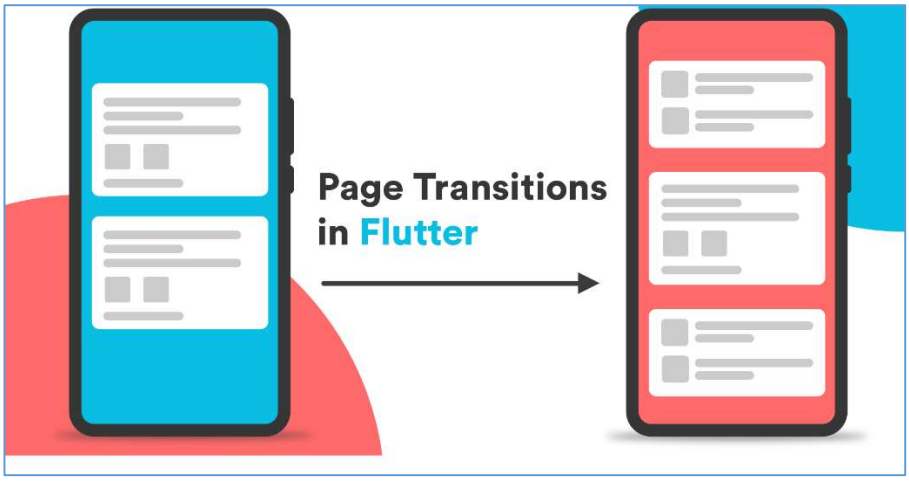
Atelier 5 Flutter
Interfaces défilantes et transitions



I. Slivers

Dans Flutter, les Slivers ne sont que des parties d'une zone de défilement. L'utilisation de Slivers nous aide à créer de nombreux effets de défilement fantaisistes et peut rendre le processus de défilement à travers un grand nombre d'enfants plus efficace grâce à la possibilité de construire paresseusement chaque élément lorsqu'il défile dans la vue.

Dans Flutter, avec des Slivers, nous pouvons créer différents effets de défilement. Les Slivers offrent une vue imprenable sur les listes lorsqu'ils défilent vers le haut ou vers le bas. Les Slivers nous permettent d'avoir un impact sur l'expérience de défilement des listes et des grilles.

Les slivers sont une catégorie spéciale de widgets utilisés dans les interfaces défilantes (scrollables). Ils permettent de créer des effets et des comportements de défilement personnalisés. Ils appartiennent à la classe Sliver et sont souvent utilisés avec des widgets comme CustomScrollView, SliverList, SliverGrid, etc.

Exemples de widgets Slivers :

SliverAppBar pour créer une AppBar incroyable.

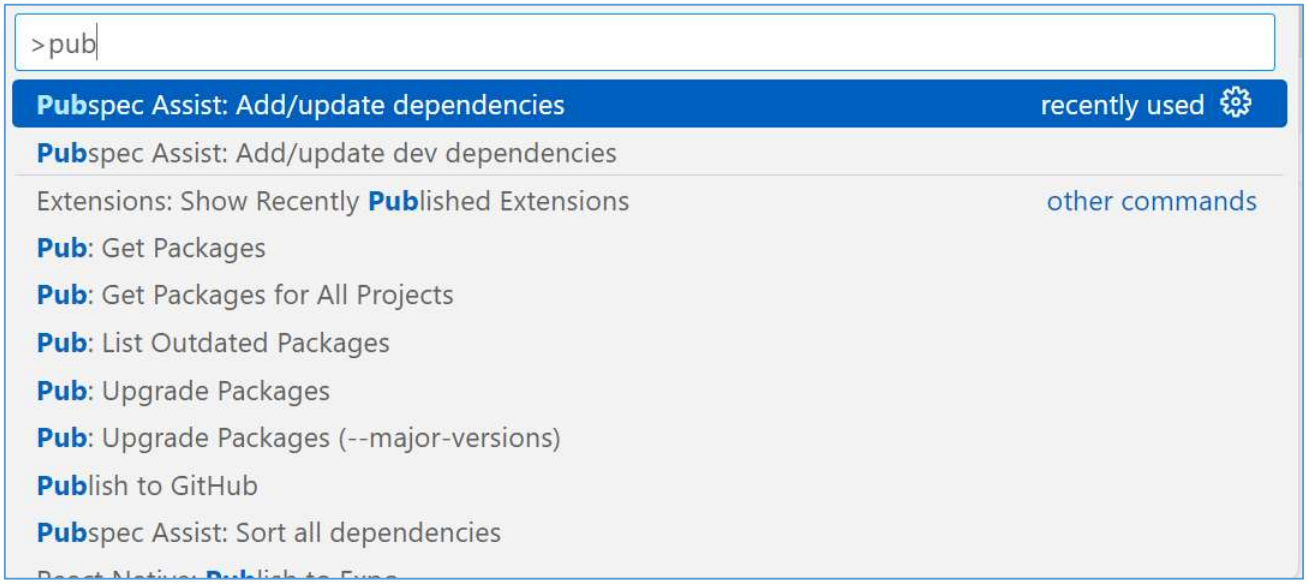
SliverList est un widget qui permet de créer des ListViews qui se comportent différemment.

