**UNIVERSITÉ DU QUÉBEC À MONTRÉAL**

**LIVRABLE SPRINT 3**

**PERSONAL ASSISTANT**

**TRAVAIL**

**PRÉSENTÉ**

**À**

**MILI HAFEDH**

**DANS LE CADRE DU COURS**

**PROJET D’ANALYSE ET DE MODÉLISATION**

**INM5151**

**PAR**

**de VIPART Lukas**

**PINEAULT Yannick**

**THIEMONGE Christophe**

**26 DECEMBRE 2019**

[**1. INTRODUCTION**](#_r9g6q69d332a) **2**

[1.1 Objectifs](#_sfhnnxbzvk7v) 3

[1.2 Vue d’ensemble du produit](#_ysw17qe4wixt) 3

[1.3 Définitions, acronymes et abréviations](#_ft7lpkyfvvux) 3

[1.5 Aperçu du document](#_5un10ld40ui1) 3

[**2. DESCRIPTION GÉNÉRALE DU LOGICIEL**](#_ys224y1isvtq) **3**

[2.1 Vue d'ensemble des fonctions du produit](#_rpo0vl7pqcrd) 4

[2.2 Contenu des sprints](#_2bq8x2qdie1t) 5

[**3. CONTENU DU SPRINT ACTUEL**](#_czfegikgpgzf) **6**

[3.1 Fonctionnalités implantés dans le sprint](#_w3x5ntdm14r3) 6

[3.2 Modèle de classe](#_w2tjoe92ws6t) 7

[Diagramme sprint 1](#_utwz1m59ib2s) 7

[Diagramme sprint 2](#_c0td2j2da6ps) 8

[Diagramme sprint 3](#_jsr4l770vjz) 9

[**REVUES**](#_yj29soiv2qsi) **10**

[Revue de Sprint](#_q7zc0ejjxme2) 10

[Revue technique](#_izhiry50ezql) 10

[Revue gestion](#_kxwaamw8iqg6) 11

[Revue du projet](#_7innhqctqr6p) 12

[Revue technique](#_4tdqstq3xhdn) 12

[Revue gestion](#_ecdl6ll1jd1n) 13

[Avis sur le développement agile](#_koj14rgu5f8u) 15

[**ANNEXES**](#_lhr7pljvkl7a) **15**

## **1. INTRODUCTION**

### **1.1 Objectifs**

Ce document vise à expliciter ce qui a été implémenté durant le second sprint de développement de Personal Assistant.

### **1.2 Vue d’ensemble du produit**

Discord est une application permettant aux gamers de discuter tout en jouant. Un chat vocal est proposé, mais aussi textuel avec de nombreuses options d’administration. Les utilisateurs doivent en premier joindre un serveur. Des rôles leur sont attribués permettant de discuter, partager des images, des vidéos, des liens, créer des salons privés, etc…

Afin de pallier à la difficulté d’administration d’un grand nombre d’utilisateurs (notamment sur des discords publiques) mais aussi pour ajouter des fonctionnalitées qui ne sont pas présentes à l’origine (mini jeu sur chat texte ou musique en chat vocal), des bots ont été introduits. Ils sont développés par la communauté et offrent une panoplie de fonctionnalités extrêmement large, allant de la simple attribution de rôles automatiquement à des utilisateurs, à la gestion poussée des messages et lancement de serveurs de jeux.

Personal Assistant sera un bot discord. Nous souhaitons par son développement améliorer la fluidité d’utilisation des serveurs sur lesquels il se trouve (administration) mais aussi permettre plus d’interaction avec les utilisateurs (mini-jeux, discussion, musique…). Il ne sera en outre pas utilisable en dehors de l’application discord.

### **1.3 Définitions, acronymes et abréviations**

Nous nommerons le bot PA pour Personal Assistant.

### **1.5 Aperçu du document**

La section 2 contient une description générale du logiciel dans laquelle seront expliqués les objectifs que le projet veut combler tout en organisant de manière générale les sous-objectifs pour chacun des sprints.

Les cas d’utilisation sont présentés dans la section 3.L’évolution du diagramme de classe est représentée par les anciens diagrammes ainsi qu’un nouveau représentant l’état actuel du bot.

