



Réf : Ing-GMAM-2019-01

Rapport de Projet de Fin d'Études

Pour obtenir le

**Diplôme d'Ingénieur en Génie Mathématiques Appliquées
et Modélisation**

Option : Fiabilité et Maintenance

Présenté et soutenu publiquement le 1 juillet 2019 Par

Yassmine JAROUDI

**Création d'assistant virtuel pour automatisation
de services internes d'une entreprise**

Organisme d'accueil



Composition du jury

Monsieur	Adnene EL YAACOUBI	Président
Madame	Emna SOUSSI	Rapporteur
Monsieur	Nasser RABEI	Encadrant Entreprise
Monsieur	Koutheair KHRIBI	Encadrant ENSIT

Année universitaire : 2018-2019

Remerciements

En premier lieu, je tiens à remercier Dr. Koutheair KHRIBI d'avoir accepté de m'encadrer et de me soutenir tout au long du stage. Je tiens à lui exprimer toute mon admiration et ma reconnaissance. Je remercie également M. Nasser RABEI Directeur Technique Opérationnel de l'entreprise BI-Ready et mon maître de stage, pour sa confiance, sa disponibilité et sa patience. Je remercie sincèrement M. Raphael DUCASSE, Directeur de BI-Ready de m'avoir acceptée pour effectuer ce stage de projet de fin d'études, pour son temps précieux et ses conseils tout au long du déroulement de ce stage.

Je tiens à remercier Dr. Adnène EL YAACOUBI, Directeur de département de Mathématiques Appliquées et Modélisation et Président du jury, de m'avoir apporté un soutien sans faille, notamment en ce qui concerne les démarches administratives relatives à un stage à l'étranger. Je remercie également Dr. Emna SOUISSI d'avoir accepté d'évaluer le travail réalisé dans ce rapport. J'exprime toutes mes reconnaissances aux membres du jury, qui ont bien voulu accepter de faire partie du jury de ce projet.

Je tiens également à remercier tout le corps professoral et administratif de l'École Nationale Supérieure d'Ingénieurs de Tunis (ENSIT). Je remercie aussi toute l'équipe de BI-Ready pour leur accueil chaleureux et leur bienveillance.

Finalement, j'adresse mes profondes gratitude à ma famille qui a été toujours à mes côtés, pour son soutien et son encouragement.

Table des matières

Introduction Générale	1
1 Présentation générale du projet	3
1.1 Présentation de l'organisme d'accueil	3
1.2 Contexte général	4
1.3 Présentation du projet	5
1.4 Conclusion	6
2 Analyse et spécification des besoins	7
2.1 Capture des besoins	7
2.1.1 Spécifications du client	7
2.1.2 Identification des acteurs	8
2.1.3 Identification des besoins	8
2.2 Étude de l'existant	8
2.3 Solution proposée	16
2.4 Analyse Globale	17
2.4.1 Diagramme de cas d'utilisation	17
2.4.2 Diagramme de classes	19
2.4.3 Architecture fonctionnelle	20
2.5 Conclusion	21
3 Module Azure chatBot	22
3.1 Conception préliminaire et détaillée	22
3.1.1 Architecture logique	22
3.1.2 Diagramme de séquence	23
3.2 Branche technique	24
3.2.1 Outils et technologies	24
3.2.2 Architecture physique	24
3.2.3 Diagramme de composants	25
3.3 Tests et codage	26
3.4 Conclusion	27
4 Module d'analyse linguistique	28
4.1 Conception préliminaire et détaillée	28
4.1.1 Architecture logique	28
4.1.2 Diagramme de conception de séquence et de classes	29
4.2 Branche technique	30
4.2.1 Outils et technologies	30
4.2.2 Architecture physique	32

4.2.3	Diagramme de composants	32
4.3	Tests et codage	33
4.4	Conclusion	36
5	Module de recherche de données	37
5.1	Conception préliminaire et détaillée	37
5.1.1	Architecture logique	37
5.1.2	Diagramme de conception de séquence et de classes	38
5.2	Branche technique	39
5.2.1	Outils et technologies	39
5.2.2	Architecture physique	39
5.2.3	Diagramme de composants	40
5.3	Tests et codage	41
5.4	Conclusion	44

Table des figures

1.1	Logo de BI-Ready	3
1.2	Principe de "Turing Test"	4
1.3	Diagramme de Gantt	6
2.1	Logique de flux de Cortana skills bot [17]	12
2.2	Logique de flux de enterprise productivity bot [17]	12
2.3	Logique de flux de IoT bot [17]	13
2.4	Logique de flux de Information bot [17]	14
2.5	Site web officiel du gouvernement de Los Angeles	16
2.6	Diagramme de cas d'utilisation	17
2.7	Raffinement CU « Demander des informations sur le département de production »	18
2.8	Raffinement CU « Demander des informations générales sur l'entreprise »	19
2.9	Diagramme de classes du chatBot	20
2.10	Architecture fonctionnelle du chatBot	21
3.1	Architecture logique de la communication avec l'utilisateur	22
3.2	Diagramme de séquence de la communication avec l'utilisateur	23
3.3	Architecture physique de la communication avec l'utilisateur	25
3.4	Diagramme de composants de la communication	26
3.5	Message d'accueil de connexion	26
3.6	Capture d'écran de débogage du message d'accueil	27
4.1	Architecture logique d'analyse linguistique	28
4.2	Diagramme de classes d'analyse linguistique	29
4.3	Diagramme de séquence d'analyse linguistique	30
4.4	Microsoft Cognitive Services	31
4.5	Architecture physique d'analyse linguistique	32
4.6	Diagramme de composants d'analyse linguistique	33
4.7	Test de l'application QnA Maker	33
4.8	Débogage de l'activité d'appel de QnA Maker	34
4.9	Test de l'application LUIS Production Departement	34
4.10	Débogage de l'activité d'appel de Production Departement	35
4.11	Débogage de l'output de l'application "Département de production"	35
5.1	Architecture logique de la recherche	37
5.2	Diagramme de classes de la recherche	38
5.3	Diagramme de séquence de la recherche	39
5.4	Architecture physique de la recherche	40

5.5	Diagramme de composants de la recherche	41
5.6	Résultat de requête de supérieurs	41
5.7	Débogage de requête de supérieurs	42
5.8	Résultat de requête d'inférieurs	42
5.9	Débogage de requête d'inférieurs	43
5.10	Test et débogage de requête de marge de prix	43
5.11	code de paramètres de requête de comparaison de prix	44
5.12	Test de requête de maximum	44
5.13	Test de requête de minimum	44

Introduction Générale

Le présent rapport fait l'objet d'un projet de fin d'études réalisé dans l'entreprise BI-Ready et porte sur le thème de "Conception et Réalisation d'un assistant virtuel d'entreprise".

Afin de répondre aux besoins du directeur de l'entreprise, l'outil proposé permet de faciliter la gestion et le contrôle du département de production et répondre instantanément et automatiquement aux requêtes saisies en langage naturel en un temps-record.

Pour la gestion et le contrôle de la chaîne de production, le directeur de l'entreprise organise des réunions avec les différents responsables du département de production. Ces employés passent beaucoup de temps à construire des rapports afin d'obtenir la meilleure représentation des données et le directeur doit payer pour tous ces heures de travail.

Les données présentées dans les rapports sont souvent confidentielles, ce qui risque de dévoiler les propos internes de l'entreprise aux autres concurrents du marché.

L'interprétation des inconvénients constatés dans le processus métier existant suscite la nécessité d'adopter une nouvelle solution alternative. Après avoir réalisé une étude du marché, on a constaté que la meilleure méthode est d'automatiser les fonctionnalités demandées par le directeur de l'entreprise. Il fallait donc développer une application qui accède à toutes les données reliées au département de production pour répondre à la recherche du directeur. Le projet sera présenté sous forme d'agent conversationnel capable d'établir un dialogue avec l'utilisateur. Ce dernier ne doit avoir aucune connaissance technique en programmation pour pouvoir utiliser l'application. La consultation des données existantes ne s'effectue pas à travers des lignes de code en langage SQL mais en saisissant des questions en langage naturel. L'utilisation de l'application est donc simple et rapide.

Le chatBot doit détecter les messages de l'utilisateur, comprendre le sens d'un message et donner une réponse correspondante à sa demande. Si une question est détectée, l'application doit chercher si elle correspond à une requête valide et retourne une réponse générée par son algorithme de logique.

Le chatBot a été développé utilisant comme éditeur Visual Studio, C# comme langage principal de programmation et .NET Core 2 comme Framework de développement. Pour le design du chatBot, on a utilisé les services du Cloud de Microsoft Azure tels que LUIS, QnA Maker, Azure Search et d'autres technologies de Microsoft notamment Microsoft Bot Framework v4.

Le chatBot est en cours de finalisation dans l'organisme d'accueil et passera en phase de test chez un client dans un réseau interne dès que l'implémentation d'une application complémentaire de gestion de tâches sera achevée. Le projet aura pour résultat final le déploiement d'un chatBot dans un canal de chat privé pour le directeur de l'entreprise. Après le succès du déploiement de la base fonctionnelle du

chatBot, d'autres services pourront être rajoutés par la suite comme l'authentification des utilisateurs.

Ce rapport est présenté en cinq chapitres. Le premier chapitre "Présentation générale du projet" représente un tracé de l'organisme d'accueil BI-Ready et le contexte de l'application. Le deuxième chapitre "Analyse et spécification des besoins" se penche sur les principaux aspects de la solution existante, analyse les exigences du client et donne une idée générale sur la solution proposée. Le troisième chapitre "Module Azure chatBot" décrit le fonctionnement du premier module de l'application et la conception mise en place pour la création de ce dernier. Le quatrième chapitre "Module d'analyse linguistique" décrit le principe de fonctionnement du deuxième module de l'application et l'architecture choisie. Le cinquième chapitre "Module de recherche de données" représente la conception du troisième module de l'application, les outils utilisés et les captures de test de fonctionnalités. Enfin ce rapport sera achevé par une conclusion générale .

Chapitre 1

Présentation générale du projet

Dans ce chapitre, on présente l'organisme d'accueil où le stage s'est déroulé, une introduction des chatBots et on donne une description du projet réalisé.

1.1 Présentation de l'organisme d'accueil



FIGURE 1.1 – Logo de BI-Ready

BI-READY est une startup créée le 28 Mars 2018. Elle offre des solutions en Business Intelligence et Artificial Intelligence pour des entreprises qui souhaitent automatiser leurs processus de travail, intégrer de nouvelles technologies et exploiter leurs données pour planification budgétaire et aide à la décision. BI-READY est un partenaire Gold de Microsoft ce qui lui donne l'avantage d'accès aux différents services du Cloud de Microsoft Azure et des solutions de Power BI.

La société est composée d'un bureau exécutif, de consultants Microsoft Power BI et d'ingénieurs d'affaires. Le bureau exécutif contient les membres suivants :

- * M. Raphael Ducasse Chief Executive Officer
- * M.Michel Rolland Chief Financial Officer
- * M.Nasser Rabei Chief Technical Officer

L'un des objectifs de l'entreprise pour l'année 2020 est d'intégrer les nouvelles technologies d'Azure et de se lancer dans le marché de l'Intelligence Artificielle. BI-READY décide donc dans le cadre d'une première initiative de réaliser un projet où elle utilise les services de l'Intelligence Artificielle IA de cloud d'Azure.

Afin d'avoir les bons conseils pour sa stratégie commerciale dans ce nouveau marché, BI-READY va présenter ce projet à Microsoft. L'équipe de Microsoft évaluera le projet d'un point de vue technique et en termes de besoins du marché. Elle se chargera ensuite de diriger BI-READY vers les clients qui sont prêts à adopter ce projet dans leurs entreprises. L'application présentée dans ce rapport est une première version de ce projet où on découvre les possibilités fonctionnelles que peut offrir Microsoft Azure.

1.2 Contexte général

Les chatBots sont des applications comparables aux applications web. Ils ont pour but d'offrir une expérience qui donne l'impression aux utilisateurs d'avoir moins à faire à un ordinateur, et plus à une personne, ou du moins à un robot intelligent.

Les chatBots existaient depuis longtemps mais ils sont devenus de plus en plus populaires et disponibles ces dernières années comme cité dans cet article [12]. Effectivement, le premier chatBot existait depuis 1950 grâce à une expérience réalisée par Alan Turing connue pour "the Turing Test" (voir figure 1.2). Au cours de cette expérience les ordinateurs devaient simuler le comportement humain dans des conversations avec des utilisateurs afin de passer le test.

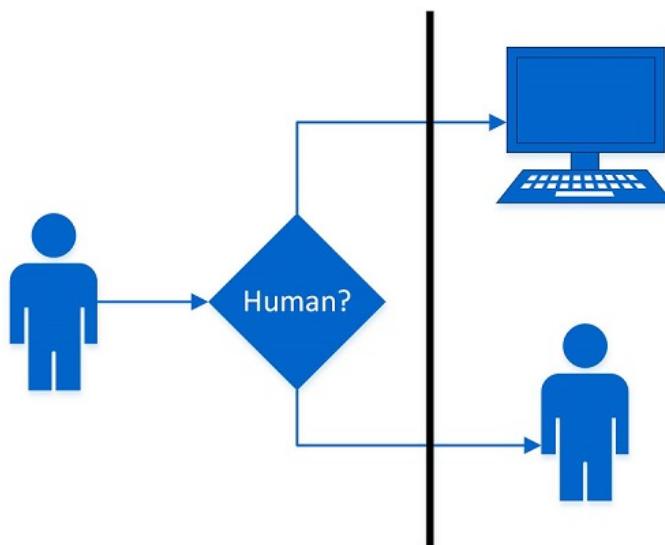


FIGURE 1.2 – Principe de "Turing Test"

Après cette expérience, d'autres chatBots sont apparus comme Elisa de Joseph Weizenbaum en 1966 qui était le premier chatBot développé avec 200 lignes de code. Malgré son échec au test de Turing, Elisa représentait la fondation des structures de chatBots comme les mots-clés, les phrases spécifiques et les réponses prédéfinies.

De plus, l'époque des téléphones portables a créé le besoin d'adapter les applications web aux écrans de petites tailles des téléphones. Le design d'application est devenu par la suite très couteux et compliqué car les développeurs avaient toujours pour priorité l'amélioration de la "User Experience". Tous ces facteurs ont suscité le besoin urgent de satisfaire l'utilisateur avec des produits sans écrans connus sous le nom de chatBot.

Aujourd’hui, l’histoire humaine a suffisamment évoluée pour donner naissance à des chatBot qui non seulement peuvent répondre à des questions prédéfinies mais aussi de s’adresser directement aux utilisateurs, jouer de la musique et faire des recherches sur internet etc. Par conséquent, les bots sont devenus très populaires dans les grandes entreprises de High Tech [12] comme Siri (2010), Google Now (2012), Alexa (2015), et Cortana (2015). Le présent projet est un modeste exemple qui montre l’impact que peut avoir un chatBot sur le processus d’un business.

1.3 Présentation du projet

Le projet effectué au sein de l’entreprise BI-Ready dans le cadre d’un projet de fin d’études consiste à développer un chatBot qui peut comprendre et répondre à la question d’un utilisateur en se basant sur des données dynamiques en back-end. Ce projet a été proposé suite à la demande d’un client qui avait besoin d’interface intelligente pour faciliter le contrôle et la gestion de son entreprise.

Le projet a été réalisé selon les phases suivantes :

1. Cadrage : dans cette phase on organise des réunions avec le client pour comprendre ses besoins et le genre de scénario qu’il souhaite avoir dans le chatBot. On obtient aussi l’accès aux données utilisées et on fixe la limite du budget pour le projet. On termine cette phase par établir une liste de priorités et obligations expliquées dans le chapitre suivant.
2. Conception : après la confirmation du client concernant le budget et la date limite du projet, on commence par la conception de l’architecture de l’application, les besoins fonctionnels, les modules et les services utilisés.
3. Développement : dans cette partie on commence à développer la solution et l’implémentation des différentes fonctionnalités selon les mesures prises dans les étapes précédentes.
4. Publication et production : une fois le développement du chatBot est réalisé, on passe au déploiement du projet dans le Cloud, sa publication et sa connexion à un canal de Facebook Messenger. Ensuite, on donne au client accès au chatBot afin de le tester et donner son feedback qui sera utilisé pour des éventuelles améliorations.
Enfin, on donne au client l’accès total au chatBot et on lui offre des services de maintenance afin de surveiller le fonctionnement de l’application.

