



Rapport - Project Management Tool

MPMT

**Réalisation et Industrialisation d'une Application
de Gestion de Projets Collaboratifs : Project
Management Tool (MPMT)**



ZANTOUR Wael

Mastère 2 Expert en Ingénierie Logicielle

2023-2024

Project Management Tool

Objet du document :

Ce document a pour but de présenter les différentes étapes de réalisation du projet PMT, en couvrant :

- La conception de l'architecture logicielle et de la base de données.
- Le développement des fonctionnalités backend et frontend.
- Les tests automatisés pour garantir la qualité du code.
- L'industrialisation avec la conteneurisation et l'intégration continue.

Il fournit également une documentation technique complète pour assurer la pérennité et l'évolutivité du projet.

Historique des révisions :

Numéro de version	Auteur	Description	Date de modification
1.0	ZANTOUR Wael	Livraison initiale	21/12/2024

Lien vers le repository du code source :

<https://github.com/wa2l99/project-management-tool>

Tables des matières :

1. Introduction	4
1.1. Contexte	4
1.2. Objectifs du projet	5
1.3. Portée et livrables	5
1.4. Résumé des fonctionnalités clés	6
1.5. Structure générales du rapport	7
2. Phase 1 - Conception	7
2.1. Introduction	7
2.2. Modélisation des entités clés	7
2.3. Schéma relationnel de la base de données	7
2.4. Création et initialisation de la base de données	8
2.5. Conclusion	12
3. Phase 2 - Développement	12
3.1. Introduction	12
3.2. Choix Techniques et Architecture	12
3.2.1. Architecture Monorepo	12
3.2.2. Technologies Utilisées	13
❖ Frontend : project-management-frontend/	14
❖ Documentation	14
❖ Industrialisation	14
3.2.3. Documentation API avec Swagger/OpenAPI	15
3.3. Satisfaction des User Stories	15
3.3.1. Inscription et Connexion des Utilisateurs	16
3.3.2. Création et Gestion des Projets	17
3.3.2.1. User Story : Création d'un projet	17
3.3.2.2. User Story : Invitation des membres	22
3.3.2.3. User Story : Attribution des rôles	26
3.3.3. Création et Gestion des Tâches	28
3.3.3.1. User Story : Création d'une tâche	28
3.3.3.2. User Story : Attribution d'une tâche	30
3.3.3.3. User Story : Mise à jour d'une tâche	31
3.3.3.4. User Story : Visualisation d'une tâche unitaire	33
3.3.3.5. User Story : Notifications par e-mail	33
3.3.3.6. User Story : Notifications par e-mail	34
3.3.3.7. User Story : Suivi de l'historique des tâches	35
3.4. Conclusion	36
4. Phase 3 - Tests et Validations	37

Project Management Tool

4.1. Introduction	37
4.2. Stratégie de Test Adoptée	37
4.3. Tests API avec Postman	37
4.4. Couverture de Code	40
4.4.1. Couverture côté Frontend	40
4.4.2. Couverture côté Backend	41
4.4.3. Analyse Générale	42
4.5. Automatisation des Tests	42
4.5.1. Pipeline Frontend	43
4.5.2. Pipeline Backend	43
4.6. Conclusion	44
5. Phase 4 - Industrialisation et Déploiement	44
5.1. Introduction	44
5.2. Dockerization	45
5.2.1. Dockerization du Backend	45
5.2.2. Dockerization du Frontend	45
5.3. Orchestration avec Docker Compose	46
5.4. Pipelines CI/CD	48
5.4.1. Backend	48
5.4.2. Frontend	49
5.4.3. Déploiement et Gestion des Images Docker sur Docker Hub	51
5.5. Conclusion :	51
6. Conclusion et Évaluation	52
6.1. Conclusion	52
6.2. Évaluation de la Conformité à la Demande	52

1. Introduction

1.1. Contexte

Le projet **PMT (Project Management Tool)** a été initié par l'entreprise **Code Solutions**. L'objectif est de concevoir une application de gestion de projets collaboratifs destinée aux équipes de développement, afin de simplifier la planification, le suivi et la collaboration autour des tâches et des projets.

1.2. Objectifs du projet

L'application **PMT** a été développée pour répondre aux objectifs suivants :

1.2.1. Fonctionnels :

- Permettre aux utilisateurs de s'inscrire et de s'authentifier.
- Offrir une gestion des projets (création, description, gestion des membres).
- Faciliter la gestion des tâches (création, suivi, mise à jour, assignation).
- Fournir des notifications par e-mail et un historique des modifications pour une traçabilité optimale.

1.2.2. Techniques :

- Implémentation d'une architecture modulaire et évolutive basée sur :
 - **Angular** pour le frontend
 - **Spring Boot** pour le backend.
 - **PostgreSQL** comme base de données relationnelle.
- Respecter les bonnes pratiques de développement logiciel, notamment en écrivant des tests automatisés pour atteindre une couverture minimale de 60 %.
- Automatiser les processus de construction, de tests, et de déploiement avec des outils comme **GitHub Actions**.
- Dockeriser l'application pour simplifier le déploiement.

1.3. Portée et livrables

Le projet est organisé en plusieurs phases clés :

1.3.1. Conception :

- Modélisation des entités principales (utilisateurs, projets, tâches) et de leurs relations.
- Création du schéma de la base de données et du script SQL correspondant.

1.3.2. Développement :

- Développement d'une interface utilisateur moderne et intuitive avec **Angular**.
- Implémentation des fonctionnalités backend avec **Spring Boot** et **PostgreSQL**.

1.3.3. Tests et couverture :

- Rédaction de tests unitaires et d'intégration pour le frontend et le backend.
- Génération de rapports de couverture du code (au moins 60 % des branches et instructions).

1.3.4. Industrialisation :

- Conteneurisation des services frontend et backend avec Docker.
- Mise en œuvre d'une pipeline CI/CD pour automatiser les tests, la construction et le déploiement.
- Documentation complète du processus de déploiement

1.4. Résumé des fonctionnalités clés

L'application PMT propose les fonctionnalités suivantes :

- **Inscription et authentification** : gestion des utilisateurs avec des rôles précis (administrateur, membre, observateur).
- **Gestion des projets** : création de projets avec description, assignation de membres et gestion des rôles.

Project Management Tool

- **Gestion des tâches** : création et suivi des tâches par statut, priorité, et échéance ; visualisation des informations détaillées ; assignation des tâches à des membres spécifiques.
- **Notifications** : envoi d'e-mails lors de l'assignation de tâches.
- **Suivi et traçabilité** : consultation de l'historique des modifications des tâches.

1.5. Structure générales du rapport

Le présent rapport documente les étapes majeures de la conception, du développement, des tests et de l'industrialisation. Il est structuré comme suit :

1. **Conception** : Modélisation des entités et schéma de la base de données.
2. **Développement** : Fonctionnalités clés et technologies utilisées.
3. **Tests et Validations** : Méthodologie et couverture des tests automatisés.
4. **Industrialisation et déploiement** : Dockerisation et mise en œuvre de la pipeline CI/CD.

2. Phase 1 - Conception

2.1. Introduction

La phase de conception a pour objectif de structurer et modéliser le système afin de répondre aux exigences fonctionnelles et techniques du projet PMT. Elle inclut la modélisation des entités, la création du schéma relationnel, et la configuration des scripts nécessaires pour la base de données.

2.2. Modélisation des entités clés

Lors de cette étape, nous avons identifié les entités clés qui représentent les concepts principaux de la plateforme :

- **User** : représente les utilisateurs (administrateurs, membres, observateurs).
- **Role** : définit les rôles attribués aux utilisateurs.
- **Project** : décrit les projets collaboratifs.
- **Task** : représente les tâches associées aux projets.
- **TaskModifiedHistory** : trace les modifications effectuées sur les tâches.
- **ProjectMembers** : relie les utilisateurs aux projets auxquels ils appartiennent.

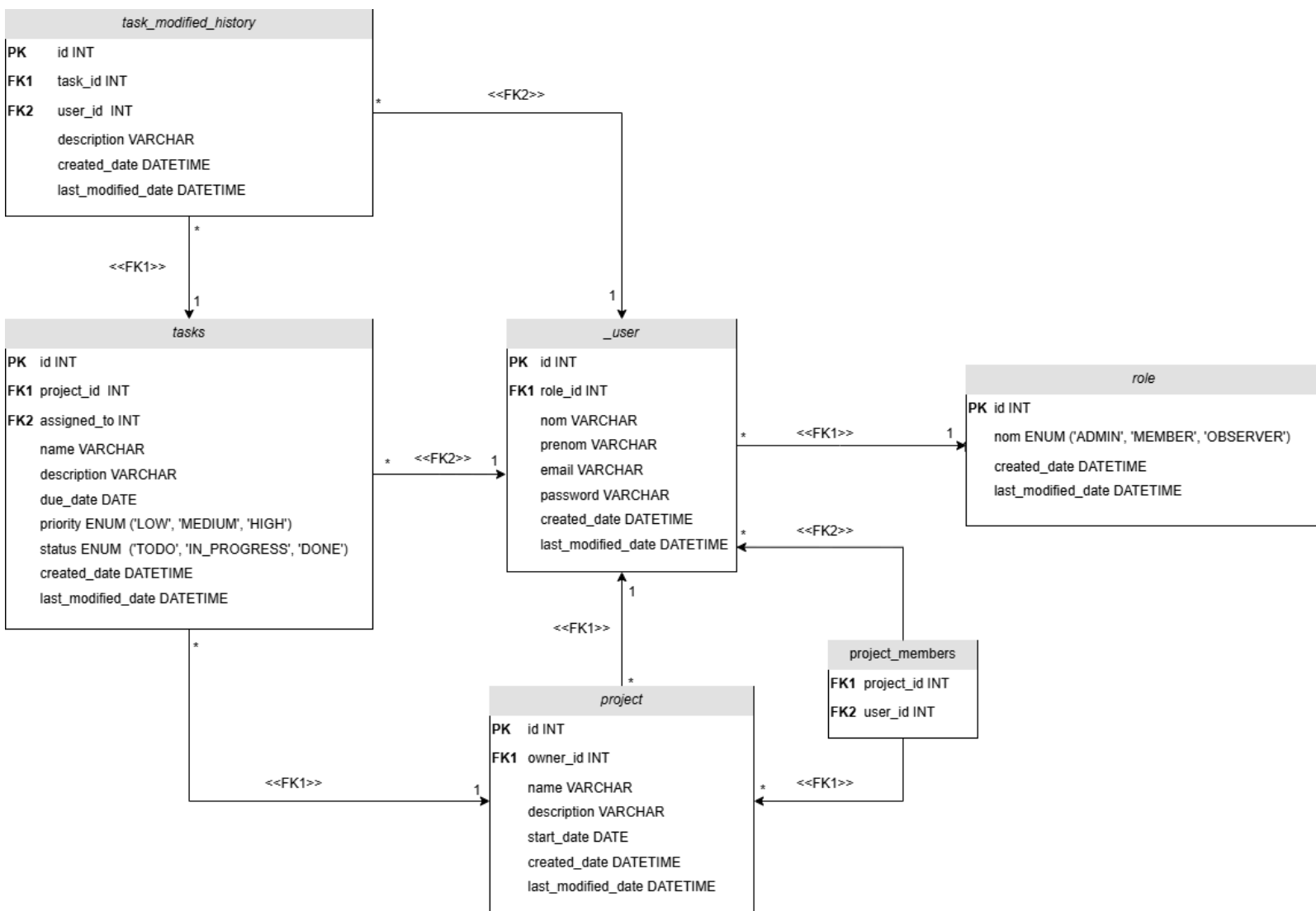
Project Management Tool

Ces entités ont été modélisées en utilisant Java et JPA avec des annotations telles que `@Entity`, `@Table`, et les relations `@OneToMany`, `@ManyToMany`, etc., afin de garantir une structure relationnelle cohérente.

2.3. Schéma relationnel de la base de données

À partir des entités identifiées, un schéma relationnel de la base de données a été conçu. Il inclut les relations et contraintes nécessaires pour assurer l'intégrité des données, notamment les clés primaires, les clés étrangères, et les relations entre les entités.

Voici une représentation graphique du schéma relationnel :



- Schéma de la base de données -

2.4. Création et initialisation de la base de données

2.4.1. Docker Compose pour créer la base de données

Le projet utilise Docker Compose pour automatiser la configuration des conteneurs nécessaires, notamment le backend (Spring Boot) et la base de données PostgreSQL. Lors du lancement du conteneur PostgreSQL, un script initial est exécuté pour créer la base de données (`CREATE DATABASE "pmt";`).

Voici un extrait de la configuration Docker Compose :

```
services:
  postgres:
    container_name: postgres
    image: postgres
    environment:
      POSTGRES_USER: wael
      POSTGRES_PASSWORD: wael
      PGDATA: /var/lib/postgresql/data
      POSTGRES_DB: pmt
    volumes:
      - postgres:/var/lib/postgresql/data
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql
```

Dans cette configuration, le script SQL suivant est placé dans le fichier `init.sql` et exécuté automatiquement par PostgreSQL lors du démarrage :

```
CREATE DATABASE "pmt"
```

→ Cette étape garantit que la base de données nommée **pmt** est disponible pour l'application.

2.4.2. Mécanisme de Migration des Données avec Flyway

Afin d'assurer la gestion de version et l'évolution structurée de la base de données, **Flyway** a été intégré au projet. Flyway facilite la **migration des schémas** et garantit que chaque nouvelle version de la base de données est appliquée de manière automatique lors du démarrage de l'application Spring Boot.

Scripts de Migration avec Flyway

Les scripts SQL sont organisés en deux fichiers principaux dans le répertoire dédié à Flyway `/resources/db/migration/` :

1. V1__create_db.sql : Création des tables

Ce script contient toutes les instructions SQL nécessaires pour initialiser la structure de la base de données.

Exemple des tables à créer :

```
-- 2. Création de la table 'role'

CREATE TABLE IF NOT EXISTS role (
    id SERIAL PRIMARY KEY NOT NULL,
    created_date TIMESTAMP DEFAULT NOW(),
    last_modified_date TIMESTAMP DEFAULT NOW(),
    nom VARCHAR(50) NOT NULL
);

-- 3. Création de la table '_user'

CREATE TABLE IF NOT EXISTS _user (
    id SERIAL PRIMARY KEY NOT NULL,
    created_date TIMESTAMP DEFAULT NOW(),
    last_modified_date TIMESTAMP DEFAULT NOW(),
    email VARCHAR(255) UNIQUE NOT NULL,
    nom VARCHAR(100) NOT NULL,
    prenom VARCHAR(100) NOT NULL,
    password VARCHAR(255) NOT NULL,
    role_id INT NOT NULL,
    FOREIGN KEY (role_id) REFERENCES role(id) ON DELETE CASCADE
);

-- 4. Création de la table 'project'

CREATE TABLE IF NOT EXISTS project (
    id SERIAL PRIMARY KEY NOT NULL,
```

Project Management Tool

```
created_by INT NOT NULL,  
created_date TIMESTAMP DEFAULT NOW(),  
last_modified_by INT NOT NULL,  
last_modified_date TIMESTAMP DEFAULT NOW(),  
description TEXT,  
name VARCHAR(255) NOT NULL,  
start_date DATE NOT NULL,  
owner_id INT NOT NULL,  
FOREIGN KEY (owner_id) REFERENCES _user(id) ON DELETE CASCADE  
);
```

2. V2__Insert_Initial_Data.sql : Insertion des données initiales

Ce script initialise la base avec des données de test, telles que des rôles, des utilisateurs, des projets et des tâches.

