

Rapport eMaya

Léandro VEYRENC-CORDERO

Sommaire :

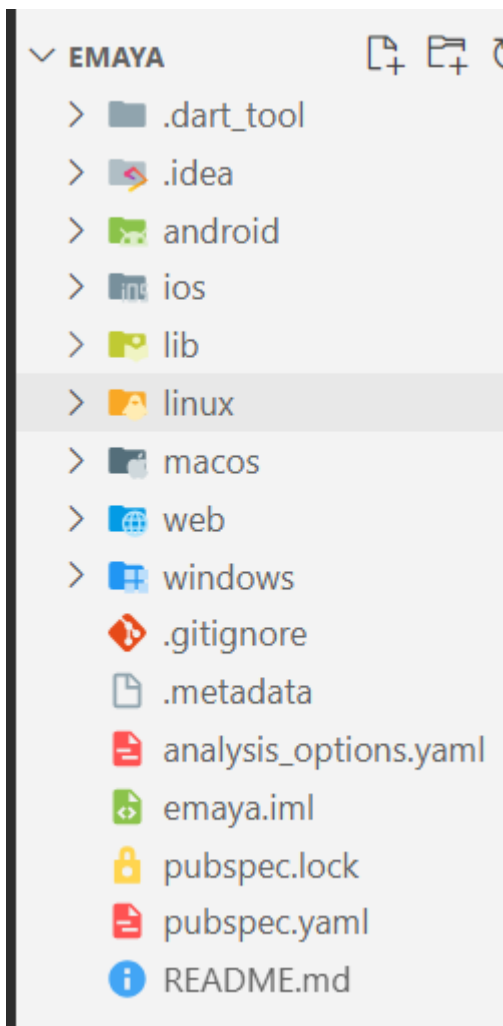
- [Étape 1 : Architecture, Thème et Navigation de base](#)
- [Étape 2 : Présentation de la page Catégories \(UI & Jeu d'essai\)](#)
- [Étape 3 : Consommation de l'API REST pour les Catégories](#)
- [Étape 4 : Présentation de la page Panier](#)
- [Étape 5 : Présentation de la page Produits en vente](#)

Rapport eMaya

Léandro VEYRENC-CORDERO

Étape 1 : Architecture, Thème et Navigation de base

L'objectif initial était de mettre en place une structure solide pour une application mobile cross-platform utilisant le SDK Flutter.



1.1 Mise en place de l'architecture

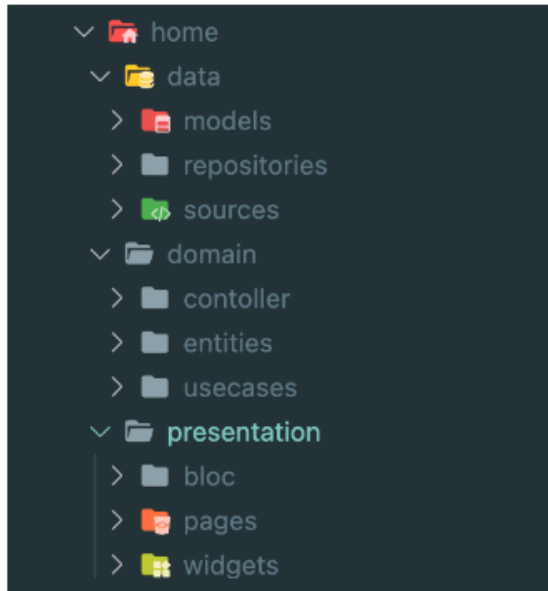
Pour garantir la maintenabilité et l'évolutivité, le projet utilise une **version simplifiée de la Clean Architecture**.

- **Installation** : Utilisation de l'extension VS Code "Clean Flutter Architecture" pour générer les dossiers **core** (éléments globaux) et **features** (fonctionnalités spécifiques).
- **Structure générée** : La logique est séparée en trois couches : *Data* (sources de données), *Domain* (entités et services) et *Presentation* (UI et widgets).