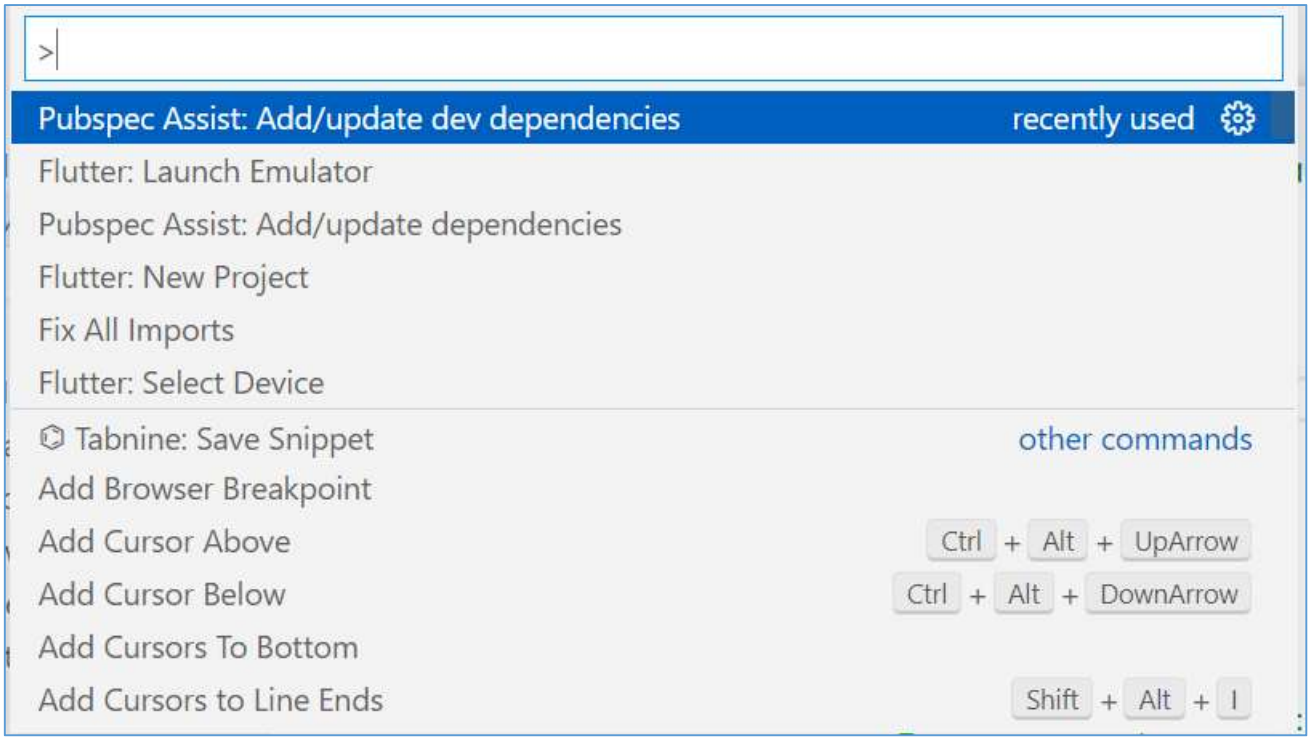
SliverGrid est un widget utilisé pour créer des GridViews qui se comportent différemment pour une meilleure

