

UNIVERSITÉ VIRTUELLE DE BURKINA FASO

(UV-BF)

RAPPORT DE PROJET

Application E-Commerce Flutter

Auteur :

Bikienga Hamza

Kassongo Moussa

Ouattara Mohamed Djalilou

Filière : Génie Logiciel

Niveau : Licence 3

Développement Web & Mobile Année académique : 2024– 2025

Tuteur :

Table des matières

| | |
|--------------------------------------|---|
| 1. Introduction | 2 |
| 1.1. Contexte du projet..... | 2 |
| 1.2. Objectifs du projet..... | 2 |
| 1.3. Périmètre du projet..... | 3 |
| 2. ANALYSE ET SPÉCIFICATION | 3 |
| 2.1. Acteurs du système..... | 3 |
| 2.2. Analyse fonctionnelle..... | 3 |
| 2.3. Règles de gestion | 3 |
| 3. CONCEPTION ET ARCHITECTURE | 4 |
| 3.1. Architecture logicielle..... | 4 |
| 3.2. Architecture technique | 4 |
| 3.3. Conception UI/UX..... | 4 |
| 4. CHOIX TECHNOLOGIQUES..... | 6 |
| 4.1.Outils de développement | 6 |
| 5.RÉALISATION ET IMPLÉMENTATION..... | 6 |
| 5.1.Structure de la solution | 6 |

| | |
|---|---|
| 5.2. Développement de la couche de Données (Mock Service) | 6 |
| 5.3. Développement Frontend | 7 |
| 6. TESTS ET RÉSULTATS..... | 7 |
| 6.1. Stratégie de tests..... | 7 |
| 7. DIFFICULTÉS ET SOLUTIONS | 7 |
| 7.1. Problèmes de navigation (Flutter Navigator) | 7 |
| 7.2. Erreurs de cache et Assets..... | 7 |
| 7.3. Gestion du panier et recalcul des totaux | 8 |
| 7.4. Filtrage des produits | 8 |
| 7.5. Upload et affichage des images | 8 |
| 7.6. Protection des routes | 8 |
| 7.7. Optimisation de performance sur Android | 8 |
| 8. CONCLUSION ET PERSPECTIVES | 9 |
| 8.1. Bilan du projet..... | 9 |
| 8.2. Améliorations futures | 9 |
| 8.3. Apports personnels | 9 |

1.Introduction

1.1.Contexte du projet

Dans le cadre de notre formation en développement mobile, il nous a été demandé de réaliser une application native performante. Le choix s'est porté sur le framework **Flutter** de Google pour sa capacité à produire des interfaces soignées et fluides. Ce projet simule une boutique en ligne complète sans dépendance à une API externe (Mock).

1.2.Objectifs du projet

L'objectif principal est de maîtriser les concepts avancés de Flutter, notamment :

- La manipulation de listes complexes (GridView).
- La gestion d'état avancée (State Management) pour synchroniser les données entre les écrans.
- La mise en place d'une interface utilisateur (UI) riche avec des animations (Hero, AnimatedContainer).
- La simulation d'un parcours client complet (De la sélection à l'achat).

1.3. Périmètre du projet

Le projet se limite à la partie "Client" de l'application mobile (Android/iOS).

- **Inclus :** Navigation, Catalogue, Panier, Fiches produits, Simulation de données.
- **Exclus :** Base de données distante, système de paiement réel, interface administrateur.

2. ANALYSE ET SPÉCIFICATION

2.1. Acteurs du système

- **L'Utilisateur (Client) :** Il navigue dans le catalogue, ajoute des produits au panier et simule une commande.
- **Le Système (Application) :** Il gère la logique d'affichage, calcule les totaux du panier et assure les transitions.

2.2. Analyse fonctionnelle

Le système permet de :

- **Consulter :** Voir la liste des produits sous forme de grille et filtrer par catégories.
- **Détailler :** Voir la description complète et le prix d'un produit spécifique.
- **Commander :** Ajouter/Retirer des articles du panier.
- **Visualiser :** Voir le récapitulatif du panier avec le montant total dynamique.

2.3. Règles de gestion

- **Unicité :** Un produit ajouté plusieurs fois incrémentera la quantité, il ne crée pas une nouvelle ligne dans le panier.
- **Stock :** (Simulé) Impossible d'ajouter un produit si la quantité demandée dépasse le stock fictif.
- **Total :** Le prix total se met à jour instantanément à chaque modification de quantité (Prix unitaire * Quantité).

3. CONCEPTION ET ARCHITECTURE

3.1. Architecture logicielle

Contrairement à une architecture MERN (Web), nous utilisons ici une architecture **MVVM (Model-View-ViewModel)** adaptée à Flutter :

- **Model** : Les structures de données (Product, CartItem).
- **View** : Les widgets Flutter (UI) qui affichent les données.
- **ViewModel (Provider/Bloc)** : La logique métier qui lie le modèle à la vue et notifie les changements.

