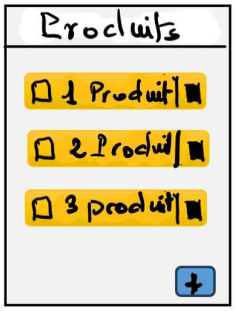
# Atelier 1: Produits App

**Objectif** : Créer une application de gestion des produits,ayant les fonctionnalités suivantes :

- Affichage de la liste des produits
- Ajout d'un produit
- Sélection d'un ou de plusieurs produits
- Suppression d'un produit
- Suppression d'une sélection de produits




## Affichage de la liste des produits

1. Créer un nouveau projet nommé « produitsapp »,en utilisant le template « Empty application »

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

Run | Debug | Profile
void main() {
  runApp(const MainApp());
}

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

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      home: Scaffold(
        body: Center(
          child: Text('Hello World!'),
        ), // Center
      ), // Scaffold
    ); // MaterialApp
  }
}
```

2. Ajouter un nouveau fichier nommé produits_list.dart
3. Ajouter un widget nommé ProduitsList.

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

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

  @override
  Widget build(BuildContext context) {
    return const Placeholder();
  }
}
```

4. Ajouter un nouveau fichier dart nommé produit_box contenant le widget Stateless ProduitBox dont le rôle est d'afficher les informations d'un produit.
5. Remplacer PlaceHolder() par Container dans ProduitBox
6. Ajouter un attribut nomProduit de type String dans la classe ProduitBox

```dart
String nomProduit;
```

7. Initialiser l'attribut dans le constructeur

```dart
ProduitBox({super.key,required  this.nomProduit});
```

8. Définir la propriété height et child du widget Container comme suit :

```dart
  return Container(
    height: 120,
    child: Text(nomProduit),
  ); // Container
}
```

9. Dans la classe ProduitsList (fichier : produits_list.dart), ajouter la déclaration d'une list de produits

```dart
// Liste des produits
List liste = [
  ["1 Produit", false],
  ["2 Produit", true],
  ["3 Produit", false],
  ["4 Produit", false],
  ["5 Produit", false]
];
```

10. Remplacer PlaceHolder() par Scaffold() et ajouter l'attribut appBar

```dart
appBar: AppBar(title: const Text("Produits")),
```

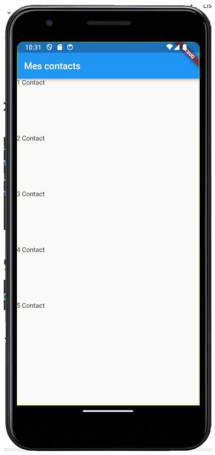
11. Dans l'attribut body de Scaffold ajouter un  widget ListView

```
body: ListView.builder(
      itemCount: liste.length,
      itemBuilder: (context, index) {
        return ProduitBox(
          nomProduit:liste[index][0],
        ); // ProduitBox
      })); // ListView.builder // Scaffold
```

12. Dans le widget MainApp, redéfinir l'attribut home comme suit :

```
home: ProduitsList(),
```

13. Exécuter et tester l'application




## Sélection des produits

Dans la classe ProduitBox nous allons ajouter un widget CheckBox pour afficher l'état de sélection d'un produit, le widget CheckBox a besoin de deux argument value et une fonction de callBack qui sera appelée lorsque l'état de la case à cocher change.

1. Ajouter dans la classe les deux attributs suivants :

```
final bool selProduit;
final Function(bool?)? onChanged;
```

2. Initilialiser selProduit et onChanged dans le constructer

```
const ProduitBox(
    {super.key,
    required this.nomProduit,
    this.selProduit = false,
    this.onChanged});
```

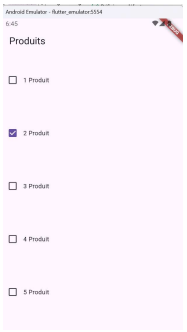
3. Encapsuler le widget Text par Row (wrap with row) et ajouter le widget Checkbox

```
child: Row(
  children: [
    Checkbox(value: selProduit, onChanged: onChanged),
    Text(nomProduit),
  ],
), // Row
```

4. Dans la classe ProduitsList, ajouter les deux nouveaux arguments du constructeur ProduitBox()

```
return ProduitBox(
  nomProduit: liste[index][0],
  selProduit: liste[index][1],
  onChanged: (value) => onChanged(value, index),
); // ProduitBox
```

Pour le moment nous avons passé au paramètre onChanged une fonction vide.




