Université de Pau et des Pays de l'Adour

Rapport projet Génie Logiciel
ProjectPulse

Auteur :
Arruzas Simon
Calmels Corentin

Décembre 2024

Table des matières

1 Introduction

Dans le cadre des unités d'enseignement de génie logiciel, un projet terminal est proposé pour appliquer les connaissances théoriques acquises.

Ce projet de génie logiciel est un outil de gestion de projet basé sur le Web, conçu pour aider les petites et moyennes équipes à planifier, suivre et gérer leurs projets efficacement. Il intègre plusieurs fonctionnalités essentielles pour la gestion des tâches, la collaboration en équipe, le suivi du temps, et bien plus encore.

L'objectif principal de ce projet est de fournir une plateforme robuste et simple à utiliser, tout en respectant les principes du génie logiciel pour garantir la qualité du code et la fiabilité de l'application.

Caractéristiques du projet

- **Authentification et autorisation :** Un système de connexion sécurisé avec des rôles distincts (administrateur, chef de projet, membre).
- **Tableau de bord :** Un tableau de bord centralisé pour voir les projets en cours, les tâches à venir, et les indicateurs clés de performance.
- **Gestion de projets et de tâches :** Création et gestion de projets, gestion des tâches avec priorités, échéances et états.
- **Collaboration d'équipe :** Un système de messagerie ou de commentaires pour améliorer la communication.
- **Suivi du temps :** Enregistrement du temps passé sur chaque tâche et génération de rapports détaillés.
- **Notifications et alertes :** Système de notifications pour alerter les utilisateurs sur les échéances et les mises à jour importantes.
- **Conception réactive :** L'application sera accessible sur tous les appareils (ordinateurs, tablettes, smartphones).
- **Déploiement et DevOps :** Utilisation de Docker et des pipelines CI/CD pour un déploiement efficace sur le cloud.

Les principes d'AdonisJS

Nous avons choisi d'utiliser le framework AdonisJS pour réaliser ce projet en raison de ses nombreux avantages et fonctionnalités.
AdonisJS est un framework Node.js qui facilite le développement d'applications web performantes et sécurisées. Il est basé sur le principe de "convention over configuration", ce qui permet de se concentrer sur la logique métier plutôt que sur la configuration du framework.

Voici quelques principes clés d'AdonisJS :

- **Structure MVC :** AdonisJS suit le modèle MVC (Model-View-Controller) qui sépare la logique de l'application en trois parties distinctes pour faciliter la gestion et la maintenance du code.
- **ORM Eloquent :** L'ORM d'AdonisJS facilite la gestion des bases de données avec des modèles clairs, permettant une gestion fluide des données.
- **Authentification intégrée :** Il dispose d'un système d'authentification prêt à l'emploi, ce qui permet de gérer facilement les sessions et les rôles des utilisateurs.
- **Sécurité renforcée :** AdonisJS inclut des mécanismes de sécurité comme la protection CSRF, le cryptage des mots de passe, et la gestion des erreurs de manière centralisée.
- **Facilité de test :** Le framework intègre des outils pour effectuer des tests unitaires et d'intégration facilement.
- **Documentation riche :** AdonisJS offre une documentation complète et bien structurée pour accompagner les développeurs tout au long de leur projet.

2 Conception

2.1 Besoins et exigences

Besoins des utilisateurs :

- **Possibilité de s'authentifier :** L'utilisateur peut se créer un compte, se connecter et gérer son profil, il possède un rôle basé sur sa place dans la hiérarchie des contributeurs au projet.
- **Accès à un tableau de bord :** L'utilisateur à accès à un tableau de bord fournissant un aperçu des projets en cours, des tâches et échéances à venir.
- **Gestion de projets et de tâches :** L'utilisateur peut créer et gérer ses projets, il a accès au nom du projet, sa description, la date de création et de complétion ainsi que le nom des différents membres de l'équipe. Il peut aussi créer des tâches, les assigner à des membres de l'équipe, leur attribuer une priorité, une échéance et un état.
- **Collaboration en équipe :** L'utilisateur peut envoyer et lire des commentaires postés sur des tâches, qui seront accessible par toute l'équipe d'un projet. Il peut aussi envoyer des messages privés à un autre utilisateur.
- **Suivi du temps :** L'utilisateur peut enregistrer le temps passé sur une tâche, et générer des rapports détaillés sur le temps passé par chaque
- **Notifications et alertes :** L'utilisateur peut recevoir des notifications et des alertes sur les échéances et les mises à jour importantes concernant les projets et les tâches.
- **Conception réactive :** L'utilisateur peut consulter l'application sur tous les appareils (ordinateurs, tablettes, smartphones) grâce à une conception réactive.