Exemple d'insertion :

```
-- Insert data into the 'role' table  
  
INSERT INTO role (id, created_date, last_modified_date, nom)  
VALUES (1, NOW(), NOW(), 'ADMIN'), (2, NOW(), NOW(), 'MEMBER'),  
(3, NOW(), NOW(), 'OBSERVER')  
ON CONFLICT (id) DO NOTHING;  
  
-- Insert data into the '_user' table  
  
INSERT INTO _user (id, created_date, email, last_modified_date, nom,  
password, prenom, role_id)  
VALUES  
(1, NOW(), 'admin@example.com', NOW(), 'Admin', 'hashed_password',  
'John', 1),  
(2, NOW(), 'member1@example.com', NOW(), 'Member1',  
'hashed_password', 'Jane', 2),  
ON CONFLICT (email) DO NOTHING;
```

2.4.3. Avantages de cette approche

Cette combinaison d'approches offre plusieurs avantages :

- **Séparation des responsabilités :**
 - `init.sql` pour la création de la base.
 - `Flyway` pour les migrations et la gestion des versions.
- **Cohérence et Maintenabilité :**
 - Les scripts sont versionnés pour faciliter le suivi des évolutions.

Project Management Tool

- Aucune migration manuelle n'est nécessaire.
- **Automatisation :**
 - La base de données est prête automatiquement au démarrage, réduisant les erreurs d'environnement.
- **Disponibilité des Livrables :**
 - Les fichiers SQL nécessaires sont fournis avec le projet pour permettre une réinitialisation rapide de la base de données.

2.5. Conclusion

Grâce à l'intégration de **Flyway** et **Docker Compose**, la gestion de la base de données est automatisée, structurée et facilement reproductible. Cette approche garantit la cohérence des environnements de développement, de test et de production tout en facilitant l'intégration des nouvelles versions du projet.

L'ensemble des scripts SQL (création des tables, insertion des données initiales) sera inclus dans les livrables du projet :

1. `V1__create_db.sql` pour la création des tables
2. `V2__Insert_Initial_Data.sql` pour l'insertion des données
3. `init.sql` pour l'initialisation de la base via Docker Compose

3. Phase 2 - Développement

3.1. Introduction

La phase de développement du projet **PMT (Project Management Tool)** a consisté à transformer les besoins fonctionnels en une application opérationnelle, tout en respectant les directives techniques fournies. Cette section détaille les choix techniques réalisés pour le **frontend** et le **backend**, y compris l'intégration de **Swagger** et **OpenAPI** pour la documentation des API et met en évidence la satisfaction des **user stories** à travers des **captures d'écran des interfaces**.

3.2. Choix Techniques et Architecture

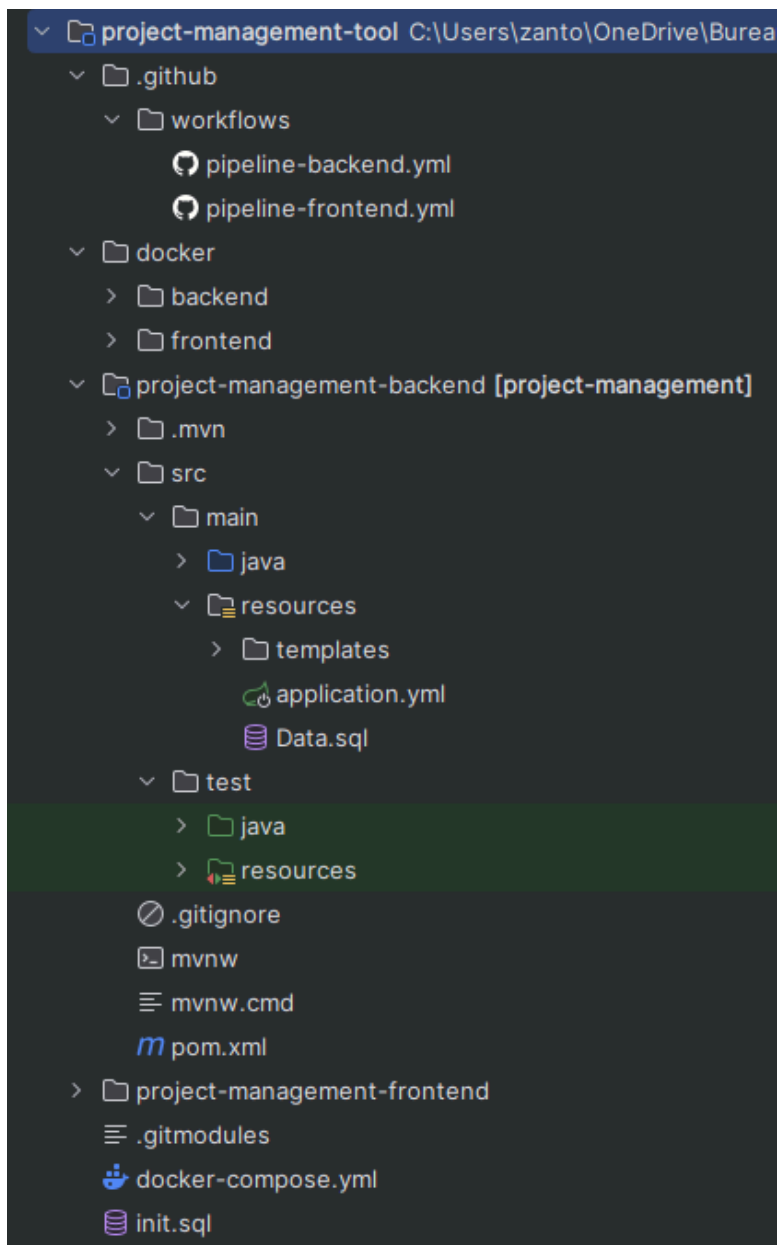
3.2.1. Architecture Monorepo

Le projet utilise une **architecture monorepo** qui regroupe le **backend**, le **frontend** et les configurations de déploiement dans un seul et unique repository Git. Ce choix

Project Management Tool

simplifie la gestion du code source, l'intégration des composants et le processus de déploiement.

Structure du projet :



3.2.2. Technologies Utilisées

❖ Backend : **project-management-backend/**

Project Management Tool

- **Spring Boot** : Implémente les services REST pour gérer les utilisateurs, les projets et les tâches.
- **API REST** : Expose des endpoints sécurisés pour le frontend.
- **Base de données** : PostgreSQL pour la persistance des données.
- **MailDev** : pour l'envoi de notifications par e-mail aux utilisateurs lorsqu'une tâche leur est assignée.
- **Flyway** : pour la gestion des migrations de base de données et l'insertion de données fictives.
- **Scripts SQL** :
 - **init.sql** : Création de la base de données via **Docker**.
 - **Data.sql** : Initialisation des données pour les tests et le développement.
- **Tests** : Mise en place de tests unitaires et d'intégration pour assurer la stabilité et la couverture du code en utilisant **JUnit 5**, **MockMvc** et **Mockito**.
- **Dockerfile** : Containerisation du backend avec Maven pour faciliter l'exécution.

❖ Frontend : **project-management-frontend/**

- **Angular** : Développement des composants et services pour consommer les APIs backend.
- **Design responsive** : Utilisation de **Bootstrap** pour assurer une interface utilisateur ergonomique et réactive.
- **Tests unitaires** : Utilisation de **Jest** pour tester les composants Angular.
- **Dockerfile** : Containerisation du frontend pour une exécution simplifiée.

❖ Documentation

- **Swagger & OpenAPI** : Pour documenter les endpoints de l'API backend.

❖ Industrialisation

- **Conteneurisation** : Utilisation de **Docker** pour conteneuriser le frontend, le backend et la base de données..

Project Management Tool

- **Orchestration** : Docker Compose pour coordonner le déploiement des services.
- **CI/CD** : Mise en place d'une pipeline GitHub Actions pour automatiser la construction, les tests et le déploiement des images Docker..

3.2.3. Documentation API avec Swagger/OpenAPI

Swagger et OpenAPI ont été intégrés pour documenter dynamiquement les endpoints exposés par le backend. Cela facilite la compréhension des API, les tests interactifs et l'intégration par des développeurs tiers.

Accès à la Documentation : <http://localhost:8080/swagger-ui/index.html>

Exemple de documentation des endpoints :

The screenshot shows the Swagger UI for the 'Project Management Tool' API. The top bar includes the Swagger logo, a search bar with '/v3/api-docs', and an 'Explore' button. Below the header, the title 'Open specification - Project Management Tool' is displayed with version '1.0' and 'OAS 3.0' tags. A link to 'v3/api-docs' is provided. The 'Servers' section shows 'http://localhost:8088 - Local ENV' and an 'Authorize' button. The main content area lists the API endpoints under the 'Task' group:

Method	Endpoint	Lock Icon	Dropdown Icon
GET	/api/tasks/{taskId}	🔒	▼
PUT	/api/tasks/{taskId}	🔒	▼
DELETE	/api/tasks/{taskId}	🔒	▼
POST	/api/tasks/{taskId}/assign	🔒	▼
POST	/api/tasks/projectId={projectId}/tasks	🔒	▼
GET	/api/tasks/projectId={projectId}	🔒	▼
GET	/api/tasks/projectId={projectId}/tasksByStatus	🔒	▼
GET	/api/tasks/projectId={projectId}/tasksByPriority	🔒	▼
GET	/api/tasks/my-projects/history	🔒	▼

- Interface Swagger -

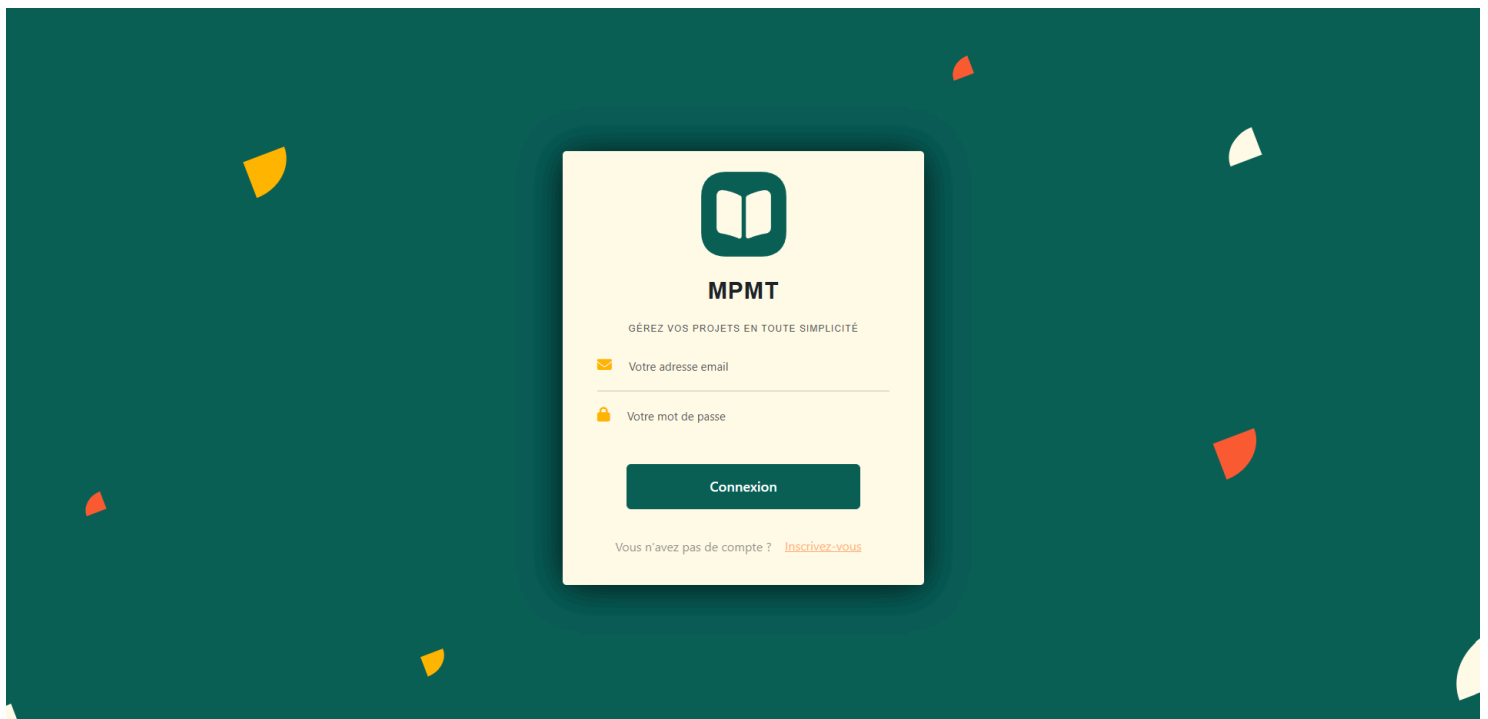
3.3. Satisfaction des User Stories

Le développement a couvert l'ensemble des **user stories** mentionnées dans le cahier des charges. Chaque fonctionnalité est accompagnée d'une interface claire et intuitive. Voici un récapitulatif des principales user stories implémentées avec des captures d'écran pour démonstration.

