R5.A.10

Nouveaux paradigmes de bases de données

Cahier des charges

Projet : Gestionnaire de tâches avec MongoDB, Node.js et Interface Web

1. Contexte du projet

Vous êtes missionnés (par groupe de 2) pour concevoir une application de gestion de tâches. Cette application permettra à un utilisateur d'ajouter, consulter, modifier, supprimer et filtrer des tâches. L'objectif du projet est de vous faire manipuler une base de données NoSQL (MongoDB), de créer une API REST pour accéder aux données, et de proposer une interface web simple pour utiliser l'application.

2. Objectifs pédagogiques

• Concevoir un modèle de données document adapté à MongoDB.

• Implémenter un CRUD complet (Create, Read, Update, Delete) via une API REST en Node.js + Express.

• Réaliser une interface web simple permettant l'interaction avec cette API.

• Appliquer des opérations de filtrage et de tri sur les données.

3. Spécifications fonctionnelles

3.1 Modèle de données

Chaque tâche doit contenir les informations suivantes (modélisées dans MongoDB) :

Champ	Type	Description
titre	String	Titre court de la tâche
description	String	Détail de la tâche
dateCreation	Date	Date de création de la tâche
echeance	Date	Date limite de la tâche
statut	String	"à faire", "en cours", "terminée", "annulée"
priorite	String	"basse", "moyenne", "haute", "critique"
auteur	Objet	{ nom, prénom, email }
categorie	String	Type de tâche (perso, travail, projet, etc.)
etiquettes	Tableau de Strings	Liste de mots-clés
sousTaches	Tableau d'objets	titre, statut, échéance
commentaires	Tableau d'objets	auteur, date, contenu
historiqueModifications	Tableau d'objets	champModifie, ancienneValeur, nouvelleValeur, date

3.2 API REST (Node.js + Express)

Votre API REST doit proposer les routes suivantes :

• GET /tasks : récupérer toutes les tâches.

• GET /tasks/:id : récupérer une tâche par son identifiant.

• POST /tasks : créer une nouvelle tâche.

• PUT /tasks/:id : modifier une tâche existante.

• DELETE /tasks/:id : supprimer une tâche.

Fonctionnalités complémentaires obligatoires :

• Filtrer les tâches (statut, priorité, catégorie, étiquette, échéance).

• Trier les résultats (par date, priorité, etc.).

• Gérer les sous-tâches et commentaires.

• (Optionnel) Historiser les modifications.

3.3 Interface web (HTML/CSS/JS)

Interface simple permettant de :

• Afficher la liste des tâches.

• Consulter les détails (sous-tâches, commentaires).

• Ajouter, modifier, supprimer une tâche.

• Ajouter des commentaires.

• Gérer les sous-tâches.

• Filtrer et trier les tâches.

3.4 Critères de sélection pour l'affichage des tâches

Filtres disponibles via l'API :

Critère	Paramètre	Exemple	Description
Statut	statut	/tasks?statut=à faire	Tâches dans un état donné
Priorité	priorite	/tasks?priorite=haute	Filtrage par priorité
Catégorie	categorie	/tasks?categorie=perso	Filtrage par catégorie
Étiquette	etiquette	/tasks?etiquette=urgent	Recherche par étiquette
Date limite avant	avant	/tasks?avant=2025-03-31	Tâches à terminer avant une date
Date limite après	apres	/tasks?apres=2025-03-24	Tâches après une date

Texte libre	q	/tasks?q=rapport	Recherche dans titre/description

Tris possibles :

Critère	Paramètre	Exemple
Date d'échéance	tri=echeance	/tasks?tri=echeance
Priorité	tri=priorite	/tasks?tri=priorite
Date de création	tri=dateCreation	/tasks?tri=dateCreation
Ordre décroissant	ordre=desc	/tasks?tri=echeance&ordre=desc

4. Contraintes techniques

- Base de données : MongoDB (local ou Atlas).
- Backend : Node.js avec Express.
- Frontend : HTML/CSS/JavaScript (framework léger autorisé).
- Pas d'authentification requise.
- Projet réalisable localement ou en ligne.

5. Livrables attendus

- Modèle de données (schéma ou JSON).
- Code source (API + interface).
- Diaporama présentant votre projet (voir annexe 1)

6. Critères d'évaluation

Critère	Points
Modélisation document pertinente	2
Fonctionnement complet du CRUD API	5
Filtres et tris bien implémentés	2
Gestion des sous-tâches et commentaires	2
Interface web fonctionnelle	4
Qualité du code (organisation, clarté)	1

Diaporama	4
Total	20

7. Annexe 1 : contenu du diaporama (8 diapositives au minimum)

Diapo 1 – Page de garde

- Titre du projet
- Noms des participants
- Technologies utilisées : MongoDB, Node.js, Express, HTML/CSS/JS
- Lien Git du projet (obligatoire)

Diapo 2 – Modélisation (MongoDB)

Présenter :

- La structure finale du document Task (version JSON simplifiée).
- Le rôle des champs principaux :
  - titre, description, dateCreation, echeance, statut, priorite, categorie, etc.
- Comment vous avez modélisé :
  - les **sous-tâches**
  - les **commentaires**
  - l'**historique des modifications** (si implémenté)
- Justification de vos choix de structure.

Diapo 3 – Architecture générale du projet

Présenter l'organisation du code :

- Structure des dossiers (exemple : server.js, routes/, controllers/, models/).
- Rôle des composants (API REST, modèle MongoDB, interface Web).
- Gestion des erreurs (ex: try/catch, codes HTTP).
- Utilisation éventuelle de middlewares.

Diapo 4 – CRUD et API REST

Présenter :

- L'ensemble des routes implémentées :
  - GET /tasks
  - GET /tasks/:id
  - POST /tasks
  - PUT /tasks/:id
  - DELETE /tasks/:id
- Exemple d'une route commentée ou expliquée (une seule).
- Explication des choix techniques.
- Problèmes rencontrés et solutions.

Diapo 5 – Filtres, tris et paramètres de requête

Présenter les fonctionnalités développées :

- Filtres disponibles (statut, priorité, catégorie, étiquette, date avant/après, texte libre q).
- Options de tri (tri + ordre).

- Exemple concret d'appel d'API :

/tasks?statut=en cours&priorite=haute&tri=echeance&ordre=desc

- Explication du fonctionnement technique (query params → requêtes MongoDB).

Diapo 6 – Interface Web

Présenter clairement :

- Captures d'écran principales
- Actions possibles :
    - affichage de la liste
    - consultation du détail
    - gestion des sous-tâches
    - gestion des commentaires
    - création, modification, suppression
    - filtrage / tri

Diapo 7 – Fonctionnalités avancées / Gestion de projet

Présenter :

- Fonctionnalités bonus (historique des modifications, design particulier, etc.).
- Gestion de projet :
    - organisation du groupe
    - répartition des tâches
    - utilisation de Git
- Méthodologie (sprints, itérations, tests, etc.).

Diapo 8 – Conclusion

Présenter :

- Difficultés rencontrées
- Ce que vous avez appris
- Limites actuelles
- Améliorations possibles

Format attendu

- **Document de type professionnel**
- Style clair et lisible
- Illustrations encouragées
- Lien Git obligatoire