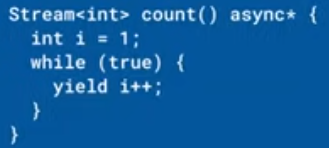
**StreamBuilder**



StreamBuilder escuchará los eventos que fluyen de un Stream

Para cada nuevo evento reconstruye el widget

Definimos el stream para el StreamBuilder

Luego para un snapshot definimos el widget que se debe ver.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Se pueden dar datos iniciales para ver que se muestra antes del primer snapshot

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Hay que verificar si se tienen datos.

Si no los hay se puede mostrar un indicador de carga.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Se deben comprobar los valores de ConnectionState

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

También hay que revisar si hay errores.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

|  |  |
| --- | --- |
| **Código** | **Comentarios** |
| import 'dart:async';  import 'package:flutter/material.dart';  class StreamBuilderScreen extends StatelessWidget {    const StreamBuilderScreen({Key? key}) : super(key: key);    @override    Widget build(BuildContext context) {      return \_MyHomeScreen();    }  }  //--------------------------------------------------------------------------  class \_MyHomeScreen extends StatefulWidget {    \_MyHomeScreen({      Key? key,    }) : super(key: key);    @override    State<\_MyHomeScreen> createState() => \_MyHomeScreenState();  }  class \_MyHomeScreenState extends State<\_MyHomeScreen> {    final colorStream = StreamController<Color>();    int counter = -1;    final List<Color> colorList = [      Colors.blue,      Colors.red,      Colors.yellow,      Colors.green,      Colors.orange,      Colors.brown,      Colors.purple,      Colors.cyan,      Colors.pink,      Colors.white,      Colors.grey,      Colors.black,      Colors.amber,      Colors.indigo,      Colors.redAccent,    ];  //---------------------------------------    @override    void dispose() {      colorStream.close();      super.dispose();    }  //---------------------------------------    @override    Widget build(BuildContext context) {      return Scaffold(        appBar: AppBar(          title: const Text('StreamBuilder'),          centerTitle: true,        ),        body: Center(          child: StreamBuilder(            stream: colorStream.stream,            //initialData: initialData,            builder: (BuildContext context, AsyncSnapshot snapshot) {              if (!snapshot.hasData) {                return const LoadingWidget();              }              if (snapshot.connectionState == ConnectionState.done) {                return const Text("No hay más colores");              }              return Container(                height: 150,                width: 150,                color: snapshot.data,              );            },          ),        ),        floatingActionButton: FloatingActionButton(          onPressed: () {            counter++;            if (counter < colorList.length) {              colorStream.sink.add(colorList[counter]);            } else {              colorStream.close();            }          },          child: const Icon(Icons.color\_lens),        ),      );    }  }  //-----------------------------------------------------------------  class LoadingWidget extends StatelessWidget {    const LoadingWidget({      Key? key,    }) : super(key: key);    @override    Widget build(BuildContext context) {      return Column(        mainAxisAlignment: MainAxisAlignment.center,        children: const [          CircularProgressIndicator(),          SizedBox(            height: 20,          ),          Text("Esperando Data...")        ],      );    }  } |  |

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -