.

Los contenedores sin hijos intentan ser tan grandes como sea posible a menos que las restricciones entrantes no tengan límites, en cuyo caso intentan ser lo más pequeños posible. Los envases con los niños se adaptan a sus hijos. Los argumentos de ancho, alto y restricciones para el constructor anulan esto.

Rectangulo chico

rectangulo() {  
 **return new** Center(  
 child: **new** Container(  
 margin: **const** EdgeInsets.all(10.0),  
 color: **const** Color(0xFF00FF00),  
 width: 48.0,  
 height: 48.0,  
 )  
 );  
}

puede ser mucho mas complejp

mucho(){  
 **return new** Container(  
 constraints: **new** BoxConstraints.expand(  
 height: Theme.*of*(context).**textTheme**.**display1**.**fontSize** \* 1.1 + 200.0,  
 ),  
 padding: **const** EdgeInsets.all(8.0),  
 color: Colors.*teal*.**shade700**,  
 alignment: Alignment.*center*,  
 child: **new** Text(**'Hello World'**, style: Theme.*of*(context).**textTheme**.**display1**.copyWith(color: Colors.*white*)),  
 foregroundDecoration: **new** BoxDecoration(  
 image: **new** DecorationImage(  
 image: **new** NetworkImage(**'https://pbs.twimg.com/card\_img/981828794553192448/BK6bbqDY?format=jpg&name=600x314'**),  
 centerSlice: **new** Rect.fromLTRB(270.0, 180.0, 1360.0, 730.0),  
 ),  
 ),  
 transform: **new** Matrix4.rotationZ(0.1),  
 );  
}

## La botonera

Column buildButtonColumn(IconData icon, String label) {

Color color = Theme.*of*(context).**primaryColor**;  
  
 **return new** Column(  
 mainAxisSize: MainAxisSize.**min**,  
 mainAxisAlignment: MainAxisAlignment.**center**,  
 children: [  
 **new** Icon(icon, color: color),  
 **new** Container(  
 margin: **const** EdgeInsets.only(top: 8.0),  
 child: **new** Text(  
 label,  
 style: **new** TextStyle(  
 fontSize: 12.0,  
 fontWeight: FontWeight.*w400*,  
Column buildButtonColumn(IconData icon, String label) {  
 color: color,  
 ),  
 ),  
 ),  
 ],  
 );  
}  
  
Widget buttonSection = **new** Container(  
 child: **new** Row(  
 mainAxisAlignment: MainAxisAlignment.**spaceEvenly**,  
 children: [  
 buildButtonColumn(Icons.*call*, **'CALL'**),  
 buildButtonColumn(Icons.*near\_me*, **'ROUTE'**),  
 buildButtonColumn(Icons.*share*, **'SHARE'**),  
 ],  
 ),  
);

MainAxisAlignment enum🡪 Es la forma en que lo widget se alinean en el eje x (Center,end, spaceAround , spaceBetween , spaceEvenly , start

spaceAround : Coloque el espacio libre de manera uniforme entre los hijos, así como la mitad de ese espacio antes y después del primer y último hijo

spaceBetween : Coloca el espacio entre los hijos

spaceEvenly : Coloca el espacio libre de manera uniforme entre los hijos, así como antes y después del primer y último hijo.

Como el código para construir cada fila sería casi idéntico, es más eficiente crear una función anidada, como buildButtonColumn (), que toma un icono y un texto, y devuelve una columna con sus widgets pintados en el color primario.

## Step 4: Implement the text section

Widget textSection = new Container(

padding: const EdgeInsets.all(32.0),

child: new Text(

'''

Lake Oeschinen lies at the foot of the Blüemlisalp in the Bernese Alps. Situated 1,578 meters above sea level, it is one of the larger Alpine Lakes. A gondola ride from Kandersteg, followed by a half-hour walk through pastures and pine forest, leads you to the lake, which warms to 20 degrees Celsius in the summer. Activities enjoyed here include rowing, and riding the summer toboggan run.

''',

softWrap: true,

),

);

//...

}

padding: const EdgeInsets.all(32.0),

Un conjunto inmutable de desplazamientos en cada uno de los cuatro puntos cardinales.