La réalisation des différentes phases du projet est organisée selon un diagramme de Gantt. Le diagramme présenté dans la figure 1.3 résume les étapes parcourues pour créer le chatBot.

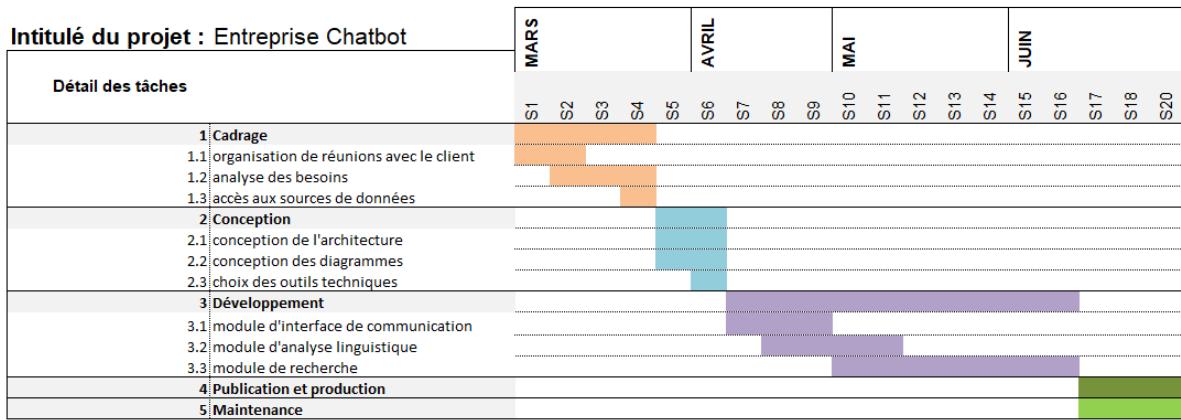


FIGURE 1.3 – Diagramme de Gantt

1.4 Conclusion

Ce chapitre nous a permis de connaître l'entreprise d'accueil, mettre le projet dans son cadre et avoir une vision globale sur les chatBots. Dans le chapitre suivant nous aborderons avec plus de détails les besoins du client, la conception du projet et sa position par rapport aux solutions existantes.

Chapitre 2

Analyse et spécification des besoins

Dans ce chapitre, on présente une description des besoins du client, la méthodologie adaptée pour la réalisation du projet, une étude des solutions existantes et une analyse globale de l'application.

2.1 Capture des besoins

C'est la phase de cadrage décrite dans le chapitre précédent où le client décrit les contraintes qu'il a, ses besoins ainsi que ses attentes de l'application à réaliser. Le client souhaite avoir une solution qui lui permet d'économiser du temps et de l'argent. Le projet doit donc donner une solution plus rapide, efficace et qui coûte moins cher.

2.1.1 Spécifications du client

Les spécifications du client ont été collectées au cours de plusieurs réunions. La phase de cadrage était d'un côté une occasion pour le client pour exprimer ses besoins et ses attentes, et une possibilité de chercher la faisabilité de ses demandes d'un point de vue technique. Les spécifications peuvent se résumer dans la liste suivante selon le degré de priorité :

1. compréhension du langage naturel du directeur de l'entreprise et les responsables des départements
2. connexion à des tables SQL contenant les données de l'entreprise
3. répondre aux questions de l'utilisateur de manière naturelle et conversationnelle
4. automatisation de la mise à jour des connaissances du chatBot lors de la modification des sources de connaissances
5. contrôle de l'accès au chatBot par un compte administrateur
6. suivi du flux d'utilisation du chatBot et ses performances au cours de son utilisation
7. disponibilité du chatBot dans différents canaux comme Messenger, Slack ou Skype

2.1.2 Identification des acteurs

Un acteur est une personne ou un organisme qui interagit avec le système. Par définition, les acteurs sont à l'extérieur du système. Ils sont composés des utilisateurs, ses responsables de la configuration et de la maintenance. Selon la base de données utilisée, les acteurs potentiels qui risquent d'interagir avec l'application sont :

- Développeur : c'est une personne qui a des connaissances en Azure Bot Service et .NET. Il est responsable du développement de l'application du chatBot qui servira d'interface de communication.
- Client : c'est une entreprise spécialisée dans la production des équipements de Cyclisme. Elle souhaite intégrer cette application dans un premier temps au niveau de la direction afin d'améliorer ses services de contrôle et de gestion. Le produit réalisé sera utilisé par le direction générale de l'entreprise.
- Microsoft : BI-Ready est partenaire Gold avec Microsoft. Ce partenariat permet au développeur de contacter le support technique de Microsoft pour intervenir en cas de défaillances imprévues dans le Cloud d'Azure. De plus, Microsoft donne accès aux différents services de Microsoft Azure qui seront exploités dans le projet.

Il est à noter que ce projet va être relié à une autre application qui a pour fonction la gestion des tâches pour le département de Ressources Humaines. Cette dernière est en cours de développement pour le moment. La phase de déploiement du chatBot n'est donc pas achevée et le test du chatBot s'effectuera ainsi en local.

2.1.3 Identification des besoins

Besoins fonctionnels

Les fonctionnalités que doit offrir l'application peuvent se résumer dans la liste suivante :

- Compréhension du message de l'utilisateur
- Capacité de répondre à la demande de l'utilisateur
- Capacité de fournir les informations appropriées à la requête
- Connexion à la base de données de l'entreprise et sa compréhension

Besoins non-fonctionnels

- Facilité d'utilisation : l'application offre une expérience simple et rapide. L'utilisateur a accès aux informations en écrivant une question simple en langage naturel qui correspond à sa requête.
- Disponibilité : l'application est utilisable 24h/24 et 7 jours/7.
- Confidentialité : les données sur lesquels se base le chatBot sont confidentielles et protégées de toutes tentatives de piratage.

2.2 Étude de l'existant

L'étude de l'existant est divisée principalement en deux parties. La première partie consiste à analyser les problèmes réels du client. La deuxième partie est la vision technique du problème et la recherche de solutions présentes sur le marché.

Le client dispose actuellement de la méthode suivante : afin de contrôler le département de production le directeur de l'entreprise organise des réunions avec des responsables pour leurs poser des questions. Ces employés passent des jours à collecter les données, chercher la réponse et présenter les résultats dans des rapports écrits. Les données proviennent de plusieurs sources. Ceci engendre souvent l'obtention de résultats contradictoires ou incohérents. Les rapports présentés dans les réunions sont pris en considération dans des stratégies commerciales. Les erreurs commises peuvent donc altérer le développement de l'entreprise. Cette procédure est lente et coûte beaucoup d'argent. En effet, le directeur doit attendre des jours pour que les employés lui donnent les meilleures descriptions de l'état du département alors que certaines situations nécessitent une intervention immédiate. Il doit aussi payer pour le salaire de tous ses employés qui ont pour mission de rédiger les rapports. De plus, les données présentées dans les rapports sont souvent confidentielles, ce qui risque de dévoiler les propos internes de l'entreprise aux autres concurrents du marché.

Après l'interprétation technique du problème, on a constaté que la meilleure solution est d'implémenter un chatBot. En effet, l'agent conversationnel offre une expérience utilisateur plus intéressante que dans les applications web et mobile comme Amazon [1] et Veepee [18] où l'utilisateur doit visiter plusieurs pages et appuyer sur plusieurs boutons pour chercher une information. Dans un chatBot, il a seulement une seule page principale de dialogue et peut écrire directement sa question pour obtenir la réponse en quelques secondes. Il existe plusieurs plateformes sur le Cloud qui offrent les outils nécessaires pour construire cette solution. Dans la liste suivante, on cite les quatre plateformes les plus utilisées dans le monde [9] :

- * IBM Watson Assistant
- * Google Dialogflow
- * Amazon Lex
- * Microsoft Bot Framework

On décrit chacun des plateformes dans les tableaux [2.2](#) et [2.2](#).

Logo	Description
	<p>Watson Assistant est un outil de création de chatBot personnalisé pour les entreprises. Ce service, qui permet aux utilisateurs d'accéder à Watson AI, est fourni via le Cloud IBM et utilise l'apprentissage automatique (machine learning) de Watson AI et la compréhension en langage naturel (NLU). Cette plate-forme a été créé en 2012 et est le résultat du projet de recherche IBM de 2006 qui a remporté le fameux défi « Jeopardy ! Grand » en 2011.</p>
 Dialogflow	<p>Dialogflow est une plate-forme rachetée par Google en 2016. Elle résout les requêtes des utilisateurs sur une base individuelle en prenant l'historique du chat en contexte et ne suit pas un flux prédéfini. Elle peut être lié à de nombreuses plates-formes telles que Google Assistant, Amazon Alexa et Facebook Messenger.</p>

TABLE 2.1 – Plateformes de développement de chatBot

Logo	Description
 Amazon Lex	<p>Amazon Lex propose un bot construit avec Lex qui peut facilement s'intégrer à l'assistant personnel numérique Alexa, l'assistant à domicile le plus populaire. Les développeurs peuvent exporter leur schéma de chatBot construit sous Lex directement dans Alexa Skills Kit (ASK), en créant une version de leur bot qui peut communiquer par la voix sans aucun échafaudage supplémentaire.</p>
 Bot Framework	<p>Microsoft Bot Framework est une plateforme très performante pour les développeurs à la recherche d'une personnalisation sérieuse et de capacités robustes dans leur chatBot. Elle permet d'intégrer plusieurs Cognitive Services d'Intelligence Artificielle parmi lesquels LUIS (Language Understanding Intelligent Service) qui représente un moteur NLU (Natural Language Understanding). Les développeurs utilisent Azure Bot Service en tant que ressource pour créer, tester, déployer et connecter des robots, le tout à partir du seul emplacement de l'environnement de développement intégré (IDE) de Microsoft.</p>

TABLE 2.2 – Plateformes de développement de chatBot (suite)

Travaux connexes

Dans l'article [17], Microsoft suggère quatre scénarios principaux à construire pour les développeurs. Ces scénarios peuvent se résumer dans la liste suivante :

- Cortana skills bot : cette application intègre la puissance de Cortana en tant qu'interface de communication agréable permettant à l'utilisateur de parler directement au bot. Ce dernier tire également parti de la connectivité de Cortana aux différentes applications Windows sur le PC ou sur le mobile et en obtient les mises à jour. Le flux logique dans ce cas est décrit dans le scénario que montre la figure 2.1.

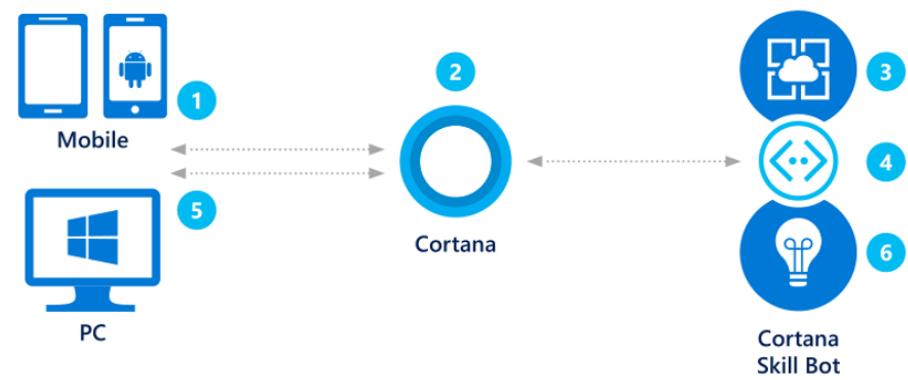


FIGURE 2.1 – Logique de flux de Cortana skills bot [17]

- Enterprise productivity bot : l'objectif principal de ce bot est d'accroître la productivité des organisations en intégrant des services de productivité tels que Office 365 et Dynamics CRM. Grâce à ce scénario, les employés d'une organisation commerciale peuvent accéder aux informations du client à l'aide de commandes simples et vérifier facilement les rendez-vous. Le flux logique dans ce cas est décrit dans le scénario présent dans la figure 2.2.

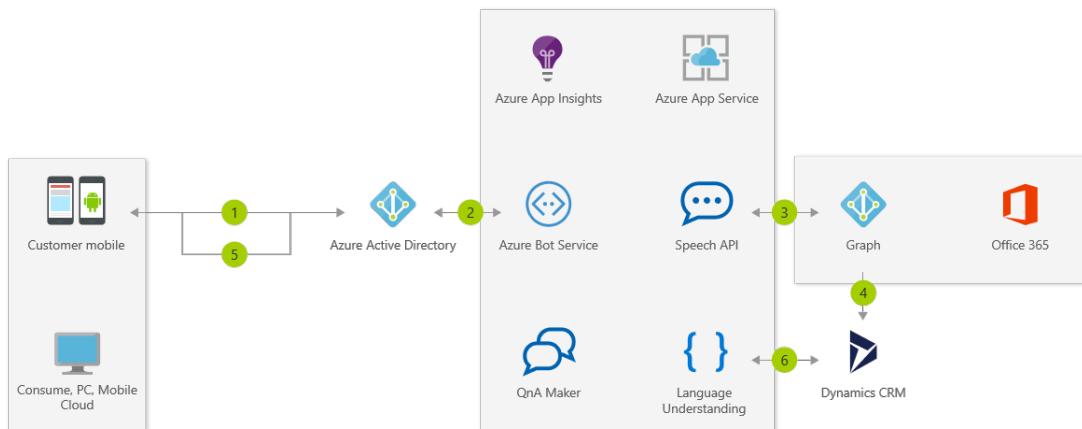


FIGURE 2.2 – Logique de flux de enterprise productivity bot [17]

Le service d'authentification d'Azure, Azure Active Directory valide l'identité de l'employé. Le bot récupère le calendrier Office 365 de l'employé via Microsoft Graph. À l'aide des données collectées dans le calendrier, le bot accède aux informations dans Dynamics CRM, qui sont renvoyées à l'employé pour filtrer les données.

- Internet Of Things (IoT) bot : avec ce bot, il est possible d'accéder, de contrôler et de surveiller des appareils à distance à l'aide de commandes de discussion vocales ou interactives. Le scénario du flux logique dans ce cas est décrit dans le figure 2.3.

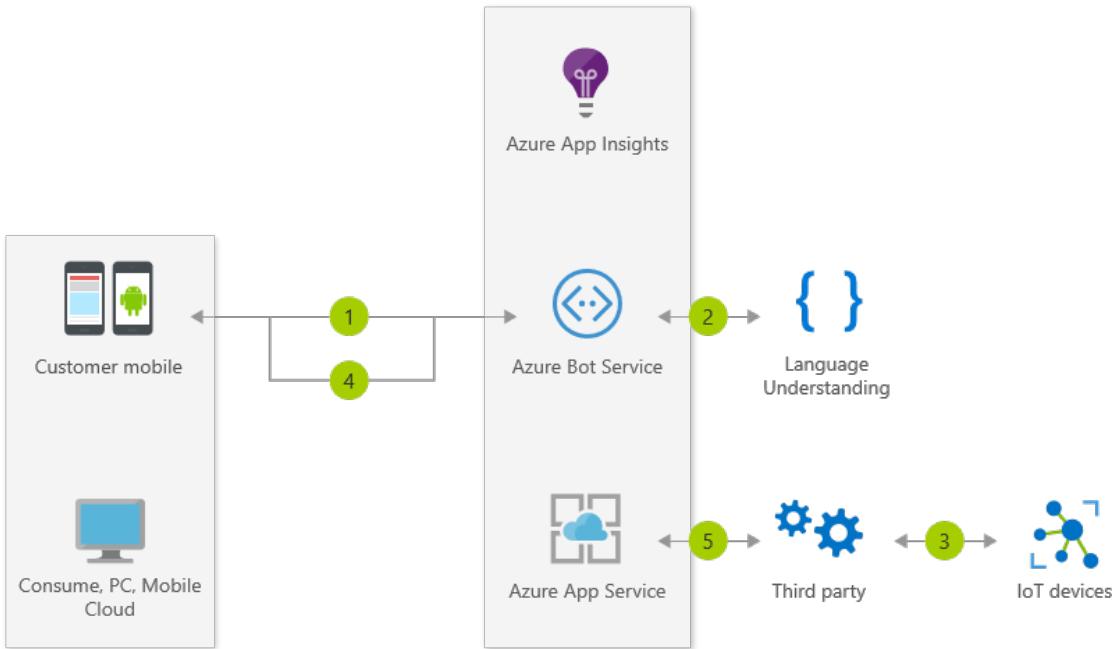


FIGURE 2.3 – Logique de flux de IoT bot [17]

L'utilisateur se connecte à Skype pour accéder au bot IoT en utilisant sa voix et demande d'allumer les lumières via l'appareil IoT. La demande est relayée vers un service tiers ayant accès au réseau de périphériques IoT et les résultats sont renvoyés à l'utilisateur.