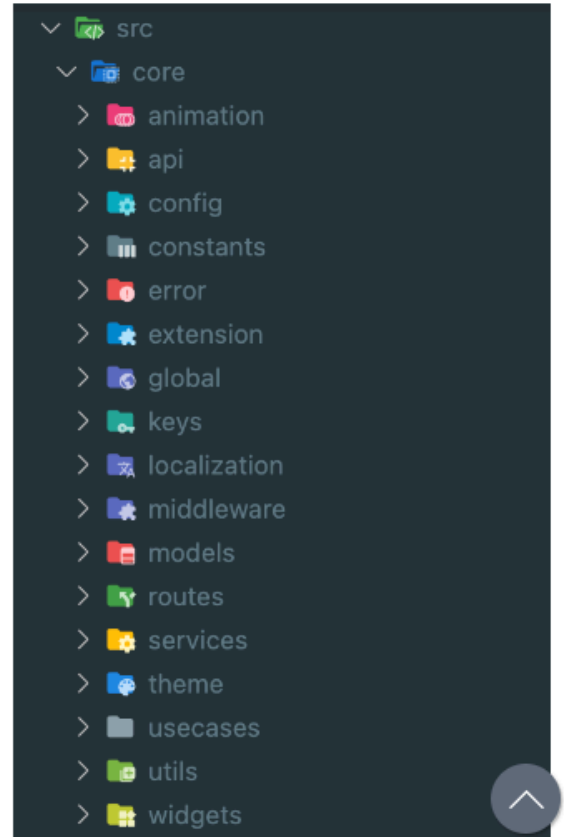
Rapport eMaya

Léandro VEYRENC-CORDERO

Feature Folder



Core Folder



1.2 Gestion des thèmes (Clair et Sombre)

L'application centralise ses styles pour respecter le principe **DRY (Don't Repeat Yourself)**.

- **Couleurs** : Définition de constantes dans `colors.dart` pour les modes light et dark.

Rapport eMaya

Léandro VEYRENC-CORDERO

Fichier `lib/src/core/constants/colors.dart`

```
library app_colors;

import 'package:flutter/material.dart';

class AppColor {
  static Color black = const Color(0xFF000000);
  static Color white = const Color(0xFFFFFFFF);

  // thème light
  static Color lightBackgroundColor = const Color(0xFFFFCF8EC);
  static Color lightPrimaryColor = const Color.fromARGB(255, 83, 146, 58);
  static Color lightCardColor = const Color.fromARGB(255, 216, 244, 192);
  static Color lightCanvasColor = Colors.white;

  // thème dark
  static Color darkBackgroundColor = const Color(0xFF00001a);
  static Color darkPrimaryColor = const Color.fromARGB(255, 83, 146, 58);
  static Color darkCardColor = const Color.fromARGB(255, 13, 124, 47);
  static Color darkCanvasColor = Colors.black;
}
```

- **Package Adaptive Theme** : Installation du package `adaptive_theme` pour permettre la bascule dynamique entre les thèmes et la persistance du choix de l'utilisateur via les *shared preferences*.

Rapport eMaya

Léandro VEYRENC-CORDERO

Flutter lib/src/core/theme/theme.dart

```
library theme;

import 'package:emaya/src/core/constants/colors.dart';
import 'package:emaya/src/core/constants/constants.dart';
import 'package:flutter/material.dart';

class AppTheme {
  static ThemeData lightThemeData = ThemeData(
    useMaterial3: true,
    scaffoldBackgroundColor: AppColor.lightBackgroundColor,
    primaryColor: AppColor.lightPrimaryColor,
    textTheme: TextTheme(
      labelSmall: TextStyle(color: AppColor.black),
      labelMedium: TextStyle(color: AppColor.black),
      labelLarge: TextStyle(color: AppColor.black),
    ),
    cardColor: AppColor.lightCardColor,
    canvasColor: AppColor.lightCanvasColor,
  );

  static ThemeData darkThemeData = ThemeData(
    useMaterial3: true,
    scaffoldBackgroundColor: AppColor.darkBackgroundColor,
    textTheme: TextTheme(
      labelSmall: TextStyle(color: AppColor.white),
      labelMedium: TextStyle(color: AppColor.white),
      labelLarge: TextStyle(color: AppColor.white),
    ),
    colorScheme: ThemeData().colorScheme.copyWith(
      secondary: const Color.fromARGB(255, 136, 145, 193),
      brightness: Brightness.dark,
    ),
    cardColor: AppColor.darkCardColor,
    canvasColor: AppColor.darkCanvasColor,
  );
}
```

