Université Cadi Ayyad
Faculté des Sciences Semlalia - Marrakech
Département : Informatique
Master Ingénierie des Systèmes d'Information

Rapport de Projet du Module de Opérations de développement (DevOps)l

Conception UML du projet de gestion des tâches Kanban

Le 21 Mars 2025

Réalisé par :

- BEGNOUG Cheikh El Hadrami
- DAKIR ALLAH Abderrahman
- HADADIA Saad
- MAKTOUB Wiam

Encadré par :

- PR. ASRI Hiba

Année universitaire: 2024/2025

Résumé

Ce projet consiste en la conception et le développement d'une application de gestion de projets basée sur la méthodologie Kanban. L'objectif est de fournir un outil simple et intuitif pour organiser les tâches, faciliter la collaboration en équipe, et suivre l'avancement des projets en temps réel. L'application est développée avec des technologies modernes, notamment Next.js pour le front-end, Express.js pour le back-end, et MongoDB pour la gestion des données.

Les principales fonctionnalités incluent la création et la gestion des tâches, la collaboration en temps réel, et une interface utilisateur ergonomique. Ce rapport présente l'analyse des besoins, la conception UML, les choix techniques, ainsi que les tests et la validation du système. Enfin, des perspectives d'amélioration sont proposées pour évoluer vers une solution encore plus performante et adaptée aux besoins des utilisateurs.

Contents

Résumé 1

1 Introduction 5
- 1.1 Champ d'application . . . 5
- 1.2 Public cible . . . 5
- 1.3 Contexte et justification . . . 5
- 1.4 Objectifs du rapport . . . 5

2 Analyse des Besoins 6
- 2.1 Besoins Fonctionnels . . . 6
  - 2.1.1 Inscription et Authentification des Utilisateurs : . . . 6
  - 2.1.2 Gestion des Tableaux Kanban : . . . 6
  - 2.1.3 Gestion des Tâches : . . . 6
  - 2.1.4 Collaboration en Temps Réel : . . . 6
  - 2.1.5 Visualisation des Projets : . . . 6
  - 2.1.6 Gestion des Utilisateurs et des Rôles : . . . 6
- 2.2 Besoins Non Fonctionnels . . . 7
  - 2.2.1 Performance : . . . 7
  - 2.2.2 Sécurité : . . . 7
  - 2.2.3 Compatibilité Multi-Plateforme : . . . 7
  - 2.2.4 Évolutivité : . . . 7
  - 2.2.5 Interface Utilisateur (UI/UX) : . . . 7
  - 2.2.6 Disponibilité : . . . 7
  - 2.2.7 Maintenabilité : . . . 7

3 Présentation du Système 8
- 3.1 Architecture . . . 8
- 3.2 Fonctionnalités Clés . . . 8
- 3.3 Technologies Utilisées . . . 8

4 Diagrammes UML 8
- 4.1 Diagramme des Cas d'Utilisation (Use Case Diagram) . . . 8
  - 4.1.1 Acteurs . . . 9
  - 4.1.2 Cas d'Utilisation . . . 10
  - 4.1.3 Relations entre Acteurs et Cas d'Utilisation . . . 10
  - 4.1.4 Conclusion . . . 10
- 4.2 Diagramme de Classes (Class Diagram) . . . 11
  - 4.2.1 Classes Principales . . . 12
  - 4.2.2 Relations entre les Classes . . . 13
  - 4.2.3 Conclusion . . . 13
- 4.3 Diagramme de Séquence (Sequence Diagram) . . . 13
  - 4.3.1 Diagramme de Séquence pour l'Inscription . . . 14
  - 4.3.2 Description des Étapes . . . 14
  - 4.3.3 Conclusion . . . 15
  - 4.3.4 Diagramme de Séquence pour la Connexion . . . 15

4.3.5 Description des Étapes 16
4.3.6 Conclusion 17

5 Test et Validation **17**
5.1 Test Cases 17
5.1.1 Unit Tests (Tests Unitaires) 17
5.1.2 Integration Tests (Tests d'Intégration) 17
5.1.3 System Tests (Tests Système) 18
5.2 Validation 18
5.2.1 Validation Fonctionnelle 18
5.2.2 Validation Non Fonctionnelle 18
5.2.3 Validation par les Utilisateurs 18
5.3 Outils de Test 18

Conclustion **19**

List of Figures

1 Diagramme des Cas d'Utilisation 9
2 Diagramme de Classes 11
3 Diagramme de Séquence pour l'Inscription 14
4 Diagramme de Séquence pour la Connexion 16

1 Introduction

L'objectif de ce projet est de concevoir et de développer une application de gestion de projets basée sur la méthodologie Kanban. Cette application vise à simplifier et à optimiser la gestion des tâches et des projets, que ce soit pour des utilisateurs individuels ou des équipes collaboratives. Le Kanban, une méthode visuelle de gestion du travail, permet de suivre l'avancement des tâches à travers des colonnes représentant les différentes étapes du workflow (par exemple : "À faire", "En cours", "Terminé"). Cette approche favorise une meilleure organisation, une transparence accrue et une productivité améliorée.

