



Flutter BMI CALCULATOR



Cosa Realizzeremo



Creiamo il Progetto

```
mkdir flutter_project
```

```
cd flutter_project
```

```
flutter create -e bmi_calculator
```

```
Creating project bmi_calculator...
Resolving dependencies in bmi_calculator... (1.7s)
Got dependencies in bmi_calculator.
Wrote 128 files.

All done!
You can find general documentation for Flutter at: https://docs.flutter.dev/
Detailed API documentation is available at: https://api.flutter.dev/
If you prefer video documentation, consider:
https://www.youtube.com/c/flutterdev

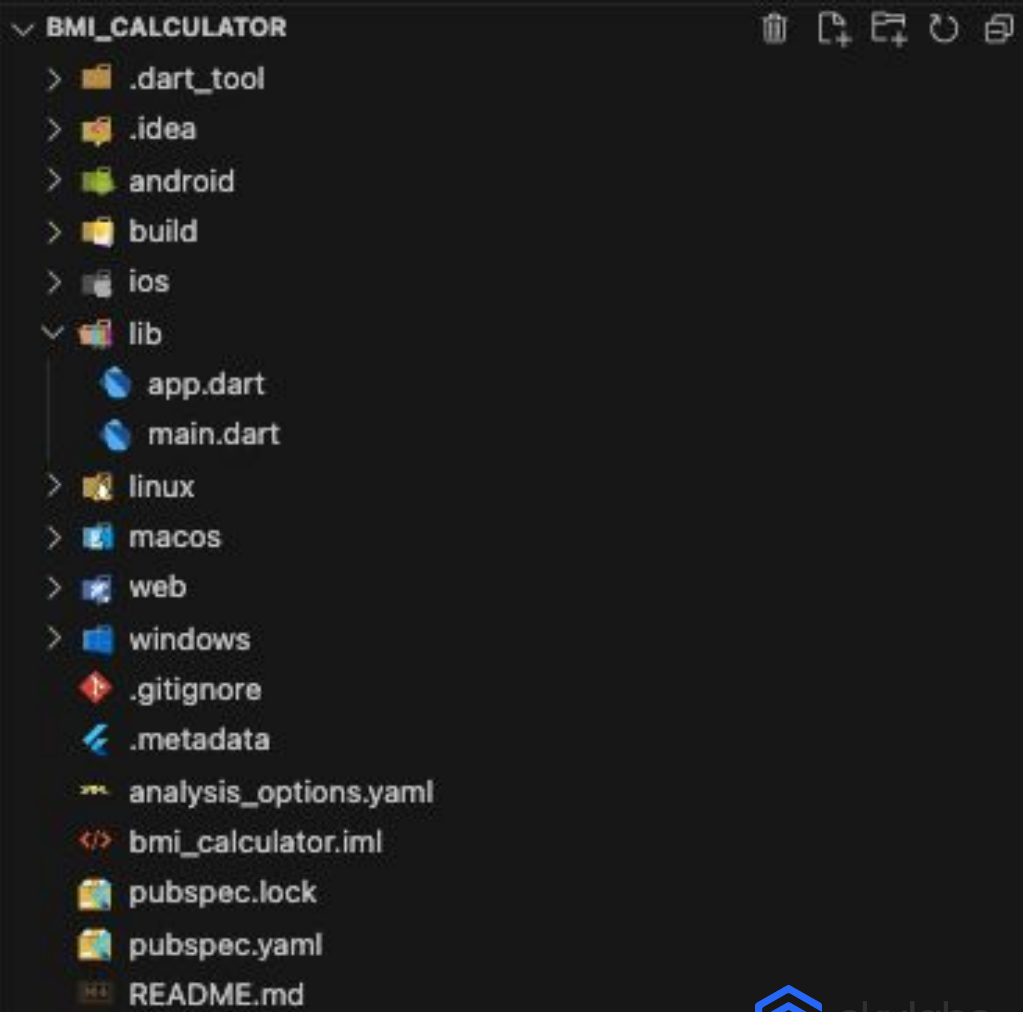
In order to run your empty application, type:

$ cd bmi_calculator
$ flutter run

Your empty application code is in bmi_calculator/lib/main.dart.
```

