

Veille technologique Back

Projet **Kanban**

Mériem Barka

4 Décembre 2023



Introduction

Cette introduction présente un projet innovant axé sur l'application Kanban Board, élaborée avec Nest.js. Dans un contexte où la gestion des tâches et des projets est cruciale pour la productivité, cette application propose une approche moderne et intuitive pour visualiser, organiser et suivre les flux de travail. L'utilisation de Nest.js garantit une fondation robuste et évolutive pour le développement du back-end, offrant ainsi une expérience utilisateur fluide et efficace.

Présentation du projet et de son contexte

L'application Kanban Board en Nest.js vise à fournir une plateforme conviviale et flexible pour créer et gérer des tableaux Kanban personnalisés. Dans un environnement professionnel dynamique où la collaboration et la communication sont essentielles, cette application se présente comme un outil central facilitant la coordination des équipes et la surveillance des étapes clés d'un projet. L'association de la simplicité d'utilisation de Nest.js avec la puissance du modèle Kanban promet une expérience utilisateur fluide et agréable.

Objectifs de l'application Kanban Board en Nest.js

L'application Kanban Board développée avec Nest.js vise à fournir une interface conviviale pour la création, l'organisation et la mise à jour intuitives des tâches dans un tableau Kanban. Favorisant la collaboration, elle facilite le partage et les commentaires entre les membres de l'équipe. La flexibilité est une priorité, permettant aux utilisateurs d'ajuster le tableau selon leurs besoins. Grâce à Nest.js pour le back-end, l'application assure une réactivité optimale, offrant ainsi une expérience utilisateur fluide sur différentes plateformes. L'objectif global est de fournir un Kanban Board robuste, sécurisé et scalable, répondant aux exigences de la gestion de projets professionnels.

Technologie utilisée

Le choix du framework backend pour ce projet s'est orienté vers Nest.js, motivé par plusieurs considérations stratégiques.

Explication du choix de Nest.js comme framework backend

Nest.js a été sélectionné en raison de sa modularité, de sa robustesse et de sa capacité à fournir une architecture évolutive pour le développement du backend. Basé sur Node.js, il offre une structure inspirée d'Angular, assurant une base solide pour les applications serveur.

Avantages de Nest.js pour ce projet

Modularité

La structure modulaire de Nest.js facilite la création d'API RESTful efficaces grâce à des modules, des contrôleurs et des services. Cela permet une organisation claire et une maintenance simplifiée du code.


TypeScript

L'intégration transparente avec TypeScript améliore la productivité en offrant des fonctionnalités de typage statique. Cela contribue à la robustesse du code et simplifie la transition pour les développeurs familiarisés avec TypeScript.

Gestion avancée des tâches asynchrones

Nest.js offre une gestion avancée des tâches asynchrones, ce qui est particulièrement bénéfique pour les opérations qui nécessitent un traitement non bloquant, améliorant ainsi les performances de l'application.

Validation des données et sécurité intégrée



Avec Nest.js, la validation des données et les mécanismes de sécurité sont intégrés, renforçant la fiabilité de l'application dans le traitement des informations sensibles.

Communauté active

Nest.js bénéficie d'une communauté dynamique et engagée, assurant un support continu, des mises à jour régulières et l'accès à une variété de ressources. Cela contribue à la stabilité et à la durabilité du projet.

Architecture de l'application

L'architecture de l'application est moderne et modulaire, conçue pour assurer une performance optimale et une évolutivité adaptée aux besoins changeants de l'application Kanban Board. Elle est organisée en plusieurs couches distinctes, chacune ayant un rôle spécifique dans le fonctionnement global de l'application.

Couche Backend :

La communication avec la base de données gère la logique métier de l'application, traitant les requêtes et assurant . Cette couche peut être développée avec un framework adapté aux besoins spécifiques du projet, garantissant sécurité et efficacité des opérations côté serveur.

Base de données :

La persistance des données est assurée par une base de données qui stocke les informations relatives aux tableaux Kanban, aux tâches et aux utilisateurs. Le choix de la base de données dépend des exigences de l'application en termes de volume de données, de performances et de sécurité.

Fonctionnalités principales

Les fonctionnalités principales d'un backend Nest.js pour un tableau Kanban visent à assurer la gestion efficace des données, la sécurité et la logique métier.

API RESTful

- Mise en place d'une API RESTful permettant aux clients frontend de communiquer avec le backend de manière standardisée.

