DOCUMENT DE CONCEPTION TECHNIQUE

Plateforme de Monitoring et Sécurité - DigitalBank France

Projet : TRIO INFERNAL
Formation : Master ESIC / CPDIA
Date : Janvier 2026
Version : 1.0

TABLE DES MATIÈRES

1. INTRODUCTION ET CONTEXTE

1.1 Présentation de DigitalBank France

DigitalBank France est une institution bancaire en ligne opérant sur le marché français dans un contexte hautement réglementé. L'établissement propose des services bancaires digitaux incluant la gestion de comptes courants, l'épargne, les

paiements par carte et les virements. Dans un environnement où la confiance des clients constitue le socle de la relation bancaire, la sécurité des systèmes d'information et la transparence opérationnelle représentent des impératifs stratégiques majeurs.

1.2 Contexte de la Cyberattaque

L'établissement a récemment subi une cyberattaque de type ransomware qui a compromis une partie de son infrastructure informatique. Cette intrusion a révélé des vulnérabilités critiques dans les mécanismes de surveillance et de contrôle d'accès aux systèmes sensibles. L'incident a entraîné une interruption temporaire de certains services, une perte de confiance partielle de la clientèle, et a mis en lumière l'urgence d'un renforcement substantiel des dispositifs de monitoring, d'audit et de sécurité.

Face à cette situation critique, la direction générale a mandaté le groupe TRIO INFERNAL pour concevoir et déployer une plateforme intégrée de monitoring et de sécurité capable de détecter, prévenir et réagir efficacement aux menaces futures.

1.3 Nécessité d'une Plateforme de Monitoring et Sécurité

La mise en place d'une plateforme de monitoring et de sécurité s'inscrit dans une démarche globale de résilience organisationnelle et de conformité réglementaire. Les directives européennes DSP2 (Directive sur les Services de Paiement) et le RGPD (Règlement Général sur la Protection des Données) imposent des exigences strictes en matière de traçabilité, de protection des données personnelles et de notification des incidents de sécurité.

La plateforme doit permettre : - Une surveillance continue des transactions bancaires avec détection automatisée des anomalies - Un système d'audit complet traçant l'ensemble des actions administratives et opérationnelles - Des tableaux de bord décisionnels pour les équipes métier, sécurité et direction - Une gestion fine des droits d'accès selon le principe du moindre privilège - Une capacité de réaction rapide en cas de détection d'incident

1.4 Objectifs du Projet

Le projet vise à concevoir une architecture technique robuste, évolutive et sécurisée, intégrant des technologies modernes de type no-code/low-code pour accélérer le déploiement tout en maintenant un niveau élevé de personnalisation et

de contrôle. L'objectif est triple : restaurer la confiance des parties prenantes, se conformer aux exigences réglementaires et établir une posture de sécurité proactive face aux menaces cyber émergentes.

---

2. OBJECTIFS TECHNIQUES

La plateforme de monitoring et de sécurité de DigitalBank France doit répondre à cinq objectifs techniques principaux :

2.1 Monitoring en Temps Réel des Transactions

Mise en place d'un système de surveillance continue des flux transactionnels permettant l'identification immédiate des comportements anormaux (montants inhabituels, géolocalisation incohérente, fréquence suspecte). Le système doit traiter plusieurs milliers de transactions par minute avec une latence inférieure à 500 millisecondes.

2.2 Détection Automatisée de Fraude

Déploiement d'un moteur d'intelligence artificielle basé sur des algorithmes de Machine Learning (classification supervisée, détection d'anomalies) capable d'attribuer un score de risque à chaque transaction. Le système doit apprendre continuellement des nouveaux patterns de fraude et réduire le taux de faux positifs en dessous de 2%.

2.3 Tableaux de Bord Décisionnels

Création de dashboards interactifs et personnalisés selon les profils utilisateurs (administrateur, analyste sécurité, service client, direction). Les visualisations doivent présenter des indicateurs clés de performance (KPI) tels que le nombre de transactions par heure, le taux de fraude, les tentatives d'accès non autorisées, et les métriques de disponibilité des services.

