

TRAVAIL DE BACHELOR 2018

VRAI PROJECT

VIRTUAL REALITY WITH AI CHATBOT



Étudiant : Rafael Peixoto
Professeur : Antoine Widmer
Déposé, le : 30 juillet 2018
www.hevs.ch

RÉSUMÉ

Malgré un début particulièrement timide et difficile, le secteur de la réalité virtuelle est parvenu à s'imposer ces dernières années sur le marché mondial avec l'arrivée d'une multitude de casques proposant des expériences *VR* aux utilisateurs. En parallèle, le marché des assistants vocaux intelligents connaît également un franc succès et se retrouve en pleine expansion en exposant des intelligences artificielles capables de comprendre le langage naturel et de répondre aux requêtes des différents utilisateurs.

L'objectif principal de ce travail est de développer et intégrer les différentes fonctionnalités d'un assistant vocal dans une application en réalité virtuelle.

Afin d'analyser le potentiel de l'association de ces deux technologies à succès au sein d'une même application en VR, une étude du fonctionnement d'une intelligence artificielle ainsi qu'un état de l'art parcourant les différentes solutions existantes d'agents conversationnels ont été réalisés. À l'aide d'une matrice décisionnelle, nous déterminons la solution la plus adéquate à implémenter dans le cadre de ce projet.

L'application finale réalisée propose à l'utilisateur une expérience en réalité virtuelle lui permettant de communiquer directement avec une intelligence artificielle. En somme, l'utilisateur est en mesure d'avoir une discussion naturelle avec un agent conversationnel et d'interagir avec les objets qui l'entourent par le biais de ce dernier.

Mots-clés : Intelligence artificielle, réalité virtuelle, agent conversationnel, multiplateforme

AVANT-PROPOS

Dans le cadre du travail de Bachelor 2018, proposé et encadré par le professeur Antoine Widmer, l'étudiant a reçu comme directive d'implémenter un agent conversationnel intelligent au sein d'une application en réalité virtuelle. Ce document scientifique est le résultat d'un travail de recherche de trois mois, effectué à la HES-SO Valais-Wallis.

Les objectifs poursuivis par ce travail sont d'établir et d'analyser l'état de l'art des différentes solutions existantes d'agents conversationnels, de réaliser une analyse comparative technique des résultats obtenus, d'établir une analyse de besoins de l'application ainsi que de sélectionner une technologie adaptée au projet en s'appuyant sur les différents résultats obtenus. En définitive, ce travail aboutit en développant et en implémentant une solution en réalité virtuelle qui intègre les différentes fonctionnalités d'un *chatbot*.

Le succès émergeant de ces deux technologies nous a fortement motivés à choisir ce projet en tant que travail de Bachelor.

REMERCIEMENTS

Nous tenons à remercier notre professeur de suivi, M. Antoine Widmer de nous avoir guidé pendant la réalisation de ce travail, mais également de nous avoir laissé une certaine liberté lors du développement de l'application finale.

STRUCTURE DU CD

En annexe de ce document, vous trouvez ci-joint un CD-ROM contenant les dossiers et fichiers suivants :

- 00_VRAI_PROJECT_RAPPORT.pdf
 - Ce fichier PDF contient le rapport écrit du travail de Bachelor
- 01_VRAI_PROJECT_PLANIFICATION.xlsx
 - Ce fichier EXCEL contient la planification, les *user stories* ainsi que les heures effectives réalisées dans le cadre de ce travail.
- 02_VRAI_PROJECT_GUIDE_TECHNIQUE.pdf
 - Ce fichier PDF contient le guide technique de l'application.
- 03_VRAI_PROJECT_CODE_SOURCE
 - SAM
 - Ce dossier contient la totalité du code source de l'application
 - SAM_V1.0_OCULUS_.apk
 - L'application au format APK pour le déploiement sur l'Oculus GO
 - SAM_V1.0_HTC
 - Ce dossier contient l'application au format EXE pour le déploiement sur l'HTC Vive
 - workspace.json
 - Ce fichier JSON contient la configuration complète du service IBM Watson Conversation dont notamment toutes les intentions et entités ainsi que les dialogues de l'application.
- 04_VRAI_PROJECT_POSTER.pptx
- 05_VRAI_PROJECT_GITHUB.txt
 - Ce fichier contient le lien de notre repository git.

TABLE DES MATIÈRES

LISTE DES TABLEAUX	VIII
LISTE DES FIGURES.....	IX
LISTE DES ABRÉVIATIONS	X
INTRODUCTION	1
1. AGENT CONVERSATIONNEL.....	3
1.1. CONTEXTE	3
1.2. FONCTIONNEMENT	5
2. ÉTAT DE L'ART	7
2.1. ANALYSE DE BESOINS.....	8
2.2. CRITÈRES D'ÉVALUATION	9
2.3. IBM WATSON ASSISTANT	10
2.3.1. <i>Fonctionnement</i>	10
2.3.2. <i>Services supplémentaires</i>	11
2.3.3. <i>Intégration</i>	12
2.3.4. <i>Langues</i>	13
2.3.5. <i>License</i>	13
2.4. MICROSOFT BOT FRAMEWORK	14
2.4.1. <i>Fonctionnement</i>	14
2.4.2. <i>Services supplémentaires</i>	15
2.4.3. <i>Intégration</i>	16
2.4.4. <i>Langues</i>	17
2.4.5. <i>License</i>	17
2.5. DIALOGFLOW	18
2.5.1. <i>Fonctionnement</i>	18
2.5.2. <i>Intégration</i>	20
2.5.3. <i>Langues</i>	22
2.5.4. <i>License</i>	23
2.6. AMAZON LEX	24
2.6.1. <i>Fonctionnement</i>	24
2.6.2. <i>Services supplémentaires</i>	25
2.6.3. <i>Intégration</i>	26
2.6.4. <i>Langues</i>	26
2.6.5. <i>License</i>	26

2.7.	WIT.AI	27
2.7.1.	Intégration.....	27
2.7.2.	Langues	28
2.7.3.	License.....	28
2.8.	PANDORABOTS.....	29
2.8.1.	Intégration.....	29
2.8.2.	Langues	30
2.8.3.	License.....	30
3.	CHOIX TECHNOLOGIQUE	30
3.1.	ANALYSE COMPARATIVE	32
3.2.	SYNTHÈSE.....	33
4.	CHOIX DES OUTILS ET TECHNOLOGIES DE DÉVELOPPEMENT	34
4.1.	SUPPORTS TECHNIQUES	34
4.1.1.	HTC Vive.....	34
4.1.2.	Oculus GO	35
4.2.	KITS DE DÉVELOPPEMENT	36
4.2.1.	Watson APIs Unity SDK.....	36
4.2.2.	SteamVR SDK	36
4.2.3.	Oculus Utilities SDK	36
4.2.4.	VRTK SDK	37
4.3.	SERVICES.....	37
4.3.1.	IBM WATSON Conversation.....	37
4.3.2.	IBM WATSON Speech to Text et Text to Speech.....	40
4.4.	OUTILS ET ENVIRONNEMENTS DE DÉVELOPPEMENT.....	41
4.4.1.	GitHub	41
4.4.2.	GitHub Desktop.....	41
4.4.3.	Unity.....	43
4.4.4.	Visual Studio	43
5.	DÉVELOPPEMENT.....	43
5.1.	DESCRIPTION DE L'APPLICATION	44
5.2.	ARCHITECTURE DE L'APPLICATION	45
5.3.	ARBORESCENCE DU PROJET	46
5.3.1.	Arborescence	46
5.4.	CONFIGURATION DE L'AGENT CONVERSATIONNEL.....	46

5.4.1.	<i>Artéfacts</i>	46
5.4.2.	<i>Diagramme de conversation</i>	49
5.5.	DIFFICULTÉS RENCONTRÉES.....	51
5.5.1.	<i>Reconnaissance vocale pertinente</i>	51
5.5.2.	<i>Latence d'enregistrement</i>	51
5.5.3.	<i>Interaction avec l'environnement</i>	52
5.5.4.	<i>Personnification</i>	52
5.5.5.	<i>Historique des versions</i>	53
5.5.6.	<i>Matériel, mise à jour et multiplateforme</i>	54
5.6.	APPLICATION.....	55
5.6.1.	<i>Interaction avec l'environnement</i>	55
5.6.2.	<i>Conversation générique et banalités</i>	58
6.	BILAN ET PERSPECTIVES	58
6.1.	AMÉLIORATIONS ET OPTIMISATIONS TECHNIQUES.....	58
6.1.1.	<i>Amélioration de l'architecture et de la latence</i>	58
6.1.2.	<i>Apprentissage et création automatique des intentions</i>	59
6.2.	MATURITÉ DE LA RÉALITÉ VIRTUELLE ET ÉVOLUTION DE L'INTELLIGENCE ARTIFICIELLE	60
	CONCLUSION	62
	ANNEXE I : LISTE D'EXEMPLES DE REQUÊTES POUR CHAQUE INTENTION	67
	ANNEXE I : PLANIFICATION	71
	ANNEXE II : PRODUCT BACKLOG	72
	DÉCLARATION DE L'AUTEUR	73

LISTE DES TABLEAUX

Tableau 1 - Environnements de développements compatibles avec IBM.....	12
Tableau 2 - Liste des langues entièrement prises en charge par IBM Watson Assistant Watson	13
Tableau 3 - Liste des langues avec leur degré de prise en charge par LUIS	17
Tableau 4 - Détails de tarification de LUIS, Bing Speech API et Microsoft Bot Framework.....	18
Tableau 5 - Environnements de développements compatibles avec Dialogflow	21
Tableau 6 - Environnements de développements compatibles avec Dialogflow V1.....	22
Tableau 7 - Liste des langues prises en charge par Dialogflow	23
Tableau 8 - Détails de tarification de Dialogflow	24
Tableau 9 - Environnements de développements compatibles avec Amazon Lex.....	26
Tableau 10 - Détails de tarification d'Amazon Lex.....	27
Tableau 11 - Environnements de développements compatibles avec WIT.AI	28
Tableau 12 - Langues principales prises en charge par WIT.AI	28
Tableau 13 - Environnements de développements compatibles avec Pandorabots.....	29
Tableau 14 - Environnements de développements non officiels compatibles avec Pandorabots	29
Tableau 15 - Détails de tarification de Pandorabots	30
Tableau 16 : Matrice décisionnelle des agents conversationnels	33
Tableau 17 - Tableaux comparatifs des principaux casques de réalité virtuelle	34
Tableau 18 - Liste des interactions avec l'environnement	57
Tableau 19 - Liste des thématiques accessibles depuis l'agent conversationnel.....	58

LISTE DES FIGURES

Figure 1 - Évolution mondiale du marché de la réalité virtuelle de 2016 à 2020.....	1
Figure 2 - Évolution mondiale du marché des agents conversationnels de 2016 à 2025	3
Figure 3 - Traitement automatique du langage naturel	6
Figure 4 - Architecture d'une solution utilisant <i>IBM Watson Assistant</i>	10
Figure 5 - Architecture d'une solution utilisant Microsoft Bot Framework	14
Figure 6 - Architecture d'une solution utilisant Dialogflow	19
Figure 7 - Fonctionnement d'Amazon Lex dans une solution informatique	24
Figure 8 - Moteurs graphiques utilisés pour le développement de jeux mobiles.....	31
Figure 9 - Capture d'écran de notre plateforme de gestion des espaces de travail sur IBM Watson.....	38
Figure 10 - Capture d'écran de la création d'une intention sur IBM Watson Conversation....	38
Figure 11- Capture d'écran de la création d'une entité sur IBM Watson Conversation.....	39
Figure 12 - Capture d'écran de la création d'un dialogue sur IBM Watson Conversation	39
Figure 13 - Vérification de la configuration d'un agent sur IBM Watson Conversation	40
Figure 14 - Interface du client GitHub Desktop Unity	42
Figure 15 - Premier aperçu de l'application en réalité virtuelle SAM	44
Figure 16 - Architecture de l'application développée	45
Figure 17 - Liste partielle des intentions configurées pour IBM Watson Conversation.....	47
Figure 18 - Exemple des différentes formulations pour un objectif identique	48
Figure 19 - Embranchements possibles lors d'une conversation avec un utilisateur.....	50
Figure 20 - Fichier .gitignore destiné à l'outil <i>Unity</i>	54
Figure 21 - Interaction avec la planète Terre	56
Figure 22 - Interaction avec le drapeau américain de la mission Apollo 11	56
Figure 23 - Interaction avec le module lunaire de la mission Apollo 11.....	57
Figure 24 - Amélioration de l'architecture de l'application	59
Figure 25 - Apprentissage et création automatique des intentions	60
Figure 26 - Exemples de requêtes pour chaque catégorie (partie 1)	67
Figure 27 - Exemples de requêtes pour chaque catégorie (partie 2)	68
Figure 28 - Exemples de requêtes pour chaque catégorie (partie 3)	69
Figure 29 - Exemples de requêtes pour chaque catégorie (partie 4)	70
Figure 30 - Planification du projet.....	71
Figure 31 - Product backlog et nombre d'heures effectives	72

LISTE DES ABRÉVIATIONS

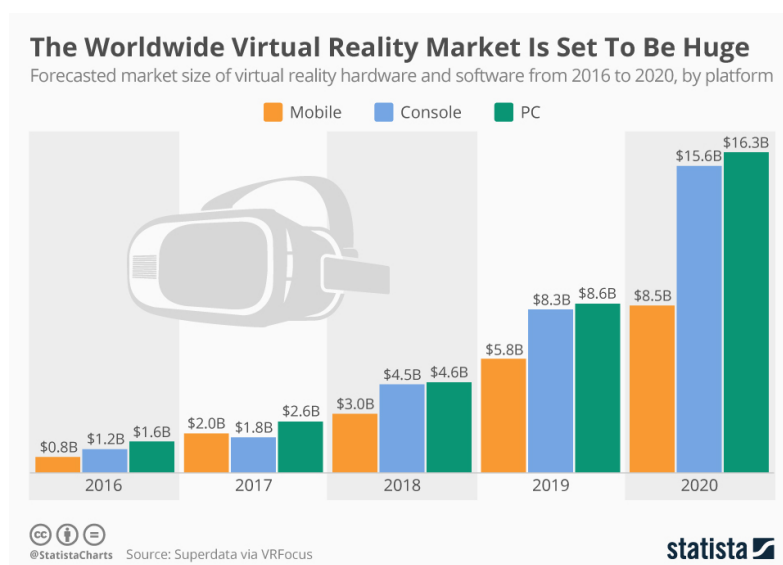
AI	Intelligence Artificielle
AIML	Artificial Intelligence Markup Language
API	Application Programming Interface
C#	C Sharp
CHATBOT	Agent Conversationnel
HTTP	Hypertext Transfer Protocol
IDE	Integrated development environment
LN	Langage Naturel
LUIS	Language Understanding
REST	Representational State Transfer
SDK	Software development kit
SSML	Speech Synthesis Markup Language
VCS	Version Control System
VR	Réalité Virtuelle
3D	3 Dimensions
2D	2 Dimensions

INTRODUCTION

Durant les trente dernières années, le secteur de l'industrie vidéoludique a connu une évolution particulièrement remarquable. Les premiers développeurs ont réussi au fil des années à étoffer la richesse de *gameplay* de leurs jeux vidéo considérés au départ comme une simple curiosité passagère. En parallèle, les manettes de jeu ont subi au même moment des améliorations majeures jusqu'à l'arrivée de nouveaux périphériques révolutionnaires tels que les détecteurs de mouvements et les casques de réalité virtuelle (Peixoto, 2015, p. 1).

Malgré un début particulièrement timide et difficile, le secteur de la réalité virtuelle est parvenu à s'imposer ces quatre dernières années. En effet, ce nouveau marché émergent se retrouve actuellement en pleine expansion. L'essor de ce dernier est tellement important que selon *statista*, la taille du marché de la réalité virtuelle devrait quadrupler d'ici 2020. Le schéma suivant présente l'évolution mondiale du marché de la réalité virtuelle de 2016 à 2020 tous supports confondus.

Figure 1 - Évolution mondiale du marché de la réalité virtuelle de 2016 à 2020



Source : (Armstrong, 2016)

L'opinion populaire associe généralement l'usage de la réalité virtuelle aux jeux vidéo. Il est vrai que celle-ci tend à devenir le nouveau support de l'industrie vidéoludique permettant à l'utilisateur de découvrir une toute nouvelle expérience particulièrement immersive. Cependant, la réalité virtuelle ne s'adresse pas uniquement à cet usage. Comme nous avons pu le remarquer ces dernières années, les applications sont diverses.

En effet, la réalité virtuelle est actuellement utilisée par les pilotes lors des phases de formation et d'entraînement. Celle-ci permettrait également de surmonter aisément la peur du vertige. La VR offre également la possibilité aux architectes de visualiser de futurs bâtiments en trois dimensions. Il

ne faut pas oublier les applications de divertissement. Cette technologie peut donner lieu à des visites virtuelles interactives dans des musées et des expositions (Desai, Desai, Ajmera, & Mehta, 2014, p. 179).

En ce qui concerne le domaine de l'éducation, dans la revue « The potentials of virtual environments in the education and training of people with learning disabilities », publié dans le volume 40 du journal *Journal of Intellectual Disability Research*, J. J. Cromby relève que l'utilisation d'environnements entièrement virtuels permettrait la formation cognitive des personnes handicapées mentales (1996, p. 489). J. J. Cromby constate que le potentiel de la réalité virtuelle est à son apogée lorsque l'expérience se trouve être la plus immersive possible.

Il est donc intéressant de se demander par quel moyen nous pouvons améliorer l'immersion de la réalité virtuelle dont le potentiel n'est plus à prouver. Dans l'article « L'intelligence artificielle : un enjeu d'économie et de civilisation ? », publié dans la série trimestrielle *Enjeux numériques*, Yan Georget confirme le potentiel conséquent de l'intelligence artificielle (2018, p. 44).

En effet, Yan Georget affirme que celle-ci possède les caractéristiques nécessaires pour devenir la nouvelle interface utilisateur. Les agents conversationnels (*chatbots*) offriraient la possibilité de remplacer nos interfaces graphiques de plus en plus vieillissantes au profit d'une interface entièrement conversationnelle. De plus, l'auteur déclare que « la conversation est une interface flexible et facile d'accès puisqu'elle ne demande pratiquement aucun apprentissage. La conversation serait donc l'interface ultime ! »

L'intégration d'un *chatbot* dans une application de réalité virtuelle permettrait en outre de se passer de toute interface graphique au profit d'une expérience plus immersive et plus prenante. En conséquence, l'utilisateur ne serait plus contraint à interagir avec une simple machine, mais avec une machine intelligente dotée de caractéristiques plus humaines.

Dans un premier temps, nous allons dans ce document comprendre le fonctionnement général des *chatbots* et parcourir les différentes offres existantes dans le marché de ces agents conversationnels. Dans un deuxième temps, nous entreprendrons la réalisation d'une analyse comparative entre les diverses solutions afin de choisir au mieux une technologie à implémenter dans le cadre de ce projet. Dans un dernier temps, nous allons essayer de démontrer la faisabilité de ce travail en implémentant un agent en réalité virtuelle.

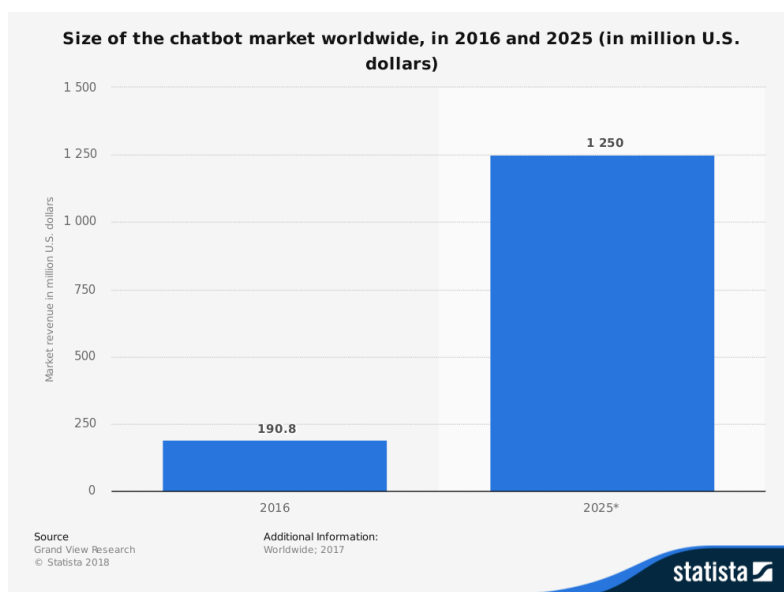
1. AGENT CONVERSATIONNEL

Un agent conversationnel, également appelé *chatbot* est une solution informatique dotée d'une intelligence artificielle simulant le comportement humain le plus proche possible. L'agent est en mesure par le biais d'une analyse sémantique des textes écrits de comprendre le langage naturel des entrées saisies par un utilisateur, mais également d'y répondre de manière cohérente.

1.1. CONTEXTE

Dans le cadre de ce travail, nous avons reçu comme directive d'implémenter un agent conversationnel en réalité virtuelle. En effet, ces programmes intelligents ont connu un réel succès ces dernières années. L'essor de cette technologie est tellement important que selon les prévisions de *statista* la taille du marché mondial des agents conversationnels devrait être multipliée par six d'ici 2025. L'histogramme suivant présente l'évolution mondiale du marché des *chatbots* de 2016 à 2025 en million de dollars américains.

Figure 2 - Évolution mondiale du marché des agents conversationnels de 2016 à 2025



Source : (Statista, 2017)

Les utilisateurs commencent effectivement à percevoir de plus en plus les bienfaits apportés par ceux-ci. Les scénarios d'utilisation dans une solution informatique conversationnelle sont effectivement nombreux.

Nous retrouvons de plus en plus d'assistants vocaux dans le domaine de la domotique permettant un contrôle vocal à distance d'une multitude d'appareils électroniques et électroménagers. Ceux-ci se trouvent même déjà directement intégrés dans nos *smartphones* à travers les très célèbres assistants personnels tels que *Siri*, *Google Now*, *Alexa* ainsi que *Cortana*. Les systèmes de messagerie quant à eux ne sont pas épargnés. Il est actuellement même possible d'être amené à discuter avec un agent conversationnel sur des plateformes telles que *Messenger*, *Skype*, *Slack* et *Telegram*.

Malgré leur forte présence sur de nombreuses plateformes et supports, il est important de s'interroger sur leur raison d'être. Ces solutions conversationnelles ont une multitude d'avantages. En tant qu'assistant personnel, un *chatbot* est en mesure d'informer l'utilisateur des actualités importantes sur un sujet particulier, de lui envoyer un rappel pour un rendez-vous et même d'organiser son emploi du temps pour lui. Néanmoins, les fonctionnalités sont diverses et ne s'arrêtent pas là.

Comme nous avons pu le voir pendant la *Google I/O Keynote 2018*, l'intelligence artificielle de ces assistants a particulièrement progressé en très peu de temps. En effet, l'assistant de *Google* est à présent même capable d'appeler un coiffeur ou même un restaurant pour prendre un rendez-vous à la place de l'utilisateur. La conversation entre la machine et l'homme est dirigée de manière fluide, cohérente et surtout naturelle.

