Chapitre 1

Analyses et spécification des besoins

1.1 Introduction

Après avoir effectué une exploration approfondie du projet dans son ensemble, nous commençons par la phase d'analyse des besoins et des spécifications techniques. Dans ce deuxième chapitre, nous développons une étude fonctionnelle du projet basée sur les diagrammes de cas d'utilisation et quelques diagrammes de séquences ainsi qu'une étude technique des technologies utilisées.

1.2 Étude fonctionnelle

L'étude fonctionnelle est une phase nécessaire pour expliquer le sujet et les tâches nécessaires à réaliser. Elle met en valeur la liste des acteurs, leurs tâches, et les interactions entre eux sous la forme des diagrammes UML.

1.2.1 Identification des acteurs de système

Dans notre projet centré sur l'apprentissage de la programmation avec des blocs pour contrôler des dispositifs, tels que le robot mBot, différents acteurs interviennent, chacun jouant un rôle déterminant :

Acteurs Principaux

- **Tuteur :** Ce rôle représente les enseignants ou les guides qui supervisent et soutiennent les apprenants dans leur apprentissage de la programmation par blocs pour contrôler le robot mBot.
- **Apprenant :** Les apprenants sont les utilisateurs finaux de la plateforme. Ils utilisent les outils de programmation par blocs pour créer des

programmes et contrôler le robot mBot dans le cadre de leurs activités d'apprentissage.

Acteurs Administratifs

- **Super Administrateur :** Ce rôle englobe l'entité ou l'organisation qui supervise l'ensemble de la plateforme. Le super administrateur a un contrôle total sur le système, y compris la gestion des utilisateurs et des paramètres globaux.
- **Administrateur Scolaire :** L'administrateur scolaire est responsable de la gestion des comptes des utilisateurs au sein de son établissement, de la surveillance des activités des apprenants et des tuteurs, ainsi que de l'accès aux rapports sur les performances des étudiants.
- **Administrateur du Robot :** Cet administrateur est chargé de la gestion des robots sur la plateforme. Il assure la disponibilité des robots pour les activités d'apprentissage, supervise les connexions matérielles, gère les mises à jour du firmware et peut ajouter de nouveaux dispositifs à contrôler. De plus, il a la capacité de créer des blocs personnalisés pour le contrôle de ces dispositifs, offrant ainsi une flexibilité accrue dans les activités d'apprentissage.

1.2.2 Identification des besoins fonctionnels

Création de Compte

Inscription :

- L'utilisateur peut s'inscrire sur la plateforme en fournissant son nom, son adresse e-mail et un mot de passe.
- L'utilisateur peut choisir son rôle lors de l'inscription, en sélectionnant soit le rôle d'apprenant, soit le rôle de tuteur.

Profil utilisateur

Fonctionnalités communes :

- **Accès au profil :** L'utilisateur peut accéder à son profil personnel pour gérer ses informations personnelles, où il peut créer des projets. Il a la possibilité de rejoindre une école, ce qui modifie son profil pour refléter son affiliation à cette école. Un bouton dans la barre de navigation lui permet de passer entre son profil personnel et son profil d'école.

Fonctionnalités spécifiques au rôle :

- **En tant que tuteur :**
  - **Gestion de projets personnels :** Le tuteur peut créer des projets personnels à partir de son profil. Les projets personnels peuvent être partagés avec d'autres utilisateurs via des invitations. Les utilisateurs invités peuvent accéder aux projets partagés et collaborer avec le tuteur.

  - **Gestion de projets d'école :** Le tuteur peut créer des méta-projets pour son école. Les méta-projets peuvent être collaboratifs ou non collaboratifs. Dans les méta-projets non collaboratifs, chaque participant crée son propre projet individuel sous le méta-projet. Dans les méta-projets collaboratifs, le tuteur peut créer des groupes de travail et partager le code avec chaque groupe. Le tuteur peut visualiser les projets des étudiants dans les méta-projets non collaboratifs et les projets des groupes dans les méta-projets collaboratifs. Le tuteur peut suivre les progrès des étudiants et fournir un soutien personnalisé.
- **En tant qu'apprenant :**
  - **Gestion de projets personnels :** L'apprenant peut créer des projets personnels à partir de son profil.
  - **Accès aux projets partagés :** L'apprenant peut rejoindre une école pour accéder aux projets partagés par son tuteur dans cette école.
  - **Rejoindre des projets d'école :** L'apprenant peut rejoindre des projets créés par son tuteur dans la même école.