1. Ajout de dépendance

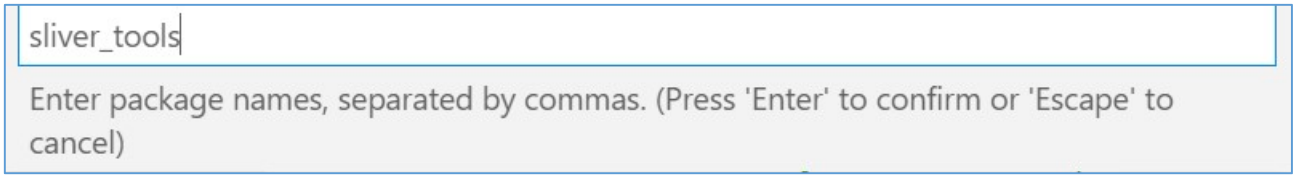
pubspec.yaml

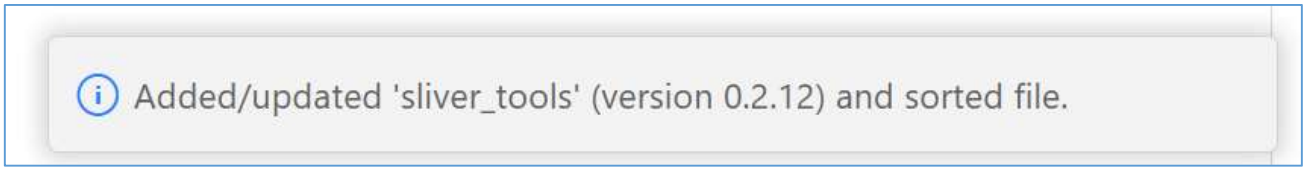
CTRL SHIFT P





sliver_tools





pubspec.yaml

```
30  dependencies:
31    cupertino_icons: ^1.0.6
32    flutter:
33      sdk: flutter
34    sliver_tools: ^0.2.12
```

2. Création du screen

Créer le fichier /lib/screens/products.dart

3. SilverAppBar

lib > screens > products.dart

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

class Products extends StatefulWidget {
  const Products({super.key});

  @override
  ProductsState createState() => ProductsState();
}

class ProductsState extends State<Products> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        body: CustomScrollView(
      slivers: [
        const SliverAppBar(
          backgroundColor: Colors.deepPurpleAccent,
```

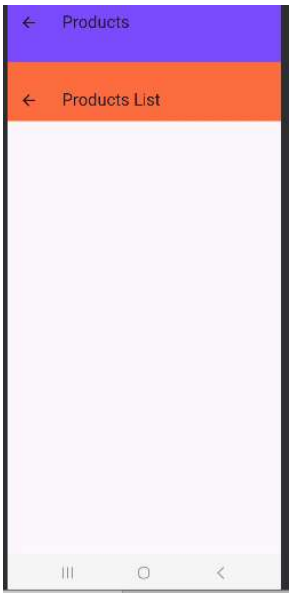
```
          title: Text('Products'),
          expandedHeight: 20,
          collapsedHeight: 80,
        ),
        const SliverAppBar(
          backgroundColor: Colors.deepOrangeAccent,
          title: Text('Products List'),
          floating: true,
        ),

      ],
    ),
    );
  }
}
```

Explication

Le paramètre floating: true dans le widget SliverAppBar en Flutter indique que l'AppBar devient "flottante", c'est-à-dire qu'elle apparaîtra dès que l'utilisateur commencera à faire défiler vers le haut, même si elle n'est pas complètement visible. En d'autres termes, lorsque l'utilisateur fait défiler une liste vers le bas, l'AppBar peut disparaître, mais dès qu'il fait défiler vers le haut, elle réapparaît immédiatement sans attendre que la position de défilement revienne tout en haut.


```
'/Products': (context) => const Products(), // Route pour l'écran Products
```

Résultat



4. SilverList

lib > screens > products.dart

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

class Products extends StatefulWidget {
  const Products({super.key});

  @override
  ProductsState createState() => ProductsState();
}

class ProductsState extends State<Products> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        body: CustomScrollView(
      slivers: [
        const SliverAppBar(
          backgroundColor: Colors.deepPurpleAccent,
          title: Text('Products'),
          expandedHeight: 20,
          collapsedHeight: 80,
        ),
        const SliverAppBar(
          backgroundColor: Colors.deepOrangeAccent,
          title: Text('Products List'),
```

```
          floating: true,
        ),
            SliverList(
          delegate: SliverChildBuilderDelegate(
            (BuildContext context, int index) {
              return Card(
                margin: const EdgeInsets.all(15),
                child: Container(
                  color: Colors.blue[100 * (index % 9 + 1)],
                  height: 80,
                  alignment: Alignment.center,
                  child: Text(
                    "Item $index",
                    style: const TextStyle(fontSize: 30),
                  ),
                ),
              );
            },
            childCount: 20, // 20 list items
          ),
        ),
      ],
    ),
    );
  }
}
```

Explication

Le SliverChildBuilderDelegate est utilisé pour générer les enfants du SliverList. Il fonctionne comme un constructeur dynamique qui crée des widgets en fonction de l'index fourni.

(BuildContext context, int index) : Cette fonction est appelée pour construire chaque élément de la liste. index représente l'index de l'élément actuellement construit.

color: Colors.blue[100 * (index % 9 + 1)] : La couleur du conteneur varie en fonction de l'index. Cette formule utilise des teintes de bleu (de Colors.blue[100] à Colors.blue[900]) en fonction de l'index. L'opérateur modulo (% 9) garantit que les couleurs reviennent à la première teinte après chaque série de neuf éléments.

