# 1.30 flutter\_planets\_tutorial

# Android Estudio V3.0.17/Flutter

Objetivo:

F.Creación: 1-4-2018

Programa: flutter\_planets\_tutorial

Fuente: [Sergeiandremplace](https://sergiandreplace.com/planets-flutter-routing-and-navigation/)

## Imágenes

|  |  |
| --- | --- |
|  |  |
|  |  |

La pantalla principal es un Scaffold que llama a una columna que contiene dos hijos el título principal y el scrool de planetas

## Veamos el titulo(GradientAppBar)

**class** GradientAppBar **extends** StatelessWidget {  
  
 **final** String **title**;  
 **final** double **barHeight** = 66.0;  
  
 GradientAppBar(**this**.**title**);  
  
 @override  
 Widget build(BuildContext context) {  
 **final** double statusBarHeight = MediaQuery  
 .*of*(context)  
 .**padding** .**top**;  
  
 **return new** Container(  
 padding: **new** EdgeInsets.only(top: statusBarHeight),  
 height: statusBarHeight + **barHeight**,  
 child: **new** Center(  
 child: **new** Text(**title**,  
 style:**const** TextStyle(  
 color: Colors.*white*,  
 fontFamily: **'Poppins'**,  
 fontWeight: FontWeight.*w600*,  
 fontSize: 36.0  
 ),  
 ),  
 ),  
 decoration: **new** BoxDecoration(  
 gradient: **new** LinearGradient(  
 colors: [  
 **const** Color(0xFF3366FF),  
 **const** Color(0xFF00CCFF)  
 ],  
 begin: **const** FractionalOffset(0.0, 0.0),  
 end: **const** FractionalOffset(1.0, 0.0),  
 stops: [0.0, 1.0],  
 tileMode: TileMode.**clamp** ),  
 ),  
 );  
 }  
}

MediaQuery🡪 Establece un subárbol en el cual las mediaQuery se resuelven los datos dados.Por ejemplo para captar el tamaño del medio actual, es decir la window conteniendo tu app, tu puedes leer la propiedad [MediaQueryData.size](https://docs.flutter.io/flutter/widgets/MediaQueryData/size.html) desde la [MediaQueryData](https://docs.flutter.io/flutter/widgets/MediaQueryData-class.html)  retornadas por medio de MediaQuery.of(context).size.

Consultando el medio actual usando [MediaQuery.of](https://docs.flutter.io/flutter/widgets/MediaQuery/of.html) causa que tu Widget se reconstruya automaticamente sin que la [MediaQueryData](https://docs.flutter.io/flutter/widgets/MediaQueryData-class.html) cambie, si el usuatio rota su dispositivo

Si [MediaQuery](https://docs.flutter.io/flutter/widgets/MediaQuery-class.html) no esta en el horizonte entonces el método  [MediaQuery.of](https://docs.flutter.io/flutter/widgets/MediaQuery/of.html) salta a una excepción , a menos que el  nullOkargument este en verdadero, en cuyo caso retorna null

MediaQuery  
 .*of*(context)  
 .**padding** .**top**;  
entonces encuentra el padding top de la pantalla actual

BoxDecoration🡪Una descripción inmutable de como pintar una caja, de varias maneras.Tiene un borde , un cuerpo y efectos de sombra.

Ejemplo

**new** Container(

decoration: **new** BoxDecoration(

color: **const** Color(0xff7c94b6),

image: **new** DecorationImage(

image: **new** ExactAssetImage('images/flowers.jpeg'),

fit: BoxFit.cover,

),

border: **new** Border.all(

color: Colors.black,

width: 8.0,

),

),

LinearGradient🡪 Variación de una magnitud en función de la distancia(A 2D linear gradient.). Esta clase es usada por BoxDecoration para representar gradientes.Un gradiente tiene dos punts de anclaje un comienzo y un final, 0.0 y 1.0 respectivamente, se expresan asi para que puedan ser usadas en varias cajas de diferente tamaño.Permite una degradacion del color, o el paso de un color a otro.

FractionalOffset🡪[FractionalOffset](https://docs.flutter.io/flutter/painting/FractionalOffset-class.html) y [Alignment](https://docs.flutter.io/flutter/painting/Alignment-class.html) son dos diferentes representaciones de la misma información: La localización dentro de un rectangulo relativo al tamaño del rectanguloe. La diferencia entre las dos clases esta en el sistema de coordenadas que ellos usan para representar la localización.

[FractionalOffset](https://docs.flutter.io/flutter/painting/FractionalOffset-class.html) usa un systema de coordenadas como un origen en laesquina top-left del rectángulo mientras [Alignment](https://docs.flutter.io/flutter/painting/Alignment-class.html) usa un sistema de coordenadas desde el centro del rectángulo.

Historicamente, [FractionalOffset](https://docs.flutter.io/flutter/painting/FractionalOffset-class.html) precede [Alignment](https://docs.flutter.io/flutter/painting/Alignment-class.html). Cuando nosotros intentamos hacer una versión de [FractionalOffset](https://docs.flutter.io/flutter/painting/FractionalOffset-class.html) que se adaptara al [TextDirection](https://docs.flutter.io/flutter/dart-ui/TextDirection-class.html), nos topamos con dificultades porque al colocar el origen en la esquina superior izquierda introdujo un sesgo de izquierda a derecha que era difícil de eliminar.

