M

PROJET FLUTTER

APPLICATION E-COM

Catalogue • Favoris • Panier • Profil • Historique

Flutter • Dart • Provider • GoRouter • Stockage local • Déploiem

SYNTHÈSE RAPIDE

- Application Flutter de type marketplace
- Parcours utilisateur complet
- Gestion du panier et des favoris
- Historique d'achat intégré
- Architecture claire pour la soutenance

FONCTIONNALITÉS PRINCIPALES

ONBOARDING
Présentation de l'application

CONNEXION
Authentification utilisateur

CATALOGUE
Produits et recherche

FAVORIS
Sauvegarde persistante

PANIER
Ajout / quantité / achat

HISTORIQUE
Commandes passées

PROFIL
Espace personnel

BESOIN CLIENT

Créer une application e-commerce moderne, lisible et évolutive permettant de parcourir des produits, gérer des favoris, valider un panier et retrouver l'historique d'achat.

- Parcours utilisateur simple et fluide
- Interface compatible web / mobile
- Gestion locale des données utilisateur
- Base technique claire pour évoluer

OBJECTIFS DU PROJET

- Offrir une expérience d'achat agréable
- Structurer le code Flutter proprement
- Assurer la persistance des favoris / panier
- Préparer une mise en ligne propre
- Rendre le projet maintenable et démontrable

TECHNOLOGIES UTILISÉES

- Flutter — UI framework
- Dart — Langage
- Provider — State management
- GoRouter — Navigation
- Local Stora — Persistance
- Hostinger — Hébergement

PARCOURS UTILISATEUR

1 Découverte → 2 Connexion → 3 Panier → 4 Achat → 5 Historique

CONTRAINTES & EXIGENCES

- Navigation structurée par routes
- Stockage local : onboarding, favoris, panier
- Écrans clairs et cohérents
- Compatibilité déploiement web
- Démonstration exploitable en soutenance

ÉCRANS CLÉS

- Onboarding
- Catalogue produits
- Connexion
- Détail produit
- Panier & historique

ARCHITECTURE

- main.dart
- providers
- services
- repositories
- router / screens / models

ÉVOLUTIONS FUTURES

- Connexion API réelle
- Paiement en ligne
- Gestion comptes avancée
- Back-office
- Notifications

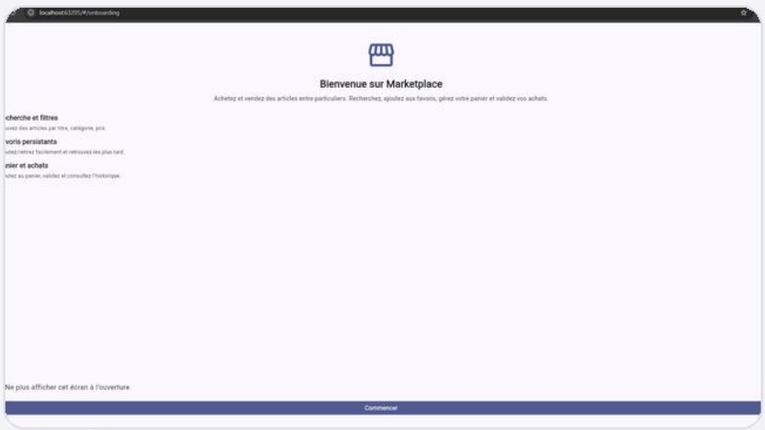
VALIDATION

- Parcours complet OK
- Ajout panier OK
- Historique OK
- Favoris persistants
- Projet prêt à évoluer