1.1 Champ d'application

L'application sera principalement destinée à la gestion de projets, qu'ils soient personnels, professionnels ou académiques. Elle permettra aux utilisateurs de créer des tableaux Kanban personnalisés, d'ajouter des tâches, de les attribuer à des membres d'équipe, de suivre leur progression et de visualiser l'état global du projet en un coup d'œil. Cette solution s'adaptera à divers contextes, que ce soit pour des projets de petite envergure ou des initiatives plus complexes nécessitant une collaboration entre plusieurs parties prenantes.

1.2 Public cible

Le public cible de cette application est large et varié, incluant :

- **Professionnels :** Gestionnaires de projet, équipes de développement, consultants, etc., qui ont besoin d'un outil simple et efficace pour organiser leurs tâches et collaborer avec leurs collègues.
- **Étudiants :** Individus ou groupes travaillant sur des projets académiques ou des travaux d'équipe.
- **Particuliers :** Personnes souhaitant organiser leurs tâches personnelles, leurs objectifs ou leurs projets quotidiens de manière structurée.

1.3 Contexte et justification

Dans un monde où la gestion du temps et des ressources est de plus en plus critique, les outils de gestion de tâches jouent un rôle essentiel pour améliorer la productivité et la collaboration. Cependant, de nombreuses solutions existantes sont soit trop complexes, soit trop limitées pour répondre aux besoins spécifiques des utilisateurs. Cette application Kanban se distingue par sa simplicité d'utilisation, son interface intuitive et sa flexibilité, permettant aux utilisateurs de s'adapter rapidement à l'outil sans avoir besoin de formations approfondies.

De plus, avec l'essor des méthodologies agiles et des pratiques DevOps, la gestion visuelle des tâches devient un élément clé pour les équipes cherchant à optimiser leurs processus de développement et de déploiement. Cette application s'inscrit donc dans une démarche moderne, en alignant les principes du Kanban avec les besoins actuels des professionnels et des particuliers.

1.4 Objectifs du rapport

Ce rapport a pour objectif de présenter la conception détaillée de l'application, en mettant l'accent sur :

- Les besoins fonctionnels et non fonctionnels de l'application.
- Les diagrammes UML qui illustrent la structure et le fonctionnement du système.
- Les outils et technologies utilisés pour le développement.

- Les pratiques DevOps qui seront mises en œuvre pour assurer un déploiement continu et une maintenance efficace.

Ce projet vise à offrir une solution innovante et accessible pour la gestion de projets, en combinant la simplicité du Kanban avec les avantages des technologies modernes.

2 Analyse des Besoins

2.1 Besoins Fonctionnels

Les besoins fonctionnels décrivent les fonctionnalités que l'application doit offrir pour répondre aux attentes des utilisateurs. Voici les principales fonctionnalités identifiées :

2.1.1 Inscription et Authentification des Utilisateurs :

- Les utilisateurs doivent pouvoir créer un compte et se connecter de manière sécurisée.
- Possibilité de réinitialiser le mot de passe en cas d'oubli.

2.1.2 Gestion des Tableaux Kanban :

- Créer, modifier et supprimer des tableaux Kanban.
- Chaque tableau doit représenter un projet ou un ensemble de tâches.

2.1.3 Gestion des Tâches :

- Ajouter, modifier et supprimer des tâches dans un tableau Kanban.
- Déplacer les tâches entre les colonnes (par exemple : "À faire", "En cours", "Terminé").
- Attribuer des tâches à des membres de l'équipe.

2.1.4 Collaboration en Temps Réel :

- Permettre à plusieurs utilisateurs de travailler sur le même tableau Kanban simultanément.
- Notifier les utilisateurs des mises à jour (par exemple : une tâche déplacée ou modifiée).

2.1.5 Visualisation des Projets :

- Afficher les tableaux Kanban de manière claire et intuitive.
- Fournir une vue d'ensemble de l'avancement des tâches (par exemple : pourcentage de tâches terminées).

2.1.6 Gestion des Utilisateurs et des Rôles :

- Définir des rôles (par exemple : administrateur, membre) avec des permissions spécifiques.
- Inviter des utilisateurs à rejoindre un tableau Kanban.

2.2 Besoins Non Fonctionnels

Les besoins non fonctionnels décrivent les caractéristiques techniques et les contraintes que l'application doit respecter pour garantir une expérience utilisateur optimale. Voici les principaux besoins non fonctionnels :

2.2.1 Performance :

- L'application doit être réactive, avec un temps de chargement minimal, même avec un grand nombre de tâches ou d'utilisateurs.
- Prise en charge de la collaboration en temps réel sans latence perceptible.

2.2.2 Sécurité :

- Sécurisation des données utilisateur via des protocoles de chiffrement (par exemple : HTTPS).
- Authentification sécurisée avec des tokens JWT (JSON Web Tokens).
- Protection contre les attaques courantes (par exemple : injection SQL, XSS).

2.2.3 Compatibilité Multi-Plateforme :

- L'application doit être accessible sur différents appareils (ordinateurs, tablettes, smartphones).
- Prise en charge des navigateurs modernes (Chrome, Firefox, Safari, Edge).

2.2.4 Évolutivité :

- L'architecture doit permettre une montée en charge pour supporter un nombre croissant d'utilisateurs et de projets.
- Possibilité d'ajouter de nouvelles fonctionnalités sans perturber le fonctionnement existant.