Exigences Fonctionnelles et non-fonctionnelles du projet :

Exigences Fonctionnelles :

- **Authentification et autorisation des utilisateurs :**
  - L'utilisateur peut s'inscrire en fournissant un nom d'utilisateur, une adresse e-mail et un mot de passe.
  - L'utilisateur peut se connecter en utilisant son nom d'utilisateur et son mot de passe.
  - L'utilisateur peut modifier son profil en mettant à jour ses informations personnelles.
  - L'utilisateur peut se déconnecter de son compte à tout moment.
  - Les profils des utilisateurs ainsi que leurs informations sont stockés en toute sécurité dans une base de données.
- **Tableau de bord :**
  - L'utilisateur peut consulter un tableau de bord affichant les projets en cours, les tâches à venir, les échéances à venir, les activités récentes et les tâches en attente.
- **Gestion de projets et de tâches :**
  - L'utilisateur peut créer un nouveau projet.
  - L'utilisateur peut gérer un projet et consulter ses informations.
  - L'utilisateur peut gérer les tâches.
  - L'utilisateur peut classer les tâches par étiquettes.
  - L'utilisateur peut définir des dépendances entre les tâches.
  - L'utilisateur peut supprimer un projet ou une tâche existante.
- **Collaboration d'équipe :**
  - L'utilisateur peut envoyer des messages à un autre utilisateur pour discuter des projets et des tâches.
  - L'utilisateur peut envoyer des commentaires sur une tâche.
  - L'utilisateur peut lire les commentaires présents sur une tâche.
  - L'utilisateur, appartenant à une équipe peut accéder aux commentaires de membres de son équipe.
  - Les messages sont stockés dans une base de données.
- **Suivi du temps :**
  - L'utilisateur peut enregistrer le teps passé sur une tâche.
  - L'utilisateur peut consulter le temps passé sur une tâche.
  - L'utilisateur peut générer des rapports détaillés sur la répartition du temps entre les tâches.
  - L'utilisateur peut générer des rapports montrant la répartition du temps entre les projets.
- **Notifications et alertes :**

— L'utilisateur est notifié lorsqu'une échéance est proche.
— L'utilisateur est notifié lorsqu'une tâche est assignée.
— L'utilisateur est notifié lorsqu'un commentaire est posté sur une tâche.
— L'utilisateur est notifié lors qu'une mise à jour importante est effectuée.

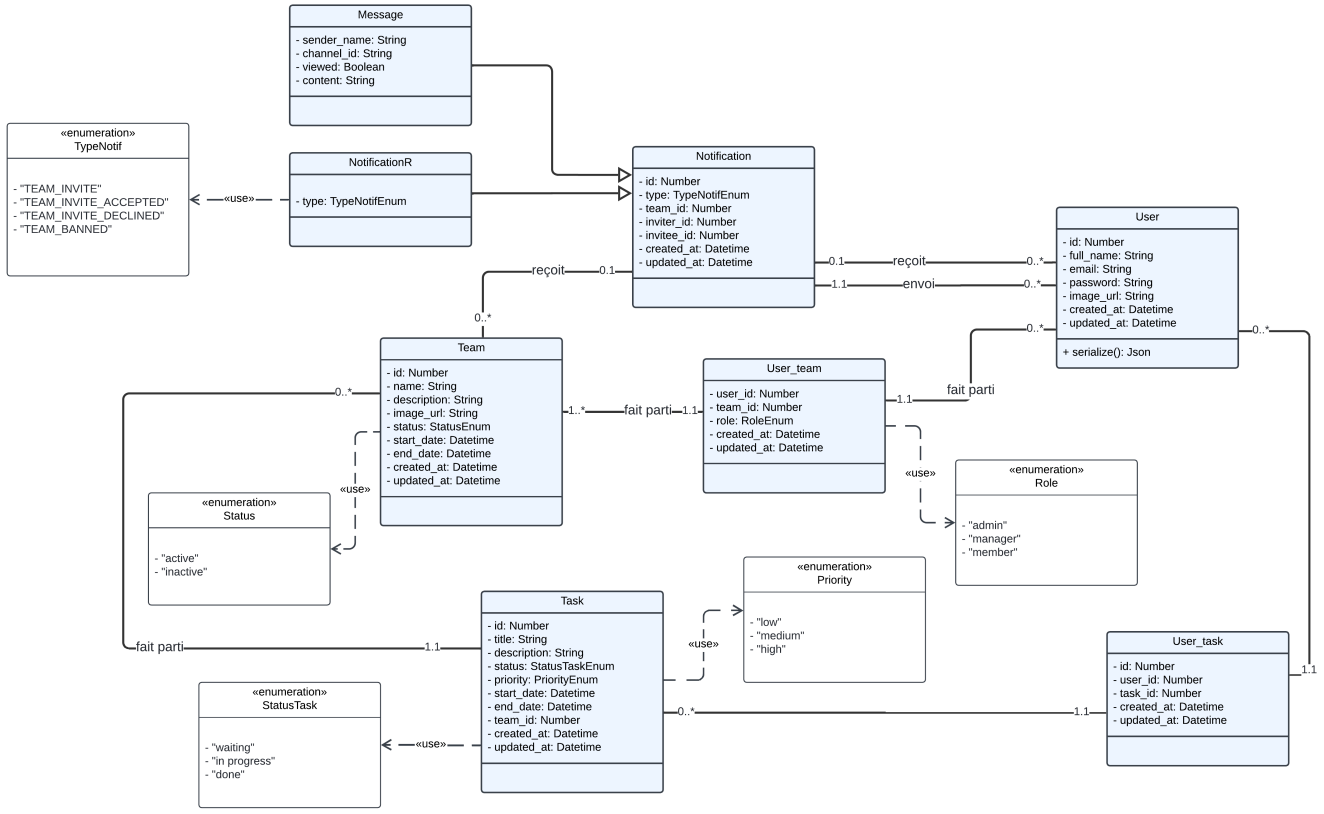
Exigences Non-Fonctionnelles :

