Document de Spécifications Fonctionnelles (DSF)

# **Africage - Super-App P2P Panafricaine**

Phase 1 : Service d'Envoi de Colis P2P

Version : 1.0

Date : 27 juin 2025

Auteur : Équipe Produit Africage

# **1. Introduction**

Ce document sert de guide détaillé pour le développement de l'application mobile Africage, Phase 1 : service d'envoi de colis P2P panafricain. Il décrit toutes les fonctionnalités requises en distinguant clairement les aspects frontend et backend pour faciliter la compréhension et le développement par les équipes techniques. Africage vise à révolutionner la logistique P2P en Afrique en connectant expéditeurs de colis et voyageurs vérifiés.

## **1.1 Objectifs du Projet**

**•** Révolutionner la logistique P2P en Afrique en connectant expéditeurs et voyageurs.

**•** Créer une économie collaborative permettant aux voyageurs de monétiser leurs trajets.

**•** Construire un écosystème de confiance via vérification d'identité et système de notation.

**•** Fournir une expérience utilisateur optimisée pour les réalités africaines (connectivité, appareils).

**•** Poser les bases architecture pour intégration future (transfert d'argent, mobilité).

## **1.2 Portée de l'Application**

L'application couvrira la gestion des utilisateurs (expéditeurs/voyageurs), création/gestion des annonces de colis, soumission/acceptation des propositions de transport, suivi de livraison, système de paiement séquestré via Mobile Money/cartes bancaires, et évaluations mutuelles. Phase 1 exclut transfert d'argent et mobilité.

## **1.3 Contexte Technique**

• Type d'Application : Application mobile native React Native (iOS/Android) + Dashboard web admin  
• Frontend Mobile : React Native avec NativeBase/Tamagui pour UI components  
• Backend : Node.js avec Express.js et TypeScript  
• Base de Données : PostgreSQL avec Prisma ORM  
• Authentification : JWT + SMS OTP (priorité sur email en Afrique)  
• Paiements : Intégration Paystack (Mobile Money + cartes) pour marché africain  
• Storage : AWS S3 pour images et documents  
• Maps : Mapbox (meilleur support Afrique vs Google Maps)  
• Push Notifications : Firebase Cloud Messaging  
  
Note pour l'équipe dev : Développer mobile app en premier, puis dashboard admin, enfin API Gateway pour scaling.

# **2. Architecture Fonctionnelle Générale**

L'application sera divisée en trois couches principales :  
  
• Couche Mobile (React Native) : Interface utilisateur mobile, interactions, logique de présentation, appels API backend, gestion cache local, géolocalisation.  
  
• Couche Backend (Node.js/Express) : Logique métier, accès base de données, authentification JWT, intégrations paiements/SMS, API RESTful, système de notifications push.  
  
• Couche Admin (React Web) : Dashboard administration pour validation KYC, gestion des litiges, analytics, modération contenus.  
  
Les interactions se feront via des requêtes HTTP/HTTPS sécurisées (JWT) vers les API backend RESTful. WebSockets pour messagerie en temps réel et notifications push.

# **3. Spécifications Fonctionnelles Détaillées**

Cette section décompose Africage en modules et composants, spécifiant pour chacun ses aspects frontend mobile et backend.

## **3.1 Module : Authentification & Gestion des Comptes**

### **3.1.1 Fonctionnalité : Inscription**

• Rôles : Expéditeur, Voyageur (Tous les utilisateurs)  
  
• Frontend Mobile :  
 ○ Composant : SignUpScreen.tsx, PhoneVerificationScreen.tsx  
 ○ Description : Écran d'inscription avec champs "Numéro de téléphone" (format international avec sélecteur pays), "Nom", "Prénom", "Mot de passe", "Confirmer mot de passe". Option "S'inscrire avec Google" (optionnelle).  
 ○ Validation UX : Validation côté client (format téléphone, complexité mot de passe, correspondance mots de passe). Messages d'erreur sous chaque champ.  
 ○ Flux : Saisie infos → Envoi OTP SMS → Vérification code OTP → Création compte  
 ○ Interaction : Envoi données à POST /api/auth/register puis vérification via POST /api/auth/verify-phone  
 ○ Retour UX : Succès → Redirection vers écran "Vérification d'identité". Échec → Message d'erreur ("Numéro déjà utilisé", "Code OTP invalide").  
  
