Rapport Technique

À: Ahmed Imed Eddine
DE: Marina Kamel

1. Introduction

Le projet CADS (Client Accounting Data Store) est une plateforme web destinée à centraliser la gestion comptable des clients et à automatiser le suivi des obligations fiscales. Conçu pour les cabinets comptables et les professionnels de la finance, le système offre une interface moderne, une architecture modulaire et une sécurité renforcée grâce à l'intégration de Clerk pour l'authentification et l'utilisation de JWT pour la sécurisation des échanges.

L'objectif principal est de simplifier la gestion quotidienne des dossiers clients tout en assurant la confidentialité, la traçabilité et la fiabilité des données. Ce rapport présente l'architecture technique, les choix technologiques, les défis rencontrés lors du développement et les perspectives d'évolution du système.

2. Architecture du Projet

2.1 Vue d'ensemble

CADS repose sur une architecture moderne de type client–serveur, séparant clairement le frontend (React 18 + Vite), le backend (Node.js, Express, TypeScript) et la base de données (PostgreSQL sur Neon.tech). Le frontend communique avec l'API REST via des requêtes HTTP sécurisées, tandis que Prisma ORM assure la cohérence et la simplicité des opérations CRUD.

2.2 Diagramme d'architecture

Frontend
React 18 + Vite
Auth via Clerk (JWT)

Backend
Node.js + Express + TS
Prisma ORM + Clerk SDK
API REST sécurisée (JWT)

Base de Données
PostgreSQL (Neon.tech)
Tables : Users, Clients,
Alerts, Deadline ,Documents

2.3 Technologies Utilisées

Frontend : React 18, Vite, Fetch API, Recharts
Backend : Node.js, Express, TypeScript
Base de données : PostgreSQL (Neon.tech)
ORM : Prisma
Authentification : Clerk (JWT)
Hébergement : Local
Sécurité : Variables d'environnement, chiffrement des données

2.4 Structure de la Base de Données

Le modèle de données suit une approche relationnelle optimisée incluant les tables suivantes : Clients (informations personnelles, coordonnées, NAS crypté), Alerts (alertes automatiques liées aux échéances), Deadlines (dates limites liées aux obligations fiscales), Users (comptes utilisateurs synchronisés avec Clerk), Documents (pièces justificatives associées aux clients). Les relations sont gérées via des clés étrangères assurant la cohérence et une traçabilité complète.

3. Authentification et Sécurité

3.1 Justification du Système d'Authentification

Clerk a été choisi pour sa facilité d'intégration avec React et Express, ainsi que pour sa gestion complète du cycle d'authentification : inscription, connexion, réinitialisation de mot de passe et gestion des sessions. Les JWT sont utilisés pour sécuriser les échanges entre client et serveur : chaque requête authentifiée inclut un jeton valide dans l'en-tête Authorization.

3.2 Implémentation Technique

Frontend : intégration de <ClerkProvider>, utilisation de useUser() et useAuth(), transmission automatique du token via fetch() (Authorization: Bearer <token>).
Backend : middleware Express vérifiant les tokens JWT via la clé secrète Clerk, liaison des utilisateurs Clerk à la table interne Users, protection des routes sensibles : /api/clients, /api/alerts, /api/documents.

3.3 Sécurité des Données

Chiffrement des informations sensibles (NAS, documents), utilisation rigoureuse des variables d'environnement, validation renforcée des entrées côté serveur, gestion des rôles et permissions pour limiter l'accès aux données critiques.

4. Fonctionnalités Frontend

4.1 Dashboard

Le tableau de bord présente des statistiques comptables sous forme de graphiques dynamiques via Recharts. Les données sont chargées de manière asynchrone pour optimiser la réactivité.

4.2 Gestion des Clients

Formulaire d'ajout et de modification, indicateurs visuels de statut (Actif/Inactif), chiffrement des données sensibles avant transmission au backend.

4.3 Page Obligations

Cette page regroupe l'ensemble des obligations fiscales d'un client, avec un système de couleur selon le niveau de priorité. Elle permet d'ajouter de nouvelles échéances et de marquer une obligation comme complétée.

4.4 Alertes et Notifications

Les alertes sont générées automatiquement à partir des données de la table Deadlines et informent l'utilisateur des échéances à venir.

4.5 Gestion des Documents

Téléversement sécurisé, prévisualisation, suppression ; stockage local dans un espace protégé.

5. Défis Rencontrés et Solutions Apportées

5.1 Transmission du Token JWT

Problème : les requêtes échouaient lorsque le token n'était pas ajouté correctement. Solution : création d'un module utilitaire chargé d'insérer automatiquement le token dans chaque requête.

5.2 Sécurisation du NAS

Problème : nécessité de protéger les numéros d'assurance sociale. Solution : chiffrement au stockage ; déchiffrement uniquement côté serveur.

5.3 Performance du Dashboard

Problème : ralentissement dû au chargement initial des statistiques. Solution : chargement asynchrone et rendu progressif des graphiques.

5.4 Gestion des Documents Volumineux

Problème : impact négatif sur les performances. Solution : chargement différé et aperçu dynamique.

6. Améliorations Futures

6.1 Module de Facturation

Génération automatique de factures PDF à partir des données clients.

6.2 Notifications Automatisées

Envoi de rappels par courriel ou SMS.

6.3 Dashboard Avancé

Filtres dynamiques, graphiques interactifs, KPIs financiers.

6.4 Gestion Avancée des Rôles

Ajout de profils hiérarchiques (Administrateur, Comptable, Assistant).

6.5 Sauvegarde et Archivage Cloud

Intégration d'un stockage externe tel qu'AWS S3.

6.6 Intégration d'API Externes

Connexion à des API fiscales ou bancaires pour automatiser la récupération d'informations.

7. Conclusion

CADS constitue une solution complète et moderne pour la gestion comptable des clients. Son architecture modulaire, son système d'authentification robuste et son interface intuitive garantissent une utilisation fiable et sécurisée. Les améliorations prévues permettront d'enrichir la plateforme et de la positionner comme un outil polyvalent et performant répondant aux standards actuels du secteur financier.