Rôles et autorisations

- **En tant que Super Administrateur :**
  - **Gestion des utilisateurs :** Le super administrateur peut créer, modifier et supprimer des comptes utilisateur. Il peut gérer les rôles et les autorisations des utilisateurs.
  - **Configuration du système :** Le super administrateur peut configurer les paramètres globaux de la plateforme.
  - **Surveillance du système :** Le super administrateur peut surveiller les activités de la plateforme.
- **En tant qu'Administrateur Scolaire :**
  - **Gestion des comptes utilisateurs :** L'administrateur scolaire peut créer, modifier et supprimer des comptes utilisateur au sein de son établissement. Il peut gérer les rôles et les autorisations des utilisateurs au niveau de l'école.
- **En tant qu'Administrateur du Robot :**
  - **Gestion des robots :** L'administrateur du robot est responsable de la gestion des robots sur la plateforme. Il assure la disponibilité des robots pour les activités d'apprentissage.
  - **Maintenance des robots :** L'administrateur du robot supervise les connexions matérielles, gère les mises à jour du firmware et assure le bon fonctionnement des robots.
  - **Personnalisation des blocs :** L'administrateur du robot peut créer des blocs personnalisés pour le contrôle des dispositifs, offrant ainsi une flexibilité accrue dans les activités d'apprentissage impliquant les robots.

1.2.3 Identification des besoins non fonctionnels

- **Sécurité :**
  - **Sécurité des données sensibles :** Assurer la protection des données personnelles des utilisateurs, notamment les informations d'identification et les données d'apprentissage, en utilisant un stockage sécurisé et des protocoles de transmission chiffrés.
  - **Sécurité des interactions robot-utilisateur :** Garantir que les interactions entre les utilisateurs et les robots, telles que l'envoi de commandes et la réception de données, sont sécurisées pour prévenir les attaques et les manipulations externes.
- **Performances :**
  - **Temps de réponse des commandes :** Assurer des temps de réponse rapides entre l'envoi de commandes depuis l'interface utilisateur et l'exécution des actions par le robot mBot, pour une expérience utilisateur réactive et fluide.
  - **Scalabilité des fonctionnalités :** Concevoir le système pour qu'il puisse prendre en charge de manière transparente l'ajout de nouvelles fonctionnalités d'apprentissage et de contrôle des robots sans compromettre les performances globales.
- **Utilisabilité :**
  - **Interface de programmation intuitive :** Développer une interface de programmation visuelle conviviale, basée sur des blocs logiques, permettant aux utilisateurs de créer des programmes pour contrôler les dispositifs sans avoir besoin de compétences en programmation textuelle.
- **Fiabilité :**
  - **Stabilité des connexions robot-utilisateur :** Assurer la stabilité et la fiabilité des connexions entre l'interface utilisateur et le robot mBot, en minimisant les interruptions de connexion et en fournissant des mécanismes de reconnexion automatique.
  - **Gestion des erreurs :** Mettre en place des mécanismes de gestion des erreurs robustes pour détecter, signaler et corriger les anomalies dans l'exécution des programmes et les interactions avec le robot mBot.
- **Performance du système :**
  - **Compatibilité multiplateforme :** Assurer la compatibilité du système avec différents périphériques et systèmes d'exploitation, y compris les ordinateurs de bureau, les tablettes et les smartphones, pour offrir une expérience utilisateur homogène sur toutes les plateformes.

1.2.4 Les diagrammes de cas d'utilisation

Le diagramme de cas d'utilisation décrit le comportement du système du point de vue de l'utilisateur. Il divise ses fonctionnalités en cas d'utilisation qui permettent d'exprimer les besoins de ses utilisateurs, il est présenté par la figure

1 ci-dessous.

1.2.7 Suivi des Mises à Jour du Workspace

Ce diagramme de séquence illustre comment l'enseignant peut suivre les mises à jour du workspace de ses étudiants en temps réel, mettant en évidence la capacité d'IBLOCK à fournir un suivi personnalisé et une assistance en temps opportun.