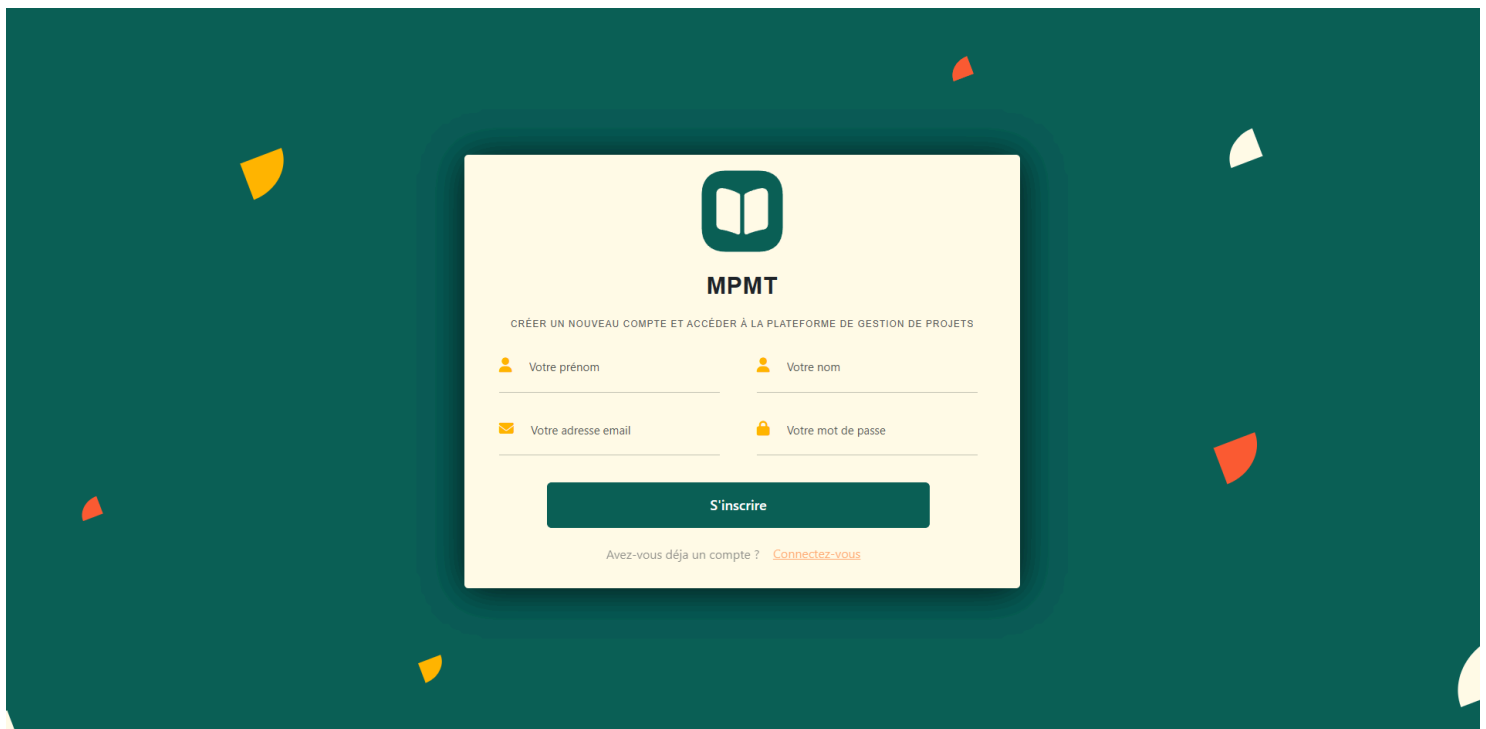
3.3.1. Inscription et Connexion des Utilisateurs

- **User Story :** *"En tant que visiteur, je veux pouvoir m'inscrire avec un nom d'utilisateur, une adresse email et un mot de passe afin d'avoir un compte sur la plateforme."*
- **User Story :** *"En tant qu'inscrit, je veux pouvoir me connecter à la plateforme avec mon adresse e-mail et mon mot de passe afin de pouvoir accéder à mon espace."*

Interface Utilisateur :



- Interface de connexion -



- Interface d'inscription -

→ Les interfaces proposent un formulaire d'inscription et de connexion simple, ergonomique et sécurisé.

3.3.2. Création et Gestion des Projets

3.3.2.1. User Story : Création d'un projet

- **User Story** : "En tant qu'utilisateur, je veux pouvoir créer un nouveau projet avec un nom, une description et une date de début afin d'être un administrateur du projet."

Scénario et Implémentation :

1. Accès conditionnel à la création de projet :

- Seuls les utilisateurs **sans rôle** ou ayant un rôle **Administrateur** peuvent accéder à la fonctionnalité de création de projet.
- Les utilisateurs ayant un rôle **Membre** ou **Observateur** ne voient **pas** le bouton de création de projet, et toute tentative d'accès direct via URL les redirige vers une **page 403 Forbidden**.

Project Management Tool

MPMT.

Créer un nouveau projet

Tableau de bord

Projets

Historique des tâches

John Doe JD

Bienvenue sur le tableau de bord, John 🖐️

NOMBRE DE PROJETS

2

TÂCHES À FAIRE

1

TÂCHES EN COURS

1

TÂCHES TERMINÉES

1

Résumé des tâches

Filtrer par statut

Nom du projet	Nom de la tâche	Priorité	Statut	Date d'échéance	Progression
Project Alpha	Task 1	Faible	À faire	01/02/2024	25%
Project Alpha	Task 2	Moyenne	En cours	10/02/2024	50%
Project Alpha	Task 3	Haute	Terminé	01/03/2024	100%

0 sur 0 < >

Répartition des tâches

À faire (33%)

En cours (33%)

Terminé (33%)

© 2024 - Project Management Tool - MPMT | all right reserved

- Affichage du bouton "Créer un projet" dans la barre latérale pour un utilisateur autorisé -

MPMT.

Tableau de bord

Projets

Historique des tâches

Mike Brown MB

Bienvenue sur le tableau de bord, Mike 🖐️

NOMBRE DE PROJETS

2

TÂCHES À FAIRE

1

TÂCHES EN COURS

1

TÂCHES TERMINÉES

1

Résumé des tâches

Filtrer par statut

Nom du projet	Nom de la tâche	Priorité	Statut	Date d'échéance	Progression
Project Alpha	Task 1	Faible	À faire	01/02/2024	25%
Project Alpha	Task 2	Moyenne	En cours	10/02/2024	50%
Project Alpha	Task 3	Haute	Terminé	01/03/2024	100%

0 sur 0 < >

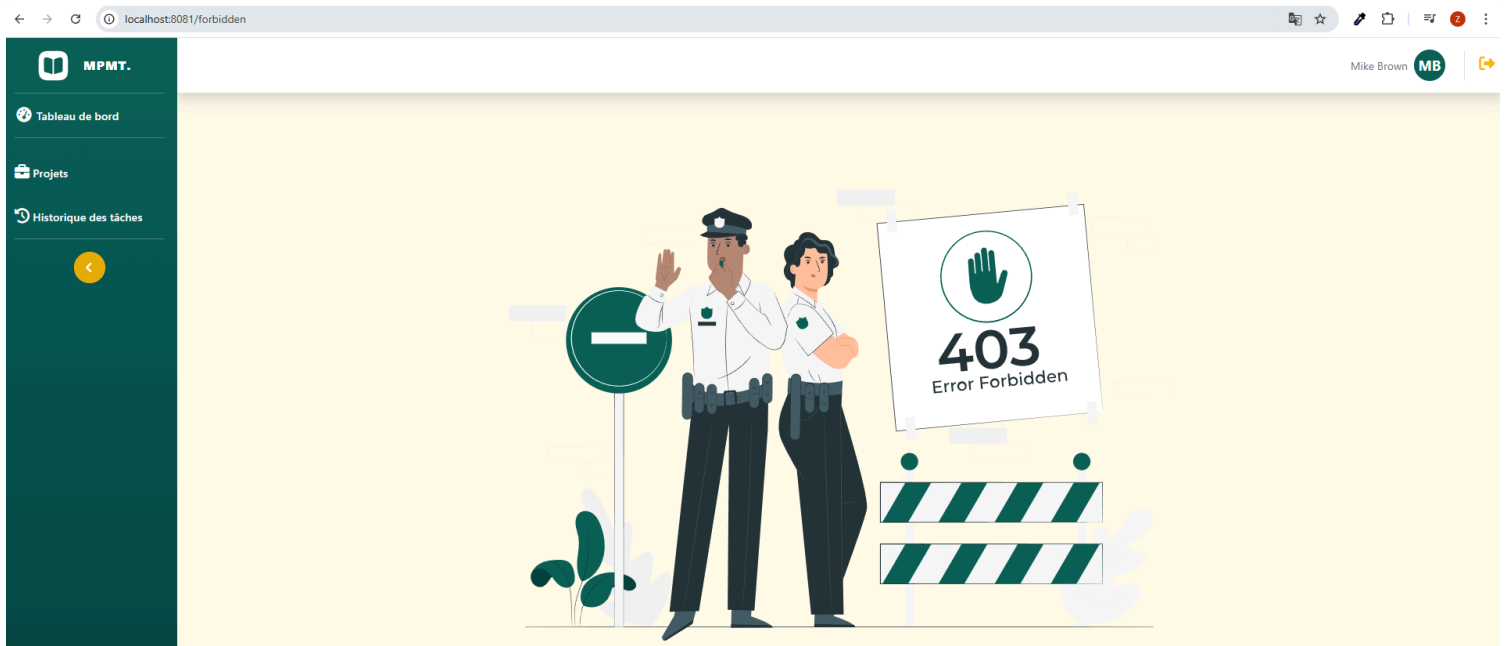
Répartition des tâches

À faire (33%)

En cours (33%)

Terminé (33%)

- Absence du bouton "Créer un projet" pour un utilisateur avec un rôle Observateur -



- Page 403 Forbidden affichée pour un utilisateur non autorisé -

2. Formulaire en étapes (Stepper) :

L'interface utilisateur utilise un **formulaire stepper** divisé en plusieurs étapes pour la création complète d'un projet (avec l'invitation des membres et l'attribution des rôles).

L'étape de création du projet est obligatoire, tandis que les étapes suivantes sont **optionnelles**.

- Étape 1 : Informations de base (obligatoire)

Cette étape permet de collecter les informations clés du projet et d'initier sa création.

Project Management Tool

MPMT.

John Doe JD

Créer un nouveau projet

Tableau de bord

Projets

Historique des tâches

Créer un Nouveau Projet

Renseignez les détails pour créer votre projet et commencez à collaborer.

← Retour

1 Créer un projet 2 Ajouter un membre au projet Optional 3 Assigner un rôle Optional 4 Récapitulatif du projet

Nom du projet*

Description*

Date de début*

Créer

Suivant

- Formulaire Étape 1 : création du projet -

- Étape 2 : Ajout de membres (optionnelle)

Cette étape permet à l'administrateur d'ajouter des membres au projet en saisissant leurs adresses e-mail après la création du projet . L'ajout de membres peut également être réalisé ultérieurement via la **page de gestion des membres** du projet.

Créer un Nouveau Projet

Renseignez les détails pour créer votre projet et commencez à collaborer.

← Retour

✓ Créer un projet (Terminé) 2 Ajouter un membre au projet Optional 3 Assigner un rôle Optional 4 Récapitulatif du projet

Email du membre*

Chercher l'email

wael@gmail.com

test@gmail.com

jane.smith@example.com

john.doe@example.com

mike.brown@example.com

Précédent Suivant

Projet créé avec succès !

© 2024 - Project Management Tool - MPMT | all right reserved

Project Management Tool

Créer un Nouveau Projet

Renseignez les détails pour créer votre projet et commencez à collaborer.

[← Retour](#)

✓ Créer un projet (Terminé) — 2 Ajouter un membre au projet Optional — 3 Assigner un rôle Optional — 4 Récapitulatif du projet

Email du membre*

Inviter membre

Membres

JD

John Doe (john.doe@example.com) – Propriétaire

ZW

Zantour Wael (wael@gmail.com)

Précédent Suivant

- Formulaire Étape 2 pour l'ajout des membres -

- Étape 3 : Attribution des rôles (optionnelle)

Cette étape offre la possibilité d'attribuer des rôles (**Administrateur**, **Membre**, **Observateur**) aux membres ajoutés précédemment.

Créer un Nouveau Projet

Renseignez les détails pour créer votre projet et commencez à collaborer.

[← Retour](#)

Créer un Nouveau Projet

Renseignez les détails pour créer votre projet et commencez à collaborer.

[← Retour](#)

✓ Créer un projet (Terminé) — 2 Ajouter un membre au projet Optional — 3 Assigner un rôle Optional — 4 Récapitulatif du projet

Choisir un membre*

zantour wael

Rôle*

Membre

Assigner le rôle

Précédent Suivant

- Interface Étape 3 pour l'attribution des rôles -

Project Management Tool

- Étape 4 : Récapitulatif

Cette étape affiche un récapitulatif des informations saisies, permettant à l'administrateur de vérifier les détails du projet créé.

✓ Créer un projet (Terminé) — Ajouter un membre au projet (Optional) — Assigner un rôle (Optional) — 4 Récapitulatif du projet

Récapitulatif du Projet

Nom du Projet : test-test
Description : test-test
Date de Début : December 18, 2024
Propriétaire: John

Membres du Projet

Nom	Email	Rôle
John	john.doe@example.com	Administrateur
zantour	wael@gmail.com	Membre

[Terminer](#) [Précédent](#)

- Résumé final du projet -

→ Après la création du projet, l'utilisateur qui a initié le projet obtient automatiquement le rôle **Administrateur**.

3.3.2.2. User Story : Invitation des membres

- **User Story :** "En tant qu'administrateur d'un projet, je veux pouvoir inviter d'autres membres à rejoindre mon projet en saisissant leur adresse e-mail afin de partager le projet."

Scénario et Implémentation :

L'invitation des membres peut se faire de deux manières :

1. Lors de la création du projet (Étape 2) :

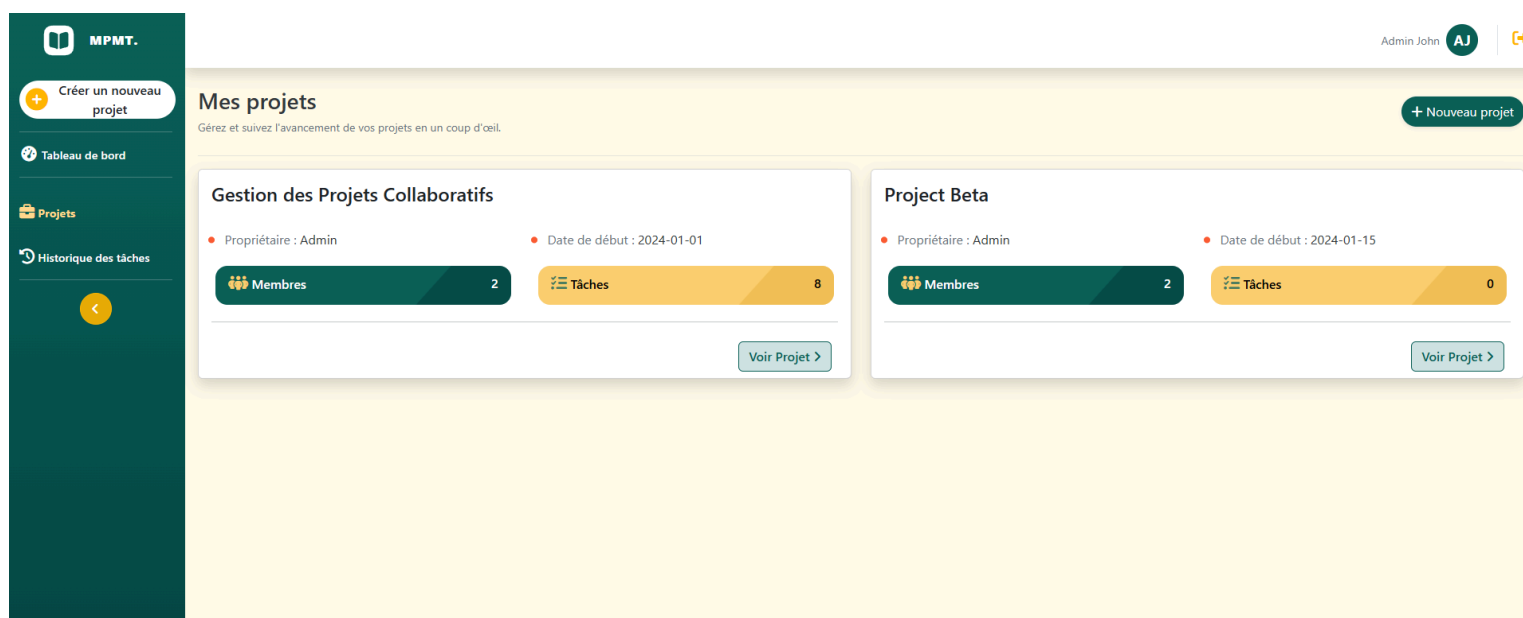
Les membres sont ajoutés directement via le formulaire. (voir capture d'écran de l'étape 2 de formulaire de création du projet).