Rapport eMaya

Léandro VEYRENC-CORDERO

Fichier `lib/root_app.dart`

```
import 'package:emaya/src/core/theme/theme.dart';
import
'package:emaya/src/features/categorie/presentation/pages/categories_page.dart';
import 'package:flutter/material.dart';

import 'package:adaptive_theme/adaptive_theme.dart';

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

  @override
  Widget build(BuildContext context) {
    return AdaptiveTheme(
      light: AppTheme.lightThemeData,
      dark: AppTheme.darkThemeData,
      initial: AdaptiveThemeMode.light, // thème initial
      builder: (theme, darkTheme) => MaterialApp(
        debugShowCheckedModeBanner: true,
        title: 'Maya',
        theme: theme,
        darkTheme: darkTheme,
        home: const CategoriePage(),
      ),
    );
  }
}
```

Rapport eMaya

Léandro VEYRENC-CORDERO

```
Fichier lib/src/features/user/presentation/pages/user_page.dart
import 'package:adaptive_theme/adaptive_theme.dart';
import 'package:flutter/material.dart';

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

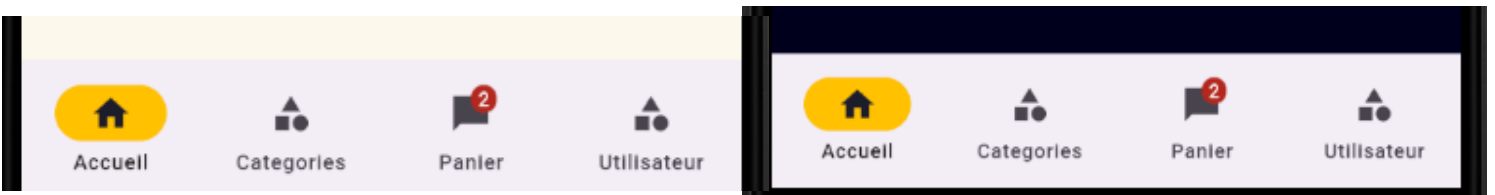
  @override
  State<UserPage> createState() => _UserPageState();
}

class _UserPageState extends State<UserPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: SwitchListTile(
          title: Text('Mode sombre',
            style: Theme.of(context).textTheme.labellarge),
          secondary: Icon(
            Icons.dark_mode_outlined,
            color: Theme.of(context).textTheme.labellarge!.color!,
          ),
          onChanged: (bool value) {
            setState(() {
              AdaptiveTheme.of(context).toggleThemeMode();
            });
          },
          value: AdaptiveTheme.of(context).mode.isDark,
        ),
      ),
    );
  }
}
```



1.3 Navigation initiale

- **Navigation Bar** : Mise en œuvre d'une `NavigationBar` dans la page `HomePage` pour naviguer entre Accueil, Catégories, Panier et Utilisateur.
- **Navigator** : Utilisation du widget `Navigator` pour la gestion de la pile d'écrans, adaptée aux petites applications.



