Architecture de la solution

a) Schéma d'architecture technique

Description détaillée des composants

1. **Frontend (Interface utilisateur)**
    a. Outils possibles : **Retool, Metabase, Grafana, Appsmith**
    b. Rôles et dashboards :
        i. **Admin** : vue complète des tables, gestion des utilisateurs, monitoring système
        ii. **Analyst** : dashboard de détection de fraude (transactions suspectes, score de fraude)
        iii. **Customer Service** : accès aux informations clients et transactions limitées
        iv. **Client** : consultation de ses comptes et transactions
    c. Fonctionnalités :
        i. Filtres par période, type de compte, catégorie de marchand
        ii. Visualisation en temps réel grâce à l'API Supabase
2. **API Layer**
    a. Outils possibles : **Supabase (REST auto-générée), Hasura (GraphQL), Xano**
    b. Rôles :
        i. Gère l'accès aux données via **Row Level Security (RLS)** ou permissions Hasura
    c. Fonctionnalités :

i. Fournir endpoints sécurisés pour le frontend

ii. Authentification JWT + MFA

iii. Realtime pour notifications et alertes

iv. Conversion des requêtes frontend en requêtes SQL ou GraphQL

3. **Base de données PostgreSQL**

   a. Tables principales : customers, accounts, cards, transactions, login_attempts, audit_logs

   b. Sécurité :

      i. **RLS** pour contrôler l'accès par rôle

      ii. Chiffrement des données sensibles (pgcrypto pour cartes, CVV)

      iii. Audit logs pour toutes les actions critiques

   c. Index pour optimiser les requêtes fréquentes :

      i. transactions(account_id)

      ii. transactions(timestamp)

      iii. login_attempts(ip_address)

4. **Automatisation / Workflow**

   a. Outils : **n8n, Make.com**

5. **Monitoring & Alerting**

   a. Outils : **Grafana, ELK Stack, Prometheus**

   b. Metrics :

i. Performance système : CPU, RAM, Disk, Network

ii. Base de données : nombre de connexions, latence

iii. API : taux d'erreur, temps de réponse

c. Alertes :

i. Transaction suspecte détectée

ii. Taux d'erreur > seuil

iii. Utilisateur multiple tentatives de login échouées

b) Modèle de données (ERD)

Tables principales et relations

Table	Description	Relations
customers	Informations clients	1:N → accounts
accounts	Comptes bancaires	1:N → transactions ; 1:N → cards
transactions	Historique des opérations	
cards	Cartes bancaires liées aux comptes	
login_attempts	Tentatives de connexion	
audit_logs	Historique des actions critiques	

Exemple de diagramme ERD :

- customers.customer_id → accounts.customer_id
- accounts.account_id → transactions.account_id
- accounts.account_id → cards.account_id

c) Architecture de sécurité détaillée

- **Authentification**

  - Méthode : **Email / mot de passe + MFA (TOTP)**
  - JWT émis par Supabase Auth
  - Expiration de session et contrôle AAL1 (Enhanced MFA Security)

- **Contrôle d'accès (RBAC)**

  - Rôles :
    - admin : accès complet
    - analyst : lecture transactions
    - customer_service : lecture clients + modification limitée
    - customer : lecture/édition de ses propres données
  - Implémenté via **RLS** ou permissions Hasura

- **Chiffrement**

  - HTTPS pour toutes les communications
  - Hash bcrypt pour les mots de passe
  - Cartes et CVV chiffrés via pgcrypto

- **Audit et traçabilité**

  - Table audit_logs
  - Triggers PostgreSQL pour INSERT/UPDATE/DELETE sur comptes et SELECT sur clients
  - Historique exploitable dans dashboards

d) Choix technologiques et justification

Couches	Outil choisi	Justification	Alternatives
Base + Auth	Supabase	PostgreSQL managé + API REST auto-générée + RLS intégré	Hasura, Xano
Dashboard Analyst	Grafana	Visualisation rapide + filtres + cartes	Retool, Metabase
Automatisation	n8n	Open-source, workflows faciles à créer	Make.com, Zapier
Monitoring	Grafana	Supervision temps réel, alertes	ELK Stack, Prometheus