

CAHIER DES CHARGES TECHNIQUE & FONCTIONNEL

Application Mobile – Marketplace Vêtements Seconde Main

1. Contexte & Vision du Projet

Ce projet consiste à concevoir et développer une application mobile de type marketplace dédiée à la vente et à l'achat de vêtements et accessoires de seconde main entre particuliers. L'application s'inspire des modèles Vinted (expérience utilisateur, marketplace internationale) et RetroChic (ancrage local, simplicité, confiance).

L'objectif principal est de fournir une plateforme :

- Simple à utiliser pour des utilisateurs non techniques
- Sécurisée pour les paiements et les échanges
- Performante et scalable pour supporter une montée en charge
- Facilement maintenable et évolutive pour des versions futures

2. Objectifs Fonctionnels & Techniques

2.1. Objectifs fonctionnels :

- Permettre à un utilisateur de créer un compte et gérer son profil :

L'application doit permettre à tout utilisateur de créer un compte personnel afin d'accéder aux fonctionnalités de la marketplace. Le développeur devra implémenter un système d'authentification sécurisé et fiable.

Fonctionnalités attendues:

- ✓ Implémenter un système d'authentification utilisateur :
 - Inscription via email + mot de passe
 - Connexion / déconnexion
 - Réinitialisation du mot de passe (email de récupération)
- ✓ Prévoir la validation du compte (email de confirmation ou équivalent)
- ✓ Créer un écran **Profil utilisateur** permettant :
 - Modification des informations personnelles (nom, photo, localisation)
 - Consultation de l'historique des ventes et achats
 - Accès aux paramètres du compte (sécurité, notifications, langue)

- ✓ Gérer les rôles utilisateurs (acheteur / vendeur, un même compte pouvant cumuler les deux rôles)

Contraintes techniques:

- Stockage sécurisé des mots de passe (hash)
 - Gestion des sessions / tokens (JWT ou équivalent)
 - Prévoir une structure extensible (connexion sociale possible en V2)
- Publier des annonces de vêtements (photos, description, prix) :

L'application doit permettre à un utilisateur (vendeur) de publier facilement une annonce de vêtement à vendre. Le processus doit être simple, guidé et optimisé pour mobile.

Fonctionnalités attendues :

- ✓ Accès à un bouton clair "Ajouter un article"
- ✓ Mettre en place un parcours de publication en plusieurs étapes :
 1. Ajout d'une ou plusieurs photos (depuis la galerie ou l'appareil photo)
 2. Saisie des informations produit :
 - Titre
 - Description
 - Catégorie (ex : homme, femme, enfant)
 - Taille
 - Marque
 - État du vêtement (neuf, très bon état, bon état, etc.)
 - Prix
 3. Validation des champs obligatoires et publication de l'annonce
- ✓ Implémenter la gestion des annonces côté backend (CRUD : créer, lire, modifier, supprimer)
- ✓ Prévoir des statuts pour chaque annonce :
 - Brouillon
 - En vente
 - Réservé
 - Vendu
- ✓ Permettre au vendeur de modifier ou supprimer une annonce tant qu'elle n'est pas vendue

Contraintes techniques:

- Optimisation et compression des images
 - Association claire annonce ↔ utilisateur
- Rechercher, filtrer et consulter des produits :

L'application doit offrir une expérience de navigation fluide permettant aux utilisateurs de trouver rapidement des produits correspondant à leurs besoins.

Fonctionnalités attendues :

- ✓ Implémenter un flux principal (feed) affichant les annonces disponibles (liste paginée ou infinie)
- ✓ Mettre en place une barre de recherche textuelle (titre, marque, mots-clés)
- ✓ Implémenter des filtres combinables :
 - Prix (min / max)
 - Catégorie
 - Taille
 - Marque
 - État
 - Localisation (optionnel)
- ✓ Tri des résultats (prix croissant/décroissant, nouveauté)
- ✓ Créer une page détail produit comprenant :
 - Galerie photos
 - Description complète
 - Prix
 - Informations vendeur
 - Commentaires et Avis
 - Boutons d'action (contacter le vendeur, acheter)

Contraintes techniques:

- Assurer des performances élevées même avec un volume important d'annonces
- Gérer les états vides (aucun résultat)
- Gérer les erreurs de recherche
- Requêtes optimisées pour éviter les lenteurs

➤ Mettre en relation acheteurs et vendeurs via une messagerie interne

L'application doit intégrer une messagerie interne permettant aux utilisateurs d'échanger avant et après une transaction, sans partager leurs coordonnées personnelles.