Rapport eMaya

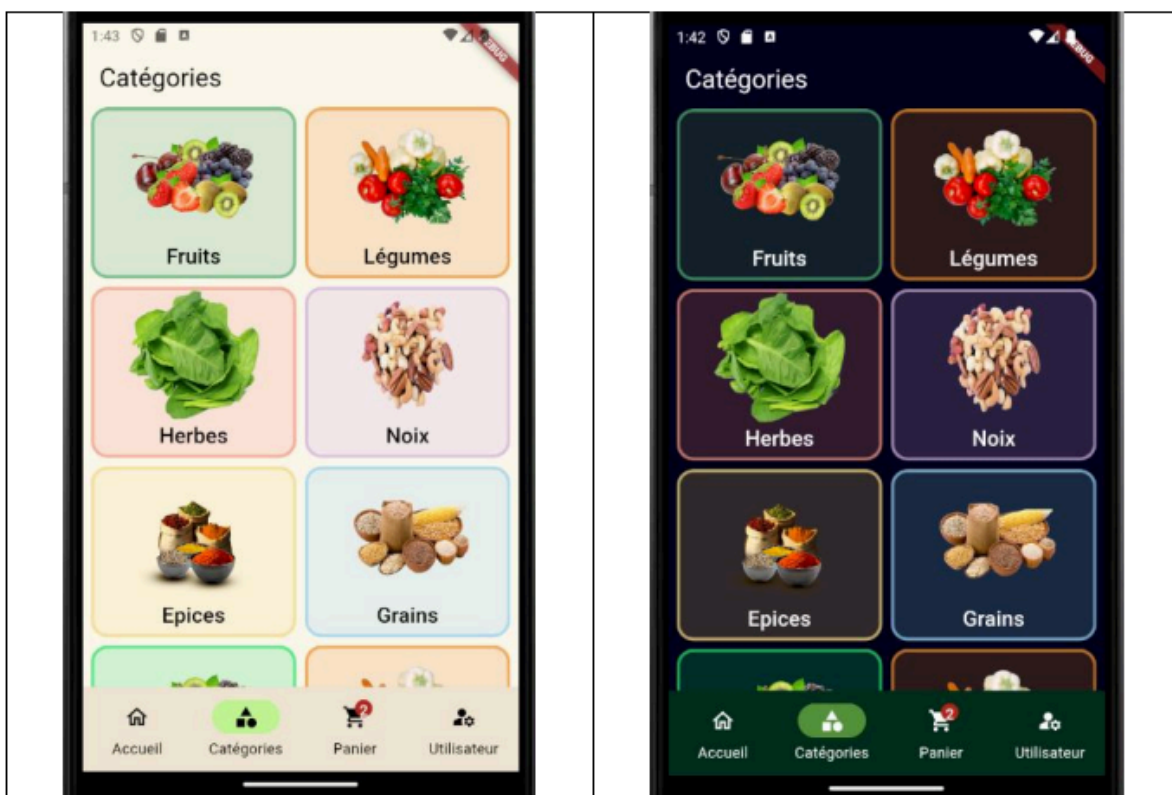
Léandro VEYRENC-CORDERO

Étape 2 : Présentation de la page Catégories

Cette étape s'est concentrée sur la création de l'interface utilisateur de la page des catégories.

2.1 Design de l'UI

- **GridView** : Utilisation d'un `GridView.builder` pour afficher les catégories sous forme de grille avec deux colonnes.
- **Stylisation** : Chaque catégorie est présentée dans un `InkWell` (pour la détection du toucher) contenant un `Container` décoré avec une image et un libellé.



2.2 Jeu d'essai (Fake Data)

Avant l'intégration de l'API, une entité `CategorieEntityFake` et un repository `CategorieRepositoryFake` ont été créés pour simuler des données "en dur".

Étape 3 : Consommation de l'API REST

L'étape suivante a consisté à remplacer les données simulées par des données réelles provenant de l'API `maya-api`.

3.1 Configuration de l'environnement

Rapport eMaya

Léandro VEYRENC-CORDERO

- **Package http** : Ajout de la dépendance pour effectuer des requêtes réseau

```
flutter pub add http
```

- **AppConfig** : Création d'une classe de configuration utilisant `String.fromEnvironment` pour définir l'URL de base de l'API et des images.

```
Fichier lib/src/core/config/app_config.dart
abstract class AppConfig {
  // URL de base pour appel API maya-api
  // ignore: constant_identifier_names
  static const String API_BASE_URL = String.fromEnvironment(
    'API_BASE_URL',
    defaultValue: 'http://10.0.2.2:8000/api/',
  );

  // URL de base pour chargement images de API maya-api
  // ignore: constant_identifier_names
  static const String API_BASE_URL_IMG = String.fromEnvironment(
    'API_BASE_URL_IMG',
    defaultValue: 'http://10.0.2.2/maya-api/public',
  );
}
```