• Backend :  
 ○ Endpoint : POST /api/auth/register  
 ○ Description : Recevoir infos inscription. Générer et envoyer code OTP SMS via Africa's Talking API. Stocker temporairement en Redis (5 min TTL).  
 ○ Validation Serveur : Unicité numéro téléphone, complexité mot de passe, format téléphone valide.  
 ○ Modèle Données (User) : id (UUID), phoneNumber (string, unique), firstName (string), lastName (string), hashedPassword (string), isPhoneVerified (boolean), kycStatus (enum: 'pending', 'submitted', 'verified', 'rejected'), role (enum: 'expeditor', 'voyageur', 'admin'), averageRating (decimal), totalTransactions (integer), createdAt (timestamp).  
 ○ Réponse API Succès (201) : { "message": "Code OTP envoyé", "tempUserId": "...", "expiresIn": 300 }  
 ○ Réponse API Erreur (400/409) : { "error": "Numéro déjà utilisé" }  
  
 ○ Endpoint : POST /api/auth/verify-phone  
 ○ Description : Vérifier code OTP, créer utilisateur si valide, générer JWT token.  
 ○ Réponse Succès (201) : { "message": "Compte créé", "token": "...", "user": {...} }

### **3.1.2 Fonctionnalité : Connexion**

• Rôles : Expéditeur, Voyageur (Tous les utilisateurs)  
  
• Frontend Mobile :  
 ○ Composant : LoginScreen.tsx  
 ○ Description : Écran connexion avec champs "Numéro de téléphone", "Mot de passe". Boutons "Se connecter", "Mot de passe oublié ?", "Créer un compte".  
 ○ Interaction : Envoi données à POST /api/auth/login  
 ○ Retour UX : Succès → Stockage token secure storage + redirection HomePage. Échec → Message "Numéro ou mot de passe incorrect".  
  
• Backend :  
 ○ Endpoint : POST /api/auth/login  
 ○ Description : Vérifier identifiants, générer JWT token (24h TTL) si valide.  
 ○ Réponse Succès (200) : { "message": "Connexion réussie", "token": "...", "user": { "id": "...", "firstName": "...", "kycStatus": "...", "averageRating": ... } }  
 ○ Réponse Erreur (401) : { "error": "Identifiants invalides" }

### **3.1.3 Fonctionnalité : Gestion du Profil**

• Rôles : Expéditeur, Voyageur  
  
• Frontend Mobile :  
 ○ Composants : ProfileScreen.tsx, EditProfileScreen.tsx, KYCScreen.tsx  
 ○ Description : Affichage infos utilisateur (nom, téléphone, photo profil, statut KYC, note moyenne, nombre transactions). Bouton "Modifier profil", "Vérifier identité".  
 ○ Interaction :   
 ■ Récupération : GET /api/users/me  
 ■ Mise à jour : PUT /api/users/me  
 ■ Upload photo : POST /api/users/me/avatar (multipart/form-data)  
  
• Backend :  
 ○ Endpoint : GET /api/users/me (auth JWT requis)  
 ○ Description : Retourne profil utilisateur authentifié.  
 ○ Réponse (200) : { "id": "...", "firstName": "...", "lastName": "...", "phoneNumber": "...", "profilePictureUrl": "...", "kycStatus": "...", "averageRating": 4.2, "totalTransactions": 15 }  
  
 ○ Endpoint : PUT /api/users/me (auth JWT requis)  
 ○ Description : Met à jour infos profil (firstName, lastName, bio).  
 ○ Réponse (200) : { "message": "Profil mis à jour" }

### **3.1.4 Fonctionnalité : Vérification d'Identité (KYC)**

• Rôles : Expéditeur, Voyageur (Obligatoire pour utiliser la plateforme)  
  
• Frontend Mobile :  
 ○ Composant : KYCSubmissionScreen.tsx, DocumentCameraScreen.tsx  
 ○ Description : Écran upload document identité (CNI, Passeport, Permis). Utilisation caméra native pour photo. Guide visuel pour cadrage optimal. Champs : Type document, Pays émission.  
 ○ Interaction : POST /api/users/kyc/submit (multipart/form-data)  
 ○ Retour UX : Confirmation soumission → "Document en cours de vérification (24-48h)"  
  
• Backend :  
 ○ Endpoint : POST /api/users/kyc/submit (auth JWT requis)  
 ○ Description : Recevoir document photo, stocker sur AWS S3, marquer kycStatus = 'submitted', créer entrée KYCDocument.  
 ○ Modèle KYCDocument : id, userId, documentType, documentUrl, submittedAt, reviewedAt, status, adminComments.  
 ○ Réponse (201) : { "message": "Document soumis pour vérification", "estimatedReviewTime": "24-48h" }  
  
 Admin Backend :  
 ○ Dashboard pour validation manuelle des documents avec approve/reject + commentaires.