2.2.5 Interface Utilisateur (UI/UX) :

- Interface intuitive et facile à utiliser, même pour les utilisateurs non techniques.
- Design moderne et responsive, adapté à différents écrans.

2.2.6 Disponibilité :

- L'application doit être disponible 24/7, avec un taux de disponibilité de 99,9 %.
- Mise en place de mécanismes de sauvegarde et de récupération en cas de panne.

2.2.7 Maintenabilité :

- Code bien documenté et structuré pour faciliter les mises à jour et la maintenance.
- Utilisation de pratiques DevOps pour assurer un déploiement continu et une intégration fluide.

3 Présentation du Système

L'application de gestion de projets Kanban est une solution web moderne conçue pour simplifier la gestion des tâches et améliorer la collaboration entre les utilisateurs. Elle repose sur une architecture client-serveur, avec un front-end interactif et un back-end robuste pour gérer les données et les opérations métier.

3.1 Architecture

- **Front-end** : Développé avec **Next.js**, il offre une interface utilisateur réactive et intuitive, compatible avec tous les appareils.
- **Back-end** : Basé sur **Express.js**, il gère les API, l'authentification et la logique métier.
- **Base de données** : Utilisation de **MongoDB** pour stocker les données de manière flexible et scalable.

3.2 Fonctionnalités Clés

- Tableaux Kanban personnalisables pour organiser les tâches.
- Collaboration en temps réel entre les membres de l'équipe.
- Sécurité renforcée avec authentification JWT et chiffrement des données.

3.3 Technologies Utilisées

- **Front-end** : Next.js, React, Tailwind CSS.
- **Back-end** : Express.js, Node.js.
- **Base de données** : MongoDB.
- **Hébergement** : Vercel ou AWS.

4 Diagrammes UML

Les diagrammes UML (Unified Modeling Language) sont utilisés pour modéliser la structure et le comportement de l'application de gestion de projets Kanban. Voici les diagrammes principaux :

4.1 Diagramme des Cas d'Utilisation (Use Case Diagram)

Le diagramme des cas d'utilisation illustre les interactions entre les acteurs (utilisateurs) et le système. Il met en évidence les principales fonctionnalités de l'application.