2.4 Gestion des Accès et Audit

Implémentation d'un système de gestion des identités et des accès (IAM) basé sur le contrôle d'accès par rôle (RBAC) et complété par une sécurité au niveau des lignes de données (RLS). Toute action sensible doit être journalisée dans un système d'audit inaltérable permettant une traçabilité complète pour les investigations post-incident et les audits de conformité.

2.5 Conformité RGPD et Réglementations Bancaires

Respect strict du RGPD en matière de protection des données personnelles : minimisation des données collectées, pseudonymisation/anonymisation des données sensibles, mise en œuvre du droit à l'oubli, notification des violations de données dans les 72 heures. Conformité aux recommandations de l'ACPR (Autorité de Contrôle Prudentiel et de Résolution) et aux normes PCI-DSS (Payment Card Industry Data Security Standard) pour le traitement des données de cartes bancaires.

---

3. ARCHITECTURE GLOBALE

3.1 Vue d'Ensemble de l'Architecture

L'architecture de la plateforme repose sur un modèle multi-couches (multi-tier architecture) garantissant une séparation claire des responsabilités et facilitant la scalabilité horizontale. Ce paradigme architectural distingue quatre couches principales :

1. **Couche Présentation** : Interfaces utilisateurs (dashboards Metabase, Grafana) et applications clientes
2. **Couche Métier** : APIs REST et GraphQL (Supabase, Flask) exposant les fonctionnalités métier
3. **Couche Données** : Base de données PostgreSQL centralisée
4. **Couche Sécurité et Transverse** : Authentification, autorisation, audit, automatisation

Cette séparation permet une évolution indépendante de chaque couche, facilite les tests unitaires et d'intégration, et optimise la résilience globale du système.

3.2 Approche Multi-Couches

3.2.1 Couche Présentation

La couche présentation se compose de deux types de dashboards spécialisés : - **Metabase** : orienté analyse métier, destiné aux équipes opérationnelles et à la direction pour le suivi des indicateurs business (volume de transactions, taux de conversion, satisfaction client) - **Grafana** : orienté monitoring technique, destiné aux équipes IT et sécurité pour le suivi des métriques système (latence, taux d'erreur API, charge serveur, alertes sécurité)

3.2.2 Couche Métier

Deux APIs complémentaires assurent l'interface entre la présentation et les données : - **Supabase Auto-API** : génération automatique d'endpoints REST et GraphQL à partir du schéma PostgreSQL, réduisant drastiquement le temps de développement - **API Flask** : microservice Python dédié aux traitements de Machine Learning (scoring de fraude, analyse comportementale)

3.2.3 Couche Données

PostgreSQL constitue le système de gestion de base de données relationnel central. Le choix d'un SGBDR s'impose dans le contexte bancaire pour garantir les propriétés ACID (Atomicité, Cohérence, Isolation, Durabilité) indispensables à la fiabilité des transactions financières.

3.2.4 Couche Transverse

- **Supabase Auth** : gestion centralisée de l'authentification avec support multi-facteurs (MFA)
- **Make.com / n8n** : orchestration de workflows automatisés (notifications, rapports, escalades)
- **Système d'audit** : journalisation exhaustive des événements de sécurité

3.3 Schéma Conceptuel

Le schéma d'architecture technique illustre les interactions entre les composants :

```
[Utilisateurs] → [Authentification MFA] → [APIs] → [Base PostgreSQL]
     ↓                                        ↓
[Dashboards]                          [Audit Logs]
```

```
      ↓                                                    ↓
[Automatisation] ← [Alertes] ← [Détection ML]
```

Les utilisateurs s'authentifient via Supabase Auth avec double facteur (mot de passe + code OTP ou biométrie). Une fois authentifiés, ils accèdent aux dashboards qui interrogent les APIs. Les APIs appliquent les règles de sécurité (RBAC, RLS) avant d'accéder aux données PostgreSQL. Toute action est enregistrée dans les logs d'audit. L'API Flask analyse continuellement les transactions et alimente un système d'alertes qui déclenche des workflows automatisés via Make.com.