Fonctionnalités attendues :

- ✓ Développer un système de messagerie interne (chat) lié à une annonce.
- ✓ Associer automatiquement une conversation à un produit précis
- ✓ Permettre l'envoi et la réception de messages texte (images optionnelles si prévu)
- ✓ Conserver l'historique des conversations
- ✓ Indication des statuts :
 - message envoyé
 - message reçu
 - message lu (optionnel)
- ✓ Implémenter des notifications (push et in-app) pour :
 - Nouveau message
 - Réponse du vendeur / acheteur

Contraintes techniques:

- Messages stockés côté serveur
- Gère Sécurisation des échanges
- Prévoir une architecture compatible temps réel

➤ Gérer des transactions sécurisées avec paiement intégré

L'application doit permettre la réalisation de transactions financières sécurisées entre acheteurs et vendeurs via un système de paiement intégré.

Fonctionnalités attendues :

- ✓ Intégrer un prestataire de paiement sécurisé (à définir)
- ✓ Implémenter un flux de paiement clair :
 1. Validation de l'achat
 2. Paiement par l'acheteur
 3. Blocage des fonds jusqu'à confirmation de réception

- ✓ Mettre en place un **portefeuille interne** :
 - Crédit des ventes
 - Historique des transactions (achats/ventes)
- ✓ Permettre au vendeur de demander un retrait vers son compte bancaire
- ✓ Gérer les cas d'erreur :
 - Paiement échoué
 - Annulation
 - Litige

Contraintes techniques:

- Aucune donnée bancaire stockée côté application
 - Respect des standards de sécurité (PCI-DSS via prestataire)
 - Gestion des statuts de paiement (en attente, validé, échoué, remboursé)
- Assurer le suivi des commandes et livraisons

L'application doit permettre aux utilisateurs de suivre l'état de leurs commandes depuis l'achat jusqu'à la réception du produit.

Fonctionnalités attendues:

- ✓ Implémenter un système de gestion des commandes avec statuts :
 - Commande créée
 - Payée
 - Expédiée
 - Livrée
 - Confirmée / terminée
- ✓ Notifications à chaque changement de statut
- ✓ Permettre au vendeur d'indiquer l'expédition de l'article
- ✓ Permettre au vendeur d'indiquer l'expédition de l'article
- ✓ Afficher l'historique des commandes dans le profil utilisateur

Contraintes techniques:

- Gestion claire des statuts côté backend
- Synchronisation commande ↔ paiement ↔ livraison
- Prévoir l'extension future avec des services de livraison externes

2.2 Objectifs techniques :

- Développer une application mobile robuste (Android & iOS)

Objectif : Fournir une application mobile stable, fiable et maintenable, fonctionnant de manière homogène sur Android et iOS.

Attentes côté développement :

- Développer l'application pour :
 - Android (versions supportées à définir)
 - iOS (versions supportées à définir)
- Garantir une base de code :
 - Lisible
 - Documentée
 - Structurée selon les bonnes pratiques (clean architecture, séparation des couches)
- Gérer correctement les états applicatifs :
 - Chargement
 - Erreur réseau
 - Absence de données
- Prévoir une gestion centralisée des erreurs (logs, crash reports)
- Assurer la stabilité de l'application en conditions réelles (réseaux faibles, interruptions, multitâche)

- Garantir des performances optimales

Objectif : Assurer une expérience utilisateur fluide, rapide et sans latence perceptible, même avec un grand volume de données.

Attentes côté développement :

- Optimiser les temps de réponse côté backend :
 - API répondant en moins de 2 secondes dans 95 % des cas
 - Pagination des listes (produits, messages, conversations)
- Optimiser les performances côté mobile :
 - Chargement progressif des contenus
 - Mise en cache des données fréquemment consultées
 - Lazy loading des images
- Mettre en place une stratégie d'optimisation des images :
 - Compression automatique
 - Redimensionnement selon l'écran

- Limiter les appels réseau inutiles :
 - Mutualisation des requêtes
 - Mise en cache locale (ex : produits récemment consultés)
 - Tester les performances sur :
 - Appareils bas/moyen/haut de gamme
 - Réseaux 3G / 4G / Wi-Fi
 - Surveiller les performances via des outils de monitoring
- Sécuriser les données utilisateurs et les transactions
- Objectif :** Garantir la confidentialité, l'intégrité et la sécurité des données personnelles et financières des utilisateurs.
- Attentes côté développement :**
- Mettre en place une authentification sécurisée :
 - Gestion sécurisée des tokens
 - Expiration et renouvellement des sessions
 - Chiffrer les données sensibles :
 - En transit (HTTPS / TLS)
 - Au repos (base de données, stockage)
 - Ne jamais stocker :
 - Données bancaires sensibles côté client
 - Informations critiques en clair
 - Mettre en place des mécanismes de protection contre :
 - Accès non autorisé
 - Attaques basiques (brute force, injection, etc.)
 - Gérer les droits et permissions :
 - Accès aux données limité à l'utilisateur concerné
 - Prévoir des mécanismes de journalisation (logs) pour audit et détection d'anomalies
 - Assurer la conformité RGPD/ CNDP :
 - Consentement utilisateur
 - Suppression de compte
 - Export des données personnelles
- Prévoir une architecture modulaire et évolutive

Objectif : Concevoir une architecture technique capable d'évoluer sans remise en cause globale du système.

