RAPPORT PROJET NPM

Introduction

Ce rapport détaille le développement d'une application web utilisant Node.js comme plateforme backend, MongoDB comme base de données, EJS comme moteur de Template pour les vues, et VSCode comme environnement de développement intégré. Le projet inclut également des tests unitaires avec Jest et des tests d'intégration avec Postman, ainsi qu'un pipeline CI/CD configuré sur GitHub Actions pour automatiser le processus de déploiement.

I- Analyse du besoin

Il nous est demandé de concevoir une photothèque permettant à l'utilisateur de créer modifier supprimer et mettre à jour ses photos qu'il publie ensuite de les trier par titre et date tout en lui offrant une navigation facile et une expérience utilisateur fluide.

II- Mise en place de l'architecture backend

Pour se faire nous avions définis un modèle album représenté par une variable titre, images qui est un tableau de photos, la date de création et de mise à jour pour une base de donnée noSql mongoDB comme suit :

III- phase de développement (logique backend)

Sur base du modèle nous avons créé :

- **deux contrôleurs** : le premier pour la création, la suppression, la modification et la récupération des albums. Le second pour catégoriser et filtrer toutes les photos de la table albums.

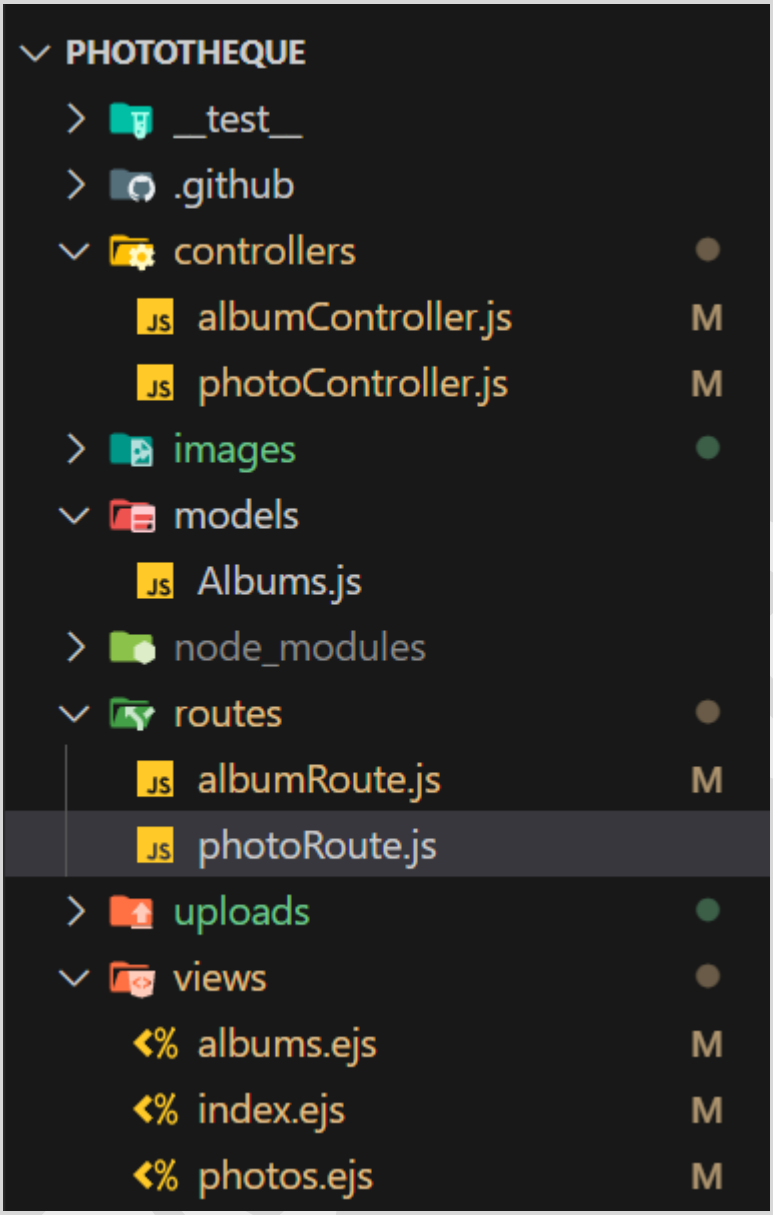
Nous mettons l’accent sur la fonction getPhoto du contrôleur **photoCotrolleur** qui se charger de traiter vraiment les photos des albums et le filtrage faites sur la page des photos.

- deux routes : la première pour les endpoints ou ressource qui concernent les albums et la deuxième pour les ressources de la liste des photos de l’album

- **trois vues** : la première pour la page d’accueil, la seconde pour la page de création, modification et de suppression et la troisième pour la page d'affichage des photos.