3.4 Principes Architecturaux

3.4.1 Scalabilité

L'architecture est conçue pour une montée en charge horizontale. Les APIs peuvent être déployées en mode cluster avec load balancing. PostgreSQL supporte la réplication en lecture (read replicas) pour répartir la charge des requêtes analytiques.

3.4.2 Sécurité (Security by Design)

La sécurité est intégrée dès la conception : chiffrement des données en transit (TLS 1.3) et au repos (AES-256), authentification forte, principe du moindre privilège, défense en profondeur avec plusieurs couches de contrôle.

3.4.3 Résilience

Le système intègre des mécanismes de tolérance aux pannes : sauvegardes automatiques de la base de données, high availability pour les composants critiques, stratégie de reprise après sinistre (Disaster Recovery Plan) avec RPO (Recovery Point Objective) de 15 minutes et RTO (Recovery Time Objective) de 1 heure.

---

4. DESCRIPTION DES COMPOSANTS

4.1 Base de Données PostgreSQL

4.1.1 Choix de PostgreSQL

PostgreSQL est un système de gestion de base de données relationnel open-source reconnu pour sa robustesse, sa conformité aux standards SQL, et sa richesse fonctionnelle. Il offre un support natif du format JSON (type JSONB) permettant une flexibilité dans le stockage de données semi-structurées tout en conservant les garanties transactionnelles du modèle relationnel.

4.1.2 Structure des Données

Le schéma de données comprend sept tables principales :

- **customers** : informations clients (identité, coordonnées, statut KYC)
- **accounts** : comptes bancaires liés aux clients (numéro de compte, type, solde)
- **transactions** : historique des opérations bancaires (virements, prélèvements, retraits)
- **cards** : cartes bancaires associées aux comptes (numéro, date d'expiration, plafonds)
- **users** : utilisateurs de la plateforme d'administration (administrateurs, analystes)
- **roles** : rôles et permissions associés aux utilisateurs
- **audit_logs** : journalisation de toutes les actions effectuées sur la plateforme

4.1.3 Mécanismes de Sauvegarde et Restauration

Trois niveaux de sauvegarde sont mis en œuvre : - **Sauvegardes complètes quotidiennes** : dump complet de la base via pg_dump - **Sauvegardes incrémentales horaires** : archivage des Write-Ahead Logs (WAL) permettant une restauration à un instant précis (Point-in-Time Recovery) - **Réplication en temps réel** : serveur PostgreSQL standby synchrone pour basculement automatique en cas de panne du serveur principal

4.1.4 Performance et Indexation

Des index B-tree sont créés sur les colonnes fréquemment utilisées dans les clauses WHERE et JOIN (customer_id, account_id, transaction_timestamp). Des index GIN (Generalized Inverted Index) sont utilisés pour les recherches full-text et les requêtes sur les colonnes JSONB. Le query planner de PostgreSQL est régulièrement analysé (EXPLAIN ANALYZE) pour optimiser les plans d'exécution des requêtes lourdes.

4.2 Couche API

4.2.1 Supabase Auto-API

Supabase génère automatiquement une API REST et GraphQL à partir du schéma PostgreSQL. Chaque table devient accessible via des endpoints CRUD (Create, Read, Update, Delete) sans écriture de code backend. Cette approche "API-first" accélère considérablement le développement et garantit une cohérence entre le modèle de données et l'interface applicative.

Les avantages incluent : - Génération automatique de la documentation OpenAPI - Support natif de la pagination, du tri et du filtrage - Gestion optimisée des relations entre tables (foreign keys) - Mise à jour en temps réel via WebSockets (Realtime subscriptions)

4.2.2 API Flask pour Machine Learning

Un microservice Python Flask dédié expose des endpoints spécialisés pour la détection de fraude. Ce service reçoit les données de transactions, applique des modèles de Machine Learning pré-entraînés (Random Forest, Isolation Forest pour la détection d'anomalies), et retourne un score de risque entre 0 et 1.