Le clic sur les cases à cocher ne produit aucun effet.

5. Convertir la classe ProduitsList en un widget statefull (utilisez l'assistant), et après dans la classe _ProduitsListeState ajouter la fonction suivante :

```
void onChanged(bool? value, int index) {
  setState(() {
    liste[index][1] = value;
  });
}
```

6. Remplacer l'attribut onchanged par :

```
return ProduitBox(
  nomProduit: liste[index][0],
  selProduit: liste[index][1],
  onChanged: (value) => onChanged(value, index),
); // ProduitBox
```

7. Tester à nouveau l'application.

## Ajout d'un produit

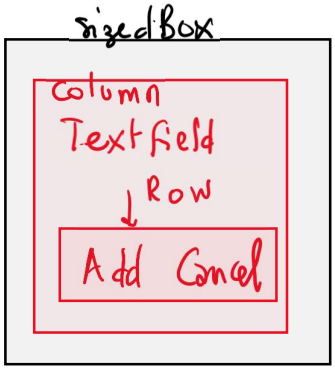
Nous utiliser le widget AlertDialog pour ajouter un nouveau produit

1. Créer un nouveau fichier add_produit.dart

2. Ajouter un nouveau widget stateless nommé AddProduit

```dart
class AddProduit extends StatelessWidget {
  final TextEditingController nomController;
  final void Function()? onAdd;
  final void Function()? onCancel;
  const AddProduit(
      {super.key,
      required this.nomController,
      required this.onAdd,
      required this.onCancel});

  @override
  Widget build(BuildContext context) {
    return AlertDialog(
        title: const Text("Ajouter un produit"),
        content: SizedBox(
          height: 120,
          child: Column(
            children: [
              TextField(
                controller: nomController,
              ),
              Row(children: [
                MaterialButton(
                  onPressed: onAdd,
                  child: const Text('Add'),
                ),
                MaterialButton(
                  onPressed: onCancel,
                  child: const Text("Cancel"),
                ),
              ]),
            ],
          ),
        ));
  }
}
```




Avant d'utiliser SizedBox vous pouvez tester Column uniquement, puis après Encapsuler par SizedBox.

3. Dans _ProduitsListState, ajouter les déclarations suivantes :

```dart
void onChanged(bool? value, int index) {
  setState(() {
    liste[index][1] = value;
  });
}

void saveProduit() {
  setState(() {
    liste.add([nomController.text, false]);
    nomController.clear(); // Efface le contenu du champ de texte
    Navigator.of(context).pop(); //Ferme la boite de dialogue
  });
}

void addProduit() {
  showDialog(
      context: context,
      builder: (context) {
        return AddProduit(
          nomController: nomController,
          onAdd: saveProduit,
          onCancel: () {
            Navigator.pop(context);
          },
        );
      });
}
```

4. Ajouter le bouton « Ajout d'un produit » en dessous de l'attribut appBar :

```dart
    ), // AppBar
    floatingActionButton: FloatingActionButton(
      onPressed: addProduit,
      child: const Icon(Icons.add),
    ), // FloatingActionButton
```

# Suppression d'un produit

Dans cette section, nous allons utiliser un widget qui n'est pas inclus dans le package 'material.dart'. Tout d'abord, nous allons le déclarer dans le fichier 'pubspec.yaml'.