Normalmente se usa para un desplazamiento de cada uno de los cuatro lados de una caja. Por ejemplo, el padding dentro de una caja se puede representar utilizando esta clase

La clase EdgeInsets especifica las compensaciones en términos de bordes visuales, izquierda, arriba, derecha e inferior. Estos valores no se ven afectados por la TextDirection. Para admitir diseños de izquierda a derecha y de derecha a izquierda, considere usar EdgeInsetsDirectional, que se expresa en términos de inicio, inicio, final e inferior, donde el inicio y el final se resuelven en términos de una dirección de texto (generalmente obtenida de la direccionalidad ambiental).

**const** EdgeInsets.all(8.0)-> Margen por todos lados

**const** EdgeInsets.symmetric(vertical: 8.0)🡪Solamente arriba y abajo

**const** EdgeInsets.only(left: 40.0)🡪 Solo a la izquierda

## Programa completo

**void** main() {  
 *//debugPaintSizeEnabled = true;* runApp(**new** MyApp());  
}  
  
**class** MyApp **extends** StatelessWidget {  
 @override  
 Widget build(BuildContext context) {  
 Widget titleSection = **new** Container(  
 padding: **const** EdgeInsets.all(32.0),  
 child: **new** Row(  
 children: [  
 **new** Expanded(  
 child: **new** Column(  
 crossAxisAlignment: CrossAxisAlignment.**start**,  
 children: [  
 **new** Container(  
 padding: **const** EdgeInsets.only(bottom: 8.0),  
 child: **new** Text(  
 **'Oeschinen Lake Campground'**,  
 style: **new** TextStyle(  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 **new** Text(  
 **'Kandersteg, Switzerland'**,  
 style: **new** TextStyle(  
 color: Colors.*grey*[500],  
 ),  
 ),  
 ],  
 ),  
 ),  
 **new** Icon(  
 Icons.*star*,  
 color: Colors.*red*[500],  
 ),  
 **new** Text(**'41'**),  
 ],  
 ),  
 );  
  
 Column buildButtonColumn(IconData icon, String label) {  
 Color color = Theme.*of*(context).**primaryColor**;  
  
 **return new** Column(  
 mainAxisSize: MainAxisSize.**min**,  
 mainAxisAlignment: MainAxisAlignment.**center**,  
 children: [  
 **new** Icon(icon, color: color),  
 **new** Container(  
 margin: **const** EdgeInsets.only(top: 8.0),  
 child: **new** Text(  
 label,  
 style: **new** TextStyle(  
 fontSize: 12.0,  
 fontWeight: FontWeight.*w400*,  
 color: color,  
 ),  
 ),  
 ),  
 ],  
 );  
 }  
  
 Widget buttonSection = **new** Container(  
 child: **new** Row(  
 mainAxisAlignment: MainAxisAlignment.**spaceEvenly**,  
 children: [  
 buildButtonColumn(Icons.*call*, **'CALL'**),  
 buildButtonColumn(Icons.*near\_me*, **'ROUTE'**),  
 buildButtonColumn(Icons.*share*, **'SHARE'**),  
 ],  
 ),  
 );  
  
 Widget textSection = **new** Container(  
 padding: **const** EdgeInsets.all(32.0),  
 child: **new** Text(  
 **'''  
Lake Oeschinen lies at the foot of the Blüemlisalp in the Bernese Alps. Situated 1,578 meters above sea level, it is one of the larger Alpine Lakes. A gondola ride from Kandersteg, followed by a half-hour walk through pastures and pine forest, leads you to the lake, which warms to 20 degrees Celsius in the summer. Activities enjoyed here include rowing, and riding the summer toboggan run.  
 '''**,  
 softWrap: **true**,  
 ),  
 );  
  
 imagenSection(){  
 **new** Image.asset(  
 **'images/lake.jpg'**,  
 width: 600.0,  
 height: 240.0,  
 fit: BoxFit.**cover**,  
 );  
 }  
 **return new** MaterialApp(  
 title: **'Flutter Demo'**,  
 home: **new** Scaffold(  
 appBar: **new** AppBar(  
 title: **new** Text(**'Top Lakes'**),  
 ),  
 body: **new** ListView(  
 children: [  
 imagenSection(),  
 titleSection,  
 buttonSection,  
 textSection,  
 ],  
 ),  
 ),  
 );  
 }  
}