## **3.2 Module : Annonces de Colis (Côté Expéditeur)**

### **3.2.1 Fonctionnalité : Créer une Annonce de Colis**

• Rôle : Expéditeur  
  
• Frontend Mobile :  
 ○ Composant : CreateParcelScreen.tsx, AddressPickerScreen.tsx, PhotoCapture.tsx  
 ○ Description : Formulaire multi-étapes avec :  
 ■ Étape 1 : Titre colis, Description, Catégorie (dropdown : Documents, Électronique, Vêtements, Alimentaire non-périssable, Autre)  
 ■ Étape 2 : Taille (Petit <2kg, Moyen 2-10kg, Grand 10-25kg), Poids estimé (slider)  
 ■ Étape 3 : Adresses départ/arrivée avec Mapbox autocomplete + géolocalisation actuelle  
 ■ Étape 4 : Dates souhaitées (DatePicker pour envoi/réception)  
 ■ Étape 5 : Photos colis (min 1, max 3) avec appareil photo natif  
 ■ Étape 6 : Prix proposé (optionnel) ou "Ouvert aux propositions"  
 ○ Validation UX : Tous champs obligatoires remplis, adresses géolocalisées, photos nettes.  
 ○ Interaction : POST /api/parcels (multipart/form-data)  
 ○ Retour UX : Succès → "Annonce publiée ! Les voyageurs vont recevoir une notification." + redirection vers MyParcels  
  
• Backend :  
 ○ Endpoint : POST /api/parcels (auth JWT requis + kycStatus = 'verified')  
 ○ Description : Valider données, géocoder adresses via Mapbox, uploader images S3, créer entrée Parcel, notifier voyageurs dans zone géographique.  
 ○ Modèle Parcel : id (UUID), expeditorId (ref), title, description, category, size, weight, originAddress, destinationAddress, originLatLng (Point), destLatLng (Point), desiredPickupDate, desiredDeliveryDate, proposedPrice (nullable), imageUrls (array), status ('pending', 'matched', 'picked-up', 'in-transit', 'delivered', 'cancelled'), activeProposalId (nullable), createdAt.  
 ○ Réponse (201) : { "message": "Annonce créée", "parcelId": "...", "estimatedMatches": 3 }  
 ○ Business Logic : Notification push aux voyageurs ayant trajets compatibles (dans rayon 50km + dates ±2 jours)

### **3.2.2 Autres Fonctionnalités Expéditeur**

• Consulter/Gérer mes Annonces :  
 ○ Frontend : MyParcelsScreen.tsx avec TabView (Active, En cours, Terminées)  
 ○ Backend : GET /api/parcels/my  
 ○ Affichage : Liste avec statut, nombre propositions reçues, actions (Modifier/Supprimer si pending)  
  
• Consulter Propositions Reçues :  
 ○ Frontend : ProposalsScreen.tsx avec profils voyageurs (photo, nom, note, prix proposé)  
 ○ Backend : GET /api/parcels/{parcelId}/proposals  
 ○ Actions : Consulter profil complet, Accepter proposition  
  
• Accepter une Proposition :  
 ○ Frontend : AcceptProposalModal.tsx avec résumé transaction + initiation paiement  
 ○ Backend : POST /api/proposals/{proposalId}/accept  
 ○ Logic : Marquer parcel.status = 'matched', rejeter autres propositions, initier paiement Paystack  
 ○ Paiement : Intégration Paystack pour Mobile Money/carte, fonds séquestrés jusqu'à livraison  
  
• Suivi en Temps Réel :  
 ○ Frontend : TrackingScreen.tsx avec carte Mapbox + statuts (Payé → Récupéré → En Transit → Livré)  
 ○ Backend : GET /api/parcels/{parcelId}/tracking + WebSocket pour live updates  
 ○ Features : Position voyageur en temps réel (si activée), messagerie intégrée, notifications push étapes

## **3.3 Module : Recherche & Proposition (Côté Voyageur)**

• Publier un Trajet :  
 ○ Frontend : CreateTripScreen.tsx  
 ○ Champs : Départ, Arrivée, Date voyage, Espace disponible (Petit/Moyen/Grand), Types colis acceptés  
 ○ Backend : POST /api/trips  
  