Struttura del progetto

Apriamo il progetto con il comando `vcode .`



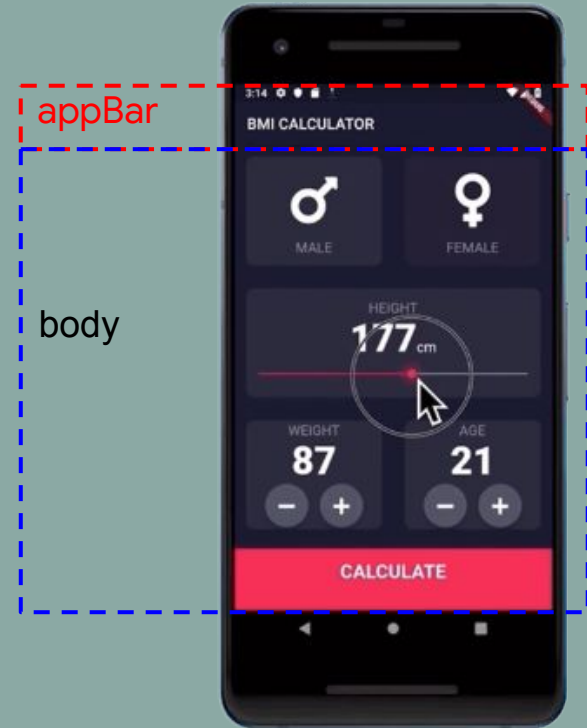
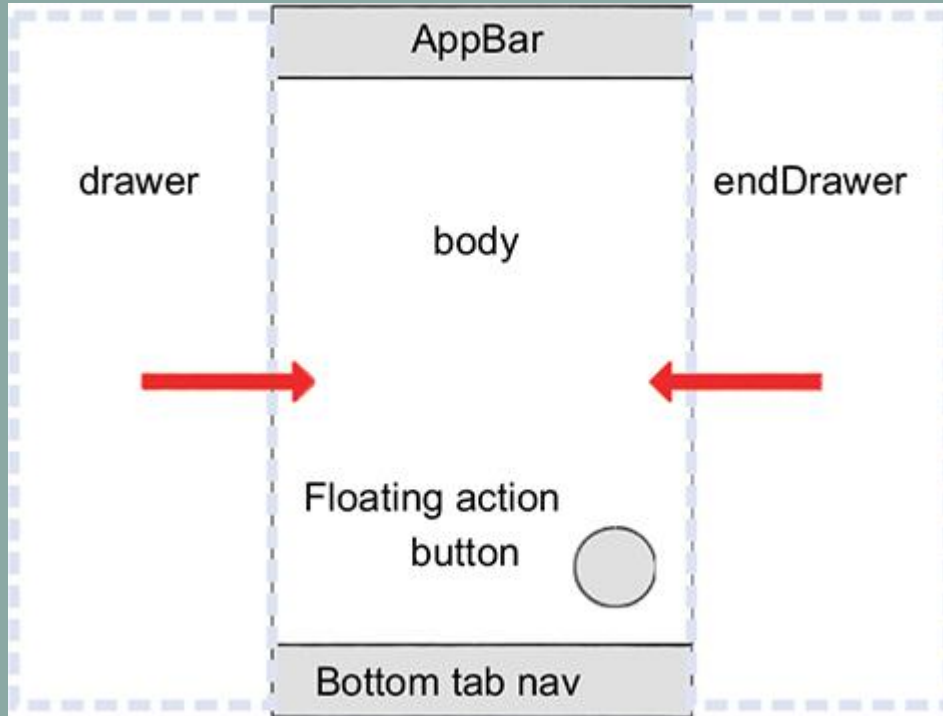
Creiamo la MaterialApp

```
import 'package:flutter/material.dart';

class App extends StatelessWidget {
  const App({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(
          seedColor: const Color(0xFF1D1E33),
          brightness: Brightness.dark,
        ),
        appBarTheme: const AppBarTheme(
          backgroundColor: Color(0xFF1D1E33),
          foregroundColor: Colors.white,
        ),
      ),
      home: Container(),
    );
  }
}
```

