

Projet Industriel

Développement d'un chatbot à usage professionnel

Guillaume IZART
Malaury KESLICK
Théo PONCET

Année 2020–2021

Projet industriel réalisé pour l'entreprise Figeac Aero.



Encadrant industriel: Loïc VERNEY

Encadrant universitaire : François CHAROY

Déclaration sur l'honneur de non-plagiat

Nous soussignons,

Noms, prénoms : Guillaume IZART, Malaury KESLICK et Théo PONCET

Élèves-ingénieurs régulièrement inscrits en 3^e année à TELECOM Nancy

Numéro de carte de l'étudiant(e) : 31816573, 31814820 et 31814825

Année universitaire : 2020–2021

Auteurs du document, mémoire, rapport ou code informatique intitulé :

Développement d'un chatbot à usage professionnel

Par la présente, nous déclarons nous être informés sur les différentes formes de plagiat existantes et sur les techniques et normes de citation et référence.

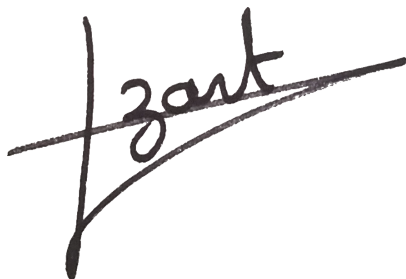
Nous déclarons en outre que le travail rendu est un travail original, issu de notre réflexion personnelle, et qu'il a été rédigé entièrement par nos soins. Nous affirmons n'avoir ni contrefait, ni falsifié, ni copié tout ou partie de l'œuvre d'autrui, en particulier texte ou code informatique, dans le but de nous l'accaparer.

Nous certifions donc que toutes formulations, idées, recherches, raisonnements, analyses, programmes, schémas ou autre créations, figurant dans le document et empruntés à un tiers, sont clairement signalés comme tels, selon les usages en vigueur.

Nous sommes conscients que le fait de ne pas citer une source ou de ne pas la citer clairement et complètement est constitutif de plagiat, que le plagiat est considéré comme une faute grave au sein de l'Université, et qu'en cas de manquement aux règles en la matière, nous encourrions des poursuites non seulement devant la commission de discipline de l'établissement mais également devant les tribunaux de la République Française.

Fait à Villers-lès-Nancy, le 15 février 2021

Signature :

Handwritten signature of Guillaume IZART in black ink.Handwritten signature of Malaury KESLICK in blue ink.Handwritten signature of Théo PONCET in black ink.

Projet Industriel

Développement d'un chatbot à usage professionnel

Guillaume IZART
Malaury KESLICK
Théo PONCET

Année 2020–2021

Projet industriel réalisé pour l'entreprise Figeac Aero

Guillaume IZART
Malaury KESLICK
Théo PONCET
guillaume.izart@telecomnancy.eu
malaury.keslick@telecomnancy.eu
theo.poncet@telecomnancy.eu

TELECOM Nancy
193 avenue Paul Muller,
CS 90172, VILLERS-LÈS-NANCY
+33 (0)3 83 68 26 00
contact@telecomnancy.eu

Figeac Aero
Zone Industrielle de l'Aiguille
46100, Figeac
+33565345252



Encadrant industriel : Loïc VERNEY

Encadrant universitaire : François CHAROY

Remerciements

Nous tenons à remercier la société Figeac Aero pour la confiance qu'elle nous a accordée.

Plus particulièrement, nous souhaitons remercier notre encadrant industriel Loïc Verney ainsi que son collègue Simon Auroy de nous avoir accompagnés et soutenu tout au long de la durée de ce projet industriel. Nos échanges, toujours très constructifs, nous ont permis de faire avancer le projet dans la bonne direction.

Enfin, merci également à notre encadrant académique François Charoy qui nous a permis d'avoir un avis extérieur sur le projet et grâce à qui toute l'équipe a pu prendre du recul sur son travail.

Table des matières

Remerciements	v
Table des matières	vi
1 Introduction	1
2 Présentation de l'entreprise	3
3 Note de cadrage	4
4 Cahier des Charges	5
4.1 Contexte et objectifs du cahier des charges	5
5 Etat de l'art	6
6 Conception	12
6.1 Structure générale et choix de conception	12
6.1.1 Le choix du Cloud de Microsoft : Azure	12
6.1.2 Notre utilisation d'Azure et de ses outils	13
6.1.3 Structure du chatbot	13
6.1.4 Sécurisation de l'accès au chatbot	15
6.2 Recherche de fichiers	16
6.3 Luis	18
6.3.1 Principe	18
6.3.2 Intégration du service à notre chatbot	19
6.4 QnA	20
6.4.1 Principe	20
6.4.2 Intégration du service à notre chatbot	23
7 Méthodologie	24
7.1 Réunions	24

7.2	Roadmap du projet	24
7.3	Affectation des postes	25
7.4	Gestion du projet	25
7.5	Livrables	25
8	Bilan	27
8.1	Difficultés rencontrées	27
8.1.1	Difficultés environnementales	27
8.1.2	Difficultés techniques	27
8.2	Heures effectuées	28
8.3	Futur du produit	28
8.4	Conclusion	30
	Bibliographie / Webographie	31
	Liste des illustrations	32
9	Glossaire	33
	Annexes	35
A	Schéma illustratif	35
B	Note de cadrage	36
C	Cahier des charges	41
D	RoadMap	50
E	Diagramme de Gantt globale effectif	53
F	Fichie technique	56
G	Comptes-rendus de réunion	68
G.1	Première réunion	69
G.2	Troisième réunion	71
G.3	Cinquième réunion	73
G.4	Réunion de confrontation des attentes	75
G.5	Septième réunion	78

Résumé	84
Abstract	85

1 Introduction

L'entreprise Figeac Aero, comptant plus de mille salariés à ce jour, fait face à une surcharge des équipes d'assistance qui doivent gérer des questions redondantes et des envois d'informations ou de fichiers déjà présents dans les espaces de stockage accessibles aux employés. Le support reçoit en moyenne 580 demandes par mois pour des demandes de fichier que les utilisateurs n'arrivent pas à trouver.

Afin de rendre ces équipes plus disponibles pour les problèmes techniques plus pointus et nécessitant une assistance humaine, l'entreprise a proposé un sujet de projet industriel à TELECOM Nancy dans l'optique de développer et intégrer un chatbot dans leur structure Microsoft Teams.

Ce chatbot aura donc pour objectif d'utiliser une base de connaissance sous forme de question-réponse, permettant de répondre aux questions les plus fréquemment posées par les utilisateurs. Ceci permettra de retirer une partie du travail de l'assistance technique.

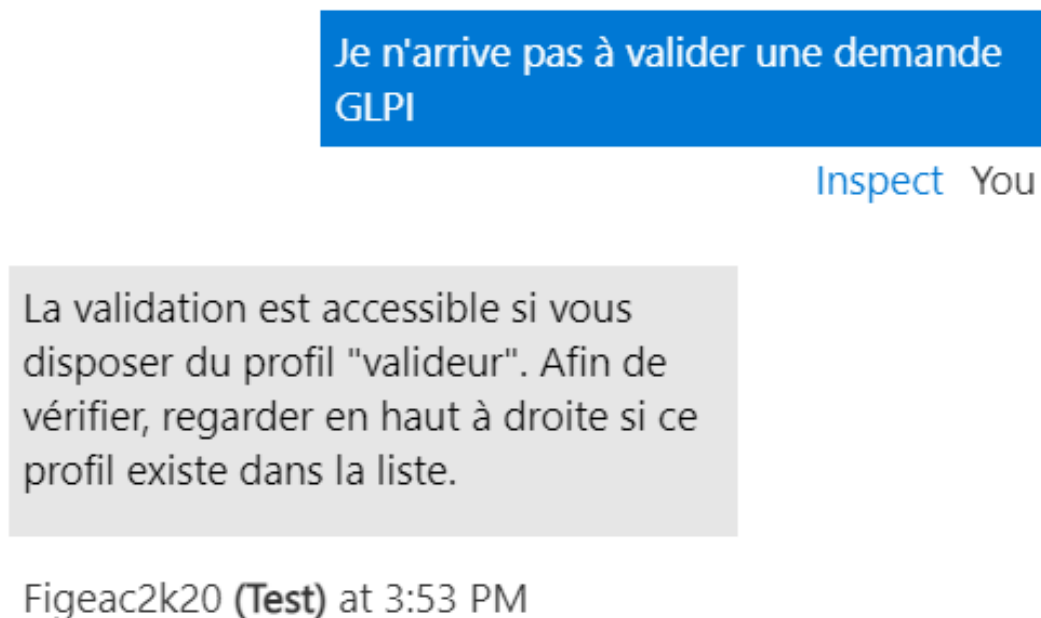


FIGURE 1.1 – Exemple de question qu'un utilisateur peut demander au chatbot

Aussi, ce chatbot aura pour second objectif de répondre à des demandes de recherche de fichier par des utilisateurs, en allant chercher les documents sur les différentes bases de données de l'entreprise (dossiers partagés Windows, Microsoft SharePoint, documents partagés sur Microsoft

Teams).

Ce chatbot sera donc principalement un outil d'aide de recherche de fichiers et de réponse aux questions courantes pour l'utilisateur final. Dans sa version Proof Of Concept le bot aura pour objectif de répondre à 5% des demandes faites au support, ce qui représente environ 30 demandes, afin de diminuer le temps que l'assistance consacre aux utilisateurs

Ce rapport rend compte du travail effectué sur toute la durée du projet industriel, projet pédagogique intégré dans le cursus ingénieur des élèves de TELECOM Nancy.

Ce document débutera par une rapide présentation de Figeac Aero, suivie du cahier des charges du projet industriel. Les parties suivantes présenteront le travail réalisé par le groupe projet pour mener à bien la réalisation du chatbot.

2 Présentation de l'entreprise

La société Figeac Aéro a été créée en 1989 et est gérée depuis lors par son PDG Jean-Claude MAILLARD. Son siège social est situé dans la ville de Figeac dans le Lot. Son dirigeant, à l'origine ingénieur mécanicien dans le domaine de l'aéronautique, a profité de son expérience pour lancer son entreprise de sous-traitance pour la production de pièces en aluminium. On compte parmi les partenaires de l'entreprise des grands noms de l'aéronautique et de l'espace comme Airbus, Boeing ou encore Stelia Aerospace.

Finalités : La raison d'être principale de Figeac Aero est bien entendu de générer du profit.

Nature de l'activité : L'activité principale de l'entreprise consiste à produire des pièces de métal pour leurs clients du domaine de l'aéronautique et de l'espace. Elle sous-traite donc ses services à plusieurs partenaires dans le monde entier.

Type d'organisation et statut juridique : Figeac Aero est une Société à Responsabilité Limitée (SRL), ayant un but lucratif.

Ressources :

- financières : L'entreprise a généré un chiffre d'affaire de 445 millions d'euros en 2020.
- humaines : Figeac Aero emploie 3 600 employés à travers le monde, dont 1 129 en France (à vérifier, les chiffres sont pas partout les mêmes)
- matérielles : L'entreprise dispose de 14 usines ainsi que de plus de 350 machines.
- immatérielles : L'entreprise dispose d'un **site** qui lui sert de vitrine et dans lequel sont présentés le groupe ainsi que les différents services et prestations qu'il propose.