Por colocar el origen en el centro , [Alignment](https://docs.flutter.io/flutter/painting/Alignment-class.html) y [AlignmentDirectional](https://docs.flutter.io/flutter/painting/AlignmentDirectional-class.html) son capaces de usar el mismo origen , usando una función lineal para resolver un  [AlignmentDirectional](https://docs.flutter.io/flutter/painting/AlignmentDirectional-class.html)  sobre  [Alignment](https://docs.flutter.io/flutter/painting/Alignment-class.html)  en ambos [TextDirection.rtl](https://docs.flutter.io/flutter/dart-ui/TextDirection-class.html) y [TextDirection.ltr](https://docs.flutter.io/flutter/dart-ui/TextDirection-class.html).

[Alignment](https://docs.flutter.io/flutter/painting/Alignment-class.html) es mejor para muchos propositos que  [FractionalOffset](https://docs.flutter.io/flutter/painting/FractionalOffset-class.html) y debe ser usado en su lugar

## El alineamiento de planetas

**class** HomePageBody **extends** StatelessWidget {  
 @override  
 Widget build(BuildContext context) {  
 **return new** Expanded(  
 child: **new** Container(  
 color: **new** Color(0xFF736AB7),  
 child: **new** CustomScrollView(  
 scrollDirection: Axis.**vertical**,  
 shrinkWrap: **false**,  
 slivers: <Widget>[  
 **new** SliverPadding(  
 padding: **const** EdgeInsets.symmetric(vertical: 24.0),  
 sliver: **new** SliverList(  
 delegate: **new** SliverChildBuilderDelegate(  
 (context, index) => **new** PlanetRow(planets[index]),  
 childCount: planets.**length**,  
 ),  
 ),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}

CustomScrollView🡪 Un [ScrollView](https://docs.flutter.io/flutter/widgets/ScrollView-class.html) que crea creates efectos personalizados de scroll usando slivers.

[CustomScrollView](https://docs.flutter.io/flutter/widgets/CustomScrollView-class.html) permite proporcionar [slivers](https://docs.flutter.io/flutter/widgets/CustomScrollView/slivers.html) directamente para crear varios efectos de scrollin tal como listras, frillas, y expanding headers. Por ejemplo para crear un scroll view que contenga un app bar seguida por una lista y una grilla se usa una lista de tres slivers: [SliverAppBar](https://docs.flutter.io/flutter/material/SliverAppBar-class.html), [SliverList](https://docs.flutter.io/flutter/widgets/SliverList-class.html), y [SliverGrid](https://docs.flutter.io/flutter/widgets/SliverGrid-class.html).

[Widget](https://docs.flutter.io/flutter/widgets/Widget-class.html)s en estos [slivers](https://docs.flutter.io/flutter/widgets/CustomScrollView/slivers.html) debe producir [RenderSliver](https://docs.flutter.io/flutter/rendering/RenderSliver-class.html) objetos.

Para controlar les ofset inicial del scroll de la vista, proporcione un controlador con su  [controller](https://docs.flutter.io/flutter/widgets/ScrollView/controller.html) y su [ScrollController.initialScrollOffset](https://docs.flutter.io/flutter/widgets/ScrollController/initialScrollOffset.html)

[Ejemplo](https://docs.flutter.io/flutter/widgets/CustomScrollView-class.html)

Podemos ver que el que hace el trabajo es PlanetRow, al que se le pasa el índice.

## PlanetRow

Hay más código pero el otro esta visto

**return new** GestureDetector(  
 onTap: () => Navigator.of(context).push(**new** PageRouteBuilder(  
 pageBuilder: (\_, \_\_, \_\_\_) => **new** DetailPage(planet),  
 )),  
 child: **new** Container(  
 height: 120.0,  
 margin: **const** EdgeInsets.symmetric(  
 vertical: 16.0,  
 horizontal: 24.0,  
 ),  
 child: **new** Stack(  
 children: <Widget>[  
 planetCard,  
 planetThumbnail,  
 ],  
 ),  
 )  
 );  
 }  
}

GestureDetector🡪 Un widget que detecta gestures,que corresponden a su funciones callbacks.

If this widget has a child, it defers to that child for its sizing behavior. If it does not have a child, it grows to fit the parent instead.

Por defecto un GestureDetector con un hijo invisible ignorar los toques, este comportamiento puede ser controlado con [behavior](https://docs.flutter.io/flutter/widgets/GestureDetector/behavior.html).

GestureDetector también escucha por efectos de accessibilida y mapas llamandin a los callbacks. Para ingnorar eventos de accesibilidad set [excludeFromSemantics](https://docs.flutter.io/flutter/widgets/GestureDetector/excludeFromSemantics.html) a true.

En este caso llama a la pantalla DetailPage conn un parámetro.

**const** EdgeInsets.symmetric🡪

**const** EdgeInsets.symmetric(vertical: 8.0),margen abajo y arriba