Rapport Technique: Secure Digital Card

1. Introduction

Ce projet consiste en le développement d'une **application bancaire sécurisée** nommée "Secure Digital Card". L'objectif principal est de fournir aux utilisateurs une plateforme robuste pour la gestion de leurs cartes bancaires virtuelles, en mettant un accent particulier sur la **sécurité des données** et l'**analyse des transactions**.

L'architecture technique repose sur une pile moderne et performante, garantissant à la fois la scalabilité du backend et une expérience utilisateur fluide et réactive.

2. Architecture Technique

L'application est conçue selon une architecture **client-serveur découplée**, utilisant les technologies suivantes :

Composant	Technologie	Rôle
Backend (API)	**Django (Python)**	Fournit les services RESTful sécurisés, gère la logique métier, l'authentification et les opérations de chiffrement.
Base de Données	**MongoDB**	Base de données NoSQL flexible, utilisée pour la persistance des données. L'intégration avec Django est assurée par **Djongo**.
Frontend (Client)	**React & Vite**	Application monopage (SPA) construite avec React pour une interface utilisateur dynamique et performante. Vite est utilisé comme outil de build rapide.

Cette architecture permet une séparation claire des responsabilités, facilitant le développement, la maintenance et l'évolution des deux parties de l'application.

3. Présentation Fonctionnelle

L'application "Secure Digital Card" offre un ensemble de fonctionnalités centrées sur la sécurité et la gestion des finances personnelles :

- **Gestion des Cartes Virtuelles :** Création, visualisation et gestion des cartes bancaires virtuelles. Les données sensibles (numéro de carte, CVV) sont stockées de manière **chiffrée**.
- **Authentification Sécurisée :** Le système intègre l'**authentification à deux facteurs (2FA)** pour renforcer la sécurité de l'accès utilisateur.
- **Suivi des Transactions :** Enregistrement et affichage en temps réel de tous les mouvements financiers liés aux cartes.
- **Surveillance de Sécurité :** Suivi des appareils connectés et journalisation des actions critiques (`AuditLog`) pour une traçabilité complète.
- **Analyse Intelligente :** Un module d'analyse est intégré pour détecter les modèles de dépenses et identifier les comportements suspects pouvant signaler une **fraude potentielle**.

4. Modélisation du Système (UML)

4.1. Diagramme de Classes

Ce diagramme illustre la structure statique du système, les classes principales et leurs relations.

[Diagramme de Classes - Voir document original, Page 1]

4.2. Diagramme de Séquence (Authentification)

Ce diagramme détaille le flux d'interactions entre l'utilisateur, le frontend et le backend lors du processus d'authentification sécurisée.

[Diagramme de Séquence (Authentification) - Voir document original, Page 2]

4.3. Diagramme de Déploiement

Ce diagramme représente l'architecture physique du système, montrant comment les composants logiciels sont déployés sur les nœuds matériels.

[Diagramme de Déploiement - Voir document original, Page 3]

4.4. Diagramme de Composants

Ce diagramme présente l'organisation et les dépendances des composants logiciels du système.

[Diagramme de Composants - Voir document original, Page 4]

5. Description Détaillée des Classes et Fonctions

5.1. Description des Classes

Les classes principales du backend (Django/MongoDB) sont les suivantes :

Classe	Rôle Principal	Attributs Clés (Exemples)
User	Représente l'utilisateur final. Gère l'identité, les préférences de sécurité (2FA) et les informations de contact.	`email`, `password_hash`, `is_2fa_enabled`
Card	Représente une carte bancaire virtuelle. Stocke les informations sensibles de manière chiffrée et gère les limites de dépenses.	`encrypted_number`, `limit`, `is_blocked`
Transaction	Enregistre chaque mouvement financier (paiement, remboursement), incluant le statut et le commerçant.	`amount`, `merchant`, `date`, `status`
Device	Suit les appareils utilisés pour la connexion afin de détecter les accès suspects ou non autorisés.	`device_id`, `last_login_ip`, `user_agent`
AuditLog	Journalise toutes les actions critiques (connexion, blocage de carte) pour la tracibilité et le calcul de risque.	`action_type`, `user`, `timestamp`, `risk_score`

5.2. Tableau des Fonctions Clés

Le tableau ci-dessous présente les fonctions essentielles implémentées dans le backend pour assurer les fonctionnalités de l'application :

Fonction	Classe Principale	Description/Rôle
`register()` / `login()`	`User`	Gère l'inscription et l'authentification des utilisateurs.
`set_number(raw)`	`Card`	Chiffre et stocke le numéro de carte de manière sécurisée dans la base de données.
`get_number()`	`Card`	Déchiffre et retourne le numéro de carte (pour l'affichage temporaire et autorisé sur le frontend).
`save()`	`Transaction`	Enregistre une transaction et met à jour les soldes ou les limites de dépenses si nécessaire.
`log_action()`	`AuditLog`	Crée une entrée de journal pour une action spécifique, incluant un score de risque associé.
`track_login()`	`Device`	Enregistre ou met à jour les informations de l'appareil lors de la connexion de l'utilisateur.

6. Interfaces Utilisateur (Frontend React)

Le frontend est structuré autour de trois interfaces principales, conçues pour une expérience utilisateur intuitive et sécurisée.

6.1. Interface de Connexion (`Login.jsx`)

- **Description :** Page d'entrée sécurisée de l'application.
- **Composants UI :** Formulaire central avec champs "Email" et "Mot de passe", bouton "Se connecter", liens "Mot de passe oublié" et "S'inscrire".
- **Comportement :** Envoie les identifiants à l'API. En cas de succès, le token JWT est stocké localement et l'utilisateur est redirigé vers le Dashboard. La gestion des erreurs est assurée par l'affichage de messages d'erreur clairs.

[Interface Login - Voir document original, Page 6]

6.2. Interface Principale (`Dashboard.jsx`)

- **Description :** Vue d'ensemble de l'activité financière de l'utilisateur.
- **Composants UI :**
  - **Sidebar :** Menu de navigation (`Dashboard`, `Cartes`, `Transactions`, `Profil`).
  - **Stats Cards :** Résumé rapide des indicateurs clés (Solde total, Dépenses du mois).
  - **Graphique :** Visualisation des tendances de dépenses récentes.
  - **Liste Rapide :** Affichage des dernières transactions effectuées.
- **Comportement :** Charge les données agrégées via des appels API asynchrones au montage du composant. Permet la navigation vers les vues détaillées.

[Interface Dashboard - Voir document original, Page 6]

6.3. Interface de Détails Carte (`CardDetails.jsx`)

- **Description :** Gestion fine et sécurisée d'une carte virtuelle spécifique.
- **Composants UI :** Représentation visuelle de la carte (format carte bancaire), bouton "Voir le numéro" (pour révéler temporairement les chiffres masqués), boutons d'action (`Bloquer/Débloquer`, `Modifier plafonds`).
- **Comportement :** Interagit avec l'API pour modifier l'état (`is_blocked`) ou pour déchiffrer temporairement le PAN (Primary Account Number) après une validation de sécurité supplémentaire.

[Interface Détails Carte - Voir document original, Page 7]