Autres caractéristiques : Cette année 2020 a été particulièrement difficile pour les entreprises, et ce particulièrement dans le domaine de l'aéronautique. La crise sanitaire a en effet contraint l'entreprise à restreindre son activité en mettant en place du chômage partiel dans l'ensemble des secteurs de l'entreprise. C'est dans ce contexte, que la DSI de Figeac aéro souhaite réfléchir à la mise en place d'une assistance utilisateur par le biais d'un chatbot.

3 Note de cadrage

La note de cadrage a été un travail demandé par l'école. Ce rapport avait pour but de clarifier notre vision du besoin de l'encadrant industriel, et de reformuler le besoin avec nos mots.

Ensuite, nous avons réalisé un recensement des contraintes du projet, ainsi que des risques potentiels, notamment temporels à cause du temps limité que nous avons à consacrer au projet, et technique, car nous allions devoir utiliser une technologie cloud Azure, à mettre en relation avec des outils Microsoft comme SharePoint ou Teams.

Dans cette note de cadrage, nous avons aussi dû mettre en place une organisation pour la gestion du projet, comme par exemple se mettre d'accord avec l'encadrant industriel pour fixer les réunions d'avancement, leur fréquence, leur horaire etc.

Aussi, nous avons réalisé un diagramme de gantt, afin de reprendre les grandes étapes du projet, la définition des jalons intermédiaires et les acteurs de chaque étapes. Ce diagramme nous servira d'outil de pilotage tout au long de la vie du projet

Enfin, nous avons répertorié les livrables à livrer à l'entreprise Figeac Aero au cours du projet.

Ceci nous a permis d'améliorer les prévisions réalisées sur le diagramme de Gantt en y apportant des jalons majeurs de rendu de livrable.

Cette note de cadrage peut être retrouvée en annexe de ce rapport [ann. B].

4 Cahier des Charges

Au début du projet, la rédaction d'un cahier des charges a été demandée à l'équipe projet afin de permettre de définir les besoins de l'encadrant industriel.

Ce cahier des charges devait reprendre les fonctionnalités attendues par Figeac Aero, ainsi qu'un ordre de priorité permettant d'ordonner les tâches à effectuer dans les semaines suivantes. Ces priorités étaient marquées grâce à un mot entre crochet au début du besoin comme par exemple "[*Haute*] L'utilisateur doit pouvoir écrire un message dans l'IHM du chatbot.", "[*Moyenne*] L'utilisateur doit avoir accès au chatbot 100% du temps sur la plage ouvrée de l'entreprise.", ou encore "[*Faible*] Le système doit pouvoir créer un ticket Helpdesk."

Le but de ce cahier des charges était de synthétiser clairement les besoins de l'encadrant industriel avec des besoins utilisateurs fonctionnels, non-fonctionnels, ainsi que des besoins systèmes fonctionnels et non-fonctionnels.

4.1 Contexte et objectifs du cahier des charges

Ce cahier des charges a été rédigé par l'équipe projet. Nous nous sommes servis d'une méthode de rédaction de cahier des charges étudié dans notre cycle à TELECOM NANCY, qui développe quatre grands axes du projet :

- Les besoins système fonctionnels
- Les besoins système non-fonctionnels
- Les besoins utilisateur fonctionnels
- Les besoins utilisateur non-fonctionnels

Une fois ce cahier de charge rédigé, nous avons organisé une réunion avec l'encadrant industriel pour discuter certains points qui pourrait ne pas être suffisamment clairs ou développés.

Loïc Verney, avec sa vision plus expérimentée sur les cahiers des charges, nous a donc guidé sur la rédaction de certains points qui nécessitaient des modifications.

Après réécriture du cahier des charges, Loïc Verney l'a validé et nous avons donc pu nous appuyer sur celui-ci pour la suite de notre projet.

Ce cahier des charges peut être retrouvé dans les annexes [ann. C].

5 Etat de l'art

Les chatbots

Définition

Un chatbot est un programme informatique permettant de simuler une discussion avec être humain, que ce soit de manière vocale ou textuelle. Dans un premier temps, les chatbots n'étaient que de grosses bases de données permettant de savoir quoi répondre à des mots-clés donnés, et donc devaient être manuellement améliorés pour ne pas devenir obsolètes. En revanche, de nos jours, l'intelligence artificielle a permis de rendre ces agents capables d'apprendre d'eux-mêmes, tout en se rapprochant de plus en plus d'un comportement humain grâce à une compréhension et reproduction du langage naturel très performante. C'est ainsi que les chatbots sont petit à petit devenus nos assistants lors de nombreuses tâches de la vie moderne [6].

Exemples d'utilisation

En effet, à chaque besoin correspond son chatbot. Que vous souhaitiez trouver une information, assouvir votre besoin de danser ou réserver un billet de train, tout ou presque devient possible avec ce nouveau type de service. Les exemples les plus probants sont sûrement les assistants vocaux des grandes puissances Amazon, Apple, Google et Microsoft [fig. 5.1]. En pouvant les déclencher à l'aide sa simple voix, ces sociétés ont révolutionné l'utilisation des outils technologiques, en allant des téléphones portables au mobilier connecté [8].

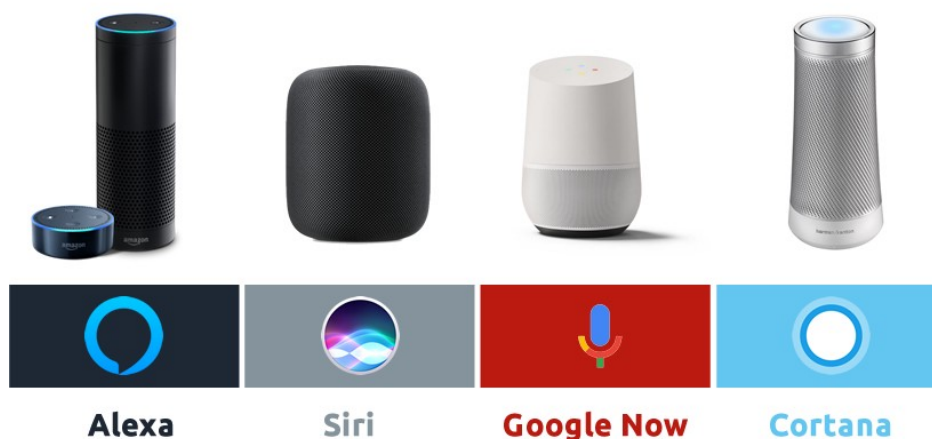


FIGURE 5.1 – Les assistants vocaux respectifs d'Amazon, d'Apple, de Google et de Microsoft

Cependant, les chatbots textuels ne sont pas en reste. Il est de plus en plus courant de voir une pastille nous incitant à entrer en communication avec un chatbot si nous en avons besoin sur un site internet. Plus rapides et disponibles que les être humains, ils permettent d'automatiser les réponses aux besoins d'assistance les plus courants. De plus, aucun domaine n'est en reste ! Du secteur bancaire [fig. 5.2] à la SNCF [fig. 5.3], les utilisateurs trouvent leur compte partout.

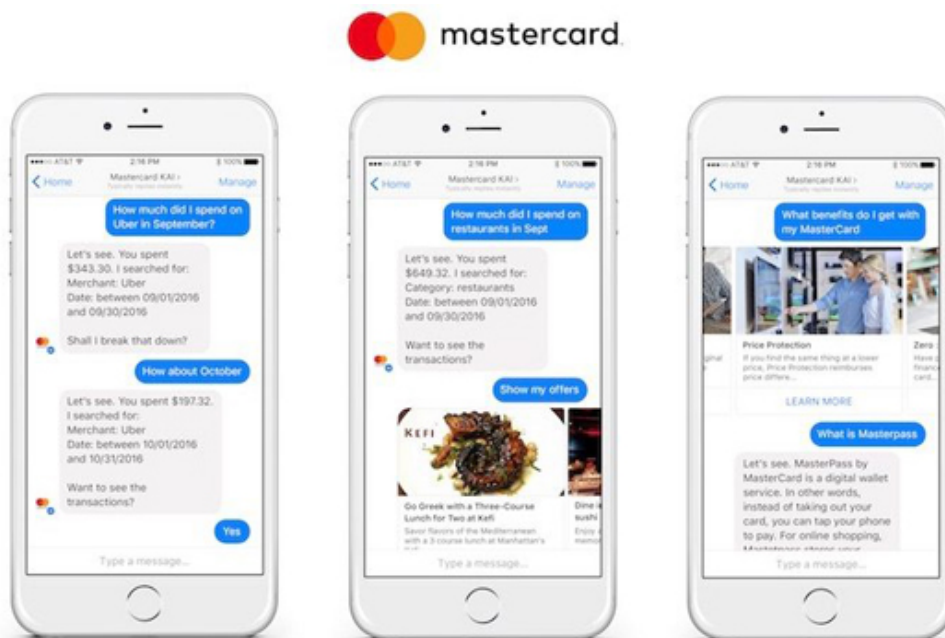


FIGURE 5.2 – Chatbot de Mastercard, accessible via Messenger

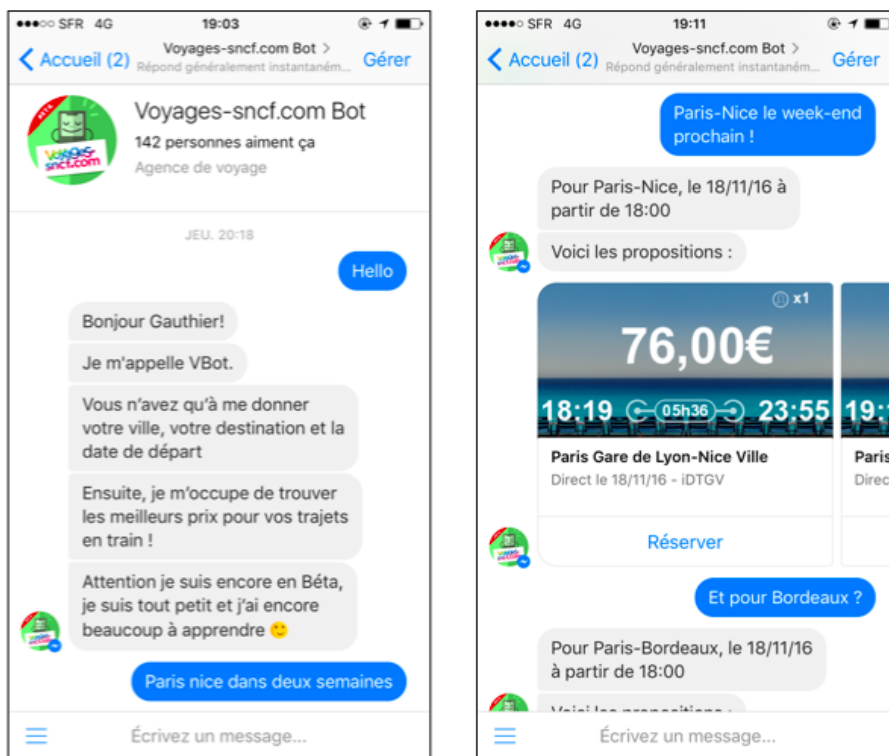


FIGURE 5.3 – VBot, le chatbot de la SNCF accessible via Messenger

Enfin, les chatbot ne sont pas seulement destinés à converser avec des clients. Ils peuvent être très utiles au sein d'une même organisation lorsque celle-ci souhaite optimiser le temps de travail de ses salariés [7]. Rechercher une adresse mail, ne plus se rappeler du nom du DRH, ne plus se souvenir de la manipulation à effectuer lorsque l'imprimante s'est déconnectée du réseau... Tant de manque d'informations qui peuvent faire perdre quelques minutes de temps en temps, et finalement représenter une portion considérable d'une journée de travail. Le chatbot permet d'accéder à l'information instantanément [fig. 5.4], et augmente ainsi la productivité de l'employé, et de tous ceux auxquels il aurait dû demander en temps normal. Si l'entreprise comporte une équipe d'assistance technique, celle-ci pourra alors libérer du temps pour les problèmes plus sérieux, dont la résolution ne peut pas être automatisée.



