# **📌 Cahier des Charges Fonctionnel – Application Mobile de Marché Agricole**

### **Projet : Conception et Développement d’une Application Mobile de Marché Agricole dans le Département du Plateau (Bénin)**

📅 **Durée de Livraisons :** **3 semaines**🎯 **Objectif :** Fournir un cadre précis pour la conception des interfaces UI/UX et le développement technique.

## **1️⃣ Présentation du Projet**

L’application a pour but de **connecter directement les agriculteurs et les acheteurs** à travers une **marketplace mobile simple et efficace**. Elle permettra :  
✅ Aux **agriculteurs** de publier et vendre leurs produits.  
✅ Aux **acheteurs** de parcourir, commander et suivre leurs achats.  
✅ Une gestion fluide des transactions avec **paiement à la livraison**.

🔹 **Technologies utilisées** :

* **Frontend (UI/UX)** : **Flutter (Dart)**
* **Backend** : **Python (FastAPI/Django REST)**
* **Base de données** : **PostgreSQL/MySQL**

🔹 **Priorité UX/UI** :  
✅ Design **simple et intuitif** 🎨.  
✅ Expérience **fluide et rapide** 🚀.  
✅ Adapté aux **zones rurales et semi-urbaines** 📶.

## **2️⃣ Liste des Interfaces et Fonctionnalités à Implémenter**

Le **UI/UX designer** doit concevoir **9 interfaces clés**, en respectant les **besoins des utilisateurs et les contraintes techniques**.

### **1️⃣ Page de Connexion (Login Page)**

🎯 **Objectif** : Permet aux **acheteurs et agriculteurs** de se connecter à leur compte.  
📌 **Éléments UI** :

* Champ **Téléphone ou Email** 📩.
* Champ **Mot de passe** 🔒 avec option d’affichage.
* Bouton **"Se connecter"** 🎯 (redirige vers la **Page d’Accueil** après validation).
* **Lien "Mot de passe oublié ?"** ➡️ Redirection vers **Page de Réinitialisation**.
* **Lien "Créer un compte"** ➡️ Redirection vers **Page d’Inscription**.

📌 **Règles UX** :  
✅ Design épuré et minimaliste.  
✅ Validation des entrées avant envoi.

### **2️⃣ Page d’Inscription (Signup Page)**

🎯 **Objectif** : Permet aux **agriculteurs et acheteurs** de s’inscrire.  
📌 **Éléments UI** :

* **Nom complet** 🆔.
* **Téléphone** 📞 (**doit être unique**).
* **Rôle (Agriculteur / Acheteur)** 📌 (sélection via bouton radio).
* **Adresse (pré-remplie via GPS si activé, modifiable manuellement)** 📍.
* **Mot de passe + Confirmation** 🔐.
* **Bouton "S’inscrire"** ✅ (redirige vers la **Page d’Accueil** après validation).

📌 **Règles UX** :  
✅ Interface **simple et fluide**, avec **icônes et placeholders clairs**.  
✅ **Vérification en temps réel** des champs avant validation.

### **3️⃣ Page d’Accueil (Marketplace)**

🎯 **Objectif** : Permet aux **acheteurs** de voir les **produits agricoles disponibles** et aux **agriculteurs** de consulter leurs publications.  
📌 **Éléments UI** :

* **Barre de recherche** 🔍 avec filtres (prix, catégorie, localisation).
* **Affichage des produits** sous forme de cartes 📦 :
  + **Image du produit** 🖼️.
  + **Nom + Description courte** 📜.
  + **Prix** 💰.
  + **Localisation du vendeur** 📍.
* **Bouton "Voir détails"** ➡️ Redirection vers la **Page Détail du Produit**.
* **Icône Panier** 🛒 permettant d’accéder à la **Page du Panier**.

📌 **Règles UX** :  
✅ **Expérience intuitive** avec **scroll fluide** 📜.  
✅ **Design épuré** avec **catégories bien visibles**.