5. Dans la section dependencies ajouter le package flutter_slidable, et enregistrer le fichier (documentation : flutter_slidable | Flutter Package (pub.dev)).

```
dependencies:
  flutter:
    sdk: flutter
  flutter_slidable: ^3.1.1
```

6. Dans le widget ProduitBox, encapsuler le widget racine Container par un widget Slidable.
7. Ajouter la déclaration de la fonction delProduit pour supprimer un produit (elle sera définie dans ProduitBox)

```
Function(BuildContext context)? delProduit;
ProduitBox({
  super.key,
  required this.nomProduit,
  this.selProduit = false,
  this.onChanged,
  this.delProduit,
});
```

8. Définir la propriété endActionPane

```
@override
Widget build(BuildContext context) {
  return Slidable(
    endActionPane: ActionPane(
      motion: const StretchMotion(),
      children: [
        SlidableAction(
          onPressed: delProduit,
          icon: Icons.delete,
          backgroundColor: Colors.red,
        ),
      ],
    ),
    child: SizedBox(
      height: 120,
      child: Row(
        children: [
          Checkbox(value: selProduit, onChanged: onChanged),
          Text(nomProduit),
        ],
      ),
    ),
  );
}
```

Remplacez Container par SizedBox

9. Ajouter un border radius et encapsuler Slidable dans le widget Padding.

```
Widget build(BuildContext context) {
  return Padding(
    padding: const EdgeInsets.all(10.0),
    child: Slidable(
      endActionPane: ActionPane(
        motion: const StretchMotion(),
        children: [
          SlidableAction(
            onPressed: delProduit,
            icon: Icons.delete,
            backgroundColor: Colors.red,
            borderRadius: BorderRadius.circular(45),
          ),
        ],
      ),
      child: Container(
        decoration: BoxDecoration(
          color: Colors.yellow,
          borderRadius: BorderRadius.circular(45),
        ),
        height: 120,
        child: Row(
          children: [
            Checkbox(value: selProduit, onChanged: onChanged),
            Text(nomProduit),
          ],
        ),
      ),
    ),
  );
```

10. Définit la méthode delProduit dans le widget _ProduitsListState

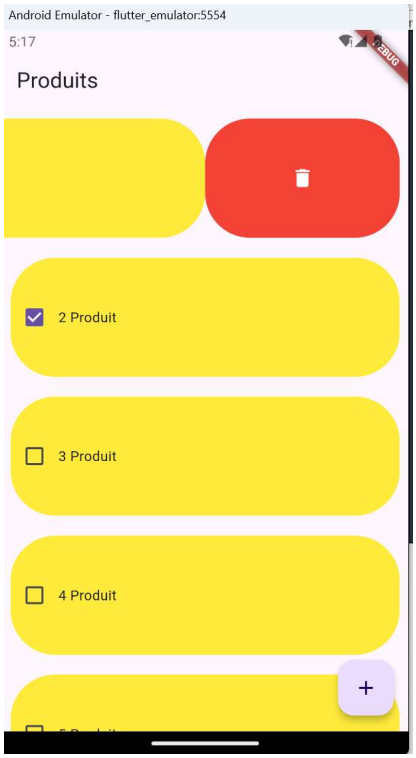
```
void delProduit(int index) {
  setState(() {
    liste.removeAt(index);
  });
}
```

11. Mettre à jour l'instanciation de ProduitBox

```
body: ListView.builder(
    itemCount: liste.length,
    itemBuilder: (context, index) {
      return ProduitBox(
        nomProduit: liste[index][0],
        selProduit: liste[index][1],
        onChanged: (value) => onChanged(value, index),
        delProduit: (context) => delProduit(index),
      ); // ProduitBox
    }), // ListView.builder
```

12. Ajouter un thème dans l'application MainApp.

```dart
@override
Widget build(BuildContext context) {
  return MaterialApp(
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: const ProduitsList());
}
```




Exercice : Ajouter la fonctionnalité de suppression des produits sélectionnés.