Attentes côté développement :

- Concevoir une architecture basée sur des modules clairement séparés :
 - Authentification
 - Utilisateurs
 - Produits
 - Messagerie
 - Paiements
 - Commandes
 - Séparer clairement :
 - Frontend mobile
 - Backend API
 - Services tiers
 - Prévoir des interfaces claires entre modules (API bien définies)
 - Faciliter l'ajout futur de fonctionnalités :
 - Nouvelles catégories de produits
 - Abonnements premium
 - Systèmes de recommandation
 - Prévoir une architecture scalable :
 - Support de la montée en charge
 - Possibilité de répartir les services (microservices ou modular monolith)
 - Prévoir des environnements distincts :
 - Développement
 - Recette / test
 - Production
 - Documenter l'architecture (schémas, descriptions)
- Assurer la maintenabilité et la qualité du code

Objectif : Garantir que l'application puisse être maintenue, corrigée et améliorée dans le temps.

Attentes côté développement :

- Respecter les standards de codage
- Documenter le code et les APIs
- Mettre en place :
 - Tests unitaires
 - Tests d'intégration
- Faciliter l'onboarding d'un nouveau développeur
- Prévoir une gestion de versions claire (git)
- Automatiser les tâches répétitives si possible (build, tests)

- Assurer la fiabilité et la disponibilité du service

Objectif : Garantir un service accessible et fonctionnel en continu.

Attentes côté développement :

- Mettre en place une infrastructure stable
- Prévoir des mécanismes de sauvegarde des données
- Gérer les pannes et redémarrages
- Mettre en place des alertes en cas d'incident
- Limiter les interruptions de service lors des mises à jour

3. Profils Utilisateurs & Rôles

3.1 Acheteur

Description du rôle : L'acheteur est un utilisateur dont l'objectif principal est de consulter les annonces et d'acheter des produits via la plateforme.

Droits & autorisations :

- Accès en lecture aux annonces publiques
- Accès aux fiches produits détaillées
- Accès à la messagerie uniquement pour les annonces consultées
- Accès à son espace personnel (profil, commandes, paiements)

Contraintes & règles métier :

- Un acheteur ne peut effectuer un paiement que s'il est authentifié
- Un acheteur ne peut accéder qu'à ses propres commandes et conversations
- Un acheteur ne peut pas modifier une annonce qui ne lui appartient pas
- Les actions critiques (paiement, confirmation de réception) doivent être confirmées explicitement

Implications techniques

- Vérification systématique de l'identité de l'utilisateur avant toute action sensible
- Gestion fine des autorisations côté API (ownership des ressources)
- Isolation stricte des données entre utilisateurs

3.2 Vendeur

Description du rôle : Le vendeur est un utilisateur autorisé à publier et gérer des annonces de vente sur la plateforme.

Un même utilisateur peut être **à la fois acheteur et vendeur**.

Droits & autorisations :

- Accès à un espace de gestion de ses propres annonces
- Accès aux conversations liées à ses annonces
- Accès à son portefeuille et à l'historique de ses transactions
- Accès aux outils de suivi des commandes liées à ses ventes

Contraintes & règles métier :

- Un vendeur ne peut modifier ou supprimer une annonce que tant qu'elle n'est pas vendue
- Un vendeur ne peut accéder qu'aux données financières qui lui sont propres
- Les fonds issus d'une vente sont soumis à validation (réception confirmée par l'acheteur)
- Toute action liée à une livraison doit être tracée (date, statut)

Implications techniques :

- Gestion des statuts d'annonce et de commande
- Liaison stricte entre annonce ↔ vendeur ↔ transaction
- Sécurisation des flux financiers et historiques

3.3 Administrateur

Description du rôle : L'administrateur est un utilisateur interne à la plateforme chargé du bon fonctionnement, de la sécurité et de la conformité du service.

Droits & autorisations :

- Accès à un back-office sécurisé
- Accès à la liste des utilisateurs et annonces
- Accès aux données de supervision (sans modification directe des paiements)

- Accès aux statistiques globales

Responsabilités :

- Modération des contenus (annonces, profils, messages signalés)
- Gestion des comptes utilisateurs (suspension, suppression)
- Supervision des transactions et résolution des litiges
- Suivi de la performance de la plateforme

Contraintes & règles métier :

- Les actions administratives doivent être tracées (logs)
- Accès limité aux données sensibles selon le principe du moindre privilège
- Aucun administrateur ne doit pouvoir modifier directement une transaction financière validée

Implications techniques :

- Séparation stricte front utilisateur / back-office admin
- Gestion des rôles et permissions avancées
- Journalisation des actions critiques

3.4 Gestion des rôles & permissions (Transversal)

Règles générales :

- Un compte utilisateur peut cumuler plusieurs rôles (acheteur + vendeur)
- Les permissions doivent être contrôlées :
 - côté frontend (UX)
 - côté backend (API – obligatoire)
- Toute tentative d'accès non autorisé doit être bloquée et journalisée