### **4️⃣ Page de Détail du Produit**

🎯 **Objectif** : Afficher les **détails complets** du produit sélectionné.  
📌 **Éléments UI** :

* **Grande image** du produit 🖼️.
* **Nom + Description complète** 📜.
* **Prix et quantité disponible** 💰📦.
* **Adresse du vendeur** 📍.
* **Bouton "Ajouter au Panier"** 🛒.
* **Bouton retour** ⬅️ pour revenir à la **Page d’Accueil**.

📌 **Règles UX** :  
✅ Priorité à la **lisibilité et aux images de qualité**.  
✅ **CTA (Call to Action) clair** et facile d’accès.

### **5️⃣ Page du Panier**

🎯 **Objectif** : Permettre aux acheteurs de **voir et modifier leur panier avant commande**.  
📌 **Éléments UI** :

* **Liste des produits sélectionnés** 📦.
* **Quantité modifiable** 🔄.
* **Total affiché** 💰.
* **Bouton "Passer la commande"** ✅ ➡️ Redirection vers la **Page de Validation de Commande**.
* **Bouton retour** ⬅️ vers **Page d’Accueil**.

📌 **Règles UX** :  
✅ **Facilité de modification** des quantités.  
✅ **Affichage clair** des informations.

### **6️⃣ Page de Validation de Commande**

🎯 **Objectif** : Permet aux acheteurs de **finaliser leur commande**.  
📌 **Éléments UI** :

* **Adresse de livraison** 📍 (**modifiable**).
* **Résumé des articles** 📦.
* **Mode de paiement sélectionné : "Paiement à la livraison"** 💰.
* **Bouton "Confirmer la commande"** ✅.

📌 **Règles UX** :  
✅ **Résumé clair et structuré**.  
✅ **Bouton de validation bien mis en avant**.

### **7️⃣ Page de Suivi des Commandes**

🎯 **Objectif** : Permet aux **acheteurs de suivre leurs commandes** et aux **agriculteurs de voir celles reçues**.  
📌 **Éléments UI** :

* Liste des **commandes avec statut** (En attente, En cours, Livrée).
* **Bouton "Voir détails"** ➡️ Informations complètes sur la commande.

📌 **Règles UX** :  
✅ **Visibilité immédiate du statut**.  
✅ **Interaction fluide et rapide**.

### **8️⃣ Dashboard de l’Agriculteur**

🎯 **Objectif** : Permet aux **agriculteurs** de **gérer leurs produits et commandes**.  
📌 **Éléments UI** :

* **Bouton "Ajouter un Produit"** 🆕.
* **Liste des produits mis en vente**.
* **Bouton "Voir commandes reçues"** 📦.

📌 **Règles UX** :  
✅ **Interface simple et rapide**.  
✅ **Graphiques minimalistes des ventes**.

### **9️⃣ Page de Messagerie (Optionnelle mais Recommandée)**

🎯 **Objectif** : Permet aux acheteurs et agriculteurs **d’échanger directement**.  
📌 **Éléments UI** :

* Liste des conversations 📩.
* Champ de saisie du message.

📌 **Règles UX** :  
✅ **Design inspiré des apps de messagerie classiques**.

## **📌 Conclusion**

🎯 **Ce cahier des charges définit toutes les interfaces essentielles** avec des détails clairs pour le **UI/UX designer**.  
🚀 **Résultat attendu : Une application fluide, intuitive et développable en 3 semaines**. ✅

# **📌 Base de Données Complète et Optimisée**

✅ **Respect des normes d’intégrité et d’ingénierie logicielle**✅ **En lien direct avec les interfaces (pages) et les scénarios utilisateurs**✅ **Évolutive et extensible** pour intégrer de futures fonctionnalités

## **1️⃣ Règles de Modélisation Appliquées**

### **✅ Principes respectés :**

1. **Normalisation (3NF)** 📊 : Évite la redondance et garantit la cohérence.
2. **Relations bien définies** 🔄 : Intégrité référentielle assurée.
3. **Respect des contraintes d’intégrité** ✅ :
   * **Clés primaires (PK)** : Garantit l’unicité des enregistrements.
   * **Clés étrangères (FK)** : Garantit la relation correcte entre les tables.
   * **Contraintes UNIQUE et CHECK** : Empêche la duplication et erreurs de saisie.