• Rechercher Colis Disponibles :  
 ○ Frontend : SearchParcelsScreen.tsx avec filtres (Lieu, Date, Taille, Prix max)  
 ○ Backend : GET /api/parcels/available?lat={}&lng={}&radius={}&date={}  
 ○ Affichage : CarteColisDisponible.tsx avec distance, prix, profil expéditeur  
  
• Soumettre Proposition :  
 ○ Frontend : CreateProposalScreen.tsx  
 ○ Champs : Prix proposé, Horaires confirmés récup/livraison, Message personnalisé  
 ○ Backend : POST /api/parcels/{parcelId}/proposals  
 ○ Logic : Vérifier colis toujours disponible, créer Proposal, notifier expéditeur  
  
• Gérer Mes Propositions :  
 ○ Frontend : MyProposalsScreen.tsx (Envoyées, Acceptées, Refusées)  
 ○ Backend : GET /api/proposals/my  
  
• Confirmer Récupération/Livraison :  
 ○ Frontend : ConfirmPickupScreen.tsx, ConfirmDeliveryScreen.tsx  
 ○ Features : Photo de confirmation, Géolocalisation obligatoire, Signature digitale destinataire  
 ○ Backend : PUT /api/parcels/{parcelId}/pickup, PUT /api/parcels/{parcelId}/deliver  
 ○ Logic : Mettre à jour statuts, notifier expéditeur, déclencher libération fonds (delivery)

## **3.4 Module : Messagerie Instantanée**

• Frontend Mobile :  
 ○ Composant : ChatScreen.tsx avec FlatList messages, TextInput + bouton envoi  
 ○ Features : Bulles messages (gauche/droite), timestamp, statut lu/non-lu, photo/localisation  
 ○ Interaction : WebSocket connexion + fallback polling 30s  
 ○ Envoi : POST /api/chats/{transactionId}/messages  
  
• Backend :  
 ○ WebSocket Server : Socket.io pour temps réel  
 ○ Endpoint : POST /api/chats/{transactionId}/messages (auth JWT)  
 ○ Modèle Message : id, transactionId, senderId, receiverId, content, messageType ('text', 'image', 'location'), timestamp, isRead  
 ○ Logic : Sauvegarder message DB, émettre via Socket.io, push notification si offline  
 ○ Règles : Messagerie limitée aux transactions actives, modération auto contenus inappropriés

## **3.5 Module : Notation & Avis**

• Frontend Mobile :  
 ○ Composant : RatingModal.tsx (déclenché après confirmation livraison)  
 ○ UI : StarRating component (1-5 étoiles), TextArea commentaire optionnel, boutons "Passer" / "Envoyer avis"  
 ○ Interaction : POST /api/reviews  
  
• Backend :  
 ○ Endpoint : POST /api/reviews (auth JWT)  
 ○ Modèle Review : id, transactionId, reviewerId, revieweeId, rating (1-5), comment, createdAt  
 ○ Logic : Une seule review par transaction/role, recalculer averageRating utilisateur noté  
 ○ Business Rules : Review obligatoire pour débloquer futures transactions (incentive qualité)

# **4. Modèle de Données (Schema PostgreSQL)**

• Users (table utilisateurs)  
 ○ id: UUID PRIMARY KEY  
 ○ phone\_number: VARCHAR(20) UNIQUE NOT NULL  
 ○ first\_name: VARCHAR(100) NOT NULL  
 ○ last\_name: VARCHAR(100) NOT NULL  
 ○ hashed\_password: VARCHAR(255) NOT NULL  
 ○ profile\_picture\_url: TEXT  
 ○ kyc\_status: ENUM('pending', 'submitted', 'verified', 'rejected') DEFAULT 'pending'  
 ○ role: ENUM('expeditor', 'voyageur', 'admin') DEFAULT 'expeditor'  
 ○ average\_rating: DECIMAL(2,1) DEFAULT 0.0  
 ○ total\_transactions: INTEGER DEFAULT 0  
 ○ is\_phone\_verified: BOOLEAN DEFAULT FALSE  
 ○ is\_active: BOOLEAN DEFAULT TRUE  
 ○ created\_at: TIMESTAMP DEFAULT NOW()  
 ○ updated\_at: TIMESTAMP DEFAULT NOW()  
  