Endpoints principaux : - `POST /api/fraud/predict` : évalue une transaction en temps réel - `POST /api/fraud/batch` : traite un lot de transactions historiques - `GET /api/fraud/model/metrics` : retourne les métriques de performance du modèle (précision, rappel, F1-score)

4.2.3 Gestion des Requêtes et Cache

Pour optimiser les performances, une couche de cache Redis est intégrée en frontal des APIs. Les requêtes fréquentes (lecture de données statiques, référentiels) sont mises en cache avec une durée de vie (TTL) configurée. Les requêtes d'écriture invalident automatiquement les entrées de cache correspondantes pour garantir la cohérence.

4.3 Sécurité

4.3.1 Authentification (Supabase Auth + MFA)

Supabase Auth gère l'ensemble du cycle de vie de l'authentification : - Inscription et connexion par email/mot de passe avec confirmation email - Support OAuth 2.0 pour connexion via fournisseurs tiers (Google, Microsoft) - Authentification multi-facteurs (MFA) basée sur TOTP (Time-based One-Time Password) selon la norme RFC 6238 - Gestion des sessions avec JWT (JSON Web Tokens) incluant les claims d'identité et de rôle

Le token JWT est signé avec une clé secrète côté serveur et vérifié à chaque requête API. La durée de vie du token est limitée à 1 heure, nécessitant un rafraîchissement via un refresh token stocké de manière sécurisée.

4.3.2 RBAC (Role-Based Access Control)

Le contrôle d'accès par rôle est implémenté via la table `roles` qui définit quatre rôles principaux : - **Administrateur** : accès complet à la plateforme, gestion des utilisateurs et des configurations - **Analyste Sécurité** : accès aux logs d'audit, dashboards de monitoring, investigation des incidents - **Service Client** : consultation des données clients et transactions, assistance clientèle - **Client** : accès limité à ses propres données via l'application mobile/web

Chaque rôle est associé à un ensemble de permissions stockées au format JSONB, permettant une granularité fine (lecture, écriture, suppression par ressource).

4.3.3 RLS (Row Level Security)

La sécurité au niveau des lignes (RLS) est une fonctionnalité native de PostgreSQL permettant de filtrer automatiquement les lignes retournées en fonction du contexte de l'utilisateur connecté.

Exemple de politique RLS sur la table `transactions` :

```
CREATE POLICY client_transactions ON transactions
FOR SELECT
USING (account_id IN (
  SELECT account_id FROM accounts
  WHERE customer_id = current_user_id()
));
```

Cette politique garantit qu'un client ne peut consulter que les transactions de ses propres comptes, même si l'application présente une vulnérabilité permettant de forger des requêtes SQL arbitraires.

4.3.4 Chiffrement des Données

Deux niveaux de chiffrement sont appliqués : - **Chiffrement en transit** : TLS 1.3 pour toutes les communications entre clients et serveurs - **Chiffrement au repos** : chiffrement AES-256 des volumes de stockage PostgreSQL via LUKS (Linux Unified Key Setup)

Les données sensibles (numéros de carte, mots de passe) sont en outre hachées avec des algorithmes résistants (Argon2 pour les mots de passe, tokenisation pour les numéros de carte).

4.3.5 Journalisation d'Audit

La table `audit_logs` enregistre exhaustivement : - L'identifiant de l'utilisateur ayant effectué l'action - Le type d'action (lecture, écriture, suppression) - La table et l'enregistrement concernés - L'horodatage précis (timestamp avec timezone) - L'adresse IP source et le user-agent

Ces logs sont immuables (insertion seule, pas de mise à jour ni suppression) et archivés dans un système externe (SIEM) pour analyse forensique.