IV- phase de développement (frontend)

Nous intégrons bootstrap afin de garantir un design convivial et intuitif, de plus nous intégrons sweetAlert afin d’interagir avec l’utilisateur lors de la création des albums et dataTables pour paginer et rendre agréable la liste des utilisateurs dans un tableau.



V- Tests

Nous utilisons postman pour faire le test de nos api afin de s'assurer que nous utilisons le format JSON et avons une bonne interaction HTTP avec l'api et la librairie jest pour les tests unitaires de façon plus détaillée du code JavaScript écrit comme le cas de notre fonction de création d'album fait dans le projet

VI- problème rencontrés et solutions apportés

1- Gestions des filtres

Problème : Initialisation incorrecte des filtres côtés client sur la page photo

Solution : Ajout d'une fonction JavaScript **updateFilterField ()** pour gérer dynamiquement l'affichage des champs de filtrage en fonction du critère de tri sélectionné.

2- Affichage des messages en l'absence de résultats

Problème : l'utilisateur ne recevait aucun feedback lorsque aucun album ne correspondait aux critères de filtrage

Solution : modification du Template HTML pour inclure une vérification.

(` **% if (album. Length===0 {%>)** et l'affichage d'un message informatif lorsque aucun album n'est trouvé après l'application des filtres.

3- Performance de la base de donnée

Problème : Requêtes lentes à la base de données lorsque l'utilisateur appliquait plusieurs filtres simultanément.

Solution : Utilisation d'indexes mongoDb sur les champs fréquemment filtrés (title , **createdAt)** pour accélérer les requêtes. Cela a significativement amélioré le temps de réponse de l'application.

VII - choix technique

1. **Environnement de Développement**
    - **VSCode :** Choisi pour son support étendu des extensions, son intégration avec Git et ses fonctionnalités avancées d'édition de code.
2. **Backend et API**
    - **Node.js :** Sélectionné pour sa performance côté serveur, sa scalabilité et sa large communauté de développeurs.
    - **Express.js :** Framework web utilisé pour simplifier la gestion des routes, des middlewares et des requêtes HTTP.
3. **Base de Données**
    - **MongoDB :** Base de données NoSQL choisie pour sa flexibilité de schéma, sa capacité à gérer des données non structurées et sa scalabilité horizontale.
4. **Moteur de Template**
    - **EJS (Embedded JavaScript) :** Utilisé pour générer dynamiquement des vues HTML basées sur des données provenant du backend.

5. **Tests**
    - **Jest :** Framework de test unitaire utilisé pour écrire et exécuter des tests JavaScript de manière efficace.
    - **Postman :** Utilisé pour les tests d'intégration, permettant de valider les API et les flux de données.
6. **CI/CD**
    - **GitHub Actions :** Mise en place d'un pipeline CI/CD pour automatiser les tests, la construction et le déploiement de l'application à partir de GitHub.

IMPLÉMENTATION ET DÉVELOPPEMENT

- **Architecture MVC :** L'application est structurée selon le modèle MVC (Modèle-Vue-Contrôleur) pour séparer la logique métier, la présentation et le contrôle des flux de données.
- **Routes et Contrôleurs :** Express.js est utilisé pour définir les routes HTTP et les contrôleurs qui gèrent les requêtes et les réponses.
- **Modèles MongoDB :** Les modèles MongoDB sont définis pour interagir avec la base de données et encapsuler la logique d'accès aux données.
- **Vues EJS :** Les vues sont rendues dynamiquement à l'aide d'EJS, intégrant des données provenant du backend pour générer du contenu HTML.
- **Tests Unitaires :** Jest est utilisé pour tester les fonctions et les composants individuels du backend, assurant la fiabilité et la qualité du code.
- **Tests d'Intégration :** Postman est employé pour tester les API dans un environnement proche de la production, validant les interactions entre les différentes parties de l'application.

CI/CD ET DÉPLOIEMENT AUTOMATISÉ

- **GitHub Actions :** Un fichier CI/CD est configuré sur GitHub, utilisant des workflows pour automatiser les étapes de test, de construction et de déploiement.
- **Pipeline CI/CD :** Le pipeline inclut des étapes telles que les tests unitaires avec Jest, les tests d'intégration avec Postman, la construction de l'application, et le déploiement automatique sur les environnements de staging et de production.

CONCLUSION

Ce rapport démontre l'utilisation efficace de technologies modernes et d'outils avancés pour développer une application web robuste et évolutive. Les choix techniques, y compris l'utilisation de Node.js, MongoDB, EJS, Jest, Postman et GitHub Actions, ont permis de répondre efficacement aux exigences du projet tout en garantissant la qualité et la sécurité de l'application développée.