• Parcels (table colis)  
 ○ id: UUID PRIMARY KEY  
 ○ expeditor\_id: UUID REFERENCES users(id)  
 ○ title: VARCHAR(200) NOT NULL  
 ○ description: TEXT  
 ○ category: ENUM('documents', 'electronics', 'clothing', 'food', 'other')  
 ○ size: ENUM('small', 'medium', 'large')  
 ○ weight: DECIMAL(5,2)  
 ○ origin\_address: TEXT NOT NULL  
 ○ destination\_address: TEXT NOT NULL  
 ○ origin\_location: POINT NOT NULL -- PostGIS pour géolocalisation  
 ○ destination\_location: POINT NOT NULL  
 ○ desired\_pickup\_date: DATE  
 ○ desired\_delivery\_date: DATE  
 ○ proposed\_price: DECIMAL(10,2)  
 ○ image\_urls: TEXT[] -- Array URLs images  
 ○ status: ENUM('pending', 'matched', 'picked-up', 'in-transit', 'delivered', 'cancelled') DEFAULT 'pending'  
 ○ active\_proposal\_id: UUID REFERENCES proposals(id)  
 ○ created\_at: TIMESTAMP DEFAULT NOW()  
  
• Proposals (table propositions transport)  
 ○ id: UUID PRIMARY KEY  
 ○ parcel\_id: UUID REFERENCES parcels(id)  
 ○ voyageur\_id: UUID REFERENCES users(id)  
 ○ proposed\_price: DECIMAL(10,2) NOT NULL  
 ○ pickup\_date: DATE NOT NULL  
 ○ delivery\_date: DATE NOT NULL  
 ○ message\_to\_expeditor: TEXT  
 ○ status: ENUM('pending', 'accepted', 'rejected', 'expired') DEFAULT 'pending'  
 ○ created\_at: TIMESTAMP DEFAULT NOW()  
  
• Transactions (table transactions complètes)  
 ○ id: UUID PRIMARY KEY  
 ○ parcel\_id: UUID REFERENCES parcels(id)  
 ○ proposal\_id: UUID REFERENCES proposals(id)  
 ○ expeditor\_id: UUID REFERENCES users(id)  
 ○ voyageur\_id: UUID REFERENCES users(id)  
 ○ amount: DECIMAL(10,2) NOT NULL  
 ○ commission: DECIMAL(10,2) NOT NULL  
 ○ payment\_status: ENUM('pending', 'authorized', 'captured', 'refunded') DEFAULT 'pending'  
 ○ transaction\_status: ENUM('created', 'pickup-confirmed', 'in-transit', 'delivered', 'completed', 'disputed')  
 ○ paystack\_reference: VARCHAR(100)  
 ○ pickup\_confirmed\_at: TIMESTAMP  
 ○ delivery\_confirmed\_at: TIMESTAMP  
 ○ current\_location: POINT -- Position actuelle voyageur  
 ○ proof\_of\_delivery\_url: TEXT  
 ○ created\_at: TIMESTAMP DEFAULT NOW()  
  
• Messages (table messagerie)  
 ○ id: UUID PRIMARY KEY  
 ○ transaction\_id: UUID REFERENCES transactions(id)  
 ○ sender\_id: UUID REFERENCES users(id)  
 ○ receiver\_id: UUID REFERENCES users(id)  
 ○ content: TEXT NOT NULL  
 ○ message\_type: ENUM('text', 'image', 'location') DEFAULT 'text'  
 ○ is\_read: BOOLEAN DEFAULT FALSE  
 ○ created\_at: TIMESTAMP DEFAULT NOW()  
  
• Reviews (table avis et notations)  
 ○ id: UUID PRIMARY KEY  
 ○ transaction\_id: UUID REFERENCES transactions(id)  
 ○ reviewer\_id: UUID REFERENCES users(id)  
 ○ reviewee\_id: UUID REFERENCES users(id)  
 ○ rating: INTEGER CHECK (rating >= 1 AND rating <= 5)  
 ○ comment: TEXT  
 ○ created\_at: TIMESTAMP DEFAULT NOW()  
  
• KYC\_Documents (table documents identité)  
 ○ id: UUID PRIMARY KEY  
 ○ user\_id: UUID REFERENCES users(id)  
 ○ document\_type: ENUM('national\_id', 'passport', 'driving\_license')  
 ○ document\_url: TEXT NOT NULL  
 ○ status: ENUM('submitted', 'approved', 'rejected') DEFAULT 'submitted'  
 ○ admin\_comments: TEXT  
 ○ submitted\_at: TIMESTAMP DEFAULT NOW()  
 ○ reviewed\_at: TIMESTAMP