## **2️⃣ Tables et Relations**

| **📌 Table** | **🎯 Description** | **🔗 Relations Clés** |
| --- | --- | --- |
| **users** | Stocke les informations des utilisateurs (Agriculteurs & Acheteurs). | **PK** : id\_user  🔗 Un utilisateur peut publier **plusieurs produits** (*1:N* avec products).  🔗 Un acheteur peut passer **plusieurs commandes** (*1:N* avec orders). |
| **products** | Contient les produits mis en vente par les agriculteurs. | **PK** : id\_product  🔗 Un produit appartient **à un seul agriculteur** (*N:1* avec users).  🔗 Un produit peut être acheté **dans plusieurs commandes** (*M:N* via order\_items). |
| **orders** | Stocke les commandes effectuées par les acheteurs. | **PK** : id\_order  🔗 Une commande appartient **à un seul acheteur** (*N:1* avec users).  🔗 Une commande peut contenir **plusieurs produits** (*M:N* via order\_items). |
| **order\_items** | Liste les produits achetés dans chaque commande (relation M:N entre orders et products). | **PK** : id\_order\_item  🔗 Relie **une commande** et **plusieurs produits** (*N:M*). |
| **payments** | Gère les transactions liées aux commandes. | **PK** : id\_payment  🔗 Un paiement est lié **à une seule commande** (*N:1* avec orders). |
| **messages** 🆕 | Stocke les échanges entre acheteurs et agriculteurs. | **PK** : id\_message  🔗 Un message a un **expéditeur et un destinataire** (*N:N* avec users). |
| **reviews** | Contient les avis et notes des acheteurs sur les produits. | **PK** : id\_review  🔗 Un acheteur peut noter **plusieurs produits** (*N:M* avec products). |

## **3️⃣ Description Complète des Tables et Relations**

### **🔹 Table : users (Utilisateurs)**

Stocke les informations de chaque utilisateur (agriculteur ou acheteur).

| **Champ** | **Type** | **Contraintes** | **Description** |
| --- | --- | --- | --- |
| id\_user | INT | PRIMARY KEY, AUTO\_INCREMENT | Identifiant unique de l’utilisateur. |
| full\_name | VARCHAR(255) | NOT NULL | Nom complet. |
| phone | VARCHAR(15) | NOT NULL, UNIQUE | Numéro de téléphone unique. |
| email | VARCHAR(255) | UNIQUE | Email de l’utilisateur. |
| password | VARCHAR(255) | NOT NULL | Mot de passe crypté. |
| role | ENUM(‘Agriculteur’, ‘Acheteur’) | NOT NULL | Type d’utilisateur. |
| address | TEXT | NOT NULL | Adresse de l’utilisateur. |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | Date d’inscription. |

🔗 **Relations Clés** :

* **Un utilisateur peut publier plusieurs produits.**
* **Un acheteur peut passer plusieurs commandes.**
* **Un utilisateur peut envoyer et recevoir des messages.**

### **🔹 Table : products (Produits)**

Stocke les produits mis en vente par les agriculteurs.

