**Atelier 9 Flutter**

**Shopping Cart**

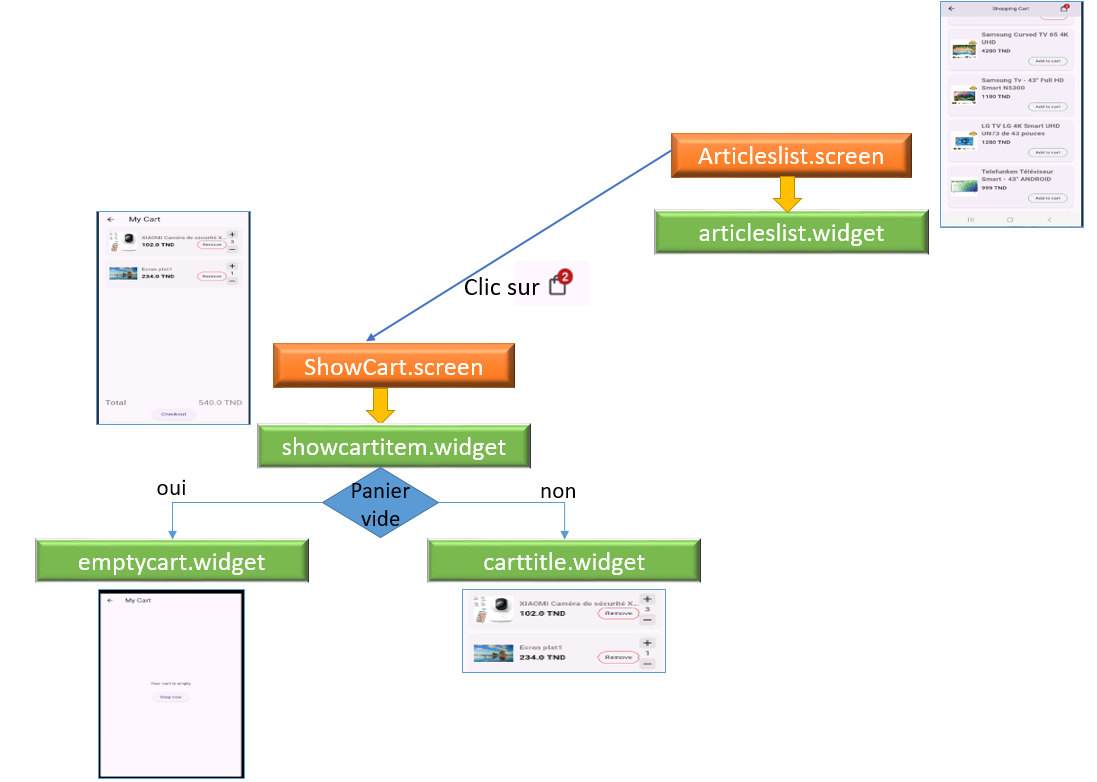


# Introduction

Le panier d'achat est une fonctionnalité essentielle de chaque application de commerce électronique et les développeurs déploient beaucoup d'efforts pour créer un panier d'achat dynamique.

Avec le Package Flutter Persistent Shopping Cart, il sera possible de gérer toutes les logiques, les calculs et fournira une interface dynamique pour créer un panier d'achat.

# Enchaînement



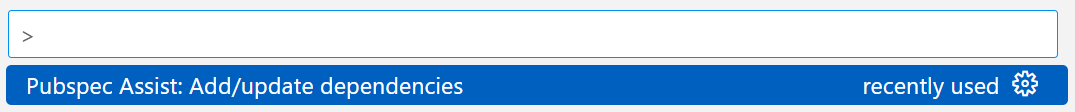
# Installer les dépendances

**Persistent Shopping Cart** est un package Flutter qui fournit une fonctionnalité de panier d'achat simple et persistante pour votre application mobile. Il utilise Hive pour le stockage local, ce qui permet aux données du panier de persister entre les sessions d'application.

Ce package gérera toutes les logiques, les calculs et fournira une interface dynamique pour créer un panier d'achat.