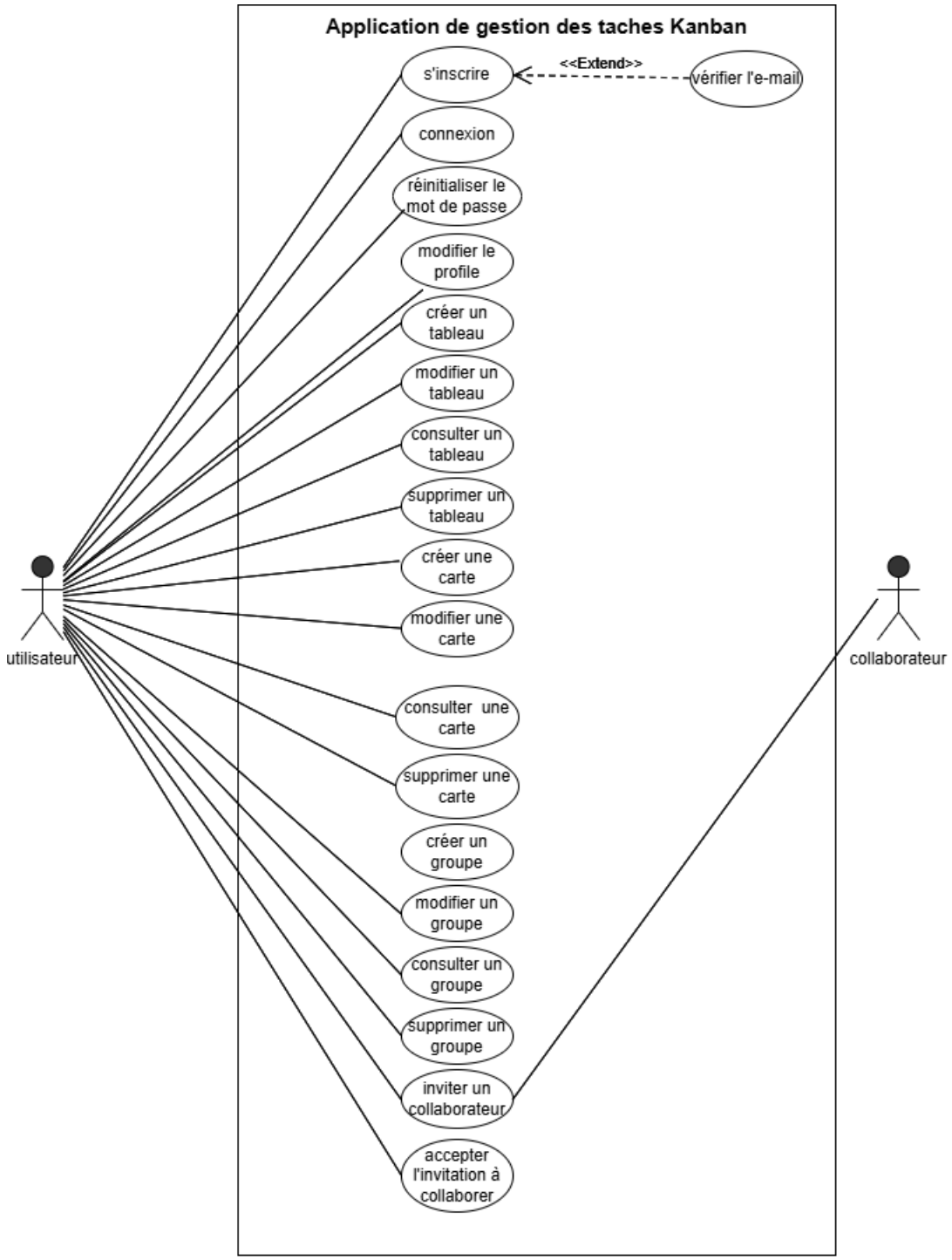

Figure 1: Diagramme des Cas d'Utilisation

4.1.1 Acteurs

Le diagramme identifie deux acteurs principaux :

- **Utilisateur** : L'utilisateur principal qui interagit avec le système pour gérer ses tâches et tableaux Kanban.
- **Collaborateur** : Un utilisateur invité à collaborer sur un tableau Kanban. Il dispose de permissions spécifiques pour interagir avec les tableaux partagés.

4.1.2 Cas d'Utilisation

Les cas d'utilisation représentent les fonctionnalités offertes par l'application. Voici une description des principaux cas d'utilisation :

- **S'inscrire** : Permet à un nouvel utilisateur de créer un compte.
- **Connexion** : Permet à un utilisateur de se connecter à son compte.
- **Réinitialiser le mot de passe** : Permet à un utilisateur de réinitialiser son mot de passe en cas d'oubli.
- **Modifier le profil** : Permet à l'utilisateur de mettre à jour ses informations personnelles.
- **Créer un tableau** : Permet à l'utilisateur de créer un nouveau tableau Kanban.
- **Modifier un tableau** : Permet à l'utilisateur de modifier les propriétés d'un tableau existant.
- **Consulter un tableau** : Permet à l'utilisateur de visualiser un tableau Kanban.
- **Supprimer un tableau** : Permet à l'utilisateur de supprimer un tableau existant.
- **Créer une carte** : Permet à l'utilisateur de créer une nouvelle carte (tâche) dans un tableau.
- **Modifier une carte** : Permet à l'utilisateur de mettre à jour les informations d'une carte existante.
- **Consulter une carte** : Permet à l'utilisateur de visualiser les détails d'une carte.
- **Supprimer une carte** : Permet à l'utilisateur de supprimer une carte existante.
- **Créer un groupe** : Permet à l'utilisateur de créer un groupe de cartes.
- **Modifier un groupe** : Permet à l'utilisateur de mettre à jour les informations d'un groupe.
- **Consulter un groupe** : Permet à l'utilisateur de visualiser les membres d'un groupe.
- **Supprimer un groupe** : Permet à l'utilisateur de supprimer un groupe existant.
- **Inviter un collaborateur** : Permet à l'utilisateur d'inviter un autre utilisateur à collaborer sur un tableau.
- **Accepter l'invitation à collaborer** : Permet à un utilisateur invité d'accepter l'invitation et de rejoindre un tableau partagé.

4.1.3 Relations entre Acteurs et Cas d'Utilisation

- L'**Utilisateur** peut effectuer toutes les actions liées à la gestion de son compte, des tableaux, des cartes, et des groupes.
- Le **Collaborateur** peut interagir avec les tableaux partagés, mais avec des permissions limitées (par exemple, il ne peut pas supprimer un tableau).

4.1.4 Conclusion

Ce diagramme des cas d'utilisation fournit une vue d'ensemble des fonctionnalités de l'application et des interactions entre les acteurs et le système. Il sert de base pour la conception et le développement des différentes composantes de l'application.

4.2 Diagramme de Classes (Class Diagram)

Le diagramme de classes décrit la structure statique du système en montrant les classes, leurs attributs, méthodes, et les relations entre elles. Voici une description des classes principales et de leurs interactions.