Lo Scaffold



Input Page

Scaffold

- AppBar
 - Text
- Body
 - Container

```
import 'package:flutter/material.dart';

class InputPage extends StatelessWidget {
  const InputPage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('BMI Calculator'),
      ),
      body: Container(
        color: Colors.white,
      ),
    );
  }
}
```

Body

Column

Row

Container

- Decoration

Expanded

SafeArea

```
import 'package:flutter/material.dart';

class InputPage extends StatelessWidget {
  const InputPage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('BMI Calculator')),
      body: SafeArea(
        child: Column(children: [
          Row(
            children: [
              Expanded(
                child: Container(
                  height: 200,
                  width: 200,
                  margin: const EdgeInsetsDirectional.all(15.0),
                  decoration: BoxDecoration(
                    borderRadius: BorderRadius.circular(10.0),
                    color: const Color(0xFF1D1E33),
                  ),
                )),
            ],
          )),
    );
  }
}
```


Riutilizziamo i Widget

- **Riduce la duplicazione del codice:**
Scrivi meno codice definendo una volta un widget e riutilizzandolo ovunque.
- **Facilità di manutenzione:** Modifica un widget in un'unica posizione e le modifiche si riflettono ovunque venga utilizzato.
- **Migliora la leggibilità del codice:** Raggruppa parti dell'interfaccia utente per renderle più chiare e comprensibili.
- **Facilita i test:**
I widget modulari sono più facili da testare, concentrando i test su singoli componenti anziché sull'intera app.
- **Aumenta la produttività:**
Costruisci nuove funzionalità rapidamente utilizzando widget già sviluppati.

```
import 'package:flutter/material.dart';

class ReusableCard extends StatelessWidget {
  const ReusableCard({
    super.key,
    required this.color,
  });
  final Color color;

  @override
  Widget build(BuildContext context) {
    return Expanded(
      child: Container(
        margin: const EdgeInsetsDirectional.all(15.0),
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(10.0),
          color: color,
        ),
      ),
    );
  }
}
```

Refactoring Code

Sostituiamo gli Expandend con le ReusableCard.

Cosa notiamo?

Quali vantaggi abbiamo?

```
import 'package:bmi_calculator/reusable_card.dart';
import 'package:flutter/material.dart';

class InputPage extends StatelessWidget {
  const InputPage({super.key});

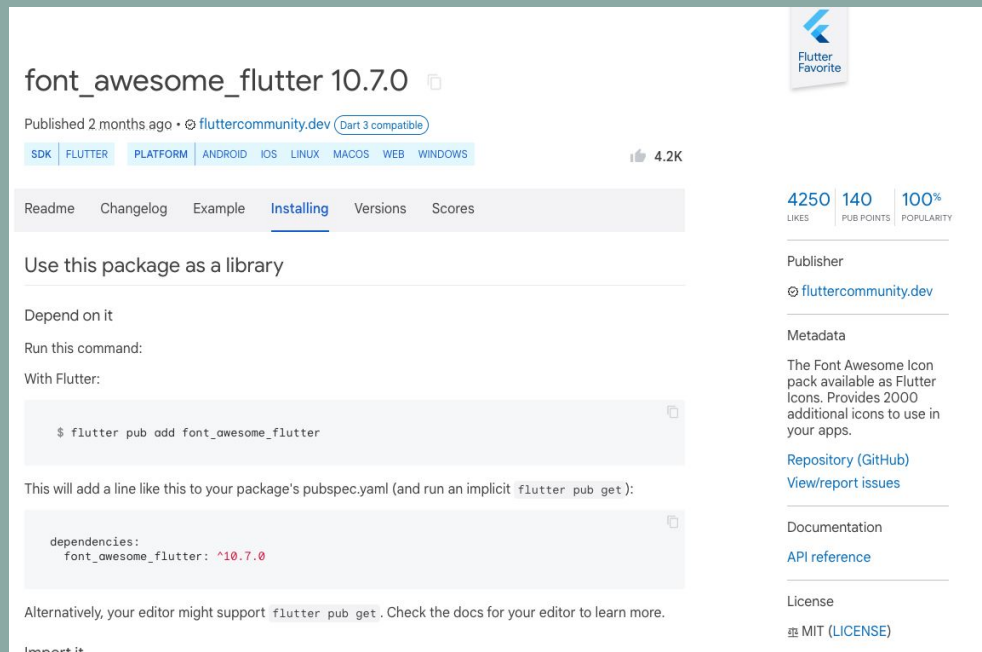
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('BMI Calculator')),
      body: const SafeArea(
        child: Column(children: [
          Row(
            children: [
              ReusableCard(color: Color(0xFF1D1E33)),
              ReusableCard(color: Color(0xFF1D1E33)),
            ],
          ),
          ReusableCard(color: Color(0xFF1D1E33)),
          Row(
            children: [
              ReusableCard(color: Color(0xFF1D1E33)),
              ReusableCard(color: Color(0xFF1D1E33)),
            ],
          ),
        ]
      ),
    );
  }
}
```