Site : <https://pub.dev/packages/persistent_shopping_cart>

CTRL SHIFT P



persistent\_shopping\_cart

# Le service

On va créer un service pour récupérer la liste des articles à partir d’une API consommant un service web provenant du backend.



import 'package:dio/dio.dart';

import 'package:myflutterapplication/utils/constants.dart';

class ArticleService {

  late Dio dio;

  ArticleService() {

    BaseOptions options = BaseOptions(

      baseUrl: baseUrl,

      receiveDataWhenStatusError: true,

      connectTimeout: const Duration(seconds: 5),

      receiveTimeout: const Duration(seconds: 5),

    );

    dio = Dio(options);

  }

  //Affichage

  Future<List<dynamic>> getArticles() async {

    try {

      Response response = await dio.get('/articles');

      print(response.data.toString());

      return response.data;

    } catch (e) {

      print(e.toString());

      return [];

    }

  }

}

# Les Models

Le model Article contient un champ scategorieID qui est un objet faisant référence à Scategorie. C’est pour ça qu’il faut au préalable créer le Model Scategorie au préalable, pour qu’on puisse y faire référence par la suite dans le Model Article.

A son tour le Model Scategorie, ayant categorieID qui est un objet faisant référence à Categorie.



import 'package:myflutterapplication/data/datasource/models/categories.model.dart';

class Scategorie {

  String? id;

  String? nomscategorieID;

  String? imagescategorie;

  Categorie? categorieID;

  Scategorie({this.id, this.nomscategorieID, this.imagescategorie, this.categorieID});

  Scategorie.fromJson(Map<String, dynamic> json) {

    id = json['\_id'];

    nomscategorieID = json['nomscategorieID'];

    imagescategorie = json['imagescategorie'];

    categorieID = json['categorieID'] != null

        ? Categorie.fromJson(json['categorieID'])

        : null;

  }

  Map<String, dynamic> toJson() {

    final Map<String, dynamic> data = <String, dynamic>{};

    data['\_id'] = id;

    data['nomscategorieID'] = nomscategorieID;

    data['imagescategorie'] = imagescategorie;

    if (categorieID != null) {

      data['categorieID'] = categorieID!.toJson();

    }

    return data;

  }

}

**Remarque** :

On a ajouté des contrôles pour le cas où categorieID ayant une valeur nulle. C’est un contrôle supplémentaire pour améliorer le code aux risques des erreurs et qui n’est pas obligatoire.



class Article {

  String? id;

  String? reference;

  String? designation;

  num? prix;

  String? marque;

  int? qtestock;

  String? imageart;

  Map<String, dynamic>? scategorieID;

  Article(

      {this.id,

      this.reference,

      this.designation,

      this.prix,

      this.marque,

      this.qtestock,

      this.imageart,

      this.scategorieID});

  Article.fromJson(Map<String, dynamic> json) {

    id = json['\_id'];

    reference = json['reference'];

    designation = json['designation'];

    prix = json['prix'];

    marque = json['marque'];

    qtestock = json['qtestock'];

    imageart = json['imageart'];

    scategorieID = json['scategorieID'] ;

  }

  Map<String, dynamic> toJson() {

    final Map<String, dynamic> data = <String, dynamic>{};

    data['\_id'] = id;

    data['reference'] = reference;

    data['designation'] = designation;

    data['prix'] = prix;

    data['marque'] = marque;

    data['qtestock'] = qtestock;

    data['imageart'] = imageart;

    data['scategorieID'] = scategorieID;

  return data;

  }

}

Explication

On a choisi le type num pour le prix. Ce type peut accepter le type Int ou Double.

