Chapter 1

Implémentation et Évaluation d'IBLOCK

1.1 Introduction

Ce chapitre plonge au cœur de l'implémentation d'IBLOCK, explorant les détails techniques qui donnent vie à cette plateforme innovante d'apprentissage de la programmation robotique collaborative. Nous examinerons en profondeur la création de blocs personnalisés, le stockage et la récupération des projets et de l'historique de travail, l'intégration avec Blockly Core, les mécanismes de collaboration en temps réel, le suivi pédagogique, et certains aspects de l'intégration du robot mBot. L'architecture du contrôle des robots sera abordée plus en détail dans le chapitre 5.

1.2 Blockly : Fondation de la Programmation Visuelle

IBLOCK repose sur Blockly, une bibliothèque JavaScript open-source développée par Google, pour offrir une expérience de programmation visuelle intuitive et engageante. Blockly permet aux utilisateurs de créer des programmes en assemblant des blocs graphiques, éliminant ainsi la nécessité d'écrire du code textuel complexe, ce qui rend la programmation accessible à un public plus large, y compris les débutants et les jeunes apprenants.

1.2.1 Blocs : Les Éléments Fondamentaux

Les blocs sont les éléments constitutifs de la programmation visuelle dans Blockly. Chaque bloc représente une instruction, une fonction ou une opération spécifique. Ils s'emboîtent comme des pièces de puzzle, guidant les utilisateurs vers des

structures de code valides et facilitant la compréhension de la logique du programme. Cette approche visuelle et tangible de la programmation favorise l'apprentissage intuitif et réduit la courbe d'apprentissage pour les novices.

Anatomie d'un Bloc

Un bloc Blockly possède une structure bien définie, composée de plusieurs éléments clés :

- **Forme :** La forme d'un bloc indique son type et sa fonction. Par exemple, les blocs de contrôle, tels que les boucles "répéter" ou les conditions "si", ont souvent une forme en "C" ou en "E" pour accueillir les blocs imbriqués. Les blocs de commande, comme "avancer" ou "tourner", peuvent avoir une forme rectangulaire.
- **Entrée :** Les blocs peuvent posséder des entrées, représentées visuellement par des encoches. Ces encoches permettent de connecter d'autres blocs, créant ainsi une séquence d'instructions. La forme de l'encoche correspond au type de données attendu, assurant une connexion logique des blocs.
- **Sortie :** Certains blocs ont des sorties, représentées par des protubérances. Ces sorties s'emboîtent dans les entrées d'autres blocs, permettant la transmission de données ou le déclenchement d'actions.
- **Champs :** Les champs sont des zones modifiables à l'intérieur d'un bloc. Ils permettent aux utilisateurs de personnaliser le comportement du bloc en spécifiant des valeurs, du texte ou en sélectionnant des options dans des menus déroulants.
- **Couleur :** Blockly utilise des couleurs distinctes pour catégoriser les blocs, facilitant ainsi l'identification visuelle des différents types de blocs et la création de programmes structurés.

1.2.2 Création de Blocs Personnalisés avec Blockly Factory

IBLOCK exploite Blockly Factory, un outil en ligne intuitif fourni par Blockly, pour créer des blocs personnalisés adaptés aux besoins spécifiques du projet. Blockly Factory permet aux administrateurs de concevoir l'apparence, le comportement et la génération de code de chaque bloc, sans nécessiter de connaissances approfondies en programmation JavaScript.

Processus de Création

La création d'un bloc personnalisé dans Blockly Factory suit un processus simple et structuré :

1. **Définition de l'Apparence :** L'administrateur commence par choisir la forme du bloc parmi une variété d'options prédéfinies, telles que des blocs rectangulaires, arrondis, en forme de puzzle, ou en "C". Il peut ensuite personnaliser la couleur du bloc et le texte affiché, assurant une représentation visuelle claire de sa fonction.