La section 4 présente une revue du troisième sprint maintenant complété mais aussi du projet dans son intégralité en passant par nos bons coups et nos faiblesses, puis par des solutions à certains problèmes, suivie par notre plan pour la suite du projet et notre avis sur le développement agile.

## **2. DESCRIPTION GÉNÉRALE DU LOGICIEL**

### **2.1 Vue d'ensemble des fonctions du produit**

Ici sont énumérés les fonctions que notre produit final sera capable d’accomplir.

On divise ses fonctions en deux sections: divertissement qui permet de fournir une expérience discord plus amusante et administration permettant la gestion des utilisateurs.

Les commandes de divertissement :

* + - Lecture de musique par youtube : Les fonctions permettant de demander au bot de lire de la musique youtube dans le canal vocal.
    - Auto apprentissage de réponses automatiques à certains messages (ou de réactions) : Permet au bot d’apprendre une liste de réponses préfaites face à divers messages.
    - Lancement de mini jeux textuels (Pierre-papier-ciseaux/ lancer de pièces) : permet de lancer des mini jeux tel que Pierre-papier-ciseaux ou le lancer de pièces.
    - Recherche d’images google images/reddit : permet de rechercher et d’afficher dans le canal textuel des images tirées de reddit ou de google image.
    - Recherche de définitions: permet de rechercher une définition et de l’afficher dans le canal textuel.
    - Lancement et gestion d'applications sur la machine hôte : permet de lancer et gérer diverses applications sur la machine hôte comme par exemple un serveur minecraft pour jouer entre amis.
    - Demander la météo pour une zone X : permet d’afficher la météo pour une ville demandée dans le canal textuel.
    - Demande l'heure pour une timezone X : permet d’afficher l’heure pour une ville demandée dans le canal textuel.
    - Ramener le résultat d’une recherche youtube : affiche dans le canal textuel le premier lien résultant d’une recherche youtube.
    - Ramener les statistiques du joueur pour un jeu : affiche dans le canal textuel les statistiques d’un joueur pour une liste de jeux prédéfinie.
    - Ramener des infos sur un film : Affiche dans le canal textuel des informations sur le film demandé.

.

Les fonctions d’administration:

* + - Autosuppression des commandes déjà réalisées : supprime une commande qui a déjà été exécutée par le bot.
    - Arrêt du bot : éteint le bot.
    - Ramener les logs d’un utilisateur : ramène les logs du serveur triés pour uniquement refléter un utilisateur.
    - Bannissement/kick d’un utilisateur : expulse un utilisateur indésirable du serveur.
    - Auto-attribution de rôles par réaction : Attribue automatiquement un rôle à un utilisateur en fonction de sa réaction à un message du bot.
    - Attribution de rôle par message : Attribue un rôle en utilisant un message dans le chat.
    - Attribution automatique d’un rôle de base : Attribue un rôle de base à tous les utilisateurs rejoignant le serveur.
    - Suppression automatique de certains messages par mots clés : Supprime automatiquement les messages contenant certains mots pris dans une liste.
    - Création de channels textuels/vocaux : Permet de créer un nouveau canal de discussion.
    - Envoie de messages privés aux utilisateurs du discord : Envoie un message privé de la part du bot à un utilisateur (message créé par un modérateur par exemple pour une annonce ou pour un avertissement).

### **2.2 Contenu des sprints**

Premier Sprint:

* Création du serveur discord et ajout du bot
* Implementation partielles des fonctions essentielles:
  + attribution de rôle
  + lecture de musique
  + kick/ban
  + arrêt du bot

Deuxième Sprint:

* Complétion des fonctions essentielles (permission de ban/kick uniquement accordé aux modérateurs et grades supérieurs, commande d'arrêt donne du feedback)
* Ajout de fonctions souhaitables :
  + suppression de messages par mots-clés
  + suppression des commandes déjà réalisées
  + envoie de messages privés aux utilisateurs
* Nettoyage et modularisation du code

Troisième Sprint:

* Complétion de ce qui ne l’avait pas été dans les sprints précédents (recherche par mot clé YouTube, pause de lecture)
* Ajouts de fonctions optionnelles:
  + Recherche d’images
  + Recherche de définitions
  + Lancer un mini-jeu textuel
* Nettoyage final du code

## **3. CONTENU DU SPRINT ACTUEL**

### **3.1 Fonctionnalités implantés dans le sprint**

**Améliorations de ce qui était livré avant**

Ajout de feedback à la console quand le bot est arrêté par un utilisateur.

