# **Documentation de Ma Caisse Remix 4.0 (MCR4)**

## **1. Introduction**

### **1.1 Objectif**

Ma Caisse Remix 4.0 (MCR4) est une application web conçue pour gérer des opérations financières (dépôts, retraits, frais) et des utilisateurs (administrateurs, superviseurs, dealers, clients) dans un contexte de microfinance ou de services bancaires communautaires, probablement à Kinshasa, RDC. L'application fournit des tableaux de bord, des statistiques par commune, et une authentification sécurisée.

### **1.2 Contexte**

* **Domaine** : Microfinance, services de paiement mobile, ou banque communautaire.
* **Public cible** : Administrateurs, superviseurs, dealers, et clients (normaux et premium).
* **Localisation** : Kinshasa, RDC (références aux communes et devises FC/USD).

## **2. Processus Métier**

### **2.1 Rôles et Responsabilités**

* **Administrateurs** :
* Gestion des superviseurs (ajout, modification, activation/désactivation).
* Consultation des statistiques globales (superviseurs et dealers par commune).
* **Superviseurs** :
* Gestion des dealers et clients premium.
* Configuration des paramètres de caisse (taux d'échange, frais de retrait).
* Suivi des transactions et soldes par commune.
* **Dealers** :
* Exécution des transactions (dépôts, retraits) pour les clients.
* Opérations dans des communes spécifiques, sous supervision.
* **Clients (normaux et premium)** :
* Transactions via dealers (dépôts, retraits).
* Clients premium : Avantages (Pas de frais d’ouverture de compte et retrait immediat).

### **2.2 Flux Métier**

**Authentification** :

* Connexion avec email (domaines valides : @gmail.com, @yahoo.com, etc.) et mot de passe (5-10 caractères, lettres/chiffres/underscores).
* Redirection vers le tableau de bord selon le rôle.

**Gestion des utilisateurs** :

* Administrateurs : Ajout/modification de superviseurs avec codes uniques (ex. SUP123ABC).
* Superviseurs : Gestion des dealers/clients, configuration des paramètres.

**Transactions** :

* Dépôts/retraits en FC/USD, calcul des frais (retrait ≤ 5 %).
* Suivi des soldes et transactions par commune.

**Statistiques** :

* Tableaux de bord avec graphiques (donut pour répartition par commune, barres/lignes pour transactions/soldes).
* Recherche de dealers/clients par nom/code.

## **3. Architecture Technique**

### **3.1 Stack Technologique**

* **Backend** : Python 3, Flask (framework web).
* **Base de données** : SQL (gérée via SQLAlchemy).
* **Frontend** :
* Templates : Jinja2.
* CSS : Tailwind CSS.
* JavaScript : Chart.js (graphiques), scripts personnalisés (ex. togglePassword).
* **Authentification** : Flask-Login, WTForms (formulaires), Passlib (hachage des mots de passe).
* **Logging** : Logging Python vers app.log.

### **3.2 Structure des Fichiers**

mcr4/

├── app.py # Point d'entrée, configuration Flask

├── routes/

│ ├── admin.py # Routes administrateur (dashboard, gestion superviseurs)

│ ├── auth.py # Routes authentification (login, logout)

│ └── supervisor.py # Routes superviseur (dashboard, gestion dealers/clients)

├── forms/

│ ├── login\_form.py # Formulaire de connexion

│ └── supervisor\_form.py # Formulaire gestion superviseurs

├── models/

│ ├── user.py # Modèle User (email, role, commune, etc.)

│ └── database.py # Configuration SQLAlchemy

├── templates/

│ ├── auth/login.html # Page de connexion

│ ├── admin/

│ │ ├── dashboard.html # Tableau de bord administrateur

│ │ ├── add\_supervisor.html

│ │ ├── view\_supervisor.html

│ │ └── edit\_supervisor.html

│ └── base.html # Template de base

├── static/

│ ├── css/tailwind.css # Styles Tailwind

│ └── js/ # Scripts (ex. Chart.js)

└── utils/

└── decorators.py # Décorateurs (@admin\_required, @login\_required)

### **3.3 Modèle de Données**

* **Table User** :
* id : Clé primaire.
* first\_name, last\_name : Nom/prénom.
* email : Unique, domaine validé.
* username : Unique.
* password : Haché (pbkdf2\_sha256).
* role : Enum (ADMIN, SUPERVISOR, DEALER).
* commune : Lieu d'opération (ex. Gombe, Kinshasa).
* city : Ville (par défaut Kinshasa).
* code : Identifiant unique (ex. SUP123ABC).
* is\_active : Statut du compte.
* created\_by\_id : Référence au créateur (auto-référentiel).
* created\_at : Date de création.

## **4. Fonctionnalités**

### **4.1 Authentification**

* **Connexion** :
* Formulaire : Email (domaines : @gmail.com, @yahoo.com, @hotmail.com, @mcr4.com, @outlook.com, @aol.com, @icloud.com, @protonmail.com) et mot de passe (5-10 caractères, lettres/chiffres/underscores).
* Erreurs : Messages pour domaine non valide, caractères spéciaux, longueur incorrecte, ou compte inactif.
* Redirection : Tableau de bord selon rôle (/admin/dashboard, /supervisor/dashboard, /dealer/dashboard).
* **Déconnexion** : Redirection vers /login.