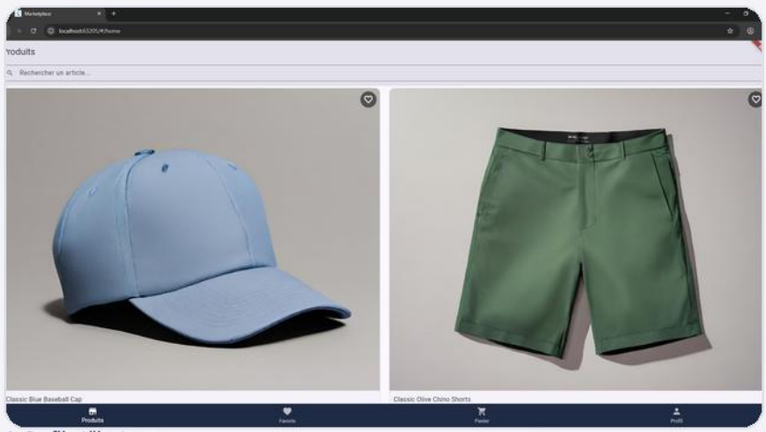
PAGE D'APPUI - CAPTURES DÉTAILLÉES

Lecture agrandie des écrans principaux du projet

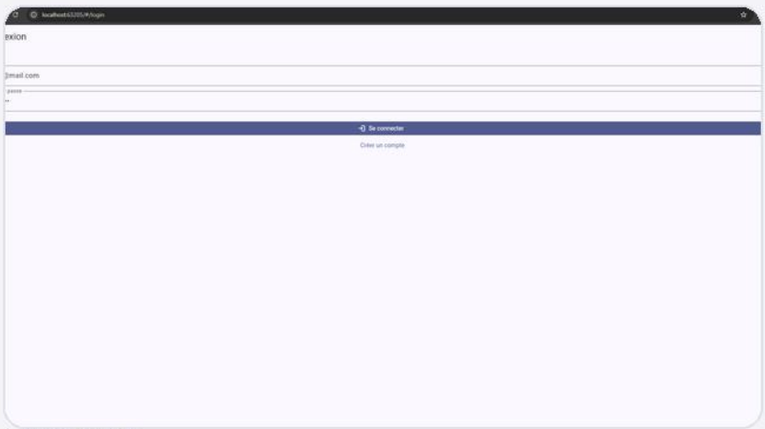
1. Onboarding



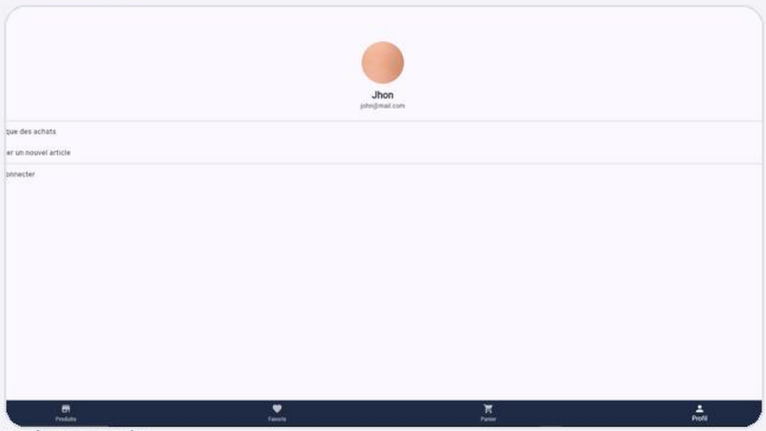
2. Catalogue produits



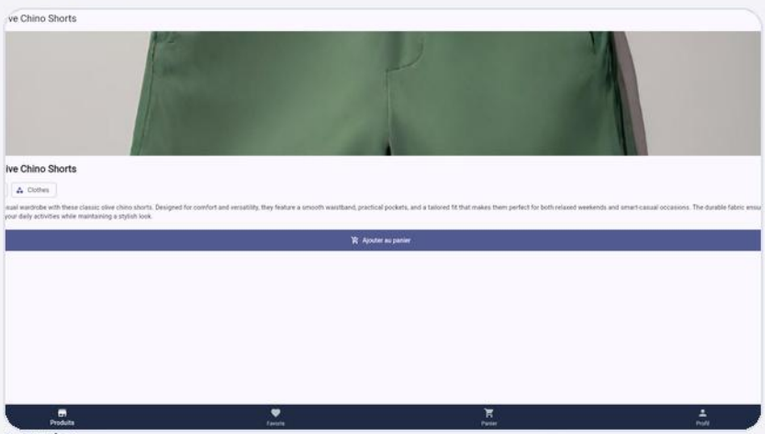
3. Connexion



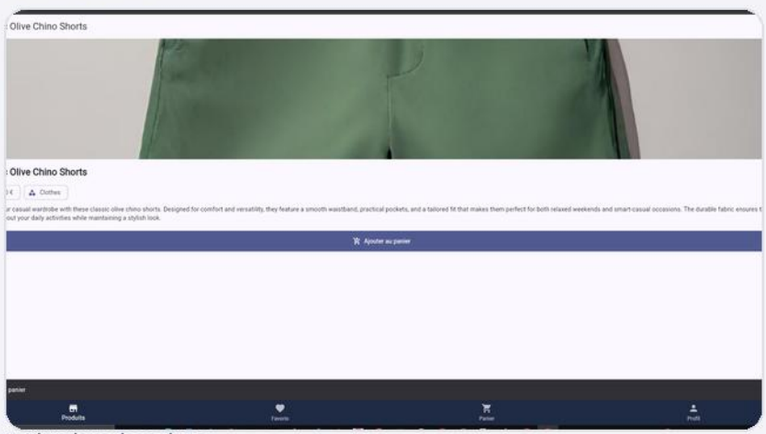
4. Profil utilisateur



5. Détail produit



6. Ajout au panier



7. Panier

8. Historique des achats

Annexe visuelle - captures agrandies pour mieux voir les fonctionnalités

ARCHITECTURE TECHNIQUE - PROJET FLUTTER

Structure claire du projet pour la soutenance et la documentation

STRUCTURE DU PROJET

main.dart	Point d'entrée de l'application, initialise le projet et les providers
core/	Constantes, configuration globale et éléments transverses
services/	Communication API et services métier
repositories/	Persistance locale / abstraction des sources de données
providers/	Gestion d'état (panier, favoris, auth, produits, onboarding)
router/	Navigation centralisée entre les écrans
screens / widgets / models	Interface utilisateur, composants réutilisables et modèles de données

LECTURE DE L'ARCHITECTURE

- main.dart constitue le point d'entrée global de l'application
- Les providers centralisent la gestion d'état (authentification, produits, panier, favoris, commandes)
- Le router organise la navigation entre les écrans principaux
- Les screens affichent l'interface visible par l'utilisateur