Résultat


Products
Products List
Item 0
Item 1
Item 2
Item 3
Item 4


Item 3
Item 4
Item 5
Item 6
Item 7
Item 8
Item 9

5. Affichage à partir d'un tableau

lib > screens > products.dart

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

class Products extends StatefulWidget {
  const Products({super.key});

  @override
  ProductsState createState() => ProductsState();
}

class ProductsState extends State<Products> {

  @override
  Widget build(BuildContext context) {
    var myList = <Product> [
      Product(1, 'NOKIA-C1', 99, "99 %"),
      Product(2, 'BENCO-Y30', 85, "87 %"),
      Product(3, 'ITEL-P38', 89, "89 %"),
      Product(4, 'SPARKGO22', 75, "80 %"),
      Product(5, 'POP2F', 70, "65 %"),
       ];

    return Scaffold(
        body: CustomScrollView(
      slivers: [
```

```
        const SliverAppBar(
          backgroundColor: Colors.deepPurpleAccent,
          title: Text('Products'),
          expandedHeight: 20,
          collapsedHeight: 80,
        ),
        const SliverAppBar(
          backgroundColor: Colors.deepOrangeAccent,
          title: Text('Products List'),
          floating: true,
        ),
             SliverList(
          delegate: SliverChildBuilderDelegate(
            (BuildContext context, int index) {
              return Card(
                margin: const EdgeInsets.all(15),
                child: Container(
                  color: Colors.blue[100 * (index % 9 + 1)],
                  height: 80,
                  alignment: Alignment.center,
                  child: Text(
                    myList[index].designation,
                    style: const TextStyle(fontSize: 30),
                  ),
                ),
              );
            },
            childCount: myList.length
          ),
        ),
      ],
    ),
    );
  }
}

class Product {

  Product(this.usn, this.designation, this.notes, this.pourcentage);

  final int usn;

  final String designation;

  final int notes;

  final String pourcentage;
}
```

Résultat

6. Affichage d'images

lib > screens > products.dart

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

class Products extends StatefulWidget {
  const Products({super.key});

  @override
  ProductsState createState() => ProductsState();
}

class ProductsState extends State<Products> {
  @override
  Widget build(BuildContext context) {
    var myList = <Product>[
      Product(1, 'NOKIA-C1', 99, "99 %",
          "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcQ4ONgEgqUfVda5qKLzf8RiBMA3r1EtPuQueg&s"),
      Product(2, 'BENCO-Y30', 85, "87 %",
          "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcQYEJF9ykkfTbblROkLVKmrjQC2k4lbA2yqFQ&s"),
      Product(3, 'ITEL-P38', 89, "89 %",
          "https://tunisiatech.tn/8033-large_default/smartphone-itel-p38.jpg"),
```

```
      Product(4, 'SPARKGO22', 75, "80 %",
          "https://www.mega.tn/assets/uploads/img/pr_telephonie_mobile/1543051941_211.jpg"),
      Product(5, 'POP2F', 70, "65 %",
          "https://www.technopro-online.com/41138-large_default/smartphone-tecno-pop-2f-noir-tecno-pop2f-black.jpg"),
    ];

    return Scaffold(
      body: CustomScrollView(
        slivers: [
          const SliverAppBar(
            backgroundColor: Colors.deepPurpleAccent,
            title: Text('Products'),
            expandedHeight: 20,
            collapsedHeight: 80,
          ),
          const SliverAppBar(
            backgroundColor: Colors.deepOrangeAccent,
            title: Text('Products List'),
            floating: true,
          ),
          SliverList(
            delegate:
                SliverChildBuilderDelegate((BuildContext context, int index) {
              return Card(
                margin: const EdgeInsets.all(15),
                child: Container(
                  color: Colors.blue[100 * (index % 9 + 1)],
                  height: 80,
                  alignment: Alignment.center,
                  child: Column(
                    mainAxisSize: MainAxisSize.min,
                    children: [
                      ListTile(
                        leading: Image.network(myList[index].image,
                            fit: BoxFit.cover),
                        title: Text(
                          myList[index].designation,
                          style: const TextStyle(fontSize: 20),
                        ),
                        subtitle: Text(myList[index].pourcentage),
                      ),
                    ],
                  ),
                ),
              );
            }, childCount: myList.length),
          ),
```

```
        ],
      ),
    );
  }
}

class Product {
  Product(this.usn, this.designation, this.notes, this.pourcentage, this.image);

  final int usn;

  final String designation;

  final int notes;

  final String pourcentage;

  final String image;
}
```

Explication

MainAxisSize.min : La Column prendra seulement la taille minimale nécessaire pour afficher les enfants qu'elle contient. Elle ne s'étendra pas au-delà de ce qui est nécessaire. La Column ne s'étendra pas pour remplir tout l'espace disponible dans son axe principal (vertical).

