'findAppBar=find.byType(AppBar)' adalah mencari widget AppBar dalam tree widget yang sedang diuji menggunakan find.byType()

Hasilnya akan disimpan dalam variabel 'findAppBar' untuk digunakan selanjutnya dalam pengujian

‘tester.widget' digunakan untuk mengambil instance widget AppBar yang ditemukan berdasarkan 'findAppBar’

```dart
import 'package:flutter/material.dart';
import 'package:flutter_projectsection16_2/main.dart';
import 'package:flutter_test/flutter_test.dart';


void main() {
  testWidgets('Test UI di HomePage2', (WidgetTester tester)async{
    await tester.pumpWidget( const MaterialApp(
      home: HomePage2(),
    ));

//Test app bar title text
    expect(find.text('Contacts'), findsOneWidget);

//Test app bar leading icon
    expect(find.descendant(
    of: find.byType(AppBar),
    matching: find.byIcon(Icons.perm_contact_calendar_rounded),
    ),
    findsOneWidget);

//Test app bar background color
//'findAppBar=find.byType(AppBar)' adalah mencari widget AppBar dalam tree widget yang sedang diuji menggunaka
n find.byType()
//Hasilnya akan disimpan dalam variabel 'findAppBar' untuk digunakan selanjutnya dalam pengujian
//'tester.widget' digunakan untuk mengambil instance widget AppBar yang ditemukan berdasarkan 'findAppBar'
    final findAppBar=find.byType(AppBar);
    final appBarWidget = tester.widget<AppBar>(findAppBar);
    expect(appBarWidget.backgroundColor, equals(Colors.blueAccent));

//Test "Halaman Baru Yang Berisi Contacts" text
    expect(find.text('Halaman Baru Yang Berisi Contacts'), findsOneWidget);

//Test icon widget
    expect(find.descendant(
    of: find.byType(SingleChildScrollView),
    matching: find.byIcon(Icons.perm_contact_calendar_rounded),
    ),
    findsOneWidget);

 //Test preferred size widget
    expect(find.byType(PreferredSize), findsOneWidget);

//Test "Create New Contacts" text
    expect(find.text('Create New Contacts'), findsOneWidget);



//Test container text
    expect(find.text(
'A dialog is a type of modal window that appears in front of app content to provide critical information, or p
rompt for a decision to be made.'
), findsOneWidget);
```

'bool isNavigatorPopped =false' adalah mendeklarasi variabel boolean 'isNavigatorPopped' dengan nilai awal false

'tester.tap' adalah untuk mensimulasikan sebuah tap(klik) pada FloatingActionButton

'tester.pumpAndSettle' adalah untuk menunggu hingga semua perubahan pada widget tree selesai diproses

Dalam kasus ini, kita tunggu hingga proses eksekusi onPressed callback selesai

'isNavigatorPopped = true' diubah menjadi true untuk menandakan bahwa onPressed callback telah dieksekusi

Kita menggunakan expect untuk memeriksa apakah onPressed callback dari FloatingActionButton telah dieksekusi dan menghasilkan nilai true pada variabel 'isNavigatorPopped'

Jika tidak, maka pesan error yang diberikan adalah 'FloatingActionButton onPressed callback not executed'

'reason' akan automatis ditampilkan jika assertion 'expect(isNavigatorPopped, isTrue)' gagal

assertion adalah sebuah mekanisme dalam pengujian perangkat lunak yang digunakan untuk memverifikasi suatu kondisi benar atau salah

Kode dibawah adalah untuk memverfikasi widget 'HomePage2' sudah di pop atau dihapus dari Navigator setelah fungsi 'Navigator.pop()'

'find.byType(HomePage2)' digunakan untuk mencari HomePage2 dalam hirarki widget yang ada di dalam tester

'findsNothing' digunakan untuk memastikan widget HomePage2 tidak ditemukan dalam hirarki