2. Après la création du projet :

Page d'ensemble des Projets :

L'interface de la page "**Mes projets**" présente de manière synthétique les projets auxquels l'utilisateur est affecté.

- **Pour chaque projet**, l'utilisateur peut voir un aperçu rapide des membres et des tâches associées.
- Le bouton "**Voir Projet**" permet à l'utilisateur d'accéder à la **page de détail** du projet sélectionné.



- Vue d'ensemble des projets affectés à l'utilisateur -

Page de Détails d'un Projet

Lorsqu'un utilisateur clique sur "**Voir Projet**", il est redirigé vers la page de détail du projet. Cette page offre une vue complète des informations et des fonctionnalités du projet.



Informations détaillées affichées sur la page :

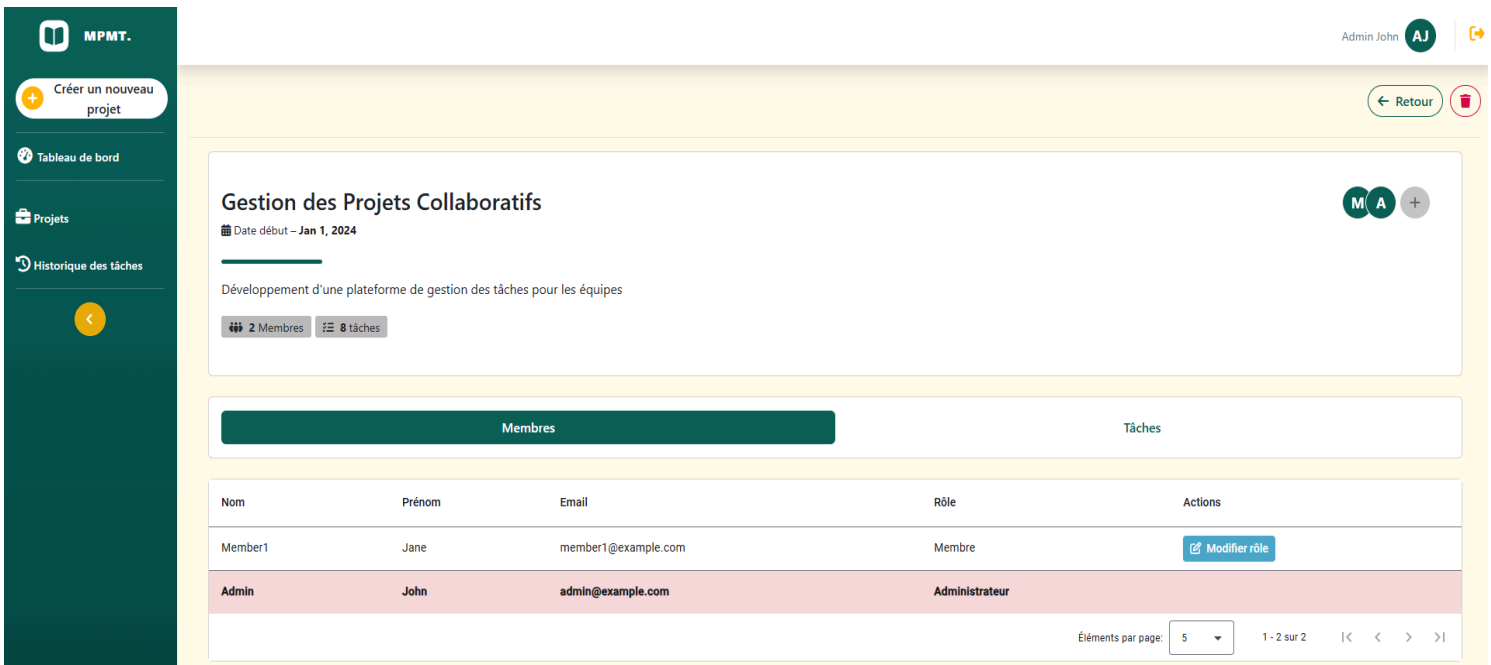
1. Informations générales du projet
 - Suppression du projet.
2. Vue des membres :
 - Tableau des membres affectés au projet.

Project Management Tool

- Permissions selon les rôles (Administrateur, Membre, Observateur).
 - **Ajout et modification des membres :**
 - Un utilisateur avec le rôle **Administrateur** peut ajouter de nouveaux membres et leur attribuer des rôles.
 - Il peut également modifier ou supprimer les membres existants.
 - **Contrôle des permissions :**
 - Les utilisateurs avec un rôle **Membre** ou **Observateur** n'ont pas accès au bouton d'ajout des membres.
3. Vue des tâches liées au projet :
- Les tâches sont affichées sous forme de tableau, **groupées par statut** (To-Do, In Progress, Done).

Scénarios d'utilisation :

1. Administrateur :
- Peut ajouter des membres, modifier leurs rôles .
 - Accès aux boutons   pour l'ajout des membres et "**Modifier rôle**" et "**Ajouter rôle**" .



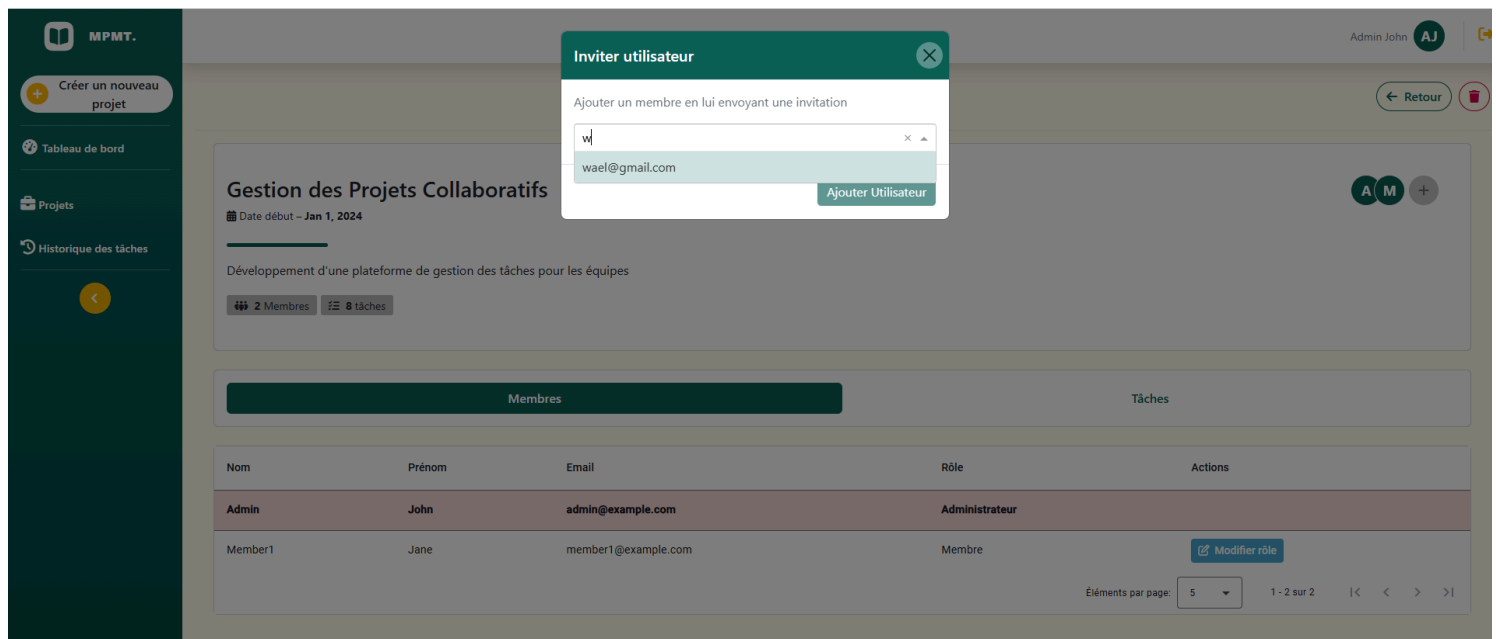
The screenshot shows the MPMT interface. On the left is a dark green sidebar with navigation links: 'Créer un nouveau projet', 'Tableau de bord', 'Projets', and 'Historique des tâches'. The main content area is titled 'Gestion des Projets Collaboratifs' and shows project details: 'Date début - Jan 1, 2024' and 'Développement d'une plateforme de gestion des tâches pour les équipes'. Below this, there are two tabs: 'Membres' (selected) and 'Tâches'. The 'Membres' tab displays a table with the following data:

Nom	Prénom	Email	Rôle	Actions
Member1	Jane	member1@example.com	Membre	Modifier rôle
Admin	John	admin@example.com	Administrateur	

At the bottom right, there is a pagination control showing 'Éléments par page: 5' and '1 - 2 sur 2'.

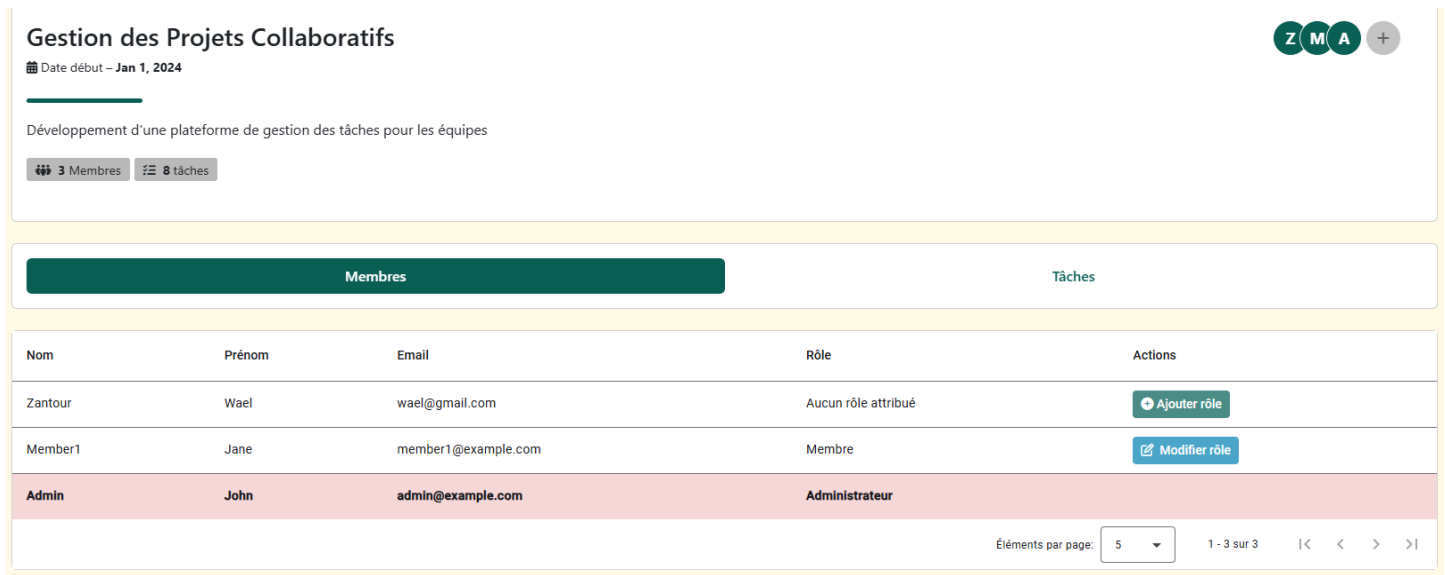
- Page de détail du projet pour un utilisateur avec rôle Administration -

Project Management Tool



- Formulaire d'invitation d'utilisateur -

- L'administrateur peut saisir l'adresse e-mail de l'utilisateur qu'il souhaite inviter. Une suggestion automatique facilite l'expérience utilisateur.



- Listes des utilisateurs après l'ajout d'un nouveau membre -

- Après avoir ajouté un membre, il apparaît dans la liste avec le statut **"Aucun rôle attribué"**.
- En cliquant sur **"Ajouter rôle"**, l'administrateur peut lui assigner un rôle approprié.

Project Management Tool

2. Membre et Observateur

- Accès limité à la visualisation des membres.
- Ne peut pas ajouter de nouveaux membres ni modifier les rôles.

MPMT.

Tableau de bord

Projets

Historique des tâches

Member1 Jane MJ

Retour

Gestion des Projets Collaboratifs

Date début - Jan 1, 2024

Développement d'une plateforme de gestion des tâches pour les équipes

3 Membres 8 tâches

Membres Tâches

Nom	Prénom	Email	Rôle	Actions
Zantour	Wael	wael@gmail.com	Aucun rôle attribué	
Admin	John	admin@example.com	Administrateur	
Member1	Jane	member1@example.com	Membre	

Éléments par page: 5 1 - 3 sur 3 |< < > >|

- Page des détails projet vue des membres pour un utilisateur Observateur ou membre -

3.3.2.3. User Story : Attribution des rôles

- **User Story** : "User En tant qu'administrateur d'un projet, je veux pouvoir attribuer des rôles aux membres du projet (administrateur, membre, observateur) afin de définir leurs permissions."

Scénario et Implémentation :

L'attribution des rôles aux membres peut se faire de deux manières :

1. **Pendant la création du projet** (Étape 3) :

Les rôles sont définis pour les membres invités dans le formulaire stepper.

Project Management Tool

Créer un Nouveau Projet

Renseignez les détails pour créer votre projet et commencez à collaborer.

← Retour

✓ Créer un projet (Terminé) ———— ✎ Ajouter un membre au projet Optional ———— 3 Assigner un rôle Optional ———— 4 Récapitulatif du projet

Choisir un membre*
Brown Morgan

Rôle*

Administrateur

Membre

Observateur

- Formulaire de création du projet "Affectation d'un rôle au membre du projet" -

2. Après la création du projet ou pour un projet existant :

Depuis la page de détail du projet, l'administrateur peut :

- Ajouter des rôles aux membres récemment ajoutés.

Ajouter un rôle à l'utilisateur

Email : wael@gmail.com

Membre

Administrateur

Membre

Observateur

Website Redesign

Date début - Jan 1, 2025

Revamping the corporate website with a modern design.