| **Champ** | **Type** | **Contraintes** | **Description** |
| --- | --- | --- | --- |
| id\_product | INT | PRIMARY KEY, AUTO\_INCREMENT | Identifiant du produit. |
| user\_id | INT | FOREIGN KEY (users.id\_user) | L’agriculteur qui vend le produit. |
| name | VARCHAR(255) | NOT NULL | Nom du produit. |
| description | TEXT | NOT NULL | Détails du produit. |
| price | DECIMAL(10,2) | NOT NULL | Prix unitaire. |
| stock | INT | NOT NULL, CHECK(stock >= 0) | Quantité disponible. |
| category | VARCHAR(50) | NOT NULL | Catégorie du produit. |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | Date d’ajout du produit. |

🔗 **Relations Clés** :

* **Un agriculteur peut publier plusieurs produits.**
* **Un produit peut être commandé plusieurs fois (relation M:N avec orders).**

### **🔹 Table : orders (Commandes)**

Stocke les commandes passées par les acheteurs.

| **Champ** | **Type** | **Contraintes** | **Description** |
| --- | --- | --- | --- |
| id\_order | INT | PRIMARY KEY, AUTO\_INCREMENT | Identifiant de la commande. |
| user\_id | INT | FOREIGN KEY (users.id\_user) | L’acheteur qui a passé la commande. |
| total\_amount | DECIMAL(10,2) | NOT NULL | Montant total. |
| status | ENUM('En attente', 'Confirmée', 'Livrée') | NOT NULL | Statut de la commande. |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | Date de création de la commande. |

🔗 **Relations Clés** :

* **Une commande est passée par un seul acheteur.**
* **Une commande peut contenir plusieurs produits (via order\_items).**

### **🔹 Table : order\_items (Produits commandés)**

Gère la relation **M:N** entre orders et products.

| **Champ** | **Type** | **Contraintes** | **Description** |
| --- | --- | --- | --- |
| id\_order\_item | INT | PRIMARY KEY, AUTO\_INCREMENT | Identifiant unique. |
| order\_id | INT | FOREIGN KEY (orders.id\_order) | Commande associée. |
| product\_id | INT | FOREIGN KEY (products.id\_product) | Produit commandé. |
| quantity | INT | NOT NULL, CHECK(quantity > 0) | Quantité commandée. |

### **🔹 Table : payments (Paiements)**

Stocke les transactions liées aux commandes.

| **Champ** | **Type** | **Contraintes** | **Description** |
| --- | --- | --- | --- |
| id\_payment | INT | PRIMARY KEY, AUTO\_INCREMENT | Identifiant du paiement. |
| order\_id | INT | FOREIGN KEY (orders.id\_order) | Commande payée. |
| amount | DECIMAL(10,2) | NOT NULL | Montant du paiement. |
| status | ENUM('En attente', 'Réglé') | NOT NULL | Statut du paiement. |

## **📌 Conclusion**

🎯 **La base de données respecte** : ✅ **Toutes les règles d’intégrité et d’ingénierie logicielle**.  
✅ **Une structure optimisée avec des relations claires**.  
✅ **Facilité d’extension future (livraisons, promotions, notifications, etc.)**.

**🚀 Résultat attendu : Un système robuste, évolutif et prêt à être implémenté ! ✅**

# **📌 Backend Sécurisé et Scalable avec FastAPI (ou Django REST Framework)**

✅ **Backend structuré en respectant l’architecture MVC**✅ **Endpoints complets et optimisés** pour chaque fonctionnalité  
✅ **Sécurisé, évolutif et rapide**

## **1️⃣ Choix de la Technologie**

| **Technologie** | **Justification** |
| --- | --- |
| **FastAPI** (Recommandé) | 🔥 Très rapide et optimisé pour les API REST, validation automatique des entrées. |
| **Django REST Framework (DRF)** | 🏗️ Bonne structure, facile à maintenir, intégré avec Django ORM. |
| **Base de Données : PostgreSQL ou MySQL** | 💾 Base robuste et performante. |
| **JWT (JSON Web Token)** | 🔐 Authentification sécurisée des utilisateurs. |

## **2️⃣ Organisation du Backend**

Le backend suit une architecture **MVC (Modèle-Vue-Contrôleur)** :

