TP 1

Objectif

Créer une application de gestion de tâches collaboratives avec une architecture N-tier.

Fonctionnalités attendues

Gestion des Tâches :

1. **Affichage des Tâches :**
   - Liste des tâches sur la page d'accueil.
   - Affichage des détails de chaque tâche (nom, description, statut, date de création, etc.).
2. **Création de Tâches :**
   - Formulaire permettant la création d'une nouvelle tâche avec les détails appropriés. Possibilité de télécharger des fichiers en tant que pièces jointes pour chaque tâche
3. **Modification de Tâches :**
   - Possibilité de modifier les détails d'une tâche existante.
4. **Suppression de Tâches :**
   - Fonctionnalité permettant la suppression d'une tâche.
5. **Notifications en Temps Réel :**
   - Réception de notifications en temps réel lors de la création ou de la mise à jour d'une tâche.

Bonus :

- **Authentification Utilisateur :**
  - Système d'authentification pour permettre l'identification des utilisateurs.
- **Interface Intuitive :**
  - Interface utilisateur intuitive pour une expérience utilisateur conviviale.

Structure du Projet

- Front SPA (Choix libre de lib js)
- Back avec MongoDB comme Base de données et Fastify
- Message Queue avec NSQ (ou Nats)
- Stockage des pièces jointes sur disques

Backend

MongoDB

La Doc Fastify/MongoDB
La Doc Docker MongoDB

```
const fastify = require('fastify')()

fastify.register(require('@fastify/mongodb'), {
  // force to close the mongodb connection when app stopped
  // the default value is false
  forceClose: true,

  url: 'mongodb://mongo/mydb'
})

fastify.get('/user/:id', async function (req, reply) {
  // Or this.mongo.client.db('mydb').collection('users')
  const users = this.mongo.db.collection('users')

  // if the id is an ObjectId format, you need to create a new Objec
  const id = new this.mongo.ObjectId(req.params.id)
  try {
    const user = await users.findOne({ id })
    return user
  } catch (err) {
    return err
  }
})

fastify.listen({ port: 3000 }, err => {
  if (err) throw err
})
```

CRUD sur les tâches

1. **Lister toutes les Tâches :**
   - **Méthode HTTP :** GET
   - **Endpoint :** `/api/tasks`
   - **Description :** Récupérer la liste complète de toutes les tâches.
2. **Récupérer une Tâche par ID :**
   - **Méthode HTTP :** GET
   - **Endpoint :** `/api/tasks/:id`
   - **Description :** Récupérer les détails d'une tâche spécifique en utilisant son identifiant.

1. **Créer une Nouvelle Tâche :**
   - **Méthode HTTP :** POST
   - **Endpoint :** `/api/tasks`
   - **Description :** Créer une nouvelle tâche avec les détails fournis dans le corps de la requête.
2. **Modifier une Tâche par ID :**
   - **Méthode HTTP :** PUT
   - **Endpoint :** `/api/tasks/:id`
   - **Description :** Modifier les détails d'une tâche spécifique en utilisant son identifiant.
3. **Supprimer une Tâche par ID :**
   - **Méthode HTTP :** DELETE
   - **Endpoint :** `/api/tasks/:id`
   - **Description :** Supprimer une tâche spécifique en utilisant son identifiant.

CR sur les pièces jointes

6. **Télécharger une Pièce Jointe :**
   - **Méthode HTTP :** GET
   - **Endpoint :** `/api/attachments/:taskId/:attachmentId`
   - **Description :** Télécharger une pièce jointe spécifique associée à une tâche.
7. **Téléverser une Nouvelle Pièce Jointe :**
   - **Méthode HTTP :** POST
   - **Endpoint :** `/api/attachments/:taskId`
   - **Description :** Téléverser une nouvelle pièce jointe pour une tâche spécifique.

Upload de Fichiers

La doc Fastify Multipart

Notifications via WebSocket

6. **Recevoir des mises à jour via Websocket :**
   - **Méthode HTTP :** GET
   - **Endpoint :** `/api/notifications`
   - **Description :** Recevoir des mises à jour via Websocket.

     Chaque maj est envoyé à tous les clients connectés

6. **Manager les identifiants de clients Websocket avec MongoDB:**

```
fastify.websocketServer.clients.forEach((client) => {
  if (client.readyState === fastify.websocketServer.OPEN) {
    client.send('Nouvelle tâche créée !');
  }
});
```

Websocket

La doc
[Fastify/Websocket]

Bonus: Authentification

8. **Inscription d'Utilisateur :**
    - **Méthode HTTP :** POST
    - **Endpoint :** `/api/auth/register`
    - **Description :** Créer un nouveau compte utilisateur.
9. **Connexion d'Utilisateur :**
    - **Méthode HTTP :** POST
    - **Endpoint :** `/api/auth/login`
    - **Description :** Authentifier un utilisateur existant.
10. **Déconnexion d'Utilisateur :**
    - **Méthode HTTP :** POST
    - **Endpoint :** `/api/auth/logout`
    - **Description :** Déconnecter un utilisateur authentifié.
11. **Ajour l’identification d'Utilisateur :**
    - **Endpoints :** `Tous les endpoints pertinents`

Github / Gitlab

git init -b main

git add .

git commit -m "backend"

git remote add origin REMOTE-URL

git push origin main