4 Membres 1 tâches

Membres

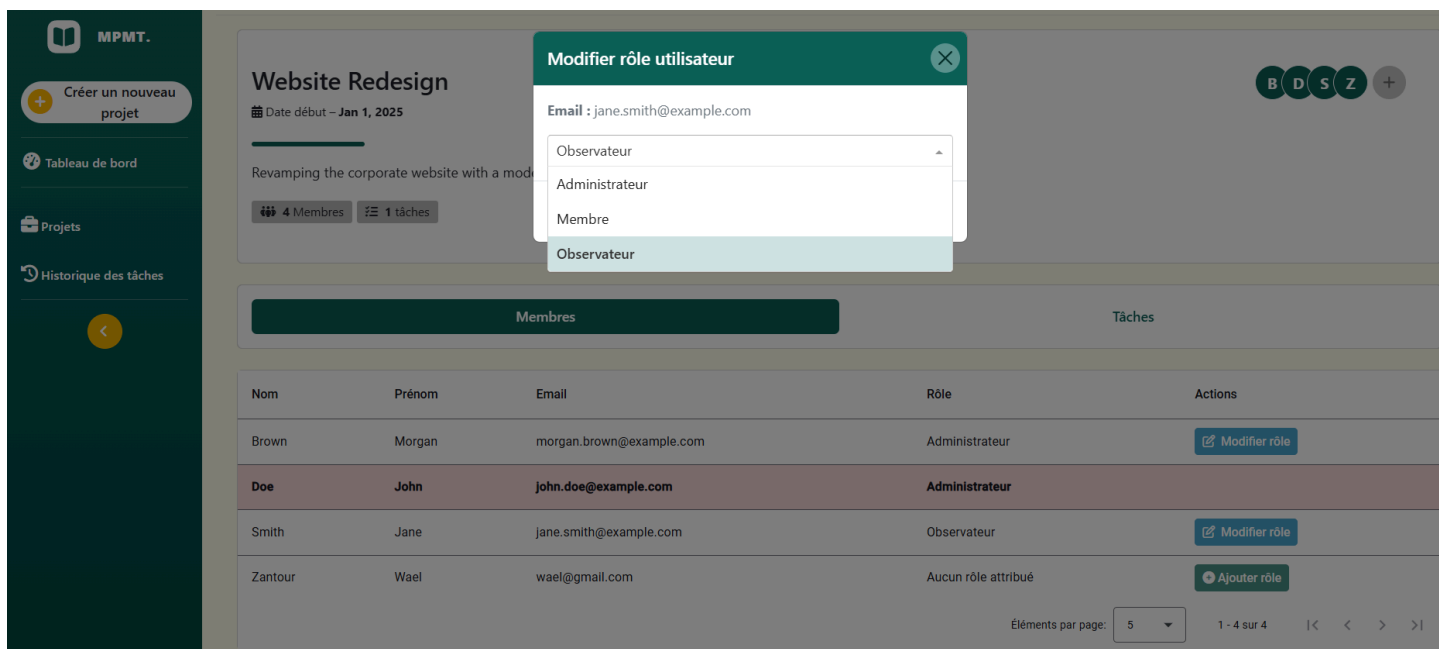
Nom	Prénom	Email	Rôle	Actions
Brown	Morgan	morgan.brown@example.com	Administrateur	Modifier rôle
Doe	John	john.doe@example.com	Administrateur	
Smith	Jane	jane.smith@example.com	Observateur	Modifier rôle
Zantour	Wael	wael@gmail.com	Aucun rôle attribué	Ajouter rôle

Éléments par page: 5 1 - 4 sur 4

- Formulaire d'affectation d'un rôle au membre du projet -

Project Management Tool

- Modifier les rôles des membres.



- Formulaire de mise à jour du rôle d'un membre dans un projet -

- **Contrôle d'accès et Masquage des fonctionnalités** : Les utilisateurs non autorisés (Membre, Observateur) ne voient pas les boutons de création ou d'administration des projets.

3.3.3. Création et Gestion des Tâches

Cette section détaille l'implémentation des fonctionnalités liées aux tâches, conformément aux user stories spécifiées. Elle met en avant la gestion collaborative des tâches et la visualisation avancée, tout en respectant les permissions basées sur les rôles. Chaque étape est illustrée par des captures d'écran pour mieux comprendre l'expérience utilisateur.

3.3.3.1. User Story : Création d'une tâche

- **User Story** : "En tant qu'administrateur ou membre d'un projet, je veux pouvoir créer des tâches pour mon projet avec un nom, une description, une date d'échéance et une priorité."

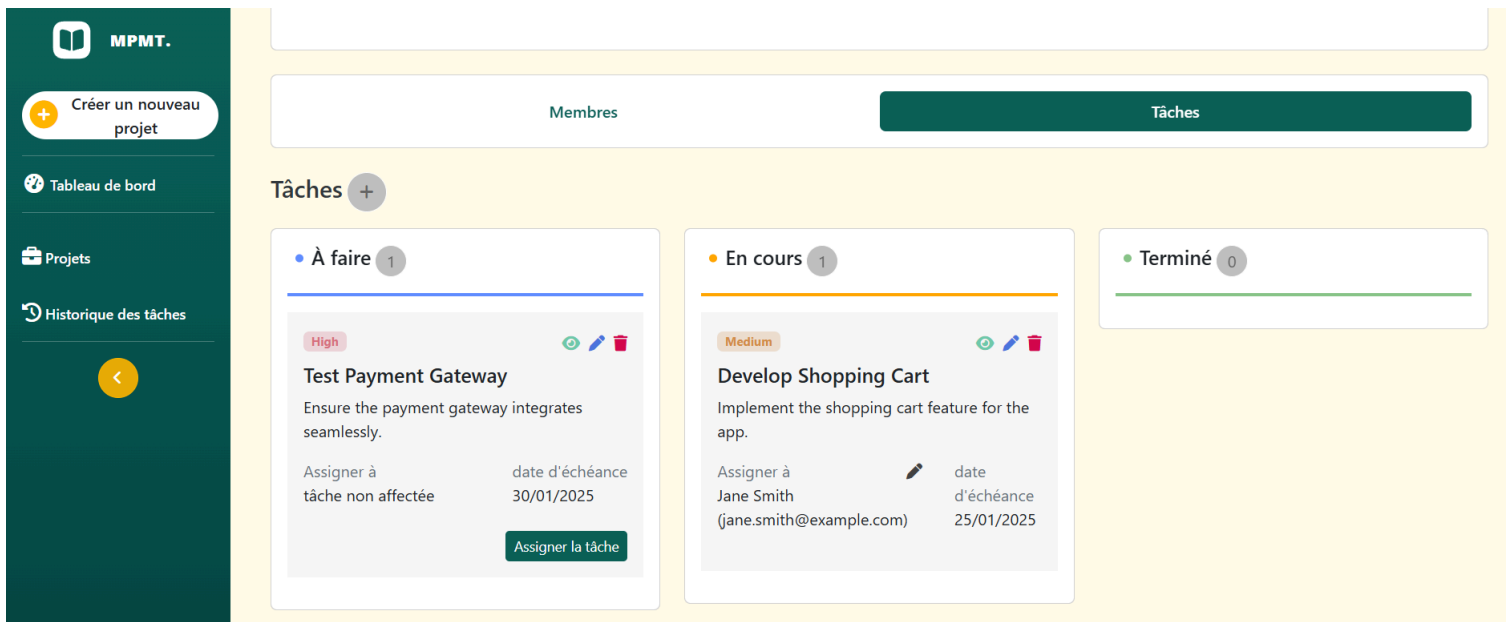
Scénario et Implémentation :

Depuis la page de **détail d'un projet**, sous la section **Tâches**, toutes les tâches sont présentées sous forme de tableau Kanban :

Project Management Tool

- **Colonnes dynamiques** : Chaque statut ("À faire", "En cours", "Terminé") correspond à une colonne.
- Les tâches changent de colonne automatiquement en fonction de leur statut.

Les utilisateurs ayant le rôle **Administrateur** ou **Membre** peuvent créer une tâche en cliquant sur le bouton "+" situé au-dessus du tableau des tâches.



- Page détails du projet "Vue Tâches" -

Processus de création :

3. **Formulaire de création** : Un modal s'affiche avec un formulaire de création
4. **Statut par défaut** : La tâche est créée avec un statut par défaut "**À faire**" et s'affiche automatiquement sous la colonne correspondant.

Project Management Tool

Nouvelle tâche

Informations générales

Nom:

Description:

Priorité:

Statut:

Dates

Date de début:

Date d'échéance:

- *formulaire de création de tâche* -

3.3.3.2. User Story : Attribution d'une tâche

- **User Story** : "En tant qu'administrateur ou membre, je veux pouvoir assigner des tâches à des membres spécifiques du projet ."

Scénario et Implémentation :

- **Bouton d'attribution** : Un bouton sur la tâche ouvre un formulaire modal permettant de sélectionner un membre du projet.

Affecter la tâche Test Payment Gateway

Assigner à un membre

- *Formulaire d'attribution d'une tâche à un membre* -

3.3.3.3. User Story : Mise à jour d'une tâche

- **User Story** : *"En tant qu'administrateur ou membre, je veux pouvoir mettre à jour une tâche afin de changer n'importe quelle information ou ajouter une date de fin. "*

Scénario et Implémentation :

- Les utilisateurs ayant le rôle **Administrateur** ou **Membre** peuvent mettre à jour les informations d'une tâche existante :
 1. Modification des informations générales :

Ce formulaire intuitif permet aux utilisateurs de rapidement mettre à jour les informations clés d'une tâche. Les champs tels que priorité et statut sont fournis sous forme de menus déroulants pour faciliter la sélection.

The screenshot shows a 'Modifier tâche' (Edit task) modal form. The form is titled 'Modifier tâche' and has a close button (X) in the top right corner. It is divided into two main sections: 'Informations générales' (General information) and 'Dates'. The 'Informations générales' section includes fields for 'Nom' (Name), 'Description', 'Priorité' (Priority), and 'Statut' (Status), each with a dropdown menu. The 'Dates' section includes fields for 'Date de début' (Start date) and 'Date d'échéance' (Due date). At the bottom right, there are two buttons: 'Annuler' (Cancel) and 'Modifier la tâche' (Edit task). The background shows a blurred view of the project management interface with a sidebar and a main content area.

- Formulaire de modification des informations générales -

2. Modification de l'affectation de la tâche :
 - Les utilisateurs peuvent réassigner la tâche à un autre membre du projet.
 - Lorsqu'une tâche est réassignée, un e-mail de notification est automatiquement envoyé au nouveau membre désigné

Project Management Tool



- Formulaire de réaffectation d'une tâches -

Accès et Restrictions :

- Les options de modification (par exemple, via les icônes bleues et grises) et de suppression sont disponibles uniquement pour les rôles disposant des permissions nécessaires (Administrateur et Membre).

• À faire 1

High👁️ ✎️ 🗑️

Test Payment Gateway
Ensure the payment gateway integrates seamlessly.

Assigner à
Wael-2 Zantour-2
(zantour.1999@gmail.com)

✎️

date
d'échéance
30/01/2025

- Carte de tâche : Vue Administrateur et Membre -

- Les Observateurs peuvent uniquement consulter les informations des tâches sans les modifier.

• À faire 1

Low👁️

Design Homepage
Create a responsive design for the homepage.

Assigner à
tâche non affectée

date d'échéance
10/01/2025

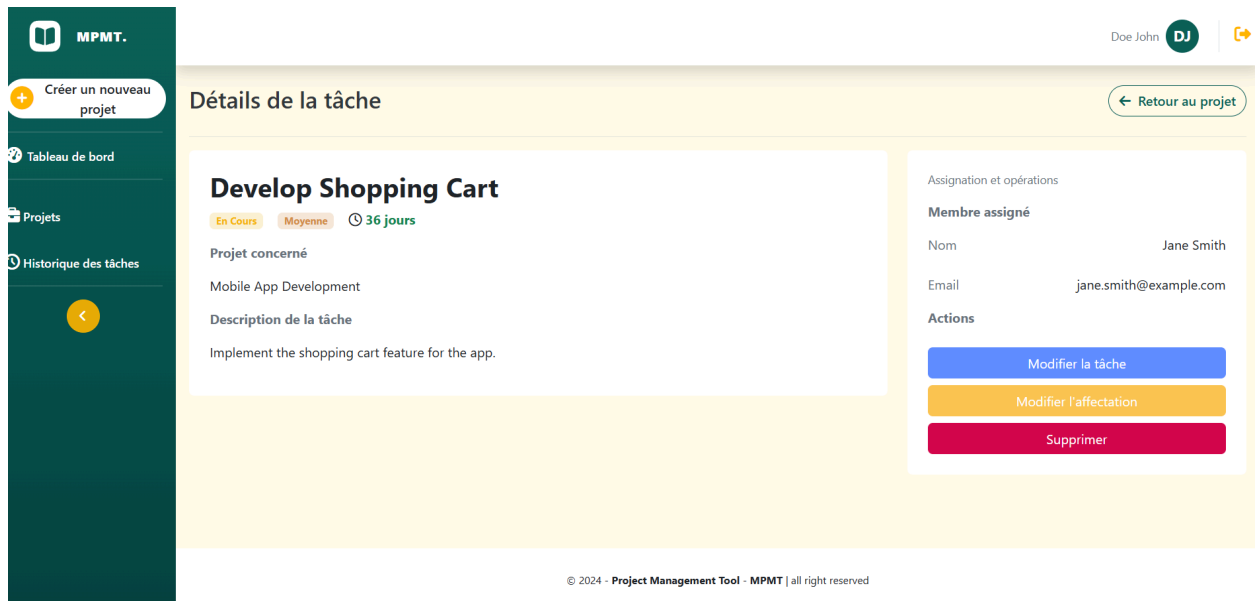
- Carte de tâche : Vue Observateur -

3.3.3.4. User Story : Visualisation d'une tâche unitaire

- **User Story** : "En tant qu'administrateur, membre ou observateur, je veux pouvoir visualiser une tâche unitaire afin d'en voir toutes les informations. "

Scénario et Implémentation :

- Cette section présente l'interface de visualisation d'une tâche unitaire, permettant à tous les utilisateurs, quels que soient leurs rôles (Administrateur, Membre ou Observateur), d'accéder aux détails complets d'une tâche dans un projet.
- **Actions disponibles (Administrateur/Membre uniquement) :**
 - **Modifier la tâche** : Affiche un formulaire pour modifier les informations telles que le nom, la description, la priorité, ou la date d'échéance.
 - **Modifier l'affectation** : Permet de réassigner la tâche à un autre membre du projet.
 - **Supprimer la tâche** : Supprime la tâche du projet.
- Ces actions sont invisibles pour les Observateurs. Ils peuvent uniquement consulter les informations de la tâche, sans possibilité de modification



- Interface dédiée à la visualisation des détails d'une tâche -

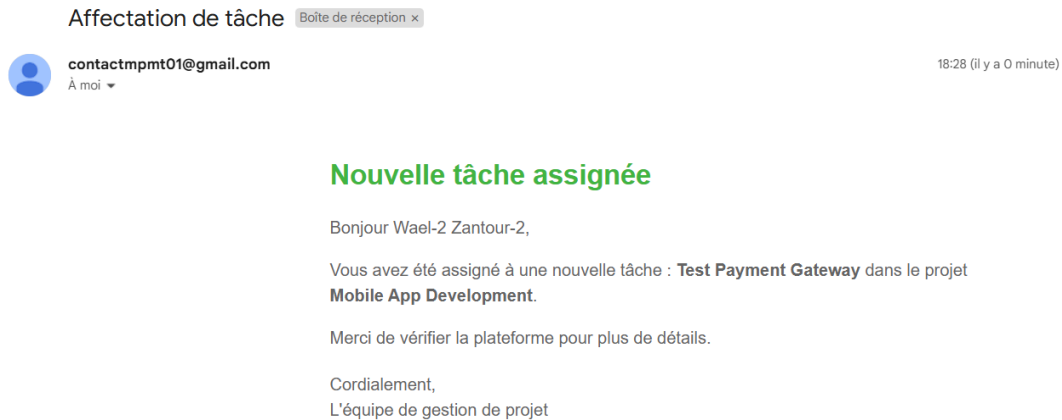
3.3.3.5. User Story : Notifications par e-mail

- **User Story** : "En tant qu'administrateur, membre ou observateur, je veux pouvoir recevoir des notifications par e-mail lorsqu'une tâche est assignée. "

Scénario et Implémentation :

Lorsqu'une tâche est attribuée à un membre du projet :

- Une notification par e-mail est automatiquement envoyée à l'adresse e-mail du membre pour l'informer de cette affectation.