Gestion des Utilisateurs

- Système d'authentification sécurisé pour gérer l'accès des utilisateurs aux tableaux Kanban.
- Création, modification et suppression de comptes utilisateur.

Gestion des Tableaux Kanban

- Création, modification et suppression de tableaux Kanban.
- Attribution des utilisateurs autorisés à accéder à chaque tableau.

Gestion des Colonnes et des Cartes

- API permettant la manipulation des colonnes du tableau Kanban, y compris l'ajout, la modification et la suppression.
- Gestion des cartes, y compris la création, l'édition et la suppression.

Relations Utilisateurs-Cartes


- Attribution et modification des utilisateurs responsables de chaque carte.

Historique des Actions

- Journalisation des actions effectuées sur les cartes et les tableaux, permettant le suivi des modifications.

Notifications en Temps Réel

- Mise en œuvre de mécanismes de notifications en temps réel pour



informer les clients frontend des mises à jour.

Sécurité

- Mise en place de mécanismes de sécurité robustes pour protéger les données sensibles et garantir l'intégrité des opérations.

Validation des Données

- Validation des données entrantes pour assurer leur conformité avec les exigences définies.

Statistiques et Rapports

- Fonctionnalités pour générer des statistiques sur l'utilisation du tableau Kanban, telles que l'avancement du projet.

Gestion des Commentaires et de la Collaboration

- API pour la gestion des commentaires sur les cartes, facilitant la collaboration entre les membres de l'équipe.

Intégration avec d'autres Services

- Possibilité d'intégrer le backend avec d'autres services tiers tels que des outils de gestion de versions, des services de messagerie, etc.

Système de Filtrage et de Recherche

- API permettant aux clients frontend de filtrer les cartes en fonction de critères spécifiques et de réaliser des recherches.

Gestion des Erreurs et des Exceptions

- Mise en place d'une gestion robuste des erreurs pour assurer la stabilité et la fiabilité du backend.


Outils et bibliothèques

L'utilisation de ces outils et bibliothèques vise à optimiser le développement, à garantir la stabilité et la sécurité de l'application, ainsi qu'à fournir une expérience utilisateur agréable et réactive. Chaque choix est guidé par les besoins spécifiques du projet et les avantages offerts par les technologies sélectionnées.

Liste des outils et bibliothèques utilisés

Backend
Node.js
Express.js
MongoDB
Socket.io
JWT (JSON Web Tokens)

Justification de l'utilisation de chaque outil



Nest.js est choisi en raison de sa modularité, de sa structure basée sur des classes, et de son architecture inspirée par Angular. Cela facilite l'organisation du code, la gestion des dépendances, et offre une structure évolutive, idéale pour le développement backend d'un tableau Kanban complexe.

Express.js est utilisé comme moteur HTTP sous-jacent par Nest.js. En simplifiant le processus de développement, Express.js propose des fonctionnalités solides pour la création de routes, la gestion des requêtes HTTP, ainsi que la définition de middleware. L'intégration d'Express.js avec Nest.js constitue une association puissante, renforçant ainsi les capacités de développement backend de manière significative.

MongoDB est privilégiée pour sa nature NoSQL, offrant une flexibilité dans le stockage des données. Avec la structure évolutive des tableaux Kanban, MongoDB permet de stocker des informations variées de manière flexible, ce qui est essentiel pour la gestion des cartes et des tableaux.

Socket.io est utilisé pour faciliter la communication bidirectionnelle en temps réel entre le frontend et le backend. Dans un tableau Kanban, cela améliore la réactivité en permettant des mises à jour instantanées des cartes et des commentaires. L'intégration de Socket.io avec Nest.js offre une gestion propre des événements en temps réel.

JWT (JSON Web Tokens) est choisi pour l'authentification sécurisée des utilisateurs. Il permet le transfert sécurisé d'informations d'identification entre le frontend et le backend. Dans le contexte d'un tableau Kanban, où la sécurité et l'authentification sont cruciales, JWT offre une méthode sécurisée pour la gestion des sessions utilisateur.

Gestion de l'état

Explication de la gestion de l'état dans l'application (Express.js)

Sessions :

Express.js propose une gestion des sessions, permettant de stocker des données d'état spécifiques à un utilisateur entre les requêtes. Ces données sont souvent stockées côté serveur et associées à un identifiant de session unique attribué à chaque utilisateur. Cela permet de conserver des informations telles que l'authentification de l'utilisateur, ses préférences, etc.