4.4 Dashboards

4.4.1 Metabase pour l'Analyse Métier

Metabase est un outil open-source de Business Intelligence permettant de créer des dashboards interactifs sans compétences en programmation. Les équipes métier peuvent construire des requêtes SQL via une interface graphique intuitive et visualiser les résultats sous forme de graphiques, tableaux et cartes.

Dashboards clés : - **Vue Executive** : KPI de haut niveau (nombre de clients actifs, volume de transactions mensuel, taux de croissance) - **Analyse des Fraudes** : nombre de transactions suspectes, répartition géographique des fraudes, top 10 des montants frauduleux - **Satisfaction Client** : temps de réponse du service client, taux de résolution au premier contact

4.4.2 Grafana pour le Monitoring Technique

Grafana est spécialisé dans la visualisation de métriques techniques et le monitoring en temps réel. Il s'interface avec PostgreSQL et des systèmes de collecte de métriques (Prometheus) pour afficher : - Métriques applicatives : taux de requêtes par seconde (RPS), latence p95/p99, taux d'erreur 5xx - Métriques infrastructure : utilisation CPU/RAM des serveurs, IOPS disque, bande passante réseau - Alertes configurées : notifications Slack/email en cas de dépassement de seuils

4.4.3 Visualisations Clés

Les dashboards intègrent divers types de visualisations : - **Time series** : évolution du nombre de transactions dans le temps - **Heatmaps** : distribution des transactions par heure et jour de la semaine - **Geo-maps** : localisation géographique des transactions - **Gauges** : indicateurs de santé système (disponibilité, taux d'erreur)

4.4.4 Alertes et Notifications

Des règles d'alerting sont configurées pour déclencher des notifications automatiques : - Détection de pics anormaux de transactions (écart > 3 sigmas par rapport à la moyenne mobile) - Taux de fraude supérieur à 5% sur une fenêtre glissante de 1 heure - Indisponibilité d'un composant critique pendant plus de 5 minutes

4.5 Automatisation

4.5.1 Make.com ou n8n

Ces plateformes d'automatisation no-code permettent de créer des workflows visuels sans écrire de code. Les cas d'usage incluent : - **Alertes fraude** : lorsqu'une transaction avec un score de risque > 0.8 est détectée, un workflow envoie automatiquement un SMS au client et un email à l'analyste sécurité - **Rapports automatisés** : génération quotidienne d'un rapport PDF récapitulant les KPI de la veille, envoyé par email à la direction - **Synchronisation de données** : export régulier de données agrégées vers un data warehouse pour analyses approfondies

4.5.2 Workflows Automatisés

Exemple de workflow de gestion d'incident : 1. Détection d'une tentative de connexion suspecte (géolocalisation inhabituelle) 2. Création automatique d'un ticket dans le système de gestion des incidents 3. Notification de l'équipe sécurité

via Slack 4. Si aucune action n'est prise sous 15 minutes, escalade au responsable sécurité 5. Blocage automatique du compte en cas de confirmation de compromission

4.5.3 Intégration avec les APIs

Make.com et n8n s'intègrent nativement avec les APIs REST via webhooks. Les APIs Supabase et Flask peuvent déclencher des workflows en envoyant des requêtes HTTP POST vers les endpoints webhook fournis par les plateformes d'automatisation. Les workflows peuvent à leur tour appeler les APIs pour récupérer des données complémentaires ou effectuer des actions (création d'enregistrement, mise à jour de statut).

---

5. ARCHITECTURE DE SÉCURITÉ

5.1 Modèle Zero Trust

L'architecture de sécurité adopte le paradigme Zero Trust (« ne jamais faire confiance, toujours vérifier »), qui part du principe qu'aucune entité (utilisateur, application, réseau) n'est intrinsèquement fiable. Chaque accès doit être authentifié, autorisé et validé continuellement.

