Rapport de Projet : Application de Suivi des Dépenses (Expense Tracker)

Équipe de Développement

18 décembre 2025

Table des matières

1 Introduction

Ce document présente le rapport technique détaillé de l'application "Expense Tracker". Il s'agit d'une application web complète (Full Stack) permettant aux utilisateurs de gérer leurs finances personnelles en suivant leurs revenus et leurs dépenses. L'application offre une interface utilisateur moderne et réactive, couplée à une API robuste pour la persistance des données.

2 Architecture Globale

Le projet repose sur l'architecture **MERN** (MongoDB, Express.js, React, Node.js), séparant clairement le frontend (client) du backend (serveur).

- **Frontend** : Application React créée avec Vite, utilisant TailwindCSS pour le style.
- **Backend** : Serveur Node.js avec le framework Express.
- **Base de Données** : MongoDB pour le stockage des transactions.

3 Backend (Serveur)

Le backend est responsable de la logique métier et de la gestion des données. Il expose une API RESTful consommée par le client.

3.1 Technologies Utilisées

- **Node.js** : Environnement d'exécution JavaScript côté serveur.
- **Express** : Framework web pour structurer l'application et gérer les routes.
- **Mongoose** : ODM (Object Data Modeling) pour faciliter les interactions avec MongoDB.
- **Dotenv** : Gestion des variables d'environnement.
- **Cors** : Middleware pour permettre les requêtes cross-origin.

3.2 Structure du Code

Le serveur est structuré de manière modulaire :

- `server.js` : Point d'entrée de l'application. Il initialise la connexion à la base de données, configure les middlewares (CORS, JSON parsing) et définit les routes principales.
- `config/db.js` : Gère la connexion à la base de données MongoDB.
- `models/Transaction.js` : Définit le schéma de données pour une transaction.
- `routes/TransactionRoutes.js` : Définit les endpoints de l'API (GET, POST, PUT, DELETE).
- `controllers/` : Contient la logique métier associée à chaque route (récupération, création, mise à jour et suppression des transactions).

3.3 API Endpoints

L'API expose les routes suivantes sous le préfixe `/api/transactions` :

- **GET** / : Récupère la liste de toutes les transactions.

— **POST** / : Crée une nouvelle transaction.
— **PUT** / **:id** : Met à jour une transaction existante.
— **DELETE** / **:id** : Supprime une transaction.

4 Base de Données

Les données sont stockées dans une base MongoDB. Le modèle principal est `Transaction`.

4.1 Schéma Transaction

Le schéma Mongoose (`Transaction.js`) définit la structure suivante :

```
const transactionSchema = new mongoose.Schema({
  description: { type: String, required: true },
  amount: { type: Number, required: true },
  type: { type: String, enum: ["income", "expense"], default: "expense" },
  category: { type: String, required: true },
  date: { type: Date, default: Date.now },
  note: { type: String }
}, { timestamps: true });
```

5 Frontend (Client)

Le frontend est une application monopage (SPA) développée avec React, offrant une expérience utilisateur fluide et dynamique.

5.1 Technologies et Bibliothèques

— **React** : Bibliothèque principale pour la construction de l'interface.
— **Vite** : Outil de build rapide et serveur de développement.
— **TailwindCSS** : Framework CSS utilitaire pour un design rapide et responsive.
— **Recharts** : Pour la visualisation des données (graphiques).
— **Axios** : Pour effectuer les requêtes HTTP vers le backend.
— **Lucide React** : Pour les icônes.
— **Radix UI** : Composants accessibles non stylés (utilisés pour certains éléments d'interface).

5.2 Composants Principaux

L'application est composée de plusieurs composants réutilisables situés dans `src/components` :
— **App.jsx** : Composant racine. Il gère l'état global (liste des dépenses, valeur nette) et orchestre les appels API initiaux.
— **BalanceCard** : Affiche le solde actuel et la valeur nette.
— **ExpenseForm** : Formulaire permettant d'ajouter ou de modifier une transaction.
— **ExpenseList** : Affiche la liste des transactions récentes avec des options pour modifier ou supprimer.
— **ExpenseChart** : Visualisation graphique des dépenses par catégorie.

— **ThemeProvider** & **ThemeToggle** : Gestion du mode sombre/clair.

5.3 Fonctionnalités Clés

— **Tableau de Bord** : Vue d'ensemble des finances avec graphiques et indicateurs clés.
— **Gestion des Transactions** : Ajout, modification et suppression intuitifs des revenus et dépenses.
— **Mode Sombre** : Support natif du thème sombre pour le confort visuel.
— **Responsive Design** : Interface adaptée aux mobiles et aux bureaux grâce à TailwindCSS.

6 Conclusion

Ce projet démontre une application web moderne et fonctionnelle. L'utilisation de la stack MERN permet une cohérence du langage (JavaScript) à travers toute la pile technologique. L'architecture modulaire du backend et l'approche par composants du frontend facilitent la maintenance et les évolutions futures du projet.