FIGURE 5.4 – Exemple d'usage d'un chatbot interne à une entreprise

Différentes implémentations

Comme montré précédemment, chaque chatbot a un but différent, et ainsi il n'existe pas de manière universelle de les créer. Le choix des technologies dépendent de plusieurs facteurs :

- le degré de complexité des demandes à satisfaire ;
- le niveau d'utilisation du produit ;
- le système dans lequel intégrer le produit ;
- le budget alloué pour le fonctionnement du produit.

Intelligence artificielle ou pas ?

Si les chatbots les plus populaires sont intelligents et peuvent répondre aux questions même les plus farfelues, ce n'est pas pour cela qu'ils sont adaptés à tout type de besoin. Dans le cas où les scénarios d'utilisation sont précis, clairs et intuitifs, il n'est pas forcément nécessaire d'avoir recours à une intelligence artificielle pour répondre à des questions dont les réponses sont toujours les mêmes. Prenons l'exemple d'un carnet d'adresse pour une PME. Il est tout à fait possible de créer un chatbot prenant le temps de décomposer la demande de l'utilisateur, puis de se connecter à un service de carnet d'adresse tiers pour aller y effectuer des requêtes. En revanche, à cette échelle, il est plus simple, plus rapide et beaucoup moins coûteux de fournir directement le carnet d'adresse au format approprié à un service de *Questions et Réponses*. La maintenance et l'ajout de données à une telle application sera de plus beaucoup plus simple et notamment très accessible pour des clients ne faisant pas forcément partie du monde de l'informatique. Il s'agira alors dans un premier temps de déterminer le niveau de complexité des requêtes auquel le chatbot devra être confronté.

Lien avec des API

Pour accéder à l'information recherchée, le chatbot peut passer par de multiples API. Effectuer une recherche Wikipédia, obtenir la météo du jour ou bien accéder à des documents stockés sur une base de données,... Tout cela est possible, à condition qu'il existe une librairie permettant de faire le lien entre l'API souhaité et votre langage de programmation. Ainsi, il est important d'avoir une idée des fonctionnalités finales du produit avant de débiter la phase de développement, afin de choisir la technologie appropriée en amont.

Converser à 2 ou à plus ?

Le niveau de complexité du produit dépendra énormément du contexte final dans lequel il sera utilisé. En effet, gérer une conversation privée avec un utilisateur sera beaucoup plus simple que de suivre le fil des discussions qui s'entremêlent sur un canal de groupe [fig. 5.5]. En effet, pour développer un chatbot, il faut avoir une idée du déroulé de la conversation, et la probabilité de choses inattendues augmente avec le nombre de membres utilisant l'outil au sein de cette même conversation. Si l'intégration à des canaux de groupe n'est pas nécessaire, il est souvent préférable de privilégier un dialogue de qualité en discussion privé.

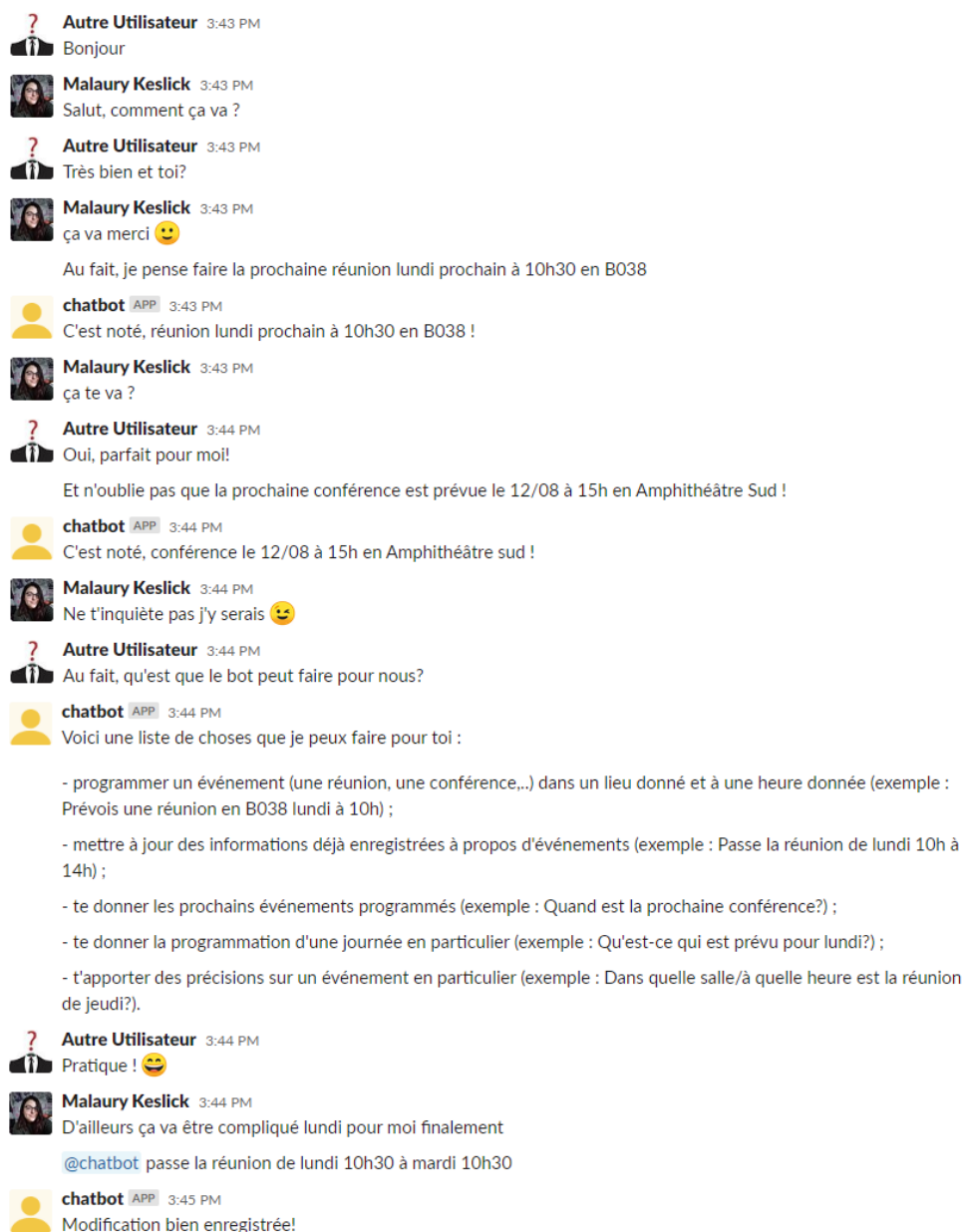


FIGURE 5.5 – Exemple de chatbot au sein d’une conversation multi-utilisateurs développé dans le cadre du stage de deuxième année de Malaury Keslick

Le moyen d’accéder au chatbot

Il existe une multitude d’applications ou de services web permettant d’accueillir un chatbot en son sein : Facebook Messenger, Microsoft Teams, Slask, Whatsapp, etc... Néanmoins, l’intégration du chatbot peut tout aussi bien se faire en trois clics, que se révéler être pratiquement impossible. Le choix du service d’hébergement du chatbot devra donc être fait en considération de cette finalité.

Combien cela coûte-t-il?

Il n’y a pas de réponse toute faite à cette question. Cela va dépendre de beaucoup de choses : des éventuels partenariats entre le client et le service d’hébergement, de l’échelle d’utilisation du pro-

duit, des fonctionnalités souhaitées (par exemple l'envoi de fichiers), du temps de développement du produit... Il existe des solutions basiques avec lesquelles le chatbot ne coûte rien pendant toute sa durée d'utilisation, mais le nombre d'échange pris en charge peut s'avérer très faible pour une entreprise, voire complètement minime dans le cas d'un chatbot destiné à la clientèle.

La solution Cloud

Les technologies Cloud sont aujourd'hui probablement les meilleures pour déployer des chatbot. Celles-ci offrent pratiquement toujours une multitude de services basiques, ce qui les rendent idéales pour tester le projet de création de cet outil dans le cadre d'un POC par exemple. De plus, les géants du Cloud [fig. 5.6] disposent tous de services pré-conçus pour créer, personnaliser, publier et maintenir son chatbot de manière simple.



FIGURE 5.6 – Plateformes des géants de la technologie Cloud

Limites

Bien sûr, les chatbots n'en sont pas encore au point de remplacer l'assistance humaine. Ils sont plutôt au stade de moyen d'optimisation du temps. Néanmoins, cet outil peut être vu comme une entrave à la relation avec la clientèle qui le trouverait impersonnel (même si tous ne remarquent pas forcément qu'ils ne s'adressent pas à un humain).

D'autres débats autour de l'éthique du procédé sont en jeu. En effet, pour apprendre, le chatbot collecte les conversations de l'utilisateur. Ainsi, comme la majorité de ces produits sont hébergés sur des plateformes des GAFA, la confiance qui leur est accordée n'est pas toujours maximale.

Enfin, si le procédé met en jeu une intelligence artificielle, il est primordial qu'un temps conséquent d'entraînement soit accordé à l'outil. En effet, cela permettra d'éviter les dérives comme avec le bot Tay de Microsoft sur Twitter en 2016 [5].

6 Conception

6.1 Structure générale et choix de conception

6.1.1 Le choix du Cloud de Microsoft : Azure

La première étape de ce projet fut de choisir quels étaient les meilleurs outils de développement pour le produit voulu. Nous nous sommes très rapidement tournés vers Azure, le cloud de Microsoft pour plusieurs raisons :

- Figeac Aero est partenaire Microsoft. Ainsi, cela permet de rester dans la logique architecturale du système d'information de l'entreprise.
- La plateforme à laquelle le chatbot doit être intégré est Microsoft Teams, ainsi utiliser une plateforme Microsoft rend cette étape faisable en quelques clics.
- Le choix d'une plateforme Cloud permet que l'environnement d'exécution du chatbot soit entièrement géré par Azure. En effet, pas besoin d'installer le langage de programmation ni les bibliothèques nécessaires dans la machine, tout est géré sur des serveurs distants.
- Implémenter un chatbot via le cloud Azure laisse le choix de trois langages de programmation : Python, Javascript et C#. Nous avons privilégié le Python car nous avons tous les trois de l'expérience et cela nous permettait d'être directement opérationnel dans le développement du projet.
- Des outils Microsoft Azure pour créer des chatbots existent et fonctionnent très bien : le développement et le maintien est donc d'autant plus facile. Les différents blocs s'intègrent les uns aux autres sans difficulté et la plateforme est intuitive. De plus, ces outils sont gratuits pour une utilisation rudimentaire, ce qui les rends parfaits pour un POC. Si l'utilisation se globalise à toute l'entreprise, quelques coûts pourraient être générés mais ils resteront raisonnables, voire seront annulés grâce au partenariat Microsoft.

