Bienvenue à notre nouveau développeur !

Nous sommes ravis de vous accueillir pour ce projet de gestion de tickets avec React.js. C'est une excellente opportunité d'apprentissage et de croissance. N'hésitez pas à poser des questions.

Sujet de Stage : Développement d'un Système de Gestion de Tickets avec React.js et .NET

Contexte : Notre entreprise cherche à améliorer son processus de gestion des demandes clients en développant un système de gestion de tickets moderne et efficace. Ce projet servira également de formation pratique pour nos nouveaux développeurs, les familiarisant avec React.js, une bibliothèque JavaScript populaire pour la création d'interfaces utilisateur, et avec .NET pour le développement du backend.

Objectif : Concevoir et développer une application web de gestion de tickets, similaire à une liste de tâches, en utilisant React.js. Cette application permettra aux utilisateurs de créer, afficher, modifier et supprimer des tickets, ainsi que de suivre leur statut.

Fonctionnalités principales :

Frontend (React.js)

1. Création de tickets avec des champs tels que titre, description, priorité, date limite, Référence, et statut.
2. Affichage des tickets dans une liste organisée avec pagination.
3. Modification et mise à jour des informations des tickets via des formulaires réactifs.
4. Suppression de tickets avec confirmation avant suppression.
5. Filtrage des tickets par statut (en attente, en cours, résolu) et utilisateur assigné.
6. Tri des tickets par priorité, date limite, ou date de création.
7. Recherche de tickets par mot-clé dans le titre ou la description.
8. Interface utilisateur intuitive et responsive, adaptée aux appareils mobiles.
9. Animations légères pour améliorer l'expérience utilisateur.
10. Gestion de l'état de l'application avec React Hooks ou Redux.

Backend (.NET)

1. Implémentation de l'API RESTful pour les opérations CRUD (Créer, Lire, Mettre à jour, Supprimer) des tickets.
2. Persistance des données avec Entity Framework Core et une base de données SQL Server.
3. Documentation de l'API avec Swagger pour faciliter le développement frontend.
4. Déploiement du backend sur un serveur cloud ou un environnement local.

Tâches :

Frontend (React.js)

1. Configurer l'environnement de développement React.js :
    a. Installer Node.js et npm.
    b. Créer un nouveau projet React en utilisant Create React App.
    c. Installer les dépendances nécessaires (e.g., React Router, Axios).
2. Concevoir l'architecture de l'application frontend et ses composants :
    a. Définir la structure des dossiers (e.g., components, pages, services).
    b. Planifier les principaux composants de l'application (e.g., TicketList, TicketForm, TicketDetail).
    c. Créer des maquettes filaires pour les principales pages de l'application.
3. Créer une interface utilisateur intuitive et réactive avec React.js :
    a. Développer les composants de base (e.g., en-tête, pied de page, navigation).
    b. Créer les formulaires pour la création et la modification des tickets.
    c. Assurer la compatibilité mobile (responsive design) en utilisant CSS ou une bibliothèque comme Bootstrap.
4. Implémenter les fonctionnalités CRUD pour les tickets dans le frontend :
    a. Créer des composants pour afficher la liste des tickets, les détails d'un ticket, et les formulaires d'ajout/modification.
    b. Utiliser Axios pour interagir avec l'API (en simulant les requêtes API pour le moment).
    c. Gérer l'état des formulaires et la validation des champs avant soumission.
5. Implémenter le filtrage, le tri, et la recherche de tickets côté frontend :
    a. Ajouter des options de filtrage (e.g., par statut, priorité) au composant TicketList.
    b. Implémenter la fonctionnalité de tri par date, priorité, ou autre critère pertinent.
    c. Mettre en place un champ de recherche pour filtrer les tickets par mot-clé.
6. Gérer l'état de l'application avec React Hooks ou Redux :
    a. Utiliser les Hooks comme useState, useEffect pour gérer l'état local.
    b. Si nécessaire, intégrer Redux pour une gestion centralisée de l'état, surtout pour des fonctionnalités comme le filtrage et la recherche.

Backend (.NET)

1. Configurer l'environnement de développement .NET pour le backend :
    a. Installer le SDK .NET et Visual Studio ou Visual Studio Code.
    b. Créer un projet ASP.NET Core Web API.
    c. Configurer les dépendances .
2. Concevoir l'architecture de l'API backend :
    a. Définir les modèles de données pour les tickets.
    b. Concevoir la structure de la base de données avec Entity Framework Core.
    c. Planifier les endpoints de l'API pour correspondre aux opérations CRUD du frontend.
3. Implémenter les fonctionnalités CRUD pour les tickets via l'API .NET :
    a. Créer les contrôleurs et les services pour gérer les requêtes HTTP (GET, POST, PUT, DELETE).
    b. Implémenter la logique métier pour créer, lire, mettre à jour, et supprimer des tickets.
    c. S'assurer que l'API renvoie des réponses cohérentes et bien formatées.

4. Persister les données avec Entity Framework et une base de données SQL Server :
    a. Configurer la connexion à une base de données SQL Server.
    b. Créer et appliquer des migrations pour structurer la base de données.
    c. S'assurer que les données des tickets sont bien stockées et récupérées depuis la base de données.
5. Documenter le code, l'API, et le processus de développement :
    a. Rédiger une documentation pour chaque endpoint de l'API avec Swagger.
    b. Commenter le code pour expliquer les principales parties de la logique.
    c. Documenter le processus d'installation et d'utilisation de l'application.

Livrables attendus :

1. Code source de l'application sur un dépôt Git
2. Documentation technique du projet
3. Guide d'utilisation de l'application

Bon courage !