- Information bot : c'est le scénario le plus populaire. Il est utilisé dans de nombreux cas, tels que les Foire aux questions (FAQ) et les supports informatiques. Un bot d'information a pour rôle de répondre aux questions posées en se basant sur un ensemble de connaissances.
Les informations sont souvent enregistrées dans des Data Stores structurés tels que SQL Server, qui peuvent être facilement retrouvées via une recherche. Grâce à des Cognitive Services comme QnA Maker ou Azure Search, ces recherches sont possibles avec des commandes de conversation.

Le flux logique dans ce cas est décrit dans la figure 2.4.

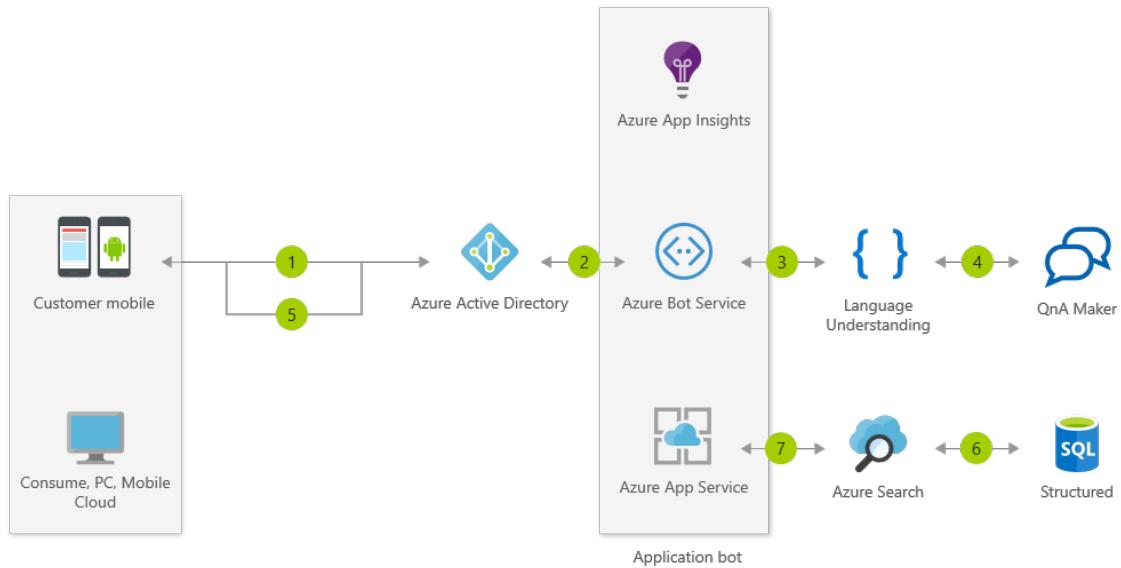


FIGURE 2.4 – Logique de flux de Information bot [17]

Pour cette architecture, un utilisateur recherche des informations spécifiques relatives à des clients spécifiques. À l'aide de QnA Maker, l'utilisateur dispose d'un ensemble d'options de recherche valables, telles que : rechercher un client, passer en revue sa dernière commande, etc. Avec le format QnA défini, l'utilisateur pose facilement des questions qui appellent par la suite Azure Search. Ce service de recherche permet d'extraire des informations stockées dans une base de données SQL.

Le chatBot réalisé est conçu à l'aide de l'architecture de Information bot. Dans la suite nous allons présenter des exemples de projets qui ont été réalisés avec cette architecture. Ces bots sont cités par Microsoft dans son site web "Témoignages Clients" [13] :

- * Guide touristique de NAVITIME JAPAN
- * Assistant technique virtuel de Fiducia & GAD IT AG

Ces projets sont résumés dans le tableau 2.2.

Logo	Description
	<p>La société de technologie de navigation NAVITIME JAPAN [17] est l'un des exemples les plus populaires. Cette société a reconnu la nécessité d'aider les visiteurs à trouver des attractions touristiques sans avoir à comprendre le japonais. Pour aider les voyageurs à trouver leur chemin, la société basée à Tokyo a utilisé Microsoft Bot Framework pour incorporer un bot appelé MIKO dans son application logicielle touristique, permettant aux visiteurs d'obtenir des réponses en temps réel à leurs questions de voyage pendant qu'ils explorent le Japon.</p>
	<p>Fiducia & GAD IT AG [15] est une autre réussite pour le Information Bot. C'est une entreprise allemande qui a adopté la plate-forme Microsoft AI pour développer rapidement un système de support conçu comme un bot pour ses employés. En cinq semaines, ils ont été en mesure de publier un Produit Minimum Viable (MVP) utilisant de nombreuses bases de connaissances existantes. Ils ont été en mesure d'alléger la charge du centre d'appels, de suivre les problèmes et les solutions, et de fournir un meilleur service client.</p>

TABLE 2.3 – Projets de Information Bot

Les deux exemples précédents sont les plus proches de l'application réalisée dans ce projet. Il existe un autre exemple cité par Microsoft comme étant l'un des projets d'Information Bot mais n'utilise pas tous les services cités dans la figure 2.4. Ce projet est appelé CHIP [16]. Il a été développé en utilisant seulement le service QnA Maker en ajoutant des centaines de question et réponses. Il a été intégré afin d'aider les citoyens de Los Angeles de s'informer sur les services du gouvernement de la ville. Les résidents peuvent naviguer dans les ressources du gouvernement ce qui permet les employés de se libérer des tâches répétitives et de se mieux concentrer sur des

opérations plus importantes. Ce chatBot est disponible pour tous les visiteurs du site web officiel [10] de la ville de Los Angeles.

La figure 2.5 présente le message d'accueil de ce chatBot.

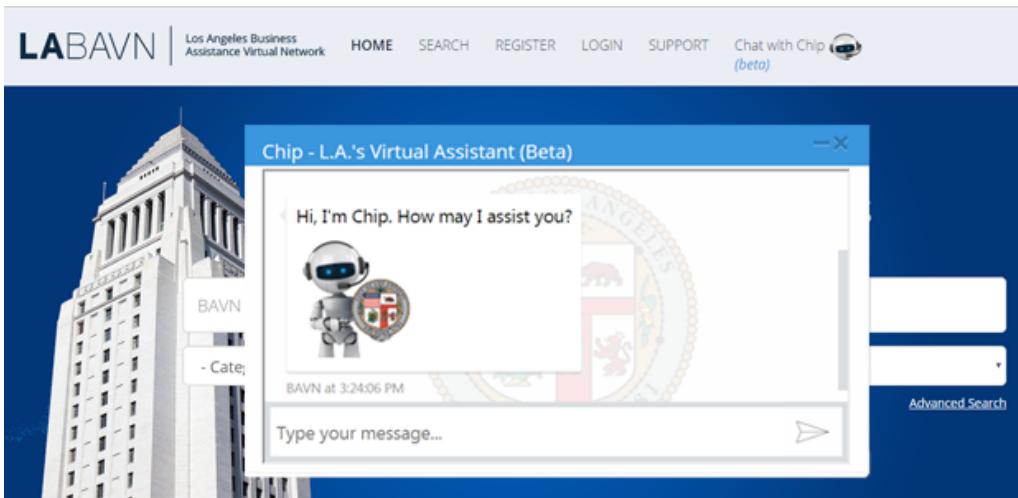


FIGURE 2.5 – Site web officiel du gouvernement de Los Angeles

2.3 Solution proposée

La solution proposée est développée à l'aide des technologies Microsoft étant donné que la société est un partenaire Gold de Microsoft et a accès aux différents services proposés par la plateforme de Cloud Computing Microsoft Azure. Les avantages de l'utilisation de cette plate-forme sont résumés comme suit :

- création de chatBots avec des templates out-of-the box personnalisées selon les exigences du l'utilisateur
- insertion d'autres services disponibles dans le Cloud comme Cognitive Services
- possibilité de publication du chatBot dans différents canaux comme Facebook Messenger, Slack, Skype, WhatsApp et Twitter
- accès aux services d'infrastructure de stockage comme Azure SQL Database et Azure Blob Storage
- sécurité d'environnement de développement qui procure l'authentification de l'accès aux données en Back-end

Ce projet représente une solution alternative au processus métier existant en offrant les avantages suivants :

- Rentabilité : le chatBot est un investissement unique qui permet d'économiser les coûts mensuels des salaires des employés.
- Rapidité : ce chatBot fera gagner beaucoup de temps à la société car il automatisera la communication entre les clients et la société. En outre, ils n'auront pas besoin de passer beaucoup de temps à surfer ici et là car les informations sont disponibles du bout des doigts, directement dans la fenêtre de discussion.
- Automatisation des mises à jour : les clients auront accès aux dernières mises à jour sur les produits et les services disponibles.
- Bonne expérience utilisateur : grâce à une interface agréable et l'apparence attrayante des canaux de discussion populaires.

- Disponibilité : crée un service de communication disponible 24h / 24 et 7j / 7.

2.4 Analyse Globale

2.4.1 Diagramme de cas d'utilisation

Les diagrammes de cas d'utilisation sont des diagrammes utilisés pour donner une vision globale du comportement fonctionnel d'un système. Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet. Maintenant que nous avons identifié les acteurs dans le paragraphe 2.1.2, nous allons pouvoir les représenter graphiquement sur un diagramme de cas d'utilisation, dont la notation graphique de base est montrée dans la figure 2.6.

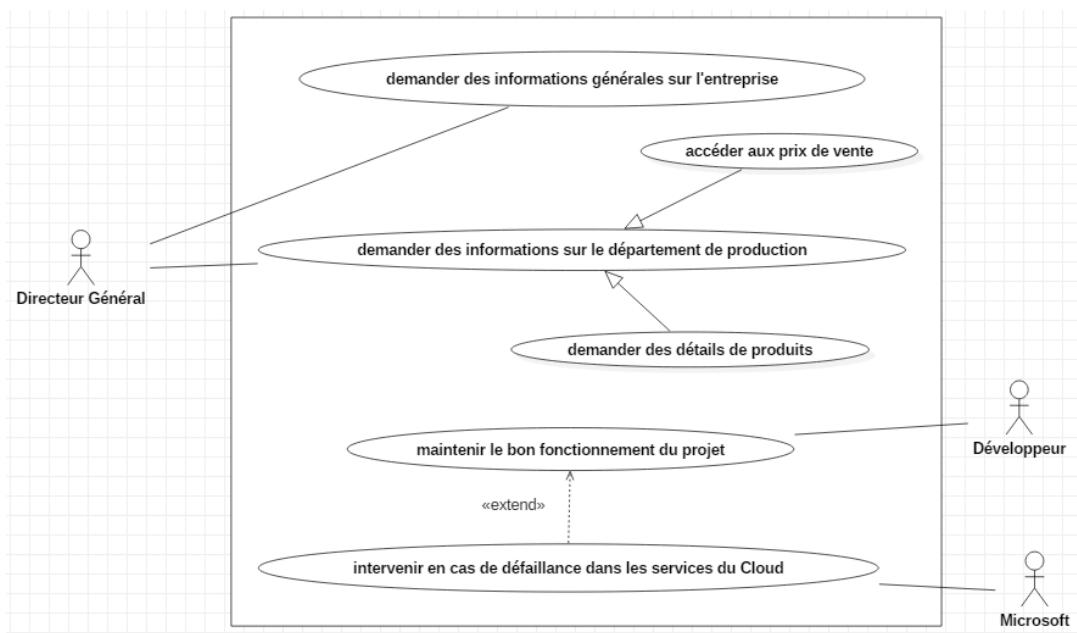


FIGURE 2.6 – Diagramme de cas d'utilisation

Affectation des priorités

Les cas d'utilisation peuvent être classés selon leur ordre d'importance pour chacun des acteurs. Ce classement donne lieu à la définition d'un ordre de priorité pour les cas d'utilisation. Dans notre cas, les cas d'utilisation qui s'avèrent les plus prioritaires ont la priorité la plus forte « 1 » et les moins prioritaires ont la priorité « 2 ». Ceci est représenté dans le tableau 2.6.

Cas d'utilisation	Acteur	Priorité
Demander des informations sur le département de production	Directeur	1
Demander des informations générales sur l'entreprise	Directeur	1
Maintenir le bon fonctionnement du projet	Développeur	2

TABLE 2.4 – Tableau d'affectation des priorités des cas d'utilisations

Raffinement du cas d'utilisation « Demander des informations sur le département de production »

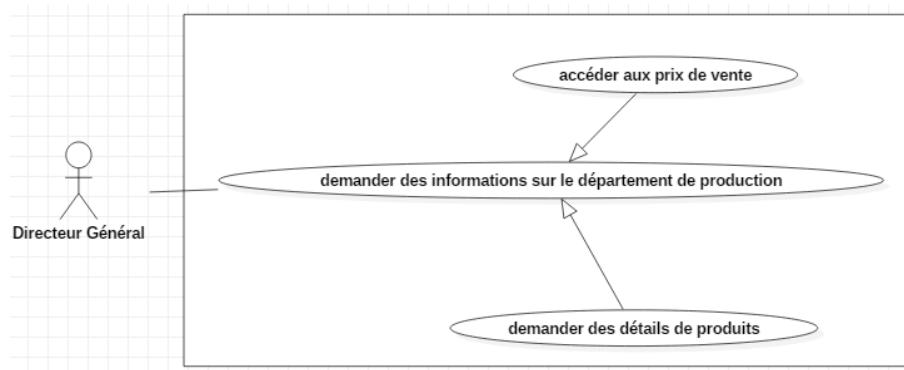


FIGURE 2.7 – Raffinement CU « Demander des informations sur le département de production »

Acteur : Directeur	CU : Demander des informations sur le département de production
Pré-condition	L'utilisateur doit être connecté au bot
Post-condition	Suggestion de nouvelle recherche à l'utilisateur
Scénario nominal	1. le bot affiche un message de bienvenue 2. l'utilisateur saisie sa question 3. les services d'analyse linguistique et de recherche sont appelés 4. le résultat de la requête est affiché. Les prix de vente et/ou des détails des produits sont présentés
Exception	Un message d'erreur est affiché dans le cas échéant

TABLE 2.5 – Description textuelle du CU Demander des informations sur le département de production

Raffinement du cas d'utilisation « Demander des informations générales sur l'entreprise »

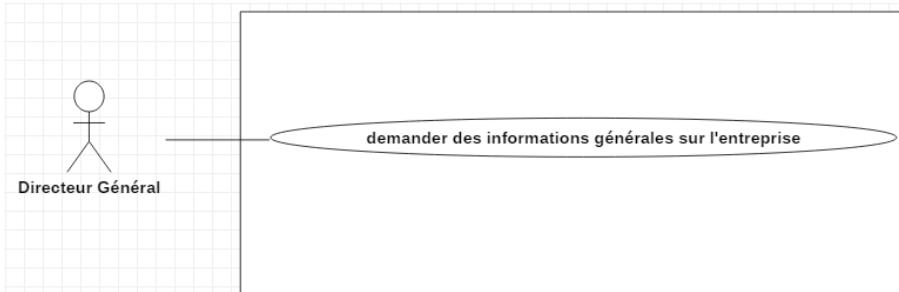


FIGURE 2.8 – Raffinement CU « Demander des informations générales sur l'entreprise »

Acteur : Directeur	CU : Demander des informations générales sur l'entreprise
Pré-condition	L'utilisateur doit être connecté au bot
Post-condition	Suggestion de nouvelle question à l'utilisateur
Scénario nominal	1. le bot affiche un message de bienvenu 2. l'utilisateur saisie sa question 3. les services d'analyse linguistique et de réponses prédéfinies sont appelés 4. la réponse prédéfinie est affichée
Exception	Un message d'erreur est affiché dans le cas échéant

TABLE 2.6 – Description textuelle du CU Demander des informations sur le département de production

2.4.2 Diagramme de classes

Le diagramme de classe (voir figure 2.9) présente les différents modèles utilisés au cours du développement de l'application. Il est composé des classes suivantes :

- * Product : cette classe contient les différentes informations sur les produits. Elle est reliée à la classe de SellDate et Category. Elle contient les attributs suivants :
 - ProductID : identifiant
 - Name : nom
 - StandardCost : prix moyen de vente
 - Description : description détaillée du produit
 - Color : couleur
 - Size : dimensions (largeur, hauteur...)
 - Weight : poids en grammes ou kilogrammes selon le produit
- * SellDate : elle contient les dates de début (SellStartDate) et de fin de production (SellEndDate) pour chaque produit
- * Category : cette classe contient les noms des types de produits (name)
- * Dialog : c'est une classe instanciée à chaque fois que l'utilisateur démarre une discussion avec le chatBot. Elle fait appel à des méthodes nécessaires pour le fonctionnement des services.

