

jSon y Futuros en Flutter

Fecha: 24-julio-2017

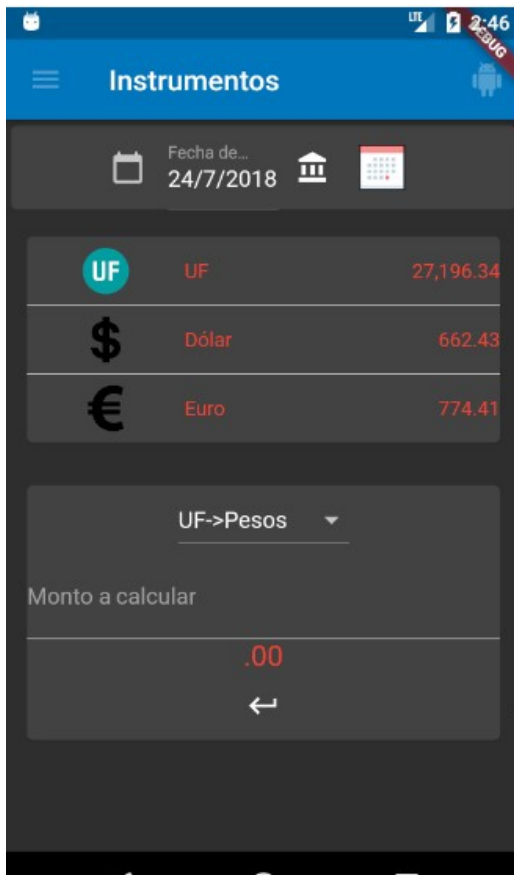
Autor Máximo Meza C.



Motivación: Necesito saber cuantos pesos (moneda chilena) son un Dólar, un Euro y una unidad de Fomento, esta información varía día a día y existe una página que da estos valores <https://mindicador.cl/>

La información la página la proporciona en formato Json. La función que busca la página se demora un lapso en entregar los resultados, motivo por el cual se habla de entregas a futuro.

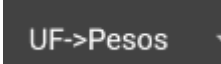
La sintáxis debiera ser similar a la que se muestra a continuación:
`valorInstrumentoFinanciero("Instrumento Financiero", "fecha");`


Presentación:




Al escribir una fecha se debe presionar  o bien usar el 

Para que se despliegan más abajo los instrumentos.

 es un `DropDownButton` que permite seleccionar entre diferentes instrumentos para convertir a pesos o a la inversa.

Al ingresar el monto a calcular y presionar  se obtiene el valor convertido



ni  funciona

Futuros: Hay futuros que te devuelven un flujo de datos (stream) y futuros que te devuelven un solo valor, este último tipo es el que nos interesa.

Para obtener el valor que representa el futuro, tiene dos palabras reservadas: `async` y `await`. La función queda entonces así

```
Future<double> valorInstrumentoFinanciero(
    TipoInstrumentoFinanciero tIF,
    DateTime dt) async {
    Cuerpo de la función
}
```

`TipoInstrumentoFinanciero` es un enumerado, Y cuando se necesite usar la función se llama de esta forma

```
double vuf= await valorInstrumentoFinanciero(TipoInstrumentoFinanciero.Uf, new DateTime.now());
Otra línea
```

El `await` indica dos cosas distintas i) El programa comienza a ejecutar la función solicitada ii) continua ejecutando la otra línea

Esta función si se desea llamar desde el main, el main debe estar definido como asincrónico

```
main() async{
}
```

O bien ser llamada desde dentro de otra función que fue lo que hice yo

```
Future<String> asyncFunction() async {
    try {
        list = await Future.wait([
            valorInstrumentoFinanciero(TipoInstrumentoFinanciero.Uf, new DateTime.now()),
            valorInstrumentoFinanciero(TipoInstrumentoFinanciero.Dolar, new DateTime.now()),
            valorInstrumentoFinanciero(TipoInstrumentoFinanciero.Euro, new DateTime.now())
        ]);
    } catch (e) {
        return e.toString();
    }
    return "";
}
```

En este caso se está definiendo una lista de llamadas a la misma función pero con parámetros distintos, esto tiene interesante que las tres llamadas se ejecutan en paralelo. Para ejecutar esta nueva función `asyncFunction()`, se utiliza la siguiente sintaxis

```
asyncFunction().then((val) {
    //Donde val es el retorno de la función
    print("Los tres son: ${list[0]} ${list[1]} ${list[2]}");
});
```

```
});
```

Donde val es el retorno de la función,"OK" en este caso

Url: La URL que nos devuelve esta función es de la página web miindicador.cl y tiene la siguiente sintáxis

https://mindicador.cl/api/{tipo_indicador}/{dd-mm-yyyy}

Por ejemplo <https://mindicador.cl/api/dolar/12-07-2018>

Devolviendonos el jSon

```
{"version":"1.5.0","autor":"mindicador.cl","codigo":"dolar","nombre":
"Dólar observado","unidad_medida":"Pesos","serie":[{"fecha":"2018-07-
12T04:00:00.000Z","valor":652.37}]}
```

Json: Al ordenarlo visualmente, queda (para ordenarlo visualmente lo más fácil es usar <https://jsonformatter.org/>)

```
{
  "version": "1.5.0",
  "autor": "mindicador.cl",
  "codigo": "dolar",
  "nombre": "Dólar observado",
  "unidad_medida": "Pesos",
  "serie": [
    {
      "fecha": "2018-07-12T04:00:00.000Z",
      "valor": 652.37
    }
  ]
}
```

El formato entregado tiene dos tipos de valores, una pareja de valores ó una lista de parejas, que es donde esta el valor que nos interesa..muy similar a los Map de Dart

La idea es lograr meter(parsear) este archivo a una clase `_IndicadorFinanciero` que contenga dentro de si una lista de la clase `_serie`.