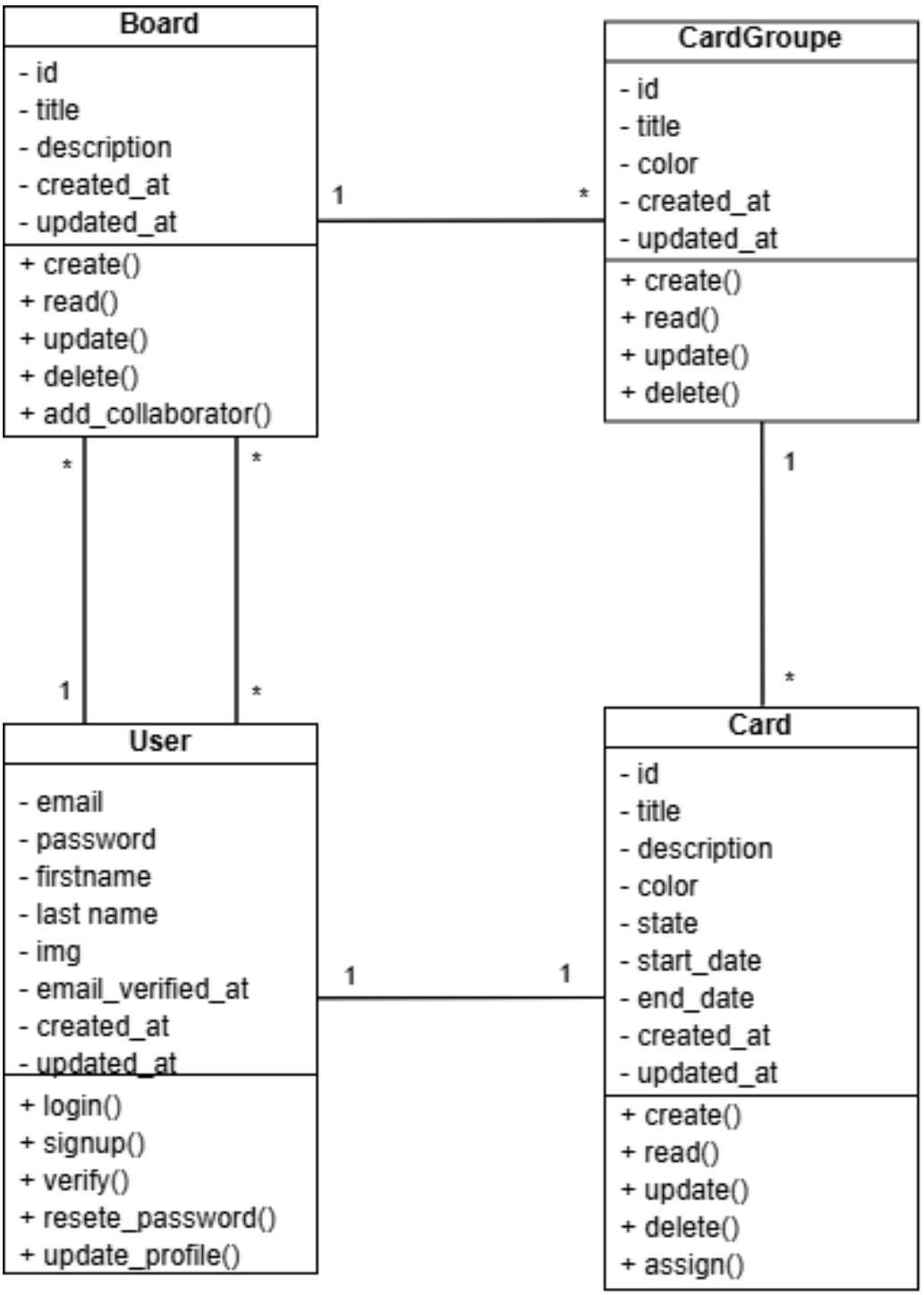

Figure 2: Diagramme de Classes

4.2.1 Classes Principales

- **User (Utilisateur)** :
  - **Attributs** :
    * `email` : Adresse e-mail de l'utilisateur.
    * `password` : Mot de passe de l'utilisateur.
    * `firstname` : Prénom de l'utilisateur.
    * `last_name` : Nom de famille de l'utilisateur.
    * `img` : Image de profil de l'utilisateur.
    * `email_verified_at` : Date de vérification de l'e-mail.
    * `created_at` : Date de création du compte utilisateur.
    * `updated_at` : Date de la dernière mise à jour du compte utilisateur.
  - **Méthodes** :
    * `login()` : Permet à l'utilisateur de se connecter.
    * `signup()` : Permet à l'utilisateur de créer un compte.
    * `verify()` : Vérifie l'e-mail de l'utilisateur.
    * `reset_password()` : Permet à l'utilisateur de réinitialiser son mot de passe.
    * `update_profile()` : Permet à l'utilisateur de mettre à jour son profil.
- **Board (Tableau)** :
  - **Attributs** :
    * `id` : Identifiant unique du tableau.
    * `title` : Titre du tableau.
    * `description` : Description du tableau.
    * `created_at` : Date de création du tableau.
    * `updated_at` : Date de la dernière mise à jour du tableau.
  - **Méthodes** :
    * `create()` : Crée un nouveau tableau.
    * `read()` : Lit les informations du tableau.
    * `update()` : Met à jour les informations du tableau.
    * `delete()` : Supprime le tableau.
    * `add_collaborator()` : Ajoute un collaborateur au tableau.
- **CardGroupe (Groupe de Cartes)** :
  - **Attributs** :
    * `id` : Identifiant unique du groupe de cartes.
    * `title` : Titre du groupe de cartes.
    * `color` : Couleur du groupe de cartes.
    * `created_at` : Date de création du groupe de cartes.
    * `updated_at` : Date de la dernière mise à jour du groupe de cartes.
  - **Méthodes** :

    - `create()` : Crée un nouveau groupe de cartes.
    - `read()` : Lit les informations du groupe de cartes.
    - `update()` : Met à jour les informations du groupe de cartes.
    - `delete()` : Supprime le groupe de cartes.

- **Card (Carte)** :
    - **Attributs** :
        - `id` : Identifiant unique de la carte.
        - `title` : Titre de la carte.
        - `description` : Description de la carte.
        - `color` : Couleur de la carte.
        - `state` : État de la carte (par exemple, "À faire", "En cours", "Terminé").
        - `start_date` : Date de début de la tâche.
        - `end_date` : Date de fin de la tâche.
        - `created_at` : Date de création de la carte.
        - `updated_at` : Date de la dernière mise à jour de la carte.
    - **Méthodes** :
        - `create()` : Crée une nouvelle carte.
        - `read()` : Lit les informations de la carte.
        - `update()` : Met à jour les informations de la carte.
        - `delete()` : Supprime la carte.
        - `assign()` : Assigne la carte à un utilisateur.