- Notification par e-mail pour l'assignation d'une tâche -

3.3.3.6. User Story : Notifications par e-mail

- **User Story** : "En tant qu'administrateur, membre ou observateur, je veux pouvoir visualiser les tâches selon les statuts afin de suivre l'avancement des tâches sur un tableau de bord. "

Scénario et Implémentation :

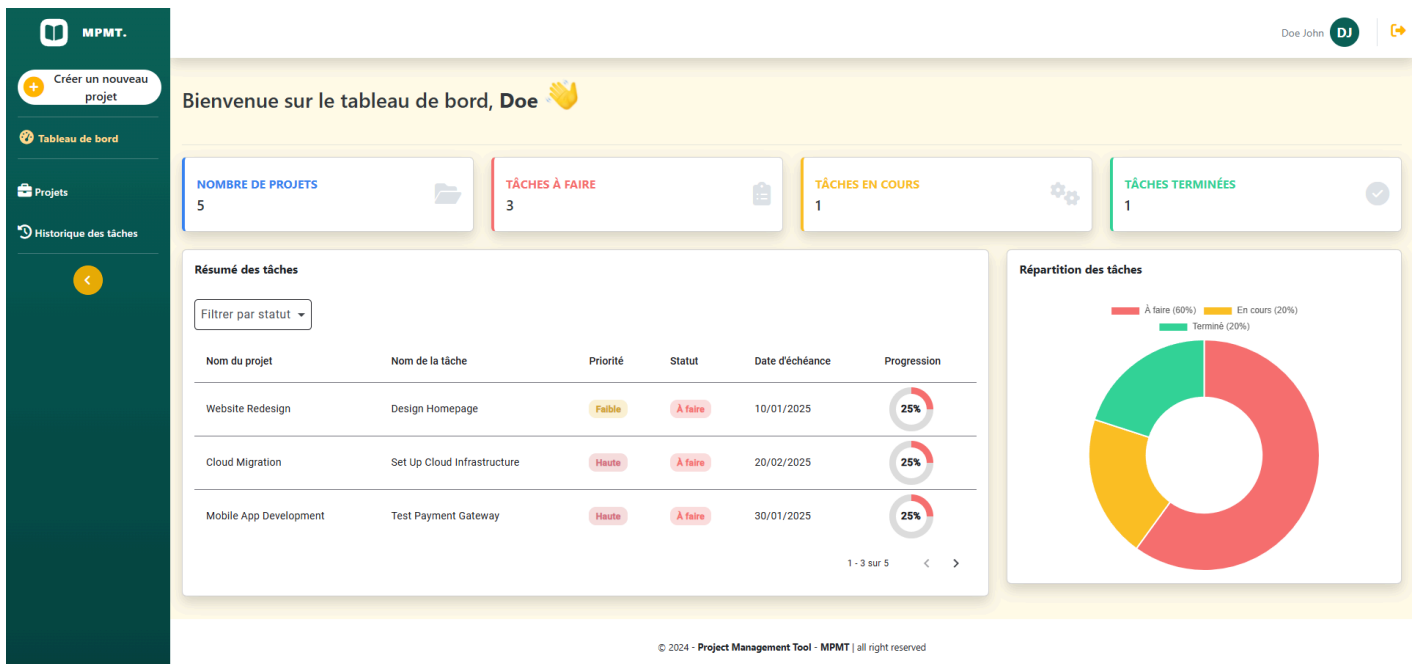
Le **tableau de bord des tâches** est accessible à tous les rôles (Administrateur, Membre, Observateur) et permet de :

- Visualiser un résumé des tâches par statut.
- Suivre l'avancement global du projet.

Fonctionnalités du tableau de bord

1. Indicateurs clés en haut de page
2. Résumé des tâches : Une table présente les informations détaillées pour chaque tâche.
3. Répartition des tâches (graphique) : Un diagramme en anneau illustre la répartition des tâches selon leur statut.

Project Management Tool



- Tableau des tâches filtré par statut avec progression indiquée via des graphiques circulaires -

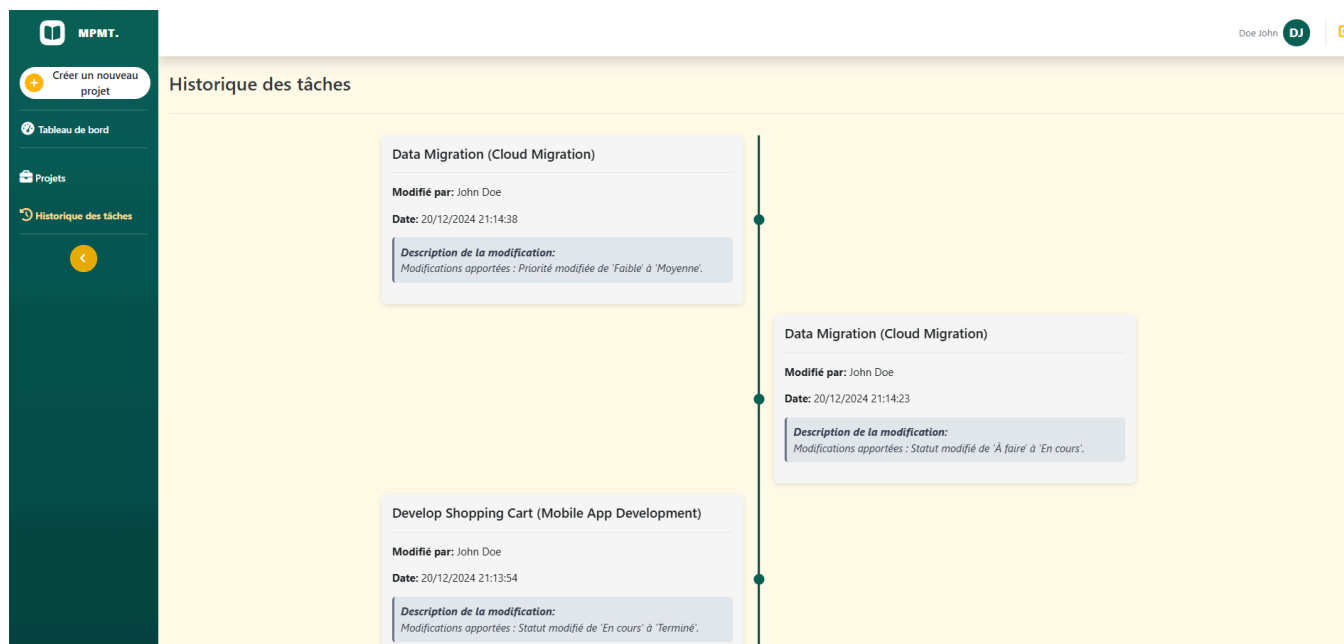
3.3.3.7. User Story : Suivi de l'historique des tâches

- **User Story** : "En tant qu'administrateur, membre ou observateur, je veux pouvoir suivre l'historique des modifications apportées aux tâches."

Scénario et Implémentation :

Chaque modification réalisée sur une tâche (changement de statut, mise à jour des informations, assignation) est tracée et accessible depuis la **page "Historique des tâches"** :

- **Affichage en timeline** : L'historique est présenté sous forme de chronologie pour une meilleure lisibilité.
- **Détails** : Chaque entrée indique la nature de la modification, l'utilisateur qui l'a réalisée, et la date/heure.



- Page Historique des tâches -

3.4. Conclusion

Cette phase de développement a permis de répondre efficacement aux user stories grâce à des fonctionnalités clés :

- **Création et gestion des projets** : Une interface fluide permet aux administrateurs de créer des projets, inviter des membres et gérer leurs rôles de manière flexible.
- **Gestion des tâches** : Les tâches peuvent être créées, assignées et mises à jour par les administrateurs et membres, avec un suivi des modifications et des notifications par e-mail pour améliorer la collaboration.
- **Tableau de bord** : Une vue synthétique et dynamique des projets et tâches, accessible à tous les rôles, permet un suivi global et intuitif.

Chaque fonctionnalité a été conçue pour offrir une expérience utilisateur optimale, garantir la sécurité et respecter les permissions selon les rôles, tout en posant les bases pour des évolutions futures.

4. Phase 3 - Tests et Validations

4.1. Introduction

La phase de tests a été essentielle pour garantir que l'application respecte les user stories spécifiées, tout en offrant une expérience utilisateur fluide et sans faille. Les efforts ont été concentrés sur plusieurs niveaux de tests pour valider les fonctionnalités, assurer une couverture de code adéquate et détecter les éventuelles régressions.

4.2. Stratégie de Test Adoptée

La stratégie de test s'est appuyée sur trois axes principaux pour garantir la robustesse de l'application :

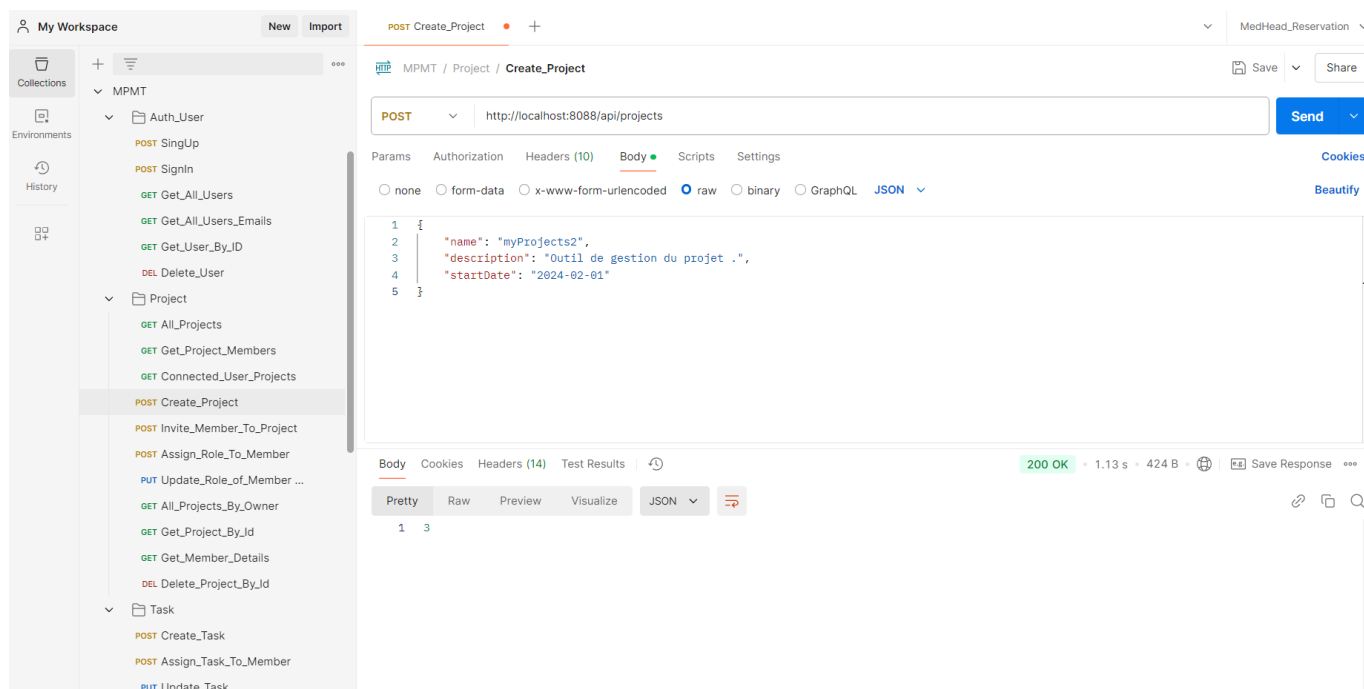
- **Tests Unitaires** : Validation individuelle des composants critiques du backend et du frontend.
- **Tests d'Intégration** : Vérification des interactions entre les différentes couches de l'application (backend, frontend et base de données).

4.3. Tests API avec Postman

Les endpoints du backend ont été testés de manière approfondie en utilisant **Postman** pour s'assurer qu'ils répondent correctement aux différentes requêtes (GET, POST, PUT, DELETE). Voici les principales actions testées :

- **Création de Projet** :
 - Vérification que seuls les utilisateurs autorisés peuvent créer un projet.
 - Validation des erreurs retournées pour les données manquantes ou invalides.

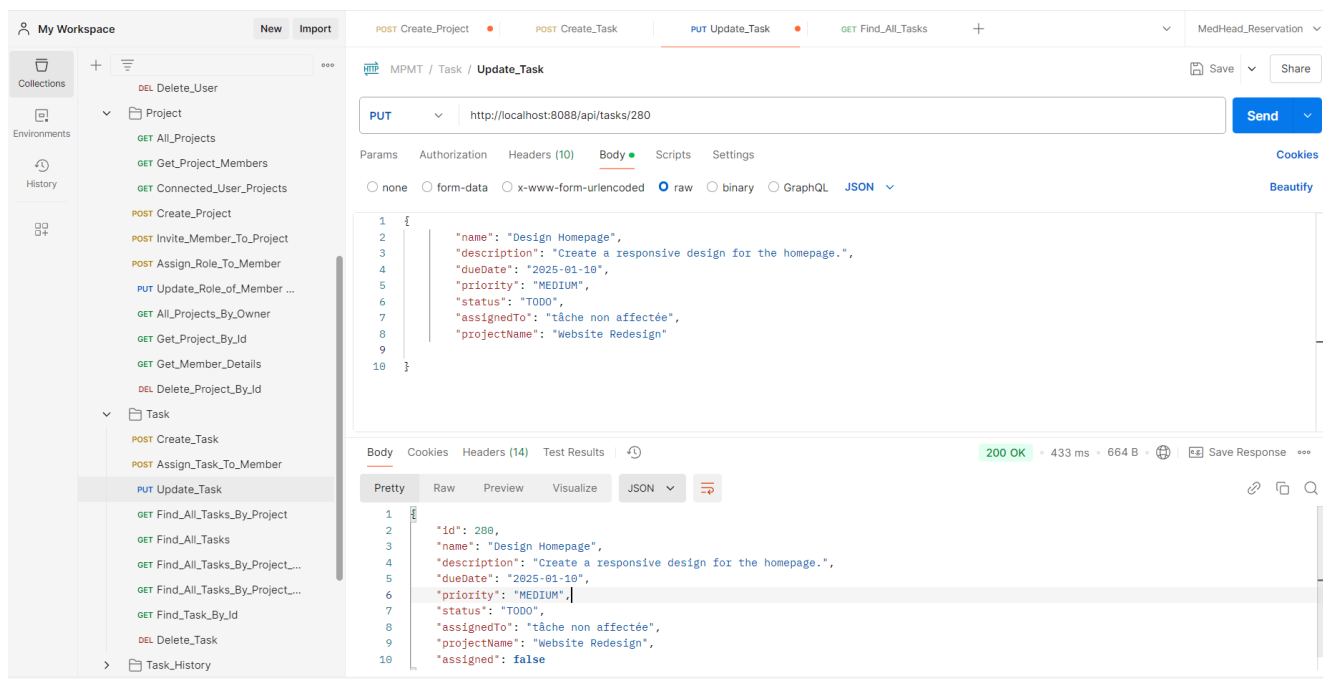
Project Management Tool



- Requête POST pour créer un projet avec les données nécessaires -

- **Gestion des Tâches :**

- Création, modification, et suppression des tâches.
- Test des restrictions basées sur les rôles pour chaque action.