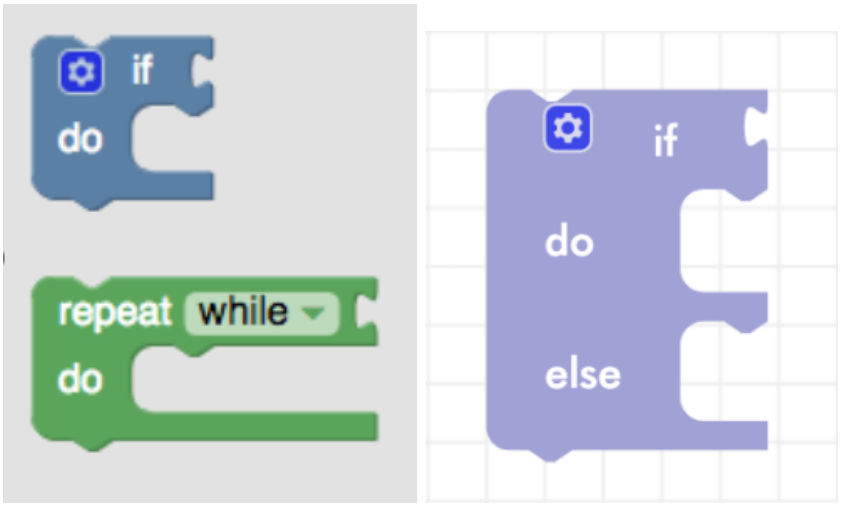

Figure 1.1: Exemples de blocs avec des formes en "C" et en "E"

2. **Ajout d'Entrées et de Sorties :** Blockly Factory permet de définir le nombre et le type d'entrées et de sorties du bloc. Les entrées sont représentées par des encoches, tandis que les sorties sont représentées par des protubérances. L'administrateur peut spécifier si une entrée accepte du texte, des nombres, des valeurs booléennes, ou d'autres types de données, assurant une connexion logique des blocs.

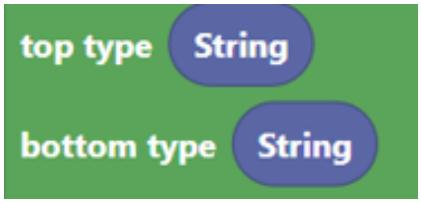

Figure 1.2: Exemple d'entrées et de sorties de type String

3. **Configuration des Champs :** Pour personnaliser davantage le comportement du bloc, l'administrateur peut ajouter des champs modifiables. Ces champs peuvent être des zones de texte pour saisir des valeurs, des

menus déroulants pour choisir parmi des options prédéfinies, ou des champs numériques pour spécifier des valeurs numériques.

4. **Spécification de la Génération de Code :** L'étape finale consiste à définir le code Python qui sera généré lorsque le bloc est utilisé dans un programme. Blockly Factory offre un éditeur de code intégré où l'administrateur peut écrire le code correspondant à la fonction du bloc, en utilisant des variables pour représenter les valeurs des champs et des entrées.

Intégration de Blockly Factory dans IBLOCK

IBLOCK intègre Blockly Factory dans son interface d'administration, permettant aux administrateurs de créer et de gérer les blocs personnalisés utilisés dans la plateforme. Ces blocs sont ensuite stockés dans la base de données MongoDB pour être utilisés dans l'éditeur visuel.

Interface de Blockly Factory

Blockly Factory nous fournit une interface intuitive pour créer nos propres blocs. C'est un peu comme un atelier où l'on peut assembler les différentes pièces d'un bloc et visualiser le résultat en temps réel.

La Figure 1.3 montre à quoi ressemble cette interface. On y retrouve plusieurs zones distinctes, chacune ayant un rôle précis dans la création du bloc.