6.1.2 Notre utilisation d’Azure et de ses outils

Le client Azure pour la création et le déploiement

Tout au long de ce projet, et plus particulièrement pour apprendre à créer et déployer notre chatbot, nous avons suivi les tutoriels de la documentation Microsoft [2][3]. Ainsi, nous sommes partis d’un template Microsoft répétant les inputs de l’utilisateur pour avoir la base de notre chatbot. Ensuite, tout le test en local et le déploiement se faisaient via l’invite de commande Windows, grâce à la librairie Azure-CLI. La phase de déploiement va alors créer 3 ressources Azure, au sein d’un même groupe de ressources :

- un *App Service* : c’est le conteneur qui permet de créer et d’héberger entre autres des applications web, et ce sans gérer l’infrastructure ;
- un *App Service Plan* : il définit l’ensemble des ressources de calcul nécessaire à l’exécution de l’application ;
- un *Bot Channels Registration* : c’est la composante qui permet de faire de l’application un chatbot. A partir de celle-ci, on pourra le tester, lui ajouter de nouvelles fonctionnalités comme *Luis* [section 6.3] et *QnA* [section 6.4], et enfin l’intégrer aux applications de notre choix (ici Microsoft Teams).

De nouvelles ressources viendront rejoindre ces dernières lors de la création des services *Luis* [section 6.3] et *QnA* [section 6.4].

Le *Bot Framework Emulator* pour le test en local

Tout au long de ce projet, nous avons pu tester très facilement notre chatbot en local grâce au *Bot Framework Emulator* de Microsoft. Nous n’avons qu’à lancer le bot en ligne de commande, puis accéder à l’émulateur à l’aide de l’URL à laquelle le chatbot était rendu disponible. De fait, nous pouvions accéder aux erreurs très facilement via l’interface de la ligne de commande, tout en pouvant comprendre l’allure des paquets qui étaient échangés à chaque interaction. Vous pourrez trouver des exemples de conversation avec le chatbot via cet outil tout au long du rapport, notamment dans la partie 6.1.3 et via la figure 6.5.

6.1.3 Structure du chatbot

Nous vous invitons à lire cette section avec comme support visuel le schéma explicatif de la structure du chatbot, trouvable en annexe [ann. A.1]. Grâce à l’intégration de notre chatbot dans Teams, chaque utilisateur peut avoir accès à une conversation privée avec ce dernier. À chaque entrée de l’utilisateur dans cette conversation, les données sont donc envoyées au bot déployé sur Azure sous forme de JSON, qui va traiter la donnée selon ce que les fonctions Python vont lui indiquer. Dans un premier temps, les données sont envoyées au QnA (dont le fonctionnement est détaillé en partie 6.4). Si la question n’est pas trouvée dans la base de connaissance, le système va donc passer à Luis (dont le fonctionnement est détaillé en partie 6.3) qui va statuer si l’information est :

- une recherche de fichier : dans ce cas la suite du processus est expliquée en partie 6.2 ;
- une salutation : dans ce cas le chatbot répondra simplement à la salutation de l'utilisateur en lui demandant en quoi il peut lui être utile ;

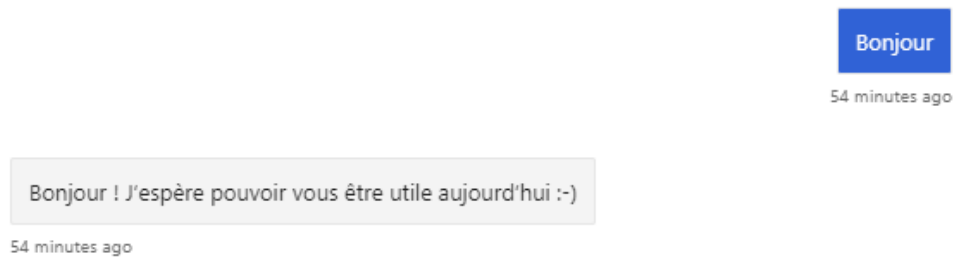


FIGURE 6.1 – Réaction du produit à une salutation

- un remerciement : dans ce cas le chatbot répondra simplement à ce remerciement ;

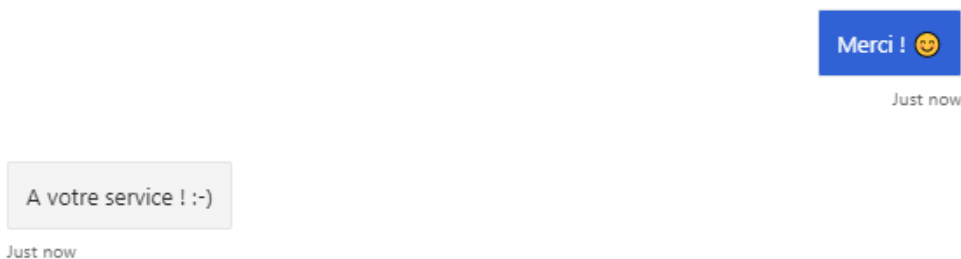


FIGURE 6.2 – Réaction du produit à un remerciement

- une demande concernant l'utilisation du chatbot : dans ce cas le chatbot renverra simplement un très court manuel expliquant son intérêt, et comment l'utiliser ;

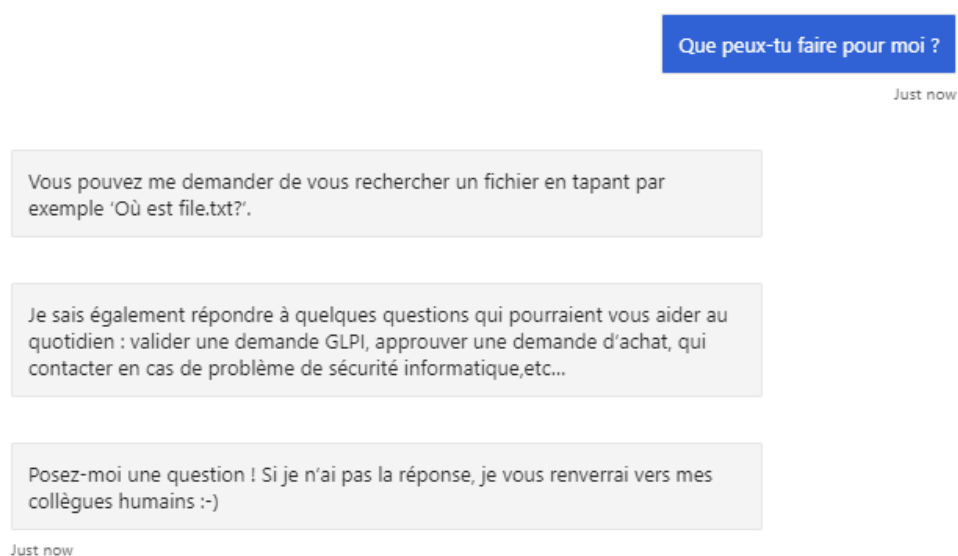


FIGURE 6.3 – Réaction du produit à une demande concernant son utilisation

- autre chose.

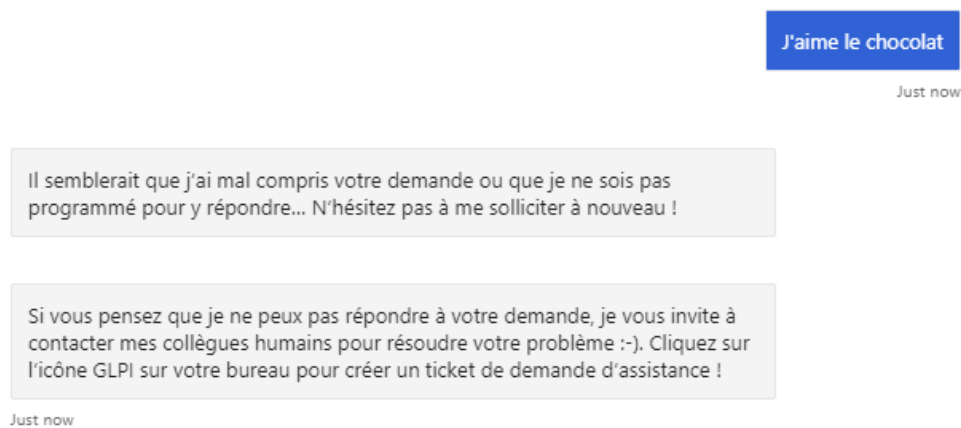


FIGURE 6.4 – Réaction du produit lorsqu'il ne reconnaît pas la demande formulée

6.1.4 Sécurisation de l'accès au chatbot

Dans un premier temps, nous avons utilisé le lien d'intégration à Microsoft Teams proposé par la plateforme Azure pour tester notre chatbot. Puis, nous nous sommes rapidement rendus compte qu'il aurait fallu communiquer ce lien à tous les employés de la société afin que la conversation avec notre chatbot puisse être créée. Cependant, cette méthode présentait une immense faille : si le lien se retrouvait de manière accidentelle entre les mains d'une tierce personne, elle pourrait avoir accès à la discussion avec le chatbot, et donc éventuellement pourrait créer une brèche pour récupérer des informations confidentielles.

Nous avons donc trouvé un moyen plus sécurisé d'intégrer le chatbot à la plateforme Teams de l'entreprise : la création d'une application personnelle depuis l'interface de Microsoft Teams. Il suffit alors de créer un manifeste de création d'application, relié au chatbot hébergé sur Azure grâce à l'identifiant de l'application. Une fois ce fichier et l'icône de l'application choisi, le tout doit être compressé puis déposer via Teams. Ainsi, tout membre du Teams en question pourra accéder à l'application, et même l'épingler pour la retrouver facilement.

6.2 Recherche de fichiers

Comme expliqué dans le cahier des charges, la tâche principale du chatbot est d'aider les employés de Figeac Aero à rechercher des fichiers parmi les différents contenants de l'entreprise. Nous avons donc trois axes de recherche : dans les fichiers Windows partagés, dans le SharePoint de l'entreprise et finalement au sein de leur Teams. L'utilisateur qui choisit alors de rechercher un fichier selon son nom aura comme réponse les liens d'accès vers les fichiers correspondants sur la totalité de ces plateformes.

Accès aux fichiers Windows

Une fois l'accès à l'arborescence établi, la recherche sur les noms des fichiers correspondants à l'input de l'utilisateur est très simple. On se place à la racine de l'arborescence et on explore chaque dossier de manière récursive. Chaque fichier correspondant est ajouté à un set de chemins, et une fois la totalité de l'arborescence explorée, on peut renvoyer les résultats correspondants. À noter qu'un fichier est considéré comme correspondant lorsque l'input de l'utilisateur est au moins inclus dans le nom du fichier considéré. Cette fonctionnalité est gérée à l'aide du module Python *re*, permettant de travailler avec les expressions régulières. De plus, le programme est fait de manière à ce que les fichiers soient comparés grâce au module Python *filecmp* pour éviter de renvoyer plusieurs fois le même fichier s'il est trouvé à divers endroits dans l'arborescence.

Malheureusement, quelques problèmes de temporalité ont fait que nous n'avons pas pu accéder à des bases de fichiers partagés Windows, ce qui fait que nous n'avons jamais pu tester cette fonctionnalité autrement qu'en local [fig. 6.5], ou alors une fois déployé, la recherche se fera parmi tous les fichiers contenus dans *archive.zip* évoqué dans le processus détaillé du déploiement [ann. F]. En revanche, de manière locale nous avons pu faire en sorte que lorsque notre chatbot trouve un seul résultat correspondant, alors il ouvre ce fichier avec l'application par défaut pour ce type de fichier de la machine.