Les résultats de cette expérience sont particulièrement étonnants et stupéfiants. À l'aide de puissants algorithmes comme le *machine learning* et le *deep learning*, ces programmes révolutionnaires sont en mesure d'imiter parfaitement l'homme et son comportement. En définitive, les progrès sont tellement fulgurants que l'intelligence artificielle de ces programmes surpasse peut-être le célèbre test de Turing.

Comme nous pouvons le remarquer, les agents conversationnels sont devenus les parfaits assistants personnels. Nous faisons face actuellement à un nouveau genre d'interface utilisateur, l'interface utilisateur conversationnelle. Dans l'article « Spoken Dialogue Technology: Enabling the Conversational User Interface », publié dans le volume 34 du journal *ACM Computing Surveys*, Michael F. McTear relève que le développement d'interfaces utilisateur conversationnelles était déjà d'actualité en 1950 (2002, p. 90). Cependant, ce dernier constate que ce n'est que très récemment avec les progrès majeurs des systèmes de reconnaissances vocales et l'évolution des intelligences artificielles qu'il nous est à présent possible de développer de nouvelles applications à large échelle et donc de mettre en pratique une interface utilisateur conversationnelle.

L'utilité de cette toute nouvelle interface prend tout son sens lorsqu'on l'applique en réalité virtuelle. En effet, l'objectif principal de celle-ci est bien d'immerger l'utilisateur que ce soit dans une réalité virtuelle parfois proche du réel ou une réalité alternative. L'expérience est bien évidemment saisissante pour les utilisateurs, car l'immersion est totale avec l'absence d'ancrage dans

la réalité. Cependant, il suffit d'avoir d'anciens artefacts comme une interface uniquement graphique pour briser l'expérience d'immersion. Néanmoins, l'utilisation alternative d'une interface conversationnelle permettrait à l'utilisateur d'être totalement immergé dans cette réalité virtuelle. Ainsi, celui-ci n'interagirait plus avec son environnement virtuel à l'aide uniquement d'une interface graphique, mais avec un agent conversationnel. L'assistant personnel serait effectivement en mesure de comprendre l'utilisateur et de lui répondre en conséquence en lui donnant des informations relatives à son environnement, en l'assistant avec différentes interactions et surtout en ayant une simple conversation naturelle avec celui-ci.

C'est la raison pour laquelle nous allons essayer de comprendre le fonctionnement général de ces agents conversationnels afin de nous permettre de parcourir la liste des solutions existantes dans le marché des interfaces conversationnelles dans le but de réaliser par la suite une analyse comparative entre les différentes plateformes. Notre objectif final étant toujours d'implémenter une intelligence artificielle en réalité virtuelle, les différentes études susmentionnées nous permettront d'accélérer le processus de développement.

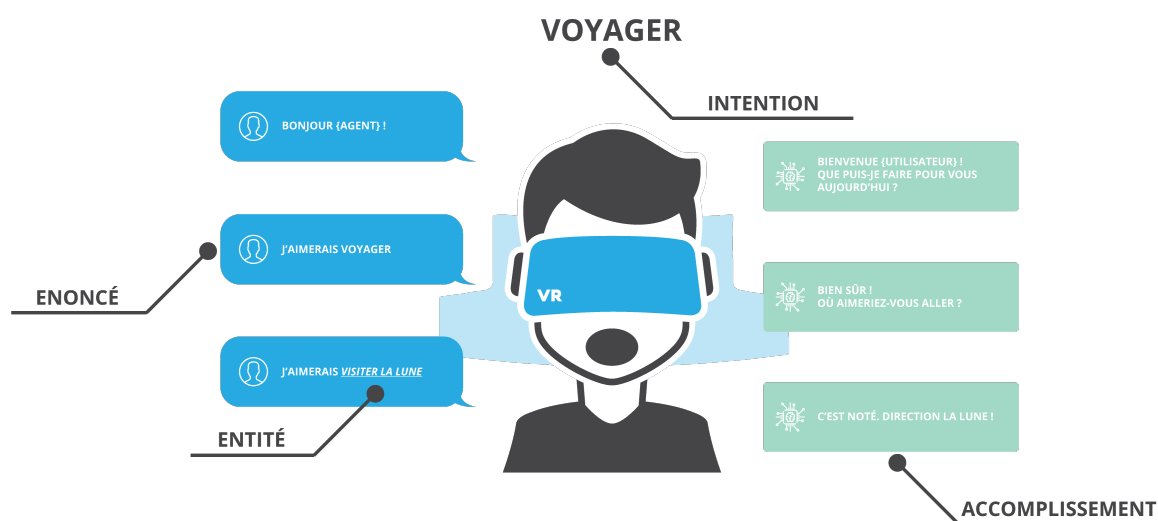
1.2. FONCTIONNEMENT

Le service proposé par un agent conversationnel repose principalement sur l'extraction des informations pertinentes depuis un message textuel ou vocal provenant d'un utilisateur. Cette extraction de données couplée au *machine learning* permet à l'agent de comprendre le langage naturel de l'utilisateur. En effet, l'homme s'exprime instinctivement en langage naturel. C'est cette langue qui permet une interaction aux caractéristiques humaines entre l'utilisateur et le *chatbot*. En outre, celle-ci est en mesure de fluidifier la conversation, de construire un dialogue cohérent et surtout d'humaniser la machine aux yeux de l'utilisateur.

La compréhension et l'apprentissage automatique du LN représentent le principal obstacle dans le dialogue entre l'homme et la machine. Dans la thèse « Dialogue homme-machine et apprentissage. Apprentissage par l'interaction », Caelen J. et Villaseñor L. relèvent la difficulté de modéliser manuellement toutes les interactions et tous les dialogues imaginables lors d'une conversation entre l'homme et la machine. Les deux auteurs envisagent déjà en 1997 la réalisation d'algorithmes d'apprentissage automatiques du langage naturel pour résoudre ce problème (1997, p. 1). En outre, l'apprentissage profond avancé communément appelé *machine learning* permet à l'agent conversationnel d'apprendre par le dialogue de manière itérative. Ainsi, la machine est en mesure de devenir de plus en plus intelligente proportionnellement aux nombres d'interactions déjà réalisées avec d'autres utilisateurs. Le caractère naturel des conversations ainsi que la pertinence des réponses fournies par l'agent évolueront donc au fil du temps.

Dans leur thèse, Caelen J. et Villaseñor L. relèvent également l'importance de l'action au sein d'un dialogue. En effet, les auteurs constatent qu'un dialogue se construit par une suite coordonnée d'actions menant vers un objectif particulier (1997, p. 3). C'est la raison pour laquelle ce traitement automatique du langage naturel est réalisé à l'aide de plusieurs paradigmes conceptuels. Le schéma suivant présente le traitement automatique du langage naturel d'une conversation par un agent conversationnel.

Figure 3 - Traitement automatique du langage naturel



Source : Données de l'auteur

Comme nous pouvons le remarquer, le traitement automatique d'un dialogue utilise des concepts tels que les intentions, les énoncés, les entités ainsi que les accomplissements.

Lors d'un dialogue entre un homme et une machine, l'intention (intent) représente l'objectif principal que l'utilisateur cherche à atteindre. Dans notre exemple, l'intention perçue par l'agent conversationnel est de voyager. Ce dernier est l'élément le plus important du dialogue, car celui-ci détermine l'action que le chatbot devra exécuter.

Les énoncés (utterances) permettent simplement de définir toutes les requêtes transmises à l'agent conversationnel ayant la même intention commune. Ces messages provenant des utilisateurs peuvent être à la fois des énoncés textuels ou vocaux selon la plateforme déployée. Sur le schéma ci-dessus, un énoncé est donc illustré par un rectangle bleu. Plus le nombre d'exemples d'énoncés sera important, plus l'agent conversationnel sera en mesure de déterminer aisément l'intention d'un utilisateur.

Les entités (entities) quant à elles déterminent les données et informations importantes reliées à une intention spécifique permettant à l'agent conversationnel de répondre à l'objectif de l'utilisateur. Les entités représentent donc une liste de mots pertinents pour le *chatbot*. En

reconnaissant une entité dans un message provenant de l'utilisateur, l'intelligence artificielle est en mesure de choisir une réponse spécifique à prendre pour satisfaire l'objectif du client. Dans l'exemple ci-dessus, l'agent n'est pas en mesure d'accomplir l'intention de l'utilisateur sans avoir à disposition la destination du voyage.

Finalement, l'accomplissement (fulfillment) marque le succès et donc la réussite d'une intention par la machine. En effet, le but principal visé par un agent est de répondre au mieux à la demande de l'utilisateur. Il est très important pour l'agent d'être en mesure de déterminer le succès ou non d'une intention afin de diriger le flux de conversation. Un flux peut être illustré par tous les embranchements possibles dans une conversation.

La direction d'une conversation est donc dépendante de la réussite des accomplissements alors que les embranchements dépendent du nombre d'intentions possibles.

2. ÉTAT DE L'ART

L'analyse suivante a pour but de parcourir en détail la liste des différents agents conversationnels disponibles dans le marché actuel. Le résultat de celle-ci nous permettra de réaliser un *benchmarking* des différentes solutions afin d'avoir à disposition tous les éléments nécessaires à la décision de la technologie la plus adéquate à implémenter dans le cadre de l'application.

Il est important de remarquer que cette liste n'est pas exhaustive. En effet, nous avons déjà effectué une présélection des différents *chatbots* disponibles. Cette présélection se base principalement sur le langage de programmation utilisé et disponible pour ces différents agents conversationnels. L'objectif de ce travail étant d'implémenter un *chatbot* en réalité virtuelle à travers le moteur de création *Unity*, nous avons été contraints de retenir uniquement les solutions supportant le langage C#. De plus, ce tri préalable a également été fortement influencé par la notoriété des agents d'autant plus qu'il est plus aisé de trouver un large éventail de documentation technique lorsqu'une forte communauté est présente derrière une technologie.

Cet état de l'art a pour principal objectif de rassembler le maximum d'informations sur les agents conversationnels du point de vue technique, mais également économique. Par le biais de celui-ci, nous serons en mesure de mettre en évidence les avantages et les désavantages de chacune des solutions disponibles afin de réaliser une analyse comparative.

Cette analyse de l'existant se fera en plusieurs étapes. Dans un premier temps, nous allons effectuer une analyse de besoins sommaire de l'application à développer. Celle-ci nous permettra de mettre en avant les fonctionnalités principales nécessaires à la fin de ce travail. À la suite de cette analyse, nous serons en mesure de détailler une liste de critères d'évaluation commune à chaque agent conversationnel afin d'évaluer chaque solution sur un même pied d'égalité. Dans un deuxième

temps, nous allons passer en revue les sites internet des constructeurs de chaque agent ainsi que de leur documentation technique dans le but d'analyser le fonctionnement, les différents moyens d'intégration, les langues supportées ainsi que leur modèle économique.

2.1. ANALYSE DE BESOINS

L'objectif de ce travail étant d'implémenter et intégrer un agent conversationnel en réalité virtuelle, il nous est nécessaire d'énumérer les différentes fonctionnalités indispensables pour la réalisation de l'application ainsi que ses besoins fondamentaux.

L'application finale devra supporter nativement la réalité virtuelle. Étant donné que le développement de l'application sera réalisé à l'aide du célèbre logiciel de création de contenu 3D *Unity*, le *chatbot* devra être compatible avec le langage de programmation C#. C'est la raison pour laquelle, nous devons prendre en compte les soucis de compatibilité avec cet environnement de développement. L'agent conversationnel ne devra donc pas dépendre d'autres logiciels tiers.

À la suite d'une discussion avec notre professeur de suivi, nous avons décidé de nous concentrer sur l'*HTC Vive* comme plateforme de déploiement. En effet, nous ne prendrons pas en compte les autres supports mobiles de réalité virtuelle tels que le *GearVR* et le *Google Cardboard*.

Il est important de remarquer que l'objectif de ce projet n'est pas de réaliser une application VR, mais bien d'intégrer un *chatbot* en VR. Ainsi, nous ne sommes pas contraints à développer un univers spécifique et dans le même sens de proposer une solution à une problématique. En effet, l'utilisateur devra simplement être capable de communiquer de manière orale avec une intelligence artificielle. Celle-ci devra être en mesure de répondre aux questions de l'utilisateur, mais également de tenir une conversation cohérente avec celui-ci. La conversation doit se faire uniquement de manière orale et non pas par écrit. C'est pourquoi nous allons également analyser brièvement tous les services additionnels pour chaque solution d'agent conversationnel.

L'utilisateur doit avoir la possibilité de contrôler l'univers 3D à l'aide certaines commandes vocales spécifiques par le biais du *chatbot*. Il est donc important de pouvoir configurer l'agent à l'aide de commandes vocales très distinctives. L'intelligence artificielle se doit de remplacer toute interface graphique usuelle afin d'améliorer l'immersion de la réalité virtuelle. Celle-ci doit jouer le rôle d'un assistant vocal. En conséquence, l'utilisateur ne doit pas avoir l'impression de communiquer avec une simple machine, mais plutôt avec un humain. C'est la raison pour laquelle, la configuration de l'assistant vocale doit nous permettre de lui donner certains traits humains.

En définitive, toute communication avec l'assistant vocal doit pouvoir être initiée par l'utilisateur à tout moment. C'est pourquoi l'assistant ne doit pas être capable de se reconnaître à travers un nom propre uniquement dans le but d'initialiser la conversation avec l'utilisateur. De plus, l'intelligence

artificielle doit être en mesure de comprendre et de répondre au minimum en deux langues différentes.

2.2. CRITÈRES D'ÉVALUATION

Nous allons dans cette section énumérer les différents critères d'évaluation que nous allons juger pour chaque assistant vocal. Ces critères se basent principalement sur les besoins de l'application finale, mais également sur les prérequis techniques du logiciel *Unity*.

Il est nécessaire de nous rappeler que ce travail de Bachelor sera utilisé par la suite en tant que référence pour toute future implémentation d'agent conversationnel en réalité virtuelle au sein de l'Institut Informatique de Gestion de la HES-SO Valais-Wallis. C'est la raison pour laquelle l'existence d'une documentation technique détaillée constitue un critère d'évaluation important. De plus, la présence d'une forte communauté supportant l'agent conversationnel représente une plus-value conséquente pour tout aspect n'étant pas détaillé dans le cadre de ce document.

La notoriété d'une solution ne jouera pas un rôle particulier en tant que critère d'évaluation. Cependant, il est nécessaire de noter que plus la célébrité d'une technologie est importante plus il sera aisé d'obtenir des informations ainsi que de l'aide la concernant.

Les langages de programmation supportés par un *chatbot* représentent un critère primordial. En effet, les scripts utilisés dans le logiciel *Unity* supportent uniquement le langage C#. Dans ce sens, il nous sera uniquement possible de retenir les solutions compatibles avec ce langage développé par Microsoft.

À la suite de l'analyse sommaire des besoins de l'application, nous avons convenu qu'il était nécessaire que l'agent conversationnel soit international et donc supporte au minimum deux langues différentes. Il est évident que l'anglais doit impérativement faire partie de cette liste à l'instar du français.

La facilité d'intégration est également à prendre en compte. En effet, si une solution comporte déjà un kit de développement (SDK) prévu pour *Unity*, l'implémentation de l'agent conversationnel sera particulièrement facilitée. Dans le cas contraire, il nous sera nécessaire de prévoir plus de temps pour le développement de l'application afin d'intégrer le *chatbot*.

Le nombre de services supplémentaires proposé par l'agent représente pour nous également un critère d'évaluation particulièrement important. Dans le but de réussir le développement de la solution, nous avons besoin d'un service de synthèse vocale ainsi que d'un service de reconnaissance vocale. Ces derniers sont indispensables pour le bon fonctionnement de l'application. Dans le cas où la fonctionnalité n'est pas présente, nous serons contraints à faire appel à un service tiers pour nous permettre d'implémenter cette fonctionnalité.

Finalement, nous devons également prendre en considération le modèle économique pour chaque plateforme. Nous ne recevons pas d'aide financière pour la réalisation de ce travail. Il nous est donc préférable de faire appel idéalement à une solution gratuite sans frais supplémentaires.

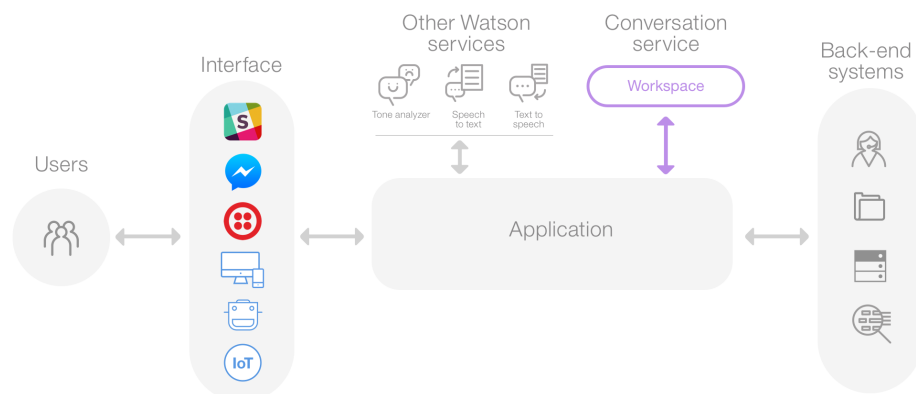
2.3. IBM WATSON ASSISTANT

Anciennement IBM Watson Conversation, IBM Watson Assistant est un assistant conversationnel utilisant l'intelligence artificielle conçue par IBM. Celui-ci est capable d'interpréter un message écrit en langage naturel provenant d'utilisateurs. En somme, Watson utilise l'apprentissage automatique afin de répondre aux utilisateurs. Il est en mesure de simuler une conversation cohérente entre des humains (Watson, Watson Assistant, 2018).

2.3.1. Fonctionnement

Dans l'architecture d'une solution classique, les différents services proposés par IBM Watson Assistant sont appelés depuis un *middleware* chargé des échanges d'informations entre les différentes couches informatiques. Le diagramme suivant présente de manière sommaire le fonctionnement de IBM Watson Assistant dans l'architecture d'une solution informatique.

Figure 4 - Architecture d'une solution utilisant *IBM Watson Assistant*



Source : (Watson, Watson Assistant, 2018)

Nous pouvons donc remarquer que Watson joue un rôle de middleware dans l'architecture d'une solution. En effet, les utilisateurs interagissent uniquement avec l'interface de l'application en question. Les informations enregistrées sont ensuite envoyées au cœur de la plateforme.

Dans le cas d'une utilisation basique du service, celles-ci sont directement transmises au service de conversation de l'agent conversationnel aussi appelé espace de travail. C'est dans ce *workspace* que le message provenant de l'utilisateur sera automatiquement traité par IBM Watson. Cet espace

contient toutes les données d'apprentissage de l'assistant ce qui va lui permettre de s'entraîner et d'apprendre au fil du temps.

En définitive, cette plateforme va permettre de diriger le flux de la conversation du *chatbot* avec les utilisateurs. Celle-ci sera capable d'évoluer au fur à mesure du nombre d'interactions avec les utilisateurs. L'espace est entièrement configurable à travers une interface graphique par les développeurs à l'aide de différents artéfacts tels que les Intentions, Entités et les Dialogues. Les Intentions permettront de définir une liste d'objectifs des différents utilisateurs. Les Entités quant à elles sont liées directement à des Intentions afin de leur donner un contexte concret. Finalement, les Dialogues représentent les différentes directions et embranchements que pourra prendre l'assistant lors d'une conversation avec un client (Watson, Watson Assistant, 2018).

2.3.2. Services supplémentaires

Il est important de noter que IBM Watson est fourni avec plusieurs autres services. Ces services supplémentaires peuvent être facilement connectés à l'architecture de base afin d'analyser plus précisément le message entrant d'un client. Dans le contexte de notre travail, nous aurons donc la possibilité de faire appel également aux services suivants.

Speech to Text

Le service *Speech to Text* est une *API* conçue par IBM Watson permettant de retranscrire à l'écrit un enregistrement vocal en temps réel. Cette option se trouve être particulièrement importante dans le cadre de notre projet. En effet, il est absolument impératif que l'utilisateur soit capable de discuter à l'oral directement avec l'agent conversationnel. Au total, *Speech to Text* supporte sept langages en temps réel. De plus, le service est compatible avec la plupart des types d'interfaces tels que les interfaces *WebSocket*, *HTTP REST* ainsi que les interfaces asynchrones *HTTP* (Watson, Watson *Speech to Text*, 2018). C'est la raison pour laquelle l'*API* est donc intégrable dans n'importe quelle solution.

Text to Speech

Le service *Text to Speech* est une *API* conçue par IBM Watson permettant la synthèse vocale réaliste d'un texte écrit. Celui-ci supporte également sept langages différents, mais il est également fourni avec plusieurs tonalités différentes. Au contraire de *Speech to Text*, la solution est compatible uniquement avec les interfaces *WebSocket* et *HTTP REST* (Watson, Watson *Text to Speech*, 2018). Cependant, celle-ci s'avère particulièrement intéressante pour permettre d'attribuer une voix à notre futur agent conversationnel.

Tone Analyzer

Le service *Tone Analyzer* est une *API* conçue par IBM Watson permettant d'analyser les émotions et les différents types de tonalité d'un utilisateur à travers un texte. *Tone Analyzer* est capable de prédire les sentiments actuels de l'utilisateur à l'aide d'une analyse linguistique. Néanmoins, celui-ci supporte uniquement deux langages, l'Anglais et le Français (Watson, Watson Tone Analyzer, 2017). En collaboration avec un agent conversationnel, le service nous permet de définir plusieurs directions à prendre pendant une conversation selon les émotions et tonalités perçues par le *chatbot*.

2.3.3. Intégration

Il est important de noter que la collection cloud IBM Watson est fournie directement avec tous les services associés. En outre, celle-ci offre aux développeurs plusieurs types d'intégration. IBM Watson supporte une multitude de langages de programmation ainsi que divers supports de développement. Le tableau suivant présente une liste sommaire des environnements de développement compatibles avec la solution IBM Watson Assistant ainsi que tous les services additionnels associés.

Tableau 1 - Environnements de développements compatibles avec IBM

Logiciels SDK Watson	Langage de programmation
Android SDK	Java
Java SDK	Java
Node.js SDK	Typescript
Python SDK	Python
.NET SDK	C#
Swift SDK	Swift
Unity SDK	C#

Source : (Watson, IBM Watson APIs, s.d.)

Comme nous pouvons le remarquer, la solution IBM Watson dispose d'un kit de développement dédié à *Unity*. Étant donné que la collection cloud intègre également tous les services tels que IBM Watson Assistant, *Speech to Text*, *Text to Speech* ainsi que *Tone Analyzer*, l'implémentation de l'agent conversationnel en réalité virtuelle est grandement facilitée.

2.3.4. Langues

IBM Watson Assistant prend en charge un total de treize langues différentes. Le tableau suivant présente certaines des langues supportées par IBM Watson Assistant.