Installiamo Pacchetti Aggiuntivi

I pacchetti di pub.dev semplificano lo sviluppo, migliorano le funzionalità delle tue applicazioni e favoriscono la collaborazione e lo scambio di conoscenze all'interno della comunità degli sviluppatori.

Come installiamo un pacchetto?

Dal terminale di VSCode digitiamo:
`flutter pub add font_awesome_flutter`



The screenshot shows the pub.dev page for the `font_awesome_flutter` package, version 10.7.0. The page is marked as a "Flutter Favorite". It includes a navigation bar with tabs for SDK, FLUTTER, PLATFORM, ANDROID, IOS, LINUX, MACOS, WEB, and WINDOWS. The "Installing" tab is active. The page provides instructions on how to use the package as a library, how to depend on it, and the command to run: `$ flutter pub add font_awesome_flutter`. It also shows the resulting entry in the `pubspec.yaml` file: `dependencies: font_awesome_flutter: ^10.7.0`. The right sidebar displays statistics: 4250 likes, 140 pub points, and 100% popularity. It also lists the publisher as `fluttercommunity.dev` and provides links to the repository (GitHub), documentation, and license (MIT).

font_awesome_flutter 10.7.0

Published 2 months ago • @ fluttercommunity.dev (Dart 3 compatible)

SDK | FLUTTER | PLATFORM | ANDROID | IOS | LINUX | MACOS | WEB | WINDOWS

4.2K

Readme | Changelog | Example | **Installing** | Versions | Scores

Use this package as a library

Depend on it

Run this command:

With Flutter:

```
$ flutter pub add font_awesome_flutter
```

This will add a line like this to your package's pubspec.yaml (and run an implicit `flutter pub get`):

```
dependencies:
  font_awesome_flutter: ^10.7.0
```

Alternatively, your editor might support `flutter pub get`. Check the docs for your editor to learn more.

Import it

Flutter Favorite

4250 140 100%

LIKES PUB POINTS POPULARITY

Publisher

@ fluttercommunity.dev

Metadata

The Font Awesome Icon pack available as Flutter Icons. Provides 2000 additional icons to use in your apps.

Repository (GitHub)

View/report issues

Documentation

API reference

License

MIT (LICENSE)

Sex Selector

- Column
 - mainAxisAlignment
- FontAwesomeIcons
- SizedBox
- Text
 - textStyle

Possiamo ottimizzarlo? Come?

```
ReusableCard(  
  color: Color(0xFF1D1E33),  
  child: Column(  
    mainAxisAlignment: MainAxisAlignment.center,  
    children: [  
      Icon(  
        FontAwesomeIcons.mars,  
        size: 80.0,  
      ),  
      SizedBox(height: 15.0),  
      Text(  
        'Maschio',  
        style: TextStyle(  
          fontSize: 18.0,  
          color: Color(0xFF8d8E98),  
        ),  
      ),  
    ],  
  )),
```

Icon Content Widget

Creiamo un widget per eliminare il codice boilerplate.

