Rapport de Projet : Application de Gestion de Tâches

# 1. Introduction

Ce projet consiste en la réalisation d'une application web de gestion de tâches, basée sur une architecture client-serveur moderne. Le backend est développé avec Node.js et Express, tandis que le frontend est construit avec React. Les données sont stockées dans une base de données MySQL, et les échanges entre le client et le serveur se font via des requêtes HTTP RESTful.

# 2. Architecture du projet

Le projet est organisé selon la structure suivante :

- api-server/ : Contient le serveur backend Express  
- client/ : Contient le frontend React  
- Base de données MySQL : utilisée pour stocker les tâches

# 3. Backend (Express.js)

Le serveur backend utilise Express.js pour définir une API RESTful qui expose des endpoints permettant de gérer les tâches (GET, POST, PUT, DELETE). CORS et body-parser sont utilisés pour gérer les requêtes entrantes et autoriser les échanges entre le client et le serveur.

Une connexion à une base de données MySQL permet de persister les données. Les opérations CRUD sont implémentées via des requêtes SQL exécutées dans les routes Express.

# 4. Frontend (React)

L'interface utilisateur est réalisée avec React. Deux composants principaux sont utilisés :  
- TaskForm : formulaire pour ajouter une nouvelle tâche  
- TaskList : liste des tâches avec options de modification et de suppression

Axios est utilisé pour effectuer les appels à l'API. Les hooks React (useState, useEffect) permettent de gérer l'état local et les effets de bord comme le chargement des données à l'initialisation.

# 5. Communication Client-Serveur

Les échanges entre le client et le serveur sont réalisés via des requêtes HTTP. Chaque opération (lecture, ajout, mise à jour, suppression) utilise les méthodes GET, POST, PUT et DELETE respectivement. Les données sont échangées au format JSON.

# 6. Base de Données

La base de données utilisée est MySQL. Une table 'tasks' stocke les informations des tâches :  
- id (INT, clé primaire)  
- title (VARCHAR)  
- completed (BOOLEAN)

# 7. Stylisation et UX

L'application est stylisée avec du CSS de base. Les éléments sont centrés pour une meilleure lisibilité et les boutons permettent une interaction directe avec les éléments de la liste.

# 8. Conclusion

Ce projet démontre la mise en œuvre d'une architecture web complète et moderne. Il montre comment React peut être utilisé pour créer une interface utilisateur dynamique, comment Express peut servir une API performante, et comment MySQL peut assurer la persistance des données.