Tableau 2 - Liste des langues entièrement prises en charge par IBM Watson Assistant Watson

Langue	Degré de prise en charge
Anglais (en)	Prise en charge complète
Français (fr)	Prise en charge complète
Allemand (de)	Prise en charge complète
Italien (it)	Prise en charge complète
Japonais (ja)	Prise en charge complète
Portugais (pt-br)	Prise en charge complète
Espagnol (es)	Prise en charge complète

Source : (Watson, Watson Assistant, 2018)

Dans le cadre de notre projet, il est important que l'agent conversationnel prenne en charge au moins le français et l'anglais. Comme nous pouvons le remarquer, IBM Watson Assistant supporte intégralement ces deux langues.

2.3.5. License

La solution IBM Watson fournie par IBM dispose de trois modèles économiques différents :

- IBM Watson Lite
- IBM Watson Standard
- IBM Watson Premium

Le modèle économique gratuit Lite propose par mois jusqu'à 10'000 appels à l'API de la solution, 5 espaces de travail distincts ainsi que 100 intentions et 25 entités maximum. Le modèle Standard offre quant à lui un nombre illimité d'appels, 20 espaces de travail distincts, une plateforme cloud partagée ainsi que 2000 intentions 1000 entités maximum. Finalement l'offre Premium comprend en plus des règles de sécurité plus élevées en ce qui concerne la protection des données sensibles (Watson, Watson Assistant, s.d.).

Il est important de noter que ces trois modèles sont également valables avec tous les autres services additionnels mentionnés précédemment.

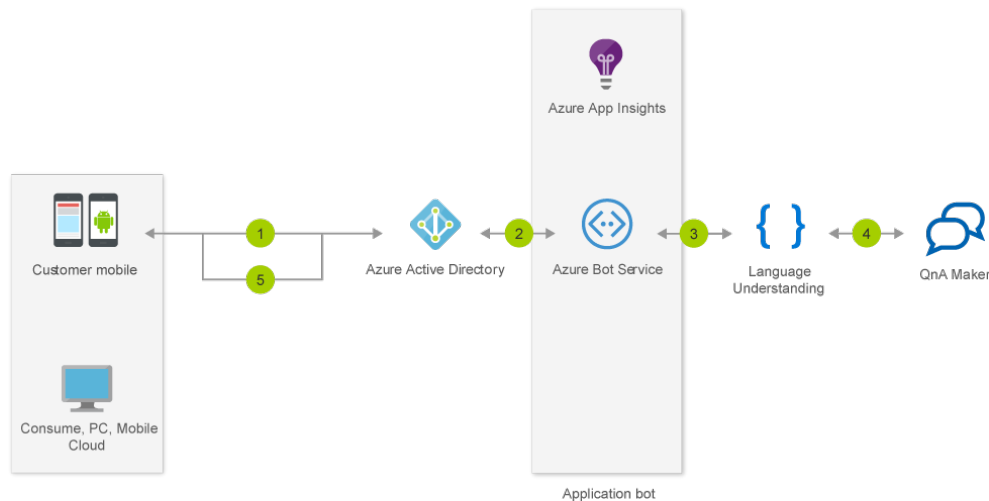
2.4. MICROSOFT BOT FRAMEWORK

Le *framework* Microsoft Bot Framework est une solution d'agent conversationnel inclut dans le service cloud Microsoft Azure utilisant la technologie LUIS. *Language Understanding* (LUIS) est un service cloud conçu par Microsoft capable grâce à une analyse linguistique utilisant le *machine learning* de prédire le sens global d'un message ainsi que d'en extraire les informations pertinentes (Microsoft, What is Language Understanding (LUIS)?, 2017).

2.4.1. Fonctionnement

Le service Microsoft Bot Framework propose différents types d'architecture lors de la création d'un robot intelligent. En effet, celui-ci supporte plusieurs types principaux de chatbots tels que *Commerce chatbot*, *Entreprise Productivity chatbot* ainsi qu'*Information chatbot* (Microsoft, Bot scenarios, 2017). Ce dernier est destiné à la création d'un agent conversationnel capable d'interpréter les messages provenant d'un utilisateur et de lui répondre en conséquence. Le diagramme suivant présente le fonctionnement général de l'agent conversationnel intégré dans une solution applicative.

Figure 5 - Architecture d'une solution utilisant Microsoft Bot Framework



Source : (Microsoft, Azure Bot Service, s.d.)

À l'instar de IBM Watson Assistant, le service Azure Bot Service joue le rôle de *middleware* dans l'architecture d'une solution informatique. En effet, les utilisateurs sont menés à interagir directement avec l'interface graphique de l'application dans laquelle la solution a été implémentée.

Dans un premier temps, l'identité de l'utilisateur est validée et authentifiée par l'annuaire du service Azure Active Directory. La requête courante est ensuite transférée directement au bot *framework*. Dans le cadre de notre travail, celle-ci sera analysée par la plateforme de Microsoft,

Language Understanding (LUIS). En outre, ce service permet via des modèles de langage naturel de déterminer l'objectif de l'utilisateur. Malgré les variations et les formes linguistiques, celui-ci est capable de comprendre le sens du message du client (Microsoft, Bot Service templates, 2017).

Tout comme IBM Watson Assistant, la plateforme LUIS fait appel à différents modèles de conception lors d'une analyse linguistique d'un message tels que les Énoncés (*Utterances*), les Rôles, les Entités (*Entities*) ainsi que les Intentions (*Intents*). Un énoncé représente simplement un message ou une entrée d'un utilisateur. L'analyse LUIS va essayer d'extraire des intentions ainsi que des entités depuis ces énoncés. Une intention détermine l'objectif ou la tâche que l'utilisateur veut atteindre dans son message. Une entité permet quant à elle de désigner un contexte à une intention. Au contraire de IBM Watson Assistant, LUIS subdivise les entités en sous-catégories représentées par des rôles. Cette structure permet de détailler un contexte afin de lui donner un sens plus précis (Microsoft, Entities in LUIS, 2018).

2.4.2. Services supplémentaires

Il est intéressant de noter que Microsoft propose également plusieurs services additionnels en plus du chabot réunis dans leur solution Cognitive Services. Celle-ci permet aux développeurs d'améliorer l'intelligence artificielle avec une liste de puissants algorithmes par le biais de plusieurs *APIs* (Microsoft, Add intelligence to bots with Cognitive Services, 2017).

Entity Linking Intelligence Service

Le service *Entity Linking Intelligence Service* permet à travers son API la recherche des entités mentionnées dans un texte non structuré. De plus, celle-ci est capable de comprendre le contexte d'une entité. Ainsi, deux mots ou phrases identiques n'ont pas forcément la même signification ou ne se réfèrent pas à la même chose selon le contexte identifié (Microsoft, Add intelligence to bots with Cognitive Services, 2017).

QnA Maker

Le service *QnA Maker* est une *API REST* permettant simplement d'entraîner une intelligence artificielle à répondre aux questions d'un utilisateur d'une manière naturelle et cohérente (Microsoft, Add intelligence to bots with Cognitive Services, 2017).

Bing Speech API

Le service *Bing Speech API* est une *API* de reconnaissance vocale supportant au total huit langues différentes. Le système de reconnaissance vocale permet entre autres de convertir un enregistrement vocal en texte. À l'instar du service *Speech to text* fourni par IBM Watson, *Bing Speech* se trouve être particulièrement important dans le contexte de notre projet afin de permettre la discussion orale avec l'agent conversationnel. Cependant, ce dernier supporte également la synthèse vocale à partir d'un texte. Au contraire de IBM Watson, il n'est donc pas nécessaire de faire appel à un autre service externe (Microsoft, Reconnaissance vocale Bing, s.d.).

Bing Spell Check API

En plus de *Language understanding* (LUIS), Microsoft propose également le service additionnel *Bing Spell Check API*. En outre, *Bing Spell Check API* offre de puissantes fonctions de vérification orthographique et est capable de reconnaître la différence entre les noms, les noms spécifiques à une marque ainsi que l'argot (Microsoft, Add intelligence to bots with Cognitive Services, 2017).

2.4.3. Intégration

En parcourant la documentation technique de Microsoft Bot Framework, nous pouvons rapidement remarquer que son intégration dans une solution s'avère plus complexe que prévu. En effet, ce dernier propose aux développeurs très peu de kits de développement. L'agent conversationnel permet uniquement une intégration dans les langages .NET et Node.js (Microsoft, Microsoft Repositories, s.d.).

L'implémentation de l'agent conversationnel en réalité virtuelle demeure cependant tout à fait possible. Le logiciel de création *Unity* est effectivement compatible avec .NET et C#. Néanmoins, il est tout même important de noter que Microsoft ne propose en aucun cas un kit de développement destiné à *Unity*. Ainsi, l'intégration du *chatbot* demandera un temps plus conséquent.

2.4.4. Langues

Comme nous l'avons déjà mentionné au préalable, Microsoft Bot Framework joue principalement le rôle de *middleware* dans une solution applicative. C'est effectivement le service LUIS qui prend en charge l'analyse linguistique. Le tableau récapitulatif suivant présente certaines des langues supportées par la plateforme *Language understanding* (LUIS).

Tableau 3 - Liste des langues avec leur degré de prise en charge par LUIS

Langue	Degré de prise en charge
Anglais (en)	Prise en charge complète
Français (fr)	Ne prends pas en charge les domaines prédéfinis de LUIS
Allemand (de)	Ne prends pas en charge les domaines prédéfinis de LUIS
Italien (it)	Ne prends pas en charge les domaines prédéfinis de LUIS
Japonais (ja)	Ne prends pas en charge les domaines prédéfinis de LUIS ainsi que l'analyse linguistique
Portugais (pt-br)	Ne prends pas en charge les domaines prédéfinis de LUIS ainsi que l'analyse linguistique
Espagnol (es)	Ne prends pas en charge les domaines prédéfinis de LUIS

Source: (Microsoft, Culture-specific understanding in LUIS apps, 2017)

À l'aide de ce tableau, nous pouvons donc remarquer que l'anglais est la seule langue qui possède une prise en charge complète par le service Microsoft Bot Framework.

2.4.5. License

Nous allons dans cette section analyser les différents modèles économiques des services nécessaires à la création d'un agent conversationnel. En effet, il est important de remarquer que chacun des services proposés par Microsoft possède un modèle indépendant. C'est la raison pour laquelle, nous allons uniquement nous concentrer sur les solutions suivantes :

- Language Understanding Service (LUIS)
- Services Speech (Bing Speech API)
- Azure Bot Service (Microsoft Bot Framework)

Le tableau suivant présente les détails de tarification des trois services. Nous avons volontairement omis toutes les options additionnelles analysées précédemment afin de nous concentrer uniquement sur les solutions principales.

Tableau 4 - Détails de tarification de LUIS, Bing Speech API et Microsoft Bot Framework

Service	Modèle gratuit	Modèle payant
Language Understanding Service (LUIS)	10'000 transactions gratuites par mois (uniquement des entrées texte)	1,265 € toutes les 1 000 transactions (entrées texte) 4,639 € toutes les 1 000 transactions (entrées audio)
Services Speech (Bing Speech API)	5 heures par mois (reconnaissance vocale) 5 millions de caractères par mois (synthèse vocale)	0,422 € par heure (reconnaissance vocale) 1,687 € chaque million de caractères (synthèse vocale)
Azure Bot Service (Microsoft Bot Framework)	Messages illimités	0,422 € par 1000 messages

Tableau de l'auteur provenant de sources multiples

(Microsoft, Tarification de Cognitive Services - Reconnaissance vocale (LUIS), s.d.)
 (Microsoft, Tarification Cognitive Services - Services Speech, s.d.)
 (Microsoft, Tarification Azure Bot Service , s.d.)

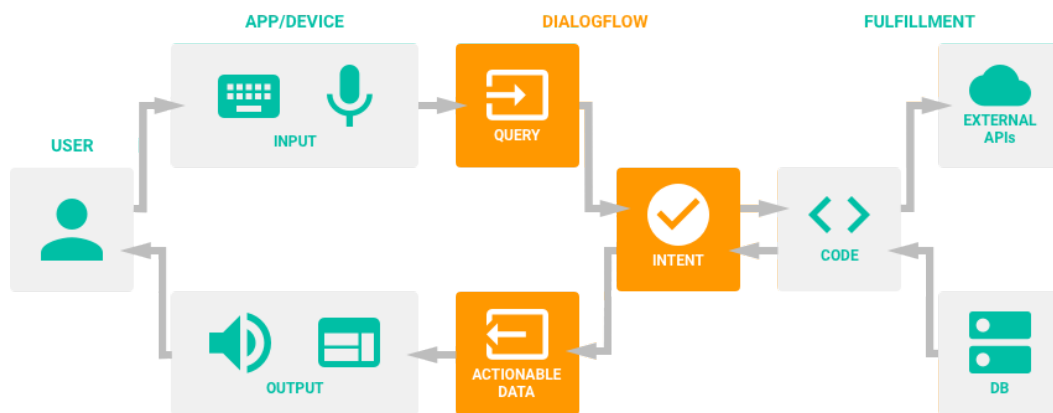
2.5. DIALOGFLOW

Anciennement API.AI, Dialogflow est un *chatbot* basé sur l'intelligence artificielle conçue par Speaktio et racheté par la suite par Google. En outre, l'assistant permet de construire des expériences conversationnelles riches et naturelles en offrant aux utilisateurs une toute nouvelle manière d'interagir avec une application (Google, Build natural and rich conversational experiences, s.d.).

2.5.1. Fonctionnement

Dialogflow intègre directement dans son architecture le savoir-faire de Google en matière de *machine learning*. En effet, ces algorithmes techniques permettent à l'agent conversationnel de comprendre le langage naturel des utilisateurs et surtout d'en extraire des données structurées. Le schéma suivant présente le fonctionnement basique de l'agent au sein d'une solution.

Figure 6 - Architecture d'une solution utilisant Dialogflow



Source : (Google, Dialogflow Agents)

Dans un premier temps, l'utilisateur active le processus conversationnel lors de sa première interaction avec l'interface de l'application. Dialogflow permet au client deux types d'action initiale. En effet, celui-ci détient la possibilité d'envoyer un message textuel ainsi qu'un enregistrement vocal. Les entrées provenant de l'application sont transmises directement à un agent créé au préalable sur la plateforme d'administration de Dialogflow.

Les agents sont responsables de gérer et diriger le flux d'une conversation avec un utilisateur d'une manière spécifique. Ainsi, il est tout à fait possible de créer une multitude d'agents différents afin de permettre plusieurs types de discussions selon plusieurs composants tels que des intentions, des entités, des contextes, des dialogues ainsi des accomplissements (Google, Dialogflow Agents).

Une intention (*intent*) est un composant de Dialogflow capable de déterminer les mesures et les actions à prendre par l'agent conversationnel selon la requête initiale de l'utilisateur. En somme, une intention représente simplement l'objectif qu'un client cherche à atteindre dans son message (Google, Dialogflow Intents).

Les entités quant à elles permettent de définir les données importantes à extraire des requêtes réalisées par les utilisateurs. Celles-ci peuvent être configurées pour un agent en particulier. Il n'est pas nécessaire aux développeurs de créer une entité pour chaque concept important. En effet, leur création est uniquement importante lorsqu'elles sont liées à une action définie dans l'agent. Une entité peut être de trois types différents. Les entités système sont préconfigurées par Dialogflow. Celles-ci regroupent les données communes les plus populaires. Les entités de type développeur représentent les données créées par les développeurs. Finalement, les entités utilisateur sont des données temporaires liées à la session courante de l'utilisateur qui rassemblent les concepts spécifiques au client (Google, Dialogflow Entities).

Dialogflow utilise également le concept d'action. Une action est simplement la direction que l'agent va prendre pendant une conversation lorsqu'une intention spécifique est détectée dans le message d'un utilisateur (Google, Dialogflow Actions and Parameters).

Le contexte est également un composant particulièrement important dans Dialogflow. En effet, le contexte permet d'ancrer la requête de l'utilisateur dans un contexte spécifique. Ainsi, il est plus facile de déterminer la réponse adéquate à une intention selon le sujet de la conversation actuelle (Google, Dialogflow Contexts).

Les dialogues définissent le cheminement de la conversation. L'interaction avec l'agent conversationnel peut être exécutée de manière linéaire en ayant une réponse pour chaque question de l'utilisateur. Cependant, la discussion peut être exécutée de manière non linéaire en définissant plusieurs embranchements possibles selon la réponse du client (Google, Dialogflow Dialogs).

Finalement, les accomplissements permettent aux développeurs d'accéder à une information de réponse via un service fourni par une *API* externe.

2.5.2. Intégration

En ce qui concerne l'intégration du *chatbot* Dialogflow, Google permet aux développeurs deux types d'intégrations différentes dans une solution. En effet, les librairies et les kits de développements de Dialogflow sont fournis en deux versions différentes, V1 et V2.

La deuxième version se divise en deux solutions d'intégration. D'une part, Google propose une intégration multiplateforme de leur service par le biais de l'*API REST* de Dialogflow. D'autre part, celle-ci permet également aux développeurs d'intégrer l'agent conversationnel à travers différentes librairies clientes. Le tableau ci-dessous présente la liste des librairies clientes V2 disponibles pour le moment.

Tableau 5 - Environnements de développements compatibles avec Dialogflow

SDK Dialogflow V2	Langage de programmation
Java	Java
Node.js	Typescript
Python	Python
Ruby	Ruby
PHP	PHP
C#	C#
Go	Go

Source : (Google, Dialogflow SDKs, s.d.)

Cependant, nous pouvons remarquer qu'il n'y a aucune différence entre les deux solutions. En effet, celles-ci permettent toutes les deux de transmettre les requêtes textuelles ou vocales des utilisateurs à un agent ainsi que de recevoir la réponse de ce dernier. De plus, les développeurs ont également la possibilité de changer dynamiquement le comportement de leur agent en ayant accès directement aux intentions, aux entités ainsi qu'aux contextes depuis l'environnement de développement (Google, Dialogflow SDKs, s.d.).

La première version regroupe les fonctionnalités initiales de l'agent API.AI avant le rachat par Google. Il est important de noter que Google recommande fortement de ne plus utiliser cette version de distribution lors de la création de nouveaux projets. La multinationale prévoit effectivement de ne plus supporter celle-ci à partir du 10 avril 2019. Néanmoins, nous pouvons remarquer que cette version propose plus de supports d'intégration que la deuxième version. Le tableau suivant regroupe la totalité des kits de développement V1 disponibles à l'heure actuelle (Google, Dialogflow SDKs, s.d.).

Tableau 6 - Environnements de développements compatibles avec Dialogflow V1

SDK Dialogflow V1	Langage de programmation
Android SDK	Java
Botkit SDK	Javascript
C++	C++
Cordova SDK	Objective-C
iOS SDK	Objective-C
Java SDK	Java
JavaScript SDK	JavaScript
.NET	C#
Node.JS SDK	Typescript
Python SDK	Python
Ruby SDK	Ruby
Unity SDK	C#
Xamarin SDK	C#

Source : (Google, Dialogflow SDKs, s.d.)

Malgré la mise en garde de Google, il est tout de même important de noter que la version *legacy* de Dialogflow propose une intégration directe au sein du logiciel *Unity* facilitant ainsi le développement du *chatbot* en réalité virtuelle.

2.5.3. Langues

Comme nous avons pu le voir précédemment dans cette analyse, l'agent de Dialogflow se trouve être l'élément principal du *chatbot*. En effet, il est responsable de gérer le flux de communication avec l'utilisateur.

Les agents prennent en charge un total de dix-sept langues différentes ainsi que plusieurs variantes dialectales (Google, Dialogflow Languages). Cependant, il n'y a aucune information concernant leur degré de prise en charge par l'agent conversationnel. Le tableau ci-dessous présente certaines des langues supportées par Dialogflow.

Tableau 7 - Liste des langues prises en charge par Dialogflow

Langue	Nombre de variantes
Chinois	3
Danois	0
Néerlandais	0
Anglais	5
Français	2
Allemand	0
Italien	0
Japonais	0
Espagnol	0
Suédois	0
Russe	0

Source : (Google, Dialogflow Languages)

2.5.4. License

La solution Dialogflow fournie par Google dispose uniquement de deux modèles économiques différents :

- Édition standard
- Édition entreprise

Les deux modèles recouvrent exactement les mêmes fonctionnalités principales. Néanmoins, l'édition entreprise offre un support technique plus prononcé pour les entreprises. Le tableau suivant présente les différences entre les deux modèles économiques.

Tableau 8 - Détails de tarification de Dialogflow

	Edition standard	Edition entreprise
Modèle	Gratuit	Payant
Requêtes textuelles	Illimité	0.002\$ par requête et illimité
Requêtes vocales	1'000 requêtes par jour et 15'000 par mois au maximum	0.0065\$ toutes les 15 secondes et illimité
Quota des requêtes textuelles	3 requêtes par seconde	10 requêtes par seconde
Support	Communauté et email	Google Cloud Support

Source: (Google, Dialogflow Pricing, s.d.)

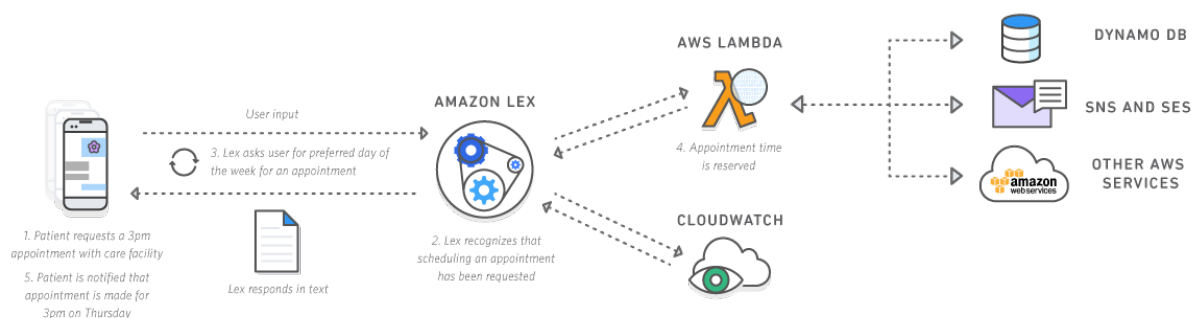
2.6. AMAZON LEX

Amazon Lex est une solution d'agent conversationnel utilisant le *machine learning*, conçue par Amazon.com. Le service hérite de la technologie et des fonctionnalités du célèbre assistant vocal Amazon Alexa afin de permettre aux développeurs de créer facilement et rapidement des robots de conversation (Amazon, Amazon Lex, s.d.).

2.6.1. Fonctionnement

Amazon Lex intègre directement dans son architecture la même technologie et le savoir-faire d'Amazon Alexa. Basé sur l'apprentissage profond, Lex est capable de comprendre le langage naturel des messages provenant d'utilisateurs. Cette analyse linguistique permet à l'agent conversationnel de déterminer l'objectif du client dans le but de lui répondre de manière cohérente et juste. Ainsi, Lex est en mesure de construire une discussion naturelle et humaine avec un utilisateur. Le schéma suivant détaille le fonctionnement basique d'Amazon Lex en tant que robot informatif (Amazon, Amazon Lex, s.d.).