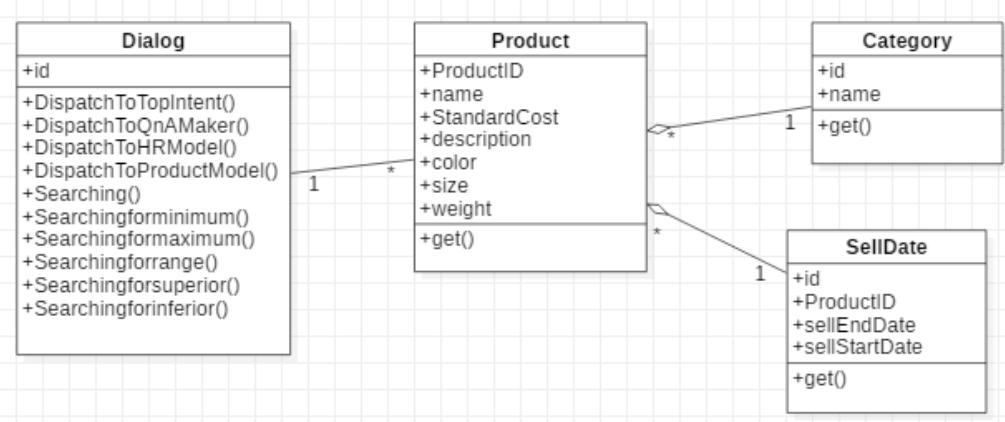


FIGURE 2.9 – Diagramme de classes du chatBot

Les classes Product, Category et SellDate représentent la base de données du département de production de l’entreprise. Elles sont enregistrées dans une base SQL à laquelle se connecte le chatBot. En effet, la recherche de réponse à la question de l’utilisateur se base sur cette base de données. La classe Dialog représente la discussion démarrée après la connexion de l’utilisateur au chatBot. Pour chaque connexion un nouvel identifiant ID est associé à la discussion et les méthodes nécessaires au fonctionnement du chatBot sont appelées.

2.4.3 Architecture fonctionnelle

L’application se compose essentiellement d’un point de vue fonctionnel de trois modules (voir figure 2.10). Chaque module est responsable de l’exécution d’une phase importante du fonctionnement du chatBot.

1. Le premier module est l’interface de communication entre l’utilisateur et le chatBot. Dans ce module l’interface reçoit le message de l’utilisateur et le transmet à la partie d’analyse linguistique.
2. Une fois la question est détectée, le module d’analyse linguistique traite le message et en extrait une intention et les mots clés appelés entités.
3. Après la compréhension du message de l’utilisateur, le module de recherche intervient. Il reçoit l’intention et les mots clés et lance une recherche. Cette recherche est divisée en deux types selon l’intention détectée :
 - Si l’intention correspond à la demande d’une information générale sur l’entreprise ou à salutation le recherche est dans ce cas dirigée vers une base de connaissance contenant des réponses statiques et prédéfinies.
 - Sinon, la recherche est dirigée vers une base SQL indexée où les informations détaillées sur l’entreprise sont enregistrées.

Une fois la recherche est terminée, le module collecte les informations nécessaires pour la construction de la réponse et les envoie au module d’interface de communication. Ce dernier affiche la réponse à l’utilisateur.

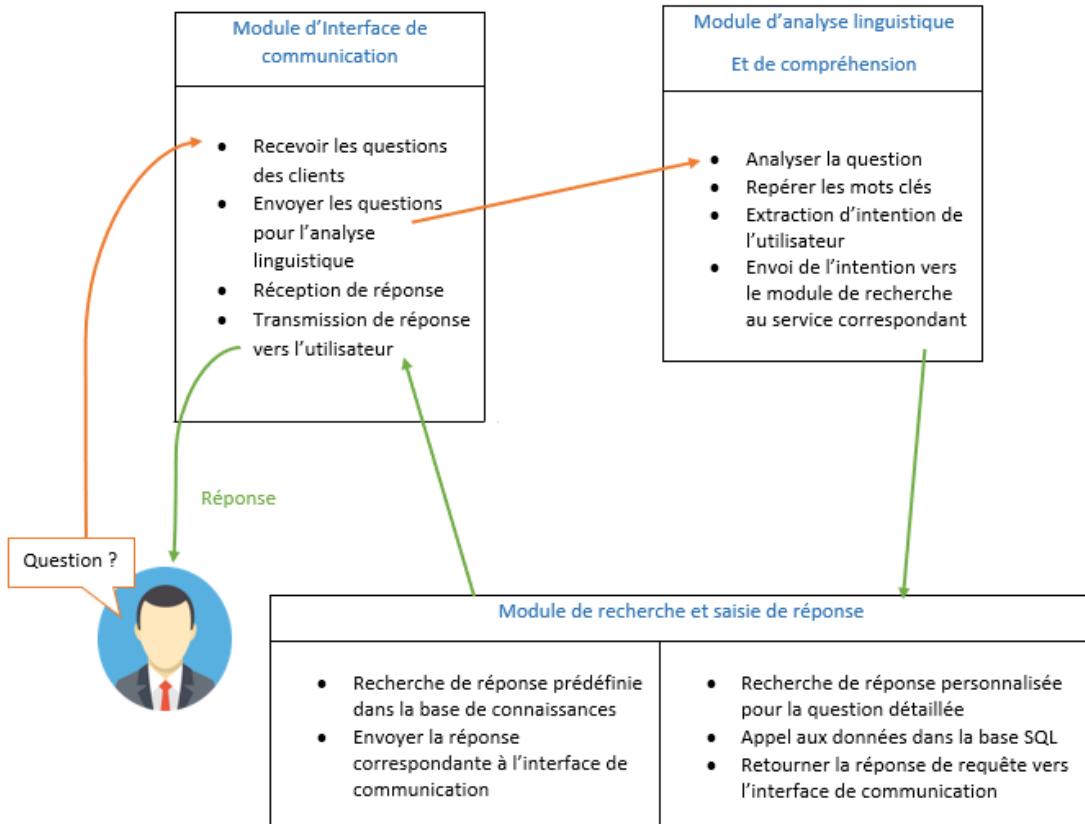


FIGURE 2.10 – Architecture fonctionnelle du chatBot

2.5 Conclusion

Ce chapitre nous a permis de bien comprendre les insuffisances du système actuel établi chez le client. L'analyse effectuée nous a ouvert la voie au processus de conception d'une nouvelle solution. Nous avons étudié les solutions existantes présentes sur le marché. Nous avons décrit par la suite la conception générale de l'alternative proposée. Dans le chapitre suivant nous allons aborder de façon approfondie le premier module du projet qui constitue l'interface de communication avec l'utilisateur.

Chapitre 3

Module Azure chatBot

Comme décrit dans la partie 2.4.3 d'architecture fonctionnelle du chapitre précédent, le module d'Azure chatBot constitue une interface intermédiaire qui sert d'une part à lier entre les services d'analyse linguistique, de recherche et l'utilisateur d'autre part. Dans ce chapitre on va présenter les technologies employées afin de développer ce module, l'architecture et la conception adaptées et on finira par tester ses fonctionnalités dans une démo.

3.1 Conception préliminaire et détaillée

3.1.1 Architecture logique

L'architecture logique de ce module modélise les éléments principaux dans le fonctionnement de l'interface ainsi que le flux de communication entre eux. Il est présenté dans le schéma de la figure 3.1.

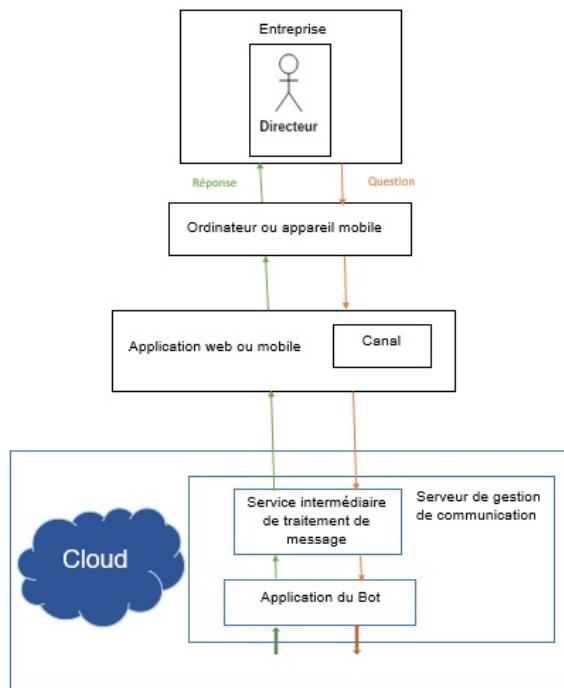


FIGURE 3.1 – Architecture logique de la communication avec l'utilisateur

3.1.2 Diagramme de séquence

L'interface Utilisateur d'un chatBot est différente de celle des sites web et mobile. Elle se distingue par l'emploi de messages au lieu des pages web et garde une page unique où l'utilisateur peut communiquer. De plus, la discussion s'effectue à travers un canal qui constitue un intermédiaire entre l'utilisateur et le chatBot. Toute interaction est appelée une activité. Dans ce cas, le bot doit être en mesure de gérer les flux de conversation afin de garder une bonne expérience utilisateur. Ce flux est généré dès que l'utilisateur se connecte au canal et se termine après l'envoi de la réponse.

Après la connexion de l'utilisateur au canal, le service intermédiaire envoie une requête HTTP POST contenant une activité de type mise à jour de conversation (Conversation Update). Cette requête informe l'application que l'utilisateur s'est connecté. Le bot confirme la réception de la requête en générant un code d'état HTTP 200 et rejoint de son coté la conversation. Ensuite, l'utilisateur écrit un message dans le canal « Bonjour ».

Ce message est transmis sous forme de requête HTTP POST qui a pour activité Message. Le bot reçoit le message, fait appel au service nécessaire et répond par une requête HTTP POST contenant la réponse « Salut ». La réponse transmise est affichée à la page du canal. Le canal confirme la réception de réponse en envoyant un code d'état HTTP 200. La discussion s'effectue donc tour à tour afin de garder un flux logique et éviter toute sorte de confusion possible.

Dans ce cadre, l'application développée communique avec l'utilisateur selon une logique modélisée dans le diagramme de séquence dans la figure 3.2.

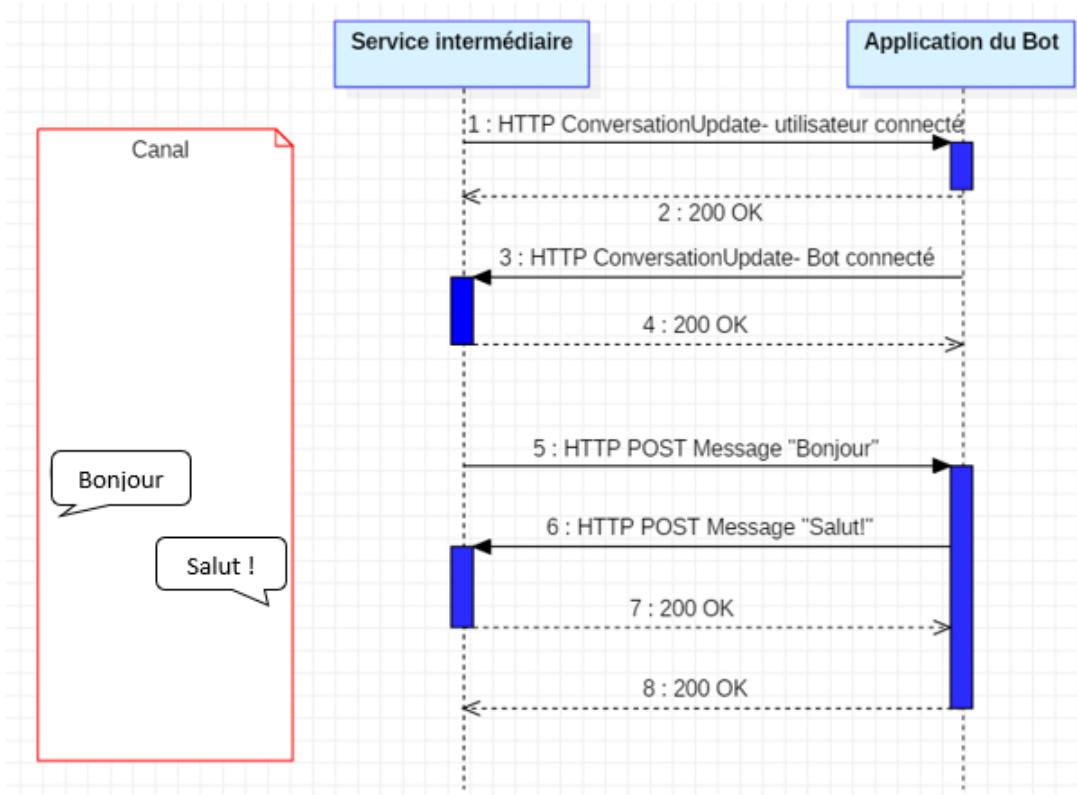


FIGURE 3.2 – Diagramme de séquence de la communication avec l'utilisateur

3.2 Branche technique

3.2.1 Outils et technologies

Les outils utilisés se résument dans la liste suivante :

- * CSharp [19] : c'est un langage de programmation orientée objet, commercialisé par Microsoft depuis 2002 et destiné à développer des applications sur la plateforme .NET.
- * ASP.NET Core [21] : c'est un Framework libre et open-source conçu pour les développeurs de langage CSharp. La version utilisée dans l'application est ASP.NET Core 2.2.
- * Microsoft Visual Studio [20] : c'est un environnement de développement intégré conçu par Microsoft afin de développer des applications web ASP.NET, des services web XML, des applications bureautiques et mobiles. La dernière version s'appelle Visual Studio 2019 et est utilisée pour le développement du chatBot.
- * Microsoft Bot Framework SDK [2] : c'est un Software Development Kit multilingue pour la création de bots utilisant CSharp, Java, Python et JavaScript. Il offre une architecture extensible permettant aux développeurs de choisir des composants spécifiques fournis, ainsi que de tirer parti d'un écosystème riche d'extensions qui peuvent leurs aider dans des tâches telles que la traduction automatique, la gestion de dialogue et la planification des réunions. La version de SDK utilisée dans cette application est la Version 4.
- * Azure Web App Bot [3] : ce service est disponible dans la plateforme du Cloud d'Azure. Il offre la capacité de lier l'application du chatBot à plusieurs canaux de chat comme Slack et Facebook Messenger. Il transmet le message de l'utilisateur du canal vers l'application du chatBot déployée sur le Cloud de Microsoft Azure.
- * Azure Bot Services [3] : c'est un ensemble d'outils qui permettent de créer, tester, déployer et gérer des bots intelligents. L'infrastructure modulable et extensible fournie par le Kit de développement logiciel (SDK), les outils, les modèles et les services d'intelligence artificielle permettent aux développeurs de créer des bots en mesure d'offrir des fonctionnalités vocales, de comprendre le langage naturel et de traiter les questions-réponses.
- * Bot Framework Emulator [8] : c'est une application de bureau qui permet aux développeurs de robots de tester et déboguer leurs bots localement ou à distance. Cet émulateur permet de discuter avec le bot et d'inspecter les messages qu'il envoie et reçoit.