4.2.2 Relations entre les Classes

- **Board** et **CardGroupe** : Un tableau peut contenir plusieurs groupes de cartes (relation 1 à plusieurs).
- **CardGroupe** et **Card** : Un groupe de cartes peut contenir plusieurs cartes (relation 1 à plusieurs).
- **User** et **Board** :
    - Un utilisateur peut créer et gérer plusieurs tableaux (relation 1 à plusieurs).
    - Un utilisateur peut contribuer dans plusieurs tableaux (relation plusieurs à plusieurs).
- **User** et **Card** : Un utilisateur peut être assigné à plusieurs cartes (relation 1 à plusieurs).

4.2.3 Conclusion

Ce diagramme de classes fournit une vue détaillée de la structure du système, en montrant les classes, leurs attributs, méthodes, et les relations entre elles. Il sert de base pour la conception et l'implémentation de l'application.

4.3 Diagramme de Séquence (Sequence Diagram)

Les diagrammes de séquence suivants illustrent les interactions entre les acteurs et les composants du système lors de l'inscription d'un utilisateur. Ces diagrammes constituent la première étape de la conception.

4.3.1 Diagramme de Séquence pour l'Inscription

Le diagramme de séquence suivant illustre le processus d'inscription d'un utilisateur, en mettant en évidence les interactions entre le frontend, le backend, la base de données (DB), et le service d'e-mail.

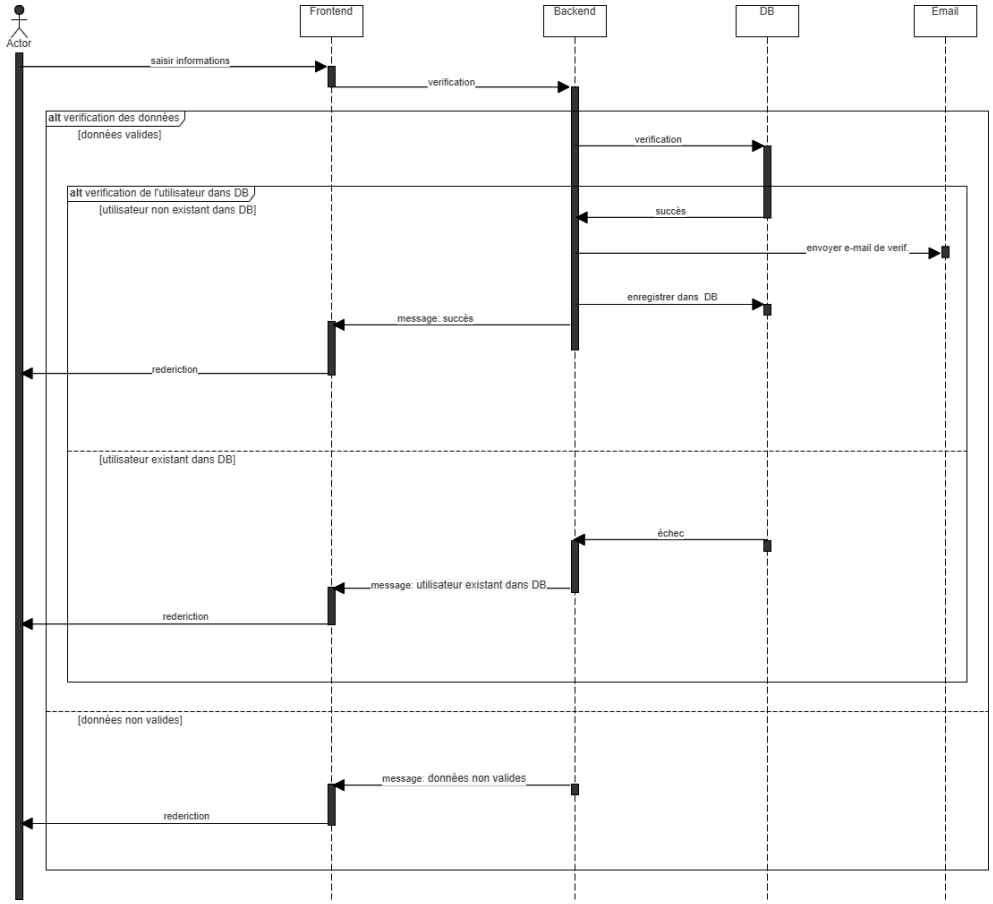

Figure 3: Diagramme de Séquence pour l'Inscription

4.3.2 Description des Étapes

1. **Saisie des Informations** :
   - L'utilisateur saisit ses informations (nom, prénom, e-mail, mot de passe) dans le formulaire d'inscription.
   - L'utilisateur clique sur "S'inscrire".
2. **Envoi des Données au Backend** :
   - Le frontend envoie les données saisies au backend pour traitement.
3. **Validation des Données** :
   - Le backend valide les données (vérifie que l'e-mail est valide, que le mot de passe est conforme, etc.).
4. **Vérification de l'E-mail dans la Base de Données** :
   - Le backend interroge la base de données pour vérifier si l'e-mail est déjà utilisé.
5. **Réponse de la Base de Données** :

- La base de données renvoie une réponse :
  - Si l'e-mail n'existe pas, elle renvoie un message de succès.
  - Si l'e-mail existe déjà, elle renvoie un message d'erreur.