— **Sécurité :** L'outil doit protéger les données des utilisateurs, la connexion au site doit se faire à l'aide d'un mot de passe.
— **Performance :** L'outil doit pouvoir gérer plusieurs utilisateurs connectés simultanément, tout en maintenant un temps de réponse minimal, il doit aussi pouvoir charger les informations sur les projets rapidement et sans ralentissement perceptible pour l'utilisateur.
— **Fiabilité :** L'outil doit assurer une disponibilité élevée afin de minimiser ses temps d'arrêts. Des sauvegardes automatiques doivent être effectués régulièrement pour éviter toute perte d'informations.
— **Extensibilité :** L'outil doit être conçu afin de pouvoir s'adapter à une augmentation du nombre d'utilisateurs sans nécessité une refonte majeure. Il doit être possible d'ajouter des fonctionnalités supplémentaires sans perturber le fonctionnement de services déjà existants.
— **Maintenance :** L'outil doit avoir un code source structuré et documenté pour permettre une maintenance facilitée de l'outil ainsi que ses potentielles évolutions futures. L'outil doit être conçu avec des tests afin de faciliter la correction de bugs.
— **Compatibilité :** L'outil doit être compatible avec tous les navigateurs web ainsi qu'être disponible sur divers systèmes d'exploitation.
— **Ergonomie :** L'outil doit avoir une interface intuitive et facile à prendre en main, les utilisateurs doivent pouvoir naviguer entre les fonctionnalités sans difficulté.
— **Réactivité :** L'outil possède une interface réactive avec des éléments restants fonctionnels et accessibles peu importe la plateforme ou l'appareil de l'utilisateur.

2.2 Diagrammes

2.2.1 Diagramme de classe

Le diagramme de classe représente la structure statique du système en montrant les classes du système, leurs attributset les relations entre les classes.



Il est important de noter que le diagramme de classe est une représentation simplifiée du système et ne montre pas tous les détails de l'implémentation.

2.2.2 Diagramme des cas d'utilisation

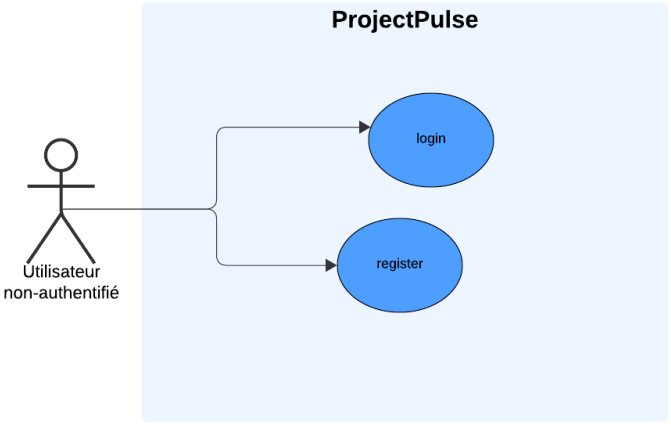


FIGURE 1 – Diagramme cas d'utilisation utilisateur non authentifié

Ce premier diagramme montre les cas d'utilisation de l'utilisateur non-authentifié. Il peut s'inscrire, se connecter et consulter la page d'accueil.

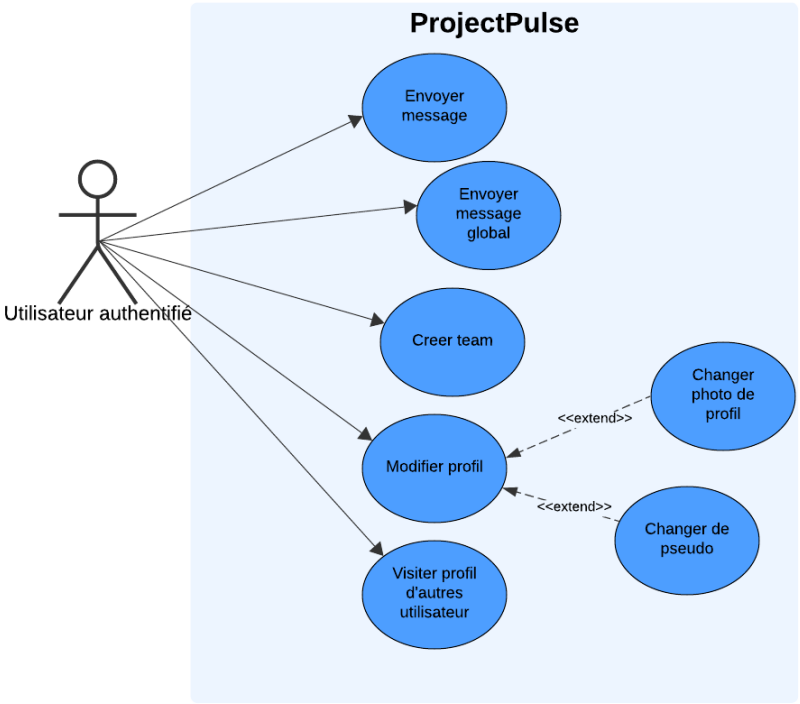


FIGURE 2 – Diagramme cas d'utilisation généraux

Ce deuxième diagramme montre les principaux cas d'utilisation de l'utilisateur authentifié. Il peut créer une team, envoyer un message à un utilisateur, envoyer un message global, modifier son profil et consulter celui des autres.