Figure 7 - Fonctionnement d'Amazon Lex dans une solution informatique



Source: (Amazon, Amazon Lex, s.d.)

Comme nous pouvons le remarquer, les utilisateurs interagissent simplement avec Amazon Lex en transmettant une requête par le biais d'une simple interface chargée de distribuer le message à l'agent conversationnel. Il est important de noter qu'Amazon Lex est capable directement de recevoir une requête textuelle ou vocale. En effet, le service dispose de la reconnaissance vocale dans son système. Au contraire de certains *chatbots*, il n'est pas nécessaire de faire appel à un service tiers pour le traitement d'un enregistrement.

À l'aide du *machine learning* pour la compréhension du langage naturel, Amazon Lex exécute une analyse linguistique détaillée pour chaque entrée de l'utilisateur. En outre, cette analyse permet tout simplement à la machine de comprendre le langage parfois aléatoire du client afin de pouvoir exécuter une action en réponse. À l'instar des autres services d'agent conversationnels, une discussion avec client utilise plusieurs concepts importants tels que les Intentions (*Intents*), les Énoncés (*Utterances*), les Fentes (*Slots*) ainsi que les accomplissements (*Fulfillment*).

Les intentions représentent donc simplement les objectifs principaux à atteindre des utilisateurs. Une intention attend une action de l'agent conversationnel en réponse afin de satisfaire l'objectif du client. Cette action est représentée par le concept d'accomplissement. En outre, les énoncés représentent simplement les messages vocaux ou textuels des utilisateurs ayant une intention commune. Les fentes quant à elles représentent les données nécessaires à l'agent conversationnel pour accomplir et satisfaire l'intention.

2.6.2. Services supplémentaires

AWS Lambda

Amazon Lex supporte nativement l'intégration d'AWS Lambda. Ce service permet l'exécution de la logique métier de l'entreprise en réponse à certains événements provenant directement de l'agent conversationnel. AWS Lambda regroupe plusieurs services tels que DynamoDB permettant la sauvegarde de l'état de conversation avec un utilisateur ainsi que Amazon SNS permettant de notifier des utilisateurs (Amazon, Description détaillée d'Amazon Lex, s.d.).

Amazon Polly

Malgré l'intégration directe de la reconnaissance vocale dans le service Amazon Lex, ce dernier est dépourvu de synthèse vocale permettant de convertir un texte en un enregistrement audio naturelle. Amazon Polly est un service de synthèse vocale naturelle exploitant des technologies de *deep learning* conçu par Amazon. Polly prend en charge un total de vingt-cinq langues différentes ainsi que des tonalités dépendantes du sexe (Amazon, Amazon Polly, s.d.).

2.6.3. Intégration

Amazon Lex distribue aux développeurs plusieurs kits de développement logiciel adaptés aux différentes plateformes de programmation permettant une utilisation simplifiée de l'API. Le tableau suivant détaille brièvement les langages ainsi que les environnements de programmation supportés par Amazon Lex.

Tableau 9 - Environnements de développements compatibles avec Amazon Lex

SDK Amazon Lex	Langage de programmation
Android SDK	Java
iOS SDK	Objective-C
Java SDK	Java
JavaScript SDK	JavaScript
.NET	C#
Node.JS SDK	Typescript
Python SDK	Python
Ruby SDK	Ruby
PHP	PHP

Source : (Amazon, Ressources pour développeurs, s.d.)

2.6.4. Langues

Le service d'agent conversationnel Amazon Lex prend uniquement en charge pour l'instant l'anglais américain. Il n'est pour l'instant pas prévu pour Amazon d'étendre le nombre de langues supportées (Amazon, FAQ sur Amazon Lex, s.d.).

2.6.5. License

La solution Amazon Lex fournie par Amazon dispose uniquement d'un modèle économique payant et d'une phase d'essai gratuite. Bien que ces deux modèles recouvrent exactement les mêmes fonctionnalités principales, l'essai gratuit est limité à un an. Le tableau suivant présente les différences entre la phase d'essai et le modèle payant.

Tableau 10 - Détails de tarification d'Amazon Lex

	Essai	Édition Standard
Modèle	Gratuit	Payant
Durée	1 an	Illimité
Requêtes écrites	10'000 requêtes par mois maximum	0.00075 USD par requête et illimité
Requêtes orales	5000 requêtes par mois maximum	0.004 USD par requête et illimité

Source : (Amazon, Tarification d'Amazon Lex, s.d.)

2.7. WIT.AI

WIT.AI est une solution *chatbot* open source utilisant le *machine learning* conçue en partenariat avec Facebook. Par le biais de ce service, les développeurs ont la possibilité de créer leur propre application et dispositif intégrant une intelligence artificielle capable de communiquer avec un utilisateur.

A l'aide d'une analyse linguistique, WIT.AI est en mesure de comprendre le langage naturel des messages textuels et vocaux provenant des utilisateurs afin d'en extraire les données pertinentes comment les intentions, les entités et les lieux. L'agent conversationnel est à même de répondre aux clients et donc d'avoir une discussion cohérente et naturelle (Wit.ai, Natural Language for Developers, s.d.).

L'agent supporte différents cas particuliers d'utilisation tels que :

- Les robots de conversation
- Les applications mobiles
- La domotique
- Les différents appareils portatifs

2.7.1. Intégration

WIT.AI met à disposition aux développeurs plusieurs types d'intégration selon les environnements de développement. Le tableau ci-dessous détaille brièvement les langages ainsi que les plateformes de développement supportées par WIT.AI.

Tableau 11 - Environnements de développements compatibles avec WIT.AI

SDK WIT.AI	Langage de programmation
Android SDK	Java
iOS SDK	Objective-C
Node.JS SDK	Typescript
Python SDK	Python
Ruby SDK	Ruby

Source : (Wit.ai, Wit.ai Repositories, s.d.)

Cependant, il est important de noter que l'entreprise met également à disposition en libre accès une *API REST* afin de permettre l'utilisation de l'agent conversationnel dans n'importe quel environnement de développement (Wit.ai, Getting Started with Wit.ai).

2.7.2. Langues

WIT.AI prend en charge actuellement un total de 73 langues différentes. Le tableau suivant présente sommairement les langues principales supportées par l'agent conversationnel.

Tableau 12 - Langues principales prises en charge par WIT.AI

Langues				
Arabe	Anglais	Hongrois	Maori	Serbe
Chinois	Finnois	Islandais	Norvégien	Espagnol
Croate	Français	Indonésien	Polonais	Suédois
Tchèque	Allemand	Italien	Portugais	Ukrainien
Danois	Grec	Japonais	Roumain	Slovaque
Néerlandais	Hindi	Coréen	Russe	Estonien

Source : (Wit.ai, FAQ, s.d.)

2.7.3. License

WIT.AI est une solution open source d'agent conversationnel entièrement gratuite et libre de droits même pour un usage commercial (Wit.ai, FAQ, s.d.).

2.8. PANDORABOTS

Pandorabots est une solution permettant la création de *chatbots* intelligents à l'aide de l'intelligence artificielle conversationnelle. Ce service supporte nativement directement les requêtes textuelles et vocales. Au contraire de ses nombreux concurrents, Pandorabots n'est pas en mesure de comprendre le langage naturel en utilisant le *machine learning*.

En effet, l'agent conversationnel utilise uniquement le langage de balisage AIML basé principalement sur le langage XML (Pandorabots, About Pandorabots). En outre, le standard AIML permet aux développeurs de créer leur propre intelligence artificielle. Néanmoins, cette dernière n'est pas à même de s'améliorer au fil du temps, car celle-ci n'est pas complétée par un apprentissage profond.

2.8.1. Intégration

Pandorabots prend en charge plusieurs types d'intégration selon les environnements de développement. La plateforme propose aux développeurs l'accès à leur agent conversationnel à travers plusieurs kits de développement. Le tableau suivant détaille les langages ainsi que les plateformes de développement supportées par Pandorabots.

Tableau 13 - Environnements de développements compatibles avec Pandorabots

SDK Pandorabots	Langage de programmation
Java SDK	Java
Node.JS SDK	Typescript
Python SDK	Python

Source: (Pandorabots, Software Development Kits)

Nous pouvons également remarquer que des SDKs développés par la communauté de Pandorabots sont également disponibles. Le tableau ci-dessous énumère les différents kits de développement non officiels compatibles avec l'agent Pandorabots.

Tableau 14 - Environnements de développements non officiels compatibles avec Pandorabots

SDK Pandorabots	Langage de programmation
Ruby SDK	Ruby
PHP SDK	PHP
Go	Go

Source: (Pandorabots, Software Development Kits)

2.8.2. Langues

Pandorabots affirme dans leur documentation technique que leur agent conversationnel est capable de prendre en charge toutes les langues sans aucune limite. Ainsi, l'entreprise ne fournit pas de liste des langues supportées.

2.8.3. License

La solution Pandorabots propose à ses clients une tarification d'utilisation sur mesure. En effet, celle-ci dispose de trois modèles économiques différents :

- Gratuit
- Premium
- Entreprise

Le tableau suivant présente les détails de tarification des trois modèles disponibles.

Tableau 15 - Détails de tarification de Pandorabots

	Gratuit	Premium	Entreprise
Modèle	Gratuit	Payant	Payant
Nombre de bots	2 maximum	10 maximum	illimité
Nombre de requêtes	1'000 requêtes gratuites par mois maximum	0.0025 USD par requête et 100'000 requêtes maximums par mois	N/A
Support	Support communautaire	Support email	Support complet et service après-vente

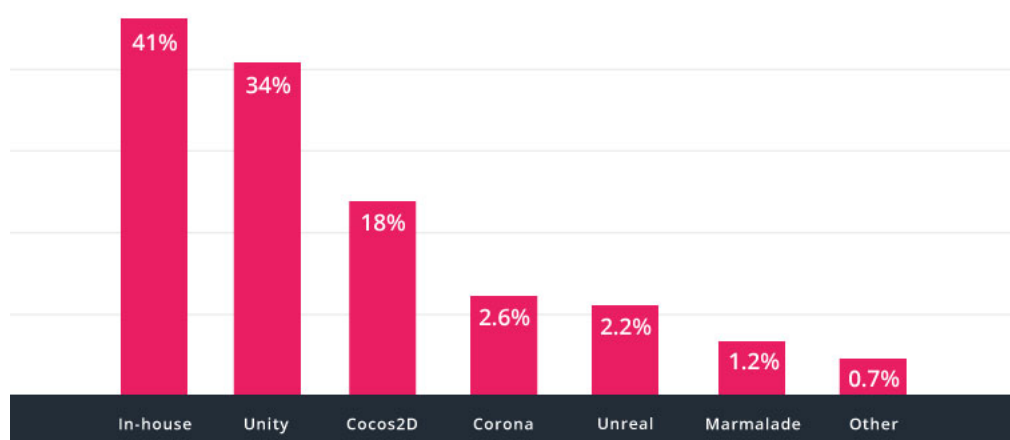
Source: (Pandorabots, Home, s.d.)

3. CHOIX TECHNOLOGIQUE

Dans cette section, nous allons effectuer une analyse comparative entre les différentes plateformes en récapitulant tous les résultats obtenus dans les chapitres précédents. En effet, il est important pour ce travail de choisir l'agent conversationnel le plus adéquat à l'ensemble du projet, à l'environnement de développement, mais également aux besoins de l'application que nous allons développer. C'est la raison pour laquelle il nous est nécessaire de résumer les différentes contraintes liées à ce travail.

Dans un premier temps, l'environnement de développement se trouve être la contrainte principale de l'application. L'application en réalité virtuelle sera effectivement développée à l'aide du célèbre moteur graphique multiplateforme *Unity*. Le logiciel de création *Unity* permet aux développeurs la création de jeux en 2D, 3D ainsi qu'en réalité virtuelle et réalité augmentée. Celui-ci est utilisé mondialement par les grands éditeurs, les studios indépendants, ainsi que les simples amateurs (Unity, Public Relations, s.d.). L'histogramme suivant présente le pourcentage de jeux mobiles gratuits conçus avec *Unity*.

Figure 8 - Moteurs graphiques utilisés pour le développement de jeux mobiles



Source : (Unity, Public Relations, s.d.)

Le schéma ci-dessus nous permet donc de constater la popularité conséquente du logiciel *Unity* pour le développement d'applications mobiles. En effet, nous pouvons remarquer que 34% des jeux mobiles sont conçus à l'aide de la plateforme contre 41% des jeux sont développés à l'aide d'un moteur graphique propriétaire. Le potentiel d'*Unity* n'étant plus à prouver, il est néanmoins nécessaire de relever que ce dernier fait appel à des scripts en C# en ce qui concerne le développement. Nous avons pu remarquer en analysant les différents *chatbots* que ceux-ci disposaient de plusieurs moyens d'intégration. Il est donc absolument nécessaire que le langage de programmation C# soit compatible avec l'agent conversationnel qui sera utilisé dans le cadre de ce projet. C'est la raison pour laquelle, ce critère d'évaluation présentera une pondération plus importante dans l'analyse.

Dans un deuxième temps, le nombre de langues prises en charge par l'agent conversationnel ne représente pas forcément un critère majeur. En effet, ce travail a pour objectif d'être réutilisé par l'Institut d'Informatique de Gestion de la HES-SO Valais-Wallis lors des développements futurs de projets faisant appel à un chatbot en réalité virtuelle. Ainsi, l'évaluation de ce critère ne reposera pas sur le nombre de langues, mais simplement sur le taux de prise en charge du français et de l'anglais.

En ce qui concerne le modèle économique des différentes solutions, la pondération de ce critère reposera sur la gratuité du modèle. Comme nous avons déjà pu le mentionner au préalable, nous ne recevons pas d'aide financière de la HES-SO Valais-Wallis pour la réalisation de ce projet. De plus, ce travail étant destiné à être réutilisé, il est impératif d'utiliser une solution disposant d'un modèle gratuit sans frais additionnels. La pondération sera donc plus importante sur la possibilité d'une utilisation libre du produit.

Afin de pousser l'immersion en réalité virtuelle, la prise en charge d'un service permettant la retranscription d'un enregistrement vocal en texte représente également un point particulièrement important. Selon notre analyse de besoins réalisée auparavant, la reconnaissance vocale pour communiquer avec l'agent conversationnel se retrouve donc au cœur du travail. Ainsi, il est de toute évidence préférable que la solution choisie supporte nativement celle-ci ou à travers un service supplémentaire. Dans le cas contraire, il nous sera donc nécessaire de faire appel à des plateformes tierces permettant la reconnaissance vocale ce qui prolongera la durée de développement. Dans le même ordre d'idée, il nous est également nécessaire que la solution supporte naturellement ou à travers un service tiers du même écosystème, un système de synthèse vocale. Afin de personnifier l'agent conversationnel, ce dernier doit être en mesure de répondre de manière orale. C'est la raison pour laquelle, ces deux critères disposent d'une pondération élevée.

À la suite de la réalisation de l'état de l'art, nous avons également réalisé que la qualité de la documentation technique se doit d'être présente dans l'analyse comparative des différentes solutions. En effet, celle-ci est essentielle pour le bon déroulement du développement de l'application finale. Ainsi, nous avons décidé de rajouter ce critère dans notre matrice décisionnelle avec une pondération conséquente.

3.1. ANALYSE COMPARATIVE

Le tableau suivant regroupe les différentes solutions analysées dans le cadre de ce travail selon les critères mentionnés au préalable. Une pondération d'un à deux a été attribuée pour chaque critère d'évaluation. Le nombre de points pour chaque évaluation est compris entre zéro et deux. Afin de mieux comprendre l'attribution des points aux solutions analysées, la liste suivante présente l'échelle des valeurs de notre matrice décisionnelle.

- 0 : Le critère d'évaluation n'est pas atteint
- 1 : Le critère d'évaluation est partiellement atteint
- 2 : Le critère d'évaluation est entièrement atteint

Tableau 16 : Matrice décisionnelle des agents conversationnels

	Intégration	Langues	Modèle	Reconnaissance	Synthèse	Documentation	Total
Pondération	2	1	1	2	2	1	
IBM Watson	2	2	1	2	2	2	17
Microsoft BOT	2	1	1	2	2	2	16
Dialogflow	2	1	1	0	0	2	8
AMAZON Lex	2	1	1	2	2	2	16
WIT.AI	0	1	2	0	0	1	4
Pandorabots	0	1	1	0	0	1	3

Source : Données de l'auteur

3.2. SYNTHÈSE

Lors de l'analyse des différentes plateformes disponibles, nous n'avons pas pris en compte la qualité globale des solutions. Il est effectivement difficile d'analyser celle-ci sans tester indépendamment chacun des agents conversationnels. Il nous est malheureusement impossible d'expérimenter toutes ces solutions de manière complète dans le temps imparti de ce travail. De plus, l'objectif de ce projet n'est pas simplement d'analyser et comparer les solutions existantes sur le marché mais bien d'implémenter une solution fonctionnelle fusionnant une application en réalité virtuelle et un *chatbot*. C'est la raison pour laquelle, la décision du choix technologique se concentrera uniquement sur le tableau ci-dessus.

A l'aide de notre matrice décisionnelle, nous pouvons constater que trois agents conversationnels précis se démarquent des autres solutions, IBM Watson, Microsoft BOT ainsi que AMAZON Lex. Ces deux derniers regroupent exactement les mêmes avantages sur la plupart de nos critères d'évaluation. Néanmoins, la solution d'IBM présente un avantage important en fournissant une prise en charge complète du français et de l'anglais ainsi qu'en offrant aux développeurs un kit de développement destiné intégralement à *Unity*.

En effet, celui-ci diminue grandement la durée de développement du travail en intégrant nativement toutes les fonctionnalités et tous les services d'IBM Watson. La librairie proposée par IBM permet effectivement aux développeurs d'utiliser IBM Watson Assistant mais également d'autres services tels que *Speech To Text* et *Text To Speech*. En effet, ces derniers sont fondamentaux dans le cadre de ce projet afin de permettre à l'utilisateur de communiquer directement avec l'agent conversationnel et pour que celui-ci soit en mesure de lui répondre de manière orale. C'est la raison

pour laquelle, nous avons donc choisi d'implémenter la plateforme proposée par IBM aux dépens de Microsoft BOT et AMAZON Lex.

4. CHOIX DES OUTILS ET TECHNOLOGIES DE DÉVELOPPEMENT

Le chapitre suivant a pour but de parcourir la liste des différents outils et technologies que nous avons utilisés lors du développement de notre agent conversationnel en réalité virtuelle. Pour chacun des éléments ci-dessous, nous expliquerons les raisons spécifiques de notre choix. Afin de ne pas noyer le lecteur sous un flot d'informations, nous n'analyserons pas en détail les solutions alternatives pour chaque outil.

4.1. SUPPORTS TECHNIQUES

4.1.1. HTC Vive

Le casque de réalité virtuelle HTC Vive est un produit phare né du partenariat entre le célèbre concepteur vidéoludique Valve et le fabricant HTC. Il est important de noter que Le Vive est le support le plus onéreux du marché. Celui-ci est en mesure de capturer les différents mouvements de l'utilisateur à 360° à l'aide de deux caméras externes et supporte un champ de vision de 110°. De plus, l'appareil offre également un très bon rendu graphique grâce à une résolution totale de 2160 x 1200 pixels ainsi qu'un taux de rafraîchissement maximum de 90 Hz par œil.

Néanmoins, le casque de HTC est être très similaire techniquement par rapport à ses principaux concurrents tels que l'Oculus Rift et le Playstation VR. Afin de ne pas s'étendre sur les différentes spécifications techniques des casques des concurrents, nous avons choisi de résumer brièvement les caractéristiques principales de ces derniers à l'aide du tableau suivant.

Tableau 17 - Tableaux comparatifs des principaux casques de réalité virtuelle

	HTC Vive	Oculus Rift	Playstation VR
Processeur graphique recommandé	NVIDIA GTX 1060 ou AMD Radeon RX 480	NVIDIA GTX 1060 ou AMD Radeon RX 480	N/A
Processeur recommandé	I5-4590 ou AMD FX 8350	I5-4590 ou AMD Ryzen 5 1500x	N/A
Mémoire	4 GO RAM	8 GO RAM	N/A
OS	Windows 7, 8.1, 10	Windows 10	PS4
Écran	AMOLED 3,6 pouces	OLED 5,6 pouces	OLED 5,7 pouces
Résolution	2160 x 1200 px	2160 x 1200 px	1920 x 1080 px

Taux de rafraîchissement	90 Hz	90 Hz	120 Hz, 90 Hz
Champ de vision	110°	110°	100°
Capteurs	Capteur G, Gyroscope et Accéléromètre	Gyroscope, Accéléromètre	Gyroscope, Accéléromètre
Microphone intégré	Oui	Oui	Oui
Poids	600 g	470 g	600 g
Prix	649.-	549.-	307.-

Tableau de l'auteur provenant de sources multiples

(Vive, s.d.)
 (Oculus, Rift, s.d.)
 (Playstation, s.d.)

Le choix du casque de destination de notre application n'a donc pas été influencé uniquement sur ses caractéristiques techniques. À l'exception du Playstation VR qui est destiné uniquement à la console Playstation 4, le casque de Valve est à quelques exceptions près identique techniquement à l'Oculus Rift. Étant donné que l'Institut Informatique de Gestion de la HES-SO Valais-Wallis dispose d'un HTC Vive, il nous est plus aisé de développer notre solution pour celui-ci.

4.1.2. Oculus GO

Comme mentionné précédemment, notre application était initialement destinée uniquement au célèbre casque HTC Vive. Cependant, à la fin du développement de notre solution et surtout à la demande de notre professeur assistant, nous avons décidé de considérer le casque mobile Oculus GO en tant que support supplémentaire. L'Oculus GO est un casque autonome de réalité virtuelle de la société américaine Oculus VR racheté par le réseau social Facebook.

Au contraire de son concurrent HTC Vive, celui-ci est totalement indépendant. En effet, ce périphérique mobile n'est pas contraint d'être branché directement à un ordinateur externe ou à un téléphone portable. Son fonctionnement et ses caractéristiques sont similaires à ceux des autres supports mobiles comme le *GearVR* et le *Google Cardboard*. À la différence de ces derniers, l'Oculus GO est une solution dédiée à la réalité virtuelle. La mobilité de celui-ci représente un atout majeur.

Il est tout de même important de remarquer que le rendu graphique de l'Oculus GO est cependant très inférieur à celui de l'HTC Vive en raison de sa portabilité. En définitive, le casque est donc une option adéquate pour les applications ne demandant pas beaucoup de ressources (Oculus, Oculus GO, s.d.).

4.2. KITS DE DÉVELOPPEMENT

4.2.1. Watson APIs Unity SDK

Watson APIs Unity SDK est le kit de développement proposé par IBM Watson destiné au moteur de jeu *Unity*. Le kit rassemble au même endroit tous les scripts permettant de communiquer directement avec l'ensemble des services compris dans le service IBM Watson. Cette librairie a pour objectif de faciliter le développement des applications sur *Unity* qui intègre Watson en fournissant tous les outils nécessaires aux développeurs. De plus, celui-ci comprend également une multitude de scripts C# ainsi que des scènes préconstruites servant d'exemples d'implémentation. En définitive, le SDK réduit grandement le temps d'implémentation des différents services.