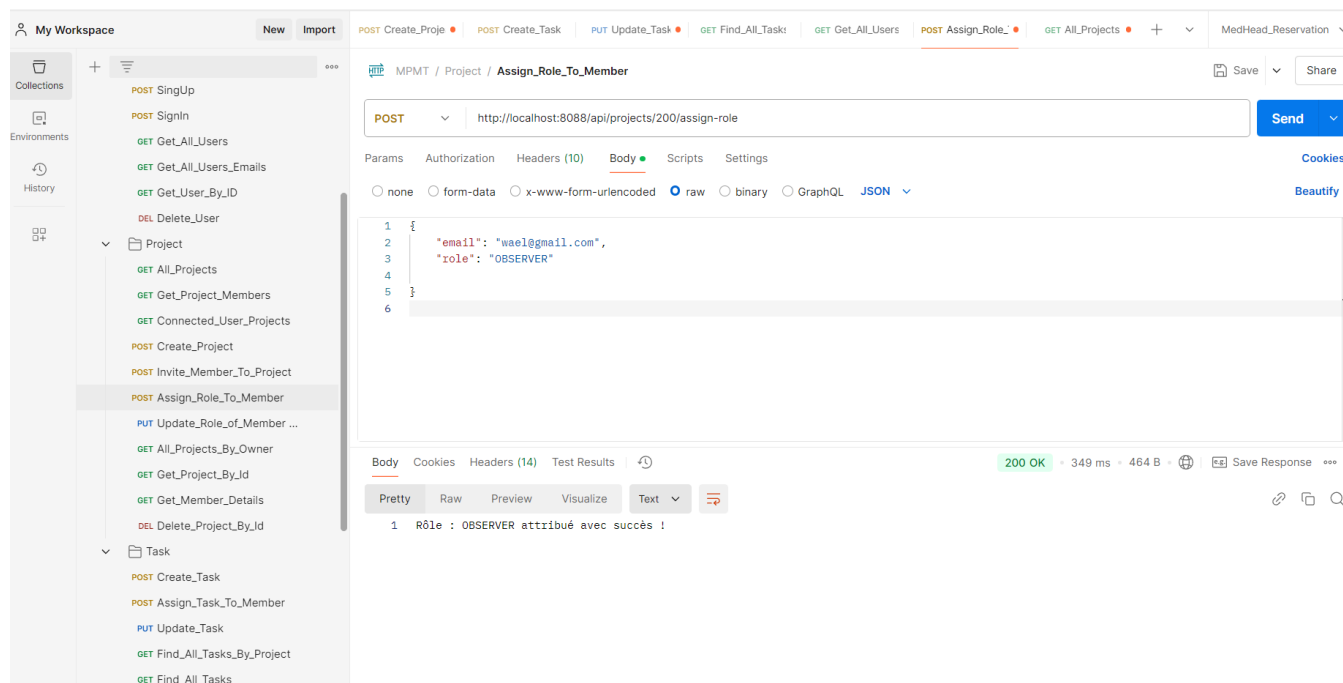


- Requête PUT pour mettre à jour une tâche existante -

Project Management Tool

- **Attribution des Rôles :**

- Validation que seuls les administrateurs peuvent créer les rôles des membres.
- Vérification des réponses en cas d'autorisation insuffisante.

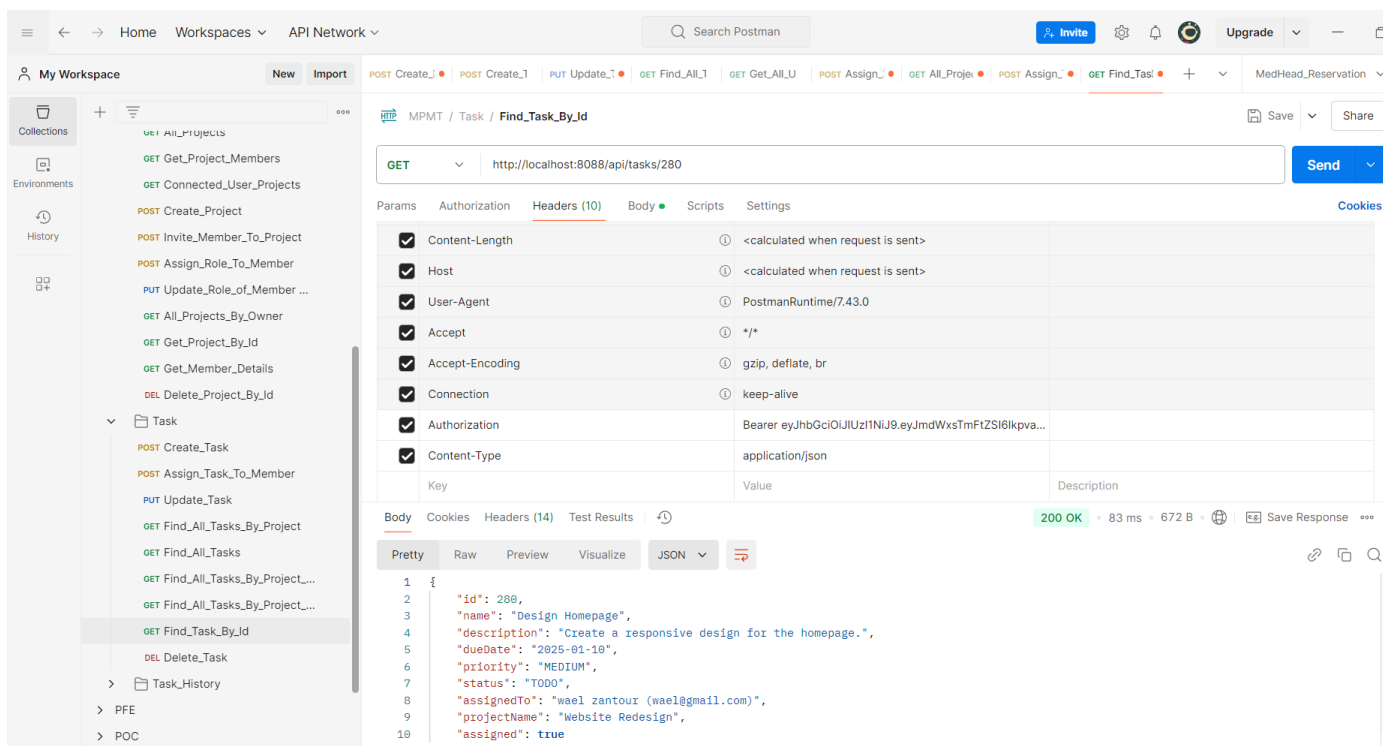


- Requête POST pour attribuer un rôle à un membre d'un projet -

- **Détails d'une tâche :**

- Récupérer les informations détaillées d'une tâche spécifique en fournissant son identifiant (**id**)

Project Management Tool



- Réponse HTTP 200 pour la visualisation des détails d'une tâche -

Résultats :

- Tous les endpoints testés avec Postman ont fonctionné comme prévu, validant ainsi leur bon fonctionnement dans les scénarios couverts.

4.4. Couverture de Code

La couverture de code est une étape essentielle pour garantir la qualité et la robustesse du projet. Des outils de couverture ont été utilisés pour évaluer l'efficacité des tests, tant pour le **frontend** que pour le **backend**. Les rapports de couverture fournissent des détails sur le pourcentage de lignes de code, de branches et de fonctions couvertes par les tests.

4.4.1. Couverture côté Frontend

La couverture de code du frontend a été réalisée en utilisant **Jest**, l'outil de test par défaut pour Angular. Voici les étapes et les commandes utilisées pour générer la couverture :

1. **Commande utilisée :**
ngx test --code-coverage

Project Management Tool

2. Rapport de couverture :

- Les résultats montrent une couverture de **73% des instructions** globalement avec des variations selon les composants.
- Le tableau ci-dessous illustre les pourcentages de couverture par fichier:

File	% Stmts	% Branch	% Funcs	% Lines
All files	73.01	44.15	68.81	72.7
app	100	100	100	100
app.component.html	100	100	100	100
app.component.ts	100	100	100	100
app/interceptors	54.54	0	50	50
http-token.interceptor.ts	54.54	0	50	50
.../projects/components/footer	100	100	100	100
footer.component.html	100	100	100	100
footer.component.ts	100	100	100	100
...nents/no-projects-component	60	100	20	55.55
...s-component.component.html	100	100	100	100
...cts-component.component.ts	55.55	100	20	50
...s/components/project-taches	56.17	43.33	51.11	55.69
project-taches.component.html	100	100	100	100
project-taches.component.ts	55.9	43.33	51.11	55.41
...projects/components/sidebar	65.11	60	36.84	65.85
sidebar.component.html	100	100	100	100
sidebar.component.ts	64.28	60	36.84	65

- Rapport détaillée de la couverture du frontend avec Jest -

```
Test Suites: 24 passed, 24 total
Tests:      149 passed, 149 total
Snapshots:  0 total
Time:       146.724 s
```

- Résumé des tests exécutés pour le frontend -

4.4.2. Couverture côté Backend

La couverture de code backend a été mesurée en utilisant **Jacoco**, un outil d'analyse de couverture pour Java. Les tests ont été exécutés avec **JUnit** et **Mockito**, et les rapports ont été générés au format HTML (<target/site/jacoco/index.html>).

1. Commandes utilisées :

- Pour exécuter les tests :
mvn test

Project Management Tool

- Pour générer le rapport de couverture Jacoco :
mvn jacoco:report

2. Rapport de couverture :

- Les résultats montrent une couverture de **70% des instructions** globalement avec des variations selon les packages :
 - Couverture des **instructions** : 70%.
 - Couverture des **branches** : 59%.
 - Couverture des **lignes** : 69%.
 - Couverture des **classes** : 85%.

project-management-backend

project-management-backend

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.pmt.project_management.task		75 %		61 %	27	69	25	167	12	38	0	5
com.pmt.project_management.security		33 %		22 %	24	32	57	81	13	21	1	5
com.pmt.project_management.project		73 %		67 %	27	64	27	153	17	44	1	4
com.pmt.project_management.handler		23 %		n/a	6	9	33	46	6	9	1	2
com.pmt.project_management.auth		86 %		100 %	2	15	4	34	2	14	0	2
com.pmt.project_management.user		88 %		71 %	8	20	5	34	4	13	0	2
com.pmt.project_management.exception		42 %		n/a	3	5	6	10	3	5	2	4
com.pmt.project_management.email		90 %		n/a	0	1	2	19	0	1	0	1
com.pmt.project_management		37 %		n/a	1	2	2	3	1	2	0	1
com.pmt.project_management.config		98 %		70 %	4	14	1	29	1	9	1	4
com.pmt.project_management.role		100 %		n/a	0	1	0	2	0	1	0	1
Total	892 of 3 001	70 %	61 of 150	59 %	102	232	162	578	59	157	6	31

- Rapport de la couverture des tests côté backend avec Jacoco -

4.4.3. Analyse Générale

Ces rapports mettent en évidence une bonne couverture des tests pour les fonctionnalités essentielles de l'application, tant pour le frontend que pour le backend. Ils révèlent cependant quelques lacunes dans des cas particuliers, comme les branches conditionnelles ou les modules de sécurité. Ces éléments pourraient être améliorés pour atteindre une couverture encore plus exhaustive.

4.5. Automatisation des Tests

L'automatisation des tests a été intégrée dans les pipelines CI/CD pour le frontend et le backend, garantissant ainsi une exécution continue et systématique des tests unitaires et la génération des rapports de couverture.

Project Management Tool

4.5.1. Pipeline Frontend

Dans le pipeline CI/CD pour le frontend, les tests unitaires sont exécutés à l'aide de **Jest**. Voici les étapes intégrées dans le workflow :

1. **Exécution des tests unitaires avec Jest** : Les tests sont exécutés avec la commande `npm run test -- --watch=false --coverage`, ce qui génère un rapport de couverture.
2. **Téléchargement du rapport de couverture** : Le rapport généré est sauvegardé en tant qu'artefact à l'aide de l'action GitHub `upload-artifact`.

```
# 4. Run tests frontend
- name: Run unit tests with Jest and generate coverage
  run: |
    cd project-management-frontend
    npm run test -- --watch=false --coverage

# 5. Upload test coverage report
- name: Upload coverage report
  uses: actions/upload-artifact@v3
  with:
    name: jest-coverage-report
    path: project-management-frontend/coverage/
```

4.5.2. Pipeline Backend

Dans le pipeline CI/CD pour le backend, les tests unitaires sont exécutés avec **Maven** en utilisant Jacoco pour générer les rapports de couverture.

1. **Exécution des tests unitaires avec Maven** : Les tests sont lancés à l'aide de la commande `./mvnw clean test verify`. Cette commande exécute tous les tests unitaires et génère un rapport de couverture via Jacoco.
2. **Téléchargement du rapport Jacoco** : Les fichiers de couverture sont sauvegardés comme artefacts à l'aide de l'action GitHub `upload-artifact`.

```
unit-tests:
  runs-on: ubuntu-latest
  name: Unit tests
  needs: compile # Exécuter après le job 'compile'
  steps:
    - name: Checkout code
      uses: actions/checkout@v4
      with:
        fetch-depth: 0 # Récupère tout l'historique Git
        submodules: recursive
```

```
# Configuration de JDK 17 pour les tests
- name: Set up JDK
  uses: actions/setup-java@v4
  with:
    java-version: 17
    distribution: 'corretto'

# Exécution des tests unitaires du projet
- name: Run Unit Tests and Generate Coverage Report
  run: |
    cd project-management-backend
    chmod +x mvnw
    ./mvnw clean test verify

# Publier les rapports de couverture comme artefacts
- name: Upload Jacoco Coverage Report
  uses: actions/upload-artifact@v3
  with:
    name: jacoco-coverage-report
    path: project-management-backend/target/site/jacoco/
```

Résultats des Pipelines

- Les tests unitaires ont été exécutés avec succès pour les deux environnements.
- Les rapports de couverture montrent que les seuils exigés par le projet ont été atteints pour les composants critiques .

4.6. Conclusion

Cette phase de tests a permis de garantir que l'application est robuste, fiable et conforme aux exigences fonctionnelles définies dans l'énoncé. L'intégration des tests dans une chaîne CI/CD assure la continuité de la qualité tout au long du cycle de vie du projet, offrant ainsi une base solide pour le déploiement et les futures évolutions.

5. Phase 4 - Industrialisation et Déploiement

5.1. Introduction

La phase d'industrialisation et de déploiement constitue une étape clé dans la livraison du projet. Elle vise à garantir une mise en production fluide, fiable et répétable. Cette phase s'appuie sur des outils modernes tels que **Docker** et **GitHub Actions** pour automatiser les processus de construction, de test et de déploiement. Voici une description détaillée des approches adoptées.

5.2. Dockerization

La conteneurisation des services backend et frontend avec Docker permet de garantir des environnements uniformes et indépendants des systèmes hôtes.

5.2.1. Dockerization du Backend

Le backend utilise Java avec Spring Boot. Le **Dockerfile** du backend est structuré en deux étapes principales :

- **Étape de build** : Compilation du projet avec Maven pour générer un fichier **.jar**.
- **Étape de runtime** : Utilisation d'Amazon Corretto (JDK 17) pour exécuter le service.