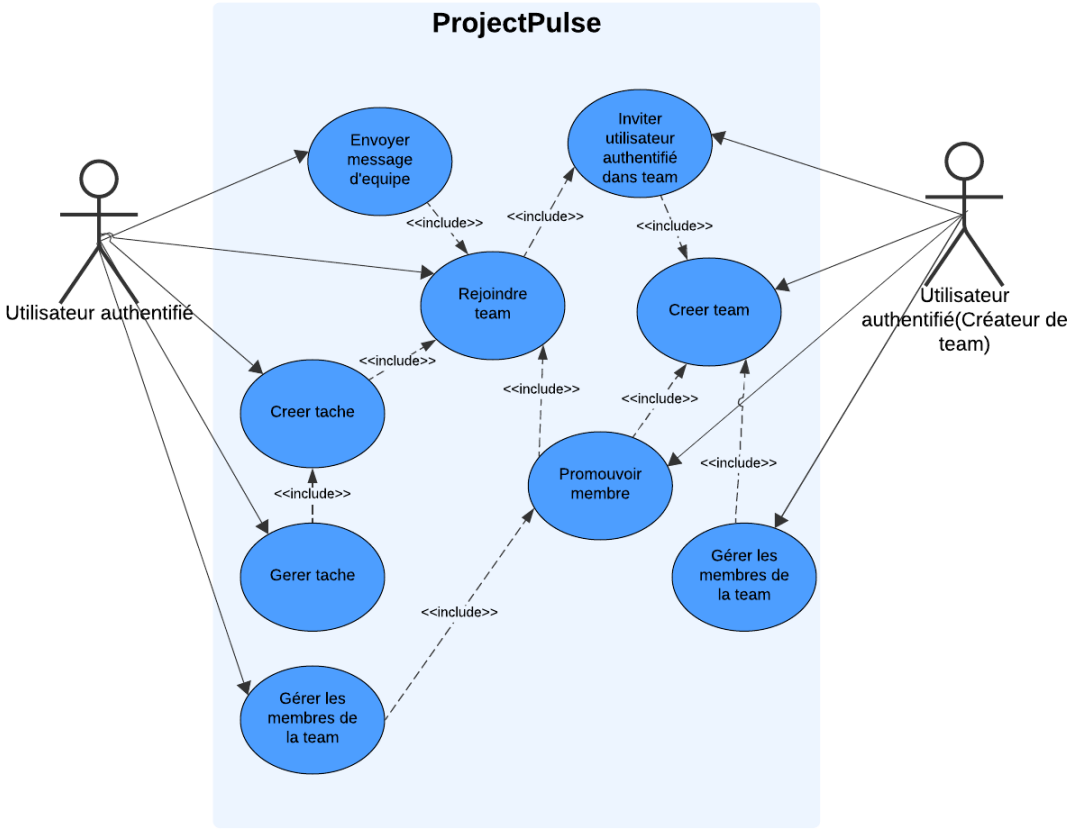


FIGURE 3 – Diagramme cas d'utilisation interaction team

Ce troisième diagramme montre les cas d'utilisation concernant les interactions par rapport aux teams. Un utilisateur qui crée une team en devient l'admin, il peut inviter des utilisateurs, les exclure, les promouvoir en tant que manager ou les rétrograder en tant que membre. Il peut aussi supprimer la team.
L'utilisateur invité peut accepter ou refuser l'invitation. S'il accepte, il devient membre de la team, il peut dans ce cas inviter d'autres utilisateurs, envoyer des messages de team, consulter les tâches. S'il est promu manager il peut gerer les membres de la team, les tâches et les projets. S'il est rétrogradé en tant que membre, il perd ces droits.

2.3 Design

Le design de l'application "ProjectPulse" a été conçues pour offrir une interface utilisateur intuitive et réactive. Voici quelques captures d'écran montrent les différentes vues de l'application, y compris le tableau de bord, la gestion des tâches, et les fonctionnalités de collaboration.