Attentes techniques :

- Système de gestion des rôles centralisé
- Vérification des droits à chaque appel API sensible
- Structure évolutive permettant l'ajout de nouveaux rôles (ex : modérateur)

4. Parcours Utilisateurs Détaillés (User Flows)

4.1 Parcours d'inscription et d'authentification

4.1.1 Inscription (nouvel utilisateur)

1. Accès écran “Créer un compte”
2. Saisie des informations (email, mot de passe)
3. Acceptation des conditions et politique de confidentialité
4. Validation du formulaire
5. Envoi d'un email de confirmation
6. Activation du compte
7. Redirection vers l'écran d'accueil

Cas alternatifs :

- Email déjà utilisé
- Mot de passe non conforme
- Erreur réseau

4.1.2 Connexion

1. Accès écran “Connexion”
2. Saisie des identifiants
3. Authentification réussie
4. Redirection vers l'accueil

Cas alternatifs :

- Identifiants incorrects
- Compte non activé
- Trop de tentatives → blocage temporaire

4.1.3 Réinitialisation du mot de passe

1. Accès “Mot de passe oublié”
2. Saisie de l'email

3. Réception lien de réinitialisation
4. Définition nouveau mot de passe
5. Confirmation

4.2 Parcours de consultation et découverte des produits

4.2.1 Navigation sur le feed

1. Accès à l'accueil
2. Chargement du feed produits
3. Scroll infini / pagination
4. Sélection d'un produit

États à gérer :

- Chargement
- Aucun résultat
- Erreur serveur

4.2.2 Recherche et filtrage

1. Accès à la recherche
2. Saisie d'un mot-clé
3. Application de filtres
4. Mise à jour dynamique des résultats
5. Consultation fiche produit

4.2.3 Ajout aux favoris (si prévu)

1. Clic icône "favori"
2. Enregistrement côté utilisateur
3. Consultation liste favoris

4.3 Parcours de publication et gestion d'annonce (Vendeur)

4.3.1 Crédit d'une annonce

1. Clic sur bouton "Ajouter un article"

2. Upload des photos
3. Saisie des informations produit
4. Validation
5. Création annonce avec statut "En vente"

Cas alternatifs :

- Champs obligatoires manquants
- Upload image échoué

4.3.2 Modification d'une annonce

1. Accès "Mes annonces"
2. Sélection annonce
3. Modification des informations
4. Sauvegarde

Règle :

- Modification interdite si statut "Vendu"

4.3.3 Suppression d'une annonce

1. Sélection annonce
2. Confirmation suppression
3. Suppression ou archivage

4.4 Parcours de messagerie (Acheteur ↔ Vendeur)

4.4.1 Démarrer une conversation

1. Accès fiche produit
2. Clic "Contacter le vendeur"
3. Création conversation liée à l'annonce
4. Envoi message

4.4.2 Échange de messages

1. Réception message

2. Notification
3. Lecture et réponse
4. Historique mis à jour

4.4.3 Signalement / blocage

1. Signalement d'un message ou utilisateur
2. Transmission à l'administration
3. Action de modération

4.5 Parcours d'achat et de paiement

4.5.1 Achat direct

1. Accès fiche produit
2. Clic "Acheter"
3. Choix livraison
4. Confirmation commande
5. Paiement sécurisé
6. Création commande (statut "Payée")
7. Notification vendeur

4.5.2 Paiement échoué

1. Tentative paiement
2. Échec
3. Message d'erreur
4. Nouvelle tentative ou abandon

4.6 Parcours de commande et livraison

4.6.1 Expédition (Vendeur)

1. Notification de vente
2. Préparation colis
3. Marquage "Expédié"

4. Notification acheteur

4.6.2 Réception (Acheteur)

1. Réception colis
2. Confirmation réception
3. Validation transaction
4. Crédit du portefeuille vendeur

4.6.3 Litige

1. Déclaration litige
2. Blocage temporaire des fonds
3. Intervention administrateur
4. Résolution

4.7 Parcours portefeuille & paiements (Vendeur)

4.7.1 Consultation du portefeuille

1. Accès espace portefeuille
2. Consultation solde et historique

4.7.2 Demande de retrait

1. Saisie montant
2. Validation
3. Traitement retrait
4. Confirmation

4.8 Parcours profil et paramètres

4.8.1 Modification profil

1. Accès profil
2. Modification informations
3. Sauvegarde

4.8.2 Suppression du compte

1. Demande suppression
2. Confirmation
3. Anonymisation / suppression données
4. Déconnexion

4.9 Parcours administrateur (Back-office)

4.9.1 Modération annonce

1. Consultation signalements
2. Analyse contenu
3. Action (suppression / suspension)

4.9.2 Gestion utilisateur

1. Recherche utilisateur
2. Consultation historique
3. Action administrative

4.9.3 Supervision plateforme

1. Accès dashboard
2. Consultation statistiques
3. Suivi incidents

5. Règles Métier & Gestion des États

5.1 Règles métier - Annonces

5.1.1 États possibles d'une annonce

État	Description
Brouillon	Annonce créée mais non visible publiquement
En attente de validation	En attente de modération (si activée)
En vente	Annonce visible et achetable

