ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

Atelier 2 : Layout Widget ou Widget de mise en forme

Partie I : Exploration des widgets Container, Column, Row,

I. Le Widget Container

Le widget **Container** permet de modifier l'apparence et la mise en page d'une application. En fait, il encapsule les widgets pour leur ajouter une hauteur, une largeur, des marges, des bordures et des couleurs d'arrière-plan à un widget.

En bref, il est utilisé par de nombreux widgets pour ajouter des propriétés de style.

- Si vous envelopper votre widget dans container sans utiliser de paramètre, vous n'allez apercevoir aucune différence.

```
class MyWidget extends StatelessWidget{

  Widget build(BuildContext context){
    return Scaffold(
        appBar:AppBar(
          title: Text("Test App"),
          centerTitle: true,
          backgroundColor: Colors.teal,
      ),
      body: Container(
        child: Text(" Add style to widget"),
      ),
    );
  }
}
```

- Si on veut ajouter une couleur au background du widget enfant de Container, on utilise le paramètre *color* du widget *Container* :

```
Container(
  child: Text(" Add style to widget"),
  color: Colors.blue,
),
```

- **Remarque** : Si on n'utilise pas les paramètres de redimensionnement, le *Container* prend la taille du widget qu'il enveloppe.

- On peut utiliser le paramètre ***padding*** pour ajouter de l'espace entre le Container et le widget qu'il enveloppe (i.e. dans ce cas entre le Container et le Widget Text).

```
Container(
  child: Text(" Add style to widget"),
  padding: EdgeInsets.all(30),
```

```
  color: Colors.blue,
),
```

- On peut ajouter le paramètre ***margin*** pour ajouter de l'espace autour du widget *Container* (c.-à-d. ajouter de l'espace à l'extérieur du widget *Container*).

```
-   Container(
      child: Text(" Add style to widget"),
      padding: EdgeInsets.all(30),
      margin: EdgeInsets.all(40),
      color: Colors.blue,
    ),
```

- L'**Utilisation** des propriétés **width** et **height** permet de définir la largeur et la hauteur du conteneur en utilisant les propriétés **width** et **height** du **Container**. Par exemple, **width: double.infinity** et **height: double.infinity** permettront au conteneur de s'étendre sur toute la largeur et la hauteur disponibles, respectivement.

- Essayer, la propriété ***alignment*** qui détermine comment le contenu du Container est aligné à l'intérieur du conteneur. Si le contenu est plus petit que le conteneur, l'alignment influencera où il est positionné.

- Voir la documentation officielle pour voir plus de détails et de propriété sur ce widget.

II. Arrangez plusieurs widgets de manière verticale et horizontale

L'un des modèles de mise en page les plus courants consiste à organiser des widgets verticalement ou horizontalement. Vous pouvez utiliser un widget Row pour organiser des widgets horizontalement et un widget Column pour organiser des widgets verticalement.

Pour créer une rangée (row) ou une colonne (column) en Flutter, vous ajoutez une liste de widgets enfants à un widget Row ou Column. Chaque enfant peut à son tour être une rangée ou une colonne, et ainsi de suite.

1. Aligner les Widgets d'une rangée (row)

Vous contrôlez comment une rangée (row) ou une colonne (column) aligne ses enfants en utilisant les propriétés *mainAxisAlignment* et *crossAxisAlignment*.

- Définir l'alignement de l'axe principal sur "spaceEvenly" pour diviser de manière égale l'espace horizontal disponible entre, avant et après chaque Icône.

```
Row(
    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
    children: const [
    Icon(size: 70,
          Icons.access_alarm_outlined,
          color: Colors.deepPurple,),  // Icon
      Icon(size: 70,
        Icons.account_balance,
        color: Colors.deepPurple,),  // Icon
      Icon(size: 70,
        Icons.ad_units,
        color: Colors.deepPurple,),  // Icon
  ]),  // Row
```

a. Exécuter le code pour voir le résultat retourné
b. Essayer le même code avec : ***MainAxisAlignment.start***, ***MainAxisAlignment.end, MainAxisAlignment.spaceBetween, MainAxisAlignment.center*** .
c. Pour le ***crossAxisAlignemnt*** essayer les options : **CrossAxisAlignement.start, CrossAxisAlignement.end, CrossAxisAlignement.center.**