Accès aux fichiers SharePoint

L'accès à la base de données SharePoint se fait via le module Python *Office365-REST-Python-Client*. Celui-ci permet de pénétrer dans la base de données dont l'URL est spécifiée dans le programme, selon les droits accordés au compte avec lequel on se connecte. Ainsi, un compte Microsoft sur le domaine de Figeac Aero a été créé spécialement pour l'occasion. Ce compte devra être conservé pendant toute la durée de vie du chatbot, et on lui a désactivé l'authentification à double facteur qui empêchait l'utilisation de l'API. Il suffit ensuite d'itérer parmi les items présents dans la base de données pour trouver le(s) fichier(s) voulu(s). Une nouvelle fois, nous utilisons les expressions régulières afin de maximiser les chances d'envoi du bon fichier lorsque l'utilisateur ne se souvient que partiellement du nom de ce qu'il recherche.

Si la recherche de l'utilisateur aboutit à 3 fichiers ou moins, alors les résultats seront proposés avec un lien d'accès direct. Dans le cas inverse, tous les fichiers trouvés seront proposés, et on incitera l'utilisateur à effectuer une nouvelle requête plus précise.

Accès aux fichiers Microsoft Teams

L'architecture de Microsoft Teams fait que lorsque l'on upload un fichier via l'interface, alors il va être stocké dans un SharePoint propre à ce Teams. Ainsi, la démarche est la même que pour le point 6.2, seule l'URL d'accès étant modifiée.

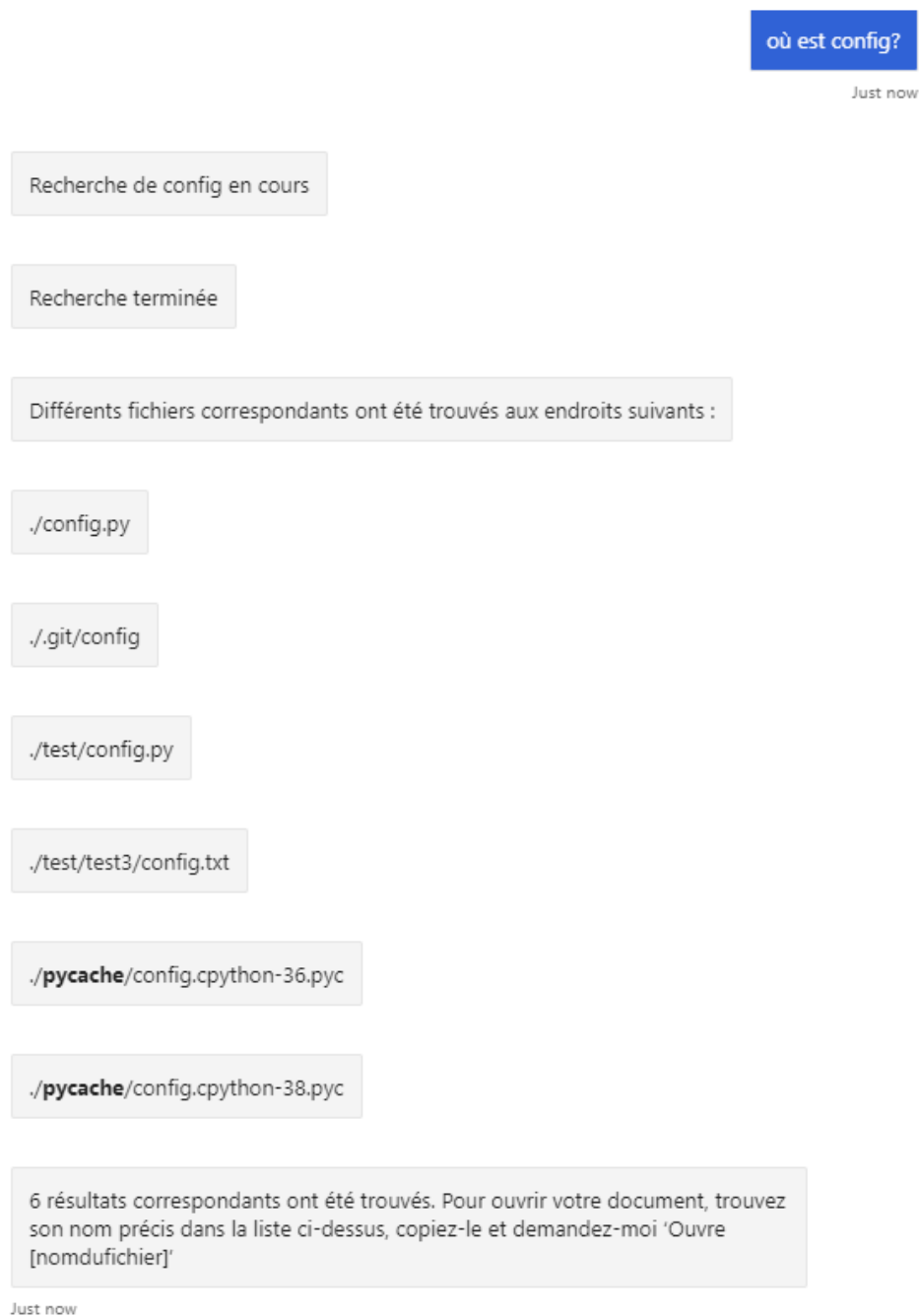


FIGURE 6.5 – Exécution d'une recherche de fichiers dans une arborescence Windows locale

6.3 Luis

6.3.1 Principe

Luis est le nom donné au service d'intelligence artificielle conversationnelle de Microsoft [4]. Grâce au machine learning, il peut analyser le sens du langage naturel et en extraire les informations voulues, comme dans notre cas le nom des fichiers à rechercher par exemple. Le principe est assez simple :

- Tout d'abord on crée les intentions, c'est-à-dire les catégories regroupant ce que l'utilisateur pourrait vouloir signifier. Dans notre cas, nous en avons 5 : la salutation, le remerciement, la demande d'aide quant à l'utilisation du chatbot, la recherche de fichier [fig. 6.6] et enfin tout le reste. Pour chacun de ces cas, nous devons fournir à Luis un pannel de formulations assez larges pour qu'il ne puisse pas se tromper d'intention. Luis apprendra de lui-même de nouvelles formulations au fur et à mesure de l'utilisation du chatbot, mais ce procédé est sur le moyen-terme et donc il faut être scrupuleux dès le départ pour ne pas orienter le produit dans la mauvaise direction.



FIGURE 6.6 – Exemple de différentes formulations pour l'intention de recherche de fichiers, avec ses entités.

- Si nécessaire, on peut repérer des entités dans une intention. C'est-à-dire que l'on va créer une "catégorie" de mots, par exemple les fichiers, et les annoter dans les différentes formulations que l'on a fourni à Luis pour l'intention "Recherche de fichiers". Une fois de plus, il faut être le plus exhaustif possible dès le départ dans ce qu'il est possible de proposer pour ne pas perdre le chatbot. À noter qu'une entité peut être découpée en sous-entités, comme dans notre cas le nom du fichier et son extension [fig. 6.6].
- Lorsque Luis détecte une intention et potentiellement ses entités associées, il les envoie au chatbot sous forme d'un .JSON pour que les informations déduites de l'input de l'utilisateur puissent être traitées.

6.3.2 Intégration du service à notre chatbot

Lors de la création et de la publication du service Luis, 2 nouvelles ressources sont créées dans le groupe hébergeant la base du chatbot [1] :

- une ressource de création nous permettant de créer, gérer, entraîner, tester et publier nos applications Luis ;
- une ressource de prédiction nous permettant d'interroger les demandes de point de terminaison de prédiction (dans notre cas Microsoft Teams). Grâce à l'option de tarification F0 (gratuite) de cette ressource, il est possible de faire jusqu'à 10 000 requêtes par mois, ce qui est idéal dans le cadre d'un POC.

Il suffit alors de renseigner les identifiants et clés associées à notre Luis au sein de notre code Python faisant tourner notre chatbot afin de pouvoir utiliser le service.

6.4 QnA

6.4.1 Principe

Si l'utilisateur ne souhaite pas rechercher un fichier mais poser une question au chatbot, alors peut être que sa réponse se trouve dans le service Question and Answers (QnA). En effet, Azure nous donne la possibilité de créer des bases de connaissances pour stocker toutes les questions/-réponses que l'on souhaite. On peut donc y renseigner toutes les questions les plus fréquemment posées par les utilisateurs pour que le chatbot puisse y répondre automatiquement : cela représente un gain de temps considérable pour l'utilisateur et pour les équipes Helpdesk qui se retrouvent déchargées.

De plus, Il est possible d'entraîner la base de connaissance en lui indiquant si les réponses qu'elle fournit sont satisfaisantes ou bien en lui rectifiant les erreurs qu'elle commet. Tout cela fonctionne avec un système d'indice de confiance associé à chaque réponse : le chatbot renvoie la réponse avec l'indice de confiance le plus élevé en fonction de la question qui a été posée par l'utilisateur.

Via les figures 6.7 et 6.8, on constate bien que la base de connaissance associe deux indices de confiance différents pour les deux réponses qu'elle a retenues, à nous donc de choisir quelle est la plus appropriée pour que la base puisse apprendre de ses erreurs et modifier les indices de confiance en conséquence. Il suffit ensuite de sauvegarder la base et les modifications seront instantanément effectives dans tous les bots reliés à celle-ci, que ce soit dans un bot en local ou bien dans un bot déployé sur un canal comme Microsoft Teams.

Ce service de QnA est totalement indépendant du code et est stocké dans le cloud Azure. Il est donc possible de l'associer à différents bots ou bien de choisir quelle base de connaissances doit être associée au chatbot. De plus, le service est très simple d'utilisation notamment grâce à son interface très claire et "user friendly", il pourra donc être aisément pris en main par les équipes de Figeac Aero lorsque le chatbot sera déployé chez eux [fig. 6.9].

C'est donc de cette manière que nous pouvons renseigner autant de questions/réponses que nous souhaitons. Notre encadrant nous a d'ailleurs pour cela fourni un document référençant un grand nombre de ces questions/réponses qu'il souhaitait voir répertoriées dans notre base de connaissance.

La base de connaissance se crée donc directement sur Azure et on peut la remplir d'autant de questions / réponses que l'on souhaite. Dans notre cas, nous avons rempli la base avec une liste de questions fournie par l'encadrant industriel et nous l'avons ensuite testée pour voir ses limites et éventuellement appliquer quelques correctifs si certaines réponses n'étaient pas pertinentes.