**Use case implantés**

1. Recherche YouTube par mots clés

Le but était de faciliter la recherche d’un lien, il suffit à présent de demander au bot et il nous envoie le premier lien trouvé pour une recherche X.

1. Recherche d’images

Cette commande permet de facilement rechercher une image sur un thème de notre choix, le bot affiche l’image dans le chat textuel.

1. Recherche de définitions

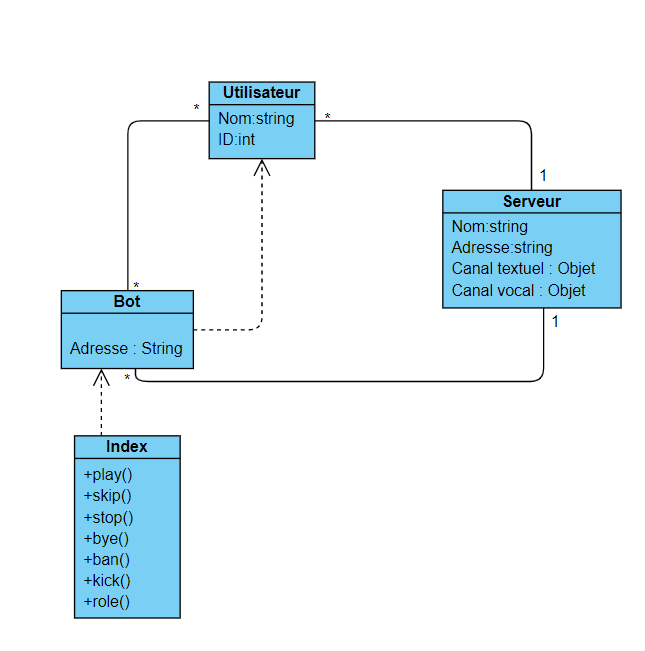
Cette commande permet de retrouver une définition à partir d’Urban Dictionary pour le sujet que l’utilisateur voudra.

1. Mini-jeux textuels

Ce use case implémente 2 commandes, la première permet de jouer un lancer de pièce, avec un paris sur le côté gagnant (le bot étant notre adversaire), la seconde permet de jouer à pierre papier ciseau contre le bot.

