**UNIVERSITÉ DU QUÉBEC À MONTRÉAL**

**LIVRABLE SPRINT 1**

**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**

**30 OCTOBRE 2019**

**TABLE DES MATIÈRES**

[**1. INTRODUCTION**](#_r9g6q69d332a) **3**

[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) **4**

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

[2.2 Contenu des sprints](#_2bq8x2qdie1t) 5

[**3. CONTENU DU SPRINT ACTUEL**](#_50lsr0f2tkyz) **6**

[3.1 Cas d’utilisation implantés dans le sprint](#_w3x5ntdm14r3) 6

[3.2 Modèle de classe](#_ocragme9dodz) 11

[**4. PROCHAIN SPRINT (1 À 2 PAGES, MAX)**](#_8ws4tvlw1yw9) **14**

[4.1 Revue du sprint](#_5152v3z5dvc5) 14

[4.2 Plan pour prochain sprint](#_9rg7vdaaqzft) 14

[**5. ANNEXES**](#_y8ng3vrjwgp1) **14**

## **1. INTRODUCTION**

### **1.1 Objectifs**

Ce document vise à expliciter ce qui devra être implémenté pour le bon fonctionnement de Personal Assistant, les différentes fonctions seront en outre plus détaillées de manière à faciliter leur développement. En outre, ce document servira de référence aux développeurs.

### **1.2 Vue d’ensemble du produit**

Discord est une application permettant aux gamers de discuter tout en jouant, en outre, un chat vocal est proposé, mais aussi un chat textuel avec de nombreuses options d’administrations, les utilisateurs rejoignent un serveur, puis se voient attribuer des rôles qui leurs permettent de discuter, partager des images, des vidéos, des liens, créer des salons privés, etc…

Afin de palier 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 un panel de fonctionnalité extrêmement large, allant de la simple attribution de rôles automatiquement à des utilisateurs, à la gestion poussé des messages et lancement de serveurs de jeux.

Personal Assistant sera un bot discord, ce qui signifie qu’il ne sera pas utilisable en dehors de cette application. 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…).

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

Nous nommerons le bot PA pour Personal Assistant.

### **1.5 Aperçu du document**

Ce document contient en premier lieu une description générale du logiciel dans laquelle sera expliqué 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 ensuite présentés via des diagrammes de séquence et des protocoles d’utilisation pour chacun des cas. La dynamique des classes est exprimée via un modèle de classes, suivie par un tableau d’attributs pour chacune des classes.

La dernière section présente une revue du premier sprint maintenant complété en passant par nos bons coups et nos faiblesses, puis par des solutions à certains problèmes, suivit par notre plan pour les prochains sprints.

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

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

Les fonctions d’administrations:

* + - 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 réactions : Attribue automatiquement un rôle en fonction de réactions 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)

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 pré faites face à divers messages.
    - Lancement de mini jeux textuels (pierre papier ciseau/ lancer de pièce) : permet de lancer des mini jeux tel qu’un pierre papier ciseau ou un jeu de lancer de pièce
    - Recherche d’image google image/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
    - Demande de la météo pour une zone X : permet d’afficher la météo pour une ville demandée dans le canal textuel
    - Demande de 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 de joueur pour un jeu : affiche dans le canal textuel les statistiques d’un joueur pour une liste de jeu prédéfinie
    - Ramener des infos sur un film : Affiche dans le canal textuel des informations sur le film demandé.

### **2.2 Contenu des sprints**

Premier Sprint: Le but sera d’implémenter une partie ou la totalité des fonctions essentielles au bon fonctionnement du bot. En priorité nous aurons la commande d’attribution des rôles et la lecture de musique via youtube.

Deuxième Sprint: Implémenter les fonctions souhaitables. En priorité nous aurons la suppression automatique des messages par mot(s) clé(s) et les résultats de recherche youtube.

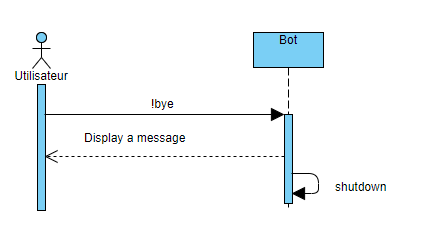
Troisième Sprint: Implémenter ce que l’on a pas pu dans les Sprints précédents et si le temps le permet des fonctions optionnelles. Les recherches de définitions et d’images auront la priorité dans le cas d’implémentation de fonctions optionnelles.