1.2.8 Fonctionnement du Tableau de Bord Collaboratif

Ce diagramme de séquence illustre le fonctionnement du tableau de bord collaboratif d'IBLOCK, mettant en évidence les deux mécanismes d'enregistrement des données : l'enregistrement automatique via WebSocket et l'enregistrement manuel via une requête HTTP.

1.2.9 Envoi de Commandes au Robot

Ce diagramme de séquence illustre comment les commandes générées à partir des blocs Blockly sont envoyées au robot via WebSocket et un serveur Python dédié.

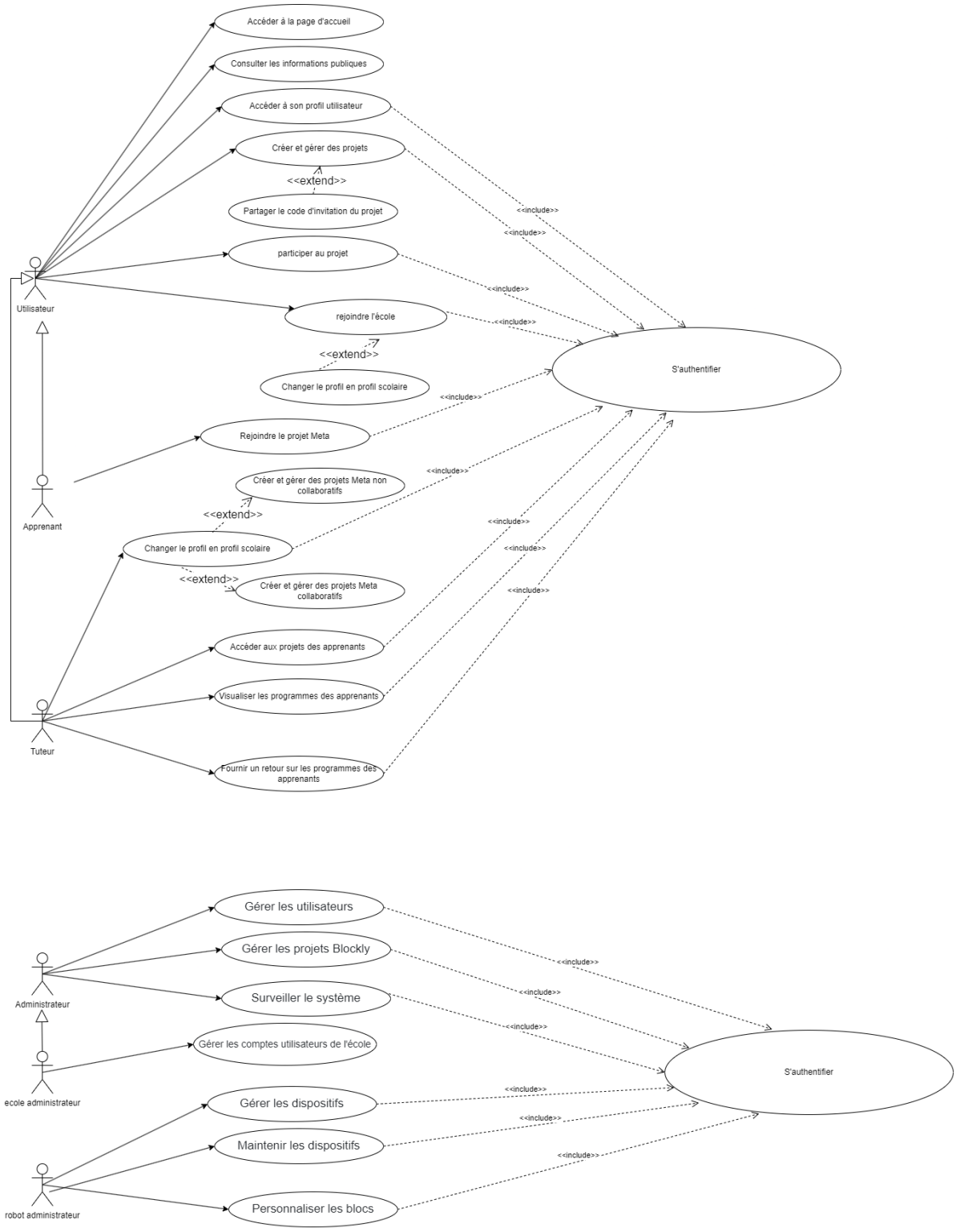

Figure 1: Diagramme globale des cas d'utilisation

1.2.5 Authentification

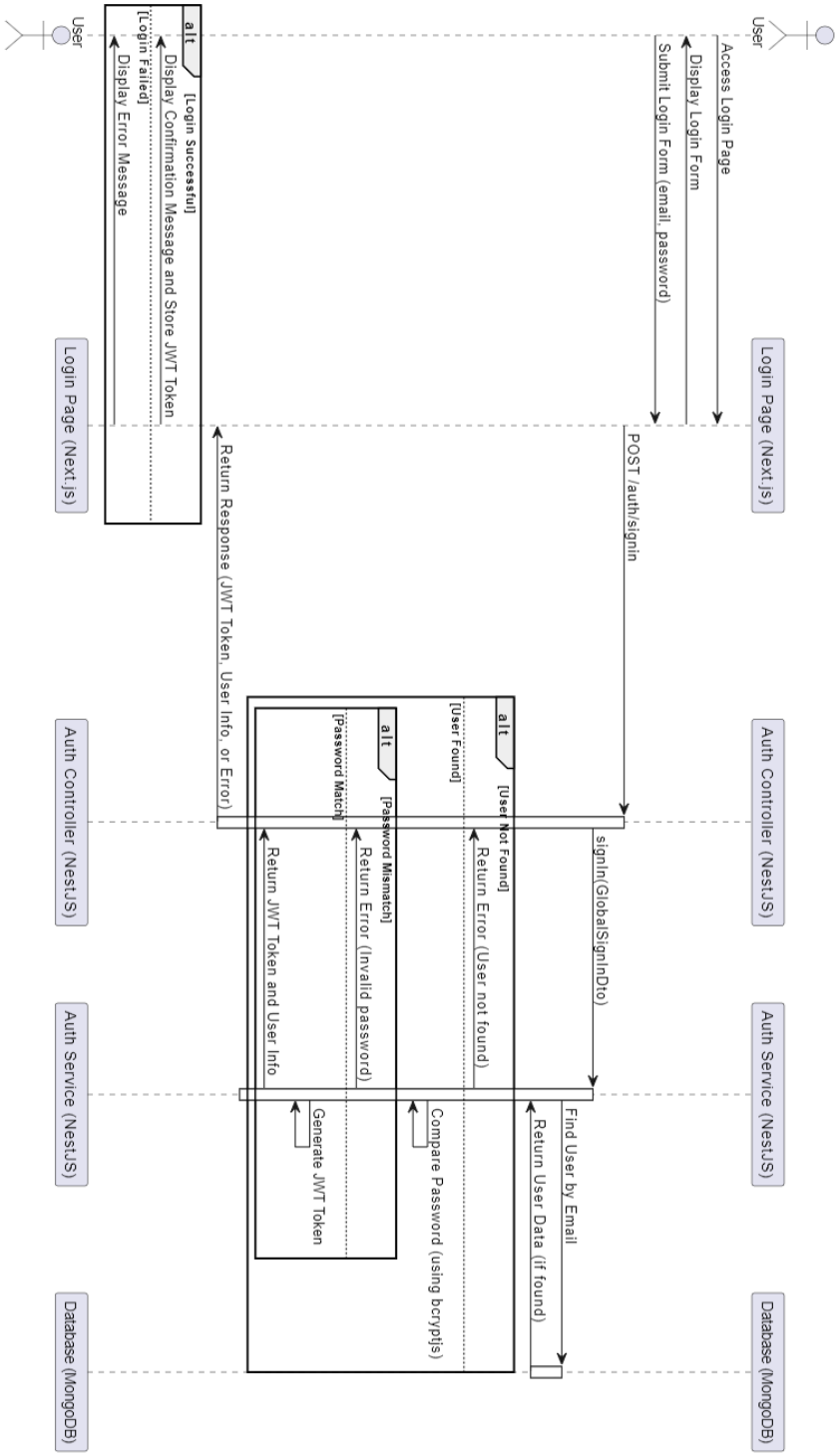

FIGURE 1.2 – Diagramme de séquence pour l'authentification. L'utilisateur soumet ses identifiants de connexion via la page de connexion. Le serveur d'API vérifie ces identifiants par rapport à la base de données. Si l'authentification est réussie, un token JWT est généré et renvoyé à l'utilisateur, lui permettant d'accéder aux fonctionnalités protégées de la plateforme.