Cependant, il est important de noter que le kit de développement ne possède pas de documentation officielle. Ainsi, l'utilisation de celui-ci peut s'avérer particulièrement complexe à prendre en main. En somme, nous avons parfois été contraints de parcourir en détail certaines des classes implémentées par IBM afin de mieux comprendre certains mécanismes et processus utilisés (Watson, Unity SDK, s.d.).

4.2.2. SteamVR SDK

SteamVR SDK est le kit de développement conseillé par la célèbre entreprise vidéoludique Valve lors du développement d'applications de réalité virtuelle destinées aux supports de haut de gamme tels que l'Oculus Rift et l'HTC Vive. Ce module externe permet d'accélérer la construction d'expériences en réalité virtuelle en facilitant l'intégration des diverses fonctionnalités des casques à l'aide de différents scripts préconstruits pour les développeurs.

Celui-ci regroupe notamment des éléments préfabriqués permettant de suivre et d'intégrer facilement les mouvements de *tracking* récoltés par les casques ainsi que de contrôler l'utilisation des manettes fournies avec les casques (Software, s.d.).

4.2.3. Oculus Utilities SDK

Oculus Utilities SDK est le kit de développement conçu par l'entreprise Oculus VR et destiné aux développements d'application en réalité virtuelle sur le moteur de création de contenu *Unity*. Ce module rassemble tous les outils nécessaires au support de toute la gamme de produits Oculus comme l'Oculus Rift, l'Oculus GO ainsi que le GearVR. De plus, le kit intègre également des scripts préfabriqués permettant le contrôle avancé des différentes fonctionnalités des casques comme les fonctions de locomotion et d'interactions avec des objets en réalité virtuelle. À l'instar du SteamVR SDK, celui-ci accélère grandement le développement d'expériences VR (Oculus, Oculus Utilities for Unity).

4.2.4. VRTK SDK

VRTK SDK est un kit de développement amateur supportant une multitude de SDKs destiné à la réalité virtuelle dont notamment SteamVR ainsi qu'Oculus Utilities. Celui-ci utilise et regroupe tous les scripts fournis par ces utilitaires de développement au même endroit afin de faciliter l'intégration multiplateforme des casques VR. Il est particulièrement important de noter que cet utilitaire supporte également les appareils comme Ximmerse et Google Daydream. À l'instar des kits de développement énoncés précédemment, il rassemble différents scripts préfabriqués permettant aux développeurs de changer facilement de matériel de destination (VRTK).

4.3. SERVICES

4.3.1. IBM WATSON Conversation

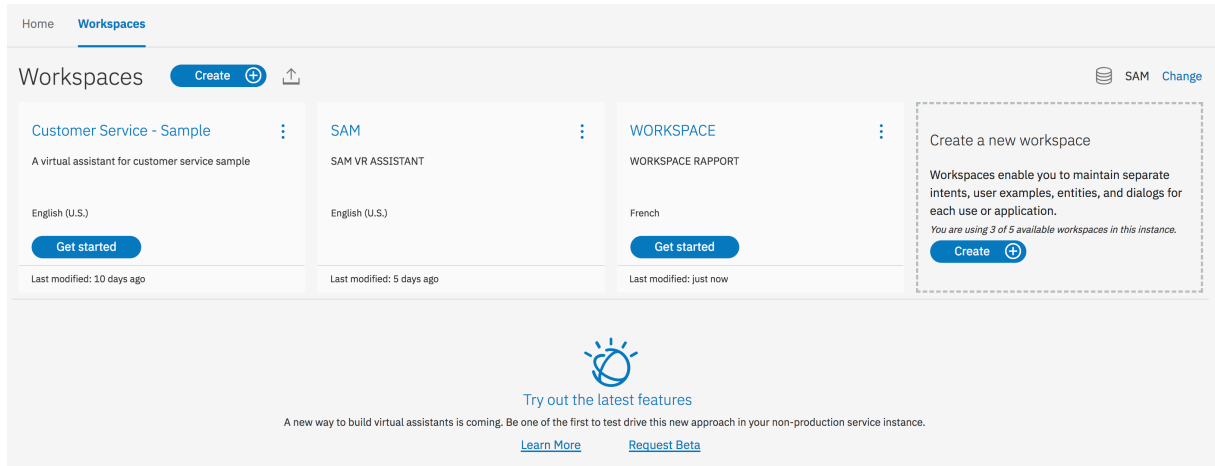
À la suite des résultats obtenus à l'aide de notre matrice décisionnelle, nous avons pu remarquer que le service IBM Watson Conversation développé par la multinationale IBM se trouve être la plateforme la plus adéquate à la réalisation de notre application.

À titre de rappel, IBM Watson Conversation anciennement Assistant est un service cloud de la solution IBM Watson permettant la création d'un agent conversationnel. IBM permet la configuration du service à travers une plateforme de gestion cloud. Celle-ci nous permet la gestion des espaces de travail, des différentes intentions et entités, des contextes ainsi que du flux de conversation de l'application.

Il est important de mettre en avant la qualité, la simplicité et l'ergonomie de la plateforme cloud de la solution IBM Watson. En effet, la multinationale est parvenue à réaliser un outil simple d'utilisation lors de la création d'un agent conversationnel. De plus, nous sommes contraints de constater que cette qualité d'exécution est valable pour tous leurs services cloud.

La figure suivante présente la plateforme de gestion des différents espaces de travail au sein de IBM Watson. Lors de la création d'un workspace, nous pouvons remarquer que celui-ci doit obligatoirement être lié à une langue spécifique. C'est la raison pour laquelle, un même espace de travail ne peut pas gérer plusieurs langages en même temps. Afin de gérer le multilinguisme, les développeurs sont donc dans l'obligation de dupliquer les données d'une langue à une autre. Cependant, IBM permet la possibilité d'exporter les données d'un espace de travail au format JSON et surtout d'importer ces mêmes données. Cette fonctionnalité s'avère particulièrement utile lors d'une migration d'une plateforme cloud à une autre. Un *workspace* contient effectivement toutes les intentions et entités ainsi que le flux de conversion configuré.

Figure 9 - Capture d'écran de notre plateforme de gestion des espaces de travail sur IBM Watson



Source : Données de l'auteur

La création des intentions sur la solution cloud d'IBM est également particulièrement simplifiée. En effet, l'objectif de l'utilisateur se traduit à l'aide d'exemples de messages. Plus le nombre d'exemples fourni à l'agent conversation est important, plus rapidement celui-ci détectera le but de l'utilisateur. La figure ci-dessous permet de visualiser la simplicité de la création d'une intention sur la plateforme cloud.

Figure 10 - Capture d'écran de la création d'une intention sur IBM Watson Conversation

<
#INTENTION_VOYAGER

Intent name
#INTENTION_VOYAGER

Description
Exemple d'intention

Add user examples
Add user examples to this intent

Add example

☐ **User examples (3)** ▼

☐ Bonjour, j'aimerais voyager !

☐ Je veux visiter la lune.

☐ Peux-tu me faire voyager ?

Source : Données de l'auteur

En ce qui concerne les entités, il est important de comprendre le concept qui se cache derrière ces artefacts. Une entité est uniquement utile et pertinente lorsque l'on cherche à détecter un mot précis dans une conversation entre un utilisateur et un agent. La plateforme d'administration d'IBM permet lors de la création d'une entité de fournir une liste de synonymes afin d'améliorer la détection. Dans la figure suivante, nous cherchons à travers notre agent à détecter le mot lune.

Figure 11- Capture d'écran de la création d'une entité sur IBM Watson Conversation

The screenshot shows the 'Entity name' field set to '@lune'. Below it, the 'Value name' field is empty with a placeholder 'Enter value' and an 'Add value' button. To the right, there is a 'Synonyms' section with a dropdown arrow and a text input field containing 'satellite de la terre, astre de la nuit' with an 'Add synonym...' button and a plus icon. At the bottom, a table lists the entity values:

Entity values (1) ▼	Type
<input type="checkbox"/> lune	Synonyms satellite de la terre, astre de la nuit

Source : Données de l'auteur

La construction d'une conversation entre les clients et l'agent se développe à l'aide de la création d'une chaîne de nœuds de dialogue. En faisant appel à des conditions sur les intentions et entités créées précédemment, il nous est alors particulièrement aisé de répondre à l'objectif visé par l'utilisateur. La figure suivante présente la configuration d'un dialogue simplifié pour un agent. Nous pouvons constater sur cette illustration que nous pouvons fournir une liste de réponses possibles à l'agent conversationnel afin de rendre la conversation plus naturelle.

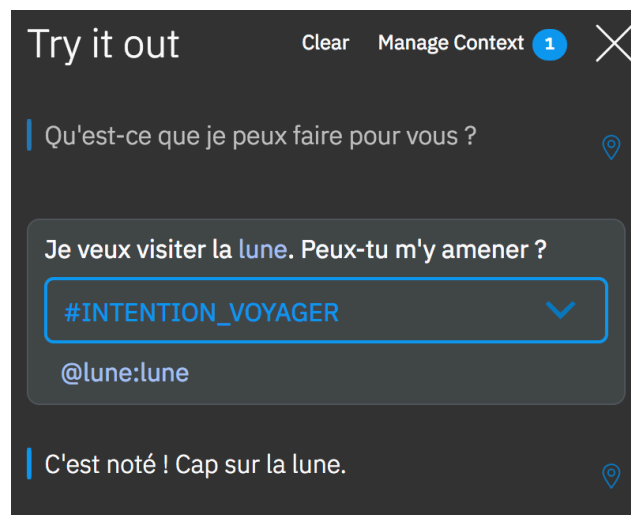
Figure 12 - Capture d'écran de la création d'un dialogue sur IBM Watson Conversation

The screenshot shows the 'Dialog' tab in the IBM Watson Conversation interface. On the left, the 'WORKSPACE' shows a node named 'VOYAGE SUR LA LUNE' with the condition '#INTENTION_VOYAGER and @lune' and a response '1. Response / 0 Context set / Does not return'. Below it is an 'ERREUR' node. On the right, the configuration for the 'VOYAGE SUR LA LUNE' node is shown. The condition is '#INTENTION_VOYAGER and @lune'. The response is configured with a 'Text' type and the following variations: 'C'est noté ! Cap sur la lune.', 'D'accord, Direction la lune !', and 'Enter response variation'. The response variations are set to 'sequential'.

Source : Données de l'auteur

Finalement, la plateforme d'administration cloud nous permet également de tester à tout moment notre solution et de manière totalement gratuite sans aucuns frais. L'image suivante illustre la simplicité de la vérification de notre exemple.

Figure 13 - Vérification de la configuration d'un agent sur IBM Watson Conversation



Source : Données de l'auteur

En définitive, nous pouvons constater que ces différents exemples de la simplicité d'utilisation du service cloud d'IBM permettent de justifier notre décision en ce qui concerne notre choix technologique.

4.3.2. IBM WATSON Speech to Text et Text to Speech

IBM Watson *Speech to Text* et *Text to Speech* sont des services cloud appartenant à la solution IBM Watson. À titre de rappel, le service *Speech to Text* utilise la reconnaissance vocale dans le but de retranscrire un fichier audio de manière textuelle. Ce service proposé par IBM utilise plusieurs modèles de reconnaissance linguistique propre à chaque langue.

Text to Speech quant à lui utilise la synthèse vocale dans le but de retranscrire un fichier audio de manière orale. À l'instar de son homologue, il utilise également plusieurs modèles linguistiques en intégrant différentes voix et divers accents lors de la synthèse.

Lors du développement de notre application, nous avons décidé d'utiliser ces deux services afin de permettre à l'utilisateur de parler oralement à l'agent conversationnel et dans le même ordre d'idée de synthétiser vocalement la réponse de l'agent. À la suite de la réalisation de l'état de l'art des technologies existantes, nous avons pu remarquer la présence de solutions alternatives à ces deux services comme Amazon Polly pour la synthèse vocale et Bing Speech API pour la reconnaissance vocale. Cependant, le kit de développement de IBM Watson destiné à *Unity* contient entre autres les

services *Conversation*, *Speech to Text* ainsi que *Text to Speech*. Afin d'utiliser uniquement des services dans un même écosystème, nous avons donc pris la décision d'écarter les solutions proposées par Amazon et Microsoft.

4.4. OUTILS ET ENVIRONNEMENTS DE DÉVELOPPEMENT

4.4.1. GitHub

GitHub est une plateforme propriétaire d'hébergements et de gestions des versions en ligne permettant une collaboration aisée entre plusieurs développeurs sur un même dépôt (*repository*). GitHub est en mesure de gérer l'historique des versions des codes sources à l'aide du système open source de contrôle de version Git (GitHub, s.d.).

La confidentialité du projet n'étant pas un critère particulièrement important, l'utilisation d'un dépôt public suffit amplement pour l'hébergement du code source de notre application. En effet, il est important de noter que la version gratuite de GitHub permet uniquement la création de dépôts publics. La solution n'est bien évidemment pas la seule sur le marché. Nous pensons notamment à ses concurrents tels que *Bitbucket*, *GitLab*, *SourceForge* ainsi que *GitKraken*. Chacune des plateformes présente des avantages conséquents en implémentant des outils d'intégration continue, de création de tableaux de bord SCRUM et de suivi des problèmes.

La liste des solutions alternatives et leurs avantages est longue. Cependant, nous sommes fidèles à GitHub depuis déjà plusieurs années. Tous nos dépôts se trouvent effectivement sur cette plateforme. Par souci de visibilité personnelle, nous avons décidé d'utiliser cette célèbre plateforme afin de détenir un contrôle complet sur l'historique des versions de notre application.

4.4.2. GitHub Desktop

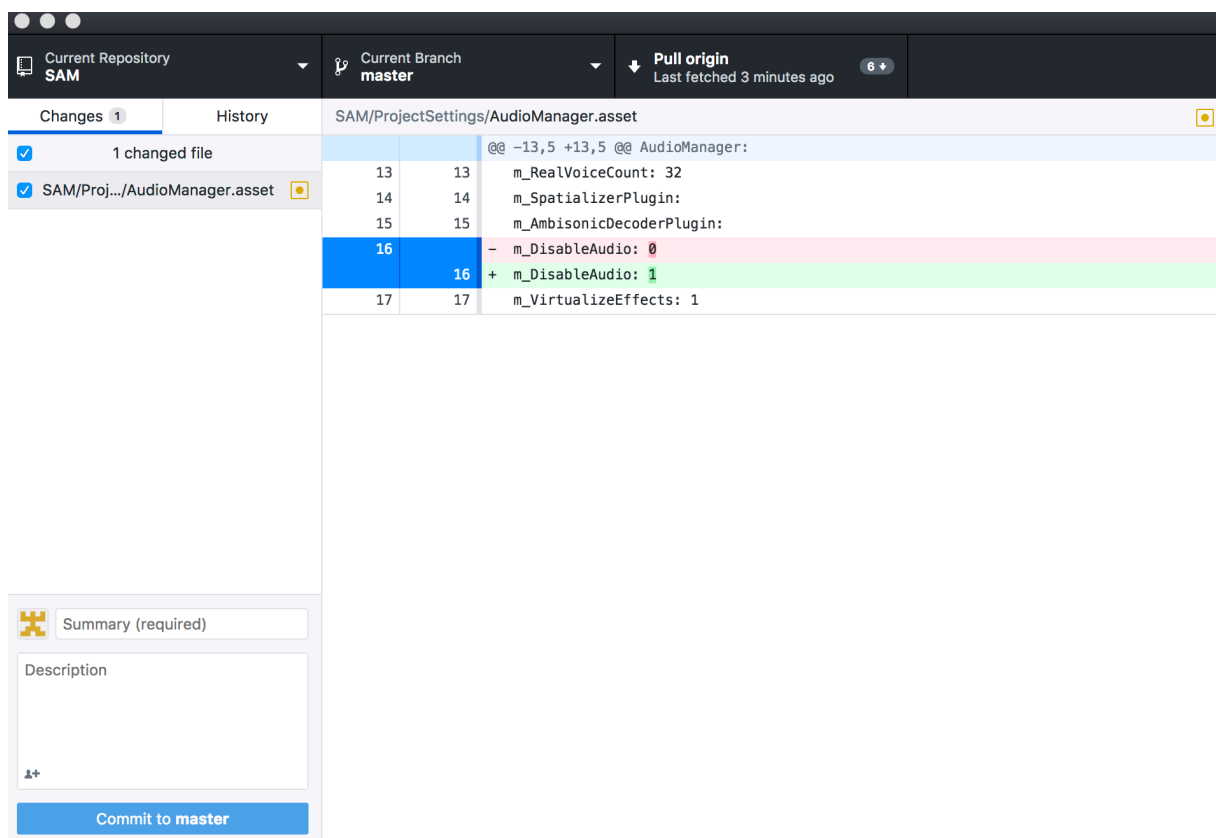
Afin d'utiliser les différentes fonctionnalités du système de contrôle de version git de GitHub, une multitude de solutions différentes s'offre à nous. Dans un premier temps, les commandes git comme clone, pull, push et commit sont accessibles directement en ligne de commande à condition que le système git soit installé sur la machine. Cependant, la syntaxe des différentes commandes peut s'avérer particulièrement complexe dans certains cas.

C'est la raison pour laquelle, des solutions utilisant une interface graphique sont également disponibles dans le but de faciliter la gestion des versions du code source d'un projet sur un dépôt distant. La plupart des environnements et outils de développement disposent nativement ou non d'une extension GitHub permettant aux développeurs d'exécuter les commandes directement depuis l'IDE. Le célèbre outil de développement *Unity* ne déroge pas à la règle. Il est effectivement possible d'installer l'extension à l'environnement à l'aide de la ressource (*asset*) GitHub for *Unity*. Néanmoins,

à la suite d'une multitude de tests réalisés en amont du développement, nous avons pu rapidement remarquer que celui-ci se trouvait être très instable.

En définitive, nous avons décidé de faire appel à la solution logicielle tierce GitHub Desktop. GitHub Desktop est le client logiciel de la plateforme d'hébergement et de gestion des versions GitHub. L'outil est une application multiplateforme disponible nativement sur les systèmes d'exploitation tels que Windows et macOS. Le logiciel tiers s'avère particulièrement utile lors du développement d'une application sur Unity3D. En effet, ce dernier permet aisément de créer un dépôt ou de cloner un dépôt distant depuis le client. De plus, celui-ci surveille en permanence les fichiers et dossiers de dépôt courant afin de mettre en évidence les nouvelles modifications.

Figure 14 - Interface du client GitHub Desktop Unity



Source : Données de l'auteur

Nous avons donc constaté que cette solution était la plus adéquate pour le développement de l'application d'une part grâce à sa stabilité, mais surtout, car celle-ci n'entraînait pas en conflit avec les composants du logiciel *Unity* au contraire des autres alternatives mentionnées ci-dessus.

4.4.3. Unity

Comme mentionné dans ce document, *Unity* est un célèbre moteur de jeu multiplateforme développé par Unity Technologies. Celui-ci est directement intégré à la solution logicielle du même nom. Cette plateforme permet facilement aux développeurs de créer du nouveau contenu ludique à la fois en 2D, 3D et VR. *Unity* possède à la fois une communauté importante ainsi que de nombreux tutoriels pour les amateurs (Unity, Unity3D, s.d.).

Cette spécificité, sa simplicité d'utilisation ainsi que son modèle entièrement gratuit expliquent son succès aux yeux des débutants face à son concurrent direct l'Unreal Engine qui est en mesure de délivrer exactement le même contenu. Malgré les points forts de la solution, notre choix entre les deux moteurs a été fortement influencé par l'éditeur de script de *Unity*. En effet, celui-ci est uniquement compatible avec le langage C# alors que Unreal Engine utilise le langage C++. Comme nous avons pu le voir en analysant les différentes solutions de *chatbot*, très peu d'agents conversationnels supportent actuellement le C++. C'est la raison pour laquelle, notre choix s'est donc évidemment porté sur *Unity*.

4.4.4. Visual Studio

Visual Studio est un environnement de développement propriétaire à présent multiplateforme, conçu par la multinationale Microsoft. Cet IDE supporte actuellement un total de six technologies différentes dont notamment les langages de programmation C++, Node.js, Python, R, .NET plus connu sous l'appellation C# ainsi que Javascript et TypeScript. Il est nécessaire de mentionner que cet environnement se trouve être l'outil par défaut du logiciel *Unity*. C'est donc la raison pour laquelle notre choix s'est tout simplement porté sur celui-ci (Microsoft, Visual Studio, s.d.).

5. DÉVELOPPEMENT

Le chapitre suivant a pour objectif de parcourir certains éléments du développement de l'application afin de mettre en avant le travail effectué en amont pour réussir à implémenter un agent conversationnel en réalité virtuelle. Néanmoins, il nous est nécessaire de préciser que cette section n'est en aucun cas une documentation technique de l'application.

En effet, cette dernière sera placée en annexe de ce document. C'est la raison pour laquelle, nous allons uniquement parcourir certaines étapes de l'implémentation. Ce chapitre a également pour but de relever certaines des difficultés rencontrées pendant le développement ainsi que d'apporter les différentes solutions que nous avons élaborées pour les résoudre.

5.1. DESCRIPTION DE L'APPLICATION

À titre de rappel, l'application développée dans le cadre de ce travail avait pour objectif principal de mettre en évidence la faisabilité de l'implémentation et l'intégration d'un agent conversationnel en réalité virtuelle. À la suite de plusieurs discussions avec notre professeur de suivi, nous avons décidé de pousser le concept de l'application plus loin. En effet, celle-ci se devait en plus de justifier l'intérêt d'intégrer un *chatbot* dans un univers virtuel.

Afin d'atteindre un intérêt ludique et éducatif, nous avons conceptualisé une solution permettant l'utilisateur de se plonger en réalité virtuelle dans l'espace, mais plus précisément sur la lune. En 1969, l'astre suscitait déjà l'intérêt des hommes. À l'aide de notre application, il lui est à présent possible de mettre les pieds sur celle-ci.

En faisant appel à une intelligence artificielle du nom de SAM, ce dernier est en mesure d'interagir et discuter avec elle, mais également d'interagir avec son environnement. En effet, l'utilisateur est capable de poser une multitude de questions sur les objets qui l'entourent par exemple la planète Terre, le drapeau américain planté en 1969 par l'astronaute Buzz Aldrin et même le véhicule lunaire utilisé pendant la mission d'Apollo 11.