Cos'è il codice boilerplate?

```
import 'package:flutter/material.dart';

class IconContent extends StatelessWidget {
  const IconContent({
    super.key,
    required this.label, required this.icon,
  });

  final String label;
  final IconData icon;

  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Icon(icon, size: 80.0,),
        const SizedBox(height: 15.0),
        Text(
          label,
          style: const TextStyle(
            fontSize: 18.0,
            color: Color(0xFF8d8E98),),),
      ],);
  }
```

Constants

Creiamo una classe per gestire le costanti.

```
class Constants {  
    static const String appName = 'BMI Calculator';  
    static const Color activeCardColor = Color(0xFF1D1E33);  
    static const Color inactiveCardColor = Color(0xFF111328);  
    static const Color activeSliderColor = Color(0xFFEB1555);  
    static const Color inactiveSliderColor = Color(0xFF8D8E98);  
    static const Color buttonColor = Color(0xFFEB1555);  
    static const double heightContainer = 80.0;  
    static const double heightButton = 80.0;  
    static const TextStyle labelTextStyle = TextStyle(fontSize: 18.0, color: Color(0xFF8d8E98));  
    static const TextStyle numberTextStyle = TextStyle(fontSize: 50.0, fontWeight: FontWeight.w900);  
}
```

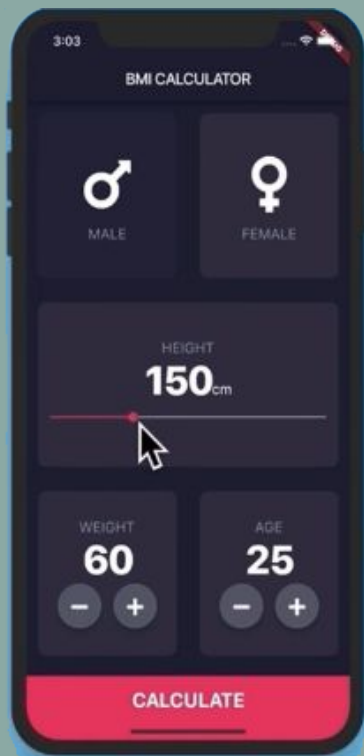
Sostituiamo tutte le occorrenze.

Bottom Button Widget

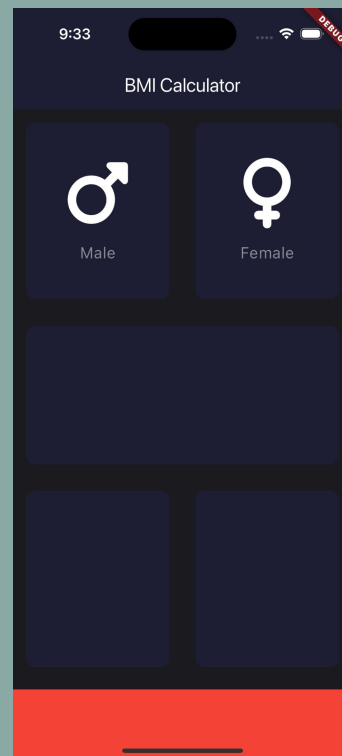
```
Container(  
  color: Colors.red,  
  margin: const EdgeInsetsDirectional.only(top: 10.0),  
  height: 80.0,  
  width: double.infinity,  
),
```

Che modifiche dobbiamo fare per allinearci al mockup?

Facciamo il punto della situazione



Mockup



Sviluppato

Il GestureDetector serve a trasformare un widget comune in un widget interattivo, consentendo all'utente di interagire con esso tramite gesti.

Ecco alcuni dei gesti più comuni che il GestureDetector può rilevare e gestire:

1. Tap: Rileva un singolo tocco su un widget.
2. Double tap: Rileva un doppio tocco veloce su un widget.
3. Long press: Rileva un tocco prolungato su un widget.
4. Vertical drag: Rileva uno swipe verticale su un widget.
5. Horizontal drag: Rileva uno swipe orizzontale su un widget.
6. Pan: Rileva il trascinamento (panning) su un widget.
7. Scale: Rileva lo zoom in/out su un widget.