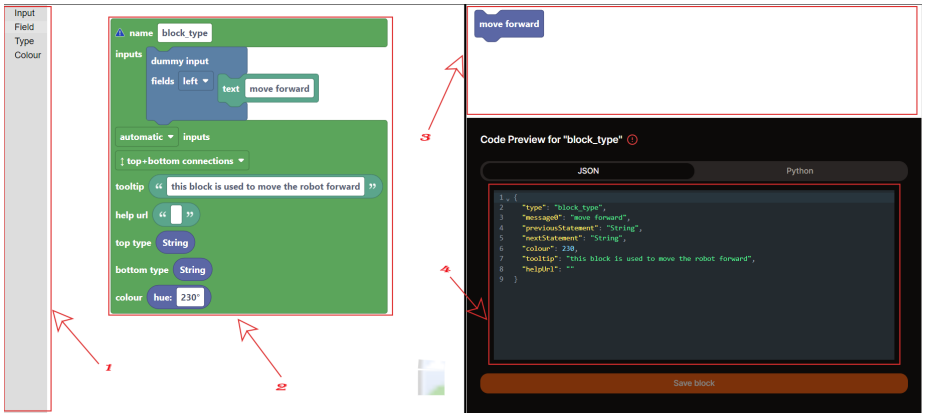

Figure 1.3: Interface de Blockly Factory pour la création de blocs personnalisés

Zone 1 (Input, Field, Type, Colour): Cette zone agit comme une boîte à outils (toolbox) où les utilisateurs peuvent sélectionner et configurer les caractéristiques principales du bloc. Voici les éléments disponibles dans cette zone :

Input (Entrée) : Cette section permet de définir les entrées du bloc, représentées par des encoches. Les utilisateurs peuvent choisir parmi différents types d'entrées, telles que des valeurs textuelles, des instructions, des entrées factices ou des entrées de fin de ligne. Chaque type d'entrée est représenté par un bloc spécifique dans la boîte à outils.

Field (Champ) : Les champs sont des zones modifiables à l'intérieur du bloc. Cette section permet aux utilisateurs de personnaliser le comportement du bloc en ajoutant des champs de texte, des menus déroulants, des champs numériques, des angles, des cases à cocher, etc. Chaque type de champ est représenté par un bloc spécifique dans la boîte à outils.

Type (Type) : Cette section permet de définir le type de données que le bloc peut accepter ou produire. Les utilisateurs peuvent choisir parmi différents types prédéfinis tels que `any`, `Boolean`, `Number`, `String`, `Array`, et `other`. Chaque type est représenté par un bloc spécifique dans la boîte à outils.

Colour (Couleur) : Cette section permet de définir la couleur du bloc, facilitant ainsi la catégorisation visuelle des blocs. Les utilisateurs peuvent choisir parmi une palette de couleurs pour personnaliser l'apparence du bloc.

Zone 2 (Connexions): Cette zone permet à l'administrateur de configurer l'apparence globale du bloc en utilisant les éléments sélectionnés dans la Zone 1. L'administrateur commence par choisir la forme du bloc parmi une variété d'options prédéfinies, telles que des blocs rectangulaires, arrondis, en forme de puzzle, ou en "C". Il peut ensuite personnaliser la couleur du bloc, le texte affiché, les entrées, les connexions, et autres paramètres visuels, assurant une représentation claire de sa fonction.

Zone 3 (Définition de l'Apparence): Cette zone permet de visualiser le bloc tel qu'il apparaîtra dans l'éditeur.

Zone 4 (Code): Blockly Factory génère automatiquement le code correspondant à notre bloc en JSON pour définir sa structure. L'administrateur peut ensuite écrire le code Python correspondant à la fonction du bloc pour son exécution.

Définition JSON d'un Bloc dans Blockly Factory

Une fois que l'on a terminé de configurer notre bloc dans Blockly Factory, un code JSON est généré automatiquement. Ce code JSON correspond à la définition du bloc, que l'on a expliquée précédemment.

La Figure 1.4 montre un exemple de code JSON généré pour un bloc simple qui fait avancer un robot.