### **3.2 Modèle d**e **classe**

#### **Diagramme sprint 1**



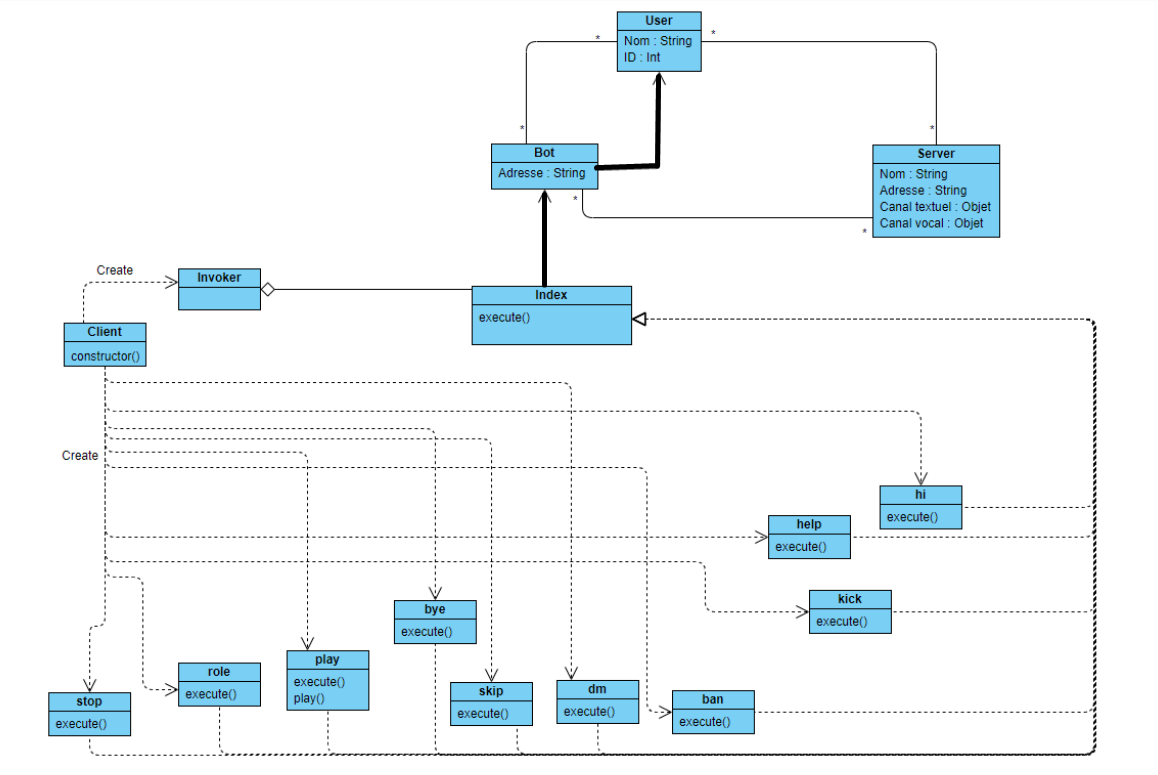
#### 

Nous avions un diagramme assez simple pour commencer, le bot n’était qu’un “faux” utilisateur avec un programme derrière, l’api discord se chargeant de le connecter grâce à un token.

#### 

#### 

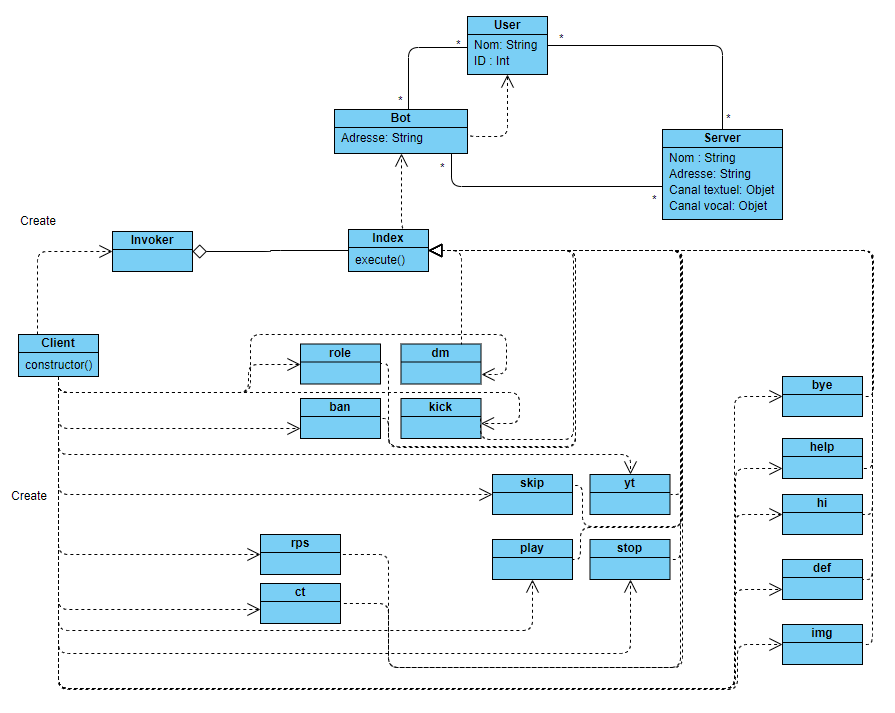
#### **Diagramme sprint 2**



Durant le sprint 2 nous avons modularisé notre code en suivant le patron de conception commande, cela permettait une plus grande flexibilité dans l’ajout et la suppression de fonctionnalités, la documentation et de manière générale nous évitait de mélanger des fonctions qui n’ont pas à l’être.

## 

#### **Diagramme sprint 3**



Ce dernier diagramme représente simplement l’ajout des commandes du sprint trois, et regroupe un peu plus les commandes en fonction de leur “type” (commandes multimédias ensemble, commandes de jeux ensemble etc).

Une commande tapée par un utilisateur est envoyée à l’index qui instancie un objet discord à travers la classe client, cet objet va permettre de réaliser les différentes commandes implémentées concrètement et pour finir le résultat de l’exécution sera renvoyé à travers l’objet “bot” sur le serveur. L’objet index se charge en outre de faire le lien entre le serveur discord (pour connecter le bot et envoyer les messages), l’objet discord local (qui permet d’utiliser les fonctions de l’api) et les différentes commandes.