Puoi associare un handler (funzione) a ciascun tipo di gesto per eseguire azioni specifiche quando quel gesto viene rilevato dall'utente.

```
class ReusableCard extends StatelessWidget {  
  const ReusableCard({  
    super.key,  
    required this.color,  
    required this.child,  
    this.onTap,  
  });  
  final Color color;  
  final Widget child;  
  final Function()? onTap;  
  
  @override  
  Widget build(BuildContext context) {  
    return Expanded(  
      child: GestureDetector(  
        onTap: onTap,  
        child: Container(  
          ...  
        ),  
      ),  
    );  
  }  
}
```

Implementiamolo

```
ReusableCard(  
  color: Constants.activeCardColor,  
  child: const IconContent(  
    label: 'Male',  
    icon: FontAwesomeIcons.mars,  
  ),  
  onTap: () {  
    print('Hai Premuto Male');  
  },  
),
```

Stateless VS Statefull

- **StatelessWidget:** Questo tipo di widget è immutabile, il che significa che una volta che viene costruito, non può cambiare i suoi attributi o lo stato interno. È ideale per rappresentare parti dell'interfaccia utente che non cambiano nel tempo, come un'icona o un testo statico.
- **StatefulWidget:** Questo tipo di widget è mutabile e può modificare il suo stato interno durante il tempo di esecuzione. Ha associato un oggetto State che contiene i dati che possono cambiare nel tempo. È adatto per rappresentare parti dell'interfaccia utente che devono essere aggiornate dinamicamente, come un elenco di elementi o un campo di input.

In breve, la differenza principale tra i due è che **StatelessWidget** è statico e non può cambiare nel tempo, mentre **StatefulWidget** è dinamico e può cambiare il suo stato durante l'esecuzione del programma.

Operatore Ternario

```
void updateColour(int gender) {  
    if (gender == 1) {  
        if (maleCardColour == inactiveCardColour) {  
            maleCardColour = activeCardColor;  
            print("male pressed");  
        } else {  
            maleCardColour = inactiveCardColour;  
            print(maleCardColour);  
        }  
    }  
}  
  
if (gender == 0) {  
    if (femaleCardColour == inactiveCardColour) {  
        femaleCardColour = activeCardColor;  
        print("female pressed");  
    } else {  
        femaleCardColour == inactiveCardColour;  
    }  
}  
}
```

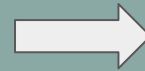


```
//Operatore ternario  
void updateColour(int gender) {  
    setState(() {  
        isMaleSelected = gender == 1 ? true : false;  
    });  
}  
  
//Versione semplificata  
void updateColour(int gender) {  
    setState(() {  
        isMaleSelected = gender == 1;  
    });  
}
```

ReusableCard

```
ReusableCard(  
  color: isMaleSelected == true  
    ? Constants.activeCardColor  
    : Constants.inactiveCardColor,  
  child: const IconContent(  
    label: 'Male',  
    icon: FontAwesomeIcons.mars,  
  ),  
  onTap: () => updateColour(1),  
),
```

```
ReusableCard(  
  color: Constants.activeCardColor,  
  child: Column(  
    mainAxisAlignment: MainAxisAlignment.center,  
    children: [  
      const Text('Height', style: Constants.labelTextStyle,  
    ),  
      const Row(  
        mainAxisAlignment: MainAxisAlignment.center,  
        crossAxisAlignment: CrossAxisAlignment.baseline,  
        textBaseline: TextBaseline.alphabetic,  
        children: [  
          Text('180', style: Constants.numberTextStyle,),  
          Text('cm', style: Constants.labelTextStyle,),  
        ],  
      ),  
      Slider(value: 1, onChanged: (value) {}),  
    ],),),
```



Refactoring Slider

```
class SliderContent extends StatelessWidget {  
  const SliderContent({super.key, required this.value, this.onChanged,});  
  final double value;  
  final Function(double)? onChanged;  
  @override  
  Widget build(BuildContext context) => Column(mainAxisAlignment: MainAxisAlignment.center,  
    children: [  
      const Text(...),  
      Row(...  
        children: [  
          Text(value.toStringAsFixed(0), style: Constants.numberTextStyle, ),  
          const Text('cm', style: Constants.labelTextStyle, )], ),  
      Slider(min: 120, max: 220, activeColor: Constants.activeSliderColor,  
        inactiveColor: Constants.inactiveSliderColor, value: value,  
        onChanged: onChanged, ),  
    ], );  
}
```