État	Description
Réservée	Annonce associée à une commande en cours
Vendue	Annonce définitivement vendue
Suspendue	Annonce désactivée par l'administration
Archivée	Annonce retirée volontairement

5.1.2 Transitions autorisées

- **Brouillon → En vente**
- **En vente → Réservée**
- **Réservée → Vendue**
- **Réservée → En vente** (annulation achat)
- **En vente → Archivée**
- **Tout état → Suspendue** (admin)

5.1.3 Règles métier associées

- Une annonce **ne peut avoir qu'un seul acheteur actif à la fois**
- Une annonce **réservée ne peut plus être achetée**
- Une annonce **vendue n'est plus modifiable**
- Une annonce suspendue n'est visible que par l'administrateur
- Toute modification d'annonce est interdite après le statut "Vendue"

6.1.4 Implications techniques

- Verrouillage logique lors de la réservation
- Vérification systématique de l'état avant action
- Historique des changements d'état

5.2 Règles métier – Annonces

5.2.1 États possibles d'une commande

État	Description
Créée	Commande initiée, non payée
Payée	Paiement validé
En préparation	En cours de préparation par le vendeur
Expédiée	Colis envoyé
Livrée	Réception confirmée
Annulée	Commande annulée
En litige	Litige ouvert

6.2.2 Transitions autorisées

- Créée → Payée
- Payée → En préparation
- En préparation → Expédiée
- Expédiée → Livrée
- Payée → Annulée
- Livrée → En litige

6.2.3 Règles métier associées

- Une commande non payée est automatiquement expirée après un délai défini
- Une commande payée ne peut être annulée que sous conditions
- Une commande livrée déclenche le déblocage des fonds
- Une commande en litige bloque les paiements associés

6.2.4 Implications techniques

- Tâches automatiques de vérification d'expiration
- Synchronisation commande ↔ annonce
- Traçabilité complète des événements

5.3 Règles métier – Annonces

5.3.1 États possibles d'un paiement

État	Description
Initié	Paiement en cours
Réussi	Paiement validé
Échoué	Paiement refusé
En attente	En attente de confirmation externe
Remboursé	Paiement annulé

5.3.2 Règles métier associées

- Le paiement est obligatoire avant réservation
- Les fonds sont conservés en séquestre
- Le vendeur n'est crédité qu'après confirmation de livraison
- En cas de litige, les fonds restent bloqués
- Les remboursements suivent un processus contrôlé

5.3.3 Implications techniques

- Intégration d'un PSP sécurisé
- Gestion des webhooks
- Idempotence des paiements
- Journalisation complète

5.4 Règles métier – Portefeuille (Vendeur)

5.4.1 États du solde

- Solde disponible
- Solde bloqué

- Solde en cours de retrait

5.4.2 Règles de gestion

- Crédit uniquement après livraison validée
- Débit uniquement après validation retrait
- Historique non modifiable
- Montant minimum de retrait

5.5 Règles métier – Litiges & Exceptions

5.5.1 Ouverture d'un litige

- Possible uniquement après livraison
- Délai maximum après réception
- Motif obligatoire

5.5.2 Traitement du litige

- Blocage des fonds
- Intervention administrateur
- Décision finale irréversible

5.6 Règles transversales & sécurité

- Toute action critique doit être authentifiée
- Toute transition d'état doit être validée côté backend
- Les actions doivent être traçables
- Aucun état ne peut être modifié directement côté frontend

5.7 Diagramme logique simplifié (description)

- Annonce ↔ Commande ↔ Paiement
- Une annonce génère une seule commande active
- Une commande est liée à un seul paiement
- Un paiement alimente un portefeuille vendeur

6. Structure Générale des Écrans & Pages de l'Application

6.1 Pages publiques (non authentifiées)

- **Page d'accueil (Feed produits)** : Cette page permet d'afficher la liste des produits disponibles à la vente. Elle contient principalement un flux d'annonces, avec les informations essentielles permettant à l'utilisateur de découvrir les articles.
- **Page de recherche** : Cette page permet à l'utilisateur de rechercher des produits via des mots-clés et d'affiner les résultats à l'aide de filtres. Elle affiche les résultats correspondants à la recherche effectuée.
- **Page fiche produit** : Cette page est dédiée à l'affichage détaillé d'un produit. Elle présente les informations complètes de l'article (description, images, prix, état, vendeur) et permet d'accéder aux actions liées au produit (contact, achat).

