# PLAN DE DÉVELOPPEMENT POUR UNE APPLI MOBILE DE TRANSFERT D'ARGENT

## 1. Définition des besoins fonctionnels

Pour cadrer le périmètre, chaque fonctionnalité doit être décrite précisément :

- Gestion des comptes utilisateurs
  - Création de compte avec vérification de l'identité (KYC)
  - Authentification (e-mail, mot de passe, OTP SMS)
  - Gestion des rôles (utilisateur, administrateur, super-admin)
- Portefeuille et transactions
  - Consultation du solde en temps réel
  - Historique détaillé (date, montant, statut, commission)
  - Dépôt de fonds (intégration Mobile Money)
  - Transfert interne (utilisateur → utilisateur)
  - Transfert vers l'administration (pour taxes, factures)
  - Retrait vers compte bancaire ou Mobile Money
- Sécurité et conformité
  - Chiffrement des données sensibles (TLS, AES-256)
  - Gestion des sessions (token JWT, rafraîchissement)
  - Limites et contrôles antifraude (seuils journaliers, blocage automatique)
  - Journalisation des actions critiques
- Notifications et alertes
  - Confirmation de transaction (push, SMS, e-mail)
  - Alertes de solde faible ou activité suspecte
  - Relances automatiques pour KYC incomplet
- Administration et reporting
  - Tableau de bord temps réel (volume de transactions, revenus, incidences)
  - Validation manuelle des comptes et transactions hors-norme
  - Export CSV/XLS des rapports financiers
  - Gestion des grilles tarifaires et commissions

## 2. Architecture technique

Cette section décrit la structure globale et la circulation des données :

- Front-end mobile
  - Flutter pour iOS et Android avec état géré par Provider ou Bloc
  - Interface modulaire (login, dashboard, transactions, profil)

- Back-end
    - Django + Django REST Framework exposant des endpoints REST
    - Authentification via JWT et rafraîchissement token
    - Services métiers découplés (micro-services ou apps Django séparées)
- Base de données
    - PostgreSQL pour la persistance ACID
    - Redis pour :
        - Mise en cache des requêtes fréquentes (solde, taux)
        - File d'attente pour traitement asynchrone (envoi SMS, e-mail)
- Intégrations externes
    - Passerelle Mobile Money (MTN, Orange) via API sécurisée
    - Service tiers pour l'envoi de SMS (Twilio ou API locale)
    - Service de vérification KYC (via OCR ou prestataire)
- Infrastructure et déploiement
    - Conteneurs Docker orchestrés par Docker Compose (dev) et Kubernetes (prod)
    - CI/CD avec GitHub Actions :
        - Linting et tests unitaires
        - Build automatique de l'image Docker
        - Déploiement sur un VPS ou sur AWS/GCP

# 3. Technologies et outils recommandés

Choix justifiés pour chaque couche :

- Langages et frameworks
    - Dart + Flutter pour une base de code unique mobile
    - Python 3.x, Django et Django REST Framework pour la robustesse
- Base de données et cache
    - PostgreSQL : transactions fiables, évolutivité
    - Redis : rapidité pour le cache et les tâches asynchrones
- Conteneurisation et CI/CD
    - Docker : isoler les environnements de développement et production
    - GitHub Actions : pipeline gratuit et intégré
- Suivi de projet et communication
    - Jira ou GitHub Projects pour gérer le backlog, les sprints et les tâches
    - Slack ou Microsoft Teams pour les échanges quotidiens
    - Confluence ou Notion pour la documentation interne
- Qualité et tests
    - pytest + coverage pour le back-end
    - Flutter Test + Mockito pour le front-end
    - SonarCloud pour l'analyse de code statique

# 4. Organisation et méthodologie

Structurer le travail pour maximiser la productivité :

- Équipe de 3 personnes :

    - Dev 1 : Flutter, UI/UX, tests front
    - Dev 2 : Django, API, intégrations externes
    - Dev 3 : DevOps, CI/CD, infra, support back/front
- Scrum léger (sprints de 2 semaines) :
    1. Sprint Planning (2 h) – définir les user stories et critères d'acceptation
    2. Daily Stand-up (15 min) – état d'avancement et blocages
    3. Sprint Review (1 h) – démonstration des fonctionnalités livrées
    4. Sprint Retrospective (1 h) – points forts, axes d'amélioration
- Backlog produit priorisé par valeur métier et complexité
- Definition of Done claire pour chaque user story
- Documentation vivante et partagée (README, API docs Swagger)

# 5. Roadmap initiale (sprints de 2 semaines)

| Sprint | Livrables clés |
|---|---|
| Sprint 0 | User stories détaillées, wireframes haute-fidélité, environnement dev conteneurisé |
| Sprint 1 | Structure monorepo, projet Flutter et projet Django initialisés, CI/CD pipeline basique |
| Sprint 2 | Authentification complète (inscription, login, mot de passe oublié, JWT) |
| Sprint 3 | Gestion des profils utilisateurs, KYC : upload de documents et statut de validation |
| Sprint 4 | Module portefeuille : affichage du solde, historique, dépôt via Mobile Money |
| Sprint 5 | Transfert interne et vers administration, calcul et application automatique des commissions |
| Sprint 6 | Intégration des passerelles de paiement externes, tests end-to-end sur flux de transaction |
| Sprint 7 | Interface administrateur (dashboard, validation manuelle), génération de rapports |
| Sprint 8 | Tests de charge, audit sécurité, corrections, préparation du MVP et déploiement en staging |