Extrait du **Dockerfile** Backend :

```
# Build stage
FROM maven:3.8.7-openjdk-18 AS build
WORKDIR /build
COPY project-management-backend/pom.xml .
RUN mvn dependency:go-offline
COPY project-management-backend/src ./src
RUN mvn clean package -DskipTests

# Runtime stage
FROM amazoncorretto:17
ARG APP_VERSION=1.0.0

WORKDIR /app
COPY --from=build /build/target/project-management-backend*.jar /app/

EXPOSE 8088

ENV DB_URL=jdbc:postgresql://postgres:5432/pmt
ENV JAR_VERSION=${APP_VERSION}

CMD java -jar -Dspring.datasource.url=${DB_URL}
project-management-backend-${JAR_VERSION}.jar
```

5.2.2. Dockerization du Frontend

Le frontend développé avec Angular est compilé en fichiers statiques, puis servi via un serveur NGINX. Le **Dockerfile** du frontend est structuré comme suit :

- **Étape de build** : Compilation des fichiers Angular en mode production.

Project Management Tool

- **Étape de runtime** : Utilisation de NGINX pour distribuer les fichiers.

Extrait du **Dockerfile** Frontend :

```
# Étape 1 : Construction de l'application
FROM node:18 AS build-stage

# Définir le répertoire de travail dans le conteneur
WORKDIR /app

# Copier les fichiers package.json et package-lock.json dans le conteneur
COPY package*.json ./

# Installer les dépendances
RUN npm install

# Copier le reste des fichiers du projet dans le conteneur
COPY . .

# Construire l'application
RUN npm run build --prod

# Étape 2 : Configuration pour servir l'application
# Utilisation d'une image NGINX pour servir les fichiers statiques
FROM nginx:alpine

# Supprimez la configuration par défaut
RUN rm /etc/nginx/conf.d/default.conf

# Copier le fichier de configuration NGINX personnalisé
COPY nginx.conf /etc/nginx/nginx.conf

# Copier les fichiers de build dans le répertoire de NGINX
COPY --from=build-stage /app/dist/project-management-frontend/browser
/usr/share/nginx/html
CMD ["nginx", "-g", "daemon off;"]

# Exposer le port 80 pour accéder à l'application
EXPOSE 80
```

5.3. Orchestration avec Docker Compose

Un fichier **docker-compose.yml** a été utilisé pour orchestrer les différents services nécessaires au projet, incluant :

- **PostgreSQL** : Base de données principale.
- **PgAdmin** : Interface graphique pour gérer PostgreSQL.
- **MailDev** : Serveur SMTP pour simuler l'envoi de courriers électroniques

Project Management Tool

Extrait du `docker-compose.yml` :

```
services:
  postgres:
    container_name: postgres
    image: postgres
    environment:
      POSTGRES_USER: wael
      POSTGRES_PASSWORD: wael
      PGDATA: /var/lib/postgresql/data
      POSTGRES_DB: pmt
    volumes:
      - postgres:/var/lib/postgresql/data
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql

    ports:
      - "5432:5432"
    networks:
      - pmt-net
    restart: unless-stopped

  pgadmin:
    container_name: pgadmin
    image: dpage/pgadmin4
    environment:
      PGADMIN_DEFAULT_EMAIL:
        ${PGADMIN_DEFAULT_EMAIL:-pgadmin@pgadmin.org}
      PGADMIN_DEFAULT_PASSWORD: ${PGADMIN_DEFAULT_PASSWORD:-admin}
      PGADMIN_CONFIG_SERVER_MODE: 'False'
    volumes:
      - pgadmin:/var/lib/pgadmin
    ports:
      - "5050:80"
    networks:
      - pmt-net
    restart: unless-stopped

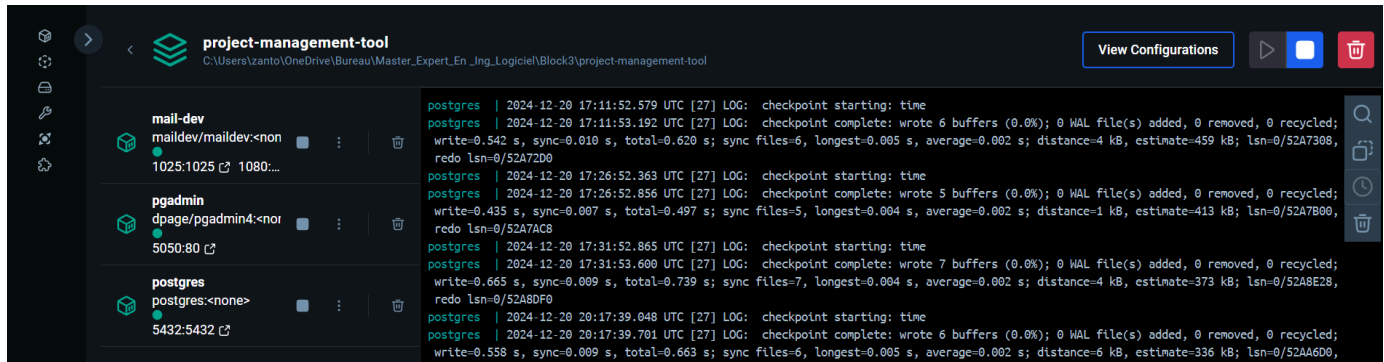
  mail-dev:
    container_name: mail-dev
    image: maildev/maildev
    ports:
      - 1080:1080
      - 1025:1025

networks:
  pmt-net:
    driver: bridge

volumes:
  postgres:
  Pgadmin:
```

Project Management Tool

Ci-dessous le tableau de bord Docker desktop des conteneurs actuellement en cours d'exécution pour notre projet.



- Vue des Conteneurs PostgreSQL, PgAdmin et MailDev dans Docker Desktop -

5.4. Pipelines CI/CD

Les pipelines CI/CD, configurés avec GitHub Actions, automatisent les tests, la construction et le déploiement.

5.4.1. Backend

Le pipeline du backend comprend :

1. **Compilation** : Compilation du projet avec Maven.
2. **Tests unitaires** : Exécution des tests avec JaCoCo pour obtenir un rapport de couverture.
3. **Dockerisation et Push** : Création de l'image Docker et publication sur Docker Hub.

Extrait du Workflow Backend :

```
jobs:
  compile:
    runs-on: ubuntu-latest
    name: Compile project
    steps:
      # Compilation du projet
      - name: Compile project
        run: |
          cd project-management-backend
          chmod +x mvnw
          ./mvnw clean compile

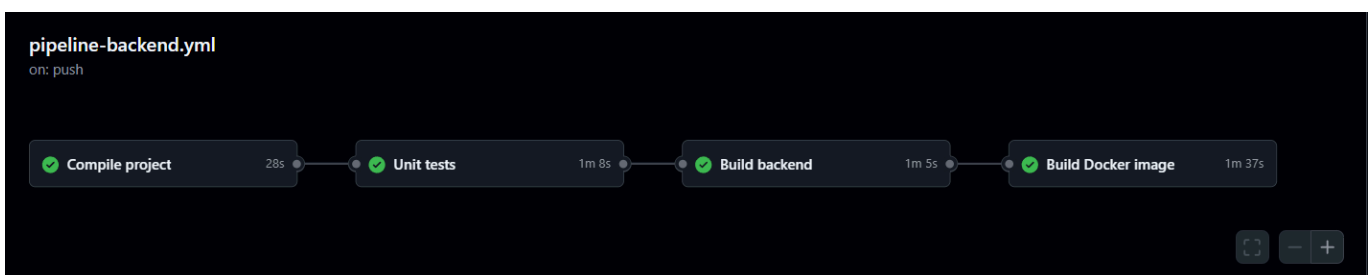
  unit-tests:
    runs-on: ubuntu-latest
```


Project Management Tool

```
name: Unit tests
needs: compile
steps:
  # Exécution des tests unitaires du projet
  - name: Run Unit Tests and Generate Coverage Report
    run: |
      cd project-management-backend
      chmod +x mvnw
      ./mvnw clean test verify

  # Publier les rapports de couverture comme artefacts
  - name: Upload Jacoco Coverage Report
    uses: actions/upload-artifact@v3
    with:
      name: jacoco-coverage-report
      path: project-management-backend/target/site/jacoco/

build-image-and-deploy:
  runs-on: ubuntu-latest
  name: Build Docker image
  needs: [build]
  steps:
    # Construction et push image backend
    - name: Build & Push Docker Image
      uses: docker/build-push-action@v5
      with:
        context: docker-build-context
        file: docker/backend/Dockerfile
        push: true
        tags: ${{ secrets.DOCKERHUB_USERNAME }}/pmt-api:latest
      build-args:
        APP_VERSION=1.0.0
```



- Pipeline CI/CD Backend : Exécution Réussie pour les Tests, la Construction et le Déploiement -

5.4.2. Frontend

Le pipeline du frontend effectue :

1. **Tests unitaires avec Jest** : Génération d'un rapport de couverture.

Project Management Tool

2. **Compilation Angular** : Production des fichiers statiques.
3. **Dockerisation et Push** : Création de l'image Docker et publication sur Docker Hub.

Extrait du Workflow Frontend :

```
name: Project-management-tool Frontend Pipeline

jobs:
  angular:
    name: Jest Unit Tests and Build
    runs-on: ubuntu-latest
    strategy:
      matrix:
        node-version: [20.x]

    steps:

      # 4. Run tests frontend
      - name: Run unit tests with Jest and generate coverage
        run: |
          cd project-management-frontend
          npm run test -- --watch=false --coverage

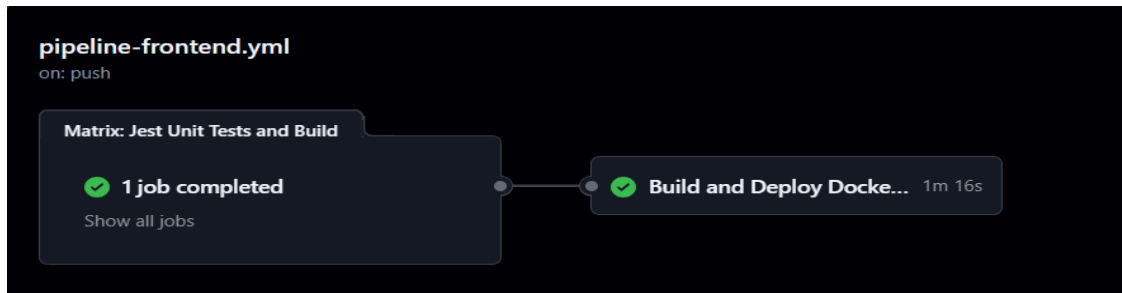
      # 5. Upload test coverage report
      - name: Upload coverage report
        uses: actions/upload-artifact@v3
        with:
          name: jest-coverage-report
          path: project-management-frontend/coverage/

  build-image-and-deploy:
    name: Build and Deploy Docker Image
    runs-on: ubuntu-latest
    needs: angular
    steps:
      #Login to DockerHub
      - name: Login to DockerHub
        uses: docker/login-action@v3
        with:
          username: ${ secrets.DOCKERHUB_USERNAME }
          password: ${ secrets.DOCKERHUB_TOKEN }

      #Build & Push to DockerHub
      - name: Build & Push to DockerHub
        uses: docker/build-push-action@v5
        with:
          context: project-management-frontend
          file: docker/frontend/Dockerfile
          push: true
```

Project Management Tool

```
tags: ${{ secrets.DOCKERHUB_USERNAME }}/  
pmt-frontend:latest,${{ secrets.DOCKERHUB_USERNAME }}/  
pmt-frontend:${{ steps.extract_version.outputs.VERSION }}
```



- Pipeline CI/CD Frontend : Exécution Réussie pour les Tests, la Construction et le Déploiement -

5.4.3. Déploiement et Gestion des Images Docker sur Docker Hub

Cette capture montre les dépôts Docker Hub contenant les images Docker pour le frontend et le backend du projet. Les dépôts, visibles publiquement, permettent de faciliter le déploiement et la distribution des images :

- **wa2l99/pmt-frontend** : Image Docker pour le frontend de l'application.
- **wa2l99/pmt-api** : Image Docker pour le backend de l'application.

Ces images ont été générées et poussées automatiquement via les pipelines CI/CD configurés dans GitHub Actions.

dockerhub Explore Repositories Organizations Usage				
wa2l99 Search by repository name All content Create a repository				
Name	Last Pushed	Contains	Visibility	Scout
wa2l99/pmt-frontend	6 days ago	IMAGE	Public	Inactive
wa2l99/pmt-api	6 days ago	IMAGE	Public	Inactive

- Dépôts Docker Hub pour le Frontend et le Backend -

5.5. Conclusion

Cette phase garantit un processus automatisé de bout en bout, allant de la compilation du code aux tests, à la Dockerisation, et au déploiement. L'utilisation de Docker et GitHub Actions permet de maintenir des environnements homogènes.

6. Conclusion et Évaluation

6.1. Conclusion





Ce projet de **Project Management Tool (PMT)** incarne une solution complète et intuitive pour la gestion collaborative de projets. En s'appuyant sur des technologies modernes telles que **Angular** pour le frontend et **Spring Boot** pour le backend, combinées à une base de données robuste comme **PostgreSQL**, l'application offre une expérience utilisateur fluide et optimisée.

Chaque étape de développement a été réalisée dans le respect des meilleures pratiques en matière de génie logiciel, notamment :




- Une conception bien réfléchie avec des entités clairement définies et des relations adaptées aux besoins fonctionnels.
- La mise en œuvre rigoureuse des user stories spécifiées, offrant des fonctionnalités comme la gestion des projets, des membres et des tâches.
- Une couverture de tests élevée pour garantir la fiabilité, avec des tests unitaires et d'intégration exécutés automatiquement via des **pipelines CI/CD**.
- Une architecture industrialisée grâce à l'intégration de **Docker**, **Flyway**, et la publication des artefacts sur **Docker Hub**.

En conclusion, le PMT est conçu pour s'adapter aux besoins des équipes modernes en leur offrant un outil fiable, évolutif et orienté collaboration.

6.2. Évaluation de la Conformité à la Demande

Compétence	Évaluation	Statut
- Développer les fonctionnalités du logiciel en modélisant un domaine métier, et en intégrant des composants externes afin d'améliorer la qualité du code et faciliter les développements futurs.	- L'application est fonctionnelle et suit les recommandations techniques.	
	- Le schéma de base de données est complet et sans incohérence relationnelle.	
	- Les frameworks Angular et Spring sont utilisés.	
- Automatiser la construction de la solution logicielle en configurant les chaînes de	- Les tests sont écrits tant pour le frontend que le backend avec une couverture minimum de 60% tant des instructions que des branches..	

Project Management Tool

build et l'exécution des tests unitaires, fonctionnels et d'intégration afin de préparer le déploiement continu du logiciel.	- Une pipeline est mise en œuvre et son fichier de configuration est disponible. Elle permet d'exécuter les tests.	
- Industrialiser le développement du logiciel à l'aide d'outils d'automatisation et le documenter en décrivant le processus de déploiement de manière à faire évoluer les logiciels développés et minimiser les erreurs de manipulation par les tiers.	- Le backend et le frontend sont dockerisés.	
	- La pipeline permet de pusher les images sur Docker HuB.	
	- Le fichier readme.md fournit la procédure de déploiement de l'application.	