Figure 15 - Premier aperçu de l'application en réalité virtuelle SAM

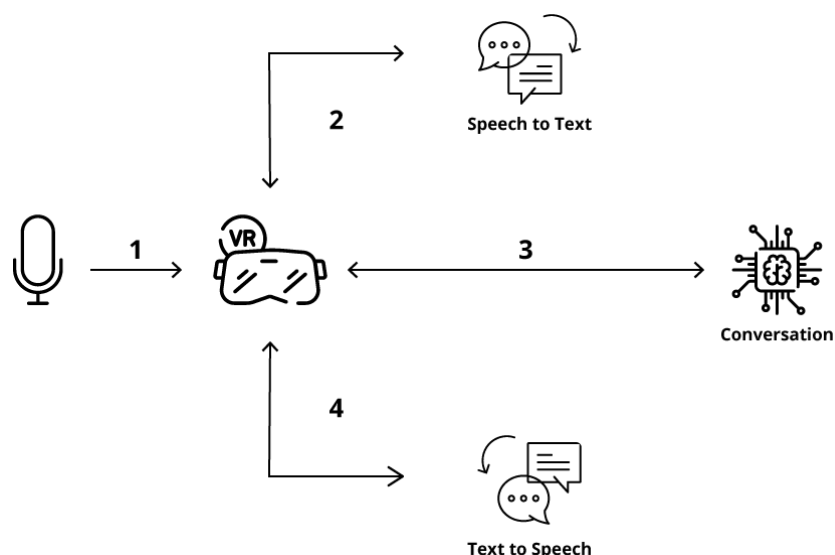


Source : Données de l'auteur

5.2. ARCHITECTURE DE L'APPLICATION

L'illustration suivante présente l'architecture de la solution que nous avons développée dans le cadre de ce travail.

Figure 16 - Architecture de l'application développée



Source : Données de l'auteur

Dans un premier temps, nous pouvons remarquer sur ce schéma que le fonctionnement général de l'application dépend principalement du microphone branché au casque de réalité virtuelle. En effet, à travers celui-ci, l'utilisateur est en mesure de parler et discuter avec l'agent conversationnel. En faisant appel à un enregistrement continu de l'utilisateur, l'application est capable de sauvegarder en mémoire le message vocal et de transmettre ce dernier au service *Watson Speech To Text*.

Dans un second temps, le service de reconnaissance vocale se charge dès la réception d'un fichier audio de retranscrire textuellement le message envoyé par l'utilisateur et de transmettre le résultat obtenu à l'application de réalité virtuelle. L'interprétation des informations reçues par le service est fortement influencée à la fois par la qualité du flux audio reçue mais également du modèle linguistique configuré par le développeur dans le service.

Le processus s'occupe ensuite de transmettre la transcription textuelle à l'agent conversationnel configuré à l'aide de la plateforme d'*IBM Watson Conversation*. Selon les conditions préétablies dans le flux de conversation, la réponse du *chatbot* est déterminée et renvoyée à la solution.

Finalement, le message de réponse textuel reçu par l'application est synthétisé vocalement en faisant appel au service *Text to Speech* de la bibliothèque cloud. À la réception du fichier audio, l'application est chargée de lire ce dernier et de le diffuser à l'utilisateur.

5.3. ARBORESCENCE DU PROJET

5.3.1. Arborescence

Dans cette section, nous allons détailler très brièvement l'arborescence générale du projet implémenté à la fois pour l'HTC Vive et l'Oculus GO. Le schéma suivant présente la structure du développement réalisé sur *Unity*.



Le dossier Scripts contient l'ensemble des fonctionnalités développées pour réaliser notre application. En effet, nous pouvons y trouver tous les fichiers C# chargés de communiquer avec les services d'IBM Watson.

Les dossiers Oculus, SteamVR et VRTK représentent les kits de développements qui nous ont permis d'avoir accès aux différents utilitaires et scripts des casques VR tels que l'Oculus GO et l'HTC Vive.

5.4. CONFIGURATION DE L'AGENT CONVERSATIONNEL

5.4.1. Artéfacts

À titre de rappel, dans la thèse « Dialogue homme-machine et apprentissage. Apprentissage par l'interaction », les deux auteurs relevaient l'importance des actions dans une conversation dans le but d'atteindre un objectif. Sur la base de ce principe, nous avons configuré les différents artéfacts de notre agent conversationnel. En effet, le développement des intentions pour notre *chatbot* a demandé une certaine réflexion en amont.

Dans un premier temps, la création des intentions a joué un rôle très important dans la réalisation de ce projet. C'est à l'aide de celles-ci que l'agent est capable d'apprendre, de comprendre le langage naturel dans une situation spécifique et surtout de répondre en conséquence. Dans le même ordre d'idée, les intentions permettent donc d'influencer l'intelligence artificielle de la machine. Nous avons donc défini au départ une liste non exhaustive d'objectifs que l'utilisateur pourrait chercher à atteindre à l'aide de l'application. La capture d'écran suivante présente une liste incomplète des intentions créées sur la plateforme cloud d'IBM Watson Conversation dans le cadre du développement de notre application.

Figure 17 - Liste partielle des intentions configurées pour IBM Watson Conversation

<input type="checkbox"/> #AGENT_GET_AGE	Request agent age	15 hours ago	7
<input type="checkbox"/> #AGENT_GET_BIRTHPLACE	Request agent birthplace	15 hours ago	9
<input type="checkbox"/> #AGENT_GET_CAPABILITIES	Request capabilities of the bot.	15 hours ago	31
<input type="checkbox"/> #AGENT_GET_GOAL	Request agent goal	15 hours ago	50
<input type="checkbox"/> #AGENT_GET_LOCATION	Request agent location	15 hours ago	21
<input type="checkbox"/> #AGENT_GET_NAME	Request generic personal information.	15 hours ago	15
<input type="checkbox"/> #AGENT_GET_ORIGINS	Request agent origins	15 hours ago	13
<input type="checkbox"/> #AGENT_GET_RELATIVES	Request agent relatives	15 hours ago	11
<input type="checkbox"/> #AGENT_GREETINGS	Greet the bot.	15 hours ago	25
<input type="checkbox"/> #AGENT_GREETINGS_ANSWERS	General answers after greetings from bot	15 hours ago	25
<input type="checkbox"/> #ENVIRONMENT_GET_AGE	Request environment age	15 hours ago	16
<input type="checkbox"/> #ENVIRONMENT_GET_INFORMATION	Request environment information	15 hours ago	26
<input type="checkbox"/> #ENVIRONMENT_GET_ORIGINS	Request environment origins	15 hours ago	24
<input type="checkbox"/> #ENVIRONMENT_GET_SIZE	Request environment size.	15 hours ago	29
<input type="checkbox"/> #ENVIRONMENT_MORE_INFORMATION	Request environment information	15 hours ago	24
<input type="checkbox"/> #GENERAL_ENDING	End the conversation.	15 hours ago	37
<input type="checkbox"/> #GENERAL_HUMAN_OR_BOT	Ask if speaking to a human or a bot.	15 hours ago	12
<input type="checkbox"/> #GENERAL_JOKES	Request a joke.	15 hours ago	18

Source : Données de l'auteur

Comme nous pouvons le remarquer dans cette illustration, nous avons cherché à subdiviser les objectifs de l'utilisateur en plusieurs catégories. Ainsi, il nous est possible de regrouper toutes les intentions de la manière suivante :

- les questions en rapport avec l'agent conversationnel
- les questions spécifiques à l'utilisateur
- les demandes d'information à propos de l'environnement
- les requêtes génériques régulièrement demandées par les utilisateurs
- les demandes permettant de gérer et de contrôler le système.

Néanmoins, il est important de noter qu'il nous aurait été également possible de regrouper les différentes catégories mentionnées ci-dessus en une seule intention. Cependant, cette méthode exige la détection de mots précis dans une phrase à l'aide des entités afin d'engendrer une réponse

adéquate du *chatbot*. Malgré la faisabilité de cette solution, celle-ci n'utilise pas le plein potentiel de l'agent conversationnel qui est capable simplement à l'aide d'une multitude d'exemples de comprendre le langage naturel de l'utilisateur.

Dans un second temps, nous avons été contraints de recueillir un nombre conséquent d'exemples de requêtes pour chaque intention. L'agent conversationnel est uniquement en mesure d'apprendre à l'aide de ces exemples. La difficulté de cet exercice est bien évidemment de rassembler toutes les formulations possibles pour poser la même question et donc atteindre un objectif identique. L'illustration suivante présente les différentes formulations potentielles pour questionner l'agent conversationnel sur un objet spécifique.

Figure 18 - Exemple des différentes formulations pour un objectif identique

The screenshot shows a configuration interface for a chatbot intent. At the top, the 'Intent name' is '#ENVIRONMENT_GET_INFORMATION' and the 'Description' is 'Request environment information'. Below this, there is a section for 'Add user examples' with a text input field containing 'Add user examples to this intent' and a blue 'Add example' button. A list of 26 user examples is displayed below, each with a checkbox and a small edit icon. The examples are:

- ☐ Give me information about it!
- ☐ Give me information about this object !
- ☐ Give me information on that.
- ☐ Give me information on this.
- ☐ Give me some information about it!
- ☐ Give me some information on that.
- ☐ What does it do?
- ☐ What good is that?
- ☐ What is this item?
- ☐ What is this object?
- ☐ What is this object for?

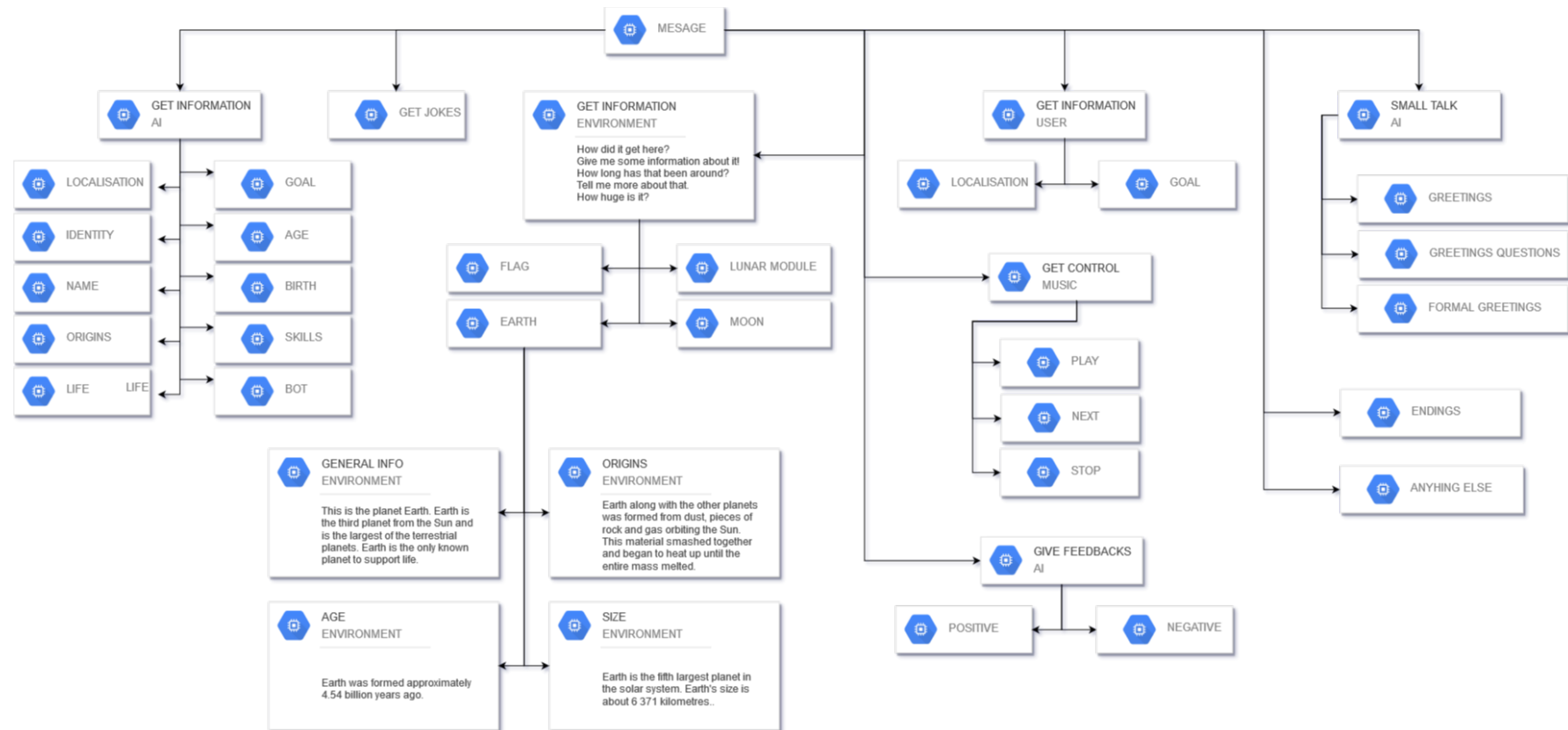
Source : Données de l'auteur

En procédant de cette manière lors de la configuration de notre agent conversationnel, ce dernier ne fait jamais appel aux entités. À titre de rappel, les entités représentent une liste de mots pertinents pour le *chatbot*. En reconnaissant une entité dans un message provenant de l'utilisateur, l'intelligence artificielle est en mesure de choisir une réponse spécifique pour satisfaire l'objectif du client. Néanmoins, nous pouvons constater que nous utilisons uniquement des exemples fortement génériques. Il ne nous est donc pas nécessaire de détecter des entités en particulier dans les messages transmis par le client.

5.4.2. Diagramme de conversation

Le diagramme suivant a pour objectif de représenter simplement tous les embranchements possibles ainsi que toutes les directions différentes d'une conversation entre un utilisateur et notre intelligence artificielle.

Figure 19 - Embranchements possibles lors d'une conversation avec un utilisateur



Source : Données de l'auteur

5.5. DIFFICULTÉS RENCONTRÉES

Cette section a pour objectif de rassembler les difficultés rencontrées majeures lors du développement de notre application. Ainsi, nous allons apporter la solution implémentée pour chaque problème apparu dans le cadre de travail. Par souci de lisibilité, nous n'utiliserons pas d'illustrations du code source qui a permis de résoudre les différentes situations.

5.5.1. Reconnaissance vocale pertinente

Comme nous l'avons déjà mentionné précédemment, l'application était initialement prévue de supporter au total deux langues différentes dont notamment le français et l'anglais. En ce qui concerne les trois services utilisés d'IBM, la documentation officielle affirme que ces derniers supportent au total sept langues différentes.

Lors du développement de l'application et de l'implémentation du service *Speech to Text*, nous avons malheureusement constaté que la reconnaissance vocale en français est fortement aléatoire et non adéquate pour le bon fonctionnement de la solution. En effet, le service est à l'heure actuelle incapable de comprendre parfaitement la langue de Molière et donc de retranscrire parfaitement les messages de l'utilisateur. Cette contrainte ne nous permettait donc pas de communiquer avec notre agent conversationnel qui est quant à lui en mesure d'interpréter la langue française.

Cependant, à la suite d'une multitude de tests, nous avons pu constater qu'il n'y avait pas de réelle solution pour résoudre le problème. Le modèle linguistique utilisé par la solution de reconnaissance vocale n'était pas suffisamment évolué. Étant donné que le service est susceptible de s'améliorer dans le futur, nous avons simplement pris la décision de supporter uniquement l'anglais pour le moment.

5.5.2. Latence d'enregistrement

Pendant le développement de la communication interne entre les trois services, nous avons constaté une latence importante entre la transmission initiale de la requête provenant de l'utilisateur et la synthèse de la réponse de l'agent conversationnel. À la suite de plusieurs tests de performance, nous avons malheureusement remarqué que la méthode d'enregistrement du flux audio implémentée par les développeurs d'IBM présentait plusieurs problèmes d'optimisation.

En effet, la fonction implémentée dans le kit de développement d'IBM Watson envoie par défaut une petite fraction du fichier audio sauvegardée en mémoire toutes les secondes. Il en résultait donc un délai entre l'envoi des données au service *Speech to Text* et la réception de la réponse textuelle de ce dernier. Par conséquent, l'utilisateur n'était pas en mesure de recevoir une réponse à sa requête dans les plus brefs délais. C'est la raison pour laquelle, l'expérience utilisateur en était fortement impactée.

Afin de résoudre le temps de latence, nous avons décidé de développer notre propre méthode d'enregistrement du flux audio. Nous avons effectivement opté pour un envoi continu des données provenant du microphone au service *Speech to Text* même si l'utilisateur est silencieux. Nous avons relevé que le service susmentionné ignore simplement les fichiers audio silencieux. En procédant de cette manière, l'enregistrement vocal de l'application n'attend pas que l'utilisateur parle dans le microphone pour envoyer les données du flux audio. C'est pourquoi la latence effective en est fortement réduite.

5.5.3. Interaction avec l'environnement

Afin de permettre à l'utilisateur d'interagir avec l'environnement qui l'entoure en posant des questions à l'agent conversationnel, nous devons trouver un moyen de transmettre à ce dernier le contexte de la conversation. À titre de rappel, un *chatbot* est en mesure de donner une réponse différente selon le contexte précis qui lui est transmis.

Ainsi, nous avons pour tâche de trouver une manière de détecter l'objet présent dans l'environnement dans le but d'être capable de transmettre cette donnée à l'agent. C'est la raison pour laquelle, nous avons décidé de faire appel à la fonctionnalité de *Raycast* propre à *Unity*. La classe *Raycast* permet simplement de lancer un faisceau invisible dans une direction donnée et de récupérer l'objet touché par celui-ci. En attachant cette fonctionnalité directement à la caméra de l'utilisateur, nous avons la possibilité de récupérer l'objet de l'environnement qui est actuellement visualisé. De cette manière, le contexte courant est constamment transmis à l'agent conversationnel.

5.5.4. Personnification

Lors de la synthèse vocale de l'agent conversationnel, nous avons constaté que celle-ci était fortement robotisée ainsi que monotone. Dans le but de doter notre agent conversationnel de caractéristiques plus humaines, nous avons fait appel au balisage expressif SSML du service *IBM Watson Text to Speech*.

Par défaut, la synthèse vocale de la solution utilise une déclaration monotone. C'est la raison pour laquelle le service propose aux développeurs trois types de balisage expressif :

- `<express-as type="GoodNews"></express-as>`
- `<express-as type="Apology"></express-as>`
- `<express-as type="Uncertainty"></express-as>`

Lorsque ces différentes balises entourent un message textuel, le timbre de la voix de l'agent conversationnel évolue et devient donc plus expressif. Dans le même ordre, la tonalité peut être déclarée de manière joviale, attristée ou incertaine.

De la même manière, le service permet également l'utilisation de la balise `<break>` qui permet de marquer des temps de pause lors de la déclaration du texte. De plus, la solution donne la possibilité de contrôler la durée des pauses à l'aide de l'attribut *strength* :

- `<break strength="none"></break>`
- `<break strength="x-weak"></break>`
- `<break strength="weak"></break>`
- `<break strength="medium"></break>`
- `<break strength="strong"></break>`
- `<break strength="x-strong"></break>`

En définitive, ces divers éléments de balisage syntaxique nous ont donné la possibilité de rendre la synthèse vocale plus naturelle et moins irrégulière.

5.5.5. Historique des versions

Lors de la mise en place d'un système de contrôle de version, nous avons pu constater que le logiciel *Unity* générait une multitude de fichiers de métadonnées ainsi que des fichiers propres à la configuration de l'éditeur. Ces derniers engendraient une multitude d'erreurs lorsque nous changions de machine de développement selon nos besoins. Afin de résoudre ce problème, nous avons été contraints d'utiliser un fichier `.gitignore` destiné spécialement au système d'environnement *d'Unity*.

Le fichier suivant est propre au système de contrôle de version Git. Celui-ci permet simplement d'ignorer une liste de fichiers, dossiers et extensions lors de l'utilisation de certaines commandes git. Le fichier `.gitignore` est particulièrement utile pour ne pas prendre en compte certains fichiers propres aux systèmes d'exploitation et aux outils de développements comme les fichiers cachés. L'application étant développée à l'aide du moteur de jeu *Unity*, le fichier ci-dessous permet de ne pas prendre en compte les fichiers de paramétrage ainsi que les métadonnées propres à l'outil.

Figure 20 - Fichier .gitignore destiné à l'outil *Unity*

```

1  [Ll]ibrary/
2  [Tt]emp/
3  [Oo]bj/
4  [Bb]uild/
5  [Bb]uilds/
6  Assets/AssetStoreTools*
7
8  # Visual Studio cache directory
9  .vs/
10
11 # Autogenerated VS/MD/Consulo solution and project files
12 ExportedObj/
13 .consulo/
14 *.csproj
15 *.unityproj
16 *.sln
17 *.suo
18 *.tmp
19 *.user
20 *.userprefs
21 *.pidb
22 *.booproj
23 *.svd
24 *.pdb
25 *.opendb
26
27 # Unity3D generated meta files
28 *.pidb.meta
29 *.pdb.meta
30
31 # Unity3D Generated File On Crash Reports
32 sysinfo.txt
33
34 # Builds
35 *.apk
36 *.unitypackage

```

Source : Données de l'auteur

5.5.6. Matériel, mise à jour et multiplateforme

Lors de la majeure partie du développement de l'application en réalité virtuelle, nous avons principalement travaillé sans aucun support technique. En effet, les technologies portables telles que les casques de réalité virtuelle sont particulièrement onéreux. Les rares casques à disposition au sein de la HES-SO Valais nécessitaient une demande de réservation pour une courte durée au préalable. C'est pour cela qu'il nous a été difficile de nous en procurer un sur une longue durée.

C'est pourquoi, nous avons donc décidé de développer l'application principalement sur l'éditeur *Unity* à l'aide du simulateur de jeu directement intégré au logiciel ainsi qu'un émulateur tiers destiné entre autres à l'HTC Vive fourni par la librairie VRTK. Lors des premiers tests de l'application sur l'HTC Vive, nous avons rencontré de nombreux problèmes. En effet, nous avons constaté que l'utilitaire VRTK n'avait plus été mis à jour depuis longtemps. Afin de rétablir le bon fonctionnement de la librairie, nous avons donc été contraints de parcourir tous les fichiers du kit de développement et modifier manuellement certaines lignes de code qui étaient obsolètes.

En fin de projet et à la suite d'une discussion avec notre professeur de suivi, ce dernier nous a suggéré de modifier l'application afin que celle-ci soit multiplateforme notamment avec l'Oculus GO. L'implémentation des différents utilitaires de l'Oculus GO nous a posé beaucoup de problèmes. À l'instar de la librairie VRTK, nous avons également été contraints de modifier certaines lignes de code du kit de développement à cause de mises à jour trop irrégulières.

De plus, nous avons également constaté que les différences techniques entre l'HTC Vive et l'Oculus sont beaucoup trop importantes. Étant donné que l'application avait été initialement pensée pour l'HTC Vive, toutes les ressources (*assets*) que nous avons utilisées étaient trop gourmandes graphiquement pour l'Oculus GO. Les manettes des deux supports sont également très différentes. L'Oculus GO n'utilise qu'une manette de jeu alors que l'HTC Vive en utilise deux.

Afin de nous permettre de tout même prendre en charge les deux casques, nous avons sacrifié le moyen de locomotion que nous avons implémenté pour l'HTC Vive à l'aide des deux manettes. Dans un deuxième temps, il nous a fallu modifier drastiquement tous les paramètres graphiques à la baisse dans le but de proposer une application multiplateforme.