# Le repository



import 'package:myflutterapplication/data/datasource/models/articles.model.dart';

import 'package:myflutterapplication/data/datasource/services/article.service.dart';

class ArticleRepository {

  final ArticleService artserv;

  ArticleRepository({required this.artserv});

//Affichage

  Future<List<Article?>?> getArticles() async {

    try {

      final articles = await artserv.getArticles();

      return articles.map((art) => Article.fromJson(art)).toList();

    } catch (error) {

      print("Error : $error");

      return null;

    }

  }

}

# Entity

Dans Entity on a choisi de prendre quelques champs et non la totalité de la base de données.



class ArticleEntity {

  final String id;

  final String designation;

  final num? prix;

  final int? qtestock;

  final String? imageart;

  ArticleEntity({required this.id, required this.designation, required this.prix, required this.qtestock, required this.imageart});

}

# UseCase



import 'package:myflutterapplication/data/repositories/article.repository.dart';

import 'package:myflutterapplication/domain/entities/article.entity.dart';

class ArticleUseCase {

  final ArticleRepository \_respository;

  ArticleUseCase({required ArticleRepository respository})

      : \_respository = respository;

  Future<List<ArticleEntity?>?> fetchArticles() async {

    final result = await \_respository.getArticles();

//Affichage

    final data = result?.map((element) {

      return ArticleEntity(

         id: element?.id ?? "",

         designation: element?.designation ?? "",

        prix: element?.prix ?? 0,

        qtestock: element?.qtestock ?? 0,

        imageart: element?.imageart ?? "",

      );

    }).toList();

    return data;

    }

}

# Le Controller



import 'package:get/get.dart';

import 'package:myflutterapplication/domain/entities/article.entity.dart';

import 'package:myflutterapplication/domain/usecases/article.usecase.dart';

class ArticleController extends GetxController {

  final ArticleUseCase \_useCase;

  var articlesList = <ArticleEntity>[].obs;

  var isLoading = true.obs;

  ArticleController({required ArticleUseCase useCase}) : \_useCase = useCase;

//Affichage

  fetchAllArticles() {

    \_useCase.fetchArticles().then((data) {

      isLoading.value = false;

      if (data != null) {

        final result = data

            .map(

              (element) => ArticleEntity(

                id: element?.id ?? "",

                designation: element?.designation ?? "",

                prix: element?.prix ?? 0,

                qtestock: element?.qtestock ?? 0,

                imageart: element?.imageart ?? "",

              ),

            )

            .toList();

        articlesList.value = result;

      }

    }).catchError((error) {});

  }

}

Explication

<ArticleEntity>[] : Cela crée une liste vide d'objets de type ArticleEntity.

.obs : GetX offre la capacité de rendre une variable observable, c'est-à-dire que toute modification de la variable est automatiquement détectée, et les widgets qui dépendent de cette variable se mettront à jour en conséquence. En rendant la liste observable, vous assurez que dès que vous modifiez articlesList, l'interface utilisateur sera mise à jour avec les nouvelles données.

var isLoading = true.obs;

true : Cela indique que le contrôleur est en mode "chargement" au départ. En effet, vous définissez la variable à true car au début, les données ne sont pas encore chargées.

.obs : Cette variable est également observable. Toute modification de la valeur de isLoading sera détectée et déclenchera une mise à jour de l'interface utilisateur.

# Le widget de la liste des articles

Le code permet d’afficher dynamiquement soit un bouton "Add to cart", soit un bouton "Remove" en fonction de la présence ou non de l'article dans le panier. Le widget permet donc de gérer de manière intuitive l'ajout et la suppression d'articles dans un panier d'achats.

Si un utilisateur consulte un article et qu'il n'est pas encore dans son panier, il verra un bouton "Add to cart" pour l'ajouter.

Une fois l'article ajouté, ce bouton se transforme en "Remove", permettant de retirer l'article du panier si désiré.



import 'package:flutter/material.dart';

import 'package:get/get.dart';

import 'package:myflutterapplication/domain/entities/article.entity.dart';

import 'package:myflutterapplication/presentation/controllers/article.controller.dart';

import 'package:persistent\_shopping\_cart/model/cart\_model.dart';

import 'package:persistent\_shopping\_cart/persistent\_shopping\_cart.dart';

class ArticlesListWidget extends StatelessWidget {

  final ArticleEntity articles;

  ArticlesListWidget({super.key, required this.articles});

  //on a enlevé const car : Can't define the 'const' constructor because the field 'controller' is initialized with a non-constant value.

  final controller = Get.find<ArticleController>();