Principes appliqués : - **Vérification explicite** : authentification systématique avec MFA - **Accès au moindre privilège** : chaque utilisateur et service dispose uniquement des permissions strictement nécessaires - **Assumer la compromission** : le système doit limiter l'impact d'une compromission en segmentant les accès et en surveillant les comportements anormaux

5.2 Gestion des Identités et des Accès (IAM)

Le système IAM repose sur trois piliers : 1. **Authentification** : vérification de l'identité (qui êtes-vous ?) 2. **Autorisation** : vérification des permissions (que pouvez-vous faire ?) 3. **Audit** : enregistrement des actions (qu'avez-vous fait ?)

Les identités sont centralisées dans Supabase Auth, qui fait office de fournisseur d'identité (IdP). Les applications et APIs délèguent l'authentification à cet IdP et vérifient les tokens JWT pour autoriser les actions.

5.3 Stratégie de Défense en Profondeur

La défense en profondeur consiste à superposer plusieurs couches de sécurité de manière à ce que la compromission d'une couche ne suffise pas à compromettre l'ensemble du système.

Couches de défense : 1. **Périmètre réseau** : firewall, WAF (Web Application Firewall) bloquant les attaques courantes (injection SQL, XSS) 2. **Application** : validation stricte des entrées utilisateurs, échappement des sorties, protection CSRF 3. **API** : authentification JWT, rate limiting pour prévenir les abus 4. **Base de données** : RLS, chiffrement, sauvegardes régulières 5. **Surveillance** : détection d'intrusion (IDS), analyse des logs, alertes temps réel

5.4 Protection contre les Menaces

5.4.1 DDoS (Distributed Denial of Service)

Protection via services de CDN/DDoS mitigation (Cloudflare, AWS Shield) qui filtrent le trafic malveillant avant qu'il n'atteigne les serveurs applicatifs.

5.4.2 Injection SQL

Utilisation exclusive de requêtes préparées (prepared statements) et d'ORM (Object-Relational Mapping) pour éviter la construction de requêtes SQL dynamiques par concaténation de chaînes.

5.4.3 XSS (Cross-Site Scripting)

Échappement systématique des données utilisateurs avant affichage dans les interfaces web. Implémentation de Content Security Policy (CSP) pour restreindre l'exécution de scripts non autorisés.

5.4.4 Élévation de Privilèges

Validation côté serveur des autorisations pour chaque action, indépendamment des contrôles côté client. Les permissions sont vérifiées à partir du token JWT et du RBAC/RLS.

5.5 Conformité RGPD

5.5.1 Principes de Protection des Données

- **Minimisation** : seules les données strictement nécessaires sont collectées

- **Limitation de la finalité** : les données ne sont utilisées que pour les finalités déclarées
- **Exactitude** : mécanismes de correction et de mise à jour des données personnelles
- **Limitation de la conservation** : suppression automatique des données après expiration de la durée légale de conservation

5.5.2 Droits des Personnes

- **Droit d'accès** : possibilité pour le client de télécharger l'ensemble de ses données
- **Droit de rectification** : modification des données inexactes
- **Droit à l'effacement** : suppression des données (sauf obligations légales de conservation pour les données bancaires)
- **Droit à la portabilité** : export des données dans un format structuré

5.5.3 Notification des Violations

En cas de violation de données personnelles, la plateforme intègre un workflow automatisé de notification à la CNIL (Commission Nationale de l'Informatique et des Libertés) dans le délai réglementaire de 72 heures.

---

6. FLUX DE DONNÉES

6.1 Flux d'Authentification

1. L'utilisateur saisit ses identifiants (email + mot de passe) dans l'interface de connexion
2. La requête est transmise à Supabase Auth via HTTPS
3. Supabase Auth vérifie les identifiants dans la table `users`
4. Si correct, génération d'un code OTP envoyé par SMS/email (MFA)
5. L'utilisateur saisit le code OTP
6. Supabase Auth valide le code et génère un JWT contenant l'identifiant utilisateur et son rôle
7. Le JWT est retourné au client et stocké en mémoire sécurisée (httpOnly cookie ou localStorage avec précautions)
8. Toutes les requêtes ultérieures incluent le JWT dans l'en-tête Authorization