Le code JSON généré est le suivant :

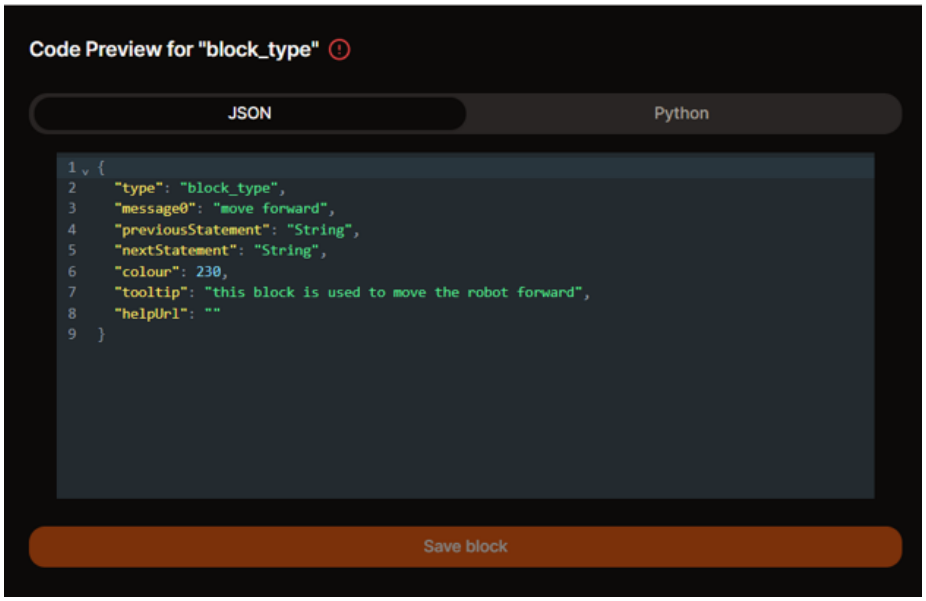

Figure 1.4: Aperçu du code JSON généré pour un bloc personnalisé dans Blockly Factory

Génération du Code Python

Dans Blockly Factory, l'administrateur a la possibilité de définir manuellement le code Python correspondant à la fonctionnalité du bloc. Une fois la définition JSON du bloc terminée, l'administrateur peut passer à l'onglet "Python" dans l'interface de Blockly Factory pour écrire le code Python.

La Figure 1.6 montre un aperçu de l'interface où l'administrateur écrit le code Python pour le bloc "avancer".

Le code Python écrit par l'administrateur pour le bloc "avancer" est le suivant :

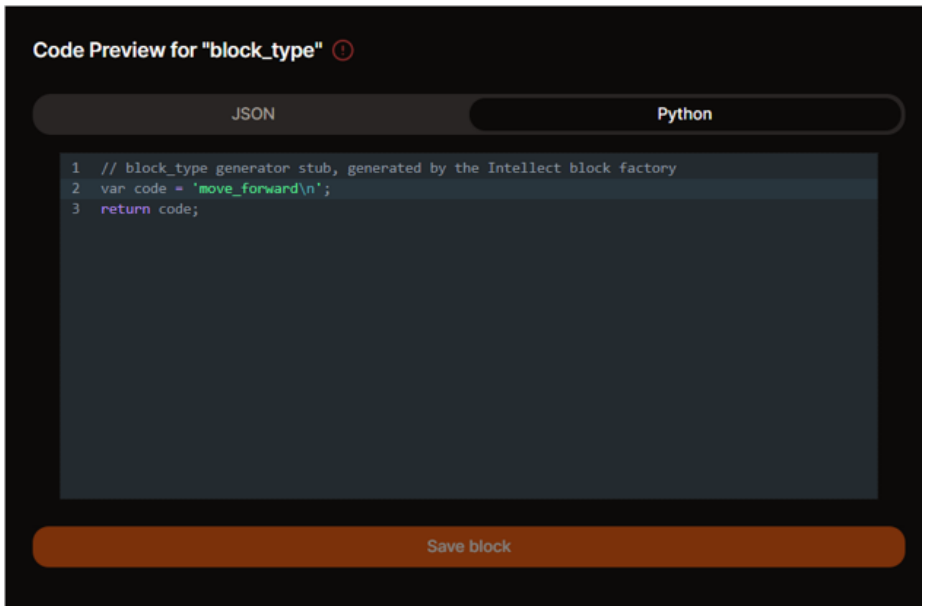

Figure 1.5: Aperçu de l'interface de Blockly Factory pour écrire le code Python du bloc "move forward"

1.3 Structure de la Base de Données et Identification du Propriétaire des Blocs

Après avoir exploré la création de blocs personnalisés avec Blockly Factory, il est crucial de comprendre comment ces blocs, ainsi que d'autres données essentielles, sont organisés et gérés dans la base de données d'IBLOCK. Cette section détaille la structure de la base de données, explique pourquoi MongoDB a été choisi comme système de gestion de base de données, et montre comment identifier le propriétaire d'un bloc.