1.2.6 Création d'un Projet

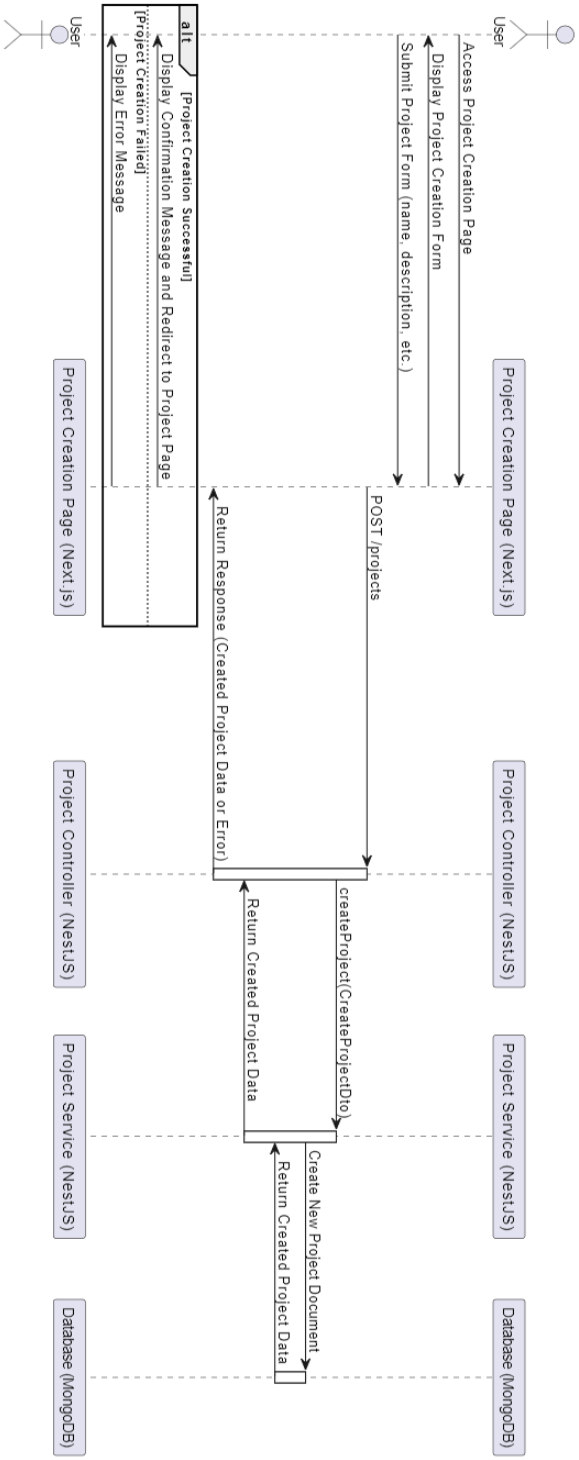

FIGURE 1.3 – Diagramme de séquence pour la création d'un projet. L'utilisateur, via la page de création de projet, soumet les détails du projet. Le serveur d'API valide ces informations, crée un nouveau document de projet dans la base de données, et renvoie les détails du projet créé à l'utilisateur.