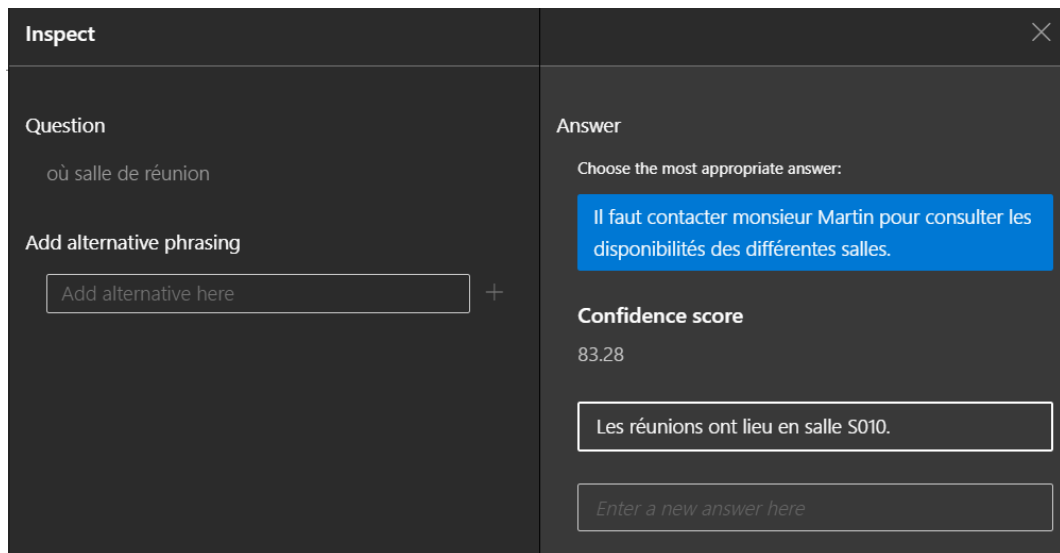


FIGURE 6.7 – Indice de confiance de la première réponse

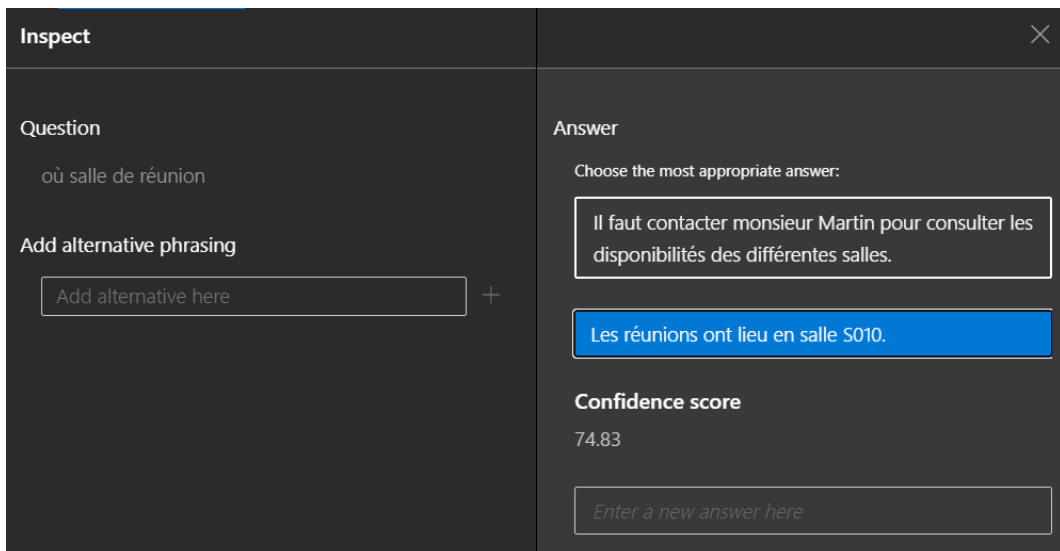


FIGURE 6.8 – Indice de confiance de la deuxième réponse

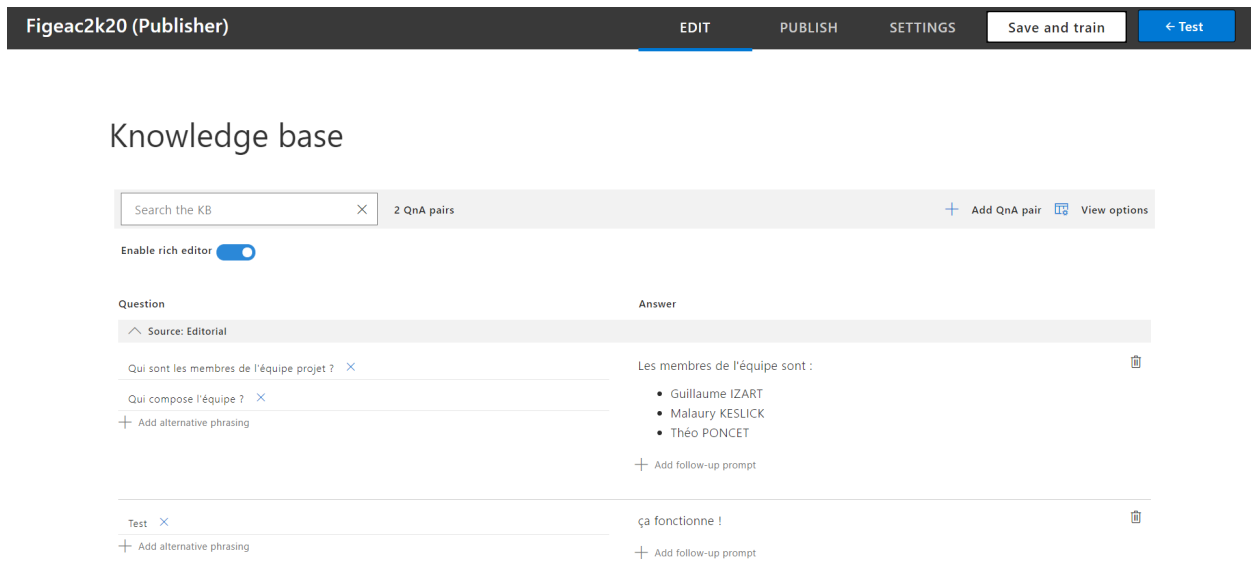


FIGURE 6.9 – Interface d'édition de la base de connaissances

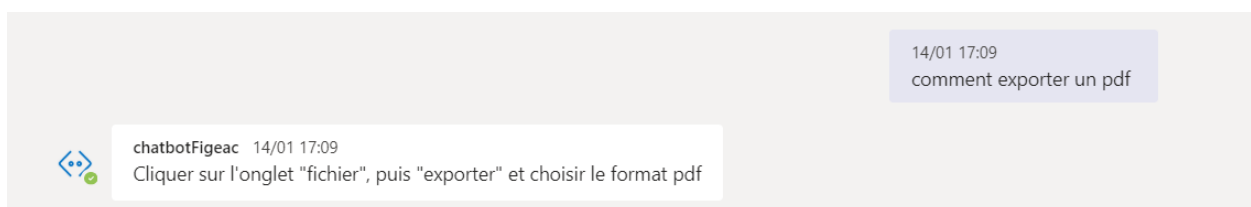


FIGURE 6.10 – Exemple de question/réponse

6.4.2 Intégration du service à notre chatbot

De la même manière que lors des premières étapes de la mise en place du chatbot sur Azure, des ressources sont générées lors de la création du système de QnA :

- un *App Service* : encore une fois, un conteneur est nécessaire pour héberger l'application QnA nouvellement créée ;
- un *App Service Plan* : de la même manière que précédemment, cette ressource est nécessaire pour contenir les ressources de calcul nécessaires au bon fonctionnement de l'application ;
- un *Search Service* : c'est le service cognitive qui sera responsable de la recherche dans la base de connaissances ;
- un *Cognitive Service* : enfin est la ressource dans laquelle on pourra récupérer nos clés d'abonnement.

Ces quatre ressources sont donc générées automatiquement lors de la création du service QnA. Au total, c'est donc 9 ressources différentes qui auront été créées pour permettre au chatbot de fonctionner comme précédemment décrit. À noter qu'il est possible d'avoir des ressources de type *Application Insight* en plus, qui collecteront des données de télémétrie sur le trafic, et conserveront les journaux d'activité de conversations et les erreurs.

7 Méthodologie

7.1 Réunions

Dès le début du projet, nous avons choisi en accord avec notre encadrant industriel de nous réunir une fois par semaine. Ainsi, sauf exceptions (vacances, avancée peu illustrable, indisponibilité de l'un des partis), nous avons pu discuter de l'avancée du projet tous les jeudis entre 17h et 18h. François Charoy nous rejoignait ponctuellement afin de prendre de nos nouvelles. Vous trouverez les comptes-rendus les plus intéressants en annexe [section G].

7.2 Roadmap du projet

A la fin de la phase de recherche sur l'état de l'art des chatbots, et au début de la phase de développement de l'application, l'encadrant industriel Loïc verney, ainsi que Simon Auroy ont demandé à ce qu'une roadmap pour le reste de la phase de développement soit rédigée.

Cette roadmap avait pour but premier de donner une ligne directrice des actions à réaliser, avec une priorisation des tâches.

Aussi, cette roadmap avait été réalisé afin de rassurer les encadrants industriels quant à l'avancement du projet.

Cette roadmap reprend donc les grandes phases du projet avec un ordonnancement chronologique à mettre en parallèle du diagramme de Gantt réalisé [ann. E.1].

Ces deux outils de gestion de projet permettent d'avoir un ligne directrice temporelle pour évoluer au cours du projet.

roadmap peut être retrouvée dans les annexes [ann. D].

7.3 Affectation des postes

Une réunion de pré-projet à été réalisée avec les membres de l'équipe projet afin d'affecter les rôles de Chef de Projet, de secrétaire, et de consultant technique.

Malaury Keslick s'est proposé pour le poste de secrétaire afin d'assurer la communication avec les différentes parties prenantes du projet, ainsi que de rédiger un compte-rendu à la fin de chaque réunion pour permettre un suivi du projet.

Guillaume Izart s'est proposé pour le poste de consultant technique du groupe, afin d'aider les autres membres projet si ils rencontrent un problème technique.

Théo Poncet s'est proposé pour le poste de chef de projet, afin de lier les membres de l'équipe projet à l'encadrant industriel, ainsi que d'animer les réunions hebdomadaire. Il a aussi été chargé de produire des éléments de gestion de projet nécessaires au suivi de l'encadrant industriel.

7.4 Gestion du projet

Le suivi de l'évolution du projet à été réalisé grâce à un Gantt mis à jour chaque semaine, montrant l'évolution de chaque tâche au sein du projet [ann. E.1]. Nous utilisons ce document avec les encadrants industriels afin de discuter du temps imparti pour chacune des tâches et ainsi avoir une meilleure planification du projet. Cette méthode à permis à l'encadrant industriel de visualiser l'avancement du projet, ainsi que d'apporter des remarques constructives d'après son expérience. Par exemple, Loïc Verney à pu nous faire ré-estimer le temps alloué au déploiement du ChatBot sur les serveurs de Figeac Aero, afin d'avoir une plus grande marge de manoeuvre sur la fin de projet.

Au milieu du projet, une RoadMap à été demandée à l'équipe projet pour avoir une vision plus clair sur les tâches restantes jusqu'à la clôture du projet. Une réunion à été dédiée à l'analyse de cette RoadMap, et des modifications ont été apportées par l'encadrant industriel afin de permettre de réaliser au mieux les objectifs de l'entreprise.

Ce diagramme de Gantt est trouvable en annexe de ce document [ann. E.1]

7.5 Livrables

- **Le cahier des charges [ann. C]** : l'équipe projet à rédigé un cahier des charges en adéquation avec les besoins de l'encadrant industriel. Ce cahier des charges a servi de ligne directrice pour les étapes du projet, et a permis d'homogénéiser les attentes de Loïc Verney avec les idées de l'équipe projet.
- **La RoadMap [ann. D]** : À mi-projet, une roadmap à été demandé par l'encadrant industriel pour permettre une meilleure visualisation des tâches restantes, ainsi que des repères

temporels pour les grosses étapes du projet.

- **Éléments majeurs de gestion de projet** : Un compte-rendu de chaque réunion était rédigé et envoyé sur le Teams du projet.

Aussi, un Gantt a été réalisé et était mis à jour à chaque réunion pour permettre de visualiser l'avancement du projet par rapport au temps restant, et donc de savoir si nous étions dans les temps ou non.