6. **Envoi d'un E-mail de Confirmation** :
   - Si l'e-mail n'existe pas, le backend envoie une demande au service d'e-mail pour envoyer un e-mail de confirmation.
7. **Confirmation de l'Envoi d'E-mail** :
   - Le service d'e-mail renvoie un message de succès après l'envoi de l'e-mail.
8. **Réponse au Frontend** :
   - Le backend renvoie une réponse au frontend :
     - Si tout s'est bien passé, il renvoie un message de succès.
     - Si une erreur est survenue, il renvoie un message d'erreur.
9. **Affichage du Résultat** :
   - Le frontend affiche un message à l'utilisateur :
     - En cas de succès : "Inscription réussie ! Vérifiez votre e-mail pour confirmer votre compte."
     - En cas d'erreur : "Erreur : Cet e-mail est déjà utilisé." ou "Erreur : Données non valides."

4.3.3 Conclusion

Ce diagramme de séquence décrit le processus complet d'inscription d'un utilisateur, en mettant en évidence les interactions entre les différents composants du système. Dans ce qui suit, nous détaillerons les autres cas d'utilisation, tels que la vérification d'e-mail, la réinitialisation du mot de passe, et la modification du profil.

4.3.4 Diagramme de Séquence pour la Connexion

Les diagrammes de séquence suivants illustrent les interactions entre les acteurs et les composants du système lors de la connexion d'un utilisateur. Ces diagrammes constituent une étape clé de la conception.

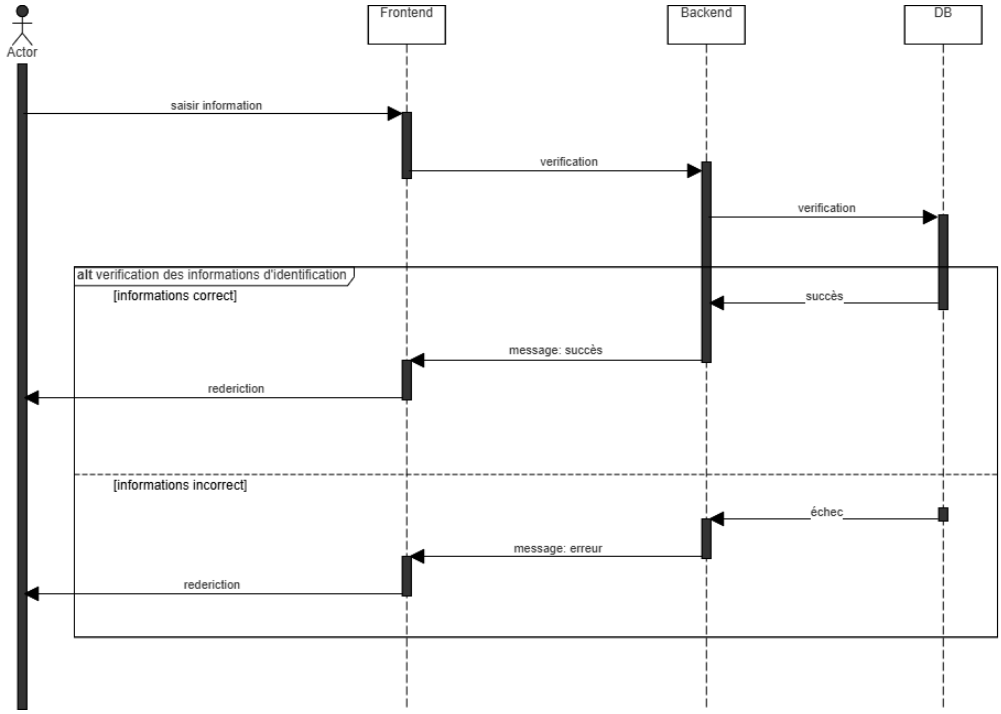

Figure 4: Diagramme de Séquence pour la Connexion

4.3.5 Description des Étapes

1. **Saisie des Informations** :
   - L'utilisateur saisit ses informations de connexion (e-mail et mot de passe) dans le formulaire de connexion.
   - L'utilisateur clique sur "Se connecter".
2. **Envoi des Données au Backend** :
   - Le frontend envoie les données saisies au backend pour traitement.
3. **Validation des Données** :
   - Le backend valide les données (vérifie que l'e-mail et le mot de passe sont valides).
4. **Vérification des Informations dans la Base de Données** :
   - Le backend interroge la base de données pour vérifier si les informations d'identification sont correctes.
5. **Réponse de la Base de Données** :
   - La base de données renvoie une réponse :
     - Si les informations sont correctes, elle renvoie un message de succès.
     - Si les informations sont incorrectes, elle renvoie un message d'erreur.
6. **Réponse au Frontend** :
   - Le backend renvoie une réponse au frontend :
     - Si les informations sont correctes, il renvoie un message de succès et **un token d'authentification**.