1.3.1 Choix de MongoDB

Pour la gestion de sa base de données, IBLOCK utilise MongoDB, un système de gestion de base de données NoSQL orienté documents. Ce choix est justifié par plusieurs avantages offerts par MongoDB :

- Flexibilité des schémas de données.
- Scalabilité horizontale.
- Performances élevées pour les opérations de lecture et d'écriture.
- Facilité de requêtage et d'indexation des données.

1.3.2 Structure de la Base de Données

La base de données d'IBLOCK, construite avec MongoDB, est organisée en collections principales :

- **Utilisateurs :** Informations sur les utilisateurs.
- **Projets :** Détails des projets créés par les utilisateurs.
- **Blocs :** Définitions des blocs personnalisés.
- **Historique des Modifications :** Enregistrements de l'historique des modifications apportées aux projets.

1.3.3 Sauvegarde et Récupération

La pérennité des données étant une priorité pour IBLOCK, des stratégies de sauvegarde et de récupération robustes sont mises en place :

Sauvegardes Régulières

Des sauvegardes complètes de la base de données sont effectuées régulièrement pour assurer la sécurité des données. Ces sauvegardes sont stockées dans un emplacement sécurisé, distinct de l'infrastructure principale.

Récupération des Données

En cas de défaillance du système, un processus de récupération est mis en œuvre pour restaurer les données à partir des sauvegardes les plus récentes. Ce processus est régulièrement testé pour garantir son efficacité.

1.3.4 Identification du Propriétaire d'un Bloc

Le propriétaire d'un bloc personnalisé est identifié grâce à la table "Blocs", où chaque enregistrement contient un champ "user_id" faisant référence à l'utilisateur ayant créé le bloc.

Exemple : Si un utilisateur avec l'identifiant "123" crée un bloc de type "move_forward", l'enregistrement correspondant dans la base de données permettra d'identifier facilement cet utilisateur comme le propriétaire du bloc.

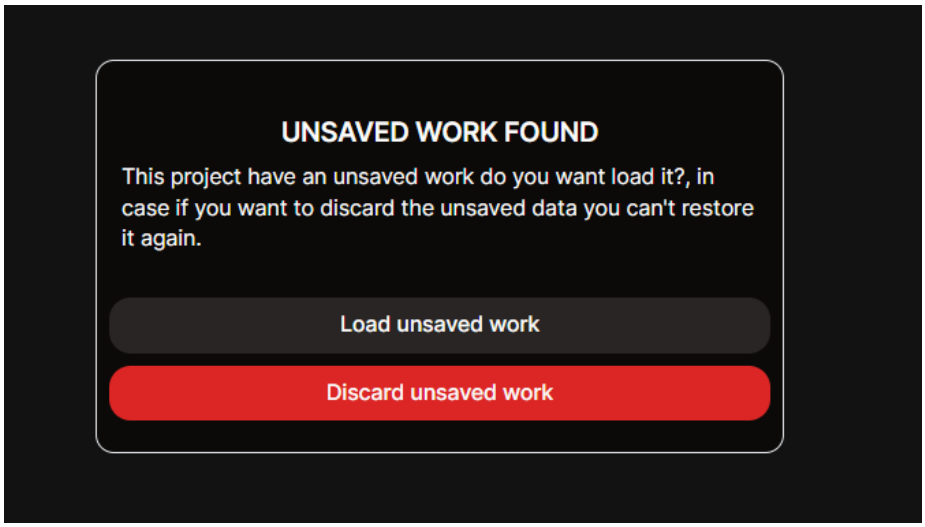

Figure 1.6: Récupération des Données

1.4 Collaboration en Temps Réel et Suivi Pédagogique

La fonctionnalité de collaboration en temps réel constitue le cœur de l'approche pédagogique d'IBLOCK, permettant aux étudiants de collaborer sur des projets de programmation de manière intuitive et engageante. Cette fonctionnalité transforme l'apprentissage de la programmation en une expérience collective et interactive, encourageant le partage d'idées et la résolution de problèmes en équipe.