Jika ditemukan maka akan tampil 'reason'

```dart
//FloatingActionButton
    final findFloatingActionButton= find.byType(FloatingActionButton);

//Test FloatingActionButton onPressed callback
//'bool isNavigatorPopped =false' adalah mendeklarasi variabel boolean 'isNavigatorPopped' dengan nilai awal f
alse
//'tester.tap' adalah untuk mensimulasikan sebuah tap(klik) pada FloatingActionButton
//'tester.pumpAndSettle' adalah untuk menunggu hingga semua perubahan pada widget tree selesai diproses
//Dalam kasus ini, kita tunggu hingga proses eksekusi onPressed callback selesai
//'isNavigatorPopped = true' diubah menjadi true untuk menandakan bahwa onPressed callback telah dieksekusi
    bool isNavigatorPopped =false;
    await tester.tap(findFloatingActionButton);
    await tester.pumpAndSettle();
    isNavigatorPopped = true;

//Kita menggunakan expect untuk memeriksa apakah onPressed callback dari FloatingActionButton telah dieksekusi
dan menghasilkan nilai true pada variabel 'isNavigatorPopped'
 //Jika tidak, maka pesan error yang diberikan adalah 'FloatingActionButton onPressed callback not executed'
//'reason' akan automatis ditampilkan jika assertion 'expect(isNavigatorPopped, isTrue)' gagal
//assertion adalah sebuah mekanisme dalam pengujian perangkat lunak yang digunakan untuk memverifikasi suatu k
ondisi benar atau salah
    expect(isNavigatorPopped, isTrue, reason: 'FloatingActionButton onPressed callback not executed');

//Test Navigator.pop()
//Kode dibawah adalah untuk memverfikasi widget 'HomePage2' sudah di pop atau dihapus dari Navigator setelah f
ungsi 'Navigator.pop()'
//'find.byType(HomePage2)' digunakan untuk mencari HomePage2 dalam hirarki widget yang ada di dalam tester
//'findsNothing' digunakan untuk memastikan widget HomePage2 tidak ditemukan dalam hirarki widget setelah Navi
gator.pop
//Jika ditemukan maka akan tampil 'reason'
    expect(find.byType(HomePage2), findsNothing, reason: 'HomePage2 widget not popped from Navigator');
  });
```

```
testWidgets('Cek Container, SizedBox, dan Padding HomePage2', (WidgetTester tester)async{
   await tester.pumpWidget(MaterialApp(
     home: Scaffold(
       body:SingleChildScrollView(
       child: Padding(
         padding: const EdgeInsets.fromLTRB(16.0, 16.0, 16.0, 0),
         child: Column(
           mainAxisAlignment: MainAxisAlignment.start,
           children: [

             const SizedBox(height: 10.0),

             const SizedBox(height: 10.0),

               Container(
                 decoration: const BoxDecoration(
                   border: Border(
                     bottom: BorderSide(
                     color: Colors.grey,
                     width: 1,
                     ),
                   ),
                  ),



               child: const Padding(
                 padding: EdgeInsets.all(10),
                 child: Text(''
                 ),
               ),

              ),
            ],
          ),
         ),
     ),
     ),
   ));


 //Test container border
 final findContainer = find.byType(Container);
 expect(findContainer, findsOneWidget);
```

```
// Verifikasi BoxDecoration di widget Container
  final containerWidget = tester.widget<Container>(findContainer);
  expect(containerWidget.decoration, equals(const BoxDecoration(
    border: Border(
      bottom: BorderSide(
        color: Colors.grey,
        width: 1,
        ),
      ),
    )));

  //Test Sizedbox
  final findSizedBox= find.byType(SizedBox);
  expect(findSizedBox, findsNWidgets(2));

  //Test padding
  final findPadding = find.byType(Padding);
  expect(findPadding, findsNWidgets(2));

  //Verifikasi nilai EdgeInsets dari 3 widget Padding
  final paddingWidgets = tester.widgetList<Padding>(findPadding);
  expect(paddingWidgets.elementAt(0).padding, equals(const EdgeInsets.fromLTRB(16.0, 16.0, 16.0, 0)));
// Verifikasi padding pertama
  expect(paddingWidgets.elementAt(1).padding, equals(const EdgeInsets.fromLTRB(0, 0, 0, 1)));
// Verifikasi padding kedua
  expect(paddingWidgets.elementAt(2).padding, equals(const EdgeInsets.all(10))); // Verifikasi padding ketiga
  });

}
```