3.2.2 Architecture physique

L'architecture physique de ce module présente une version plus détaillée par rapport à l'architecture logique 3.1.1. Elle montre le nom des outils employés pour chaque partie. Le flux de communication dans ce module est décrit dans les étapes suivantes :

1. Le directeur se connecte au chatBot à travers le canal.
2. Le chatBot confirme la connexion et rejoint la discussion à travers Azure Web App Bot.

3. Le chatBot envoie un message d'accueil à l'utilisateur.
4. Le directeur envoi une question en langage naturel.
5. Le message est transmis à Azure Web App Bot. Ce dernier s'occupe de la conversion de la question sous format JSON compatible.
6. L'application du bot reçoit l'objet JSON et fait appel aux autres serveurs du Cloud en leurs transmettant une requête contenant cet objet.
7. Le chatBot reçoit une réponse sous forme de données structurées dans un objet JSON qu'elle envoie vers le canal à travers Azure Web App Bot.
8. Le canal reçoit la réponse et l'affiche sur l'écran de l'ordinateur sous forme lisible par l'utilisateur.

La liaison entre les composants de ce modèle est représentée dans le schéma 3.3.

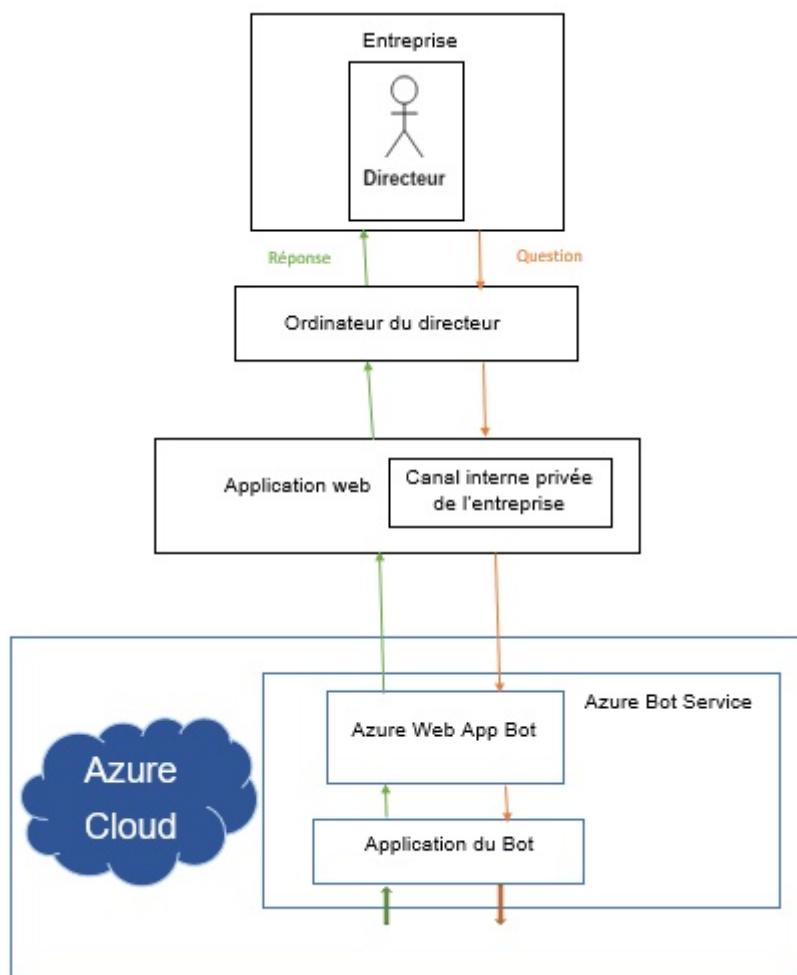


FIGURE 3.3 – Architecture physique de la communication avec l'utilisateur

3.2.3 Diagramme de composants

Le diagramme de composants (voir figure 3.4) suivant montre la structure des fichiers responsables au fonctionnement de ce module. Après la connexion de l'utilisateur au canal, le bot réalise l'instanciation de ses services. Le fichier principal

de l'application appelé EntrepriseBot.cs fait appel au fichier Startup.cs. Ce fichier reçoit l'identifiant de l'application de AppSettings.json et les clés du Web App Bot Service du fichier EntrepriseBot.bot. Ensuite, le fichier principal reçoit de Startup.cs un objet ConfigureServices contenant une instance du service Web App Bot. Cette instance est utilisée pour répondre au message envoyé par l'utilisateur.

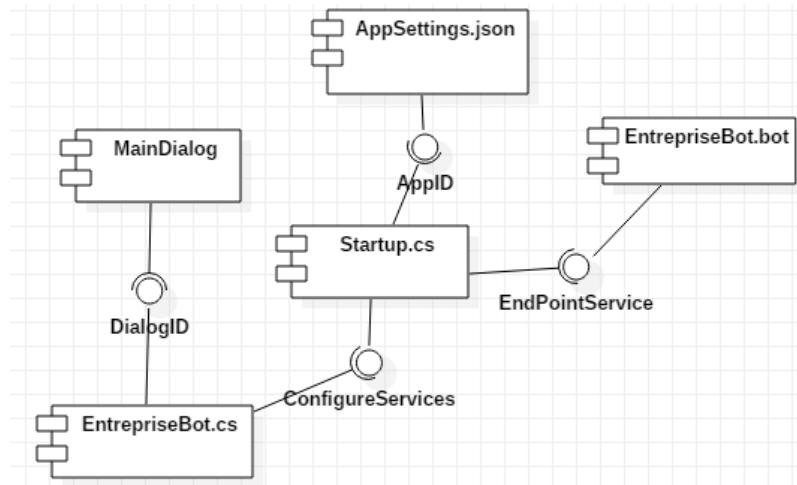


FIGURE 3.4 – Diagramme de composants de la communication

3.3 Tests et codage

Comme modélisé dans le diagramme de composants précédent le flux de communication entre l'utilisateur et le bot est géré par le fichier principal `EntrepriseBot.cs`. Dans la figure 3.5, le bot envoie un message d'accueil à l'utilisateur dès qu'il se connecte à l'application.

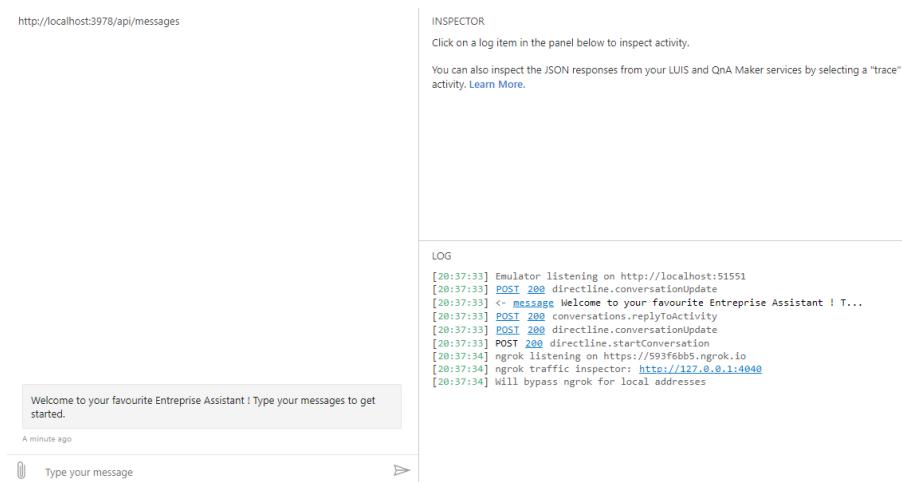


FIGURE 3.5 – Message d'accueil de connexion

La partie de LOG dans la figure présente les étapes par lesquels passe le flux de communication entre l'application du bot et Azure Web App Service. Ces étapes sont ordonnées comme décrit précédemment dans le diagramme de séquence de la figure 3.2. La connexion de l'utilisateur est détectée en affichant un message

"ConversationUpdate". Ensuite, le bot rejoint la discussion et envoie un message d'accueil. Le débogage du message d'accueil affiché donne les informations en objet JSON présentées dans la figure 3.6.

The screenshot shows a browser developer tools window with two panes. The left pane displays a message card with the text "Welcome to your favourite Entreprise Assistant ! Type your messages to get started." and a timestamp "3 minutes ago". Below the card is a text input field with the placeholder "Type your message" and a send button. The right pane is divided into two sections: "INSPECTOR - JSON" and "LOG". The "JSON" section shows a complex nested JSON object representing the message. The "LOG" section shows a series of API requests and responses, including "Emulator listening on http://localhost:51551", "POST 200 directline.conversationUpdate", and several "POST 200 conversations.replyToActivity" entries.

```

http://localhost:3978/api/messages
INSPECTOR - JSON
{
  "channelId": "emulator",
  "conversation": {
    "id": "50193170-8953-11e9-821d-731380c14ec0|livechat"
  },
  "from": {
    "id": "501539d0-8953-11e9-91eb-3b9dfbb56008",
    "name": "Bot",
    "role": "bot"
  },
  "id": "508f72e0-8953-11e9-821d-731380c14ec0",
  "inputHint": "acceptingInput",
  "localTimestamp": "2019-06-07T20:37:33+02:00",
  "locale": "en-US",
  "recipient": {
    "id": "e1f6123-37e9-4ac6-9070-e8b51502e4fe",
    "role": "user"
  },
  "replyToId": "50615e00-8953-11e9-821d-731380c14ec0",
  "serviceUrl": "http://localhost:51551",
  "text": "Welcome to your favourite Entreprise Assistant ! Type your messages to get started.",
  "timestamp": "2019-06-07T18:37:33.837Z",
  "type": "message"
}

LOG
[20:37:33] Emulator listening on http://localhost:51551
[20:37:33] POST 200 directline.conversationUpdate
[20:37:33] <- message Welcome to your favourite Entreprise Assistant ! T...
[20:37:33] POST 200 conversations.replyToActivity
[20:37:33] POST 200 directline.conversationUpdate
[20:37:33] POST 200 directline.startConversation

```

FIGURE 3.6 – Capture d'écran de débogage du message d'accueil

L'objet JSON affiché dans la figure représente le contenu de l'activité de type Message. Cette activité est composée d'informations suivantes :

- l'émetteur connecté : le "Bot" qui a pour rôle "bot"
- le receveur connecté : l'utilisateur qui a pour rôle "User"
- le message envoyé avec la date et l'heure d'envoi

3.4 Conclusion

Dans ce chapitre on a pu comprendre le fonctionnement du premier module avec détails grâce à la présentation de l'architecture logique physique et les différents diagrammes de classe, de séquence et de composants. On a clôturé cette modélisation par un test de code. Dans le chapitre suivant, nous aborderons le mode de fonctionnement de l'analyse linguistique du message en langage naturel.

Chapitre 4

Module d'analyse linguistique

Le module est une interface intermédiaire qui sert à lier entre le module Azure chatBot et le module de recherche. Dans ce chapitre on présente les technologies employées afin de développer ce module, la conception adaptée et on teste ses fonctionnalités dans une démo.

4.1 Conception préliminaire et détaillée

4.1.1 Architecture logique

L'architecture logique de ce module modélise les éléments principaux dans l'analyse du message textuel et sa compréhension ainsi que la liaison entre eux. Il est représenté dans le schéma de la figure 4.1.

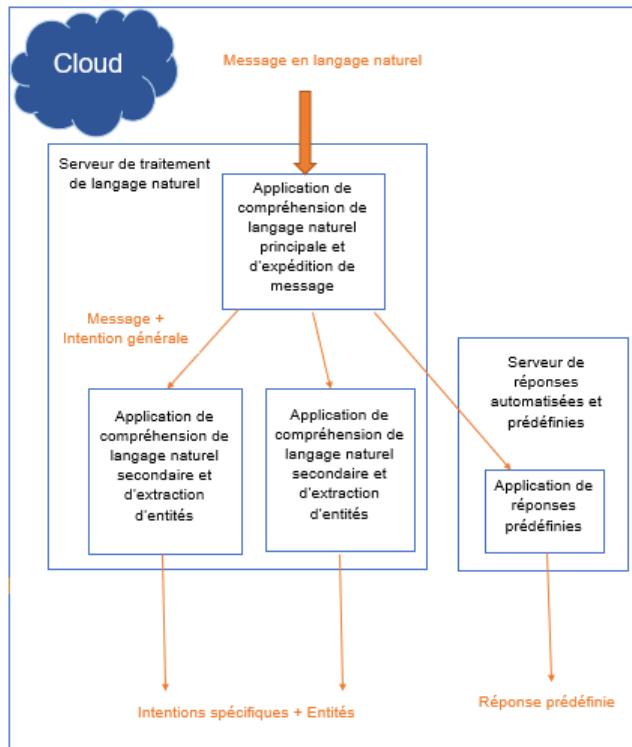


FIGURE 4.1 – Architecture logique d'analyse linguistique

4.1.2 Diagramme de conception de séquence et de classes

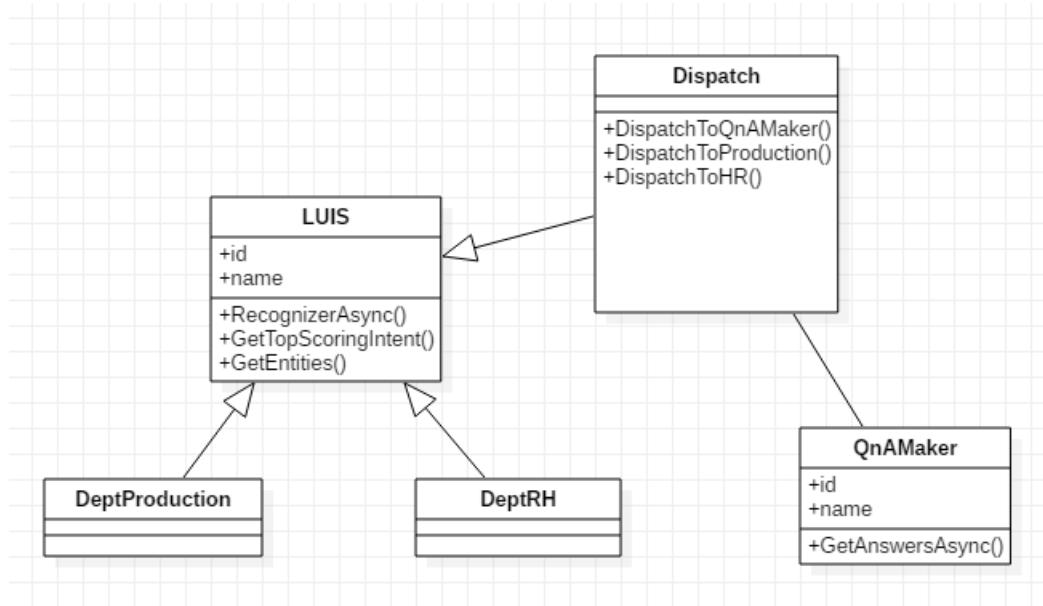


FIGURE 4.2 – Diagramme de classes d’analyse linguistique

Le diagramme de classe présente les différents modèles utilisés au cours du développement de ce module. Il est composé des classes suivantes :

- LUIS : c'est la classe d'analyse linguistique où les classes Dispatch, DeptRH et DeptProduction héritent les attributs et les méthodes. Elle effectue l'extraction de l'intention et les entités pour chaque message reçu.
- Dispatch : c'est la classe de distribution d'intention. Elle est liée au QnAMaker et hérite les attributs et les méthodes de la classe LUIS.
- QnAMaker : cette classe génère les réponses prédéfinies par la méthode GetAnswersAsync().

La conversion du langage naturel du message en une intention spécifique et des entités s'exécute selon les étapes suivantes :

1. l'application du Bot reçoit la question en langage naturel envoyée par l'utilisateur.
2. Par la méthode OnTurnAsync l'application du Bot appelle l'application d'analyse linguistique principale Dispatch.
3. Dispatch exécute l'extraction de l'intention générale avec la méthode RecognizerAsync et envoie le score de prédiction appelé TopScoringIntent au bot.
4. Dispatch appelle à une autre application avec la méthode DispatchTopScoringIntent. Le type d'application appelé varie selon la valeur de l'intention générale :
 - Pour l'intention "QnA" l'application "qna" qui contient des réponses pré-définies est appelée avec la méthode DispatchToQnAMaker.
 - Pour l'intention "DeptProduction" l'application "DeptProduction" est appelée avec la méthode DispatchToProduction.
 - Pour l'intention "DeptRH" l'application "DeptRH" est appelée avec la méthode DispatchToHR.