3.2 Modélisation et Services

Rapport eMaya

Léandro VEYRENC-CORDERO

- **Modèle** : Création de la classe `Categorie` avec des méthodes de sérialisation JSON (`fromJson`, `toJson`).

```
Fichier lib/src/features/categorie/data/models/categorie.dart
import 'dart:convert';

List<Categorie> categoriesFromJson(String str) =>
  List<Categorie>.from(json.decode(str).map((x) => Categorie.fromJson(x)));

Categorie categorieFromJson(String str) => Categorie.fromJson(json.decode(str));

class Categorie {
  int id;
  String libelle;
  String imageUrl;

  Categorie({
    required this.id,
    required this.libelle,
    required this.imageUrl,
  });

  factory Categorie.fromJson(Map<String, dynamic> json) => Categorie(
    id: json["id"],
    libelle: json["libelle"],
    imageUrl: json["imageUrl"],
  );

  Map<String, dynamic> toJson() => {
    "id": id,
    "libelle": libelle,
    "imageUrl": imageUrl,
  };
}
```

Rapport eMaya

Léandro VEYRENC-CORDERO

- **Data Source (CategorieApi)** : Responsable de l'envoi de la requête GET et du décodage UTF-8 des données.

```
Fichier lib/src/features/categorie/data/sources/categorie_api.dart
import 'dart:convert';
import 'package:emaya/src/core/config/app_config.dart';
import 'package:emaya/src/features/categorie/data/models/categorie.dart';
import 'package:http/http.dart' as http;

//
// classe qui est responsable de la récupération des données à partir de l'API
REST

// Le tableau de catégories est ensuite retourné (List<Categorie> est le
type de retour de fonction).
return categoriesFromJson(
    const Utf8Decoder().convert(response.bodyBytes));
} else {
    throw Exception('Erreur lecture catégories : ${response.body}');
}
}
}
```

- **Service (CategorieService)** : Sert d'intermédiaire entre l'UI et l'API.

```
Fichier lib/src/features/categorie/domain/service/categorie_service.dart
import 'package:emaya/src/features/categorie/data/models/categorie.dart';
import 'package:emaya/src/features/categorie/data/sources/categorie_api.dart';

// classe qui sert d'intermédiaire entre l'interface utilisateur et l'API
class CategorieService {
    final _api = CategorieApi();

    Future<List<Categorie>?> getAllCategories() async {
        return _api.getAllCategories();
    }
}
```

Rapport eMaya

Léandro VEYRENC-CORDERO

```
@override
Widget build(BuildContext context) {

  return Scaffold(
    appBar: AppBar(
      title: const Text('Catégories'),
    ),
    body: isLoading && lesCategories != null
      ? SafeArea(
        child: Padding(
          padding: const EdgeInsets.only(left: 8.0, right: 8.0),
          child: GridView.builder(
            gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
              crossAxisCount: 2,
              mainAxisSpacing: 8,
              crossAxisSpacing: 8,
              childAspectRatio: 300 / 250,
            ),
            itemCount: lesCategories!.length,
            itemBuilder: (_, int index) =>
              CategorieWidget(categorie: lesCategories![index], indCategorie: index),
          ),
        ),
      )
      : const Center(child: CircularProgressIndicator()),
  );
}
```

Étape 4 : Présentation de la page Panier

Le développement de la page Panier a introduit des interactions plus complexes sur les éléments.

4.1 Structure de la page

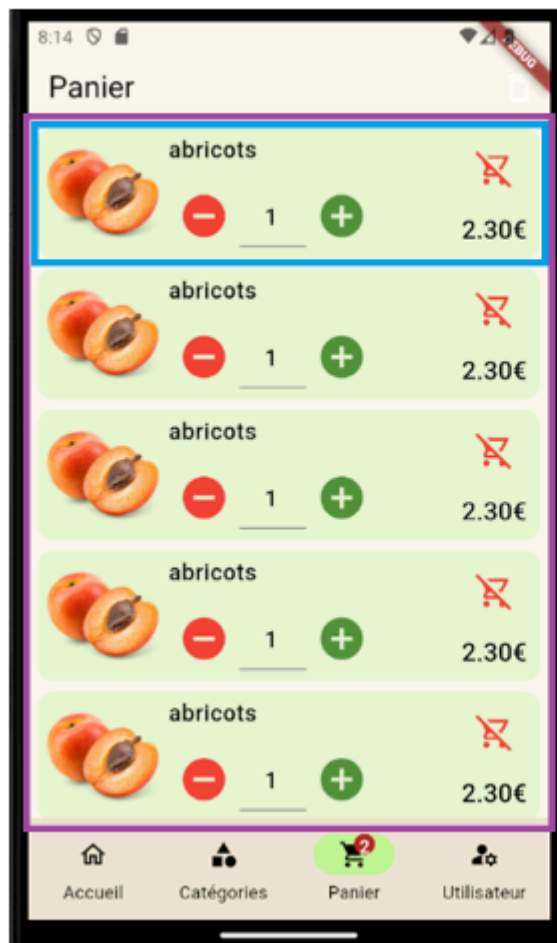
Rapport eMaya

Léandro VEYRENC-CORDERO

- **AppBar** : Ajout d'un `IconButton` avec l'icône `delete_outlined` pour permettre de vider le panier.