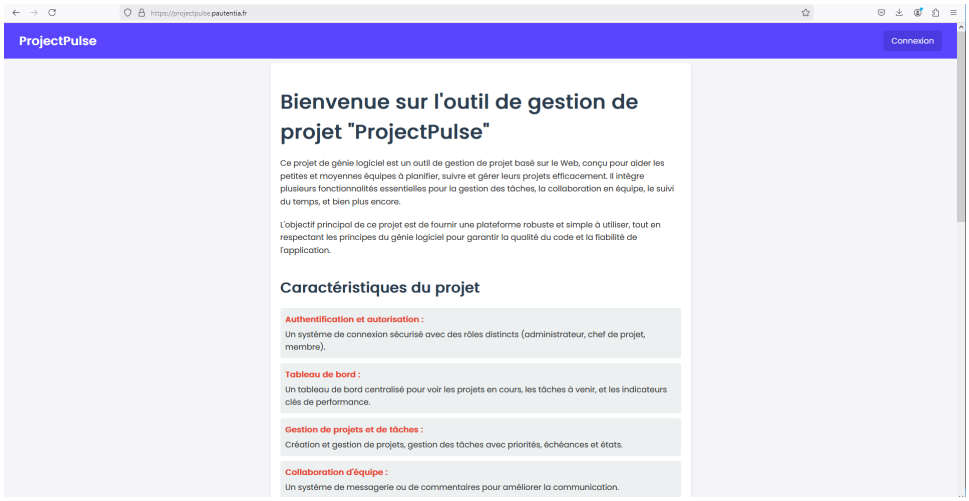


FIGURE 4 – Page d'accueil

FIGURE 5 – Page de connexion

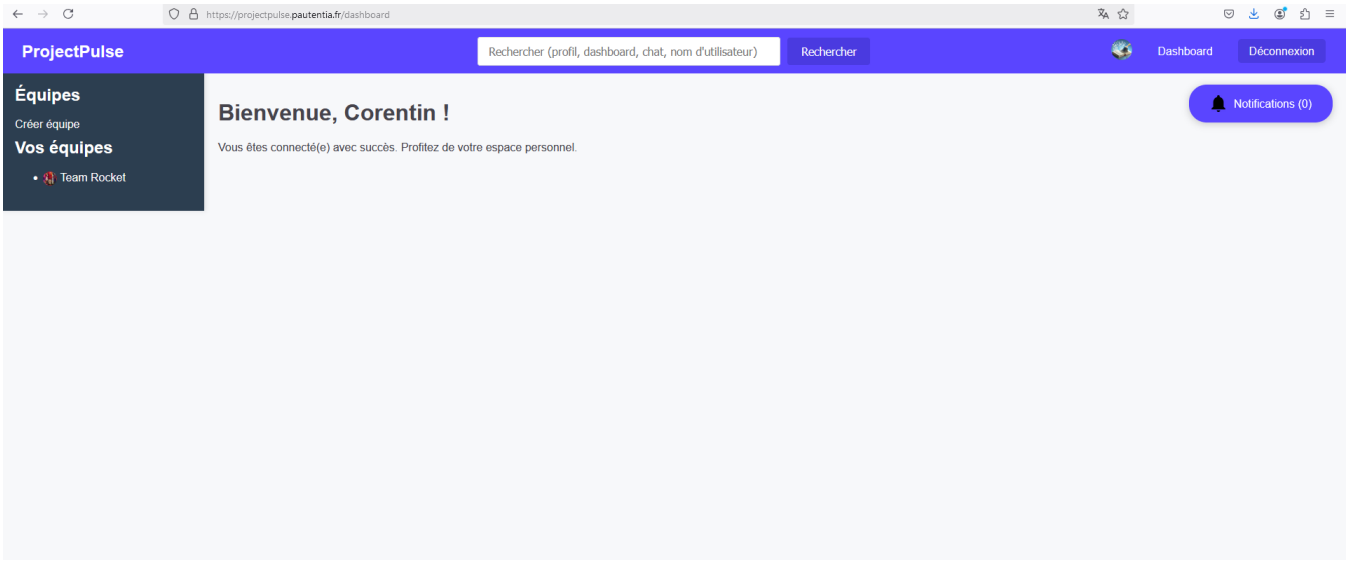


FIGURE 6 – Dashboard

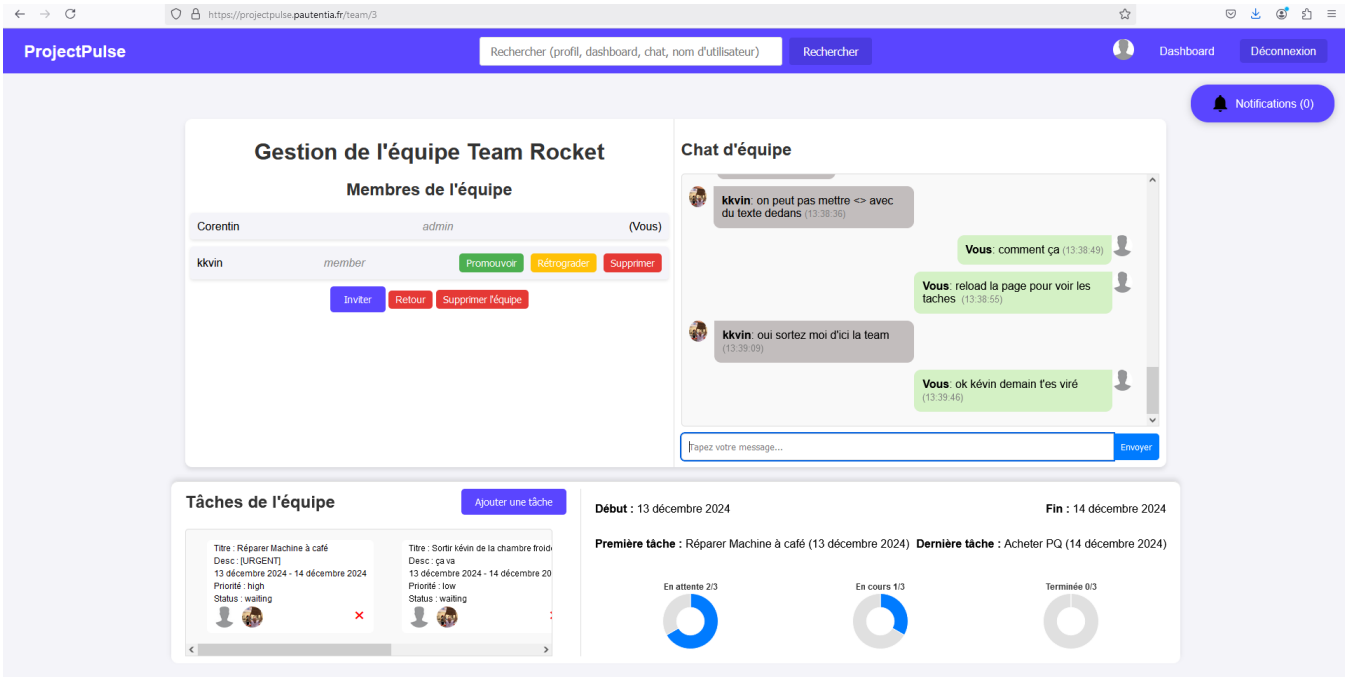


FIGURE 7 – Dashboard d'équipe en chat

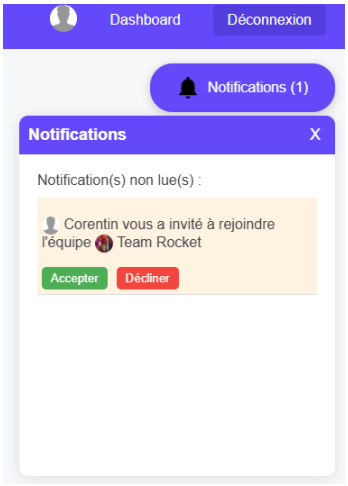


FIGURE 8 – Notification d'invitation à une équipe

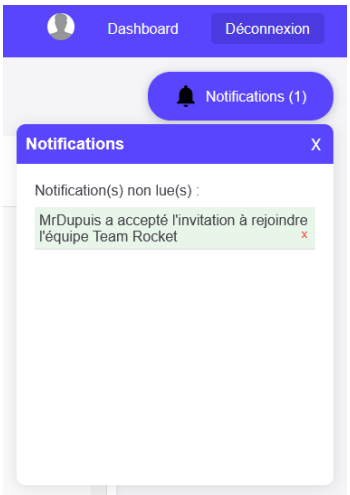


FIGURE 9 – Notification d'invitation acceptée à une équipe

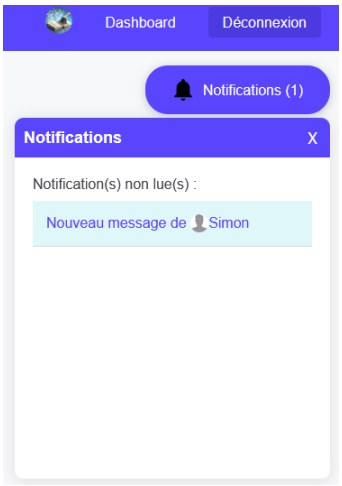


FIGURE 10 – Notification d'un message reçu

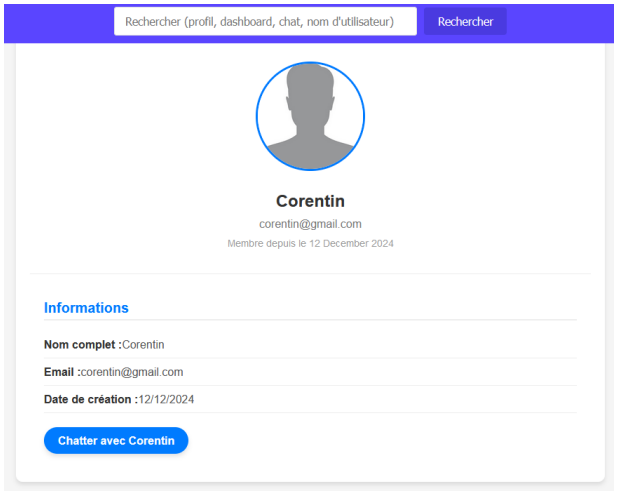


FIGURE 11 – Profil public

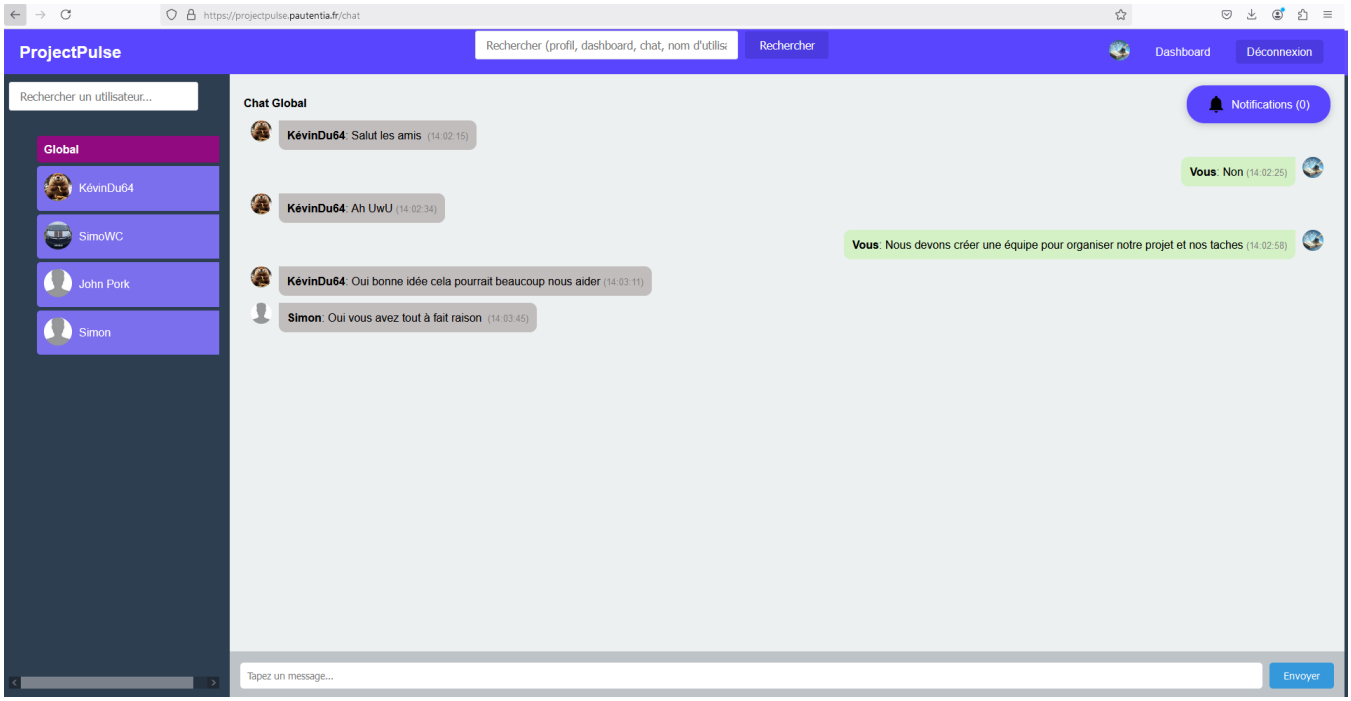


FIGURE 12 – Page de chat (chat global en l'occurence)

3 Implémentation

3.1 Fonctionnalités implémentées