5. Si l'application DeptRH ou DeptProduction est appelée, il y a extraction d'entités et d'intention spécifique avec les méthodes RecognizerAsync, GetTopScoringIntent et GetEntities. L'intention spécifique et les entités sont envoyées au Bot.
6. Si l'application "qna" est appelée, une réponse prédéfinie trouvée avec la méthode GetAnswersAsync est envoyée au Bot.

Ces étapes décrites sont modélisées dans le diagramme de séquence de la figure 4.3.

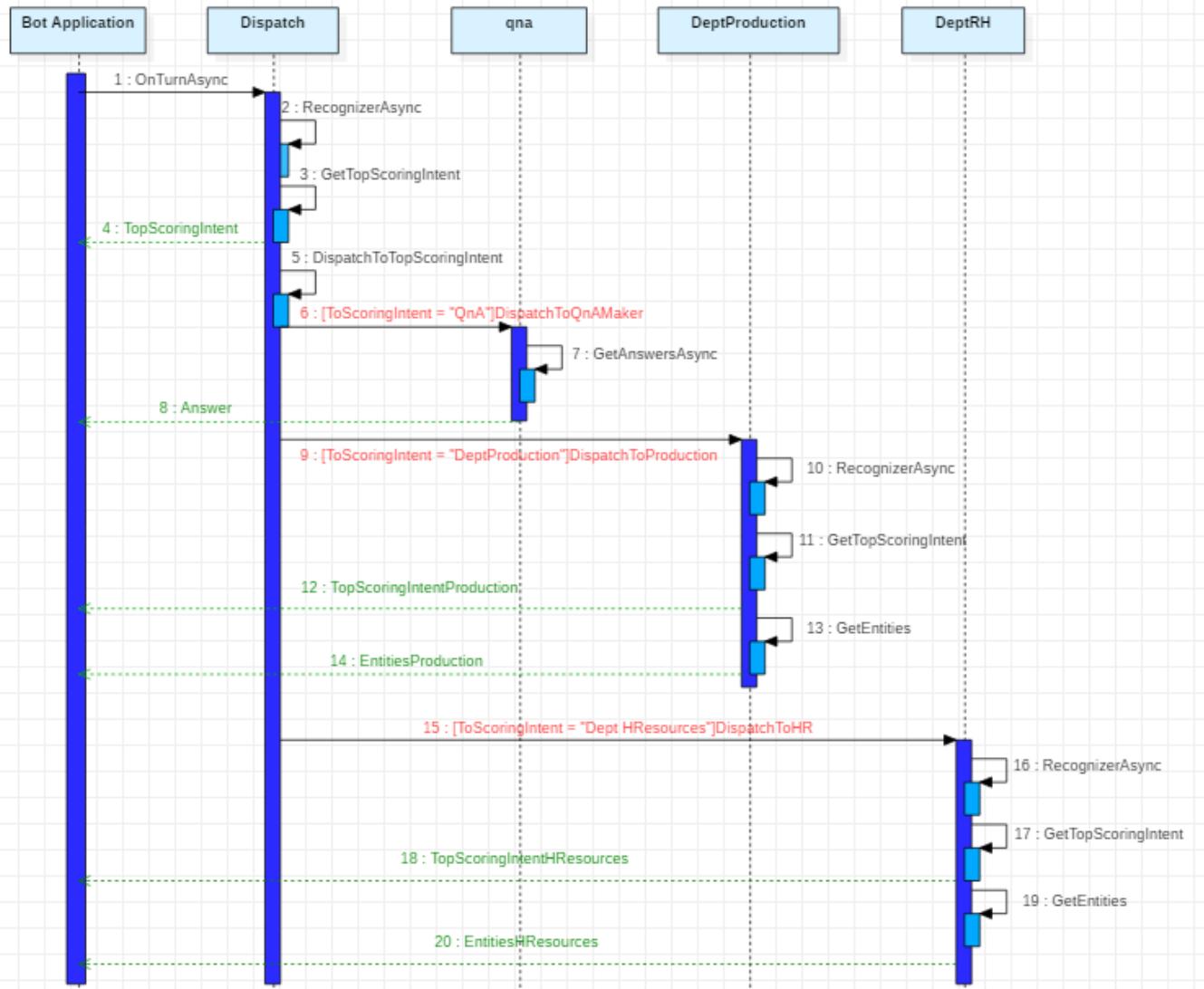


FIGURE 4.3 – Diagramme de séquence d'analyse linguistique

4.2 Branche technique

4.2.1 Outils et technologies

Les outils utilisés se résument dans la liste suivante :

- * Cognitive services [5] : c'est des services provenant d'algorithmes intelligents qui peuvent être intégrés dans un site web ou un bot. Ils permettent de rendre l'application capable de voir, écouter, parler, comprendre et interpréter les besoins des utilisateurs à travers des méthodes naturelles de communication. Microsoft présente les différents services qu'elle offre dans ce cadre dans la figure 4.4

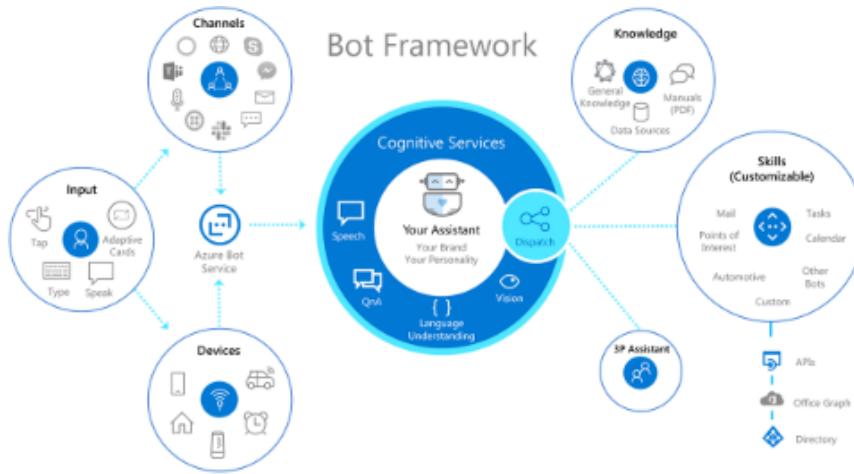


FIGURE 4.4 – Microsoft Cognitive Services

- * LUIS [7] : c'est l'abréviation de Language Understanding Intelligent Service. C'est l'un des Cognitive Services de Microsoft Azure. Ce service permet aux développeurs de construire des applications intelligentes capables de comprendre le langage naturel humain et de répondre aux messages de l'utilisateur.
- * Application LUIS [11] : elle appelée aussi LUIS Model et est définie par le développeur pour un domaine spécifique. Elle a pour output un web service avec une HTTP endpoint auquel le développeur fait référence à partir de l'application client où il veut ajouter la compréhension de langage naturel.
- * Intention [11] : c'est le sens du contenu textuel analysé par l'application LUIS. Elle reflète l'objectif que l'utilisateur veut accomplir. Elle est associée avec un score qui reflète la précision de l'analyse linguistique.
- * Entité [11] : c'est un mot clé détecté par l'application LUIS dans le message envoyé par l'utilisateur. Elle contient souvent une information importante liée à l'intention.
- * Dispatch [14] : c'est une application LUIS de type dispatch. Elle permet d'utiliser plusieurs applications LUIS et QnA Maker à partir d'une application parente à l'aide du modèle de répartition selon la valeur de l'intention générale détectée.
- * QnA Maker [6] : c'est l'un des Cognitive Services de Microsoft Azure. Il permet d'intégrer un service de questions-réponses à partir de contenu semi-structuré, comme des documents de questions fréquentes (FAQ), des URL et des manuels de produit. Le service QnA Maker répond aux questions des utilisateurs en langage naturel en se référant à une base de connaissances contenant les réponses.

4.2.2 Architecture physique

L'architecture physique de ce module indique les éléments principaux implémentés ainsi que le flux de communication entre eux. Le message en langage naturel est transmis au module d'analyse linguistique afin d'en extraire les entités et l'intention spécifique. Cette opération est divisée en deux parties. Dans la première partie, le message est envoyé à une application LUIS appelée Dispatch. Cette application génère une intention générale et fait appel à la deuxième partie du module. Si l'intention générale correspond à une question dans le domaine de Production ou Ressources Humaines, les applications Département de Production ou Département de Ressources Humaines sont appelées pour donner une intention spécifique du domaine choisi et les entités. Si l'intention générale correspond à une question générale sur l'entreprise ou une salutation, l'application QnA du QnA Maker Service est appelée pour donner une réponse prédéfinie. L'architecture physique est représentée dans le schéma de la figure 4.5.

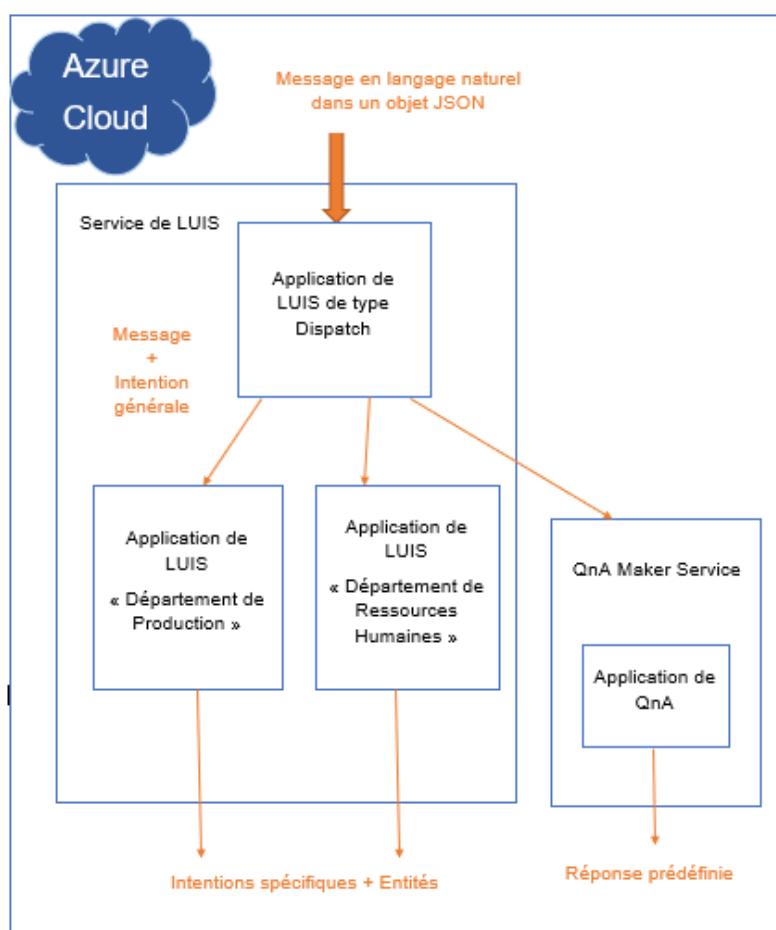


FIGURE 4.5 – Architecture physique d'analyse linguistique

4.2.3 Diagramme de composants

L'instanciation des services d'analyse linguistique se réalise quand l'utilisateur se connecte au bot. Le fichier principal EnterpriseBot.cs fait appel à Startup.cs. Ce fichier reçoit les clés de configuration de chaque service ServicesKey à partir

du fichier EntrepriseBot.bot. Ce fichier contient les clés des autres services de LUIS ProductionKey et HRKey et les clés QnAKeys de QnA Maker. Startup.cs retourne à la fin un objet ConfigureServices qui sera directement utilisé par le fichier principal. Ces liaisons sont modélisées dans la figure 4.6.

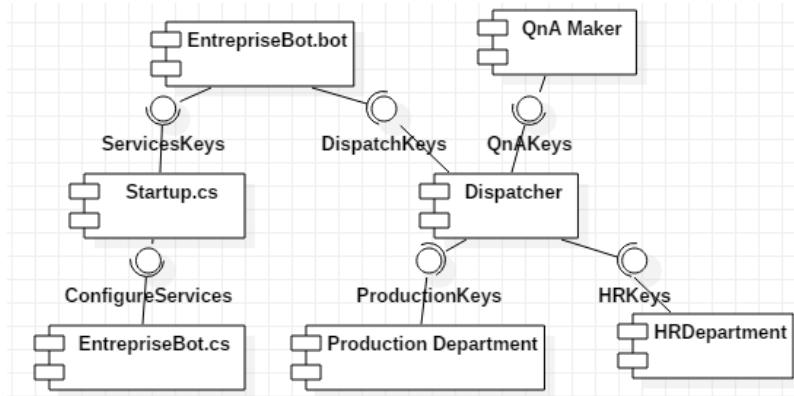


FIGURE 4.6 – Diagramme de composants d'analyse linguistique

4.3 Tests et codage

Le test de ce module est divisé en deux parties. Un test de l'application du service QnA Maker et un test de l'application de LUIS "Département de production". Comme expliqué dans les paragraphes précédents ces deux applications passent par une application LUIS Dispatch qui dirige l'analyse linguistique. La figure 4.7 présente le test du bot en envoyant le message "Bonjour".

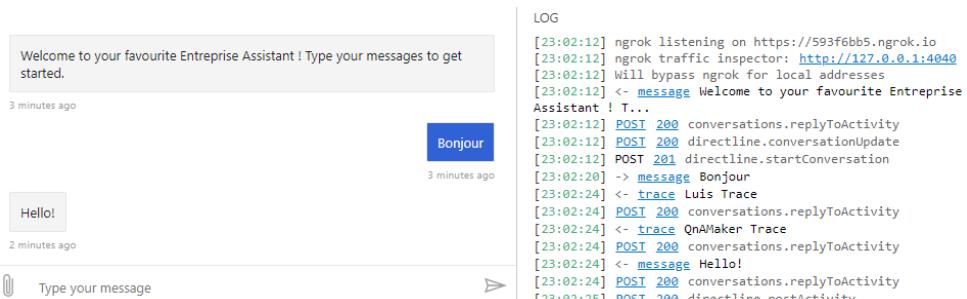


FIGURE 4.7 – Test de l'application QnA Maker

La partie de LOG dans la figure présente les étapes par lesquels passe le flux de communication entre l'application du bot et les services de LUIS et QnA Maker. Ces étapes sont ordonnées comme décrit précédemment dans le diagramme de séquence de la figure 4.3. Le message "LUIS trace" indiqué dans la partie LOG de la figure 4.7 représente l'étape d'analyse linguistique du message par l'application Dispatch. Le débogage du flux sortant de Dispatch donne des informations en objet JSON présentées dans la figure 4.8. L'output de la fonction de LUIS appelée RecognizerResult contient la valeur de l'intention générale, le score de précision et le message reçu.

INSPECTOR - LUIS

App ID: 807efc6a-1c15-4eff-8571-c5fad60268eb

[Recognizer Result](#) [Raw Response](#)

```
{
  "recognizerResult": {
    "alteredText": null,
    "entities": {
      "$instance": {}
    },
    "intents": {
      "q_qnaqna": {
        "score": 0.96908015
      }
    },
    "text": "bonjour"
  }
}
```

FIGURE 4.8 – Débogage de l’activité d’appel de QnA Maker

La figure 4.9 présente le test du bot en envoyant le message "What are the available Mountain Bikes for production ?".