  @override

  Widget build(BuildContext context) {

    return Card(

      child: Padding(

        padding: const EdgeInsets.all(8.0),

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          crossAxisAlignment: CrossAxisAlignment.start,

          children: [

            Row(

              mainAxisAlignment: MainAxisAlignment.start,

              crossAxisAlignment: CrossAxisAlignment.center,

              mainAxisSize: MainAxisSize.max,

              children: [

                Image.network(

                  articles.imageart.toString(),

                  width: 68,

                  height: 68,

                ),

                const SizedBox(

                  width: 10,

                ),

                Expanded(

                  child: Column(

                    mainAxisAlignment: MainAxisAlignment.start,

                    crossAxisAlignment: CrossAxisAlignment.start,

                    children: [

                      Text(

                        articles.designation,

                        style: const TextStyle(

                            fontSize: 18,

                            fontWeight: FontWeight.w700,

                            color: Color.fromARGB(255, 104, 103, 103)),

                      ),

                      const SizedBox(

                        height: 5,

                      ),

                      Text(

                        "${articles.prix} TND",

                        style: const TextStyle(

                            fontSize: 16, fontWeight: FontWeight.w500),

                      ),

                      Align(

                        alignment: Alignment.centerRight,

                        child: PersistentShoppingCart()

                            .showAndUpdateCartItemWidget(

                          inCartWidget: Container(

                            height: 30,

                            width: 70,

                            decoration: BoxDecoration(

                              borderRadius: BorderRadius.circular(20),

                              border: Border.all(color: Colors.red),

                            ),

                            child: Center(

                              child: Text(

                                'Remove',

                                style: Theme.of(context).textTheme.bodySmall,

                              ),

                            ),

                          ),

                          notInCartWidget: Container(

                            height: 30,

                            width: 100,

                            decoration: BoxDecoration(

                              border: Border.all(color: Colors.green),

                              borderRadius: BorderRadius.circular(20),

                            ),

                            child: Padding(

                              padding:

                                  const EdgeInsets.symmetric(horizontal: 5),

                              child: Center(

                                child: Text(

                                  'Add to cart',

                                  style: Theme.of(context).textTheme.bodySmall,

                                ),

                              ),

                            ),

                          ),

                          product: PersistentShoppingCartItem(

                              productId: articles.id,

                              productName: articles.designation,

                              unitPrice: articles.prix!.toDouble(),

                              productImages: [

                                articles.imageart!

                              ], // Mettre l'image dans une liste

                              quantity: 1),

                        ),

                      )

                    ],

                  ),

                )

              ],

            )

          ],

        ),

      ),

    );

  }

}

Explication

La ligne suivante : ***PersistentShoppingCart().showAndUpdateCartItemWidget*** fait appel à une méthode showAndUpdateCartItemWidget de la classe PersistentShoppingCart, provenant du package persistent\_shopping\_cart.

Il s'agit de la classe qui gère le panier d'achats (shopping cart) de manière persistante, c'est-à-dire qu'elle conserve l'état du panier même après que l'utilisateur quitte ou redémarre l'application.

Cette classe permet de manipuler les articles ajoutés au panier (ajout, mise à jour, suppression).

Elle gère l'affichage et la mise à jour des éléments du panier. Elle détermine si un article est déjà dans le panier ou non, et affiche un widget en conséquence.

La méthode prend deux widgets en paramètre :

inCartWidget : Ce widget est affiché si l'article est déjà dans le panier. Ici, c'est un bouton "Remove" qui permet de retirer l'article du panier.

notInCartWidget : Ce widget est affiché si l'article n'est pas dans le panier. Ici, c'est un bouton "Add to cart" qui permet d'ajouter l'article au panier.

product : s'agit de l’objet de type PersistentShoppingCartItem qui représente l'article dans le panier. Cet objet contient :

productId : L'ID de l'article.

productName : Le nom de l'article.

unitPrice : Le prix unitaire de l'article.

productImages : Une liste d'images associées à l'article. Dans ce cas, c'est l'image de l'article (convertie en liste).

quantity : La quantité de cet article dans le panier (ici initialisée à 1).