### **4.2 Gestion des Utilisateurs**

* **Administrateurs** :
* Ajout de superviseurs (/admin/supervisor/add) : Formulaire avec validation (email, username, téléphone uniques).
* Modification (/admin/supervisor/<id>/edit) : Mise à jour des informations.
* Activation/désactivation (/admin/supervisor/<id>/enable|disable).
* Consultation (/admin/supervisor/<id>) : Détails et dealers associés.
* **Superviseurs** :
* Gestion des dealers et clients premium.
* Configuration des paramètres (taux d'échange, frais de retrait ≤ 5 %).

### **4.3 Tableaux de Bord**

* **Administrateur (/admin/dashboard)** :
* Statistiques : Nombre total de superviseurs/dealers.
* Graphiques : Donut pour répartition par commune.
* Liste des superviseurs : Nom, username, code, date de création, actions (voir/modifier).
* **Superviseur** :
* Statistiques : Dealers actifs/inactifs, clients, transactions, soldes, frais.
* Graphiques : Donut (répartition dealers), barres (transactions), ligne (soldes).
* Recherche : Dealers/clients par nom/code.

### **4.4 Transactions**

* Dépôts/retraits en FC/USD.
* Calcul des frais (retrait, ouverture de compte).
* Suivi des soldes par devise et commune.

## **5. Installation**

### **5.1 Prérequis**

* Python 3.8+
* Base de données SQL (ex. SQLite, PostgreSQL)
* Dépendances Python :pip install flask flask-login flask-wtf sqlalchemy passlib

### **5.2 Configuration**

1. Cloner le dépôt :git clone https://github.com/MCR4-0/Ma\_Caisse\_Remix\_4.0\_Final.git
2. cd mcr4
3. Installer les dépendances :pip install -r requirements.txt
4. Configurer la base de données :

* Modifier models/database.py pour la connexion (ex. SQLALCHEMY\_DATABASE\_URI).
* Initialiser la base :from models.database import init\_db
* init\_db()

1. Compiler Tailwind CSS :npx tailwindcss -i ./static/css/input.css -o ./static/css/tailwind.css

### **5.3 Lancement**

python app.py

* Accéder à http://localhost:5000.

## **6. Utilisation**

### **6.1 Connexion**

1. Aller à /login.
2. Entrer un email (admin : admin@mcr4.com) et un mot de passe (mdp : Admin123).
3. Redirection vers le tableau de bord selon le rôle.

### **6.2 Tableau de Bord Administrateur**

* **URL** : /admin/dashboard
* **Actions** :
* Voir les statistiques (superviseurs/dealers par commune).
* Ajouter un superviseur (/admin/supervisor/add).
* Consulter/modifier un superviseur.

### **6.3 Gestion des Superviseurs**

* **Ajout** : Formulaire avec nom, email, téléphone, commune, etc.
* **Modification** : Mise à jour des informations, sauf mot de passe (optionnel).
* **Activation/Désactivation** : Contrôle de l'accès.

### **6.4 Dépannage**

* **Erreurs de connexion** :
* Vérifier le domaine email et la longueur du mot de passe.
* Consulter app.log pour les erreurs.
* **Graphiques vides** :
* Vérifier les données dans la base :from models.user import User
* from sqlalchemy import func
* print(db\_session.query(User.commune, func.count(User.id)).filter\_by(role=Role.SUPERVISOR).group\_by(User.commune).all())

## **7. Maintenance et Évolution**

### **7.1 Maintenance**

* **Logs** : Vérifier app.log pour les erreurs (connexion, base de données).
* **Sauvegardes** : Sauvegarder la base de données régulièrement.
* **Mises à jour** : Mettre à jour les dépendances (pip install --upgrade -r requirements.txt).

### **7.2 Évolution**

* **Nouveau backend** :

1. Isoler la logique métier dans business\_logic.py.
2. Migrer vers FastAPI (recommandé) ou Django pour compatibilité Jinja2.
3. Réutiliser les templates avec ajustements mineurs.
4. Traduire les modèles SQLAlchemy en ORM du nouveau framework (ex. Django ORM, Sequelize).

## **8. Annexes**

### **8.1 Exemple de Données**

* **Superviseur** :{
* "id": 1,
* "first\_name": "keren",
* "last\_name": "minimbu",
* "email": "kerto@mcr4.com",
* "username": "kerto",
* "role": "SUPERVISOR",
* "commune": "Matete",
* "code": "SUP123ABC",
* "is\_active": true
* }
* **Statistiques** :{
* "supervisor\_count": 5,
* "dealer\_count": 20,
* "supervisor\_locations": [["Gombe", 2], ["Kintambo", 3]],
* "dealer\_locations": [["Gombe", 10], ["Kintambo", 10]]
* }

### **8.2 Routes Principales**

* GET /login : Page de connexion.
* POST /login : Soumission du formulaire de connexion.
* GET /logout : Déconnexion.
* GET /admin/dashboard : Tableau de bord administrateur.
* GET|POST /admin/supervisor/add : Ajout d’un superviseur.
* GET /admin/supervisor/<id> : Détails d’un superviseur.
* GET|POST /admin/supervisor/<id>/edit : Modification d’un superviseur.
* POST /admin/supervisor/<id>/enable|disable : Activation/désactivation.

### **8.3 Validations**

* **Email** : Domaines valides (@gmail.com, @yahoo.com, etc.).
* **Mot de passe** : 5-10 caractères, lettres/chiffres/underscores.
* **Frais de retrait** : Maximum 5 %.

MAJ

Mettre en place l’usage des api pour se connecter a d’autres plateforme!

Mettre en place des tests unitaires (Sur des fonctionnalites critiques) (des api)