Conception Technique

1. Introduction

1.1 Contexte du projet

Ce document décrit la conception technique de la plateforme de gestion et de monitoring des données bancaires pour **DigitalBank**.
La solution vise à centraliser les données clients et transactions, détecter les activités frauduleuses, sécuriser les accès via un contrôle fin des permissions et fournir des tableaux de bord adaptés aux différents profils utilisateurs.

1.2 Objectifs techniques

- Mettre en place une architecture sécurisée, scalable et maintenable
- Exploiter des outils no-code / low-code pour accélérer le développement
- Garantir la confidentialité, l'intégrité et la traçabilité des données
- Fournir une visualisation temps réel et des alertes automatisées

2. Vue d'ensemble de l'architecture

2.1 Description globale

L'architecture repose sur une base de données PostgreSQL centrale, exposée via des couches API sécurisées, consommées par des dashboards métier et des workflows automatisés.

Les principaux composants sont :

- Base de données PostgreSQL (DigitalBank DB)
- API layer (Supabase et Hasura)
- Plateforme de visualisation (Metabase)
- Automatisation des workflows (Make.com)
- Système d'alertes et reporting
- Mécanismes de sécurité et d'audit

3. Composants techniques

3.1 Base de données – PostgreSQL (DigitalBank DB)

La base de données centralise l'ensemble des données bancaires :

Tables principales :

- customers
- accounts
- cards
- transactions
- audit_logs
- login_attempts

Cette structuration permet :

- Une séparation claire entre données métier et données de sécurité
- Une traçabilité complète des actions sensibles
- Une exploitation efficace pour l'analytique et la détection de fraude

3.2 Couche API

3.2.1 Supabase

Supabase est utilisé comme backend principal :

- PostgreSQL managé
- Authentification intégrée
- API REST auto-générée
- Realtime pour les événements (transactions, alertes)

Fonctionnalités clés :

- Gestion des utilisateurs
- Authentification par JWT
- Mise en place du Row Level Security (RLS)

3.2.2 Hasura

Hasura fournit une API GraphQL auto-générée au-dessus de PostgreSQL.

Rôles principaux :

- Exposition flexible des données pour les dashboards
- Gestion avancée des permissions par rôle
- Optimisation des requêtes complexes (clients → comptes → transactions)

4. Frontend et visualisation

4.1 Dashboards analytiques

Metabase est utilisé pour :

- Dashboard Analyste Sécurité
- Suivi des transactions et fraudes
- Indicateurs clés (KPI)

Fonctionnalités :

- Filtres temporels
- Visualisation agrégée
- Accès contrôlé selon les rôles

4.2 Plateforme utilisateurs

Les utilisateurs accèdent à la plateforme selon leur profil :

- Administrateur système
- Analyste sécurité
- Agent service client
- Client (consultation limitée)

Chaque rôle dispose d’une vue adaptée à ses besoins métier.

5. Automatisation et alertes

5.1 Workflows automatisés

Make.com est utilisé pour :

- Détection automatique de fraude
- Alertes sur transactions élevées
- Tentatives de connexion suspectes
- Génération de rapports quotidiens (PDF / Email)

5.2 Système d’alertes

Les alertes sont envoyées via :

- Email
- Microsoft Teams
- Slack

6. Architecture de sécurité

6.1 Authentification

- Authentification basée sur JWT
- Gestion centralisée via Supabase Auth
- Support du MFA (authentification multi-facteurs)

6.2 Contrôle d'accès (RBAC)

Le contrôle d'accès repose sur :

- Des rôles applicatifs (admin, analyst, service client, client)
- Des permissions définies au niveau API
- Des règles RLS au niveau base de données

6.3 Chiffrement

- Données chiffrées en transit (HTTPS / TLS)
- Sécurité native fournie par Supabase pour le stockage

6.4 Audit et traçabilité

- Table audit_logs pour tracer les actions sensibles
- Enregistrement de :
    - l'utilisateur
    - l'action
    - la table impactée
    - l'adresse IP
    - l'horodatage

7. Monitoring et supervision

La plateforme inclut un suivi :

- Des accès utilisateurs
- Des tentatives de connexion échouées
- Des événements de sécurité

Les données sont exploitées dans les dashboards de monitoring.

8. Conclusion

Cette architecture technique répond aux exigences fonctionnelles et sécuritaires du projet DigitalBank tout en respectant les contraintes de temps et l'approche no-code / low-code. Elle est évolutive et peut être enrichie ultérieurement (SIEM avancé, ML temps réel, haute disponibilité).