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

### **3.1 Cas d’utilisation implantés dans le sprint**

- arrêt du bot :

1. L’utilisateur entre la commande !bye
2. Le bot reconnaît la commande
3. Le bot envoie un message indiquant qu’il a bien reçu la commande
4. Le programme du bot se termine sur la machine hôte



- la lecture de musique youtube:

Ajout d’une musique :

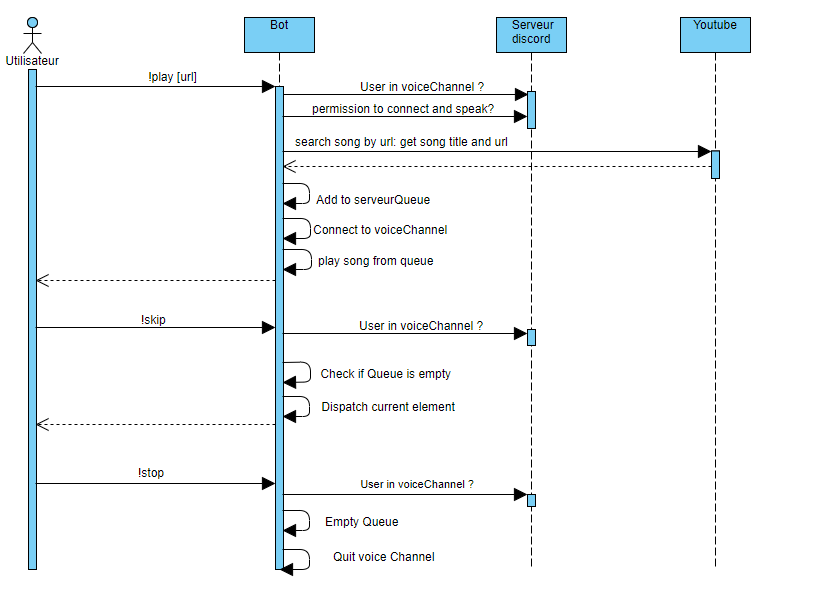
1. L’utilisateur entre la commande !play [url]
2. Le bot vérifie si l’utilisateur est dans un canal vocal
   1. L’utilisateur n’est pas dans un canal vocal
      1. Le bot envoie un message d’erreur
      2. On retourne à l’étape 1
3. Le bot vérifie s’il a le droit d’entrer sur le canal et de communiquer
   1. Le bot n’a pas les permissions requises
      1. Le bot envoie un message d’erreur
      2. On retourne à l’étape 1
4. Le bot cherche ensuite la musique sur youtube via son url et récupère son titre puis en fait un objet song
5. Le bot créé une liste de lecture et ajoute l’objet song
   1. La liste de lecture existe déjà
      1. le bot ajoute son objet song à la suite de la liste existante
      2. On passe à l’étape 6
6. Le bot lit la liste de lecture
7. L’utilisateur entend la musique

Skip de musique

1. L’utilisateur entre la commande !skip
2. Le bot vérifie si l’utilisateur est dans un canal vocal
   1. L’utilisateur n’est pas dans un canal vocal
      1. Le bot envoie un message d’erreur
      2. On retourne à l’étape 1
3. Le bot vérifie s’il a le droit d’entrer sur le canal et de communiquer
   1. Le bot n’a pas les permissions requises
      1. Le bot envoie un message d’erreur
      2. On retourne à l’étape 1
4. Le bot vérifie la liste de lecture
   1. La liste de lecture est vide
   2. Le bot envoie un message d’erreur
   3. Retour à l’étape 1
5. Le bot retire l’objet courant de la liste de lecture

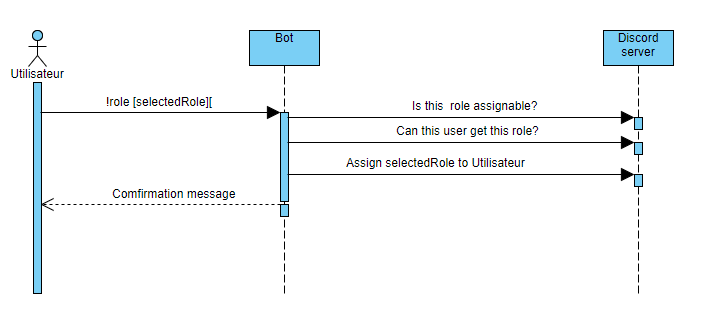
Stop de musique

1. L’utilisateur entre la commande !stop
2. Le bot vérifie si l’utilisateur est dans un canal vocal
   1. L’utilisateur n’est pas dans un canal vocal
      1. Le bot envoie un message d’erreur
      2. On retourne à l’étape 1
3. Le bot vérifie s’il a le droit d’entrer sur le canal et de communiquer
   1. Le bot n’a pas les permissions requises
      1. Le bot envoie un message d’erreur
      2. On retourne à l’étape 1
4. Le bot vide la liste de lecture
5. Le bot quitte le canal vocal



