JUSTIFICATION DES CHOIX TECHNOLOGIQUES

Plateforme de Monitoring et Sécurité
DigitalBank France

Groupe : TRIO INFERNAL

Formation : Master ESIC / CPDIA

Janvier 2026

TABLE DES MATIÈRES

- 1. Introduction
- 2. Critères de Décision
- 3. Choix de la Base de Données : PostgreSQL
- 4. Choix de la Plateforme Backend : Supabase
- 5. Choix de l'API ML : Flask (Python)
- 6. Choix des Dashboards
- 7. Choix de l'Automatisation : Make / n8n
- 8. Architecture de Sécurité
- 9. Synthèse Comparative
- 10. Conclusion

1. INTRODUCTION

Ce document a pour objectif de détailler, d'analyser et de justifier l'ensemble des choix technologiques opérés pour la conception et le développement de la plateforme de monitoring et de sécurité de **DigitalBank France**. Ce projet intervient dans un contexte critique de reconstruction post-incident, suite à une cyberattaque majeure (ransomware) ayant impacté l'infrastructure précédente.

La sélection des technologies ne repose pas uniquement sur des critères de performance brute, mais s'inscrit dans une démarche globale visant à restaurer la confiance, garantir l'intégrité des données financières et assurer une conformité stricte avec les régulations en vigueur.

La méthodologie employée est une analyse comparative multicritères, mettant en balance les solutions leaders du marché face aux contraintes spécifiques d'un établissement bancaire digital : sécurité par conception (Security by Design), souveraineté des données, et agilité de déploiement.

2. CRITÈRES DE DÉCISION

Pour assurer la pérennité et la robustesse de la solution, sept axes majeurs ont guidé nos décisions techniques :

Sécurité et conformité réglementaire

Critère absolu pour une banque. Les solutions doivent supporter nativement les standards de chiffrement forts, l'authentification multi-facteurs (MFA) et permettre la conformité avec la DSP2 (Directive sur les Services de Paiement 2) et les normes PCI-DSS pour les données de cartes.

Scalabilité et performance

La plateforme doit pouvoir absorber des pics de charge transactionnels sans latence, garantissant une expérience fluide pour les clients et une réactivité temps réel pour les analystes sécurité.

Coût total de possession (TCO)

Au-delà du coût de licence, nous évaluons le coût d'infrastructure, de maintenance et de formation des équipes. L'utilisation de solutions Open Source est privilégiée pour maîtriser ces coûts.

Facilité de mise en œuvre

Dans un contexte de reconstruction d'urgence, la rapidité de déploiement (Time-to-Market) est cruciale. Les solutions Low-Code ou "Batteries Included" sont favorisées.

Support et communauté

La pérennité des technologies repose sur la vitalité de leur communauté et la disponibilité d'une documentation exhaustive, réduisant les risques de dette technique.

Conformité RGPD et souveraineté

La capacité à héberger les données sur le territoire européen ou en propre (Self-hosting) est indispensable pour respecter le RGPD et garantir la souveraineté numérique de la banque.

Intégration et interopérabilité

L'architecture doit être modulaire. Les composants doivent communiquer via des standards ouverts (API REST, JSON, SQL) pour éviter le verrouillage technologique (Vendor Lock-in).

3. CHOIX DE LA BASE DE DONNÉES : POSTGRESQL

Le cœur du système d'information repose sur **PostgreSQL**. Ce choix est stratégique et s'impose comme le standard industriel pour les applications financières modernes.

Justification détaillée

- **Robustesse et Intégrité :** PostgreSQL est un SGBD relationnel mature respectant strictement les propriétés ACID (Atomicité, Cohérence, Isolation, Durabilité), garantissant qu'aucune transaction financière ne soit perdue ou corrompue, même en cas de panne.
- **Polyvalence des données (JSONB) :** Contrairement aux bases purement relationnelles, PostgreSQL excelle dans la gestion de documents JSON via le type `JSONB`. Cela permet de stocker des logs d'audit semi-structurés ou des métadonnées de transaction complexes sans rigidifier le schéma.
- **Sécurité Avancée :** Il intègre nativement le RLS (Row Level Security), permettant de définir des politiques d'accès à la donnée au niveau de la ligne, une fonctionnalité clé pour isoler les données clients dans une architecture multi-tenant.
- **Extensibilité :** L'écosystème d'extensions est vaste. Nous utiliserons `pgcrypto` pour le hachage et le chiffrement, et `pg_audit` pour une traçabilité immuable des actions

administrateurs.