La méthode showAndUpdateCartItemWidget vérifie si l'article est déjà dans le panier grâce à l'ID du produit (dans PersistentShoppingCartItem).

Si l'article est déjà dans le panier, le widget défini dans inCartWidget (le bouton "Remove") sera affiché, permettant à l'utilisateur de retirer l'article du panier.

Si l'article n'est pas encore dans le panier, le widget défini dans notInCartWidget (le bouton "Add to cart") sera affiché, permettant à l'utilisateur d'ajouter l'article au panier.

# Le screen de la liste des articles



import 'package:flutter/material.dart';

import 'package:get/get.dart';

import 'package:myflutterapplication/presentation/controllers/article.controller.dart';

import 'package:myflutterapplication/presentation/widgets/articleslist.widget.dart';

import 'package:persistent\_shopping\_cart/persistent\_shopping\_cart.dart';

class ArticlesListScreen extends StatelessWidget {

  const ArticlesListScreen({super.key});

  @override

  Widget build(BuildContext context) {

    var controller = Get.find<ArticleController>();

    controller.fetchAllArticles();

    return Scaffold(

      appBar: AppBar(

        title: const Text(

          'Shopping Cart',

          style: TextStyle(fontSize: 15),

        ),

        centerTitle: true,

        actions: [

          PersistentShoppingCart().showCartItemCountWidget(

            cartItemCountWidgetBuilder: (itemCount) => IconButton(

              onPressed: () {

                Navigator.pushNamed(context, '/cartView');

              },

              icon: Badge(

                label: Text(itemCount.toString()),

                child: const Icon(Icons.shopping\_bag\_outlined),

              ),

            ),

          ),

          const SizedBox(width: 20.0)

        ],

      ),

      body: Obx(

        () => controller.isLoading.value == true

            ? const Center(

                child: CircularProgressIndicator(),

              )

            : SafeArea(

                child: Padding(

                  padding: const EdgeInsets.symmetric(horizontal: 15),

                  child: ListView.builder(

                      itemCount: controller.articlesList.length,

                      itemBuilder: (context, index) {

                        final articles = controller.articlesList[index];

                        return Padding(

                          padding: const EdgeInsets.only(bottom: 5),

                          child: ArticlesListWidget(articles: articles,)

                        );

                      }),

                ),

              ),

      ),

    );

  }

}

Explication

La méthode PersistentShoppingCart().showCartItemCountWidget est utilisée pour afficher un widget qui indique le nombre d'articles dans le panier.

cartItemCountWidgetBuilder : Cette fonction de rappel est appelée avec le nombre actuel d'articles (itemCount) et construit un widget qui sera affiché dans l'AppBar.

# main.dart



1. Injection des dépendances

 //injection articles getx

  Get.put(ArticleService());

  Get.put(ArticleRepository(artserv: Get.find()));

  Get.put(ArticleUseCase(respository: Get.find()));

  Get.put(ArticleController(useCase: Get.find()));

1. Importer le package persistent\_shopping\_cart et l’initialiser

 //Initialiser shoppingCart

 await PersistentShoppingCart().init();

Le code complet devient :

import 'package:flutter/material.dart';

import 'package:myflutterapplication/approuter.dart';

import 'package:get/instance\_manager.dart';

import 'package:myflutterapplication/data/datasource/localdatasource/user\_local\_data\_source.dart';

import 'package:myflutterapplication/data/datasource/services/article.service.dart';

import 'package:myflutterapplication/data/datasource/services/categorie.service.dart';

import 'package:myflutterapplication/data/repositories/article.repository.dart';

import 'package:myflutterapplication/data/repositories/categorie.repository.dart';

import 'package:myflutterapplication/data/repositories/user.repository.dart';

import 'package:myflutterapplication/domain/usecases/article.usecase.dart';

import 'package:myflutterapplication/domain/usecases/categorie.usecase.dart';

import 'package:myflutterapplication/domain/usecases/user.usecase.dart';

import 'package:myflutterapplication/presentation/controllers/article.controller.dart';

import 'package:myflutterapplication/presentation/controllers/categorie.controller.dart';

import 'package:myflutterapplication/presentation/controllers/user.controller.dart';

import 'package:persistent\_shopping\_cart/persistent\_shopping\_cart.dart';