6.2 Flux de Transaction Bancaire

1. Le client initie une transaction (virement, paiement carte) via l'application mobile
2. L'application envoie une requête POST à l'API Supabase avec le JWT
3. Supabase vérifie le JWT, extrait l'identifiant client et le rôle
4. Les politiques RLS de PostgreSQL filtrent automatiquement l'accès aux données autorisées
5. L'API insère un enregistrement dans la table `transactions`
6. Un trigger PostgreSQL déclenche un appel à l'API Flask pour scoring de fraude
7. L'API Flask retourne un score de risque
8. Si score > 0.8, un webhook est envoyé à Make.com pour alerting
9. La transaction est marquée "en attente de validation" ou "validée" selon le score
10. Une confirmation est retournée au client

6.3 Flux de Détection de Fraude

1. L'API Flask reçoit les données de transaction (montant, merchant, géolocalisation, heure)
2. Extraction de features : montant normalisé, distance par rapport aux transactions précédentes, heure inhabituelle
3. Application du modèle de Machine Learning (Random Forest pré-entraîné)
4. Calcul d'un score de probabilité de fraude entre 0 et 1
5. Retour du score à l'API appelante
6. Si score > seuil configuré, déclenchement d'une alerte
7. L'analyste sécurité reçoit une notification et peut confirmer/infirmer la fraude
8. Le feedback (vrai positif / faux positif) est utilisé pour réentraîner le modèle périodiquement

6.4 Flux d'Audit et Logging

1. Toute action sensible (lecture/écriture/suppression) déclenche un trigger PostgreSQL

2. Le trigger insère un enregistrement dans la table `audit_logs` avec : user_id, action, table_name, record_id, timestamp, ip_address
3. Les logs sont répliqués vers un système SIEM externe (Splunk, ELK) via log shipping
4. Le SIEM analyse les logs pour détecter des patterns suspects (accès inhabituels, tentatives de brute-force)
5. En cas de détection d'anomalie, génération d'une alerte pour investigation

---

7. CONTRAINTES ET LIMITES

7.1 Contraintes Techniques

7.1.1 Performance

La latence cible de 500 ms pour le scoring de fraude implique une optimisation fine du modèle ML (réduction de complexité, utilisation de modèles légers) et l'usage d'un cache pour les résultats récents.

7.1.2 Scalabilité

Le passage à l'échelle horizontal nécessite une gestion de session distribuée (Redis Cluster) et une répartition de charge pour les APIs. PostgreSQL peut devenir un goulot d'étranglement au-delà de plusieurs milliers de transactions par seconde, nécessitant du sharding ou une migration vers une architecture distribuée (Citus, CockroachDB).

7.2 Limitations des Outils No-Code

Make.com et n8n, bien que puissants, présentent des limites : - **Complexité des workflows** : au-delà d'une dizaine d'étapes, la maintenance devient difficile - **Gestion des erreurs** : mécanismes de retry et de rollback moins robustes que du code applicatif - **Vendor lock-in** : dépendance à la plateforme SaaS (Make.com)

Pour les workflows critiques, un développement en code (Python, Node.js) est préférable.

7.3 Dépendances Externes

La solution repose sur des services tiers : - **Supabase Cloud** : dépendance à la disponibilité du service (SLA de 99.9%) - **Make.com** : limitation du nombre d'opérations mensuelles selon l'abonnement - **Metabase Cloud** : stockage des dashboards sur infrastructure externe

Une stratégie d'atténuation consiste à privilégier les solutions self-hostables (n8n, Grafana, Metabase) pour conserver le contrôle total.