# **5. Spécifications des API Backend**

Exemples détaillés d'API critiques pour guide développement :

## **Exemple 1 : Créer une Annonce de Colis**

POST /api/parcels  
  
Authentification : JWT Token requis (header Authorization: Bearer <token>)  
Content-Type : multipart/form-data  
  
Request Body :  
{  
 "title": "Documents professionnels urgents",  
 "description": "Contrat de travail et diplômes dans enveloppe scellée",  
 "category": "documents",  
 "size": "small",  
 "weight": 0.5,  
 "originAddress": "Plateau, Abidjan, Côte d'Ivoire",  
 "destinationAddress": "Cocody, Abidjan, Côte d'Ivoire",  
 "desiredPickupDate": "2025-07-01",  
 "desiredDeliveryDate": "2025-07-01",  
 "proposedPrice": 2000,  
 "images": [File, File] // Files multipart  
}  
  
Response Success (201 Created) :  
{  
 "success": true,  
 "message": "Annonce de colis créée avec succès",  
 "data": {  
 "parcelId": "550e8400-e29b-41d4-a716-446655440000",  
 "estimatedMatches": 5,  
 "createdAt": "2025-06-27T10:30:00Z"  
 }  
}  
  
Response Error (400 Bad Request) :  
{  
 "success": false,  
 "error": "VALIDATION\_ERROR",  
 "message": "Données invalides",  
 "details": {  
 "weight": "Le poids doit être supérieur à 0",  
 "images": "Au moins une photo est requise"  
 }  
}

## **Exemple 2 : Rechercher Colis Disponibles**

GET /api/parcels/available  
  
Authentification : JWT Token requis  
Query Parameters :  
- lat: Latitude position actuelle (required)  
- lng: Longitude position actuelle (required)  
- radius: Rayon recherche en km (default: 50, max: 200)  
- maxWeight: Poids maximum acceptable (optional)  
- category: Catégorie colis (optional)  
- pickupDate: Date souhaitée récupération (optional)  
  
Example Request :  
GET /api/parcels/available?lat=5.3364&lng=-4.0267&radius=30&maxWeight=10&pickupDate=2025-07-01  
  
Response Success (200 OK) :  
{  
 "success": true,  
 "data": {  
 "parcels": [  
 {  
 "id": "parcel-uuid-1",  
 "title": "Documents professionnels urgents",  
 "category": "documents",  
 "size": "small",  
 "weight": 0.5,  
 "proposedPrice": 2000,  
 "distance": 12.5,  
 "originAddress": "Plateau, Abidjan",  
 "destinationAddress": "Cocody, Abidjan",  
 "expeditor": {  
 "id": "user-uuid-1",  
 "firstName": "Fatou",  
 "averageRating": 4.8,  
 "totalTransactions": 23,  
 "profilePictureUrl": "https://..."  
 },  
 "imageUrls": ["https://s3.../image1.jpg"],  
 "createdAt": "2025-06-27T10:30:00Z"  
 }  
 ],  
 "totalCount": 12,  
 "pagination": {  
 "page": 1,  
 "limit": 20,  
 "hasNext": false  
 }  
 }  
}

# **6. Exigences Non Fonctionnelles**

• Performance :  
 ○ Chargement initial app mobile : < 4 secondes sur 3G  
 ○ Temps réponse API CRUD : < 500ms (95% des requêtes)  
 ○ Recherche géospatiale : < 800ms pour rayon 100km  
 ○ Upload image : < 10 secondes pour 3MB sur 4G  
 ○ WebSocket messages : Latence < 200ms  
  
• Sécurité :  
 ○ Toutes API authentifiées JWT (24h TTL, refresh token 7 jours)  
 ○ Autorisation basée rôles (expéditeur ne peut modifier colis voyageur)  
 ○ Validation stricte inputs (sanitization, rate limiting)  
 ○ Paiements tokenisés Paystack (pas stockage direct infos bancaires)  
 ○ Images scannées anti-malware avant stockage S3  
 ○ Chiffrement données sensibles au repos (AES-256)  
  