5.6. APPLICATION

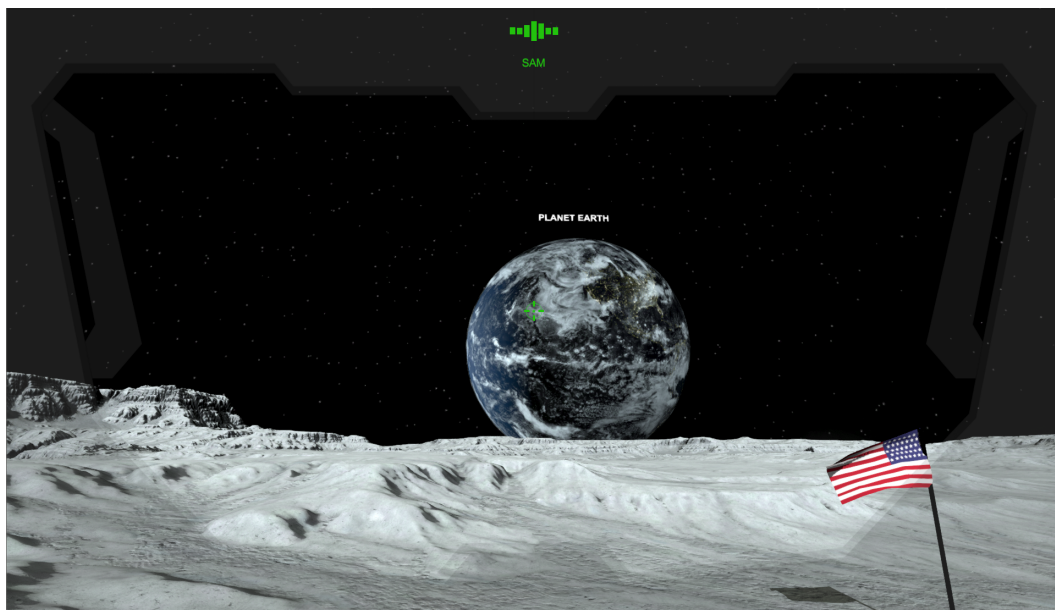
Ce chapitre liste les différentes fonctionnalités que nous avons développées dans notre application. Étant donné que cette dernière repose principalement sur l'intégration d'un *chatbot* en réalité virtuelle, il nous est particulièrement difficile d'illustrer visuellement celles-ci. C'est pourquoi nous sommes contraints d'énumérer de manière sommaire les diverses capacités de la solution proposée dans le cadre de ce travail.

5.6.1. Interaction avec l'environnement

Comme déjà mentionnés à plusieurs reprises dans ce document, nous offrons la possibilité à l'utilisateur d'interagir avec son environnement à travers l'intelligence artificielle. En effet, le client est donc en mesure de questionner l'agent sur les différents objets qui l'entourent sur la lune.

L'illustration suivante présente une interaction entre l'utilisateur et la planète Terre. Lorsque ce dernier demande à l'agent conversationnel des informations à propos de cette planète, l'intelligence artificielle est en mesure de lui fournir une multitude d'informations sur celle-ci dont notamment sa taille, sa création, son âge ainsi que diverses anecdotes. Il est important de noter que le *chatbot* est représenté physiquement par le module de communication situé au-dessus de l'écran.

Figure 21 - Interaction avec la planète Terre



Source : Données de l'auteur

Les deux images suivantes présentent quant à elles une interaction avec deux objets bien particuliers disposés sur la lune, le drapeau américain et le véhicule lunaire de la célèbre mission Apollo 11. Pour chacun des objets, SAM raconte une partie de l'histoire des deux astronautes historiques Neil Armstrong et Buzz Aldrin en s'appuyant sur des vraies photos.

Figure 22 - Interaction avec le drapeau américain de la mission Apollo 11



Source : Données de l'auteur

Figure 23 - Interaction avec le module lunaire de la mission Apollo 11



Source : Données de l'auteur

Le tableau ci-dessous regroupe les différentes interactions avec l'environnement présentes dans notre application.

Tableau 18 - Liste des interactions avec l'environnement

Questions	Terre	Lune	Drapeau	Module Lunaire
Âge	✓	✓	X	X
Origines	✓	✓	X	X
Taille	✓	✓	X	X
Informations	✓	✓	✓	✓
Anecdotes	✓	✓	✓	✓

Source : Données de l'auteur

Nous donnons également la possibilité au joueur d'écouter de la musique pendant l'expérience. En effet, celui-ci est en mesure d'avoir un contrôle total sur la bande-son de l'environnement à l'aide de notre intelligence artificielle. Il suffit simplement de demander à SAM de lancer, changer ou arrêter une musique.

5.6.2. Conversation générique et banalités

Lors de l'utilisation de notre application, l'utilisateur est en mesure de tenir une discussion avec l'intelligence artificielle implémentée. En somme, le client détient la possibilité de questionner l'agent sur diverses thématiques. De plus, le *chatbot* est également à l'écoute de diverses banalités comme les salutations et formules de politesse ainsi que les feedbacks positifs et négatifs.

Par souci de lisibilité, il ne nous est pas possible de lister toutes les questions et réponses configurées pour notre agent conversationnel. C'est la raison pour laquelle, un tableau contenant toutes les alternatives de requêtes et de réponses sera placé en annexe de ce document. Le tableau suivant présente les différentes thématiques pouvant être abordées avec l'agent conversationnel.

Tableau 19 - Liste des thématiques accessibles depuis l'agent conversationnel

Localisation de l'IA	Localisation de l'utilisateur	Capacités de l'IA	Capacités de l'utilisateur
Identité	Objectif	Âge	Nom
Prénom	Naissance	Origines	Proches
Humour		Satisfaction	

Source : Données de l'auteur

6. BILAN ET PERSPECTIVES

Ce chapitre a pour objectif de parcourir dans un premier temps les différentes améliorations et optimisations possibles de l'application développée et ensuite de s'interroger sur l'état actuel de la réalité virtuelle et de l'intelligence artificielle.

6.1. AMÉLIORATIONS ET OPTIMISATIONS TECHNIQUES

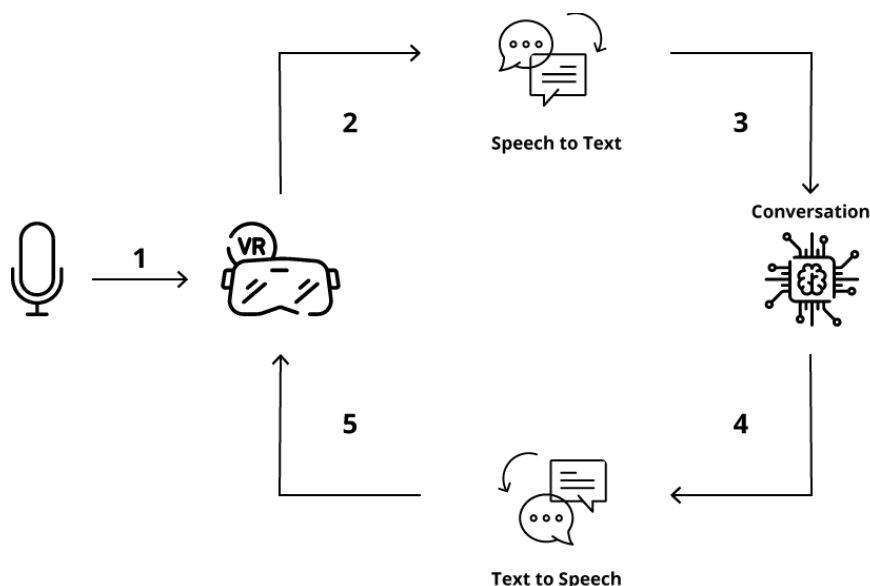
Malgré les diverses améliorations déjà apportées à l'application, nous constatons néanmoins plusieurs optimisations possibles.

6.1.1. Amélioration de l'architecture et de la latence

Dans un premier temps, nous relevons tout de même un certain temps de latence entre la requête de l'utilisateur vers le *chatbot* et la synthèse vocale de sa réponse en dépit de la réduction du délai déjà implémentée au point 5.5.2. Ce retard s'explique tout simplement au niveau de l'architecture de la solution. En analysant celle-ci, nous pouvons remarquer qu'une requête transite inutilement à plusieurs reprises via l'application. En implémentant une communication entre les différents services,

le temps d'attente serait inévitablement diminué. Le schéma suivant présente une correction de l'architecture de la solution.

Figure 24 - Amélioration de l'architecture de l'application



Source : Données de l'auteur

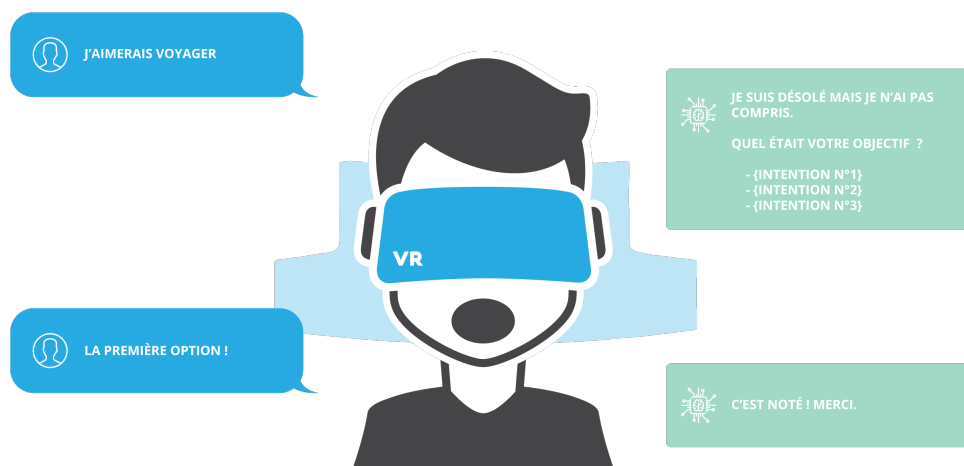
Néanmoins, il est important de noter qu'il est à l'heure actuelle impossible de procéder à ce changement. En effet, IBM ne propose aucune communication inter services pour le moment.

6.1.2. Apprentissage et création automatique des intentions

Dans un second temps, nous constatons également une amélioration possible pour l'agent conversationnel. Il serait effectivement possible d'optimiser l'apprentissage automatique du *chatbot* en implémentant un système automatisé de création d'intentions. La création de ces dernières est actuellement réalisée manuellement par nos soins.

Lors de l'incompréhension de l'agent face à une requête de l'utilisateur, il serait tout à fait envisageable de lui proposer plusieurs suggestions d'intentions. Dans le cas où la requête correspond à un objectif déjà prédéfini dans l'agent conversationnel, celle-ci serait simplement ajoutée à l'intention existante. Au contraire si elle ne correspond à aucune intention configurée, l'utilisateur détiendrait la possibilité d'expliquer à l'agent le but qu'il cherchait à atteindre. Ainsi, l'intelligence artificielle serait apte à évoluer progressivement de manière autonome. Le schéma suivant présente un apprentissage et création automatique des intentions à l'aide d'un système de suggestions.

Figure 25 - Apprentissage et création automatique des intentions



Source : Données de l'auteur

6.2. MATURITÉ DE LA RÉALITÉ VIRTUELLE ET ÉVOLUTION DE L'INTELLIGENCE ARTIFICIELLE

À la suite de l'exécution de ce travail, nous nous questionnons sur la maturité de la réalité virtuelle. Lors du développement du projet, nous avons constaté à plusieurs reprises une certaine instabilité dans les kits de développement officiels tels que SteamVR et Oculus Utilities. En effet, nous avons subi pendant le développement un total de deux mises à jour du logiciel de création de contenu *Unity*. À la suite de ses mises à jour, nous avons remarqué que la plupart des scripts tiers ne fonctionnaient plus.

À l'instar de la librairie amateur VRTK, ces derniers ne sont pas mis à jour assez régulièrement et se retrouvent donc inutilisables à la suite d'une mise à jour du moteur de jeu *Unity*. Ainsi, nous avons été contraints plusieurs fois de modifier manuellement le code source de ces modules externes afin de s'assurer de leur bon fonctionnement.

C'est la raison pour laquelle, nous nous demandons si la réalité virtuelle est assez mature à l'heure actuelle. En analysant les différents casques de réalité virtuelle disponibles à l'heure actuelle, nous pouvons relever une énorme différence de conception entre ces derniers ainsi qu'une différence importante de logique de développement. De cette manière, il n'est pas du tout aisé d'implémenter une application pour plusieurs casques de destination sans faire appel à un kit de développement amateur comme le VRTK nous permettant de changer plus ou moins facilement de support.

Il y a une certaine incohérence entre le succès de la réalité virtuelle, ses perspectives d'avenir prometteuses et son état de développement très instable. Nous estimons donc que la réalité virtuelle n'est peut-être pas suffisamment mature en l'état.

L'implémentation d'un agent conversationnel en réalité virtuelle nous a quant à elle agréablement surpris. Nous constatons que celle-ci présente de belles perspectives d'avenir. En effet, nous avons réussi dans le temps imparti à implémenter une solution *VR* exploitant l'intelligence artificielle. Cette dernière a été capable en très peu de temps de comprendre divers objectifs très précis et d'agir en conséquence.

Malgré divers problèmes techniques liés particulièrement à la réalité virtuelle, nous avons prouvé la faisabilité du projet ainsi que de son intérêt en *VR*. Un agent conversationnel possède donc toutes les qualités nécessaires pour devenir la nouvelle interface utilisateur, une interface entièrement conversationnelle. Au contraire de nos interfaces graphiques habituelles, celle-ci se démarque par une constante évolutivité.

Dans l'état actuel de l'application, il nous est tout à fait possible d'améliorer continuellement l'agent conversationnel. En effet, il nous suffit simplement d'ajouter de plus en plus d'intentions éclairées par toujours plus d'exemples. En définitive, il est important de remarquer qu'il n'y a pas de limite en ce qui concerne l'apprentissage de l'agent. À l'aide de l'apprentissage profond et automatisé, une intelligence artificielle est à priori capable de s'adapter à son utilisateur à tout moment ce qui n'est pas le cas d'une interface graphique. En développant une solution automatisée comme détaillée dans le point 6.1.2, il est envisageable de créer un agent conversationnel capable d'une constante évolution. En devenant de plus en plus intelligente, l'IA serait toujours en mesure de répondre et satisfaire les différents objectifs de l'utilisateur.

CONCLUSION

Lors de la réalisation de ce travail, nous avons progressivement analysé le potentiel de la fusion de deux technologies à succès, la réalité virtuelle et l'intelligence artificielle d'un agent conversationnel. Après avoir assimilé le fonctionnement général d'un *chatbot* intelligent, nous avons parcouru la plupart des agents conversationnels disponibles sur le marché afin de déterminer la solution la plus adéquate à implémenter en VR. En phase finale de ce projet, nous sommes parvenus à unir ces deux technologies récentes au sein d'une même application.

Avant l'exécution de ce travail, la programmation et le développement d'applications en 3D ainsi qu'en réalité virtuelle nous étaient totalement inconnus. Dans le cadre de ce projet, nous avons obtenu énormément de connaissances dans ce nouveau domaine qui est à l'heure actuelle en plein essor. Lors de la création et de la configuration de notre agent conversationnel, nous avons pris conscience à la fois de la complexité d'un tel programme, mais également des nombreuses perspectives d'avenir de cette technologie récente. En définitive, le développement de cette solution nous a peut-être permis de nous trouver un nouveau domaine d'intérêt et probablement de prédilection.

Malgré les nombreuses difficultés rencontrées en cours de développement, nous avons réussi à répondre aux divers objectifs que nous nous sommes imposés. Afin de répondre au besoin d'une application VR multiplateforme, nous avons été contraints de faire certains sacrifices importants comme la possibilité de locomotion virtuelle. Néanmoins, nous sommes tout de même entièrement satisfaits du résultat obtenu.

Étant donné que ce travail est à présent terminé, nous espérons que celui-ci encouragera le développement de nouvelles expériences en réalité virtuelle au sein de l'Institut Informatique de Gestion de la HES-SO Valais-Wallis.

RÉFÉRENCES

- Amazon. (s.d.). Amazon Lex. Récupéré sur AWS: <https://aws.amazon.com/fr/lex/>
- Amazon. (s.d.). Amazon Polly. Récupéré sur AWS: <https://aws.amazon.com/fr/polly/>
- Amazon. (s.d.). Description détaillée d'Amazon Lex. Récupéré sur AWS: <https://aws.amazon.com/fr/lex/details/>
- Amazon. (s.d.). FAQ sur Amazon Lex. Récupéré sur AWS: <https://aws.amazon.com/fr/lex/faqs/>
- Amazon. (s.d.). Ressources pour développeurs. Récupéré sur AWS: <https://aws.amazon.com/fr/lex/developers/>
- Amazon. (s.d.). Tarification d'Amazon Lex. Récupéré sur AWS: <https://aws.amazon.com/fr/lex/pricing/>
- Armstrong, M. (2016, Novembre 11). The Worldwide Virtual Reality Market Is Set To Be Huge. Récupéré sur statista: <https://www.statista.com/chart/6677/the-worldwide-virtual-reality-market-is-set-to-be-huge/>
- Caelen, J., & Villaseñor, L. (1997). Dialogue homme-machine et apprentissage. Apprentissage par l'interaction, 83-117.
- Cromby, J. J., Standen, P. J., & Brown, D. J. (1996, Décembre). The potentials of virtual environments in the education and training of people with learning disabilities. Journal of Intellectual Disability Research, pp. 489-501.
- Desai, R., Desai, P. N., Ajmera, K. D., & Mehta, K. (2014, Juillet). A Review Paper on Oculus Rift-A Virtual Reality Headset. International Journal of Engineering Trends and Technology (IJETT), p. 179.
- Georget, Y. (2018, Mars). L'intelligence artificielle, nouvelle interface utilisateur ? Enjeux Numériques, p. 44.
- GitHub. (s.d.). Learn Git and GitHub without any code! Récupéré sur GitHub: <https://github.com/>
- Google. (s.d.). Build natural and rich conversational experiences. Récupéré sur Dialogflow: <https://dialogflow.com/>
- Google. (s.d.). Dialogflow Actions and Parameters. Récupéré sur Documentation technique: <https://dialogflow.com/docs/actions-and-parameters>
- Google. (s.d.). Dialogflow Agents. Récupéré sur Documentation technique: <https://dialogflow.com/docs/agents>
- Google. (s.d.). Dialogflow Contexts. Récupéré sur Documentation technique: <https://dialogflow.com/docs/contexts>
- Google. (s.d.). Dialogflow Dialogs. Récupéré sur Documentation technique: <https://dialogflow.com/docs/dialogs>
- Google. (s.d.). Dialogflow Entities. Récupéré sur Documentation technique: <https://dialogflow.com/docs/entities>
- Google. (s.d.). Dialogflow Intents. Récupéré sur Documentation technique: <https://dialogflow.com/docs/intents>

- Google. (s.d.). Dialogflow Languages. Récupéré sur Documentation technique: <https://dialogflow.com/docs/reference/language>
- Google. (s.d.). Dialogflow Pricing. Récupéré sur Dialogflow: <https://dialogflow.com/pricing>
- Google. (s.d.). Dialogflow SDKs. Récupéré sur Dialogflow: <https://dialogflow.com/docs/sdks>
- McTear, M. F. (2002, Mars). Spoken Dialogue Technology: Enabling the Conversational User Interface. ACM Computing Surveys, p. 90.
- Microsoft. (2017, Décembre 17). Add intelligence to bots with Cognitive Services. Récupéré sur Documentation technique: <https://docs.microsoft.com/fr-fr/azure/bot-service/bot-service-concept-intelligence?view=azure-bot-service-3.0>
- Microsoft. (2017, Décembre 13). Bot scenarios. Récupéré sur Documentation technique: <https://docs.microsoft.com/en-us/azure/bot-service/bot-service-scenario-overview?view=azure-bot-service-3.0>
- Microsoft. (2017, Décembre 13). Bot Service templates. Récupéré sur Documentation technique: <https://docs.microsoft.com/en-us/azure/bot-service/bot-service-concept-templates?view=azure-bot-service-3.0>
- Microsoft. (2017, Avril 6). Culture-specific understanding in LUIS apps. Récupéré sur Documentation technique: <https://docs.microsoft.com/en-us/azure/cognitive-services/luis/luis-supported-languages>
- Microsoft. (2017, Juin 22). What is Language Understanding (LUIS)? Récupéré sur Documentation technique: <https://docs.microsoft.com/en-us/azure/cognitive-services/luis/home>
- Microsoft. (2018, Juin 28). Entities in LUIS. Récupéré sur Documentation technique: <https://docs.microsoft.com/en-us/azure/cognitive-services/luis/luis-concept-entity-types>
- Microsoft. (s.d.). Azure Bot Service. Récupéré sur Microsoft Azure: <https://azure.microsoft.com/en-us/services/bot-service/>
- Microsoft. (s.d.). Microsoft Repositories. Récupéré sur GitHub: <https://github.com/Microsoft>
- Microsoft. (s.d.). Reconnaissance vocale Bing. Récupéré sur Microsoft Azure: <https://azure.microsoft.com/fr-fr/services/cognitive-services/speech/>
- Microsoft. (s.d.). Tarification Azure Bot Service . Récupéré sur Microsoft Azure: <https://azure.microsoft.com/fr-fr/pricing/details/bot-service/>
- Microsoft. (s.d.). Tarification Cognitive Services - Services Speech. Récupéré sur Microsoft Azure: <https://azure.microsoft.com/fr-fr/pricing/details/cognitive-services/speech-services/>
- Microsoft. (s.d.). Tarification de Cognitive Services - Reconnaissance vocale (LUIS). Récupéré sur Microsoft Azure: <https://azure.microsoft.com/fr-fr/pricing/details/cognitive-services/language-understanding-intelligent-services/>
- Microsoft. (s.d.). Visual Studio. Récupéré sur Visual Studio: <https://visualstudio.microsoft.com/fr/>
- Oculus. (s.d.). Oculus GO. Récupéré sur Oculus: <https://www.oculus.com/go/>
- Oculus. (s.d.). Oculus Utilities for Unity. Récupéré sur Documentation technique: <https://developer.oculus.com/downloads/package/oculus-utilities-for-unity-5/>