Ambas clases contendran un factory (un objeto que crea una instancia)

Búsqueda del indicador financiero

```
Future<_IndicadorFinanciero> _buscaIndicador(String url) async {

    final response = await http.get(url);
    if (response.statusCode == 200) {
        return _IndicadorFinanciero.fromJson(json.decode(response.body));
    } else {
        throw Exception(
            '${TipoError.IndicadorNoDisponible.index}==>${
                lErrores[TipoError.IndicadorNoDisponible.index] } ${response.statusCode} ');
    }
}
```

La URL nos devuelve un status de 200, indicando que todo esta ok, la función que hace el trabajo es la funcion decode, que cambia un poco la estructura del archivo

```
{version: 1.5.0, autor: mindicador.cl, codigo: dolar, nombre: Dólar observado, unidad_medida: Pesos,
serie: [{fecha: 2018-07-12T04:00:00.000Z, valor: 652.37}]} y esto claramente son Map
```

Llama entonces a la fabrica del _IndicadorFinanciero, que es la que hace el trabajo, abajo van ambas clases

Fuentes: [Run async operation on widget creation](#)

```

class _IndicadorFinanciero {

    final String version;
    final String autor;
    final String codigo;
    final String nombre;
    final String unidadMedida;
    final List<_Serie> lSerie;

    _IndicadorFinanciero(
        {this.version,
         this.autor,
         this.codigo,
         this.nombre,
         this.unidadMedida,
         this.lSerie});

    factory _IndicadorFinanciero.fromJson(Map<String, dynamic> json) {

        List<dynamic> list = json['serie'] as List;
        List<_Serie> serieList = list.map((i) => _Serie.fromJson(i)).toList();

        return _IndicadorFinanciero(
            version: json['version'],
            autor: json['autor'],
            codigo: json['codigo'],
            nombre: json['nombre'],
            unidadMedida: json['unidad_medida'],
            lSerie: serieList,
        );
    }
}

```

La factory recibe de parámetro el Map comentado, siendo una pareja compuesta por (string, y "cualquier cosa")

`list.map((i) => _Serie.fromJson(i)).toList();` es una de las magias de dart, para mapear.

```

class _Serie {

    String fecha;
    double valor;

    _Serie({this.fecha, this.valor});

    factory _Serie.fromJson(Map<String, dynamic> json) {
        return _Serie(fecha: json['fecha'], valor: json['valor']);
    }
}

```

Poniendo esto en una pantalla

La pantalla tiene que ser statfull y en la inicialización del estado usamos `async/await`. Como la pantalla se despliega antes de que esten los datos, debemos contemplar las dos situaciones.

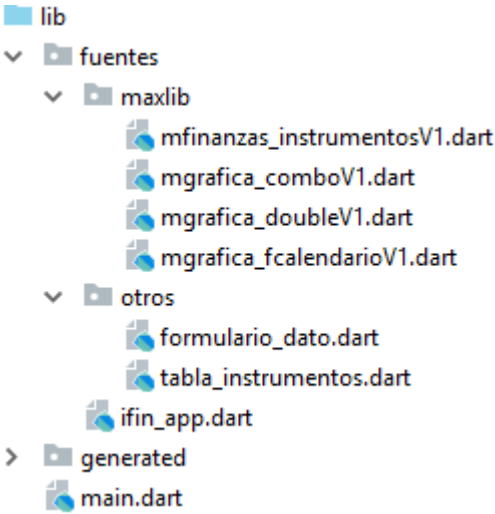
```
bool terminoTraidada=false;

Future<bool> traeInstrumentos() async {
  try {
    vIF = await Future.wait([
      valorInstrumentoFinanciero(
        TipoInstrumentoFinanciero.Uf, new DateTime.now()),
      valorInstrumentoFinanciero(
        TipoInstrumentoFinanciero.Dolar, new DateTime.now()),
      valorInstrumentoFinanciero(
        TipoInstrumentoFinanciero.Euro, new DateTime.now())
    ]);
  } catch (e) {
    print(e.toString());
  }
  return true;
}

@override
void initState() {
  super.initState();
  traeInstrumentos().then((rta) {
    setState(() {
      terminoTraidada=rta;
    });
  });
}

@override
Widget build(BuildContext context) {
  if(!terminoTraidada){
    //Poner algo apropiado por que faltan los datos
  } else {
    //llegaron los datos
  }
}
```

La estructura de los fuentes

	<p>Adopto la estructura de library ifin; y de part of ifin;</p> <p>hay un par de formas de manejar los fuentes pero esta es la más cómoda a mi gusto</p> <p>main fuentes principal</p>
---	--

De esta forma el principal tiene el siguiente contenido

```
library ifin;                                -->Defino la libreria

import 'dart:async';                         -->Librerias de flutter
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:http/http.dart' as http;
import 'package:intl/intl.dart';
import 'package:validator/validator.dart' as val;
import 'dart:convert';

part 'maxlib/mgrafica_fcalendarioV1.dart';   --> mis librerias utiles
part 'maxlib/mgrafica_doubleV1.dart';
part 'maxlib/mgrafica_comboV1.dart';
part 'maxlib/mfinanzas_instrumentosV1.dart';

part 'otros/formulario_dato.dart';           --> el código del proyecto
part 'otros/tabla_instrumentos.dart';
```

en cada uno de los otros archivos subordinados va **part of** ifin;