7.4 Budget et Ressources

Les coûts principaux incluent : - **Infrastructure** : hébergement PostgreSQL (serveur dédié ou DBaaS), serveurs applicatifs - **Licences/abonnements** : Make.com, Supabase Pro (si fonctionnalités avancées requises) - **Ressources humaines** : développeurs backend/frontend, data scientist pour le ML, administrateur sécurité

Le budget estimé pour la phase MVP (Minimum Viable Product) est de 50 000 € (infrastructure + développement 3 mois).

---

8. CONCLUSION

8.1 Récapitulatif de l'Architecture

La plateforme de monitoring et de sécurité conçue pour DigitalBank France repose sur une architecture moderne, modulaire et sécurisée, intégrant des technologies open-source éprouvées (PostgreSQL, Flask, Grafana, Metabase) et des solutions no-code/low-code (Supabase, Make.com) pour accélérer le time-to-market.

L'architecture multi-couches garantit une séparation claire des responsabilités, facilite les évolutions futures et optimise la résilience face aux pannes et aux cyberattaques. L'implémentation de mécanismes de sécurité multicouches (authentification forte, RBAC, RLS, chiffrement, audit) répond aux exigences réglementaires et rétablit la confiance des parties prenantes suite à l'incident de ransomware.

8.2 Avantages de la Solution

- **Rapidité de déploiement** : l'usage de Supabase et Make.com réduit le temps de développement de 40% par rapport à une stack entièrement codée
- **Coûts maîtrisés** : privilégier l'open-source limite les coûts de licensing
- **Sécurité renforcée** : adoption du modèle Zero Trust et implémentation de défenses en profondeur
- **Conformité RGPD** : respect des principes de protection des données dès la conception
- **Évolutivité** : architecture scalable horizontalement pour accompagner la croissance

8.3 Perspectives d'Évolution

Phase 2 (6-12 mois)

- **Machine Learning avancé** : déploiement de modèles de Deep Learning (réseaux de neurones LSTM) pour détecter des patterns complexes de fraude
- **Analyse comportementale** : profiling des utilisateurs pour détecter les déviations comportementales (changement d'habitudes de navigation, tentatives d'accès inhabituelles)
- **Blockchain pour l'audit** : exploration d'une blockchain privée pour garantir l'inaltérabilité des logs d'audit

Phase 3 (12-24 mois)

- **Architecture microservices** : découpage de l'application monolithique en microservices indépendants pour améliorer la résilience et faciliter les déploiements continus
- **Conteneurisation et orchestration** : migration vers Docker et Kubernetes pour optimiser l'utilisation des ressources et automatiser le scaling
- **Intelligence artificielle explicable (XAI)** : implémentation de techniques d'explicabilité (SHAP, LIME) pour justifier les décisions du modèle de détection de fraude

8.4 Recommandations

Pour garantir le succès du projet, les recommandations suivantes sont formulées :

1. **Gouvernance de la sécurité** : désigner un RSSI (Responsable de la Sécurité des Systèmes d'Information) dédié pour superviser la conformité continue
2. **Formation des équipes** : investir dans la montée en compétences des équipes sur les technologies déployées (PostgreSQL, Flask, Supabase)
3. **Audits réguliers** : conduire des audits de sécurité trimestriels (pentests, revues de code) pour identifier et corriger les vulnérabilités
4. **Culture DevSecOps** : intégrer la sécurité dans le cycle de développement (Security as Code, tests de sécurité automatisés)
5. **Communication transparente** : informer régulièrement les clients des mesures de sécurité mises en place pour restaurer la confiance

---

FIN DU DOCUMENT

Références

- ANSSI. (2021). Guide d'hygiène informatique. Agence Nationale de la Sécurité des Systèmes d'Information.
- CNIL. (2018). Règlement Général sur la Protection des Données (RGPD).
- OWASP. (2021). OWASP Top 10 - 2021. Open Web Application Security Project.
- PostgreSQL Global Development Group. (2024). PostgreSQL 16 Documentation.
- Supabase. (2024). Supabase Documentation. https://supabase.com/docs

---

Document rédigé par : Groupe TRIO INFERNAL

Version : 1.0
Date : Janvier 2026