Cookies :

Sont souvent utilisés pour stocker de petites quantités de données sur le navigateur de l'utilisateur. Express.js peut être configuré pour générer et gérer des cookies, permettant ainsi de maintenir un état entre les différentes requêtes du même utilisateur.

Middleware :

Express.js utilise des middlewares pour effectuer des opérations spécifiques lors du traitement d'une requête. Certains middlewares peuvent être utilisés pour gérer l'état.

Variables Locales :

Express.js permet également de définir des variables locales, accessibles uniquement dans le contexte d'une requête particulière. Cela peut être utilisé pour transmettre des informations spécifiques à une vue ou à d'autres parties du code dans le cadre d'une requête donnée.

API RESTful :

Dans le cas d'une API RESTful, l'état peut souvent être géré à l'aide de tokens JWT (JSON Web Tokens). Ces tokens sont générés côté serveur, signés, et renvoyés au client, permettant ainsi au client de les inclure dans ses requêtes ultérieures pour maintenir son état d'authentification.

Intégration avec le backend

L'application communique de manière fluide avec le backend en utilisant des requêtes HTTP vers des points d'API définis. Axios, intégré dans l'application, facilite ces communications. Cette approche assure un échange harmonieux de données entre le frontend et le backend, assurant un fonctionnement cohérent et réactif de l'application.

Comment l'application communique-t-elle avec le backend ?

L'application communique avec le backend en utilisant des requêtes HTTP, facilitées par des outils tels qu'Axios intégrés dans l'application. Lorsqu'une action nécessitant des données du backend est déclenchée, comme la récupération de tâches ou la mise à jour d'un état, une requête HTTP est envoyée au serveur backend.

Côté backend, des points d'API sont configurés pour répondre à ces requêtes, exécutant des opérations sur la base de données via des technologies telles que Node.js et Express.js. Ces opérations incluent la lecture et la modification des données. L'utilisation de mécanismes d'authentification comme JSON Web Tokens (JWT) assure la sécurité des échanges entre le frontend et le backend.

Une fois les données traitées côté serveur, la réponse est renvoyée à l'application frontend, mettant à jour l'interface utilisateur en conséquence. Cette intégration fluide entre le frontend et le backend garantit un fonctionnement cohérent et réactif de l'application.

Présentation des API utilisées :

Pour le Kanban consiste à décrire les points d'API utilisés par l'application pour communiquer avec le backend. Ces points d'API, également appelés endpoints, facilitent des opérations telles que la récupération, la création, la mise à jour et la suppression de données.

Voici la structure de l' API pour un Kanban :

Récupération des Tâches :

- Endpoint : [/api/tasks](#)
- Description : Récupère la liste complète des tâches du Kanban.

Création d'une Nouvelle Tâche :

- Endpoint : [/api/tasks/create](#)
- Description : Crée une nouvelle tâche dans le Kanban.

Mise à Jour de l'État d'une Tâche :

- Endpoint : [/api/tasks/update/:taskId](#)
- Description : Met à jour l'état d'une tâche spécifique dans le Kanban.

Suppression d'une Tâche :

- Endpoint : [/api/tasks/delete/:taskId](#)
- Description : Supprime une tâche spécifique du Kanban.

Récupération des Utilisateurs :

- Endpoint : [/api/users](#)
- Description : Récupère la liste des utilisateurs associés au Kanban.

Authentification d'un Utilisateur :

- Endpoint : [/api/authenticate](#)
- Description : Vérifie et authentifie les utilisateurs avant d'accéder au Kanban.

Mise à Jour du Tableau Kanban :

- Endpoint : [/api/kanban/update](#)
- Description : Met à jour la configuration du tableau Kanban



Performances

Mesures prises pour garantir de bonnes performances :

Optimisation des Requêtes à la Base de Données :

L'optimisation des requêtes à la base de données est cruciale. Assurez-vous d'utiliser des index, de limiter le nombre de requêtes, et d'adopter des modèles de données efficaces. Une base de données bien optimisée contribue significativement aux performances globales du backend.

Utilisation d'un Framework Performant :

Le choix d'un framework performant est essentiel. Frameworks tels que Nest.js, basé sur Node.js, sont conçus pour des performances optimales. Assurez-vous que le framework utilisé répond aux exigences de votre application et offre une gestion efficace des opérations asynchrones.

Mise en Cache Stratégique :

La mise en cache stratégique est une mesure puissante. En stockant temporairement des résultats de requêtes fréquentes, vous réduisez la charge sur le backend et améliorez la réactivité de l'application en fournissant des données rapidement accessibles.