- Si les informations sont incorrectes, il renvoie un message d'erreur.

7. **Affichage du Résultat** :
   - Le frontend affiche un message à l'utilisateur :
     - En cas de succès : "Connexion réussie ! Redirection en cours..."
     - En cas d'erreur : "Erreur : E-mail ou mot de passe incorrect."

4.3.6 Conclusion

Ce diagramme de séquence décrit le processus complet de connexion d'un utilisateur, en mettant en évidence les interactions entre les différents composants du système. **En cas de succès, le backend envoie un token d'authentification au frontend**, ce qui permet à l'utilisateur de rester connecté et d'accéder aux fonctionnalités protégées. Dans ce qui suit, nous détaillerons les autres cas d'utilisation, tels que la vérification d'e-mail, la réinitialisation du mot de passe, et la modification du profil.

5 Test et Validation

5.1 Test Cases

Pour garantir la qualité et la fiabilité de l'application, plusieurs types de tests seront effectués :

5.1.1 Unit Tests (Tests Unitaires)

Les tests unitaires vérifient le bon fonctionnement des composants individuels du système. Voici quelques exemples de tests unitaires :

- **Test de l'inscription** : Vérifier que l'utilisateur peut créer un compte avec des informations valides.
- **Test de l'authentification** : Vérifier que l'utilisateur peut se connecter avec des identifiants corrects.
- **Test de la création de tâches** : Vérifier qu'une tâche peut être créée avec un titre et une description valides.
- **Test de la modification de tâches** : Vérifier qu'une tâche peut être modifiée et que les changements sont correctement enregistrés.

5.1.2 Integration Tests (Tests d'Intégration)

Les tests d'intégration vérifient que les différents modules du système fonctionnent correctement ensemble. Voici quelques exemples :

- **Test de l'intégration utilisateur-tâche** : Vérifier qu'un utilisateur peut créer une tâche et qu'elle apparaît dans le tableau Kanban.
- **Test de la collaboration en temps réel** : Vérifier que les modifications apportées par un utilisateur sont visibles en temps réel par les autres utilisateurs.
- **Test de la base de données** : Vérifier que les données sont correctement stockées et récupérées depuis la base de données.

5.1.3 System Tests (Tests Système)

Les tests système vérifient que l'ensemble du système fonctionne comme prévu. Voici quelques exemples :

- **Test de performance** : Vérifier que l'application reste réactive même avec un grand nombre d'utilisateurs et de tâches.
- **Test de sécurité** : Vérifier que les données utilisateur sont protégées et que l'authentification est sécurisée.
- **Test de compatibilité** : Vérifier que l'application fonctionne correctement sur différents navigateurs et appareils.

5.2 Validation

La validation du système consiste à s'assurer que l'application répond aux exigences fonctionnelles et non fonctionnelles définies dans le cahier des charges. Voici les étapes de validation :

5.2.1 Validation Fonctionnelle

- **Validation des cas d'utilisation** : Tous les cas d'utilisation (inscription, authentification, gestion des tâches, etc.) doivent être testés et validés.
- **Validation des workflows** : Les workflows principaux (création de tâches, déplacement de tâches, collaboration en temps réel) doivent fonctionner sans erreur.

5.2.2 Validation Non Fonctionnelle

- **Performance** : L'application doit répondre rapidement, même sous charge élevée.
- **Sécurité** : Les données utilisateur doivent être protégées contre les accès non autorisés.
- **Compatibilité** : L'application doit fonctionner sur tous les navigateurs et appareils cibles.
- **Disponibilité** : L'application doit être disponible 24/7 avec un taux de disponibilité de 99,9 %.

5.2.3 Validation par les Utilisateurs

- **Tests utilisateurs** : Des tests utilisateurs seront réalisés pour recueillir des feedbacks sur l'ergonomie et la facilité d'utilisation.
- **Corrections et améliorations** : Les retours des utilisateurs seront utilisés pour apporter des corrections et des améliorations à l'application.

5.3 Outils de Test

Les outils suivants seront utilisés pour les tests et la validation :

- **Jest** : Pour les tests unitaires et d'intégration.
- **Cypress** : Pour les tests end-to-end (E2E).
- **Postman** : Pour tester les API.
- **JMeter** : Pour les tests de performance.
- **SonarQube** : Pour l'analyse de la qualité du code.

Conclusion

Ce projet de conception et de développement d'une application de gestion de projets Kanban a permis de créer une solution moderne et efficace pour la gestion des tâches et la collaboration en équipe. Grâce à une analyse approfondie des besoins, une conception UML rigoureuse, et une implémentation basée sur des technologies robustes (Next.js, Express.js, MongoDB), l'application répond aux exigences fonctionnelles et non fonctionnelles définies dans le cahier des charges. Les principaux points forts incluent une interface utilisateur intuitive, une collaboration en temps réel, et une sécurité renforcée via l'authentification JWT.

En conclusion, cette application offre une solution adaptée aux besoins des utilisateurs, tout en ouvrant la voie à des améliorations futures telles que l'ajout de fonctionnalités avancées, l'optimisation des performances, et le développement d'une version mobile. Ce projet démontre l'importance d'une approche méthodique et de l'utilisation de technologies modernes pour répondre aux défis de la gestion de projets.