6.2 Pages d'authentification

- **Page d'inscription** : Cette page permet à un nouvel utilisateur de créer un compte sur la plateforme. Elle contient les champs nécessaires à l'inscription et les liens vers les conditions d'utilisation.
- **Page de connexion** : Cette page permet à un utilisateur existant de s'authentifier afin d'accéder à son espace personnel.
- **Page de récupération du mot de passe** : Cette page permet à l'utilisateur de lancer la procédure de réinitialisation de son mot de passe.

6.3 Pages utilisateur (authentifié)

- **Page profil utilisateur** : Cette page permet à l'utilisateur de consulter et modifier ses informations personnelles ainsi que d'accéder à ses paramètres de compte.
- **Page messagerie** : Cette page permet la communication entre acheteurs et vendeurs. Elle affiche la liste des conversations et le contenu des échanges liés aux annonces.
- **Page commandes** : Cette page permet à l'utilisateur de consulter l'historique et le statut de ses commandes, ainsi que le suivi des livraisons.
- **Page favoris (si activée)** : Cette page permet à l'utilisateur de retrouver les produits qu'il a enregistrés pour consultation ultérieure.

6.4 Pages vendeur

- **Page création d'annonce** : Cette page permet au vendeur de publier un nouvel article à la vente. Elle contient les champs nécessaires à la saisie des informations produit et à l'ajout de photos.

- **Page gestion des annonces :** Cette page permet au vendeur de consulter, modifier ou archiver ses annonces existantes selon leur statut.
- **Page portefeuille :** Cette page permet au vendeur de consulter son solde, l'historique des transactions et d'initier des demandes de retrait.

6.5 Pages liées au paiement et à la livraison

- **Page paiement :** Cette page permet à l'acheteur de finaliser un achat en affichant le récapitulatif de la commande et les moyens de paiement disponibles.
- **Page suivi de livraison :** Cette page permet à l'acheteur et au vendeur de suivre l'état de la livraison associée à une commande.

6.6 Pages administrateur (Back-office)

- **Page tableau de bord :** Cette page permet à l'administrateur d'avoir une vue globale sur l'activité de la plateforme (utilisateurs, annonces, transactions).
- **Page gestion des utilisateurs :** Cette page permet à l'administrateur de consulter et gérer les comptes utilisateurs.
- **Page modération des annonces :** Cette page permet de modérer les contenus publiés sur la plateforme.
- **Page gestion des litiges :** Cette page permet à l'administrateur de traiter les litiges entre utilisateurs.

6.7 UI / UX & direction artistique (à définir ultérieurement)

L'interface utilisateur (UI) et l'expérience utilisateur (UX) ne sont pas figées dans ce cahier des charges.

Le style graphique, la charte visuelle, la navigation détaillée et l'organisation précise du contenu de chaque page devront être définis :

- en collaboration avec le développeur
- sous validation du superviseur et du porteur de projet

L'objectif est d'aboutir à un thème cohérent, moderne et adapté au public cible, tout en respectant les contraintes fonctionnelles et techniques décrites dans le présent document.

7. Back-Office & Administration

7.1 Accès & Sécurité du Back-Office

Accès administrateur :

- Accès via authentification sécurisée
- Comptes administrateurs distincts des comptes utilisateurs
- Possibilité de définir plusieurs niveaux d'accès (admin, modérateur, support)

Sécurité :

- Journalisation de toutes les actions administratives
- Gestion des sessions et déconnexion automatique
- Restrictions d'accès aux données sensibles selon le rôle

7.2 Tableau de Bord Administrateur

Description :

Le tableau de bord constitue la page d'accueil du back-office et fournit une vue synthétique de l'activité de la plateforme.

Contenu attendu :

- Nombre total d'utilisateurs
- Nombre d'annonces actives
- Volume de transactions
- Commandes en cours
- Litiges ouverts
- Alertes (contenus signalés, paiements bloqués)

Objectifs :

- Permettre un suivi rapide de la santé de la plateforme
- Identifier immédiatement les anomalies ou incidents

7.3 Gestion des Utilisateurs

Fonctionnalités :

- Liste des utilisateurs avec filtres (statut, date d'inscription)
- Accès au profil utilisateur détaillé
- Consultation de l'historique d'activité
- Actions administratives :
 - suspension de compte
 - réactivation
 - suppression ou anonymisation

Règles :

- Aucune modification directe des données sensibles sans traçabilité
- Les actions critiques doivent être confirmées

7.4 Gestion des Annonces

Fonctionnalités :

- Liste des annonces avec filtres par statut
- Consultation du contenu complet de l'annonce
- Accès à l'historique des modifications
- Actions possibles :
 - suspension
 - suppression
 - archivage

Modération :

- Traitement des annonces signalées
- Justification obligatoire pour toute action de modération
- Notification automatique de l'utilisateur concerné

7.5 Modération des Contenus

Contenus concernés :

- Annonces

- Profils utilisateurs
- Messages (suite à signalement)

Fonctionnement :

- Réception des signalements
- Analyse du contenu
- Décision de modération
- Archivage des décisions

Objectifs :

- Garantir un environnement sûr
- Assurer la conformité aux règles de la plateforme

7.6 Gestion des Litiges

Description :

Le back-office doit permettre la **gestion complète des litiges** entre acheteurs et vendeurs.