Le widget ListTile est utilisé pour représenter chaque élément de la liste avec une structure classique. Il organise son contenu de manière pratique, avec des éléments comme une image à gauche, un titre et un sous-titre.

leading: Image.network(myList[index].image, fit: BoxFit.cover) :

Le paramètre leading permet d'afficher une image ou une icône à gauche du ListTile.

fit: BoxFit.cover : Cela ajuste l'image de manière à couvrir l'espace disponible sans déformer l'image, coupant éventuellement une partie de celle-ci pour maintenir les proportions.

title: Text(myList[index].designation) :

Le title est un champ texte qui affiche la désignation de l'élément.

subtitle: Text(myList[index].pourcentage) :

Le subtitle affiche un sous-titre sous le titre principal. Ici, il affiche le pourcentage de l'élément, tiré de la propriété pourcentage de l'élément de la liste.

Résultat

Remarque : Pour avoir l'effet de Sliver il est important d'avoir une liste longue. Exemple de données :

```
    var myList = <Product>[
      Product(1, 'NOKIA-C1', 99, "99 %",
          "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcQ4ONgEgqUfVda5qKLzf8RiBMA3r1EtPuQueg&s"),
      Product(2, 'BENCO-Y30', 85, "87 %",
          "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcQYEJF9ykkfTbblROkLVKmrjQC2k4lbA2yqFQ&s"),
      Product(3, 'ITEL-P38', 89, "89 %",
          "https://tunisiatech.tn/8033-large_default/smartphone-itel-p38.jpg"),
      Product(4, 'SPARKGO22', 75, "80 %",
          "https://www.mega.tn/assets/uploads/img/pr_telephonie_mobile/1543051941_21
1.jpg"),
      Product(5, 'POP2F', 70, "65 %",
          "https://www.technopro-online.com/41138-large_default/smartphone-tecno-
pop-2f-noir-tecno-pop2f-black.jpg"),
     Product(6, 'OPPO-F9', 99, "99 %",
```

```
          "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcThPYOt-
Lk95pmWASWIkepdISwow5dPRsgzOmZXZq0jqIQJ3_d-Sy5vsVLdgcwtYfgFGD4&usqp=CAU"),
      Product(7, 'SPARKGO2022', 85, "87 %",
          "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcQHqp2vuXX3a5yqjAWkHFgOnZ7rJ8Y-mnk3EQ&s"),
      Product(8, 'TECNO', 89, "89 %",
          "https://www.mega.tn/assets/uploads/img/pr_telephonie_mobile/8c929-
smartphone-tecno-pop-6-2go-32go.jpg"),
      Product(9, 'SPARKGO22', 75, "80 %",
          "https://www.mega.tn/assets/uploads/img/pr_telephonie_mobile/1543051941_21
1.jpg"),
      Product(10, 'TECNO SPARK', 70, "65 %",
          "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcRY0hq7VvmOWGtOvtJVdZ8fVgBRBGSMEaanddr3yoi97ek6Wj
EYsP00t3WlQmhg3O8K60k&usqp=CAU"),
    ];
```

Résultat

II. Hero

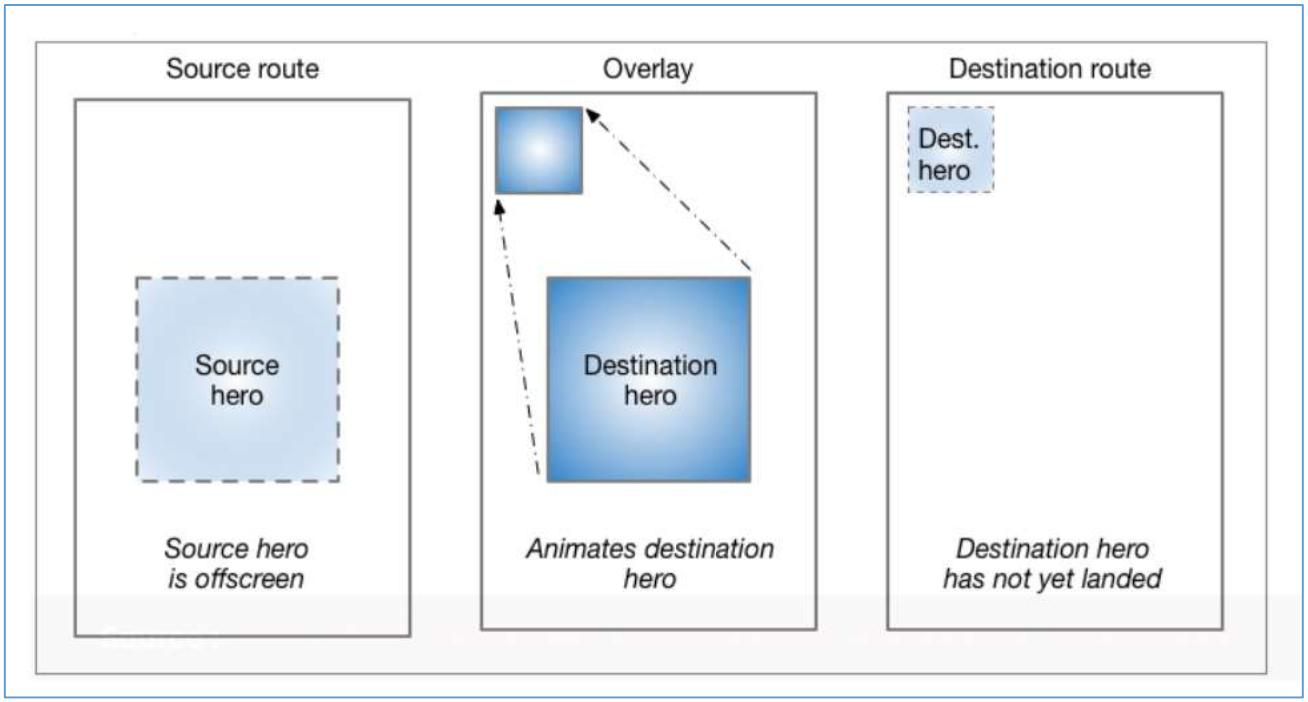
Le widget Hero appartient à la catégorie des widgets de transition dans Flutter. Il est utilisé pour créer des animations visuelles entre deux routes (pages) lors de la navigation. Le widget Hero fait partie de la classe Transition Widgets car il facilite les transitions d'éléments entre différentes pages.