Tableau comparatif des SGBD

Critère	PostgreSQL (Choix)	MySQL / MariaDB	MongoDB	Oracle DB
Type	Relationnel + NoSQL	Relationnel	NoSQL (Document)	Relationnel
Transactions	ACID Strict	ACID	ACID (récent/complexe)	ACID Excellent
Fonctionnalités	Avancées (RLS, JSONB)	Standard	Flexibles mais moins strictes	Très avancées
Coût	Gratuit (Open Source)	Gratuit (Open Source)	Gratuit / Licence Ent.	Très élevé (Licences)
Verdict	**Retenu**	Rejeté (Moins riche)	Rejeté (Intégrité -)	Rejeté (Coût/Complexité)

4. CHOIX DE LA PLATEFORME BACKEND : SUPABASE

Pour accélérer le développement tout en maintenant un niveau de sécurité maximal, nous avons sélectionné **Supabase** comme plateforme Backend-as-a-Service (BaaS). Supabase se positionne comme une alternative Open Source à Google Firebase, construite directement au-dessus de PostgreSQL.

Justification détaillée

Architecture centrée sur la base de données : Contrairement à d'autres solutions qui masquent la base de données, Supabase embrasse PostgreSQL. Chaque table créée génère automatiquement une API REST et GraphQL documentée et sécurisée. Cela élimine des centaines d'heures de développement d'API "boilerplate" (CRUD).

Sécurité et Authentification (Auth) : Supabase fournit un système d'authentification complet prêt à l'emploi, supportant l'email/mot de passe, les liens magiques, et surtout le MFA (Multi-Factor Authentication), indispensable pour la conformité bancaire. L'intégration avec les JWT (JSON Web Tokens) permet de propager l'identité de l'utilisateur jusqu'au moteur de base de données.

Row Level Security (RLS) : C'est l'atout majeur. La logique d'autorisation n'est pas dans le code applicatif (faillible), mais dans la base de données. Une règle RLS comme `auth.uid() = user_id` garantit mathématiquement qu'un client ne peut voir que ses propres comptes, quel que soit le point d'entrée de l'API.

Souveraineté : Étant Open Source, Supabase peut être auto-hébergé (Self-hosted) sur nos propres serveurs ou dans un cloud privé souverain, répondant ainsi aux exigences strictes de localisation des données bancaires, contrairement aux solutions propriétaires américaines.

Tableau comparatif Backend

Critère	Supabase (Choix)	Firebase (Google)	Développement Custom (Node/Django)	AWS Amplify
Base de données	PostgreSQL (SQL)	Firestore (NoSQL)	Au choix	DynamoDB (NoSQL)
Souveraineté	Oui (Self-hostable)	Non (Google Cloud)	Oui	Non (AWS)
Vitesse Dév.	Très élevée	Très élevée	Faible (Tout à coder)	Moyenne
Complexité requêtes	Forte (SQL complet)	Faible (Limitée)	Forte	Moyenne
Vendor Lock-in	Faible (Standard SQL)	Élevé	Nul	Élevé

5. CHOIX DE L'API ML : FLASK (PYTHON)

Pour le module spécifique de détection de fraude basé sur l'Intelligence Artificielle, nous avons opté pour une architecture micro-service utilisant **Flask**.

Justification

- **L'écosystème Python :** Python est le langage incontesté de la Data Science. Utiliser un framework Python permet d'intégrer nativement les bibliothèques de Machine Learning comme *scikit-learn*, *pandas*, ou *TensorFlow* sans couches d'interopérabilité complexes.
- **Légèreté et Performance :** Flask est un "micro-framework". Il n'impose pas de structure lourde, ce qui est idéal pour créer une API REST simple et performante dédiée à une tâche unique : recevoir des données de transaction, appliquer un modèle prédictif, et retourner un score de fraude.
- **Facilité de déploiement :** Un service Flask se conteneurise très facilement via Docker, permettant une scalabilité horizontale indépendante du reste de l'application. Si la charge d'analyse augmente, nous pouvons multiplier les instances de l'API ML sans toucher au backend principal.

Comparaison Frameworks ML

Framework	Flask	FastAPI	Django	Node.js
Usage principal	Micro-services, ML	APIs haute perf.	Applications Web Fullstack	Web I/O
Complexité	Faible	Moyenne	Élevée	Moyenne
Support ML	Natif (Python)	Natif (Python)	Natif (Python)	Faible
Verdict	**Retenu (Simplicité)**	Alternative sérieuse	Trop lourd	Inadapté au calcul

6. CHOIX DES DASHBOARDS

La visualisation des données est scindée en deux outils distincts pour répondre à deux besoins différents : le monitoring métier et le monitoring technique.

6.1 Metabase (Métier & Décisionnel)

Metabase est retenu pour l'analyse métier (Business Intelligence). C'est un outil Open Source permettant aux équipes non techniques (analystes fraude, direction) de créer des requêtes et des tableaux de bord sans écrire de SQL. Il se connecte directement à PostgreSQL.

- **Avantages :** Interface "No-Code", envois de rapports programmés par email, intégration (embedding) facile dans des portails internes.
- **Usage :** Suivi des volumes de transactions, répartition géographique des clients, KPI financiers.

6.2 Grafana (Technique & Sécurité)

Grafana est la référence pour le monitoring de séries temporelles. Il sera utilisé par les équipes IT et Sécurité (SOC).