- Les services gèrent la logique métier et les appels éventuels
- Les repositories gèrent la persistance et la récupération des données
- Cette structure facilite la maintenance, les tests et l'évolution du projet

PROVIDERS PRINCIPAUX

- OnboardingProvider
- AuthProvider
- ProductProvider
- FavoritesProvider
- CartProvider
- OrdersProvider

PARCOURS TECHNIQUE

- L'application démarre dans main.dart
- Les providers sont injectés globalement au lancement
- GoRouter gère les routes principales de navigation
- Les écrans consomment les données exposées par les providers
- Les services / repositories gèrent la logique et la persistance
- Le stockage local permet de conserver l'état utilisateur entre sessions
- L'architecture reste modulaire et facile à présenter en soutenance

POINTS FORTS

- Architecture lisible
- Séparation claire des responsabilités
- Facile à expliquer en soutenance
- Facile à faire évoluer
- Base stable pour la suite
- Organisation professionnelle du code
- Bonne lisibilité pour la maintenance

CONCLUSION TECHNIQUE

Cette architecture Flutter met en avant une organisation claire du projet, une séparation nette des responsabilités et une logique facilement compréhensible pour une présentation de type BTS SIO SLAM. La structure retenue rend l'application évolutive, maintenable et adaptée à une démonstration technique en rapport ou en soutenance.

Page architecture simplifiée - sans schéma de flux ni preuve d'environnement