3.2. Architecture technique

L'application est construite comme un monolithe modulaire côté client.

- **Langage** : Dart.
- **Moteur de rendu** : Skia (via Flutter).
- **Persistance** : Mémoire vive (RAM) pour la session courante (Mock).

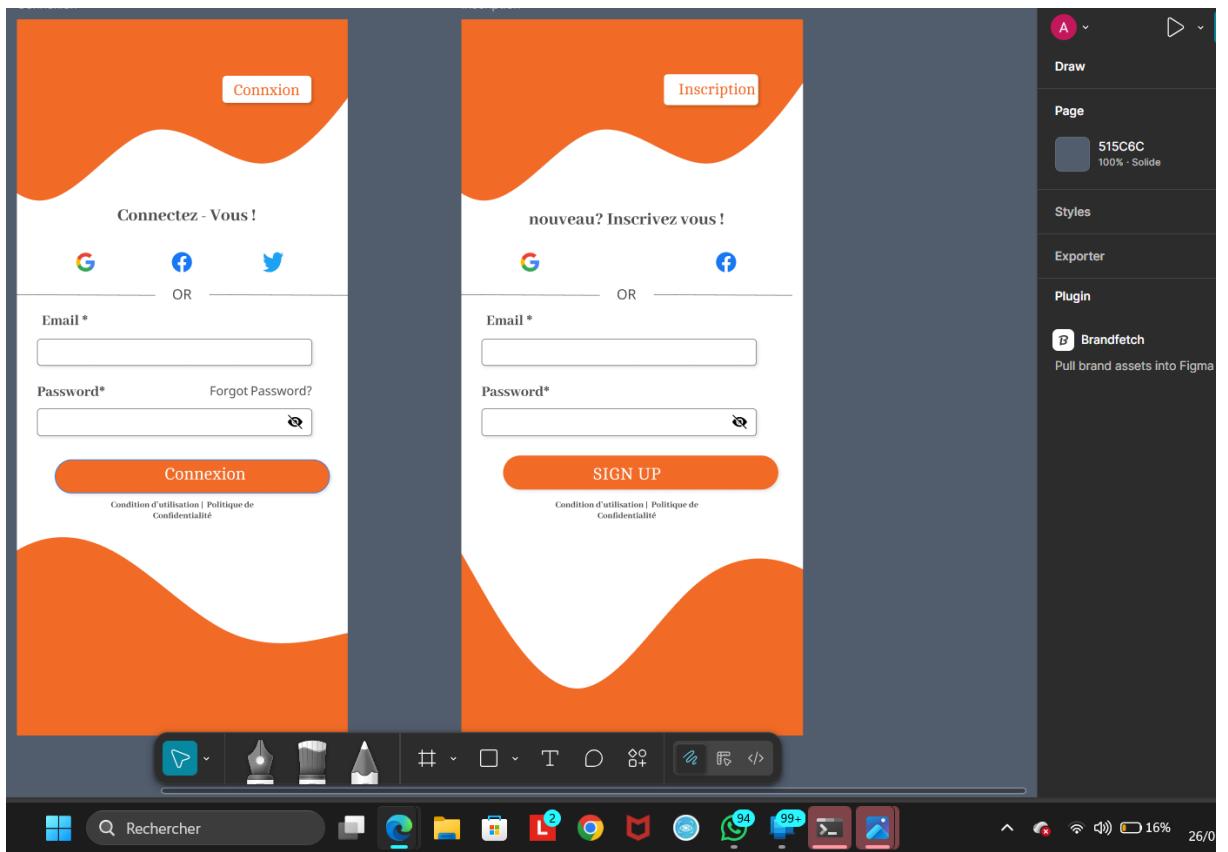
3.3. Conception UI/UX

-Principes UX appliqués

- **Écran d'accueil d'authentification**
- **Écran d'accueil**
- **Accès rapide au panier**
- **Utilisation de messages de type toast**
- **Feedback immédiat**

- Écrans principaux

Authentification



Accueil , Détails d'un plat, panier

Mon Panier

- Apple W-series 6 - Size: 36 ₣ 45,000
- Siberia 800 - Size: M ₣ 45,000
- Lycra Men's shirt - Size: S ₣ 45,000
- Nike/Lv Airforce 1 - Size: 42 ₣ 45,000

Total ₣ 180,000FCFA

Achetez

Salut Hamza

Prêt à faire du shopping ?

Profitez de -20 % ce week-end

-20 % ce week-end

Catégories populaires

- Electronics
- Clothing
- Shoes
- Accessories

Redmi Note 4 ₣ 18500

Apple Watch - series 6 ₣ 23750

Apple Watch Series 6

★★★★★

17500 ₣ 18,000 en stock

Description

Le S6 Sif amélioré fonctionne jusqu'à 20 % plus rapidement, ce qui permet aux applications de se lancer jusqu'à 20 % plus vite, tout en conservant la même autonomie de 18 heures sur toute la journée.