## REVUES

### Revue de Sprint

#### Revue technique

**Complété**

1. Faire une recherche de définition sur le site web “urban dictionary”
2. Faire une recherche d’images
3. Faire une recherche YouTube par mots clé
4. Jeu textuel de roche-papier-ciseaux
5. Jeu textuel de lancer de pièce

**Non Complété**

1. Ajouter la pause lors de la lecture de vidéo YouTube
2. Attribution de rôle par un modérateur

**·Bien fonctionné**

1. Le jeux textuel de roche-papier-ciseaux fut assez facile à implémenter tout comme le jeu de lancer de pièce qui utilise une logique similaire.
2. La recherche de définitions avec “urban dictionary”.

**Pas bien fonctionné**

1. Le formatage du texte de manière générale, que ce soit pour les définitions ou les jeux, est plus compliqué qu’il n’y paraît, il est pour le moment assez sommaire.

**· Obstacles et solutions**

1. La recherche de définition fut une combinaison de 2 efforts. D’un côté la partie esthétique de la recherche était bonne, mais la recherche ne retournait pas le bon résultat. D’un autre la recherche retournait ce que l’on voulait, mais avec du texte sans images ou couleures.
2. Apprendre à utiliser l’ API de “urban dictionary” pour savoir quels champs retourner dans notre recherche (Ex: définition, auteur, titre)
3. La recherche d’images à posé quelques soucis car il y a beaucoup de méthodes proposées et certaines ne sont plus fonctionnelles, au début rien n’était retourné alors qu’une image était bel et bien chargée, après quelques recherches une solution a été trouvée en relisant bien la documentation.
4. La recherche par mots clés YouTube a été le plus compliqué car il faut combiner l’utilisation de l’API de google (qui utilise un token comme celui de discord et demande d’aller faire des réglages pour les autorisations) avec celle d’un autre api qui vient s’occuper de traiter la partie purement YouTube, tous les api traitant Youtube n’étaient pas fonctionnels, il a fallu en trouver un qui fonctionne bien et cela a solutionné la plupart des difficultés rencontrées.

#### Revue gestion

**Personnes**

* Jusqu’au sprint 2 la plupart des tâches ont été complétées par la même personne. Durant le sprint 3 la participation des autres paires fut plus importantes.

**Relations**

* Notre projet ne demande pas beaucoup d'interactions entre les membres de l’équipe. Aucun des comportements de notre bot dépend d’un autre. Nous interagissons au début et à la fin du sprint pour la répartition des tâches. Il y a eu quelques collaborations lorsque quelqu’un bloquait sur quelque chose par exemple la recherche de définition.

**Processus**

* Nous avons commencé par les tâches non complétés. Si trop de temps était passé sur ces tâches sans progression nous passions aux tâches du sprint courant. Nous testions en même temps que nous implémentions. Il y a eu peu de rétroaction sur le projet dû au manque de temps.

**Outils**

* Il fut assez facile d’utiliser l’API de “urban dictionary”. Plusieurs personnes l’utilisent déjà, donc on trouve rapidement une solution à son problème. Celle-ci se combine bien avec les fonctions de la librairie Discord.js.
* Node.js permet l’implémentation de plusieurs librairie variées. Ces librairies ont facilités nos tâches, mais cependant elles augment les dépendances du projet par rapport à celles-ci.
* Google doc a une fois de plus été l’outil privilégié pour les rapports
* GitHub permet un suivi facil des différentes versions du projet et de le récupérer facilement d’une machine à une autre, en addition pastebin a été utilisé pour pouvoir transférer juste des bouts de code

**Améliorations potentielles**

*Jeu textuel*

* Pour le jeu de roche papier ciseaux et de lancer de pièce il aurait été bien qu’ils soient plus interactifs.

*Recherche YouTube par mots clés*

* Une intégration avec la fonction de lecture de musique pour lire directement par mot clé plutôt que par url pourrait être une bonne piste d’amélioration.

· **Plan**

*Jeu textuel*

1. Avoir des réactions (sortes d'icônes sous les messages discord) que l’on peut cliquer (Ex: icône ciseaux, roche, papier) pour le jeu de pierre-papier-ciseaux. Permettre à deux utilisateurs de jouer l’un contre l’autre.
2. Ajouter une animation de pièce et afficher la face gagnant pour le lancer de pièce.