Ce super widget permet de créer une transition entre deux pages. Il a la faculté de se déplacer et se transformer.

Le widget Hero est identifié grâce à un tag unique par page. Il est important de prendre en compte cette contrainte si vous l'utilisez dans des ListView par exemple. Lors d'une transition, tous les widgets Hero des pages source et cible sont récupérés afin de créer une animation pour chaque correspondance avec leurs contraintes (taille, position, etc).

À la transition un Overlay est créé dans le Navigator (grâce au widget OverlayEntry) pour interpoler le rendu pendant la durée de transition.

Voici en image ce qui se passe :



Pour prendre en compte les gestures, il faut spécifier la valeur true à la propriété transitionOnUserGestures sur chaque widget Hero.

Commençons par créer le fichier /lib/**models**/**Product.class.dart**

Puis couper et y copier la classe Product

lib > models > Product.class.dart

```
class Product {
  Product(this.usn, this.designation, this.notes, this.pourcentage, this.image);

  final int usn;

  final String designation;
```

```
  final int notes;

  final String pourcentage;

  final String image;
}
```

Dans :

La déclaration de la classe sara remplacée par l'appel import

```
import 'package:flutter/material.dart';
import 'package:myflutterapplication/models/Product.class.dart';

class Products extends StatefulWidget {
  const Products({super.key});

  @override
  ProductsState createState() => ProductsState();
}

class ProductsState extends State<Products> {
  @override
  Widget build(BuildContext context) {
    var myList = <Product>[
      Product(1, 'NOKIA-C1', 99, "99 %",
          "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcQ4ONgEgqUfVda5qKLzf8RiBMA3r1EtPuQueg&s"),
      Product(2, 'BENCO-Y30', 85, "87 %",
          "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcQYEJF9ykkfTbblROkLVKmrjQC2k4lbA2yqFQ&s"),
      Product(3, 'ITEL-P38', 89, "89 %",
          "https://tunisiatech.tn/8033-large_default/smartphone-itel-p38.jpg"),
      Product(4, 'SPARKGO22', 75, "80 %",
          "https://www.mega.tn/assets/uploads/img/pr_telephonie_mobile/1543051941_21
1.jpg"),
      Product(5, 'POP2F', 70, "65 %",
          "https://www.technopro-online.com/41138-large_default/smartphone-tecno-
pop-2f-noir-tecno-pop2f-black.jpg"),
    ];

    return Scaffold(
      body: CustomScrollView(
        slivers: [
          const SliverAppBar(
            backgroundColor: Colors.deepPurpleAccent,
```

```
            title: Text('Products'),
            expandedHeight: 20,
            collapsedHeight: 80,
          ),
          const SliverAppBar(
            backgroundColor: Colors.deepOrangeAccent,
            title: Text('Products List'),
            floating: true,
          ),
          SliverList(
            delegate:
                SliverChildBuilderDelegate((BuildContext context, int index) {
              return Card(
                margin: const EdgeInsets.all(15),
                child: Container(
                  color: Colors.blue[100 * (index % 9 + 1)],
                  height: 80,
                  alignment: Alignment.center,
                  child: Column(
                    mainAxisSize: MainAxisSize.min,
                    children: [
                      ListTile(
                        leading: Image.network(myList[index].image,
                            fit: BoxFit.cover),
                        title: Text(
                          myList[index].designation,
                          style: const TextStyle(fontSize: 20),
                        ),
                        subtitle: Text(myList[index].pourcentage),
                      ),
                    ],
                  ),
                ),
              );
            }, childCount: myList.length),
          ),
        ],
      ),
    );
  }
}
```

Maintenant on va mettre en place Hero. L'animation "Hero" permet de créer un effet visuel où un widget semble se déplacer d'une page à une autre, donnant une impression de continuité visuelle.