Custom Button

```
import 'package:bmi_calculator/constants.dart';
import 'package:flutter/material.dart';

class CustomButton extends StatelessWidget {
  const CustomButton({super.key, required this.child, this.onPressed,});
  final Widget child;
  final Function()? onPressed;

  @override
  Widget build(BuildContext context) {
    return RawMaterialButton(
      elevation: 6.0,
      constraints: const BoxConstraints.tightFor(width: 56.0,height: 56.0,),
      shape: const CircleBorder(),
      fillColor: Constants.floatingButtonColor,
      onPressed: onPressed,
      child: child,
    );
  }
}
```


Il Navigator è un componente fondamentale per gestire la navigazione tra diverse schermate all'interno di un'app. Esso gestisce uno stack di "route" (percorsi) che rappresentano le diverse schermate dell'app.

Il Navigator offre metodi per aggiungere nuove route allo stack (come push), rimuovere route (come pop), sostituire route esistenti e altro ancora.

È uno strumento potente per gestire la navigazione dell'applicazione in modo flessibile e intuitivo.

```
Navigator.push(  
  context,  
  MaterialPageRoute(  
    builder: (context) => ResultPage(  
      resultText: calc.calculateBMI(),  
      bmiResult: calc.getResult(),  
      descriptionText: calc.getDescription(),  
    ),  
  ),  
);
```

```
import 'dart:math';

class Calculator {

  Calculator({required this.height, required this.weight,});

  final int height; final int weight;

  double _bmi = 0;

  String calculateBMI() {_bmi = weight / pow(height / 100, 2); return _bmi.toStringAsFixed(1);}

  String getResult() {
    if (_bmi < 18.5) {return 'Sottopeso';
    } else if (_bmi >= 18.5 && _bmi < 25) {return 'Intervallo normale';
    } else if (_bmi >= 25 && _bmi < 30) {return 'Obeso';
    }
  }

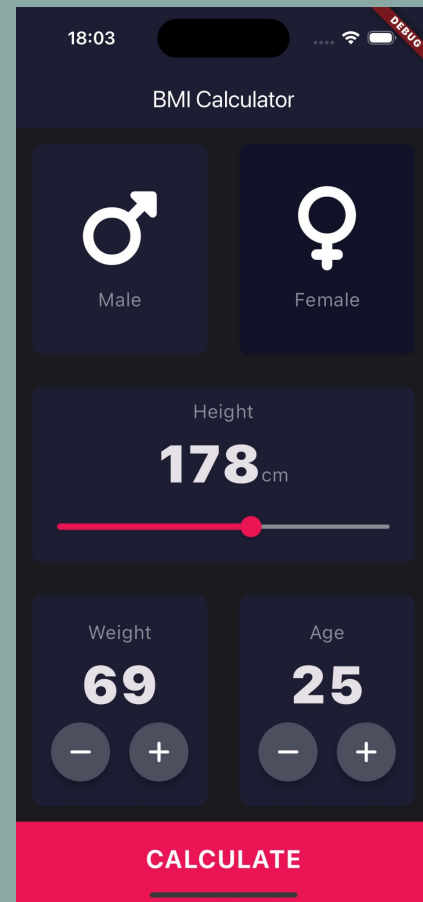
  String getDescription() {
    if (_bmi < 18.5) {return 'Hai un peso corporeo inferiore alla media. Puoi mangiare un po\' di più';
    } else if (_bmi >= 18.5 && _bmi < 25) {return 'Hai un peso corporeo normale. Ottimo lavoro';
    } else {return 'Hai un peso corporeo superiore alla media. Prova a fare più esercizio';}}
  }
}
```

Creiamo APK

Creiamo il nostro apk digitando

```
flutter build apk --release
```

Proviamo ad installarlo sul nostro device



Grazie per l'attenzione

Domande?

Contatti



Web: www.mariorefetto.it



Web: www.skylabs.it