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: const Text('Panier'), actions: [
      IconButton(
        onPressed: () {},
        icon: const Icon(
          Icons.delete,
        ), // Icon
      ), // IconButton
    ]), // AppBar
    body: Padding(
```

- **ListView** : Affichage scrollable des produits du panier via un `ListView.builder`.



un élément (item) qui sera développé dans le widget `ItemPanierWidget`

ListView : tableau de widgets scrollable
Il est dans un widget `Column`.

Rapport eMaya

Léandro VEYRENC-CORDERO

4.2 Widget ItemPanierWidget

Chaque ligne du panier est gérée par un widget spécifique permettant :

- **Incrémentation/Décrémentation** : Gestion des quantités via des boutons "+" et "-" avec une limite minimale de 1.
- **Saisie numérique** : Utilisation d'un `TextField` avec `inputFormatters` pour restreindre la saisie aux chiffres uniquement.

```
1 import 'package:flutter/material.dart';
2
3 class ItemPanierWidget extends StatefulWidget {
4   const ItemPanierWidget({super.key});
5
6   @override
7   State<ItemPanierWidget> createState() => _ItemPanierWidgetState();
8 }
9
10 class _ItemPanierWidgetState extends State<ItemPanierWidget> {
11   @override
12   Widget build(BuildContext context) {
13     return const Placeholder();
14   }
15 }
```

Fichier lib/src/features/panier/presentation/widgets/item_panier_widget.dart

```
...
class _ItemPanierWidgetState extends State<ItemPanierWidget> {
  @override
  Widget build(BuildContext context) {
    Size size = MediaQuery.of(context).size;
    return GestureDetector(
      onTap: () {},
      child: const Text('produit du panier'),
    );
  }
}
...
```

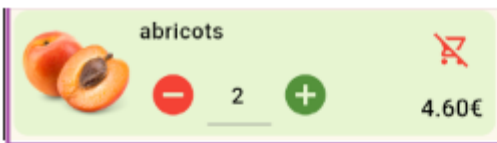
Rapport eMaya

Léandro VEYRENC-CORDERO

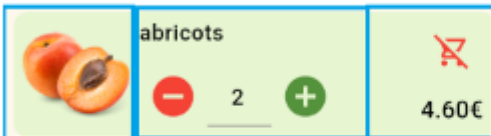
Fichier lib/src/features/panier/presentation/pages/panier_page.dart

```
...
body: Padding(
  padding: const EdgeInsets.only(left: 8.0, right: 8.0),
  child: Column(
    children: [
      Expanded(
        child: ListView.builder(
          itemCount: 8,
          itemBuilder: (ctx, index) {
            return const ItemPanierWidget();
          },
        ),
      ),
    ],
  ),
),
...