Hero doit figurer dans les 2 pages Products et Détail pour avoir la transition.

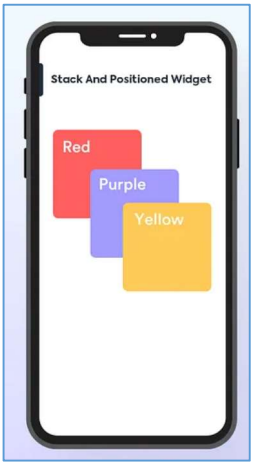
Créer le fichier /lib/**screens**/**details.dart**


```
import 'package:flutter/material.dart';
import 'package:myflutterapplication/models/Product.class.dart';

class Details extends StatelessWidget {
  final Product myListElement;

  const Details({
    super.key,
    required this.myListElement,
  });

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(myListElement.designation),
        backgroundColor: const Color.fromARGB(255, 55, 147, 15),
      ),
      body: Stack(
        children: [
          Container(
            width: double.infinity,
            height: double.infinity,
            color: Colors.green,
            margin: const EdgeInsets.all(10),
            child: Hero(
              tag: myListElement.usn,
              transitionOnUserGestures: true,
              child: Image.network(myListElement.image, fit: BoxFit.cover),
            ),
          ),
         Positioned(
          top: 600,
          left: 20,
          child : Container(
            width: 220,
            height: 50,
            color: Colors.white,
            child: Text("Pourcentage : ${myListElement.pourcentage}",style: const
TextStyle(fontSize: 24)),
          ),
```

```
          ),
             Positioned(
           top: 600,
           left: 250,
           child : Container(
             width: 100,
             height: 50,
              color: Colors.white,
             child: Text("Note : ${myListElement.notes.toString()}",style: const
TextStyle(fontSize: 24),),
           ),

       ),
       ],
       ),
     );
   }
}
```

Explication

Dans Flutter, Stack est un conteneur qui permet de placer ses widgets enfants les uns sur les autres, le premier widget enfant sera placé en bas. Stack est une solution pour économiser de l'espace de l'application. Il est possible de modifier l'ordre des widgets enfants pour créer une animation simple.



Le widget Positioned en Flutter est utilisé pour positionner un widget à des emplacements spécifiques dans un conteneur parent de type Stack. Il permet de contrôler précisément la position d'un enfant en définissant des distances par rapport aux bords du parent.

Le widget Positioned n'est utilisable qu'à l'intérieur d'un widget Stack, qui permet de superposer des widgets les uns sur les autres. Le Positioned fixe la position d'un de ces enfants à des distances spécifiques des bords du conteneur.

Le Hero ci-dessus se distingue par le paramètre « tag ».

```
child: Hero(
            tag: myListElement.usn,
```

Le paramètre tag dans un widget Hero est utilisé pour identifier de manière unique l'élément qui participera à l'animation "Hero" lors d'une transition entre deux pages (ou routes). Le paramètre tag est utilisé pour associer un widget sur la page source et un widget sur la page destination. **Ces deux widgets doivent avoir le même tag** pour que Flutter sache que l'animation doit être réalisée entre eux.

En conséquence, nous pouvons accéder au Hero Widget particulier par le “tag” dans n’importe quelle page.

Le paramètre transitionOnUserGestures: true dans un widget Hero en Flutter permet de déclencher l'animation de transition "Hero" même lorsque l'utilisateur effectue un geste de navigation (comme un balayage) pour revenir à la page précédente.

Par défaut, l'animation "Hero" se déclenche uniquement lorsque la navigation est initiée par un changement de route explicite (par exemple, via un bouton de navigation ou un appel à Navigator.push). Si l'utilisateur effectue un geste de navigation (comme un glissement pour revenir à la page précédente sur iOS), l'animation "Hero" n'est pas automatiquement exécutée.