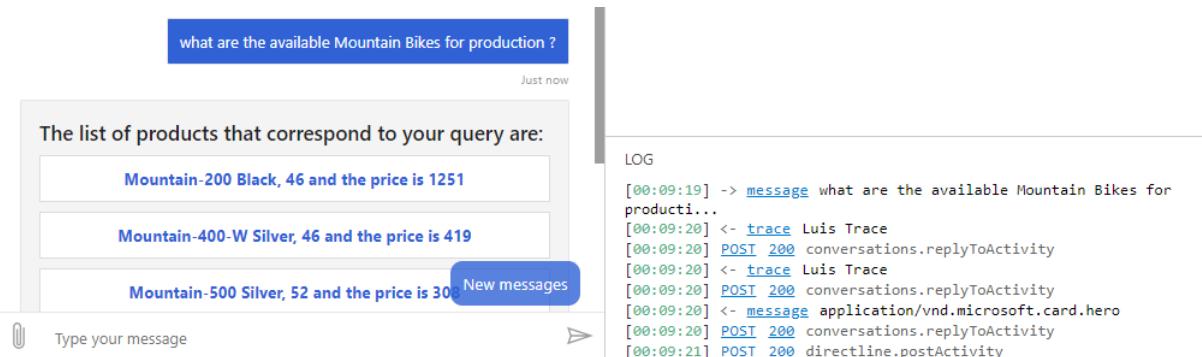


FIGURE 4.9 – Test de l’application LUIS Production Departement

Le débogage du flux sortant de Dispatch donne des informations en objet JSON présentées dans la figure 4.10. L’object JSON a la même structure que celle de la figure 4.8 mais avec une valeur d’intention générale "lProductionDepartment".

INSPECTOR - LUIS

App ID: 807efc6a-1c15-4eff-8571-c5fad60268eb

[Recognizer Result](#) [Raw Response](#)

```
{
  "recognizerResult": {
    "alteredText": null,
    "entities": {
      "$instance": {}
    },
    "intents": {
      "l_Production_Department": {
        "score": 0.9983291
      }
    },
    "text": "what are the available
Mountain Bikes for production ?"
  }
}
```

FIGURE 4.10 – Débogage de l'activité d'appel de Production Departement

Le deuxième message "LUIS trace" indiqué dans la partie LOG de la figure 4.9 représente l'étape d'analyse linguistique du message par l'application "Département de production". Le débogage du flux sortant de l'application "Département de production" donne des informations en objet JSON présentées dans la figure 4.11.

INSPECTOR - LUIS

App ID: 5745d6a8-e3c9-49d1-82fc-c0b892cd7826

[Recognizer Result](#) [Raw Response](#)

```
{
  "recognizerResult": {
    "alteredText": null,
    "entities": {
      "$instance": {
        "categories": [
          {
            "endIndex": 37,
            "startIndex": 23,
            "text": "mountain bikes",
            "type": "categories"
          },
          {
            "endIndex": 37,
            "startIndex": 32,
            "text": "bikes",
            "type": "categories"
          }
        ]
      },
      "categories": [
        [
          "Bikes"
        ],
        [
          "Bikes"
        ]
      ]
    }
  }
}
```

FIGURE 4.11 – Débogage de l'output de l'application "Département de production"

L'objet JSON affiché dans la figure précédente contient les informations suivantes :

- l'intention spécifique "ask about category"
- les entités détectées "Categories" et leurs valeurs "Mountain Bikes"

4.4 Conclusion

Dans ce chapitre on a analysé le flux de données dans le deuxième module du projet grâce à la présentation de l'architecture logique, physique et les différents diagrammes de classe, de séquence et de composants. On a clôturé cette modélisation par un test de code. Dans le chapitre suivant, on expliquera le fonctionnement du module de recherche de données.

Chapitre 5

Module de recherche de données

Ce module de recherche a pour fonction la conversion du message de langage naturel en une requête SQL exécutable sur la base de données. Dans ce chapitre on présente les technologies employées afin de développer ce module, la conception adaptée et on teste ses fonctionnalités dans une démo.

5.1 Conception préliminaire et détaillée

5.1.1 Architecture logique

L'architecture logique de ce module modélise les éléments principaux dans la recherche de données ainsi que la liaison entre eux. Elle est présentée dans le schéma de la figure 5.1.

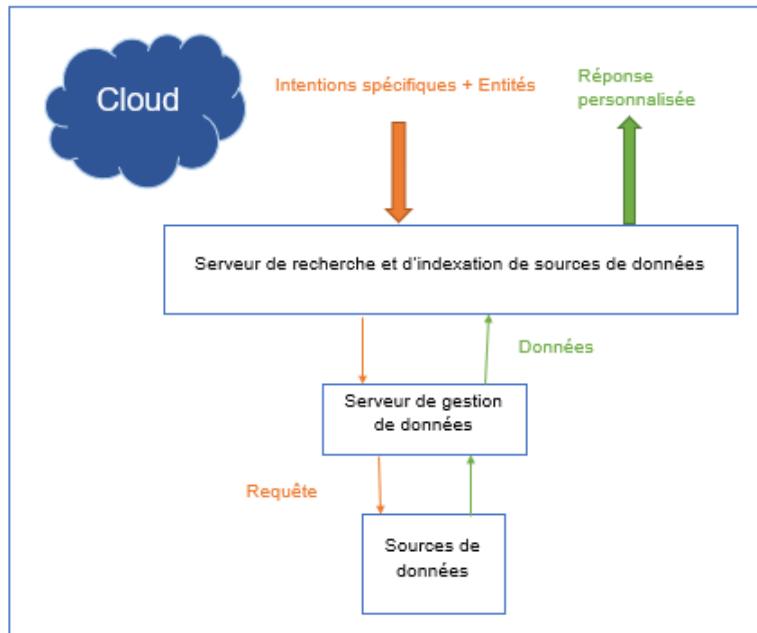


FIGURE 5.1 – Architecture logique de la recherche

5.1.2 Diagramme de conception de séquence et de classes

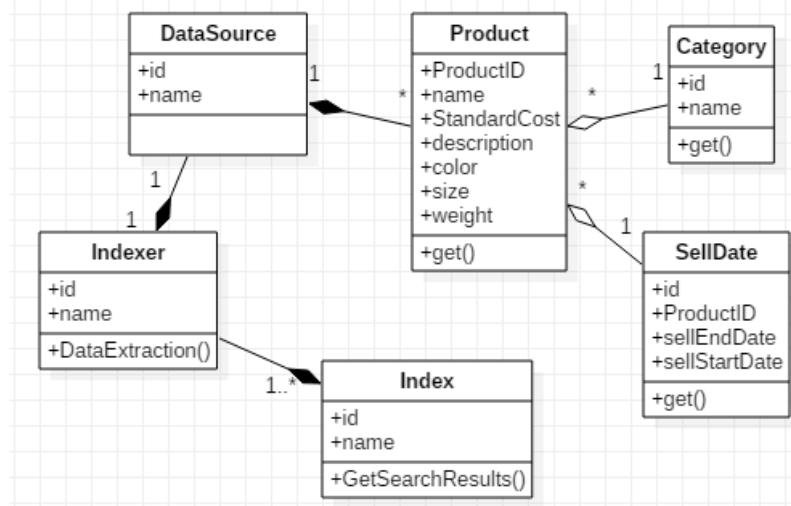


FIGURE 5.2 – Diagramme de classes de la recherche

Le diagramme de classe (voir figure 5.2) présente les différents modèles utilisés au cours du développement de cette application. Il est composé des classes suivantes :

- Product : cette classe contient les différentes informations sur les produits. Elle est reliée à la classe de SellDate et Category.
- SellDate : elle contient la date de début et de fin de production pour chaque produit.
- Category : cette classe contient les noms des types de produits.
- Indexer : elle est générée par Azure Search Service et contient les colonnes des tables et leurs types.
- Index : elle contient les données structurées et prêtes pour la requête.
- Data Source : cette classe rassemble les modèles dans une version prête pour l'indexation.

Dans le service de Azure Search, la DataSource ne peut lire que des données provenant d'une seule table SQL. On a donc combiné les tables Category, SellDate et Product dans une seule table sous forme de view appelée Production afin que tous les données s'indexent à partir d'une seule DataSource.

Les entités extraites dans le module précédent servent de mots clés pour la recherche. Cette recherche est établie selon la valeur de l'intention spécifique. Chaque intention correspond à une fonction et des paramètres de recherche. Cette fonction est appelée par l'application du bot et s'exécute sur l'index. Ce dernier appelle la fonction GetSearchResults qui s'exécute sur l'indexer. Cet indexer extrait les données de la requête de la Data Source. La Data retornnée est convertit en variable SearchResults qui est envoyée au bot sous format Documents. Ce scénario est modélisé dans le diagramme de séquence de le figure 5.3.

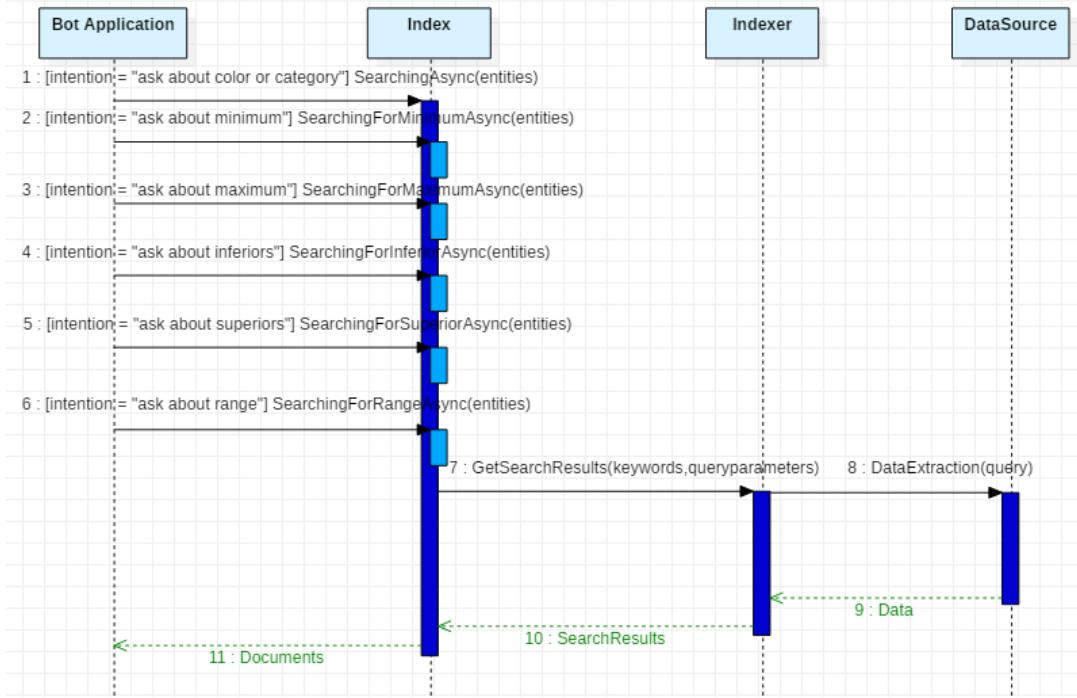


FIGURE 5.3 – Diagramme de séquence de la recherche

5.2 Branche technique

5.2.1 Outils et technologies

Les outils utilisés peuvent se résumer dans la liste suivante :

- * Azure Search [4] : c'est une solution de recherche en tant que service permettant aux développeurs d'intégrer des expériences de recherche de qualité dans des applications sans gérer l'infrastructure ni devoir devenir des experts en recherche. Il a pour sources de données des informations structurées et semi-structurées stockées dans des serveurs Microsoft.
- * Azure SQL Server : c'est un service de gestion de stockage disponible dans le cloud d'Azure. Il offre la gestion de bases de données des entreprises on-premises.
- * Azure SQL Database : c'est une base de données d'entreprise hébergée dans le Cloud d'Azure.

5.2.2 Architecture physique

L'architecture physique de ce module indique les éléments principaux implémentés ainsi que le flux de communication entre eux.

Elle peut être représentée dans le schéma de la figure 5.4.

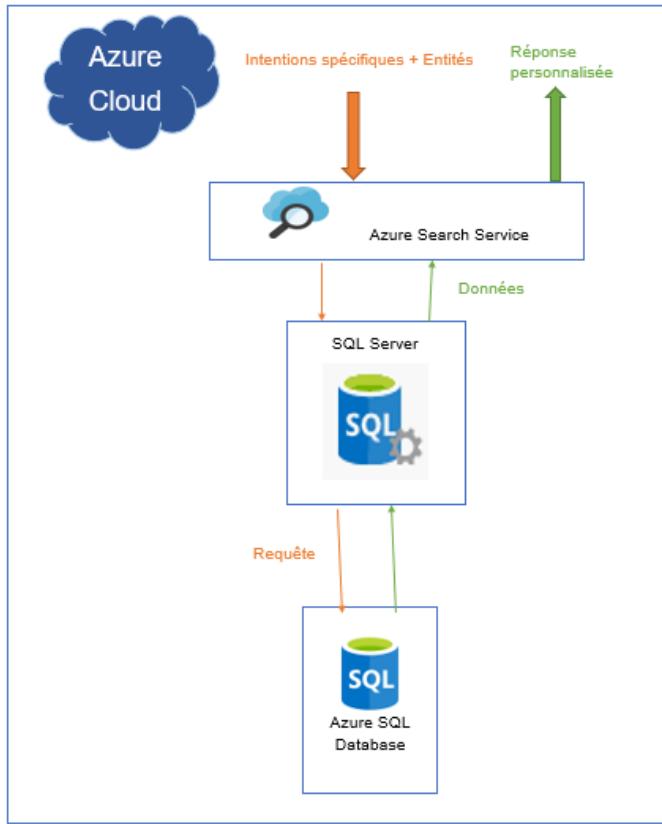


FIGURE 5.4 – Architecture physique de la recherche

L'intention spécifique et les entités sont traitées par ce module de recherche, convertit en requête puis en données. Cette procédure passe par le service de recherche d'Azure Search puis par le serveur de gestion de bases de données SQL Server relié au service de stockage Azure SQL Database.

5.2.3 Diagramme de composants

Linstanciation du service de recherche de données se réalise quand lutilisateur se connecte au bot. Le fichier principal EnterpriseBot.cs fait appel à Startup.cs. Ce fichier reçoit les clés de configuration de chaque service ServicesKey à partir du fichier EnterpriseBot.bot. Ce fichier contient les clés de Azure Search Service et lIndex. Startup.cs retourne à la fin un objet ConfigureServices qui sera directement utilisé par le fichier principal. Ces liaisons sont modélisées dans le diagramme de composants de la figure 5.5.

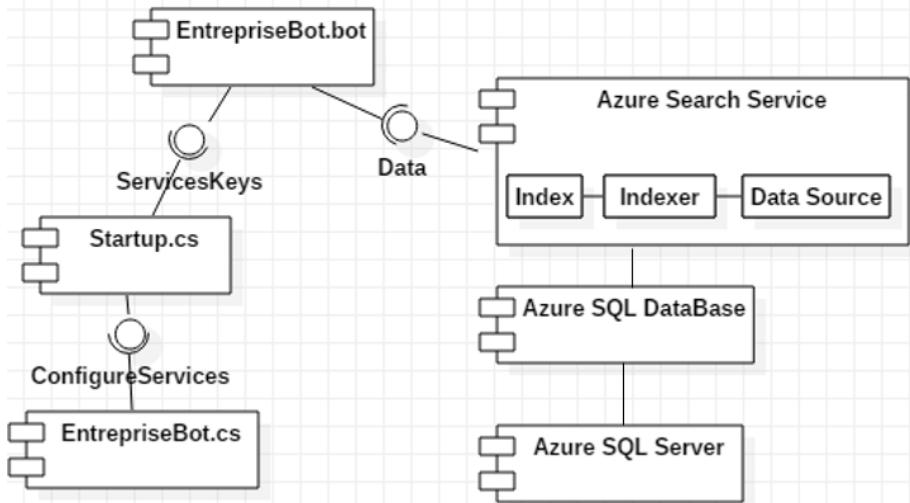


FIGURE 5.5 – Diagramme de composants de la recherche

5.3 Tests et codage

Le test de ce module se compose de plusieurs tests de questions saisies par l'utilisateur. Chaque question fait appel à une fonction provenant d'une intention spécifique. La figure 5.6 présente le résultat de question correspondant à une demande d'informations sur les cadres rouges dont le prix est supérieur à 100 euros.

<http://localhost:3978/api/messages>

is there any red road frames that cost more than 100 euro ?
2 minutes ago

The list of products that correspond to your query are:

- LL Road Frame - Red, 62 and the price is 187
- LL Road Frame - Red, 60 and the price is 187
- ML Road Frame - Red, 52 and the price is 352
- LL Road Frame - Red, 58 and the price is 187
- ML Road Frame - Red, 60 and the price is 352
- HL Road Frame - Red, 62 and the price is 868

FIGURE 5.6 – Résultat de requête de supérieurs

Comme le montre la figure 5.7, LUIS extrait les entités de nombre, catégorie et couleur et les associe avec une intention spécifique "get the superiors".