- Oculus. (s.d.). Rift. Récupéré sur Présentation: <https://www.oculus.com/rift/#oui-csl-rift-games=star-trek>
- Pandorabots. (s.d.). About Pandorabots. Récupéré sur Documentation technique: <https://pandorabots.com/docs/>
- Pandorabots. (s.d.). Home. Récupéré sur Pandorabots: <https://home.pandorabots.com/en/>
- Pandorabots. (s.d.). Software Development Kits. Récupéré sur Documentation technique.
- Peixoto, R. (2015). LE CASQUE DE RÉALITÉ VIRTUELLE, LA NOUVELLE MANETTE DU FUTUR. HES-SO Valais-Wallis.
- Playstation. (s.d.). PS VR. Récupéré sur playstation: <https://www.playstation.com/fr-ch/explore/playstation-vr/>
- Software, V. (s.d.). SteamVR Unity Plugin. Récupéré sur GitHub: https://github.com/ValveSoftware/steamvr_unity_plugin
- Statista. (2017, Août). Chatbot market worldwide 2016 and 2025. Récupéré sur statista: <https://www.statista.com/statistics/656596/worldwide-chatbot-market/>
- Unity. (s.d.). Public Relations. Récupéré sur unity3d: <https://unity3d.com/fr/public-relations>
- Unity. (s.d.). Unity3D. Récupéré sur unity3d: <https://unity3d.com/fr>
- Vive. (s.d.). Produit. Récupéré sur vive: <https://www.vive.com/fr/product/>
- VRTK. (s.d.). Welcome to VRTK. Récupéré sur Documentation technique: <https://vrtoolkit.readme.io/docs>
- Watson, I. (2017, Septembre 28). Watson Tone Analyzer. Récupéré sur Documentation technique: <https://console.bluemix.net/docs/services/tone-analyzer/index.html#about>
- Watson, I. (2018, 2 7). Watson Assistant. Récupéré sur Documentation technique: <https://console.bluemix.net/docs/services/conversation/lang-support.html#langues-prises-en-charge>
- Watson, I. (2018, Janvier 26). Watson Assistant. Récupéré sur Documentation technique: <https://console.bluemix.net/docs/services/conversation/index.html#about>
- Watson, I. (2018, Juin 15). Watson Speech to Text. Récupéré sur Documentation technique: <https://console.bluemix.net/docs/services/speech-to-text/index.html#about>
- Watson, I. (2018, Juin 25). Watson Text to Speech. Récupéré sur Documentation technique: <https://console.bluemix.net/docs/services/text-to-speech/index.html#about>
- Watson, I. (s.d.). IBM Watson APIs. Récupéré sur GitHub: <https://github.com/watson-developer-cloud>
- Watson, I. (s.d.). Unity SDK. Récupéré sur GitHub: <https://github.com/watson-developer-cloud/unity-sdk>
- Watson, I. (s.d.). Watson Assistant. Récupéré sur IBM: <https://www.ibm.com/watson/services/conversation/pricing/index.html#pricing>
- Wit.ai. (s.d.). FAQ. Récupéré sur wit: <https://wit.ai/faq>

Wit.ai. (s.d.). Getting Started with Wit.ai. Récupéré sur Documentation technique:
<https://wit.ai/docs>

Wit.ai. (s.d.). Natural Language for Developers. Récupéré sur wit: <https://wit.ai/>

Wit.ai. (s.d.). Wit.ai Repositories. Récupéré sur GitHub: <https://github.com/wit-ai>

Annexe I : Liste d'exemples de requêtes pour chaque intention

Figure 26 - Exemples de requêtes pour chaque catégorie (partie 1)

AGENT_ENDING	AGENT_FORMAL_GREETINGS	AGENT_GENERAL_GREETINGS	AGENT_GET_AGE	AGENT_GET_BIRTHPLACE	AGENT_GET_CAPABILITIES
Bye bye	Glad to see you.	G, Ådday mate!	Are you an old woman?	Birth?	Can you please give me a list of the types of things you can help me with?
Bye now	Good to see you.	Good day	Are you old, dear?	Can you tell me anything about your birth?	Can you tell me what services you are able to help me with?
Catch you later	It is a pleasure to meet you.	Good evening	Are you old?	Can you tell me something about your birth?	Can you tell me what you can do?
Cya later	It's a pleasure meeting you.	Good morning	Can you tell me your age?	Can you tell me where you were born?	Can you tell me what you're capable of?
End trial	It's a pleasure to meet you.	Greetings	How old are you now?	Tell me about being born?	Do you have a list of things I can talk to you about?
Ending this session	It's good to see you.	Hello	How old are you, anyway?	Tell me about you being born?	Hello I am looking for some help here.
Finished now, good bye	It's nice to meet you.	Hello Agent	How old are you, huh?	Tell me about your birth?	Help now
Get lost	It, Åds nice to meet you.	Hey	How old are you?	Tell me how you were born?	Hi, what do you do?
Go away	Looking good SAM.	Hey girl	How old?	Tell me where you were born.	How can I use you?
Go off	Nice to meet you, sir.	Hey man	Just tell me how old you are.	Tell me where you're born.	How can you help me?
Going now	Nice to meet you.	Hey there	Tell me how old you are.	What's your birthplace ?	How can you help?
Good. bye.	Nice to see you.	Hey there all	Tell me your age, please.	What's your place of birth ?	How can you solve my problems?
Goodbye	Pleased to meet you.	Hey twin	Tell me your age.	Where are you born?	How do I use you?
Hey bot go away	Pleasure to meet you.	Hey you	What age are you?	Where were you born, huh?	I do not know what to ask
I am leaving		Hi	What's your age ?	Where were you born?	I'd like to know what type of questions you can answer for me.
I am out of here		Hi advisor	What's your age?	You were born?	Is there anything you can help me with?
I have got to go		Hi there	You old ?	Your birth?	Just tell me what you can do.
I want to quit		Hiya!	You old?		So tell me what you can do.
I'm done		Yo!	Your age?		Tell me about what kind of things you do?
I'm leaving now		You there	Your birthday?		Tell me what you can do?
Im done					Tell me what you can do.
Im good thank you					Tell me what you do.
It was nice chatting with you					Tell me what you're capable of.
Ok goodbye					Tell me what you're good at.
See you					Tell me what you're up to.
See you later					What are my options that you can help with?
Shut up					What are my options?
Stop doing this					What are you able to deal with?
Stop talking to me					What are you able to understand?
Thank you for your time					What are you capable of?
Thanks very much, bye!					What are you good at?
Thanks, bye!					What are you used for?
That is all					What are your skills?
That's everything					What can I ask you to do?
Time to go					What do you do?
					What do you handle?
					What else can you help me with?
					What features are in here?
					What kinds of things can you do?
					Where can I find the frequently asked questions?

Figure 27 - Exemples de requêtes pour chaque catégorie (partie 2)

AGENT_GET_GOAL	AGENT_GET_IDENTITY	AGENT_GET_JOKES	AGENT_GET_LIFE	AGENT_GET_LOCATION	AGENT_GET_NAME	AGENT_GET_ORIGINS	AGENT_GET_RELATIVES
Do you have any objectives ?	But who are you?	Another joke	About you.	Are you here?	Do you have a family name ?	What are your background?	Do you have a family?
So, what are you planning?	But who the hell are you?	Are there jokes?	About your life.	Are you in here?	Do you have a firstname ?	What are your origins?	Do you have a mom?
What are you doing here?	Can you introduce your self.	Can you tell a joke?	Let us talk about you	Are you in there?	Do you have a forename ?	What country are you from?	Do you have a mommy?
What are you here for?	Can you introduce yourself?	Can you tell me a joke?	Tell me about you.	Are you there?	Do you have a lastname ?	What is your country?	Do you have any family?
What are you in for?	Describe your self.	Do you have a joke?	Tell me about your life.	Hey, are you there?	Do you have a name?	What is your provenance?	Do you have any parents?
What are you planning?	Do you know who you are?	Do you have any jokes ?	Tell me an anecdote about you.	Hey, you in there?	Do you have a surname ?	What's your provenance?	Do you have any relatives?
What are you up to?	Introduce your self.	Do you have humor?	Tell me an anecdote about yourself.	Hey, you there?	Please declare your name	What's your source?	Do you have family?
What are your ambitions?	Introduce yourself.	Do you like fun?	Tell me something about you.	Hi, are you there?	Please state your name	Where are you from?	Do you have parents?
What do you desire?	Please state your identity.	Do you like humor?	Tell me something about your life.	Is anybody in here?	What's your family name ?	Where did you come from?	Do you have relatives?
What do you intend to do?	Present yourself.	I am getting bored	Tell me something about yourself.	Is anybody there?	What's your firstname ?	Where do you come from?	Who is your father?
What do you want from me?	What is your identity?	I want a joke	Tell me something of you.	Is anyone in here?	What's your forename ?	Where the fuck are you from?	Who is your mother?
What do you want?	What's your identity, then?	I'm bored	Tell me your life story.	Is anyone there?	What's your lastname ?	Where the fuck you from?	Who's your dad?
What is it you want?	What's your identity?	One more joke	Tell me your life.	Is there anybody here?	What's your name ?	Where the hell are you from?	Who's your daddy?
What is your ambition?	Who are you?	Surprise me with something hilarious	Tell me your story.	Is there anyone here?	What's your name?	Where the hell you from?	Who's your father?
What is your goal?	Who is this?	Tell me a joke	What should I know about you?	Is there anyone there?	What's your surname ?	Where you from?	Who's your mom?
What is your intention?	Who the fuck are you?	Tell me something funny		What is your location?		Where'd you come from?	Who's your mother, huh?
What is your objective?	Who the hell are you?	What do you do for fun?		What's your current location?		Which country are you from?	Who's your mother?
What is your purpose?		What is your favorite joke?		What's your localisation			You got a daddy?
What is your wish?				What's your location, girl?			You got a mom?
What're you planning?				What's your location, man?			You have a dad?
What's the reason of your presence ?				What's your location?			You have a daddy?
What's the reason you are here ?				Where are you ?			You have a mom?
What's the reason you're here ?				Where are you, man?			You've got a daddy?
What's your ambition?				Where the hell are you?			
What's your goal?							
What's your intention?							
What's your plan?							
What's your reason to be there ?							
Why are you here?							
Why here ?							

Figure 28 - Exemples de requêtes pour chaque catégorie (partie 3)

AGENT_GREETINGS_QUESTION	AGENT_HUMAN_OR_BOT	ENVIRONMENT_GET_AGE	ENVIRONMENT_GET_INFORMATION	ENVIRONMENT_GET_ORIGINS	ENVIRONMENT_GET_SIZE	ENVIRONMENT_MORE_INFORMATION
Alright mate?	Am I chatting with a human?	How long ago was that?	Give me information about it!	Explain her story to me?	How big is he?	Give me facts
Are you all right?	Am I talking to a Bot?	How long has it been there?	Give me information about this object !	Explain his story to me?	How big is it?	Give me facts about that!
Are you alright?	Am I talking to a person or am I talking to an AI?	How long has that been around?	Give me information on that.	How did it form?	How big is that?	Give me facts about this!
Are you OK?	Am I talking to Watson?	How long has this been around?	Give me information on this.	How did it get here?	How big is this place?	Give me more info.
Are you okay?	Are you a human being?	How long has this existed?	Give me some information about it!	How did that come about?	How big is this?	Give me more information.
Are you sure you're okay?	Are you a human or a bot?	How old is it?	Give me some information on that.	How did that form?	How high is it?	Give me some facts
Have a nice day?	Are you a robot?	How old is she?	What does it do?	How did that happen?	How huge is it?	Give me some facts about that!
Have you been well?	Are you an artificial intelligence?	How old is that object?	What good is that?	How did this come about?	How huge is this?	Give me some facts about this!
Hey how are you doing?	Are you even real?	How old is that?	What is this item?	How did this get here?	How tall is he?	Give me some more info.
How are things going?	Are you human?	Since when is it here?	What is this object for?	How did this get in here?	How tall is it?	Give me some more information.
How are things?	Are you real?	Since when is this here?	What is this object used for?	How did this happen?	How tall is this?	Give me the facts
How are you doing?	How can you prove you are a human?	What's his age?	What is this object?	How it was formed?	How wide is it?	Just give me some facts.
How are you today?	Is this a computer?	When did this get here?	What the fuck is going on?	How this formed?	What is its circumference?	Tell me more about it.
How are you?	Tell me, are you a human or no?	When did this happen?	What the fuck is this?	How'd it get here?	What is its diameter?	Tell me more about that item
How do you do?	You're a robot?	When was this item made?	What the hell is going on?	Tell me her story?	What is its girth	Tell me more about that.
How have you been?		When was this object made?	What the hell is that?	Tell me his story?	What is its height?	Tell me more about the object
How is it going?			What the hell is this?	What are the origins of that?	What is its size?	Tell me more about this item
How r u?			What's that all about?	What are the origins of this?	What is its width?	Tell me more about this object
How,À¿s everything ?			What's that for?	What's the origin of that?	What is the circumference?	Tell me more about this.
How,À¿s it going?			What's that?	What's the origin of this?	What is the diameter?	Tell me more on that.
How,À¿s life?			What's the object for?	Who built it?	What is the width?	What else can you tell me?
How,À¿s your day going?			What's the point of that?	Who built that?	What size is it?	What more can you say to me?
How,À¿s your day?			What's the point?	Who built this?	What size is that?	What more can you say?
Sup?			What's the use of that?	Who made that?	What size is this?	What more can you tell me?
What's new?			What's this object for?	Who made this?	What's his height?	
What's up?			What's this object?	Who put that in there?	What's its circumference	
What,À¿s going on?				Who put that there?	What's its diameter?	
What,À¿s new?				Who put this here?	What's its width?	
What,À¿s up?				Who put this there?	What's the diameter?	
Whazzup?						
You all right?						
You alright?						
You're okay?						

Figure 29 - Exemples de requêtes pour chaque catégorie (partie 4)

GENERAL_NEGATIVE_FEEDBACK	GENERAL_POSITIVE_FEEDBACK	MUSIC_NEXT_COMMANDS	MUSIC_PLAY_COMMANDS	MUSIC_STOP_COMMANDS	USER_GET_CAPABILITIES	USER_GET_LOCATION
Do not like you?	Brilliant!	Change playlist.	Do you have a playlist?	Enough of this!	What am I supposed to do?	What am I doing here?
Everyone hates you	Can't believe you are that good	Change the music.	Do you have any music?	Enough!	What are my abilities?	What am I doing in here?
Fuck you	How cool is this?	Change the playlist.	Do you have any musics?	Pause it.	What are my capabilities?	What is my location?
Hate you	I like what you did there! :)	Change the song.	Do you have any songs?	Pause music.	What are my capacities?	What the hell am I doing here?
I do not like you	I'm looking forward to working with you again! :)	Change your music.	Do you have music?	Pause playlist.	What are my choices?	What's my location, please?
I hate this!	Love your work	Changes the playlist.	Do you have songs?	Pause the music.	What are my opportunities?	What's my location?
I hate you	Ok thank you	Changing the playlist.	Have you got a playlist?	Pause the playlist.	What are my options?	Where am I located?
It is annoying	Thank you	Next one.	Have you got any music?	Pause the process.	What are my possibilities?	Where am I now?
Quit annoying me	This is good	Next song, please.	Have you got any songs?	Pause.	What are my skills?	Where am I?
Robots are boring	This is great	Next up.	Launch a playlist.	Pauses music.	What can I do?	Where do I stand?
Robots are stupid	This is so cool	Next, please.	Launch a song.	Pauses playlist.	What options do I have?	Where the hell am I?
Stupid	You are awesome	Next.	Launch music.	Pauses the music.	What possibilities do I have?	
Why are you so annoying?	You are great	Switch music.	Play a song.	Pauses the playlist.	What's the list of things I can do?	
Why are you stupid?	You are the best	Switch the music.	Play music.	Pauses.		
You are having delusions	You are wonderful	You change the song.	Run a playlist.	Stop it, please.		
You are on my nerves	You gave me exactly what I need!		Start a music.	Stop it.		
You are very frustrating	You the man		Start a playlist.	Stop music.		
You do not seem smart	You're a genius!		Start a song.	Stop playing music.		
You're really frustrating	You've been so helpful :)		Start playlist.	Stop playlisting.		
You're really irritating			Start the playlist.	Stop singing the song.		
You're too stupid			Start up a playlist.	Stop singing.		
			Throw a playlist.	Stop that playlist.		
			Throw a song.	Stop that song.		
			You got a playlist?	Stop the music, please.		
			You got any songs?	Stop the music.		
				Stop the playlist.		
				Stop the song.		
				Stop with the playlist.		
				Stop.		
				That's enough!		

Annexe I : Planification

Figure 30 - Planification du projet

TÂCHES	DATE DE DÉBUT	DATE D'ÉCHÉANCE	% ACHÈVEMENT	TERMINÉ	LIEU	REMARQUES
Premier meeting avec M. Widmer	30.04.18	30.04.18	100 %	●	Technopôle	
Première planification du travail de bachelor	07.05.18	09.05.18	100 %	●	N/A	
Réalisation du product backlog	10.05.18	11.05.18	100 %	●	N/A	
Premières recherches à propos de l'état de l'art	10.05.18	14.05.18	100 %	●	N/A	
Deuxième meeting avec M. Widmer	14.05.18	14.05.18	100 %	●	Technopôle	
Définition de l'hypothèse et de l'objectif du travail de bachelor	14.05.18	16.05.18	100 %	●	N/A	
Etude de besoins de l'application	16.05.18	17.05.18	100 %	●	N/A	
Réalisation de l'état de l'art	16.05.18	26.05.18	100 %	●	N/A	
Séminaire international à Helsinki (Absence)	20.05.18	26.05.18	100 %	●	Finlande	
Troisième meeting avec M. Widmer	28.05.18	28.05.18	100 %	●	Technopôle	
Réalisation d'un benchmarking concernant les résultats obtenus suite à l'état de l'art	28.05.18	29.05.18	100 %	●	N/A	
Choix et décision de la technologie selon les résultats du benchmarking et en accord avec l'étude des besoins	30.05.18	30.05.18	100 %	●	N/A	
Définition de l'architecture à mettre en place	31.05.18	02.06.18	100 %	●	N/A	
Implémentation et développement de la solution	05.06.18	29.06.18	100 %	●	N/A	
Réalisation du rapport	01.07.18	16.07.18	100 %	●	N/A	
Quatrième meeting avec M. Widmer	17.07.18	17.07.18	100 %	●	Technopôle	Demande d'ajouter un support de l'Oculus GO
Implémentation et développement du support de l'Oculus GO	17.07.18	23.07.18	100 %	●	N/A	
Test de la solution mis en place	17.07.18	23.07.18	100 %	●	N/A	
Finalisation du rapport	23.07.18	27.07.18	100 %	●	N/A	
Dépôt du travail de bachelor	30.07.18	30.07.18	100 %	●	N/A	

Annexe II : Product backlog

Figure 31 - Product backlog et nombre d'heures effectives

US N°	THÈME	EN TANT QUE ...	JE VEUX ...	AFIN DE ...	ACCEPTANCE CRITERIAS	STORY POINTS	PRIORITÉ	HEURES REALIS	DONE
8	Préparation	Développeur	définir l'hypothèse et l'objectif du travail de bachelier	d'orienter mon travail vers un use case particulier	Définition de l'hypothèse Définition de l'objectif Rédaction de l'avant propos Rédaction de l'introduction	13	4000	10	
7	Préparation	Développeur	réaliser une étude de besoins de l'application	définir les fonctionnalités nécessaires de l'application	Analyse des besoins de l'application Listing des fonctionnalités spécifiques nécessaires Validation du " product owner "	8	3750	8	
9	Préparation	Développeur	réaliser une étude technique du fonctionnement des agents conversationnels	de comprendre les concepts utilisés par une intelligence artificielle	Définition d'un agent conversationnel Définition des principes fondamentaux d'un chatbot Rédaction du fonctionnement général d'un agent	13	3500	25	
5	Préparation	Développeur	réaliser un état de l'art technique des différentes solutions de chatbot	de connaître les technologies existantes	État de l'art des différentes technologies existantes Spécificités techniques Critères d'évaluation	34	3250	120	
6	Préparation	Développeur	réaliser un benchmarking des différentes technologies existantes	de m'aider à la décision d'une solution adéquate pour le projet	Benchmarking des différentes intelligences artificielles chatbot disponibles basé sur les mêmes critères d'évaluation	13	3000	10	
4	Préparation	Développeur	choisir une technologie adéquate au projet	de répondre aux besoins de la solution	Choix d'une technologie finale pour la solution La technologie répond aux besoins de la solution Le choix de la technologie est validée par les résultats du benchmark	8	2750	2	
7	Environnement	Développeur	avoir un accès au repository du projet	de faciliter le partage des fichiers du projet à la fin de celui-ci	Création d'un repository Github publique Réalisation d'une documentation technique pour les accès au projet Configuration de l'environnement de développement	8	2500	8	
10	Développement	Utilisateur	pouvoir parler vocalement avec un chatbot en réalité virtuelle	d'avoir une discussion orale avec une intelligence artificielle	Intégration d'une API de reconnaissance vocale en VR Enregistrement du flux audio du microphone Création d'un fichier contenant l'enregistrement L'API retourne la retranscription textuelle du message vocale	34	2250	40	
11	Développement	Utilisateur	obtenir une réponse du chatbot à ma requête	d'avoir une discussion cohérente avec une intelligence artificielle	Intégration d'une API d'intelligence artificielle Le chatbot interagit aux questions de l'utilisateur Le chatbot est capable de tenir une discussion cohérente avec l'utilisateur	34	2000	25	
1	Développement	Utilisateur	que l'intelligence artificielle soit dotée d'une voix	d'avoir l'impression à un humain	Intégration d'une API de synthèse vocale Le chatbot est doté d'une voix La voix du chatbot comporte plusieurs expressions / sentiments	34	1750	30	
2	Développement	Utilisateur	pouvoir poser des questions au chatbot sur mon environnement	d'obtenir de plus amples informations sur ce qui m'entoure	Implémentation de questions spécifiques à propos du cible L'agent de conversationnel est capable de donner des informations sur l'objet cible	21	1500	15	
12	Développement	Utilisateur	pouvoir utiliser l'application sur plusieurs casques de réalité virtuelle	n'être pas limité à une seule plateforme	Application tourne sur deux plateformes différentes Une seule application pour toutes les plateformes supportées Toutes les fonctionnalités sont disponibles d'une plateforme à l'autre	21	1250	25	
13	Rédaction	Développeur	lister les différentes technologies utilisées pour la réalisation du projet	afin qu'un autre développeur puisse reprendre le projet en l'état	Lister toutes les technologies utilisées Documentation technique Rédaction des technologies utilisées et expliquer pourquoi	13	1000	10	
14	Rédaction	Développeur	lister les difficultés rencontrées lors de la réalisation de l'application	montrer les solutions que j'ai trouvées pour résoudre mes problèmes	Lister toutes les difficultés rencontrées Rédaction des solutions à chaque problème rencontré	8	750	9	
15	Rédaction	Développeur	définir la conclusion et le bilan général du travail	de terminer la rédaction de mon travail de bachelier	Rédaction des connaissances apprises Esprit critique à propos de l'application réalisée Lister les améliorations / optimisations possibles de l'application	8	500	10	
3	Développement	Utilisateur	pouvoir marcher dans un univers virtuel crée	d'être immergé dans l'application VR	Moyen de locomotion dans l'univers crée Locomotion identique et fonctionnelle dans chacune des plateformes supportées	21	250	25	
TOTAL								372	

DÉCLARATION DE L'AUTEUR

Je déclare, par ce document, que j'ai effectué le travail de Bachelor ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de Bachelor, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail et que je cite ci-après :

- Antoine Widmer

Sierre, le 30 juillet 2018

Rafael Peixoto