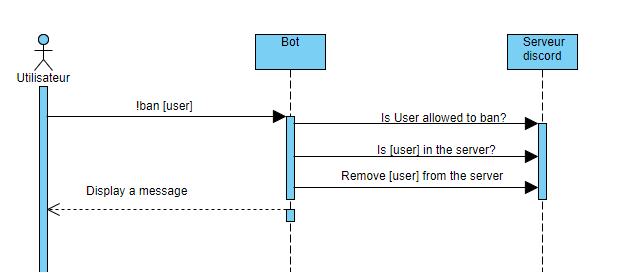
- l’attribution de rôle par message :

1. L’utilisateur entre la commande !role [nom du rôle]
2. Le bot vérifie si le rôle existe
   1. Le rôle n’existe pas
      1. Le bot affiche un message d’erreur
      2. On retourne à l’étape 1
3. Le bot vérifie si l’utilisateur peut attribuer ce rôle (niveau de droits)
   1. L’utilisateur n’a pas le droit de toucher à ce rôle
      1. Le bot affiche un message d’erreur
      2. On retourne à l’étape 1
4. Le bot attribue le rôle
5. Le bot met un message de confirmation dans le chat
6. L’utilisateur voit son nouveau rôle



- la commande de bannissement/kick : ces deux commandes fonctionnent de la même manière

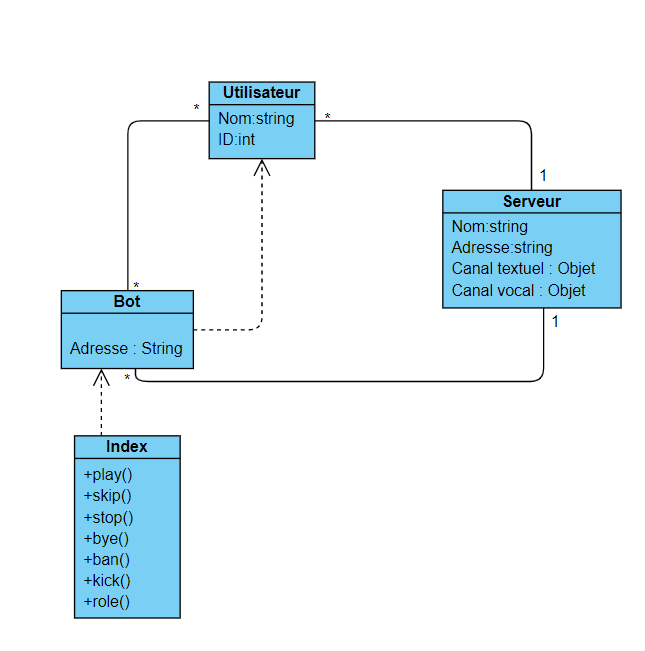
1. L’utilisateur entre la commande !ban [user]/ !kick [user]
2. Le bot vérifie que [user] fait partie du serveur
   1. [user] ne fait pas partie du serveur
      1. Le bot envoie un message d’erreur
      2. Retour à l’étape 1
3. Le bot vérifie que l’utilisateur a les droits suffisants pour bannir/expulser [user]
   1. L’utilisateur n’a pas les droits suffisant
      1. Le bot envoie un message d’erreur
      2. Retour à l’étape 1
4. Le bot expulse/bannit [user du serveur]



| Signature | <Map> <serverQueue>(Queue, song) |
| --- | --- |
| Description | Fonction qui associe une liste de chanson à un voiceChannel |
| Références | Dans les cas d’utilisation de la lecture de musique sur youtube |
| Entrées | • Le nom du paramètre 1: Queue  • Un type abstrait : Map  • La signification : c’est la liste des serveurs sur lesquels est le bot  • Le nom du paramètre 2 : song  • Un type abstrait : objet  • La signification : c’est un objet regroupant url et titre d’une musique |
| Sorties | une Map |
| Exceptions | aucune |
| Pré-conditions | La queue n’est pas vide |
| Post-conditions | Une nouvelle queue est créée associant les voiceChannel à des liste de song |

| Signature | <map> <skip>(<message>< serverQueue>) |
| --- | --- |
| Description | Elle retire la lecture courante de la liste de lecture |
| Références | Dans les cas d’utilisation de la lecture de musique sur youtube |
| Entrées | • Le nom du paramètre 1: message  • Un type abstrait : string  • La signification : c’est le message qui a donné l’ordre au bot de skip  • Le nom du paramètre 2 : serverQueue  • Un type abstrait : map  • La signification : c’est un tableau regroupant la liste de lecture du serveur |
| Sorties | Une nouvelle liste de lecture |
| Exceptions | aucune |
| Pré-conditions | La liste de lecture n’est pas vide |
| Post-conditions | La liste de lecture (serverQueue) verra sa taille réduite de 1 |