Surveillance et Optimisation Continue :

La surveillance constante des performances et l'optimisation continue sont essentielles. Utilisez des outils de surveillance pour identifier les points de congestion, les temps de latence, et les problèmes potentiels. En optimisant en permanence en fonction des résultats, vous garantisiez un backend performant et réactif.



Perspectives d'avenir

Perspectives d'avenir côté Back-end

Évolutivité Horizontale :

L'évolutivité horizontale est cruciale pour faire face à une croissance potentielle du nombre d'utilisateurs. Assurer une architecture qui peut facilement s'étendre horizontalement permettra de maintenir des performances élevées même avec une augmentation de la charge.

Adoption de Microservices :

L'adoption de l'architecture de microservices offre une flexibilité accrue. La décomposition de l'application en services indépendants facilite la maintenance et permet de mettre à l'échelle des composants spécifiques en fonction des besoins.

Intégration d'Intelligence Artificielle :

L'intégration de l'intelligence artificielle peut apporter des avantages significatifs en améliorant les fonctionnalités de gestion de projet. L'utilisation de l'IA pour la prédiction des délais et l'optimisation des flux de travail peut augmenter l'efficacité de l'ensemble du processus.

Sécurité Renforcée :

La sécurité renforcée est une priorité, surtout lorsqu'il s'agit de données sensibles de gestion de projet. Mettre en place des protocoles de sécurité avancés, une gestion des identités et des accès rigoureuse, ainsi que des audits réguliers, assurent la protection des informations.



Fonctionnalités Futures Envisagées

Automatisation des Flux de Travail :

L'automatisation des flux de travail permet une gestion plus efficace des tâches et des projets. Elle contribue à réduire la charge manuelle, minimiser les erreurs humaines, et accélérer la progression des projets.

Analyse Prédictive :

L'analyse prédictive offre une vision proactive des tendances et des obstacles potentiels. Cela permet aux équipes de prendre des décisions éclairées, d'anticiper les retards et d'optimiser les processus de manière continue.

Intégration de Plugins et d'Extensions :

La possibilité d'intégrer des plugins et des extensions permet une personnalisation avancée, offrant aux utilisateurs la flexibilité d'adapter l'application à leurs besoins spécifiques. Cela favorise l'adoption et la satisfaction des utilisateurs.

Gestion Fine des Autorisations :

Une gestion fine des autorisations assure une sécurité renforcée en permettant aux administrateurs de définir des niveaux d'accès précis. Cela garantit que chaque utilisateur a accès aux informations nécessaires en fonction de son rôle dans le projet.



Améliorations possibles

Intégration avec des Outils de Communication (Slack, Microsoft Teams) :

Facilite la communication en temps réel et permet aux membres de l'équipe de rester informés des mises à jour du projet sans quitter leur plateforme de communication préférée.

Connectivité avec des Outils de Gestion de Code Source (GitHub, GitLab) :

Permet d'associer directement des tâches du projet aux branches, commits et pull requests correspondants, offrant une visibilité complète sur l'avancement du développement.

Interopérabilité avec des Outils de Planification (Trello, Asana) :

Simplifie la synchronisation des données de planification, permettant une gestion intégrée des tâches et des délais, tout en offrant une vue globale du projet.

Connectivité avec des Outils de Gestion des Problèmes (Jira, Redmine) :

Assure une traçabilité complète des problèmes, des demandes et des incidents associés à chaque tâche, favorisant une gestion cohérente des problèmes rencontrés.

Intégration avec des Outils de Surveillance des Performances (New Relic, Datadog) :

Permet une surveillance proactive des performances de l'application, en identifiant rapidement les goulets d'étranglement et les problèmes potentiels pour maintenir une expérience utilisateur optimale.

Conclusion

- **Résumé des points clés.**

La veille technologique back-end met en avant des choix stratégiques pour le développement d'applications Kanban robustes. Parmi les technologies émergentes, Nest.js est préféré pour sa modularité et son évolutivité, utilisant TypeScript pour une structure organisée. Express.js reste une option fiable avec des fonctionnalités robustes pour la gestion HTTP. MongoDB, en tant que base de données NoSQL, offre une flexibilité adaptée aux données non structurées des tableaux Kanban. Socket.io garantit une communication en temps réel, essentielle pour la réactivité. Enfin, JSON Web Tokens (JWT) assure une authentification sécurisée. Ces choix témoignent d'une recherche d'efficacité, de sécurité, et de réactivité dans le développement backend.