- **Le Proof Of Concept** : Le but du projet était de réaliser un chatbot intégré sur un canal Microsoft Teams. Ce chatbot est donc le livrable le plus important et est celui qui a nécessité le plus de temps au cours de ce projet.

Ce chatbot intègre donc un module QuestionAnswers, un module d'intelligence artificielle permettant de mieux comprendre la demande utilisateur, ainsi que des modules de recherches de document dans différentes sources de données, comme un serveur Windows, un SharePoint ou encore un Teams. Ce chatbot est déployé sur un cloud Azure.

- **La fiche technique du chatbot [ann. F]** : Cette fiche reprend la conception du chatbot, ainsi que les manipulations nécessaires permettant par exemple de déployer le chatbot, ou encore de modifier les bases de connaissances du chatbot.

8 Bilan

8.1 Difficultés rencontrées

8.1.1 Difficultés environnementales

Tout d’abord, ce projet s’est inscrit dans un contexte atypique à cause de la pandémie mondiale.

La COVID-19 a donc contraint les parties prenantes du projet à, tout d’abord, restreindre les contacts interpersonnels, puis à les arrêter totalement suite au confinement. Ceci a impacté d’une manière significative les méthodes de travail de l’équipe projet.

En effet, il n’était plus possible aux membres de l’équipe de se réunir, de travailler ensemble, ou encore avoir des rencontres physiques avec les encadrants académique (ce qui peut influencer sur la qualité de l’échange). Ces conditions de travail ont particulièrement affecté certains membres du groupe projet, car ce travail en distanciel impactait la motivation de chacun, et donc la productivité des membres.

Nous avons alors dû compenser ce manque de temps en retirant des fonctionnalités prévues au départ lors de l’élaboration du cahier des charges. Grâce à la méthodologie AGILE, ces fonctionnalités manquantes n’affectent cependant pas le fonctionnement de base du chatbot, et sont donc des fonctionnalités qui pourront être aisément rajoutées à l’avenir.

8.1.2 Difficultés techniques

Notre projet a débuté avec une longue phase de compréhension du besoin de l’encadrant, ainsi qu’un état de l’art des chatbots.

Cette phase avait pour but d’établir ce qui existait, ce qui était utile pour le besoin de l’industriel, et ce qui était réalisable dans le temps qui nous était imparti pour ce projet afin de réaliser un cahier des charges en adéquation avec le besoin client et avec les possibilités de l’équipe.

Une fois que nous nous étions accordés avec Figeac Aero sur le besoin, la plupart des membres

du groupe ont dû commencer à apprendre le fonctionnement d'un chatbot, car nous n'en avons jamais réalisé auparavant.

Aussi, une difficulté supplémentaire était la prise en main du cloud Azure, qui devait servir de support à notre chatbot Teams.

Nous avons été particulièrement freiné par le besoin de recréer un projet tous les mois, car nous étions sur une licence gratuite, permettant d'accéder aux fonctionnalités nécessaires comme Luis ou le QA, mais nous avions une durée d'expiration cette licence.

Enfin, une des difficulté majeure à été de faire coïncider tous les éléments du chatbot déployé, et de les faire fonctionner et de transférer ensuite le chatbot sur les infrastructures de Figea Aéro .

8.2 Heures effectuées

	Malaury Keslick	Guillaume Izart	Théo Poncet	Total
Phase 1 : Etude des chatBots	30	20	20	70
Phase 2 : développement ChatBot	120	105	65	290
Gestion du projet	10	10	20	40
Note de cadrage	10	10	10	30
Cahier des charges	2	8	8	19
RoadMap	2	7	10	19
Rapport	25	20	35	80
Soutenance	10	10	17	37
Préparation de réunions/ réunions	25	20	30	75
Total du nombre d'heures effectuées	234	210	215	659

8.3 Futur du produit

Lors de ce projet, nous n'avons pas pu apporter toutes les fonctionnalités souhaitées au chatbot, faute de temps.

En effet, la notion de sécurité n'a pas pu être traitée suffisamment pour assurer un seuil minimum de sécurité permettant d'échanger des informations confidentielles sur le canal du chatbot. C'est en cela que se trouverait un des axes d'amélioration du chatbot dans le futur.

Aussi, la notion de hiérarchie n'a pas eu le temps d'être mise en place. Si le POC convainc l'entreprise, il faudrait alors qu'ils explorent la fonctionnalité qui laisserait ne laisse l'accès qu'aux documents dont l'utilisateur dispose au moins des droits de lecture.

Enfin, le chatbot aura besoin de continuer à évoluer en continuant d'ajouter par exemple des questions standardisées au module Q&A, permettant ainsi un "apprentissage" continu.

8.4 Conclusion

Le but de ce projet industriel était de réaliser le POC d'un chatbot d'entreprise pour assistance helpdesk. Celui-ci devait être capable de répondre à environ 5% des questions les plus courantes des employés ainsi que de rechercher des fichiers sur les différents serveurs de l'entreprise, que ce soit des serveurs de fichiers classiques ou bien des serveurs Sharepoint.

Nous avons donc pu concevoir un bot comportant certaines de ces fonctionnalités mais avons été limité par les outils à notre disposition. En effet, nous ne disposions pas des licences Azure et Sharepoint et nous avons donc du composer avec les versions gratuites de ces différents logiciels. Nous avons cependant pu déployer le bot sur les différents serveurs de Figeac Aero et donc leur fournir la solution attendue.

Ce POC n'est cependant qu'un début et pourrait donner lieu à beaucoup d'améliorations : en effet, il serait éventuellement possible de mettre en place une composante d'intelligence artificielle et d'apprentissage continu. De cette façon, le bot pourrait s'améliorer en apprenant de ses erreurs de manière autonome là où cela nécessite aujourd'hui l'intervention de l'administrateur.

Ce projet a donc été l'occasion pour nous de travailler en collaboration avec une entreprise et de découvrir le monde du travail via une approche nouvelle. Malgré la crise sanitaire qui a rendu les interactions difficiles, nous avons tout de même réussi à mener le travail à bien.

Au sein de l'équipe, l'ambiance de travail a été plus que satisfaisante, et les interactions avec les parties prenantes étaient productives.

L'objectif du POC est de pouvoir prouver l'intérêt technique d'une solution. Ici, le résultat est positif selon l'encadrant industriel. Il conviendra ensuite de décider si Figeac Aero souhaite continuer à développer son propre chatbot, ou si ils iront vers des chatbots présents sur le marché. Le POC a donc permis aux décideurs de visualiser rapidement l'intérêt de la technologie.

Nous sommes globalement satisfait du travail qui a été effectué au cours de ce projet industriel avec Figeac Aéro, et nous les remercions de s'être mis dans le rôle d'un client pour se rapprocher au maximum des interactions dans une entreprise lors d'un projet.

Bibliographie / Webographie

- [1] 09 2020. 19
- [2] Didacticiel : créer un bot de base. 10 2020. 13
- [3] Déployer votre bot. 06 2020. 13
- [4] Qu'est-ce que le service language understanding (luis)? 11 2020. 18
- [5] Anne Brigaudeau. Tay, le robot de microsoft, quitte twitter après des dérapages racistes. 03 2016. 11
- [6] Futura. Chatbot : qu'est-ce que c'est? 6
- [7] OC. Le fonctionnement et les avantages du chatbot. 8
- [8] Adrien Rivierre. Assistants vocaux : quand les marques donnent de la voix. 01 2018. 6

Liste des illustrations

1.1	Exemple de question qu'un utilisateur peut demander au chatbot	1
5.1	Les assistants vocaux respectifs d'Amazon, d'Apple, de Google et de Microsoft . .	6
5.2	Chatbot de Mastercard, accessible via Messenger	7
5.3	VBot, le chatbot de la SNCF accessible via Messenger	7
5.4	Exemple d'usage d'un chatbot interne à une entreprise	8
5.5	Exemple de chatbot au sein d'une conversation multi-utilisateurs développé dans le cadre du stage de deuxième année de Malaury Keslick	10
5.6	Plateformes des géants de la technologie Cloud	11
6.1	Réaction du produit à une salutation	14
6.2	Réaction du produit à un remerciement	14
6.3	Réaction du produit à une demande concernant son utilisation	14
6.4	Réaction du produit lorsqu'il ne reconnaît pas la demande formulée	15
6.5	Exécution d'une recherche de fichiers dans une arborescence Windows locale . .	17
6.6	Exemple de différentes formulations pour l'intention de recherche de fichiers, avec ses entités.	18
6.7	Indice de confiance de la première réponse	21
6.8	Indice de confiance de la deuxième réponse	21
6.9	Interface d'édition de la base de connaissances	22
6.10	Exemple de question/réponse	22
A.1	Annexe 1 : Schéma récapitulatif de la conception du chatbot	35
E.1	Première partie diagramme de gantt effectif du projet PI	54
E.2	Seconde partie diagramme de gantt effectif projet PI	55

9 Glossaire

-*ChatBot* : Un chatbot est un robot logiciel pouvant dialoguer avec un individu via un service de conversation automatisé

-*SharePoint* : SharePoint est un logiciel produit par Microsoft offrant les fonctionnalités de gestion de contenu, la gestion électronique de documents, les forums, la possibilité de créer des formulaires et des statistiques décisionnelles.

-*Teams* : Teams est une application de communication collaborative produit par Microsoft

-*Cloud* : Le Cloud fournit de l'espace de stockage, de la puissance de calcul et des logiciels exécutables dans un centre de données distant. Nous utilisons le Cloud Azure de Microsoft pour notre projet

-*POC* : Proof of Concept, preuve de concept permettant de démontrer l'existence d'une opportunité (on parle également de "désirabilité") et/ou la faisabilité d'un système