- **Avantages :** Capacités d'alerting avancées (Seuils CPU, tentatives de login échouées), visualisation en temps réel, agrégation de logs.
- **Usage :** Surveillance de la santé des serveurs, détection d'anomalies de trafic, monitoring des API.

Comparativement à Power BI ou Tableau, ces solutions évitent des coûts de licence prohibitifs et conservent la souveraineté des données en restant hébergées en interne.

7. CHOIX DE L'AUTOMATISATION : MAKE / N8N

L'orchestration des flux de travail (Workflows) est essentielle pour réagir rapidement aux incidents.

Make.com (ex-Integromat)

Solution SaaS retenue pour sa simplicité visuelle lors du prototypage rapide. Elle permet de connecter des API disparates (Slack, Email, Supabase) via une interface "Drag-and-Drop".

n8n (Production & Sécurité)

Pour la production critique, nous privilégions **n8n**. C'est un outil d'automatisation de workflow "fair-code" qui est self-hostable. Cela signifie que les données sensibles transitant dans les workflows (ex: détails d'une transaction suspecte) ne quittent jamais l'infrastructure de la banque, contrairement à une solution purement SaaS comme Zapier.

Cas d'usage type : Lorsqu'une fraude est détectée par l'API Flask (> score 0.8), n8n déclenche automatiquement : 1. Le blocage temporaire du compte via Supabase, 2. L'envoi d'un email au client, 3. Une notification Slack au service sécurité.

8. ARCHITECTURE DE SÉCURITÉ

L'approche adoptée est celle de la **Défense en Profondeur** (Defense-in-Depth), multipliant les barrières de protection.

- **Authentification Forte :** Supabase Auth gère l'identité avec obligation de MFA pour les accès administrateurs.
- **Contrôle d'accès (Authorization) :** Le modèle RBAC (Role-Based Access Control) définit les rôles (Admin, Analyste, Client). Ces rôles sont appliqués au niveau le plus bas via les politiques RLS de PostgreSQL.
- **Chiffrement :** Toutes les communications sont chiffrées en transit (TLS 1.3). Les données sensibles (mots de passe, clés API) sont chiffrées au repos (At-Rest) via pgcrypto.
- **Auditabilité :** L'utilisation combinée des logs applicatifs et de `pg_audit` assure une traçabilité inviolable des actions, répondant aux exigences d'investigation post-incident (Forensics).
- **Isolation :** En cas de compromission d'un service frontal, la base de données reste protégée par ses propres règles de sécurité internes.

9. SYNTHÈSE COMPARATIVE

Le tableau ci-dessous résume l'adéquation de notre stack technique avec les critères initiaux.

Critère	PostgreSQL	Supabase	Flask (ML)	Metabase	n8n
Sécurité	⭐⭐⭐⭐⭐	⭐⭐⭐⭐⭐	⭐⭐⭐⭐	⭐⭐⭐⭐	⭐⭐⭐⭐
Coût	⭐⭐⭐⭐⭐	⭐⭐⭐⭐	⭐⭐⭐⭐⭐	⭐⭐⭐⭐⭐	⭐⭐⭐⭐
Scalabilité	⭐⭐⭐⭐⭐	⭐⭐⭐⭐	⭐⭐⭐⭐	⭐⭐⭐	⭐⭐⭐⭐
Facilité	⭐⭐⭐	⭐⭐⭐⭐⭐	⭐⭐⭐⭐	⭐⭐⭐⭐⭐	⭐⭐⭐⭐⭐
RGPD	⭐⭐⭐⭐⭐	⭐⭐⭐⭐⭐	⭐⭐⭐⭐	⭐⭐⭐⭐	⭐⭐⭐⭐
Support	⭐⭐⭐⭐⭐	⭐⭐⭐⭐	⭐⭐⭐⭐⭐	⭐⭐⭐⭐	⭐⭐⭐⭐

La cohérence globale est assurée par l'omniprésence du standard SQL et de l'architecture API-First, facilitant l'interconnexion de tous les modules.

10. CONCLUSION

La stack technologique proposée pour le projet **TRIO INFERNAL** de DigitalBank France constitue une réponse pragmatique, moderne et sécurisée aux défis posés. En s'appuyant sur des fondations Open Source éprouvées (PostgreSQL, Python) et des accélérateurs de développement (Supabase, n8n), nous atteignons un équilibre optimal entre :

- La **sécurité et la conformité**, impératifs non négociables du secteur bancaire.
- La **maîtrise des coûts**, via l'élimination des licences propriétaires onéreuses.
- La **souveraineté des données**, grâce à la capacité d'auto-hébergement de l'ensemble de la chaîne.

Cette architecture est conçue pour évoluer. Elle permet de démarrer rapidement avec des outils Low-Code tout en laissant la porte ouverte à des développements sur-mesure (via Flask ou des fonctions PostgreSQL) lorsque la complexité l'exigera.