### **3.2 Modèle de classe**



Description de la classe:

| Nom | Utilisateur |
| --- | --- |
| Description | La classe qui représente les usagers de discord |
| Attributs | Nom<Nom> Type <String> |
|  | Description <C’est le nom de l’utilisateur sur la plateforme> |
|  | Nom <ID> Type <int> |
|  | Description <C’est l’ID des utilisateurs, utilisé pour différencier les utilisateurs de même nom> |

| Nom | Bot |
| --- | --- |
| Description | La classe qui représente le bot |
| Attributs | Nom <Adresse> Type <String> |
|  | Description <C’est l’adresse du bot pour pouvoir ensuite l’implémenter> |
|  | Nom <ID> Type <int> |
|  | Description <C’est l’ID des utilisateurs, utilisé pour différencier les utilisateurs de même nom> |

| Nom | Index |
| --- | --- |
| Description | C’est la classe qui implémente les fonctions du bot en utilisant son adresse  Attributs. Cette classe contient les différentes méthodes que le bot va utiliser pour réaliser ses commandes. |

| Nom | héritage bot/utilisateur |
| --- | --- |
| Description | La classe bot hérite de la classe utilisateur (le bot étant une sorte d’utilisateur robot) |
| Entité source : utilisateur | Entité destination :bot |

| Nom | Implémentation bot |
| --- | --- |
| Description | La classe Index implémente la classe bot à travers son adresse |
| Entité source : bot | Entité destination :index |

| Nom | communication bot/utilisateur |
| --- | --- |
| Description | Un bot peut communiquer avec plusieurs utilisateurs, un utilisateur peut communiquer avec plusieurs bots |
| Entité source : utilisateur | Entité destination :bot |

| Nom | communication bot/serveur |
| --- | --- |
| Description | Un bot peut communiquer avec le serveur pour obtenir diverses informations (et pour afficher son statut de connexion) |
| Entité source : bot | Entité destination :serveur |

| Nom | communication serveur/utilisateur |
| --- | --- |
| Description | Un utilisateur peut communiquer avec le serveur, notamment pour afficher son statut de connexion |
| Entité source : utilisateur | Entité destination :serveur |

| Nom | Serveur |
| --- | --- |
| Description | C’est la classe qui représente l’un des serveurs sur lequel agit note bot |
| Entité source : utilisateur | Entité destination :serveur |
| Attributs |  |
|  | Nom : Nom Type : String |
|  | Description : C’est le nom du serveur |
|  | Nom : adresse Type : String |
|  | Description : C’est l’adresse du serveur |
|  | Nom : canal textuel Type : liste |
|  | Description : C’est la liste des canaux textuels |
|  | Nom : canal vocal Type : liste |
|  | Description : C’est la liste des canaux vocaux |

## **4. PROCHAIN SPRINT**

### **4.1 Revue du sprint**

*Revue technique*

Il y a eu quelques difficultés du côté de l’apprentissage des technologies et de l’installation des différentes librairies node.js.

Les fonctionnalités essentielles sont globalement fonctionnelles mais pas totalement complétées.   
L'arrêt du bot fonctionne mais le message de feedback n’est pas affiché.

La lecture de musique sur youtube n'inclut pas de pause.

L’identification des rôles pour pouvoir ensuite faire des attributions ou des bans/kicks est plus complexe à gérer que prévue. Afin de palier à ces problèmes les rôles sont pour le moment modifiable sans restriction, le bot ne peut pas bannir d’administrateur pour éviter des débordements pour le moment.

.

*Revue gestion*

Un coéquipier a prit l’initiative sur une partie importante du projet car deux d’entre nous ont l’habitude de travailler à la dernière minute. Notre communication pourrait s’améliorer, mais restait acceptable à travers une conversation facebook de groupe. Nos documents se font sur google doc ce qui permet un accès facile pour tous.

La création des fonctionnalités s’est bien passée, mais le plus gros problème reste au sein d’une répartition égale du projet.

Un coéquipier a une habitude de travailler quelques jours d’avance (heureusement) et les deux autres devront s’y habituer et réaliser l’ampleur de chacun des travaux afin d’aider davantage celui-ci.

### **4.2 Plan pour prochain sprint**

Voici la liste des cas d’utilisation prévus pour le sprint 2 :

* auto suppression des commandes déjà réalisées
* ramener le résultat d’une recherche youtube
* envoie de messages privés aux utilisateurs du discord
* suppression automatique de certains messages par mots clés

## 

## **5. ANNEXES**