📂 **backend/**├── 📂 models/ ➝ Définition des classes de la base de données (ORM)  
├── 📂 routes/ ➝ Définition des **endpoints API**├── 📂 controllers/ ➝ Gestion de la logique métier  
├── 📂 middlewares/ ➝ Sécurité et authentification  
├── 📂 config/ ➝ Connexion à la base de données  
├── 📂 tests/ ➝ Tests unitaires et d’intégration  
└── 📜 main.py ➝ Point d’entrée de l’API

## **3️⃣ Endpoints API Complets**

Voici les **endpoints** qui correspondent aux fonctionnalités de l’application mobile.

### **📌 1. Authentification (Login/Signup)**

| **Méthode** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /auth/signup | Inscription d’un utilisateur (Agriculteur/Acheteur). |
| POST | /auth/login | Connexion avec JWT. |
| GET | /auth/me | Récupérer les informations de l’utilisateur connecté. |
| POST | /auth/logout | Déconnexion de l’utilisateur. |

### **📌 2. Gestion des Produits**

| **Méthode** | **Endpoint** | **Description** |
| --- | --- | --- |
| GET | /products/ | Liste de tous les produits (Marketplace). |
| GET | /products/{id} | Détails d’un produit spécifique. |
| POST | /products/ | Ajout d’un nouveau produit (Agriculteur uniquement). |
| PUT | /products/{id} | Mise à jour d’un produit existant. |
| DELETE | /products/{id} | Suppression d’un produit. |

### **📌 3. Gestion du Panier**

| **Méthode** | **Endpoint** | **Description** |
| --- | --- | --- |
| GET | /cart/ | Afficher le panier de l’utilisateur. |
| POST | /cart/add | Ajouter un produit au panier. |
| PUT | /cart/update/{id} | Modifier la quantité d’un produit dans le panier. |
| DELETE | /cart/remove/{id} | Supprimer un produit du panier. |

### **📌 4. Commandes et Suivi**

| **Méthode** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /orders/create | Création d’une nouvelle commande. |
| GET | /orders/ | Liste des commandes de l’utilisateur connecté. |
| GET | /orders/{id} | Détails d’une commande spécifique. |
| PUT | /orders/{id}/status | Mise à jour du statut d’une commande. |

### **📌 5. Paiements (Paiement à la Livraison Priorisé)**

| **Méthode** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /payments/cash | Enregistrement d’un paiement à la livraison. |
| GET | /payments/{id} | Détails d’un paiement effectué. |

### **📌 6. Gestion des Messages (Chat Acheteur-Agriculteur)**

| **Méthode** | **Endpoint** | **Description** |
| --- | --- | --- |
| GET | /messages/ | Récupérer les messages de l’utilisateur connecté. |
| POST | /messages/send | Envoyer un message à un utilisateur. |

### **📌 7. Avis et Notes**

| **Méthode** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /reviews/{product\_id} | Laisser un avis sur un produit. |
| GET | /reviews/{product\_id} | Voir les avis sur un produit. |

## **4️⃣ Sécurité et Optimisation**

✅ **Sécurité** :

* 🔐 **JWT (JSON Web Token)** pour authentification.
* 🛡️ **Validation des entrées utilisateur** pour éviter les injections SQL/XSS.
* 🚧 **Rate Limiting** (Limiter le nombre de requêtes pour éviter les abus).

✅ **Optimisation** :

* ⚡ **Requêtes asynchrones (FastAPI)** pour accélérer le temps de réponse.
* 📦 **Pagination** sur les endpoints /products/ et /orders/ pour éviter la surcharge.

## **📌 Conclusion**

🎯 **Backend complet et optimisé** avec toutes les **routes nécessaires** pour le **frontend Flutter**.  
🚀 **Résultat attendu** : API sécurisée, rapide et scalable, parfaitement intégrée à la base de données. ✅