Getting started guide för Flutter

Flutter är ett ramverk för utveckla appar för mobil, webb och desktop från en kodbas.

Layout

I Flutter bygger man upp sitt gränssnitt med nästlade *widgets*. En *widget* är en komponent i gränssnittet med ett specifikt utseende eller funktionalitet som i sin tur kan innehålla andra *widgets*.
Varje *widget* har en **build()**-metod som beskriver hur *widgeten* ska ritas i gränssnittet. I **build**-metoden kan en utvecklare lägga till både egenskapade *widgets* och *widgets* försedda av ramverket. Det finns olika typer av *widgets* försedda av Flutter. Det finns *widgets* för att placera andra *widgets* i gränssnittet, till exempel **Row**, **Column** och **Center**. Det finns *widgets* för att visa grafik, till exempel **Text**, **Image** och **Icon**. Det finns också *widgets* för interaktion med användaren som **RaisedButton**, **TextFormField** och **CheckBox**.

Interaktion

Ett vanligt sätt att agera på event från användaren är genom *callbacks*. Ett typiskt exempel på en *callback* är parametern **onPressed**, en lambda-funktion, i **RaisedButton**. Denna funktion kommer att kallas när användaren trycker på knappen. Detta designmönster kan och borde appliceras även på egna *widgets*.

```
RaisedButton(
    onPressed: () { print('I was pressed!'); },
    child: Text('Press me!'),
);
```

Det finns också ett mer generellt sätt lyssna på events från en användare även för *widgets* som inte har någon *callback*. Det är genom att använda en **GestureDetector** i eller runt en *widget*. En **GestureDetector** har flera olika parametrar för *callbacks* som till exempel **onLongPress**, **onDoubleTap** och **onHorizontalDragEnd**.

```
GestureDetector(
    onDoubleTap: () { print('I was double tapped!'); },
    child: MyAwesomeWidget(),
);
```

Navigering

För att navigera mellan skärmar i Flutter används klassen **Navigator**. Varje app har ett **Navigator** objekt som man kan få tillgång till genom nuvarande **BuildContext** som är tillgängligt i **build**-metoden. På **Navigator** kan flera metoder kallas för att hantera navigering där de vanligaste är **push** och **pop**. Man kan tänka att en **Navigator** innehåller en stack och

när **push** kallas med en ny skärm som argument läggs den längst upp på stacken. För att sedan ta bort denna skärm kallas **pop** och då visas istället skärmen som var näst längst upp i stacken.
Det finns varianter av **push** och **pop** för specialfall, till exempel **pushReplacement** som “pushar” en ny skärm på stacken som ersätter den tidigare och **popUntil** som “poppar” skärmar tills ett givet predikat är uppfyllt.
Argumentet som ges till push är oftast en **MaterialPageRoute** där man implementerar en **builder**-funktion som skapar skärmen som ska pushas. Det är också möjligt att använda **pushNamed** som fungerar precis likadant som **push** men argumentet är istället en sträng som representerar en *route* till nya skärmen. För att det ska fungera måste dessa *routes* specificeras i parametern **routes** i app-objektet (oftast **MaterialApp**) för att de ska kunna kopplas till en skärm.

Referenser

- https://flutter.dev/docs/development/ui
- https://api.flutter.dev/flutter/widgets/GestureDetector-class.html
- https://api.flutter.dev/flutter/widgets/Navigator-class.html