1.4.1 Socket.IO : Facilitation de la Collaboration en Temps Réel

Pour permettre cette collaboration en temps réel, IBLOCK utilise Socket.IO, une bibliothèque JavaScript essentielle pour les applications web qui nécessitent une communication bidirectionnelle en temps réel entre les clients web et les serveurs. Socket.IO joue un rôle crucial dans la synchronisation des actions des utilisateurs, permettant aux utilisateurs de collaborer de manière intuitive et engageante.

Comment Fonctionne Socket.IO ?

Socket.IO utilise la technologie WebSocket pour établir une connexion persistante entre le client et le serveur. Cette connexion full-duplex permet une communication bidirectionnelle fluide, rendant les interactions en temps réel plus efficaces et réactives.

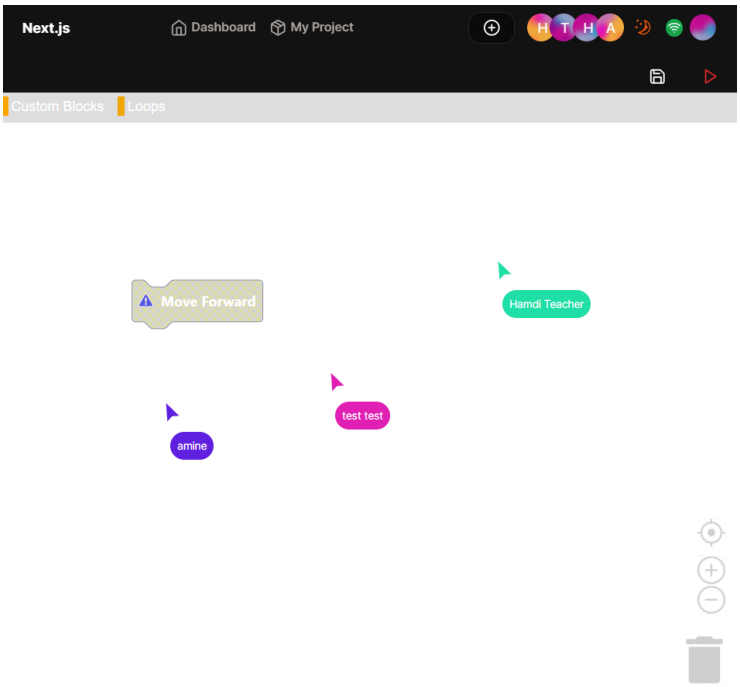

Figure 1.7: Illustration du mécanisme de collaboration en temps réel dans IBLOCK.

Les fonctionnalités clés facilitant la collaboration en temps réel incluent :

- **Communication Bidirectionnelle :** Permet l'envoi de données dans les deux sens entre client et serveur.
- **Connexion Persistante :** Maintient la connexion ouverte pour l'envoi de données à tout moment.
- **Compatibilité :** Assure une compatibilité étendue entre différents navigateurs et environnements.
- **Simplicité d'Utilisation :** Offre une API simple pour une intégration et utilisation aisées.
- **Indicateurs de Présence :** Chaque utilisateur actif est représenté par un indicateur unique, permettant de visualiser qui participe à la session en temps réel.
- **Synchronisation des Actions :** Les modifications apportées par un utilisateur sont synchronisées en temps réel avec tous les participants, assurant une cohérence de l'état du projet.

Utilisation de Socket.IO pour la Collaboration sur des Projets

Dans IBLOCK, Socket.IO synchronise les actions des utilisateurs en temps réel, permettant une collaboration simultanée sur les projets de programmation. Les modifications apportées par un utilisateur sont immédiatement diffusées à tous les autres collaborateurs, assurant une cohérence et une expérience utilisateur harmonieuse.