INSPECTOR - LUIS

Train Publish

App ID: 5745d6a8-e3c9-49d1-82fc-c0b892cd7826 Version: Unknown Slot: Production

[Recognizer Result](#) [Raw Response](#)

```
{
  "recognizerResult": {
    "alteredText": null,
    "entities": {
      "$instance": {
        "categories": [
          {
            "endIndex": 28,
            "startIndex": 17,
            "text": "road frames",
            "type": "categories"
          },
          {
            "endIndex": 28,
            "startIndex": 22,
            "text": "frames",
            "type": "categories"
          }
        ],
        "color": [
          {
            "endIndex": 16,
            "score": 0.814441144,
            "startIndex": 13,
            "text": "red",
            "type": "color"
          }
        ]
      }
    }
  }
}
```

Top-Scoring Intent
get_the_superiors
(0.7818961)

Entities
categories -->
Frames, Frames
color -->
red
number --> 100

FIGURE 5.7 – Débogage de requête de supérieurs

Les figures 5.8 et 5.9 montrent le test de l'intention spécifique "get the inferiors" : l'utilisateur dans ce cas demande les vélos de route rouges qui coûtent moins de 500 euro.

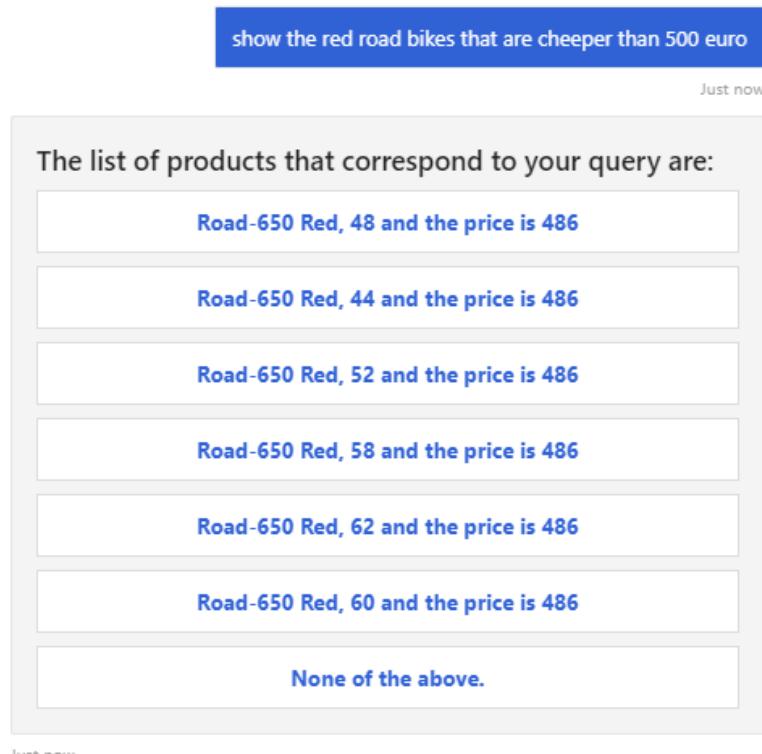


FIGURE 5.8 – Résultat de requête d'inférieurs

INSPECTOR - LUIS Train Publish

App ID: 5745d6a8-e3c9-49d1-82fc-c0b892cd7826 Version: Unknown Slot: Production

[Recognizer Result](#) [Raw Response](#)

```
{
  "recognizerResult": {
    "alteredText": null,
    "entities": {
      "$instance": {
        "categories": [
          {
            "endIndex": 23,
            "startIndex": 13,
            "text": "road
bikes",
            "type": "categories"
          },
          {
            "endIndex": 23,
            "startIndex": 18,
            "text": "bikes",
            "type": "categories"
          }
        ],
        "color": [
          {
            "endIndex": 12,
            "score": 1
          }
        ]
      }
    }
  }
}
```

Top-Scoring Intent
get_the_inferiors (0.2042364)

Entities
categories --> Bikes, Bikes
color --> red
number --> 500

FIGURE 5.9 – Débogage de requête d'inférieurs

La figure 5.10 montre le test de l'intention spécifique "ask with between condition" : l'utilisateur dans ce cas demande les vélos de montagne noirs qui coûtent entre 200 et 600 euro. Le paramétrage associé à la fonction "SearchingForRangeA-async()" dans la requête du service Azure Search est présenté dans la figure 5.11. Ce code permet de traduire les entités en une requête SQL exécutable sur la table SQL. Les entités de type String sont enregistrées dans une liste de String "CombinedKeywords" et celles de type Int dans la variable "IntKeywords"

http://localhost:3978/api/messages

The list of products that correspond to your query are:

- Mountain-400-W Silver, 38 and the price is 419
- Mountain-500 Silver, 40 and the price is 308
- Mountain-500 Silver, 44 and the price is 308
- Mountain-500 Black, 42 and the price is 294
- Mountain-300 Black, 44 and the price is 598
- Mountain-400-W Silver, 42 and the price is 419
- Mountain-500 Silver, 42 and the price is 308

INSPECTOR - LUIS Train Publish

App ID: 5745d6a8-e3c9-49d1-82fc-c0b892cd7826 Version: Unknown Slot: Production

[Recognizer Result](#) [Raw Response](#)

```
{
  "recognizerResult": {
    "alteredText": null,
    "entities": {
      "$instance": {
        "categories": [
          {
            "endIndex": 32,
            "startIndex": 18,
            "text": "mountain bikes",
            "type": "categories"
          },
          {
            "endIndex": 32,
            "startIndex": 27,
            "text": "bikes",
            "type": "categories"
          }
        ],
        "number": [
          {
            "endIndex": 54,
            "startIndex": 51,
            "subtype": "integer",
            "text": "++++- 200"
          }
        ]
      }
    }
  }
}
```

Top-Scoring Intent
ask_with_between_condition (0.24206011)

Entities
categories --> Bikes, Bikes
number --> 200, 600

FIGURE 5.10 – Test et débogage de requête de marge de prix

```

// Prepare the parameters used for the search
SearchParameters parameters = new SearchParameters()
{
    SearchMode = SearchMode.All,
    Skip=4,
    Filter = "StandardCost lt "+sup.ToString()+" and StandardCost gt "+inf.ToString(),
    SearchFields = new[] { "Name", "Color", "ProductcategoryName" }
};


```

FIGURE 5.11 – code de paramètres de requête de comparaison de prix

Le chatBot peut aussi donner le produit le moins cher et le plus cher. Ceci est démontré dans les figures 5.12 et 5.13.

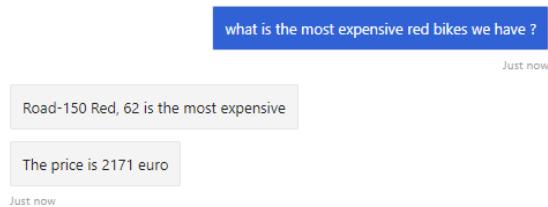


FIGURE 5.12 – Test de requête de maximum

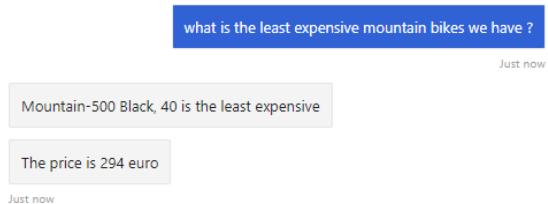


FIGURE 5.13 – Test de requête de minimum

5.4 Conclusion

Ce chapitre nous a permis de bien comprendre comment le bot réussit à traduire un message écrit en langage naturel en une requête SQL exécutable par le service de recherche. Nous avons modélisé cette fonctionnalité avec plusieurs diagrammes qui modélise les liaisons entre les composants du module et la synchronisation des données entre eux. Ceci est clôturé par le test de codage de différents scénarios d'utilisation du bot.

Conclusion générale

Ce projet de fin d'étude est le fruit d'un travail d'un stage de quatre mois, il décrit d'une façon détaillée la conception et la réalisation d'un chatBot.

Le but de ce travail est de concevoir et développer un chatBot qui répond aux questions du directeur de l'entreprise avec fidélité et en un temps record. Le projet permet à l'utilisateur d'entamer une discussion en langage naturel afin de contrôler le département de production. Le chatBot est une alternative à une solution existante qui consiste à organiser des réunions et réaliser des rapports afin de répondre à la question du directeur de l'entreprise.

Cependant, cette application est en cours de développement. Elle ne satisfait pas encore tous les besoins du client. En effet, sa base de connaissance est limitée et elle ne permet de contrôler qu'un seul département de l'entreprise.

Comme perspective, nous souhaitons pour les prochaines améliorations élargir nos sources de données en rajoutant des documents et des bases de données reliées à d'autre départements de l'entreprise. Il y a aussi une possibilité d'intégrer un service de reconnaissance vocale pour l'authentification de l'utilisateur lors de sa connexion.

Pour terminer, ce stage m'a permis d'élargir et d'approfondir mes connaissances sur le développement des chatBots, l'intégration des services d'intelligence artificielle de Microsoft Azure et m'a rendu plus ambitieuse et plus motivée pour continuer dans ce domaine.

Bibliographie

- [1] Amazon. Amazon.com. <https://www.amazon.com/>, Mai 2019. Dernière consultation 2019-05-20.
- [2] Microsoft Azure. Microsoft bot framework v4 sdk is now generally available. <https://azure.microsoft.com/en-in/updates/microsoft-bot-framework-v4-sdk-is-now-generally-available/>, Septembre 2018. Dernière consultation 2019-05-20.
- [3] Microsoft Azure. Azure bot service documentation. <https://azure.microsoft.com/en-in/updates/microsoft-bot-framework-v4-sdk-is-now-generally-available/>, Mai 2019. Dernière consultation 2019-05-20.
- [4] Microsoft Azure. Azure search fundamentals. <https://azure.github.io/LearnAI-KnowledgeMiningBootcamp/labs/lab-azure-search.html>, Janvier 2019. Dernière consultation 2019-05-20.
- [5] Microsoft Azure. Microsoft bot framework. <https://dev.botframework.com/>, Février 2019. Dernière consultation 2019-05-20.
- [6] Microsoft Azure. Qna maker documentation. <https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/overview/overview>, Mai 2019. Dernière consultation 2019-05-20.
- [7] Microsoft Azure. Qu'est-ce que le service language understanding (luis) ? <https://docs.microsoft.com/fr-fr/azure/cognitive-services/luis/what-is-luis>, Janvier 2019. Dernière consultation 2019-05-20.
- [8] Microsoft Azure. What's new in bot framework. <https://azure.microsoft.com/en-in/updates/microsoft-bot-framework-v4-sdk-is-now-generally-available/>, Mai 2019. Dernière consultation 2019-05-20.
- [9] discover.bot. Popular chatbot frameworks. <https://discover.bot/bot-talk/beginners-guide-bots/popular-chatbot-frameworks/>, Mai 2018. Dernière consultation 2019-05-15.
- [10] Los Angeles Government. Official website. <https://www.lacity.org/chat-chip>, Mars 2019. Dernière consultation 2019-05-20.
- [11] Murali K. An introduction to luis (language understanding intelligent service). <https://dzone.com/articles/luis-language-understanding-intelligent-service>, Janvier 2018. Dernière consultation 2019-05-20.
- [12] Chatbots Magazine. A visual history of chatbots. <https://chatbotsmagazine.com/a-visual-history-of-chatbots-8bf3b31dbfb2>, Mai 2019. Dernière consultation 2019-05-20.

- [13] Microsoft. Témoignages clients. <https://customers.microsoft.com/fr-fr/home?sq=&ff=&p=0>, Juillet 2017. Dernière consultation 2019-05-20.
- [14] Microsoft. Dispatch. <https://github.com/Microsoft/botbuilder-tools/tree/master/packages/Dispatch>, Mai 2019. Dernière consultation 2019-05-20.
- [15] Microsoft Customer Stories. Banking bots : The future of communication. <https://customers.microsoft.com/en-us/story/fiducia-gad-cortana-azure-powerbi-services-bot-framework-banking-germany-en>, Septembre 2017. Dernière consultation 2019-05-20.
- [16] Microsoft Customer Stories. How the city of los angeles is enriching citizen and employee experiences with chip the chatbot. <https://customers.microsoft.com/en-us/story/los-angeles-azure-developer-government-united-states>, Juillet 2017. Dernière consultation 2019-05-20.
- [17] Microsoft Customer Stories. Navigation technology company creates chatbot to help tourists get around japan. <https://customers.microsoft.com/en-us/story/navitime-japan-travel-hospitality-microsoft-cognitive-services>, Septembre 2017. Dernière consultation 2019-05-20.
- [18] Veepee. Ventes privées. <https://www.veepee.fr/>, Mai 2019. Dernière consultation 2019-05-20.
- [19] Wikipedia. C#. https://fr.wikipedia.org/wiki/C_sharp, Avril 2019. Dernière consultation 2019-05-20.
- [20] Wikipedia. Microsoft visual studio. https://fr.wikipedia.org/wiki/Microsoft_Visual_Studio, Avril 2019. Dernière consultation 2019-05-20.
- [21] Wikipedia. .net core. https://fr.wikipedia.org/wiki/.NET_Core, Janvier 2019. Dernière consultation 2019-05-20.

الملخص

يصف هذا التقرير العمل المنجز كجزء من مشروع التخرج في المنظمة المضيفة BI-READY. يركز المشروع على موضوع "إنشاء مساعد افتراضي لأنجذبة الخدمات الداخلية للشركة". الغرض العام من التطبيق هو أنجذبة مراقبة قسم الإنتاج للشركة.

يتمثل هذا التطبيق في وكيل محادثة قادر على التحدث مع المستخدم بإستعمال اللغة الطبيعية وتوفير المعلومات الحديثة عن منتجات بصدق التصنيع. إن الحل المقترن في إطار هذا المشروع هو بديل لطريقة تقليدية تتمثل في تنظيم الاجتماعات مع الموظفين وكتابة تقارير. الحل المقترن في هذا التقرير أرخص وأسرع من الحل الحالي.

من خلال دراسة الطريقة التقليدية السابقة وتصميم البديل وتطوير نموذج أولي، واجهتني مشاكل التحليل وفهم اللغة الطبيعية من قبل الكمبيوتر، ولكن أيضاً فهرسة واستخراج المعلومات من قاعدة البيانات.

الكلمات الرئيسية ChatBot, Microsoft Bot Framework, Azure Search, LUIS

Résumé

Ce présent rapport fait état du travail effectué dans le cadre de projet de fin d'études dans l'organisme d'accueil BI-READY. Le projet porte sur le thème « Crédation d'assistant virtuel pour automatisation de services internes d'une entreprise ». L'objectif général de l'application est d'automatiser le contrôle du département de production d'une entreprise.

Cette application est un agent conversationnel capable d'établir une discussion avec l'utilisateur en langage naturel et de fournir des informations récentes sur des produits. Il représente une alternative à une méthode traditionnelle qui consiste à organiser des réunions avec les employés et de rédiger des rapports. La solution proposée dans ce rapport est moins chère et plus rapide que la solution existante.

Au travers de l'étude de l'état de l'art, de l'analyse du besoin et du développement d'un prototype, j'ai été confronté aux problématiques de l'analyse et de la compréhension du langage naturel par l'ordinateur, mais également de l'indexation et de l'extraction d'information de la base de données.

Mots clés: ChatBot, Microsoft Bot Framework, Azure Search, LUIS

Abstract

This report describes the work done as part of a graduation project which took place at BI-READY. The project focuses on the theme "Creating Virtual Assistant for Automating Internal Services of a Company". The general purpose of the application is to automate the control of a company's production department.

This application is a conversational agent able to establish a discussion with the user in natural language and to provide updated information about products. It is an alternative to a traditional method of organizing meetings with employees and writing reports. The solution proposed in this report is cheaper and faster than the existing one.

Through the study of the state of the art, the analysis of the need and the development of a prototype, I was confronted with the problems of the analysis and the understanding of the natural language by the computer, but also indexing and extracting information from the database.

Keywords: ChatBot, Microsoft Bot Framework, Azure Search, LUIS