// Fonction principale qui lance l'application Flutter

void main() **async** {

 //Initialiser shoppingCart

 await PersistentShoppingCart().init();

  //injection articles getx

  Get.put(ArticleService());

  Get.put(ArticleRepository(artserv: Get.find()));

  Get.put(ArticleUseCase(respository: Get.find()));

  Get.put(ArticleController(useCase: Get.find()));

  //injection categories getx

  Get.put(CategorieService());

  Get.put(CategorieRepository(catserv: Get.find()));

  Get.put(CategorieUseCase(respository: Get.find()));

  Get.put(CategorieController(useCase: Get.find()));

// injection authentification

   Get.put(UserLocalDataSource());

  Get.put(UserRepository(localDataSource: Get.find()));

  Get.put(AuthenticateUserUseCase(repository: Get.find()));

  Get.put(AuthController(userUseCase: Get.find()));

  // Lance l'application en exécutant MyApp

  runApp(const MyApp());

}

// Définition de la classe stateless MyApp

class MyApp extends StatelessWidget {

  // Constructeur constant avec une clé facultative

  const MyApp({super.key});

  // Ce widget est la racine de l'application

  @override

  Widget build(BuildContext context) {

    // Retourne un MaterialApp configuré

    return MaterialApp(

       // Titre de l'application

      title: 'Flutter Application',

      // Thème de l'application avec une palette de couleurs personnalisée

      theme: ThemeData(

        // Utilisation d'un ColorScheme basé sur une couleur de départ

        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),

        // Activation de Material Design 3

        useMaterial3: true,

        ),

      // Désactivation du bandeau "DEBUG"

      debugShowCheckedModeBanner: false,

       initialRoute: '/',

       routes: appRoutes(),  // Utilisation des routes depuis le fichier séparé

    );

  }

}

# Ajouter la route



'/shopping': (context) => const ArticlesListScreen(), // Route pour l'écran ArticlesList



 actions: <Widget>[

         IconButton(

          icon: const Icon(Icons.shopping\_cart),

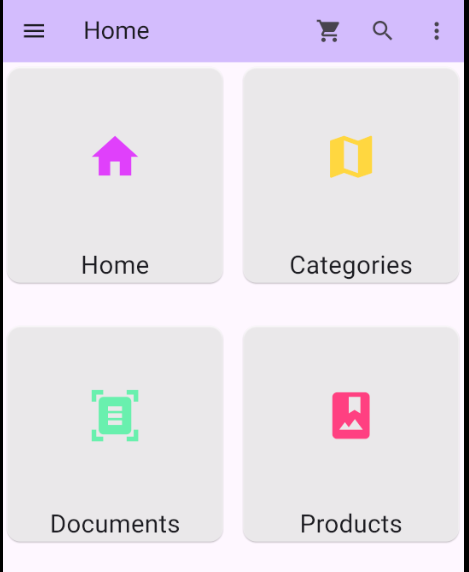
          onPressed: () {

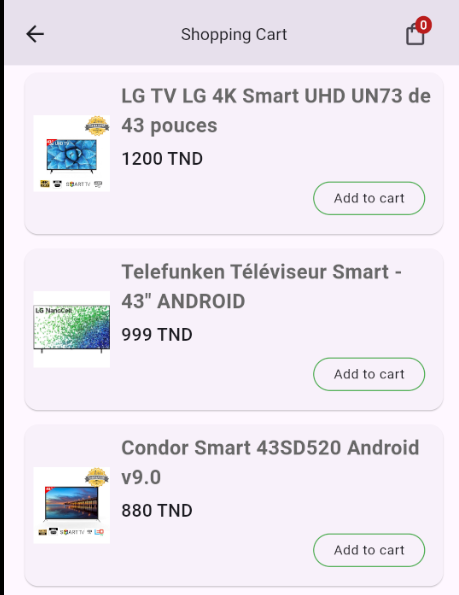
              Navigator.pushNamed(context, '/shopping');