*Recherche YouTube par mots clés*

* Ajouter un appel vers la recherche par mot clé lors de l’appel d’une vidéo YouTube pour la lecture.

### Revue du projet

#### Revue technique

**Complété**

* Lecture de musique par YouTube
* Lancement de mini-jeux textuels
* Recherche d’images
* Recherche de définitions
* Ramener le résultat d’une recherche YouTube
* Auto-suppression des commandes déjà réalisées
* Arrêt du bot
* Attribution de rôles par message
* Auto-suppression de messages par mots-clés
* Envoie de messages privés aux utilisateurs du discord
* Commande pour dire bonjour
* Bannissement/expulsion d’un utilisateur

**Non Complété**

* Auto-apprentissage de réponses automatiques
* Attribution de rôle par réaction
* Pause de lecture YouTube
* Recherche de météo
* Recherche d’heure pour une timezone
* Gestion d’application sur la machine hôte
* Ramener les statistiques d’un joueur
* Rechercher d’infos sur un film
* Création de channels
* Attribution automatique d’un rôle de base
* Ramener les logs d’un utilisateur

**·Bien fonctionné**

* Globalement tout ce qui a été implanté à bien fonctionné

**Pas bien fonctionné**

* La pause de vidéo YouTube est bien plus compliquée que prévu et ne fonctionne pas du tout
* Le formatage du texte réponse du bot pour certaines commandes (définitions, messages privés, et jeux surtout) demande un certain sens artistique

**Obstacles et solutions**

Les obstacles rencontrés ont surtout été au niveau de la méconnaissance du langage et des outils associés, la solution a été de se documenter encore et encore pour progresser.

#### Revue gestion

**Personnes**

* Jusqu’au sprint 2 la plupart des tâches ont été complétées par la même personne. Durant le sprint 3 la participation des autres paires fut plus importantes.

**Relations**

* Notre projet ne demande pas beaucoup d'interactions entre les membres de l’équipe. Aucun des comportements de notre bot ne dépend d’un autre. Nous interagissons au début et à la fin des sprints pour la répartition des tâches. Nos relations se sont améliorées à chaque sprint, dû à une meilleure communication et aux simples vertus du temps. Il y a eu de l’entraide au sprint 2 et 3 pour résoudre des problèmes de développement dû à une meilleure participation des membres de l’équipe ce qui était inexistant au sprint 1.

**Processus**

* Nous avons commencé par les tâches non complétés. Si trop de temps était passé sur ces tâches sans progression nous passions aux tâches du sprint courant. Nous testions en même temps que nous implémentions. Nous avons fait plus de rétroaction sur notre projet aux sprint 1 et 2 qu’au sprint 3.

**Outils**

* Node.js nous a permis d’intégrer tous les api dont nous avions besoin, malheureusement ces derniers peuvent vite devenir obsolète car node évolue vite
* Google drive a été l’outil privilégié pour les rapports, les backlogs et tous les documents à partager de manière général
* GitHub accompagné de pastebin ont permis de facilement maintenir le code depuis le début du projet
* Des machines virtuelles sous Ubuntu/Mint ont permis de faire tourner le bot lors des différentes présentations et lors du développement.

**Améliorations potentielles**

* Il y a d’abord les améliorations liées au Sprint 3, notifiées plus haut dans ce document
* Et il faudrait envisager de regrouper les commandes en sous dossiers pour pouvoir garder une certaine cohérence et propreté au niveau du code, il faudra en outre adopter une norme sur les noms et les manières de développer puisque le code est un peu anarchique par endroit

· **Plan**

Prendre le temps de trier chaque commande puis les relire une par une après avoir décidé d’une norme à suivre pour tout remettre sur un pied d’égalité avant d’aller plus loin.

#### 

#### Avis sur le développement agile

Cette méthode de développement entre en parfaite adéquation avec la manière dont nous avons choisi de développer notre bot, à chaque sprint il est fonctionnel et peut être utilisé tel quel.Nous ajoutons des fonctionnalités au fur et à mesure selon un patron qui se prête efficacement à la méthodologie agile et il y a toujours du contenu à présenter à la fin des sprint.

C’est une méthode à la fois efficace et intéressante, nous avons beaucoup apprécié développer de cette manière.

## ANNEXES