Spécifications Backend Django – Application QABY

1. Présentation du projet

Qaby est une application web et mobile de cagnottes collaboratives inspirée de Cotiz'up, adaptée aux besoins de l'Afrique de l'Ouest. Les utilisateurs peuvent créer, gérer et participer à des cagnottes (pour des mariages, anniversaires, aides sociales, etc.). Le backend est développé avec **Django + Django REST Framework**.

2. Schéma de données – Modèles principaux

Utilisateur (User)

- id (UUID) - nom - prénom - email (unique) - téléphone (unique) - mot_de_passe (haché via Django) - pays - photo_profil - date_inscription - vérifié (booléen) - rôle (user, admin)

Cagnotte (Fund)

- id (UUID) - créateur (ForeignKey → User) - titre - description - objectif_montant (Decimal) - montant_actuel (Decimal) - date_début / date_fin - statut (ouverte, clôturée, supprimée) - confidentialité (publique/privée) - image (optionnelle)

Contribution (Contribution)

- id (UUID) - cagnotte (ForeignKey → Fund) - contributeur (ForeignKey → User, nullable pour contribution anonyme) - montant (Decimal) - message (optionnel) - date_contribution - moyen_paiement (Mobile Money, carte, etc.) - transaction_id - statut (succès, échec, en attente)

Transaction

- id (UUID) - utilisateur - type (débit/crédit) - montant - référence - statut - date_transaction

Notification

- id - utilisateur - titre - message - lu (booléen) - date

Paramètres de sécurité

- gestion des tokens JWT - limitation de tentatives de connexion - vérification par email/SMS - 2FA (authentification à deux facteurs optionnelle)

3. Bonnes pratiques de sécurité Django

- **Utiliser `django-environ`** pour stocker les variables sensibles (SECRET_KEY, DB, etc.). - **Désactiver DEBUG en production.** - **Toujours utiliser HTTPS** (avec Let's Encrypt ou Cloudflare). - **Activer HSTS (HTTP Strict Transport Security).** - **Utiliser le hachage sécurisé de mot de passe (PBKDF2, Argon2).** - **Activer CSRF & XSS protection par défaut.** - **Limiter le nombre de requêtes par IP** (via django-ratelimit). - **Logger toutes les activités critiques (connexion, échecs, création cagnotte, etc.).** - **Utiliser des UUIDs comme identifiants.** - **Isoler la base de données dans un VPC (si sur cloud).** - **Mettre à jour Django et dépendances régulièrement.**

4. Architecture recommandée

- **Backend :** Django REST Framework - **Frontend Web :** React.js (consomme les endpoints REST) - **Mobile :** React Native (mêmes endpoints via API REST sécurisée JWT) - **Base de données :** PostgreSQL - **Stockage fichiers :** AWS S3 / DigitalOcean Spaces - **CI/CD :** GitHub Actions + Docker - **Monitoring :** Sentry + Grafana

5. Étapes suivantes

- Créer l’environnement virtuel et initialiser le projet Django - Configurer Django REST Framework - Mettre en place JWT Auth avec `djangorestframework-simplejwt` - Créer les modèles ci-dessus et les serializers - Définir les routes API (users, funds, contributions, transactions, notifications) - Ajouter des tests unitaires et de sécurité - Déployer sur un serveur sécurisé (ex: Railway, Render, AWS EC2)