Avec transitionOnUserGestures: true, vous permettez à l'animation de se déclencher même lorsque l'utilisateur utilise un geste pour naviguer. Cela crée une expérience plus fluide et naturelle, car l'animation "Hero" se déclenche également dans ces cas de navigation gestuelle.

lib > screens > products.dart

```
import 'package:flutter/material.dart';
import 'package:myflutterapplication/models/Product.class.dart';

class Products extends StatefulWidget {
  const Products({super.key});

  @override
  ProductsState createState() => ProductsState();
}
```

```
class ProductsState extends State<Products> {
  @override
  Widget build(BuildContext context) {
    var myList = <Product>[
      Product(1, 'NOKIA-C1', 99, "99 %",
          "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcQ4ONgEgqUfVda5qKLzf8RiBMA3r1EtPuQueg&s"),
      Product(2, 'BENCO-Y30', 85, "87 %",
          "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcQYEJF9ykkfTbblROkLVKmrjQC2k4lbA2yqFQ&s"),
      Product(3, 'ITEL-P38', 89, "89 %",
          "https://tunisiatech.tn/8033-large_default/smartphone-itel-p38.jpg"),
      Product(4, 'SPARKGO22', 75, "80 %",
          "https://www.mega.tn/assets/uploads/img/pr_telephonie_mobile/1543051941_21
1.jpg"),
      Product(5, 'POP2F', 70, "65 %",
          "https://www.technopro-online.com/41138-large_default/smartphone-tecno-
pop-2f-noir-tecno-pop2f-black.jpg")
    ];

    return Scaffold(
      body: CustomScrollView(
        slivers: [
          const SliverAppBar(
            backgroundColor: Colors.deepPurpleAccent,
            title: Text('Products'),
            expandedHeight: 20,
            collapsedHeight: 80,
          ),
          const SliverAppBar(
            backgroundColor: Colors.deepOrangeAccent,
            title: Text('Products List'),
            floating: true,
          ),
          SliverList(
            delegate:
                SliverChildBuilderDelegate((BuildContext context, int index) {
              return Card(
                margin: const EdgeInsets.all(15),
                child: Container(
                  color: Colors.blue[100 * (index % 9 + 1)],
                  height: 80,
                  alignment: Alignment.center,
                  child: Column(
                    mainAxisSize: MainAxisSize.min,
                    children: [
                      ListTile(
                        onTap: () {
```

```
                        Navigator.of(context)
                            .pushNamed("/details", arguments: myList[index]);
                      },
                      leading: Hero(
                        tag: myList[index].usn,
                        transitionOnUserGestures: true,
                        child: Image.network(myList[index].image,fit:
BoxFit.cover),
                      ),
                      title: Text(
                        myList[index].designation,
                        style: const TextStyle(fontSize: 20),
                      ),
                      subtitle: Text(myList[index].pourcentage),
                    ),
                  ],
                ),
              ),
            );
          }, childCount: myList.length),
        ),
      ],
    ),
  );
 }
}
```

Ajouter la route vers le détail article avec l'argument qui représente l'article choisi.

```
 '/details': (context) {
      final product = ModalRoute.of(context)!.settings.arguments as Product;
      return Details(myListElement:product); // Retourne la page des détails avec le
produit passé en argument
    },
```

Explication

ModalRoute.of(context) : Permet d'accéder aux arguments passés à la route actuelle. Dans ce cas, l'argument est de type Product.

Route /details : La route utilise une fonction anonyme qui récupère les arguments via ModalRoute.of(context) et retourne le widget Details avec le produit passé en paramètre.

Utilisation de WidgetBuilder : Chaque route doit retourner un widget avec un contexte, ce qui est respecté ici.

Le code complet est :

```
// Importation des packages nécessaires
import 'package:flutter/material.dart';
import 'package:myflutterapplication/models/Product.class.dart';
import 'package:myflutterapplication/screens/details.dart';
import 'package:myflutterapplication/screens/documents.dart';
import 'package:myflutterapplication/screens/exitscreen.dart';
import 'package:myflutterapplication/screens/menu.dart';
import 'package:myflutterapplication/screens/myproducts.dart';
import 'package:myflutterapplication/screens/products.dart';
import 'package:myflutterapplication/widgets/myappbar.dart';
import 'package:myflutterapplication/widgets/mybottomnavbar.dart';
import 'package:myflutterapplication/widgets/mydrawer.dart';

// Définition d'une fonction qui retourne les routes
Map<String, WidgetBuilder> appRoutes() {
  return {
    '/': (context) => const Scaffold(
          appBar: Myappbar(),
          body: Menu(),
          drawer: MyDrawer(),
          bottomNavigationBar: MyBottomNavigation(),
        ),
    '/Items': (context) => Scaffold(
          appBar: AppBar(
            title: const Text('My Products'),
          ),
          body: const Myproducts(),
          drawer: const MyDrawer(),
          bottomNavigationBar: const MyBottomNavigation(),
        ),
    '/Exit': (context) =>
        const ExitScreen(), // Route associée à l'action de fermeture
    '/Documents': (context) =>
        const Documents(), // Route pour l'écran Documents
    '/Products': (context) => const Products(), // Route pour l'écran Products
    '/details': (context) {
      final product = ModalRoute.of(context)!.settings.arguments as Product;
      return Details(
          myListElement:
```

```
              product); // Retourne la page des détails avec le produit passé en
argument
    }, // Route pour l'écran Details
  };
}
```

Résultat