• Scalabilité :  
 ○ Architecture cloud-native AWS (auto-scaling ECS, RDS Multi-AZ)  
 ○ CDN CloudFront pour images (optimisation bande passante Afrique)  
 ○ Cache Redis pour sessions et recherches fréquentes  
 ○ Database sharding par zones géographiques (préparation expansion)  
  
• Expérience Utilisateur :  
 ○ Design System cohérent (couleurs, typographie, animations)  
 ○ Support hors-ligne (cache dernières transactions, mode lecture)  
 ○ Optimisation batterie (géolocalisation intelligente, dark mode)  
 ○ Accessibilité (VoiceOver, contraste élevé, tailles police ajustables)  
 ○ Onboarding progressif avec tutoriels interactifs  
 ○ Notifications push intelligentes (pas de spam, timing optimal)  
  
• Robustesse :  
 ○ Backend retry logic avec exponential backoff  
 ○ Circuit breakers pour services externes (Paystack, SMS)  
 ○ Monitoring APM complet (NewRelic, logs structurés)  
 ○ Error tracking client (Sentry) avec contexte utilisateur  
 ○ Health checks automatisés + alerting Slack  
  
• Conformité & Légal :  
 ○ RGPD compliance (opt-in explicite, droit à l'oubli)  
 ○ Réglementations locales transport marchandises par particuliers  
 ○ KYC conforme standards bancaires africains  
 ○ Conditions d'utilisation adaptées par pays  
 ○ Processus résolution litiges transparent (médiation 48h)

# **7. Scénarios d'Utilisation Détaillés**

## **Scénario : Cycle de Vie Complet Transaction P2P**

Acteurs : Fatou (Expéditrice, Abidjan) et Kwame (Voyageur, trajet Abidjan→Bouaké)  
  
Étape 1 - Inscription et Vérification :  
1. Fatou télécharge Africage, s'inscrit avec numéro +225 07 XX XX XX XX  
2. Reçoit code OTP SMS, le saisit, compte créé  
3. Upload photo CNI via KYCScreen, statut "En cours de vérification"  
4. Admin valide document après 6h, statut passe "Vérifié"  
5. Kwame suit même processus, également vérifié  
  
Étape 2 - Création Annonce :  
6. Fatou ouvre CreateParcelScreen  
7. Saisit : "Documents familiaux urgents", catégorie Documents, taille Petit  
8. Adresses : Plateau Abidjan → Bouaké centre, date demain  
9. Prend photo enveloppe scellée, prix proposé 2000 FCFA  
10. Soumet annonce → API créé, notification envoyée voyageurs zone  
  
Étape 3 - Découverte et Proposition :  
11. Kwame reçoit push notification "Nouveau colis sur votre trajet"  
12. Ouvre SearchParcelsScreen, voit annonce Fatou (5 km de sa position)  
13. Consulte profil Fatou (4.8★, 15 transactions), clic "Faire une proposition"  
14. CreateProposalScreen : propose 1500 FCFA, confirme horaires, message "Transport soigné garanti"  
15. Soumet → API créé proposition, notification Fatou  
  
Étape 4 - Acceptation et Paiement :  
16. Fatou reçoit notification, consulte proposition Kwame (4.9★, 32 transactions)  
17. Accepte proposition → AcceptProposalModal avec résumé 1500 + 150 commission  
18. Initie paiement via Paystack Mobile Money Orange Money  
19. Paiement autorisé et séquestré, statut colis passe "Matched"  
20. Kwame reçoit notification "Paiement sécurisé, récupération autorisée"  
  
Étape 5 - Communication et Coordination :  
21. ChatScreen s'active entre Fatou et Kwame  
22. Échangent coordonnées de rencontre : "Devant pharmacie Plateau, 14h demain"  
23. Kwame envoie sa position en temps réel via WebSocket  
  
Étape 6 - Récupération :  
24. Lendemain 14h, rencontre physique  
25. Kwame vérifie colis correspond description, prend photo confirmation  
26. Clic "Confirmer récupération" → géolocalisation vérifiée, photo uploadée  
27. Statut passe "Récupéré", Fatou reçoit notification + photo  
  
Étape 7 - Transport et Suivi :  
28. Kwame active partage position, démarre voyage Abidjan→Bouaké  
29. Fatou suit trajet temps réel sur TrackingScreen avec carte Mapbox  
30. Statut automatique "En Transit" dès mouvement détecté  
  