Authentification et autorisation :

Nous avons utilisé le système d'authentification "Session" d'Adonijs, qui utilise les sessions pour stocker les informations de connexion de l'utilisateur.
Les autorisations dans la gestion des équipes sont gérées avec la table "users_team" qui contient les rôles des utilisateurs dans les équipes.

Tableau de bord :

Le tableau de bord affiche les équipes auxquelle l'utilisateur appartient, il peut en créer une nouvelle, lorsqu'il clique sur une équipe il est redirigé vers le dashboard de l'équipe, on peut voir les différents membres, leurs rôles, les tâches en cours, les tâches à venir, les tâches terminées, les messages de l'équipe.

Gestion de projets et de tâches :

L'utilisateur peut créer un projet, ajouter des tâches, les assigner à des membres de l'équipe, leur attribuer une priorité, une échéance et un état. Il peut aussi les supprimer.

Collaboration en équipe :

L'utilisateur peut envoyer des messages à un autre utilisateur, envoyer des messages globaux, consulter les messages reçus, envoyer des messages à une équipe, consulter les messages de l'équipe.

Personalisation du profil :

L'utilisateur peut modifier son profil, ajouter une photo de profil, changer son pseudo.

Notifications et alertes :

L'utilisateur est notifié lorsqu'il reçoit un message, lorsqu'il est invité dans une équipe, lorsqu'un utilisateur accepte une invitation de son equipe, lorsqu'un utilisateur est viré de son equipe.