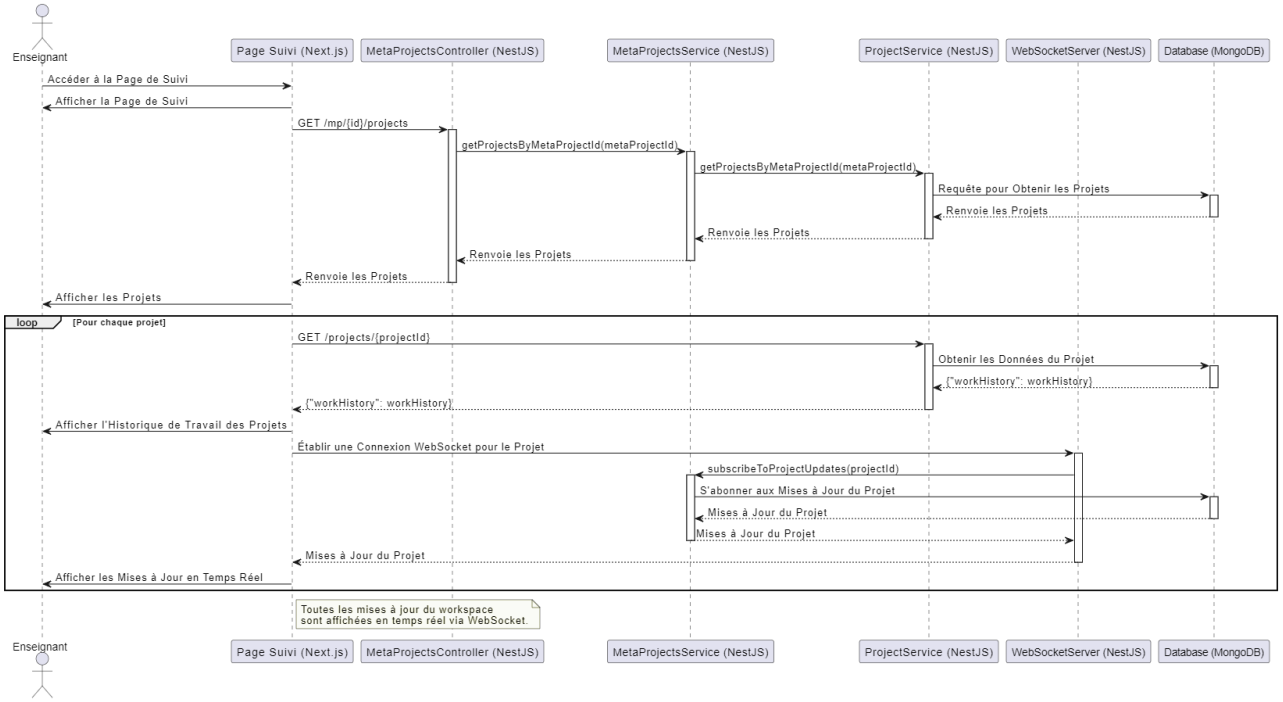

FIGURE 1.4 – Diagramme de séquence pour le suivi des mises à jour du workspace par l'enseignant. Lorsqu'un étudiant modifie son workspace, le client web envoie une notification au serveur d'API via WebSocket. Le serveur d'API diffuse ensuite cette mise à jour à tous les clients connectés qui sont autorisés à visualiser le projet, y compris l'enseignant. L'enseignant peut ainsi visualiser en temps réel les progrès de ses étudiants sur leurs projets respectifs.