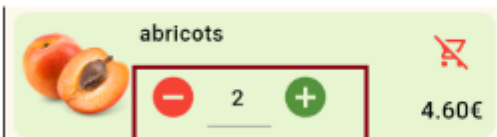
```



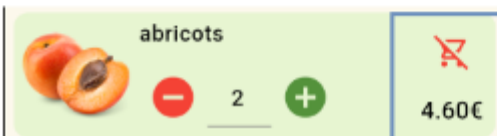
Row



3 children principaux dans la Row :
- Container avec image
- Column
- Column



Row avec
- IconButton -
- TextField
- IconButton +



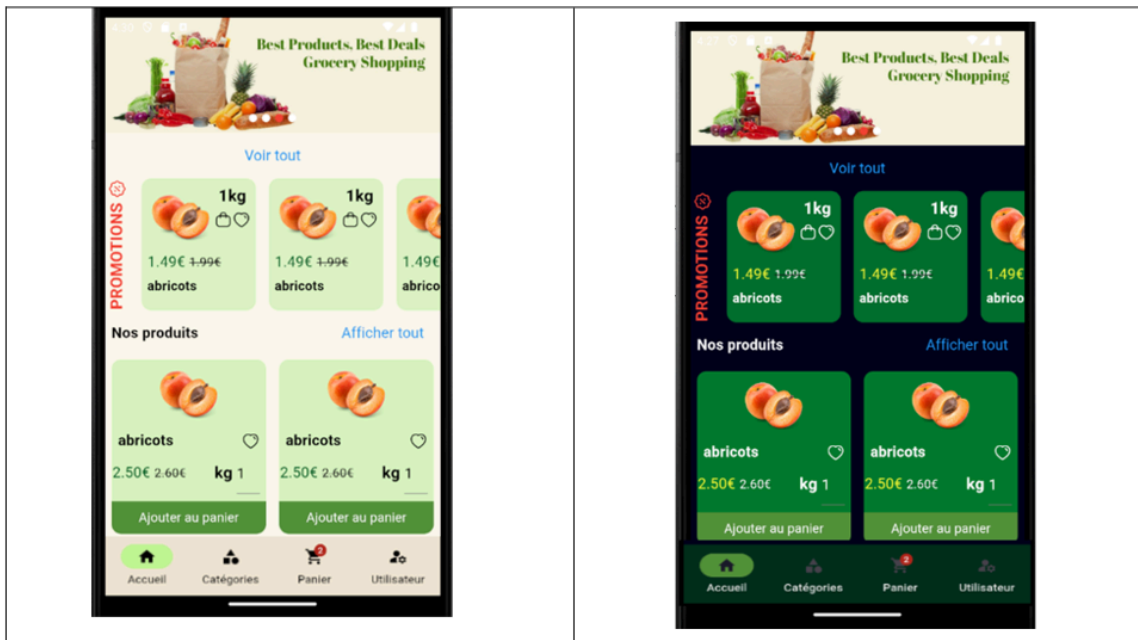
Column avec
- IconButton
- Text

Rapport eMaya

Léandro VEYRENC-CORDERO

Étape 5 : Présentation de la page Produits en vente

Cette page constitue l'écran d'accueil principal, combinant plusieurs types de widgets de présentation.



Rapport eMaya

Léandro VEYRENC-CORDERO



5.1 Carrousel d'images

Utilisation du package `card_swiper` pour afficher un carrousel d'images de la ferme en haut de la page.

5.2 Sections Promotion et Vente

- **Promotions** : Utilisation d'un `ListView` horizontal précédé d'un `RotatedBox` pour afficher le texte "PROMOTIONS" à la verticale.

Rapport eMaya

Léandro VEYRENC-CORDERO

Fichier `lib/src/features/produit/presentation/widgets/produit_promotion_widget.dart`

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

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

  @override
  State<ProduitPromotionWidget> createState() => _ProduitPromotionWidgetStateState();
}

class _ProduitPromotionWidgetStateState extends State<ProduitPromotionWidget> {
  @override
  Widget build(BuildContext context) {
    return const Placeholder(
      child: Text('promotion')
    );
  }
}
```

Fichier `lib/src/features/produit/presentation/pages/promotions_page.dart`

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

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

  @override
  Widget build(BuildContext context) {
    return const Text('Promotions page');
  }
}
```

D

- **Produits standard** : Utilisation d'un `GridView.builder` avec l'option `shrinkWrap: true` pour s'intégrer dans le scroll principal de la page (`SingleChildScrollView`).

Rapport eMaya

Léandro VEYRENC-CORDERO

Fichier lib/src/features/produit/presentation/widgets/produit_vente_widget.dart

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

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

  @override
  State<ProduitVenteWidget> createState() => _ProduitVenteWidgetState();
}

class _ProduitVenteWidgetState extends State<ProduitVenteWidget> {
  @override
  Widget build(BuildContext context) {
    return const Placeholder(
      child: Text('vente'),
    );
  }
}
```