3.2 Architecture

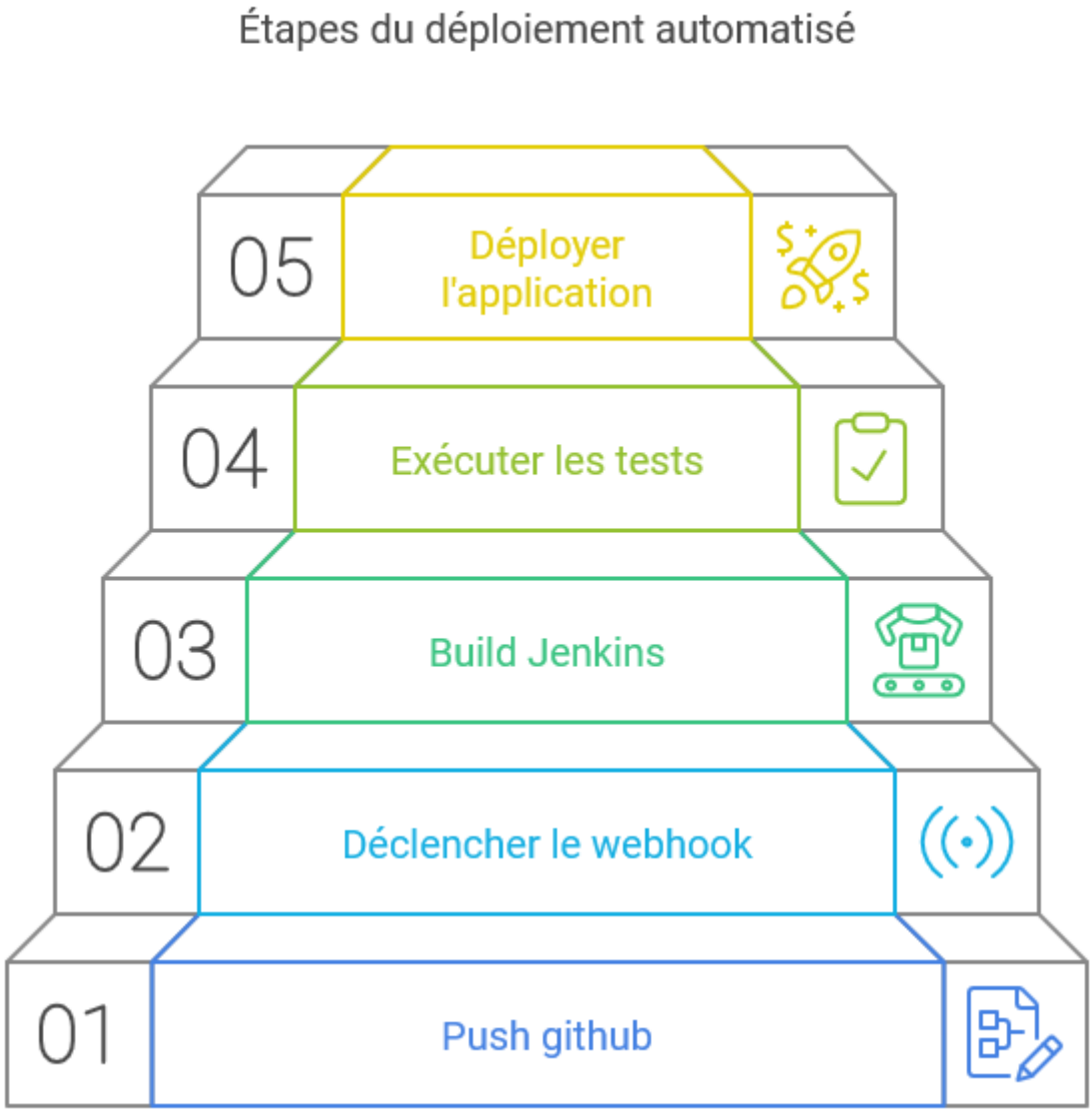
3.2.1 Modèle MVC

Nous avons utilisé l'architecture MVC (Modèle-Vue-Contrôleur) pour structurer l'application. Cette architecture permet de séparer la logique métier, la présentation et le contrôle des données, ce qui facilite la gestion et la maintenance du code.
Concretemnet il y a un fichier qui regroupe les routes (start/routes.ts), ces routes font

appel à des fonctions de contrôleurs qui vont interagir avec les modèles pour récupérer ou modifier des données, ces données sont ensuite envoyées à la vue (ou en JSON pour les requêtes API) pour être affichées.
Il y a aussi une partie services qui contient des fonctions utilitaires qui peuvent être utilisées par les contrôleurs.
Tous ces fichiers sont organisés dans des dossiers spécifiques, dans le dossier "app" de l'application.

4 Déploiement

Nous avons implémenté un pipeline CI/CD pour automatiser le processus de déploiement de l'application "ProjectPulse". Le pipeline est basé sur Jenkins et utilise Docker pour containeriser l'application.



Nous ne sommes donc à l'origine que de la première étape, le push github.

4.1 Docker

Docker est utilisé pour containeriser l'application "ProjectPulse", ce qui permet de garantir la cohérence de l'environnement de développement et de production.

Listing 1 – Dockerfile de ProjectPulse

```
# tape 1: Utiliser une image Node.js comme base
FROM node:22

# tape 2: Dfinir le rpertoire de travail dans le conteneur
WORKDIR /app

# tape 3: Copier les fichiers de votre projet dans le conteneur
COPY package*.json ./

# tape 4: Installer les dpendances
RUN npm install

# tape 5: Copier le reste du projet dans le conteneur
COPY . .

# tape 7: Dmarrer l'application AdonisJS
CMD ["npm", "run", "dev"]
```

Étant donné que nous sommes encore dans le développement, nous avons décidé de lancer l'application en mode "dev".

4.2 Jenkins

Jenkins est utilisé pour automatiser le processus de déploiement continu (CI/CD) de l'application.
Il est installé sur un serveur VPS lié au domaine de l'application, il est accessible via le sous-domaine "jenkins.pautentia.com".
Suite à l'éxécution du pipeline Jenkins, l'application est déployée, disponible sur le sous-domaine "projectpulse.pautentia.com".

Listing 2 – Pipeline Jenkins de ProjectPulse

```
pipeline {
    agent any
    environment {
        DOCKER_IMAGE = 'adonisjs-project'
        CONTAINER_NAME = 'adonisjs-container'
        ENV_FILE_PATH = '/var/secret/.env' // Chemin du fichier .env sur le
            ↪  VPS
        ENV_TEST_FILE_PATH = '/var/secret/.env.test' // Chemin du fichier .
            ↪ env.test sur le VPS
        UPLOAD_DIR = '/var/projectpulse/uploads' // Chemin du rpertoire d'
            ↪ uploads sur le VPS
    }
    stages {
        stage('Build Docker Image') {
            steps {
                script {
                    // Construire l'image Docker  partir du Dockerfile
                    sh 'docker build -t $DOCKER_IMAGE .'
                }
            }
        }

        stage('Run Tests') {
            steps {
                script {
                    // Excuter les tests dans un conteneur temporaire
                    sh "docker run --rm -v $UPLOAD_DIR:/app/uploads -v
                        ↪ $ENV_FILE_PATH:/app/.env -v $ENV_TEST_FILE_PATH:/
                        ↪ app/.env.test $DOCKER_IMAGE node ace test "
                }
            }
        }
```

```
        stage('Stop and Remove Old Container') {
            steps {
                script {
                    // Arrter et supprimer l'ancien conteneur s'il existe
                    sh "docker rm -f $CONTAINER_NAME || true"
                }
            }
        }

        stage('Run Docker Container') {
            steps {
                script {
                    // Monter le fichier .env dans le conteneur
                    sh "docker run -d --network=host --name $CONTAINER_NAME
                        ↪ -v $UPLOAD_DIR:/app/uploads -v $ENV_FILE_PATH:/
                        ↪ app/.env $DOCKER_IMAGE"
                }
            }
        }
    }
}
```

4.3 Github webhook

Le github webhook est utilisé ici pour déclencher le pipeline Jenkins à chaque fois qu'un commit est effectué sur la branche principale du dépôt Github.
La configuration est triviale, il suffit de spécifier l'URL du serveur Jenkins.