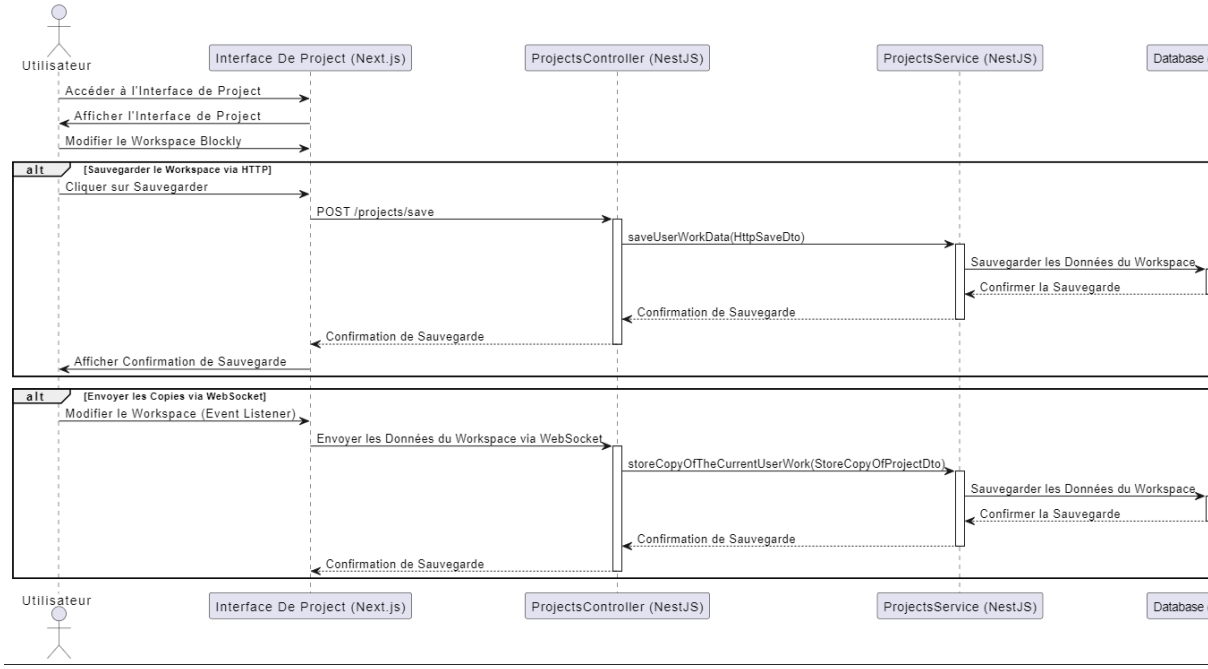

FIGURE 1.5 – Diagramme de séquence pour le fonctionnement du tableau de bord collaboratif. Ce diagramme montre comment les modifications du workspace sont envoyées au serveur via WebSocket et comment les utilisateurs peuvent sauvegarder leur travail manuellement via une requête HTTP. Le diagramme met également en évidence la sauvegarde automatique des données du workspace à intervalles réguliers via WebSocket.

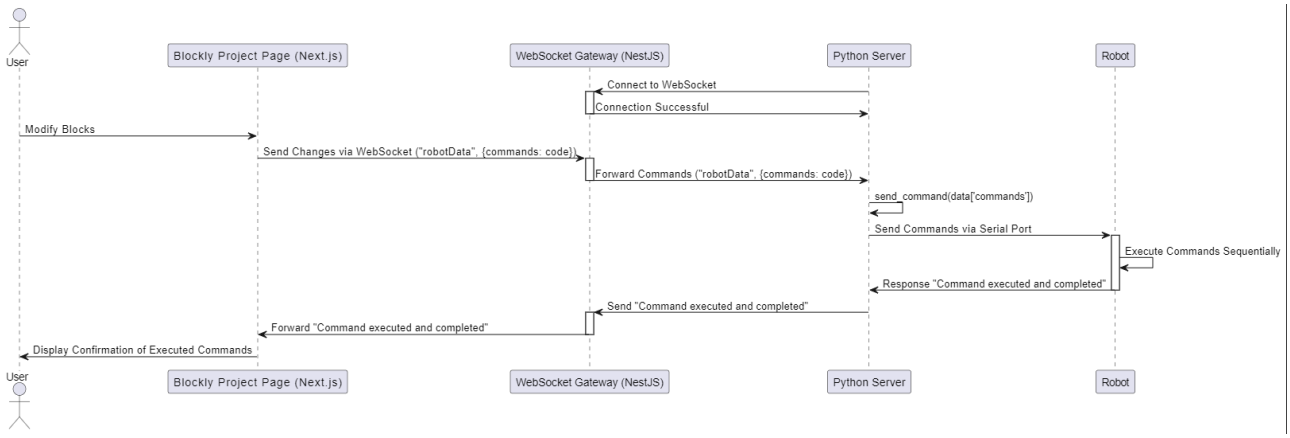

FIGURE 1.6 – Diagramme de séquence pour l'envoi de commandes au robot. Lorsqu'un utilisateur modifie les blocs de code Blockly, le client web envoie les commandes mises à jour au serveur d'API via WebSocket. Le serveur d'API transmet ces commandes à un serveur Python dédié, qui est responsable de la communication avec le robot via le port série. Le serveur Python envoie les commandes au robot et reçoit les réponses du robot, qui sont ensuite renvoyées au client web via le serveur d'API et WebSocket.