35 36 37 38 39 40

Ajouter au Panier

4. CHOIX TECHNOLOGIQUES

4.1.Outils de développement

- **Framework :** Flutter (Version Stable).
- **IDE :** VS Code avec extensions Dart/Flutter pour le linter et le déboggage.
- **Gestion d'état :** Provider (pour sa simplicité et son efficacité sur ce type de projet) ou Bloc (pour une séparation plus stricte des événements).
- **Assets :** Gestion locale des images via pubspec.yaml.

5.RÉALISATION ET IMPLÉMENTATION

5.1.Structure de la solution

5.1. Structure de la solution

L'arborescence du projet suit les bonnes pratiques Flutter :

```
lib/
  ├── models/    # Modèles de données (Product, Cart)
  ├── providers/ # Gestion d'état (Logique métier)
  ├── screens/   # Écrans (HomePage, ProductDetail, CartScreen)
  ├── widgets/   # Widgets réutilisables (ProductItem, Badge)
  └── main.dart   # Point d'entrée
```

5.2. Développement de la couche de Données (Mock Service)

Au lieu d'un backend Node.js, nous avons créé une classe MockDataService.

- Cette classe instancie des listes statiques d'objets Product.
- Elle simule les délais de chargement (futures) pour tester les indicateurs de chargement (spinners) dans l'UI.

5.3. Développement Frontend

- **GridView :** Implémentation de SliverGridDelegateWithFixedCrossAxisCount pour une grille responsive à 2 colonnes.
- **Animations :** Utilisation du widget Hero sur les images produits. Lors du clic, l'image "vole" de la grille vers la page détail.
- **Interactivité :** Utilisation de InkWell et GestureDetector pour gérer les clics.

6. TESTS ET RÉSULTATS

6.1. Stratégie de tests

- **Tests Unitaires :** Vérification de la logique du panier (ajout, suppression, calcul du total).
- **Tests Widgets :** Vérification que le widget `ProductItem` affiche bien le prix et le titre corrects.

7. DIFFICULTÉS ET SOLUTIONS

7.1. Problèmes de navigation (Flutter Navigator)

- *Problème :* Difficulté à passer des données (arguments) entre les routes nommées.
- *Solution :* Utilisation de `onGenerateRoute` pour passer l'objet Product entier en argument vers la page de détail.

7.2. Erreurs de cache et Assets

- *Problème :* Les images ne s'affichaient pas après ajout dans le dossier assets.
- *Solution :* Redémarrage complet de l'application (Cold restart) et vérification de l'indentation dans `pubspec.yaml`.

7.3. Gestion du panier et recalculation des totaux

- *Problème* : Le total ne se mettait pas à jour dans la barre de navigation quand on ajoutait un produit.
- *Solution* : Utilisation du Consumer<CartProvider> pour envelopper le widget texte du prix, forçant le redessin (rebuild) uniquement de cette zone lors d'un changement d'état.

7.4. Filtrage des produits

- *Problème* : Afficher uniquement les produits d'une catégorie sélectionnée.
- *Solution* : Création d'une méthode filterByCategory(String catId) dans le Provider qui retourne une sous-liste filtrée à la vue.

7.5. Upload et affichage des images

- *Contexte Mock* : Pas d'upload réel.
- *Défi* : Gérer des images de tailles différentes qui cassaient la grille.
- *Solution* : Utilisation du widget BoxFit.cover à l'intérieur d'un Container à taille fixe pour uniformiser les visuels.

7.6. Protection des routes

- *Problème* : Empêcher l'accès au panier s'il est vide.
- *Solution* : Ajout d'une vérification conditionnelle avant la navigation Navigator.push.

7.7. Optimisation de performance sur Android

- *Problème* : Ralentissement (jank) lors du scroll rapide de la GridView.
- *Solution* : Utilisation de const constructors pour les widgets statiques afin d'éviter des re-rendus inutiles.

8. CONCLUSION ET PERSPECTIVES

8.1. Bilan du projet

Ce projet a permis de livrer une maquette fonctionnelle haute fidélité. L'architecture mise en place via **Provider** permet une scalabilité aisée. L'utilisation de **GridView** et des animations **Hero** offre une expérience utilisateur proche des standards professionnels. **Notifications push avancées** pour certains événements (étendu dans les prochaines versions)

8.2. Améliorations futures

- Connecter l'application à un backend réel (Firebase ou API REST).
- Implémenter l'authentification utilisateur.
- Ajouter un mode sombre (Dark Theme).
- **Tests automatisés et CI/CD** : mise en place d'un pipeline pour automatiser les tests et déploiements.

8.3. Apports personnels

Nous avons acquis une compréhension solide du cycle de vie des widgets Flutter (Lifecycle) et de l'importance de séparer la logique métier de l'interface graphique pour maintenir un code propre.