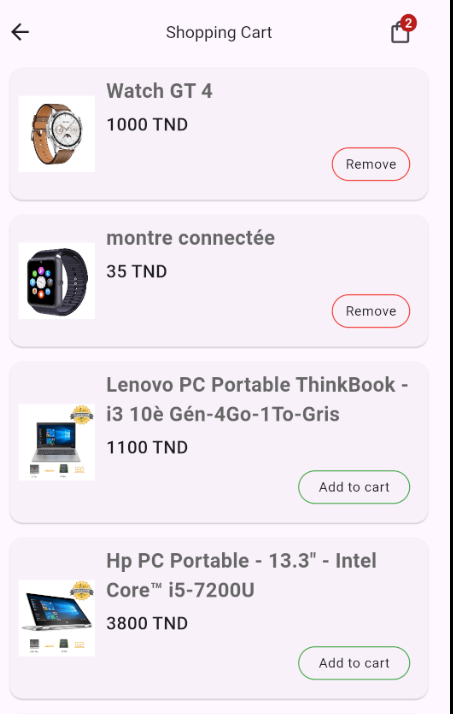
          },

        ),

Résultat







# Le widget de Empty Cart



import 'package:flutter/material.dart';

class EmptyCartMsgWidget extends StatelessWidget {

  const EmptyCartMsgWidget({super.key});

  @override

  Widget build(BuildContext context) {

    Size size = MediaQuery.of(context).size;

    return Center(

        child: SizedBox(

          height: size.height \* .7,

          child: Column(

            mainAxisAlignment: MainAxisAlignment.center,

            children: [

              const Text('Your cart is empty'),

              const SizedBox(height: 20),

              ElevatedButton(

              onPressed: () {

                Navigator.pop(context);

              },

              child: const Text('Shop now')),

        ],

      ),

    ));

  }

}

# Le widget qui affiche et met à jour le Cart Items



import 'package:flutter/material.dart';

import 'package:persistent\_shopping\_cart/model/cart\_model.dart';

import 'package:persistent\_shopping\_cart/persistent\_shopping\_cart.dart';

class CartTitleWidget extends StatelessWidget {

  final PersistentShoppingCartItem data;

  CartTitleWidget({super.key, required this.data});

  final PersistentShoppingCart \_shoppingCart = PersistentShoppingCart();

  @override

  Widget build(BuildContext context) {

    return SingleChildScrollView(

      child: Container(

        padding: const EdgeInsets.all(10),

        margin: const EdgeInsets.only(bottom: 10),

        width: double.infinity,

        decoration: BoxDecoration(

          color: Colors.grey.withOpacity(.05),

          borderRadius: BorderRadius.circular(10),

        ),

        child: Row(

          children: [

            Image.network(

              data.productImages![0].toString(),

              width: 68,

              height: 68,

            ),

            const SizedBox(width: 10),

            Expanded(

              child: Column(

                mainAxisAlignment: MainAxisAlignment.spaceBetween,

                crossAxisAlignment: CrossAxisAlignment.start,

                children: [

                  Text(

                    data.productName,

                    style: const TextStyle(

                        fontSize: 14,

                        fontWeight: FontWeight.w700,

                        color: Color.fromARGB(125, 17, 3, 3)),

                    maxLines: 1, // Add this to prevent overflow

                    overflow: TextOverflow.ellipsis, // Truncate if necessary

                  ),

                  Row(

                    children: [

                      Expanded(

                        child: Text(

                          "${data.unitPrice} TND",

                           style: const TextStyle(

                        fontSize: 16,

                        fontWeight: FontWeight.w700,

                        color: Colors.black

                        ),

                     ),

                      ),

                      const SizedBox(width: 12),

                      InkWell(

                        onTap: () async {

                          bool removed = await \_shoppingCart

                              .removeFromCart(data.productId);

                          if (removed) {

                            showSnackBar(context, removed);

