Développement d'une Mini Plateforme de Gestion de Projets Collaboratifs

Développement d'une Mini Plateforme de Gestion de Projets Collaboratifs

Test Technique

Présenté par :

TOSSA Fortuné Wladimir

Date de soumission :

Jeudi, 19 Septembre 2024

Introduction

La gestion de projets est une capacité indispensable au succès de toute organisation ; il est donc nécessaire voire indispensable de disposer d'outils efficaces permettant de gérer les projets en eux-mêmes, les tâches en lesquels ils sont subdivisés mais aussi et surtout les membres des équipes assignées à la réalisation de ces projets.

Ce projet vise donc à répondre à ce besoin en fournissant une application web de gestion de projet nommée **SynergyFlow**. SynergyFlow est conçue pour permettre la création des projets, la planification, l'attribution et le suivi des tâches relatives à ces projets, tout en garantissant une facilité d'utilisation. Elle permet aussi la collaboration en temps réel grâce à un chat intégré.

Ce document détaille les aspects techniques de ce projets, les différentes fonctionnalités ainsi que les procédures d'installation et d'utilisation ; ceci afin de garantir la simplicité de prise en main à tout utilisateur.

1 Architecture du système

L'application a été développé suivant une architecture client-serveur garantissant ainsi le contrôle des requêtes destinées à la base de données.

Pour la constitution de cette architecture, **Django** à travers son framework **REST** à été utilisé pour la création du backend. Cela a permis une structuration optimale garantissant une séparation des modules et des fonctionnalités et facilitant la maintenance du code.
Pour le frontend, le framework **React** a été utilisé, couplée à la librairie **Redux**. Ces choix ont garanti une UI/UX fluide et simple grâce à une gestion optimale de l'état globale de l'application, offerte par Redux.
En ce qui concerne la base de données, **Postgresql** a été utilisée en raison du fait que les données à gérer suivent une structure donnée.

2 Fonctionnalités

- **Gestion des utilisateurs et authentification**
  Le système est accessible par 2 types d'utilisateurs : l'**Admin** et le **Membre**.
  L'Admin a tous les droits sur le système. Il peut créer effectuer des tâches comme la création des Projets, des Tâches et des autres membres de l'équipe.
  Le Membre quant lui ne peut que modifier les tâches qui lui sont assignées.
  L'authentification est assurée grâce à un token JWT fournie par le backend à la connexion de l'utilisateur et valable pour 24 heures.
- **Gestion des projets et des tâches**
  L'Admin a la possibilité de créer, de modifier et de supprimer les projets et les tâches.
  À la création d'une tâche, l'admin doit assigner au moins un membre à la tâche. Cette valeur pourra être modifier ultérieurement.
  L'application facilite le suivi des tâches en offrant des filtres et des options de tri par statut, date d'échéances et priorité.

- **Collaboration en temps réel**
  SynergyFlow met à la disposition des membres d'une équipe, un espace collaboratif spécifique à chaque projet en cours. Cela permet à l'équipe de garder un sujet de discussions unique pour une meilleur productivité.

3 API

L'utilisation de Django REST a permis de concevoir une API robuste avec des endpoints complet exigeant un format spécifique de données.
Pour chacune des ressources suivantes, les requêtes POST, PUT, GET et DELETE ont été automatiquement implémentées, accessibles par la route `/api/project/<ressource-name>/` pour les requêtes POST, GET et `/api/project/<ressource-name/<resource-id>/` pour les requêtes GET, PUT et DELETE :

- project ;
- tasks ;
- custom_user ;
- notification.

Le chat étant une fonctionnalité nécessitant des configurations spécifiques, elle a été implémentée dans une application : `chat`. La route d'accès suis la même structure que l'application `project` et le websocket (permettant l'envoie et la réception des messages en temps réel) est accessible à la route `/ws/chat/project_id/`.

En ce qui concerne l'authentification, l'API reçoit une donnée ayant la structure `{'username': 'user_username or email', 'password': 'user_password'}`. Une fois le mot de passe validé, l'API renvoie une réponse ayant pour structure

```
{
'id': user.id,
'username': user.username,
'email': user.email,
'firstname': user.first_name,
'lastname': user.last_name,
'token': token,
'refresh': refresh,
'type': user.user_type
} .
```

Le token est sauvegardé au niveau du frontend et doit être soumis à chaque prochaine requête sinon une erreur `401` est renvoyée.

4 Tests et validation

Afin de garantirs le bon fonctionnement de l'application, des tests unitaires ont été écrits au niveau du backend. Les fonctionnalités testées sont :

- La gestion des projets ;
- la gestion des tâches ;

- la gestion de l'authentification.

Ces fonctionnalités ont été déclinés en différents cas d'utilisations et testées afin de garantir que l'API réagisse comme il devrait le faire.

4.1 Gestion des projets

Les cas testés sont :

- Un admin peut lister les projets : **testé et validé**
- Un membre peut lister les projets : **testé et validé**
- Un admin peut créer un projet : **testé et validé**
- Un membre ne peut pas créer un projet : **testé et validé**
- Un admin peut consulter les détails d'un projet : **testé et validé**
- Un membre peut consulter les détails d'un projet : **testé et validé**
- Un admin peut mettre à jour un projet : **testé et validé**
- Un membre ne peut pas mettre à jour un projet : **testé et validé**
- Un admin peut supprimer un projet : **testé et validé**
- Un membre ne peut pas supprimer un projet : **testé et validé**

4.2 Gestion des tâches

Voici les cas testés :

- Un admin peut lister les tâches : **testé et validé**
- Un membre peut lister les tâches : **testé et validé**
- Un admin peut créer une tâche : **testé et validé**
- Un membre ne peut pas créer une tâche : **testé et validé**
- Un admin peut consulter les détails d'une tâche : **testé et validé**
- Un assignee (membre à qui la tâche est assignée) peut consulter les détails d'une tâche : **testé et validé**
- Un admin peut mettre à jour une tâche : **testé et validé**
- Un assignee peut mettre à jour une tâche : **testé et validé**
- Un membre ne peut pas mettre à jour une tâche : **testé et validé**
- Un admin peut supprimer une tâche : **testé et validé**
- Un membre ne peut pas supprimer une tâche : **testé et validé**

4.3 Gestion de l'authentification

Voici les cas testés :

- Connexion avec nom d'utilisateur réussie : **testé et validé**
- Connexion avec email réussie : **testé et validé**
- Connexion avec mot de passe incorrect : **testé et validé**
- Connexion avec un utilisateur inexistant : **testé et validé**

Conclusion

Ce projet assure une gestion efficace des projets et des tâches, avec des contrôles d'accès bien définis pour les administrateurs et les membres. Grâce à l'utilisation de Django, React, et de websockets pour le chat en temps réel, il permet une collaboration fluide et sécurisée. Les tests valident la fiabilité du système, et l'architecture permet d'ajouter facilement de nouvelles fonctionnalités pour répondre aux besoins futurs.