

# Flutter e Dart - Le basi

Flutter - Applicazioni adaptive e responsive

Luigi Durso

luigi.durso@si2001.it



**SI2001**

29 settembre 2022



# Sommario

- 1** Lezione precedente
- 2** Responsive

- 3** Adaptive
- 4** Esercitazione
- 5** Fine



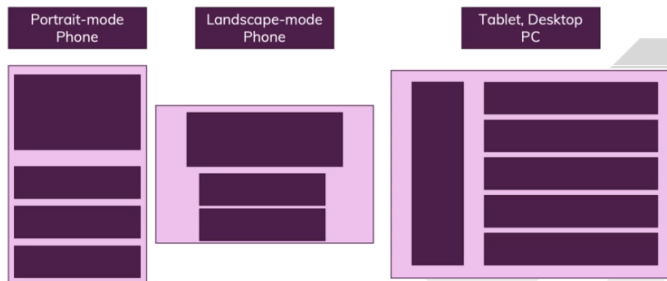
# Un po' di codice



**Analizziamo l'elaborato precedente!**



# Cosa vuol dire responsive



# Flexible Widget

**Un widget che ci è molto utile nella creazione di UI responsive è sicuramente il Flexible Widget:**

- Controlla la modalità di riempimento degli spazi dei figli;
- Da combinare con widget come Column, Row, Flex
- Si può usare il Widget **Expanded** per ottenere il risultato di un Flexible con fit tight



# Misure dinamiche

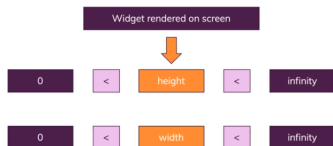
## Inserire delle misure statiche può non essere il modo giusto per la creazione di UI Responsive:

- Possiamo calcolare dinamicamente le misure assegnate ai nostri widget
- Un'ottima base per il calcolo delle misure è sicuramente lo spazio disponibile a schermo
- Recuperiamo dal contesto le misure del nostro display attraverso **MediaQuery**
- Altra informazione molto utile recuperabile da MediaQuery è **L'orientation** del dispositivo

**Attenzione!** MediaQuery richiama automaticamente la rebuild ad ogni suo cambiamento!



# Layout Builder



## Uno strumento per controllare lo spazio disponibile:

- Widget che permette di sfruttare lo spazio disponibile, spazio definito dal padre esplicitamente oppure implicitamente
- Controllo effettuato attraverso l'uso dei constraints
- Nei constraints abbiamo le info su ampiezza ed altezza disponibili
- I constraints sono implicitamente utilizzati ogni volta che definiamo delle misure, oppure utilizziamo widget che ne fanno uso. **Es. Expanded**



# Diverse piattaforme

**Essendo Flutter di natura cross-platform, abbiamo strumenti per gestire diverse piattaforme:**

- Possiamo creare widget **Adattivi** che vengono renderizzati in modo differente in base alla piattaforma in uso
- Molti widget integrano un costruttore **.adaptive** che permette di gestire il multi-piattaforma
- Per individuare la piattaforma possiamo usare la classe **Platform** di dart.io





# Cupertino widget

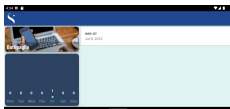
## Flutter non è solo Material Design

- Diamo una occhiata ai widget basati su uno stile "ios-centrico"

► [Documentazione](#)



# Migliorare la precedente esercitazione



## Alcuni spunti:

- Utilizzare i widget Flexible ed Expanded dove possibile,
- Sfruttare MediaQuery e Constraints per l'assegnazione delle sizes
- Disegnare in versione **Landscape** un dettaglio sulla sede corrente (uno spunto in foto)



# Grazie per l'attenzione!