1.4.2 Suivi des Mises à Jour du Workspace en Temps Réel

Pour les enseignants, IBLOCK propose un tableau de bord dynamique utilisant la technologie WebSocket pour une communication en temps réel. Ce tableau de bord permet aux enseignants de suivre l'évolution des projets de leurs étudiants de manière efficace et interactive.

Les enseignants peuvent voir en temps réel tout le travail effectué par les étudiants, ce qui facilite le suivi et l'évaluation des progrès de chaque élève. Le tableau de bord affiche les activités des étudiants sous forme de blocs Blockly, permettant aux enseignants d'observer directement les modifications apportées par chaque étudiant.

Les fonctionnalités clés du tableau de bord pour les enseignants incluent :

- **Vue d'Ensemble en Temps Réel :** Les enseignants peuvent voir en temps réel les actions des étudiants sur leurs projets, y compris l'ajout, la modification et la suppression de blocs. Cela permet une surveillance continue sans avoir besoin d'interventions manuelles.
- **Historique des Modifications :** Le tableau de bord enregistre et affiche un historique détaillé des modifications apportées aux projets. Chaque bloc est affiché avec un indicateur montrant l'étudiant qui a effectué la modification. Les enseignants peuvent ainsi retracer les étapes de développement de chaque projet et identifier les contributions individuelles des étudiants.
- **Indicateurs de Présence :** Chaque étudiant actif est représenté par un indicateur unique, permettant aux enseignants de voir qui travaille sur quel projet en temps réel. Cela facilite la gestion de la classe et l'intervention rapide en cas de besoin.
- **Intervention Directe :** Les enseignants peuvent entrer dans les projets des étudiants pour fournir une aide directe en temps réel. Ils peuvent collaborer avec les étudiants, apporter des corrections et des suggestions immédiatement, ce qui améliore l'apprentissage et la compréhension des concepts de programmation.

Ces fonctionnalités permettent aux enseignants de fournir un suivi pédagogique personnalisé et de haute qualité. En observant les progrès en temps réel, ils peuvent intervenir immédiatement pour offrir des conseils et des corrections, améliorer l'apprentissage des étudiants et assurer une meilleure compréhension des concepts de programmation.

1.5 Conclusion

En conclusion, l'implémentation d'IBLOCK représente une avancée significative dans l'éducation à la programmation, en particulier dans le domaine de la programmation robotique collaborative. Grâce à son intégration étroite avec Blockly, IBLOCK rend la programmation accessible et engageante pour un large éventail d'apprenants, y compris les débutants et les jeunes étudiants. Les blocs visuels de Blockly, associés à la capacité d'IBLOCK de créer des blocs personnalisés, offrent une plateforme riche pour explorer des concepts de programmation complexes de manière intuitive.

La collaboration en temps réel, facilitée par Socket.IO, transforme l'apprentissage de la programmation en une expérience collective, où les étudiants peuvent partager des idées, résoudre des problèmes ensemble et apprendre les uns des autres. Cette approche ne favorise pas seulement l'acquisition de compétences en programmation, mais développe également des compétences essentielles en communication et en travail d'équipe.

La structure de la base de données d'IBLOCK, basée sur MongoDB, assure une gestion efficace et flexible des données, permettant une sauvegarde et une récupération robustes des projets et de l'historique de travail. Cette backbone de données soutient la nature dynamique de la collaboration en temps réel et la création de contenu personnalisé.

En somme, IBLOCK se démarque comme une solution complète et sécurisée pour l'apprentissage de la programmation collaborative. Ses fonctionnalités avancées de création de blocs personnalisés, de collaboration en temps réel et de suivi pédagogique positionnent IBLOCK comme une plateforme éducative de premier plan, capable de s'adapter aux besoins évolutifs des apprenants et des éducateurs dans le domaine passionnant de la programmation robotique.

L'avenir d'IBLOCK promet encore plus d'innovations et d'améliorations, avec pour objectif de continuer à enrichir l'expérience d'apprentissage et de rendre la programmation encore plus accessible et captivante pour tous.