Fichier lib/src/features/produit/presentation/pages/catalogue_page.dart

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

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

  @override
  Widget build(BuildContext context) {
    return const Text('Catalogue page');
  }
}
```

- **Widgets dédiés** : Création de `ProduitPromotionWidget` et `ProduitVenteWidget` pour encapsuler le design des cartes produits (images, prix barrés, boutons d'ajout au panier).

Rapport eMaya

Léandro VEYRENC-CORDERO

Etape 6 : Synchronisation des Produits avec l'API

Le catalogue de produits a été rendu dynamique via l'API.

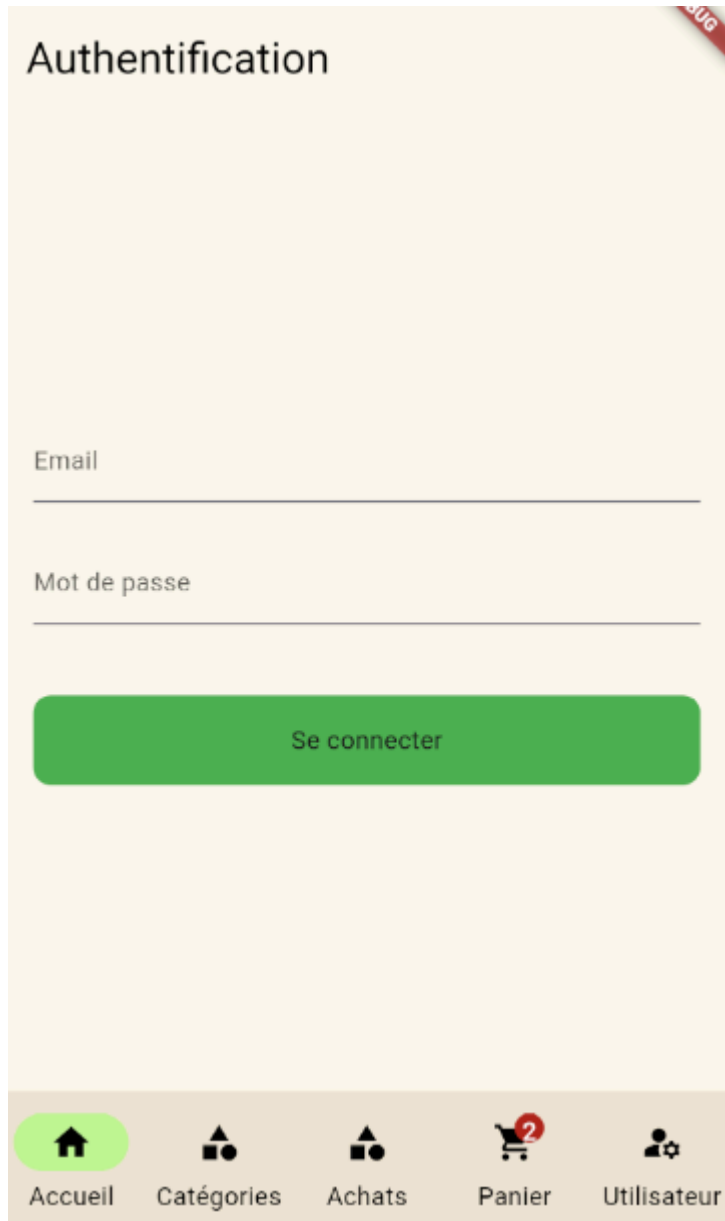
- **Modèle Produit** : Création de la classe `Produit` en omettant les champs inutiles (comme la date de création) pour optimiser les performances.
- **Tri des données** : Modification de l'API (`maya-api`) pour garantir que les produits sont systématiquement renvoyés par ordre alphabétique via l'annotation `ApiResponse(order: ['libelle' => 'ASC'])`.

Rapport eMaya

Léandro VEYRENC-CORDERO

Etape 7 : Système d'Authentification

L'étape majeure a été la sécurisation de l'accès.



- **Stockage sécurisé** : Utilisation de `flutter_secure_storage` pour conserver le jeton (token) d'authentification de manière chiffrée sur l'appareil.
- **Login** : Création d'une page `LoginPage` gérant le formulaire de connexion.
- **Feedback** : Utilisation de `fluttertoast` pour informer l'utilisateur du succès ou de l'échec de la connexion.
- **Logique de connexion** : Adaptation de la page d'accueil pour rediriger automatiquement l'utilisateur vers le login s'il n'est pas authentifié.