DOCUMENTATION TECHNIQUE

Système de Détection de Fraude Bancaire Automatisé

Projet:	DigitalBank Fraud Detection Pipeline
Date:	22/01/2026
Version:	1.0
Technologies:	Python, Flask, Supabase, Make.com, ngrok

1. VUE D'ENSEMBLE

Ce système automatise la détection de fraude bancaire en temps réel en combinant un modèle de machine learning avec une infrastructure cloud et une automatisation workflow. Chaque transaction est analysée automatiquement et une alerte est envoyée en cas de détection de fraude.

2. ARCHITECTURE DU SYSTÈME

Composant	Technologie	Rôle
Base de données	Supabase (PostgreSQL)	Stockage des transactions
Modèle IA	Python + scikit-learn	Classification fraude/légitime
API REST	Flask	Exposition du modèle
Tunnel Web	ngrok	Exposition API locale au web
Automatisation	Make.com	Orchestration du workflow
Notification	Gmail API	Alertes en temps réel

2.1 Flux de Données

1. **Supabase** → Récupération de la dernière transaction
2. **Make.com** → Extraction des features (amount, merchant_category, location, hour)
3. **API Flask** → Analyse via modèle ML et prédiction
4. **Router** → Décision basée sur is_fraud
5. **Gmail** → Envoi d'alerte si fraude détectée

3. COMPOSANTS TECHNIQUES

3.1 API Flask (fraud_api.py)

L'API Flask sert de pont entre Make.com et le modèle de machine learning. Elle expose deux endpoints :

Endpoint	Méthode	Description
/	GET	Health check + liste des catégories/localisations
/predict	POST	Analyse de transaction et prédiction

Dépendances Python :

• **Flask** : Framework web pour l'API REST
• **joblib** : Chargement du modèle sauvegardé (.pkl)
• **numpy** : Manipulation des arrays pour le modèle
• **scikit-learn** : Bibliothèque ML (RandomForest)

Format de Requête POST /predict :

```
{ "features": [ 7500, // amount (float) "Cryptocurrency", // merchant_category (string)
"Nigeria", // location (string) 3 // hour_of_day (int) ] }
```

Format de Réponse :

```
{ "is_fraud": true, "fraud_score": 0.94, "encoded_features": { "amount": 7500,
"merchant_code": 1, "location_code": 7, "hour_of_day": 3 } }
```

3.2 Encodage des Features

Le modèle nécessite des valeurs numériques. L'API encode automatiquement les catégories et localisations :

Catégories de Marchands :

Clothing: 0	Cryptocurrency: 1	Electronics: 2
Entertainment: 3	Food & Beverage: 4	Fuel: 5
Gambling: 6	Groceries: 7	Health: 8
Jewelry: 9	Travel: 10	Utilities: 11

Localisations :

France: 9	Dubai: 0	Hong Kong: 1
Las Vegas: 2	London: 3	Nigeria: 7
Online: 8	Russia: 10	

4. CONFIGURATION MAKE.COM

Module	Configuration	Rôle
Supabase Search Rows	Table: transactions Limit: 1 Sort: created_at DESC	Récupération dernière transaction
HTTP Make Request	Method: POST URL: ngrok Body: JSON features	Envoi vers API pour analyse
Router	Condition: is_fraud == true	Aiguillage décisionnel
Gmail Send Email	To: yacinediabi11@gmail.com Subject: Alerte Fraude	Notification en cas de fraude

5. GUIDE DE DÉPLOIEMENT

5.1 Prérequis

• Python 3.8+ avec environnement virtuel
• Compte Supabase avec base de données configurée
• Compte Make.com (plan gratuit suffisant)
• Compte Gmail pour les notifications
• ngrok installé et configuré

5.2 Étapes d'Installation

1. Environnement Python :

```
python -m venv venv
source venv/bin/activate
pip install flask joblib numpy scikit-learn
```

2. Lancer l'API Flask :

```
python fraud_api.py
```

3. Exposer avec ngrok :

```
ngrok http 5000
```

4. Configurer Make.com :
• Créer un nouveau scénario
• Ajouter les 4 modules (Supabase, HTTP, Router, Gmail)
• Copier l'URL ngrok dans le module HTTP
• Activer le scénario avec scheduling (ex: toutes les 5 min)

6. TESTS ET VALIDATION

Test API local (Postman) :
POST http://localhost:5000/predict
Body: {"features": [8500, "Cryptocurrency", "Nigeria", 3]}
Résultat attendu: is_fraud = true, fraud_score > 0.8

Test intégration (Make.com) :

```
INSERT INTO transactions (account_id, amount, merchant_category,
location, timestamp, transaction_id) VALUES (12, 8500,
'Cryptocurrency', 'Nigeria', Now(), '991');
```

Puis: Lancer "Run once" dans Make.com et vérifier l'email à yacinediabi11@gmail.com

7. MAINTENANCE ET MONITORING

Points de surveillance : Disponibilité de l'API Flask, Validité de l'URL ngrok (se renouvelle à chaque redémarrage), Connectivité Supabase, Quota Make.com (opérations/mois), Taux de faux positifs du modèle.

Logs : Flask enregistre chaque requête avec les features encodées et le résultat de prédiction. Make.com conserve l'historique d'exécution de chaque scénario.