Nama : Windi Risma Wati

NIM : 191011401795

UAS : Mobile Programming

1. Apa Yang Di maksud dengan State Management pada Flutter ?

**State Management** adalah sebuah cara untuk mengatur data / **state** kita bekerja, bisa juga untuk memisahkan antara logic dan view dimana logic tersebut juga bisa re-usable.

2. Sebutkan Apa saja State management pada flutter !

Flutter stateful dan stateless widget

3. maaf pak sya kirimkan codenya.. source codenya sya ga masukin ke github krn laptop sya kurang mendukung, sudah cba run tapi tidak bisa tampil datanya

import 'dart:convert';

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

void main() {

runApp(const MyApp());

}

class MyApp extends StatelessWidget {

const MyApp({Key? key}) : super(key: key);

// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Flutter Demo',

theme: ThemeData(

// This is the theme of your application.

//

// Try running your application with "flutter run". You'll see the

// application has a blue toolbar. Then, without quitting the app, try

// changing the primarySwatch below to Colors.green and then invoke

// "hot reload" (press "r" in the console where you ran "flutter run",

// or simply save your changes to "hot reload" in a Flutter IDE).

// Notice that the counter didn't reset back to zero; the application

// is not restarted.

primarySwatch: Colors.blue,

),

home: const MyHomePage(title: 'Flutter Demo Home Page'),

);

}

}

class MyHomePage extends StatefulWidget {

const MyHomePage({Key? key, required this.title}) : super(key: key);

// This widget is the home page of your application. It is stateful, meaning

// that it has a State object (defined below) that contains fields that affect

// how it looks.

// This class is the configuration for the state. It holds the values (in this

// case the title) provided by the parent (in this case the App widget) and

// used by the build method of the State. Fields in a Widget subclass are

// always marked "final".

final String title;

@override

State<MyHomePage> createState() => \_MyHomePageState();

}

class \_MyHomePageState extends State<MyHomePage> with TickerProviderStateMixin {

late Future<List<Show>> shows;

String searchString = "";

late AnimationController controller;

@override

void initState() {

super.initState();

shows = fetchShows();

controller = AnimationController(

vsync: this,

duration: const Duration(seconds: 5),

)..addListener(() {

setState(() {});

});

controller.repeat(reverse: true);

}

@override

void dispose() {

controller.dispose();

super.dispose();

}

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'List API',

debugShowCheckedModeBanner: false,

home: Scaffold(

appBar: AppBar(

title: const Text('List API'),

),

body: Column(

mainAxisSize: MainAxisSize.max,

mainAxisAlignment: MainAxisAlignment.start,

crossAxisAlignment: CrossAxisAlignment.start,

children: [

const SizedBox(height: 10),

Padding(

padding: const EdgeInsets.symmetric(horizontal: 15.0),

child: TextField(

onChanged: (value) {

setState(() {

searchString = value.toLowerCase();

});

},

decoration: const InputDecoration(

labelText: 'Cari by id', suffixIcon: Icon(Icons.search)),

),

),

const SizedBox(height: 10),

Expanded(

child: FutureBuilder(

future: shows,

builder: (context, AsyncSnapshot<List<Show>> snapshot) {

if (snapshot.hasData) {

return Center(

child: ListView.separated(

itemCount: snapshot.data!.length,

itemBuilder: (BuildContext context, int index) {

return snapshot.data![index].id.toString().

contains(searchString)

? Container(

decoration: BoxDecoration(

color: Colors.white,

borderRadius: BorderRadius.circular(

5), //border corner radius

boxShadow: [

BoxShadow(

color: Colors.grey

.withOpacity(0.5), //color of shadow

spreadRadius: 1, //spread radius

blurRadius: 4, // blur radius

offset: const Offset(

0, 2), // changes position of shadow

),

//you can set more BoxShadow() here

],

),

margin: const EdgeInsets.only(

top: 5, left: 15, right: 15, bottom: 5),

// color: Colors.amber,

child: ListTile(

title: Text(

'UserId : ${snapshot.data?[index].userId}',

style: const TextStyle(

fontWeight: FontWeight.bold),

),

subtitle: Column(

mainAxisAlignment:

MainAxisAlignment.spaceBetween,

mainAxisSize: MainAxisSize.max,

crossAxisAlignment:

CrossAxisAlignment.start,

children: [

Container(

margin: const EdgeInsets.only(top: 5),

),

Row(

mainAxisAlignment:

MainAxisAlignment.spaceBetween,

children: [

Flexible(

child: Text(

'id : ${snapshot.data?[index].id}',

style: const TextStyle(

fontSize: 16,

color: Colors.black),

overflow: TextOverflow.ellipsis,

),

),

],

),

Row(

mainAxisAlignment:

MainAxisAlignment.spaceBetween,

children: [

Flexible(

child: Text(

'judul : ${snapshot.data?[index].title}',

style: const TextStyle(

fontSize: 16,

color: Colors.black),

overflow: TextOverflow.ellipsis,

),

),

],

),

Row(

mainAxisAlignment:

MainAxisAlignment.spaceBetween,

children: [

Flexible(

child: Text(

'body : ${snapshot.data?[index].body}',

style: const TextStyle(

fontSize: 16,

color: Colors.black),

overflow: TextOverflow.ellipsis,

),

),

],

),

],

),

onTap: () {

},

),

)

: Container();

},

separatorBuilder: (BuildContext context, int index) {

return snapshot.data![index].id.toString().

contains(searchString)

? Container(

margin: const EdgeInsets.all(0),

)

: Container();

},

),

);

} else if (snapshot.hasError) {

print(snapshot.error);

return const Center(child: Text('Something went wrong :('));

}

return Padding(

padding: const EdgeInsets.all(20.0),

child: Row(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

Text(

'Harap Menunggu ...',

style: Theme.of(context).textTheme.headline6,

),

CircularProgressIndicator(

value: controller.value,

semanticsLabel: 'Linear progress indicator',

),

],

),

]));

},

),

),

],

),

),

);

}

}

class Show {

final int userId;

final int id;

final String title;

final String body;

Show({

required this.userId,

required this.id,

required this.title,

required this.body

});

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

return Show(

userId: json['userId'] ?? 0,

id: json['id'] ?? 0,

title: json['title'] ?? '',

body: json['body'] ?? '',

);

}

}

Future<List<Show>> fetchShows() async {

final response = await http.get(

Uri.parse('https://jsonplaceholder.typicode.com/posts'),

);

if (response.statusCode == 200) {

var topShowsJson = jsonDecode(response.body) as List;

return topShowsJson.map((show) => Show.fromJson(show)).toList();

} else {

throw Exception('Failed to load');

}

}