Fonctionnalités

- Liste des litiges ouverts
- Accès au détail du litige (commande, messages, preuves)
- Blocage des fonds associés
- Décision finale :
 - validation vendeur
 - remboursement acheteur

Règles :

- Toute décision est définitive et tracée
- Les actions doivent déclencher les mises à jour financières correspondantes

7.7 Supervision des Transactions

Fonctionnalités :

- Consultation des transactions
- Visualisation des statuts de paiement
- Détection des anomalies
- Accès aux journaux financiers

Contraintes :

- Aucune modification manuelle des paiements validés
- Accès restreint aux données financières

7.8 Statistiques & Reporting

Indicateurs clés :

- Croissance utilisateurs
- Activité vendeurs
- Volume des ventes
- Taux de litiges
- Revenus plateforme

Exports :

- Génération de rapports
- Export CSV / Excel
- Périodes personnalisées

Objectifs :

- Aider à la prise de décision
- Suivre les performances de la plateforme

7.9 Exigences Techniques Back-Office

- Interface web responsive
- Séparation stricte des rôles
- Architecture modulaire
- Performances optimisées

- Compatibilité multi-navigateurs

8. Architecture Technique & Choix Technologiques

**Stack recommandée : Flask first

8.1 Architecture globale

Principe général : L'application repose sur une architecture **client-serveur** avec séparation claire des responsabilités :

- Application mobile (Flutter)
- Backend API (services métiers)
- Base de données
- Services tiers (paiement, notifications, stockage)

Cette séparation permet :

- une évolution indépendante des composants
- une meilleure sécurité
- une maintenance facilitée

8.2 Application mobile (Frontend)

Technologie recommandée : Flutter (Dart)

Architecture Flutter recommandée :

- Clean Architecture
- Séparation :
 - Presentation (UI)
 - Domain (use cases)
 - Data (repositories, services)

8.3 Backend (API & logique métier)

Technologie recommandée : Node.js + NestJS (TypeScript)

Architecture Backend : Controllers, Services, Modules, Guards & Interceptors

Authentification & Autorisation : JWT (Access + Refresh tokens), OAuth 2.0 (optionnel), RBAC (Role-Based Access Control)

8.4 Base de données

Base principale: PostgreSQL

ORM : Prisma

- typage fort
- migrations
- compatibilité NestJS

Cache : Redis

- sessions
- tokens
- performances

8.5 Stockage fichiers & images

Solution recommandée : AWS S3 ou équivalent (GCP / Azure), onedrive ,

Accès :

- URLs signées
- Upload direct depuis Flutter

8.6 Paiements

PSP recommandé : Stripe

- Compatible Flutter
- Webhooks backend

Gestion paiements :

- Séquestre
- Webhooks sécurisés
- Idempotence

8.7 Notifications

Push notifications : Firebase Cloud Messaging (FCM)

Emails : SendGrid ou Amazon SES

8.8 Tests & Qualité

Flutter : Unit tests, Widget tests, Integration tests

Backend : Jest, Tests unitaires & e2e

CI/CD : GitHub Actions, Tests automatiques, Build Flutter

8.9 Sécurité

- HTTPS obligatoire
- Hash mots de passe (bcrypt)
- Protection API (rate limiting)
- Validation des entrées
- Logs & monitoring

8.10 Déploiement

Backend : Docker, AWS ECS / EC2 / GCP Cloud Run

Base de données : AWS RDS

Application mobile

- Google Play Store
- Apple App Store

8.11 Monitoring & Logs

- Sentry
- Prometheus + Grafana
- Winston (logs backend)

8.12 Documentation

API : Swagger / OpenAPI

Projet : README technique, Diagrammes d'architecture, Conventions de code

8.13 Scalabilité & Évolutivité

- Architecture modulaire
- Services découplés
- Prêt pour microservices
- Support multi-pays

9. Tests & Qualité

9.1 Tests Unitaires

Objectif : Vérifier le fonctionnement de chaque composant individuel (fonction, service, module) de manière isolée.

Cibles :

- Fonctions et classes du backend (NestJS)
- Logique métier (ex: calcul du solde, validation des commandes)
- Services Flutter (ex: form validation, calculs, conversion de données)

Outils recommandés :

- **Flutter** : flutter_test
- **Backend NestJS** : Jest

Livrables :

- Couverture minimale recommandée : **>70 %** des fonctions critiques
- Rapports automatisés intégrés dans le CI/CD

9.2 Tests Fonctionnels

Objectif : Vérifier que l'ensemble de l'application fonctionne correctement, du point de vue de l'utilisateur.