                          }

                        },

                        child: Container(

                          height: 30,

                          width: 70,

                          decoration: BoxDecoration(

                            borderRadius: BorderRadius.circular(20),

                            border: Border.all(color: Colors.red),

                          ),

                          child: Center(

                            child: Text(

                              'Remove',

                              style: Theme.of(context).textTheme.bodySmall,

                            ),

                          ),

                        ),

                      )

                    ],

                  ),

                ],

              ),

            ),

            Column(

              children: [

                InkWell(

                  onTap: () {

                    \_shoppingCart.incrementCartItemQuantity(data.productId);

                  },

                  child: Container(

                    padding: const EdgeInsets.all(2),

                    decoration: BoxDecoration(

                      color: Colors.grey.withOpacity(.2),

                      borderRadius: BorderRadius.circular(6),

                    ),

                    child: const Icon(Icons.add),

                  ),

                ),

                Text(

                  data.quantity.toString(),

                  style: Theme.of(context).textTheme.bodyLarge,

                ),

                InkWell(

                  onTap: () {

                    \_shoppingCart.decrementCartItemQuantity(data.productId);

                  },

                  child: Container(

                    padding: const EdgeInsets.all(2),

                    decoration: BoxDecoration(

                      color: Colors.grey.withOpacity(.2),

                      borderRadius: BorderRadius.circular(6),

                    ),

                    child: const Icon(Icons.remove),

                  ),

                )

              ],

            ),

          ],

        ),

      ),

    );

  }

  void showSnackBar(BuildContext context, bool removed) {

    final snackBar = SnackBar(

      content: Text(

        removed

            ? 'Product removed from cart.'

            : 'Product not found in the cart.',

      ),

      backgroundColor: removed ? Colors.red : Colors.green,

      duration: const Duration(seconds: 1),

    );

    ScaffoldMessenger.of(context).showSnackBar(snackBar);

  }

}

# Le widget qui affiche le Cart



import 'package:flutter/material.dart';

import 'package:myflutterapplication/presentation/widgets/cart/carttitle.widget.dart';

import 'package:myflutterapplication/presentation/widgets/cart/emptycart.widget.dart';

import 'package:persistent\_shopping\_cart/persistent\_shopping\_cart.dart';

class CartViewItem extends StatefulWidget {

  const CartViewItem({super.key});

  @override

  State<CartViewItem> createState() => \_CartViewState();

}

class \_CartViewState extends State<CartViewItem> {

  @override

  Widget build(BuildContext context) {

    return SafeArea(

      child: Padding(

        padding: const EdgeInsets.symmetric(horizontal: 15),

        child: Column(

          children: [

            Expanded(

              child: PersistentShoppingCart().showCartItems(

                cartTileWidget: ({required data}) =>

                    CartTitleWidget(data: data),

                showEmptyCartMsgWidget: const EmptyCartMsgWidget(),

              ),

            ),

            PersistentShoppingCart().showTotalAmountWidget(

              cartTotalAmountWidgetBuilder: (totalAmount) => Visibility(

                visible: totalAmount == 0.0 ? false : true,

                child: Column(

                  children: [

                    Row(

                      mainAxisAlignment: MainAxisAlignment.spaceBetween,

                      children: [

                        const Text(

                          'Total',

                          style: TextStyle(

                              color: Color.fromARGB(255, 113, 107, 107),

                              fontSize: 22),

                        ),

                        Text(

                          "$totalAmount TND",

                          style: const TextStyle(

                              color: Color.fromARGB(255, 143, 133, 133),

                              fontSize: 22),

                        ),

                      ],

                    ),

                    ElevatedButton(

                        onPressed: () {}, child: const Text('Checkout'))

                  ],

                ),

              ),

            )

          ],

        ),

      ),

    );

  }

}

# Show cart screen

import 'package:flutter/material.dart';

import 'package:myflutterapplication/presentation/widgets/cart/showcartitem.widget.dart';

class CartView extends StatelessWidget {

  const CartView({super.key});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: const Text('My Cart')),

      body: const CartViewItem()

    );

  }

}

# La route



'/cartView': (context) => const CartView(), // Route pour l'écran cart view

Résultat