Étape 8 - Livraison :  
31. Arrivée Bouaké, Kwame livre à famille Fatou  
32. Demande signature digitale destinataire sur son téléphone  
33. Photo remise + signature → "Confirmer livraison"  
34. Statut passe "Livré", Fatou notifiée immédiatement  
  
Étape 9 - Confirmation et Paiement :  
35. Fatou appelle famille, confirme réception bonne condition  
36. Clic "Confirmer réception" → Déclenche libération fonds  
37. 1500 FCFA transférés compte Orange Money Kwame (commission 150 FCFA retenue Africage)  
  
Étape 10 - Évaluations :  
38. RatingModal apparaît pour Fatou : 5★ Kwame "Transport rapide et sécurisé"  
39. Kwame note Fatou 5★ "Expéditrice sérieuse, colis bien emballé"  
40. Notes mises à jour profils (Fatou 4.85★, Kwame 4.91★)  
  
Postconditions :  
• Transaction complète archivée  
• Historique accessible 30 jours  
• Confiance communautaire renforcée  
• Metrics business : nouveau GMV, retention users

# **8. Cas Limites & Gestion des Erreurs**

• Authentification :  
 ○ Code OTP expiré (5 min) → "Code expiré, demander nouveau code"  
 ○ Connexion avec compte non-vérifié KYC → Redirection forcée KYCScreen  
 ○ Trop de tentatives login (5/heure) → Blocage temporaire 1h  
  
• Création Annonce :  
 ○ Utilisateur non-vérifié → "Vérification identité requise pour publier"  
 ○ Poids >25kg → "Colis trop lourd pour transport P2P (max 25kg)"  
 ○ Distance >500km → "Distance trop importante pour service actuel"  
 ○ Upload image échoue → Retry 3x automatique + mode dégradé sans photo  
  
• Propositions :  
 ○ Colis déjà pris par autre voyageur → "Colis plus disponible"  
 ○ Auto-expiration propositions après 48h → Notification voyageur  
 ○ Voyageur propose prix négatif → Validation côté client et serveur  
  
• Paiements :  
 ○ Échec autorisation Paystack → "Vérifiez solde Mobile Money / carte"  
 ○ Timeout paiement (10 min) → Annulation automatique, notification expéditeur  
 ○ Remboursement si litige → Processus médiation admin 48h max  
  
• Géolocalisation :  
 ○ GPS désactivé → "Géolocalisation requise pour sécurité"  
 ○ Position incohérente livraison → Flag admin + vérification manuelle  
 ○ Perte signal pendant transport → Mode dégradé, dernière position connue  
  
• Communication :  
 ○ WebSocket déconnecté → Fallback polling 30s, reconnexion auto  
 ○ Message contenu inapproprié → Modération auto + warning utilisateur  
 ○ Spam protection → Max 20 messages/heure entre mêmes users  
  
• Cas Extrêmes :  
 ○ Voyageur ne confirme pas récupération 2h après horaire → Rappel push + SMS, annulation auto après 6h  
 ○ Voyageur disparaît en transit → Procédure urgence : contact téléphonique, géolocalisation forcée, escalade autorités si nécessaire  
 ○ Colis endommagé/volé → Processus assurance : photos avant/après, témoignages, remboursement selon grille  
 ○ Faux profils détectés → Suspension immédiate, vérification renforcée, blacklist dispositifs

================================================================================

IMPORTANT - Guide d'Implémentation :  
  
1. Ordre de développement recommandé :  
 ✓ Setup infrastructure (DB, AWS, Paystack sandbox)  
 ✓ Backend API core (auth, users, parcels) + tests unitaires  
 ✓ Mobile app MVP (inscription, création annonce, recherche)  
 ✓ Intégration paiements Paystack + KYC workflow  
 ✓ Fonctionnalités avancées (WebSocket chat, géolocalisation)  
 ✓ Dashboard admin + monitoring  
  
2. Phases de test :  
 ✓ Alpha interne équipe (2 semaines)  
 ✓ Beta fermée 50 early adopters Abidjan (1 mois)  
 ✓ Soft launch Côte d'Ivoire (3 mois)  
 ✓ Expansion régionale (6 mois)  
  
3. Métriques de succès Phase 1 :  
 ✓ 1000+ utilisateurs vérifiés  
 ✓ 100+ transactions complétées  
 ✓ NPS >60, rating app stores >4.0  
 ✓ Time to first transaction <24h  
  
Ce DSF sera mis à jour selon retours techniques et validations terrain.  
Contact équipe produit pour clarifications : product@africage.africa