2. Aligner les Widgets d'une colonne (Column)

```
Column(
    mainAxisAlignment: MainAxisAlignment.start,
    crossAxisAlignment: CrossAxisAlignment.start,
    children: const [
    Text("Heeeeeeey", style: TextStyle(fontSize: 30),),
      Icon(size: 70,
        Icons.account_balance,
        color: Colors.deepPurple,),  // Icon
      Icon(size: 70,
        Icons.ad_units,
        color: Colors.deepPurple,),  // Icon
  ]),  // Column
```

a. Testez le code ci-dessus
b. Essayer avec les différentes possibilités de MainAxisAlignment et CrossAxisAlignment : start, end, center …

3. Utiliser le widget **SizedBox** pour mettre de l'espace horizontalement ou verticalement entre les widgets. **SizedBox** peut avoir comme paramètre ***width*** pour spécifier l'espace à garder horizontallement et le paramètre ***height*** pour spécifier l'espace à garder verticalement.

4. A quoi sert le widget **Expanded**?

5. **Application**
   - Construire l'interface graphique flutter suivant :

Partie II : Travail à Réaliser

1. *Le widget Container*

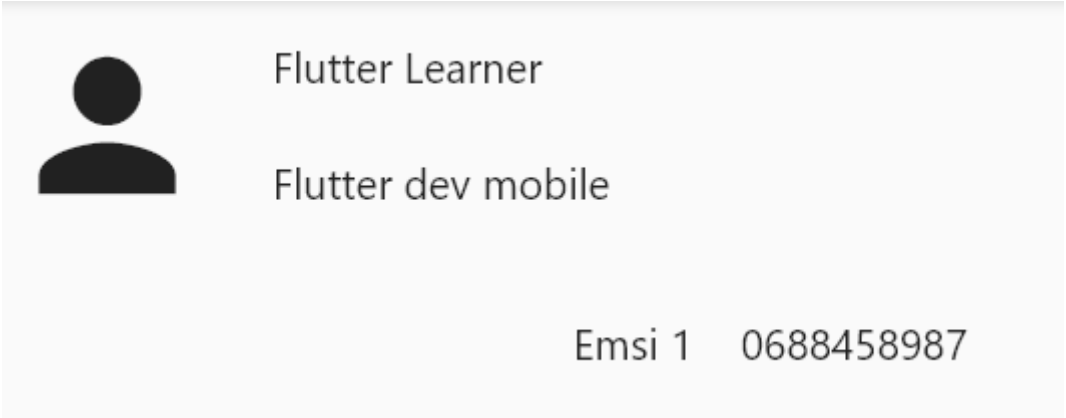
a. Essayez le widget Container, et mettre dedans un autre Widget (de type Text Par Exp)
b. Essayez les propriétés suivantes du Widget Container :
   - color
   - width and height
   - margin
   - padding
c. Qu'est-ce que vous remarquez

2. *Les Widgets Rows et Colums*

a. Utilisez le widget Column ayant comme widgets enfant plusieurs widget (soit le widget Text Par Exemple)
b. Utilisez les propriétés ' MainsAxisAlignment' et 'CrossAxisAlignment' du Wiget Column
c. Utilisez le widget Row ayant commen widgets enfant plusieurs widget (soit le widget Text Par Exemple)
d. Utilisez les propriétés ' MainsAxisAlignment' et 'CrossAxisAlignment' du Wiget Row.
e. Qu'est-ce que vous remarquez ?

3. *Application*

Construisez l'interface suivante :



4. *Le Widget Expanded*

a. Ajouter un Widget TextField
b. Qu'est-ce que vous remarquez

c. Envelopper le widget TextFiel avec e widget Expanded peut envelopper le widget et le force à remplir l'extra-space,

d. Ajoutez la propriété flex du widget Expanded

5. Autres Widget à explorer

   Notez bien : Pour la construction de vos UIs, vous pouvez avoir également d'autre widgets de mis en page, par exemple :
   - **SizedBox**
   - **ListView**
   - **SingleChildScrollView**

Essayer d'explorer ces widgets, leur utilité, leur utilisation, et pour plus de widgets, consulter la documentation officielle :

https://docs.flutter.dev/development/ui/widgets/layout