Cibles :

- Parcours utilisateur principaux :
 - Création compte
 - Publication d'annonce
 - Achat et paiement

- Messagerie
- Gestion des commandes
- Back-office :
 - Gestion utilisateurs
 - Modération annonces
 - Supervision transactions

Outils recommandés :

- **Flutter** : integration_test ou Flutter Driver
- Backend : SuperTest ou Postman/Newman pour API

Méthodologie :

- Scénarios end-to-end (E2E) pour tous les user flows
- Validation des états (ex: annonce “vendue” ne peut plus être modifiée)
- Tests automatisés intégrés au pipeline CI/CD

9.3 Tests de Performance

Objectif : S’assurer que l’application reste rapide et stable même sous forte charge.

Cibles :

- Temps de réponse API : < 2s
- Chargement feed produits
- Publication annonce (upload images)
- Paiements et transactions
- Notifications

Outils recommandés :

- Backend : Artillery, JMeter, ou k6
- Flutter : flutter_driver pour mesurer temps d’exécution de screens

Scénarios :

- 1000 utilisateurs simultanés (simulé)

- Upload d'images multiples
- Transactions simultanées

9.4 Phase de Beta Test

Objectif : Valider l'application en conditions réelles, avec de vrais utilisateurs.

Méthodologie :

- Sélection d'un groupe d'utilisateurs (beta testers)
- Collecte des retours UX/UI et bugs
- Priorisation et correction avant release officielle
- Suivi des métriques clés :
 - Crash rate
 - Feedback utilisateur
 - Taux d'usage des fonctionnalités

Outils :

- **Firebase Crashlytics** (Flutter)
- **Feedback form / surveys**
- **Analytics** pour suivi des comportements

9.5 Processus qualité & intégration continue

- Mise en place d'un pipeline CI/CD :
 - Exécution tests unitaires et fonctionnels
 - Analyse de couverture
 - Déploiement automatisé en staging
- Validation obligatoire avant merge en production
- Documentation des tests et procédures

10. Planning, Livrables & Estimation

Phase 1 – Analyse & Conception (2 semaines)

Objectifs :

- Analyse des besoins fonctionnels et techniques
- Établir les user flows, schéma BDD, endpoints principaux
- Architecture technique globale

Livrables :

- architecture schématique
- Wireframes et parcours utilisateurs
- modèles BDD

Phase 2 – Setup technique & Backend initial (4 semaines)

Objectifs :

- Initialisation du projet Flutter et backend
- Mise en place BDD PostgreSQL + Prisma
- Authentification / gestion utilisateurs (JWT + rôles)
- Endpoints API principaux : annonces, produits, commandes, paiement

Livrables :

- Backend fonctionnel avec endpoints de base
- Base de données opérationnelle
- Authentification et sécurité initiale

Phase 3 – Développement mobile MVP (Flutter) (6 semaines)

Objectifs :

- Intégration des écrans essentiels
- Intégration avec le backend existant (API REST)

Livrables :

- Application mobile fonctionnelle (MVP)
- Navigation & état fonctionnel

Phase 4 – Backend avancé & Admin (4 semaines)

Objectifs :

- Back-office / administration :
 - Tableau de bord
 - Gestion utilisateurs & annonces
 - Modération contenu
 - Gestion litiges & statistiques
- Finalisation endpoints avancés pour commandes, paiements et messagerie

Livrables :

- Back-office fonctionnel
- API complète pour toutes les fonctionnalités principales

Phase 5 – Mobile complet & fonctionnalités avancées (4 semaines)

Objectifs :

- Publication annonce / upload images
- Paiement intégré Stripe
- Messagerie interne entre acheteurs & vendeurs
- Notifications push via FCM
- Portefeuille vendeur

Livrables :

- Application mobile complète
- Toutes les fonctionnalités MVP opérationnelles

Phase 6 – Tests & QA (3 semaines)

Objectifs :

- Tests unitaires (Flutter + backend)
- Tests fonctionnels / E2E (user flows critiques)
- Tests performance et temps de réponse (<2s)
- Beta test interne / externe

Livrables :

- Rapports de tests
- Liste de bugs corrigés
- Version stabilisée du MVP

Phase 7 – Déploiement & release (3 semaines)**Objectifs :**

- Déploiement backend (AWS / Docker / RDS)
- Publication mobile sur App Store & Play Store
- Monitoring, logs et alertes
- Documentation technique finale (API + dev + release)

Livrables :

- Version production accessible
- Documentation complète prête pour maintenance

10.2 Estimation globale

Phase	Durée estimée
Analyse & Conception	2 semaines
Setup technique & Backend initial	4 semaines
Développement mobile MVP (Phase 1)	6 semaines
Backend avancé & Admin	4 semaines
Mobile complet & fonctionnalités avancées	4 semaines
Tests & QA	3 semaines
Déploiement & release	3 semaines

10.3 Livrables clés

- Cahier des charges et wireframes
- Backend fonctionnel + API documentée
- Application mobile Flutter complète
- Back-office administrateur
- Tests unitaires, fonctionnels et performance
- Documentation technique + manuel utilisateur
- Version production sur stores et backend live