Annexes

A Schéma illustratif

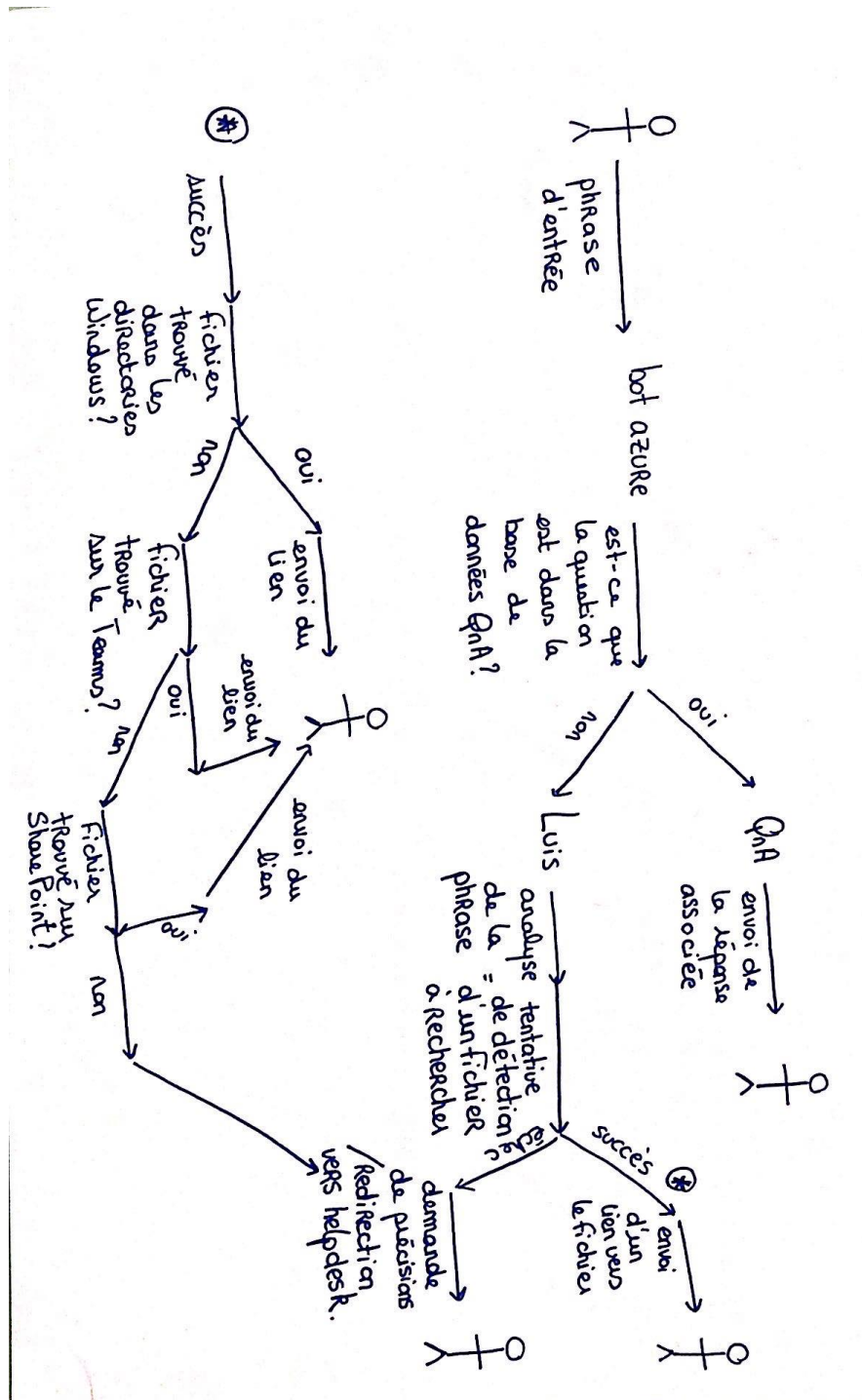


FIGURE A.1 – **Annexe 1** : Schéma récapitulatif de la conception du chatbot

B Note de cadrage

Projet Industriel
Telecom Nancy - Figeac Aero

Note de cadrage

POC - Mise en place d'un chatbot entreprise
pour Assistance Helpdesk IT

POC - Enterprise chatbot development for IT Helpdesk assistance



Encadrant universitaire :
François CHAROY

Equipe projet :
chef de projet

Théo PONCET theo.poncet@telecomnancy.eu

Encadrant industriel :
Loïc VERNEY

Guillaume IZART guillaume.izart@telecomnancy.eu
Malaury KESLICK malaury.keslick@telecomnancy.eu

2020-2021

Contexte et Objectif

L'entreprise dispose d'une documentation technique hétérogène qui rend fastidieuse la recherche d'une information particulière. Actuellement, l'utilisateur va avoir tendance à créer un ticket helpdesk ou encore effectuer un appel téléphonique afin de trouver l'information souhaitée, ce qui surcharge les équipes helpdesk qui deviennent donc moins efficaces.

C'est donc de cette problématique qu'est née l'idée du projet proposé par Figeac Aero. Il s'agit d'étudier l'intérêt d'un chatbot pour l'entreprise. Celui-ci doit pouvoir permettre aux utilisateurs d'avoir un accès rapide à l'information. Le but est donc de rendre autonome l'utilisateur et ainsi faire gagner du temps de travail "efficace" aux équipes helpdesk. Enfin, il pourra éventuellement créer automatiquement des tickets helpdesk avec les informations déjà données par l'utilisateur.

Il faudra donc explorer les différentes possibilités au regard du contexte technique et économique, puis choisir la meilleure solution à développer en accord avec l'encadrant industriel. Pour cela, nous devons rédiger un POC (Proof Of Concept) de la solution retenue afin de justifier son intérêt pour l'entreprise.

Contraintes/Risques

Contrainte temporelle : Nous avons besoin d'une grande partie du temps accordé au projet pour de l'auto-formation sur les chatbots, ainsi que sur la technologie qui sera sélectionnée pour le développement du POC. Ceci ne laisse pas beaucoup de temps pour développer l'application, et cela risque de nuire à la qualité du code produit.

Contrainte/Risque technique : Besoin impératif d'un POC à la fin du projet, ce qui peut déborder sur le temps accordé à la recherche de l'état de l'art sur les chatbots.

Le choix du langage de programmation appartient à l'équipe projet pour le développement du chatbot.

Les licences utilisées devront être majoritairement gratuites et libres d'accès, sauf besoin spécifique, qui demandera l'approbation du tuteur industriel via une justification du choix technique pour appuyer la demande. Dans la mesure du possible utiliser des licences Microsoft.

Contrainte physique : Étant donné les conditions sanitaires actuelles, le travail à distance offre moins de confort et de possibilités d'entraide que le travail en présentiel. Ceci pourrait impliquer une baisse de productivité de l'équipe projet.

Contrainte administrative : Les informations manipulées durant le projet seront soumises à un NDA qui devra être signé par l'équipe projet. La propriété du code final sera conservée par l'équipe projet, l'entreprise sera autorisée à utiliser le travail produit pour un besoin interne.

Organisation et Communication

Une méthode agile a été choisie pour menée à bien ce projet. De manière hebdomadaire, nous nous entretiendrons avec Mr. VERNEY et Mr. AUROY (et Mr. CHAROY lorsqu'il le souhaitera) via visio-conférence sur Microsoft Teams afin d'expliquer notre avancée et de nous fixer de nouveaux objectifs pour la semaine suivante. Cela nous permettra d'être en permanence certains d'aller vers une solution qui convient à l'entreprise ou de nous ré-orienter en cas de besoin. Des comptes-rendus seront rédigés après chaque réunion par la secrétaire du groupe, Malaury KESLICK, avant d'être envoyés aux membres du groupe et aux encadrants et publiés sur la plateforme Teams. Pour toute demande de renseignements entre les réunions, les échanges par mail seront privilégiés selon la préférence de Mr. VERNEY.

Dans un premier temps, nos rapports concernant les différentes recherches jusqu'à l'écriture du POC seront également partagés sur Teams. Ensuite, lorsque nous commenceront la partie implémentation du chatbot, le code et les ressources nécessaires seront partagées sur une plateforme Gitlab. Un README permettra de suivre l'évolution du développement.

Livrables

Le livrable d'avant projet est une note de cadrage qui permet d'accorder les idées du tuteur industriel avec celles de l'équipe projet. En outre, ce document sert à définir la portée globale du projet.

Le premier livrable du projet sera un cahier des charges reprennant les fonctionnalités requises de l'application. Aussi, il y sera détaillé les besoin de l'entreprise pour le chatbot.

Le principal livrable du projet sera un POC de la solution retenue. Celui-ci devra résumer l'intérêt et la faisabilité d'un chatbot pour l'entreprise et permettra de déterminer si oui ou non l'entreprise doit se lancer dans son développement.

Enfin, nous devons également rédiger une fiche technique à destination de l'entreprise pour permettre une bonne reprise du projet. Nous y détaillerons notre travail afin de rendre la tâche plus simple pour nos successeurs sur ce projet.

Livrable	Echéance	Priorité
Note de cadrage	23 Octobre 2020	Obligatoire
Cahier des Charges	1 Novembre 2020	Obligatoire
Fiche technique	15 Janvier 2021	Obligatoire
Proof Of Concept	15 Janvier 2021	Obligatoire
Fonctionnalités : IA, multi-plateforme	15 Janvier 2021	Optionnel

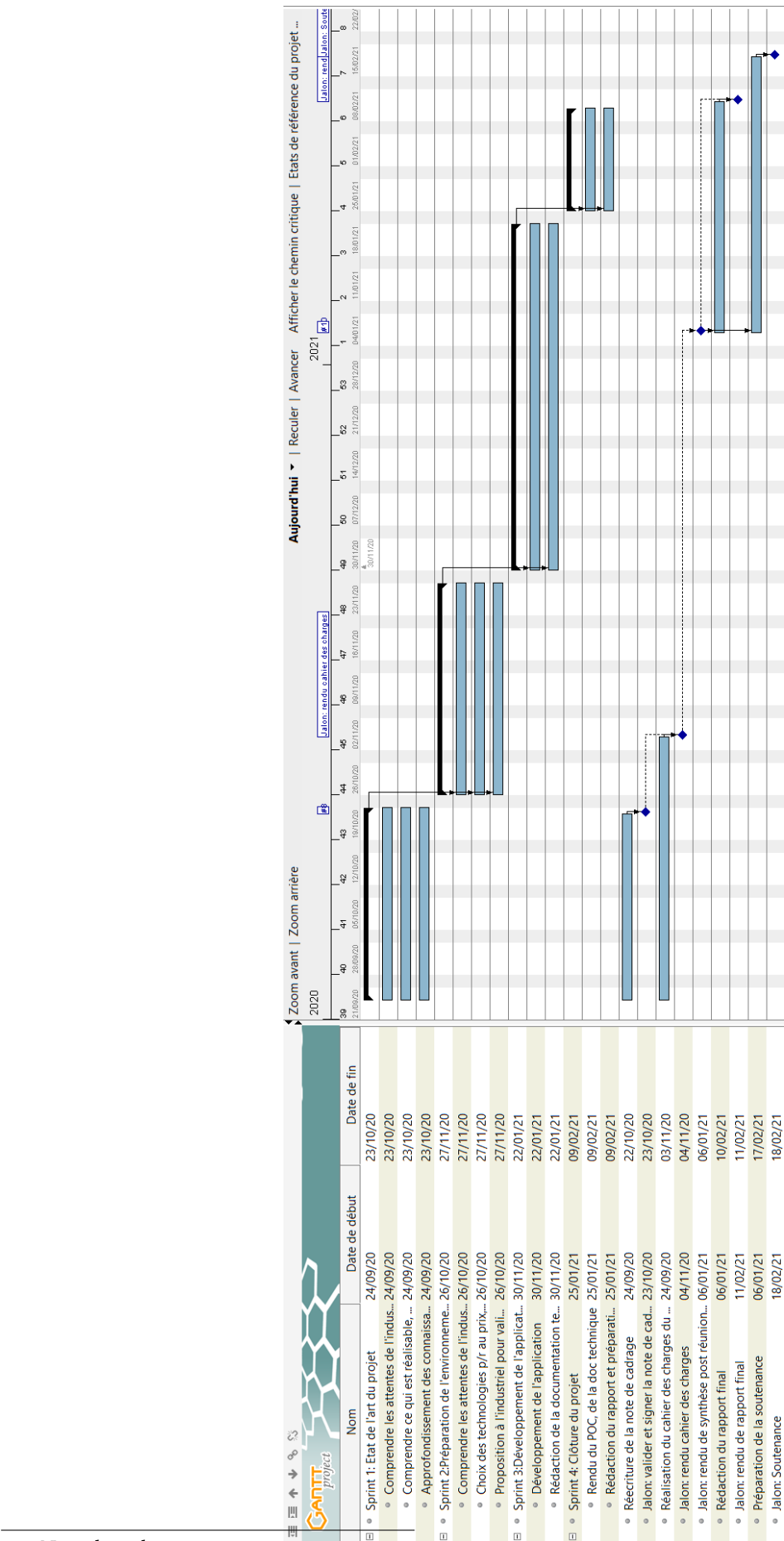
Encadrant Industriel
Lu et approuvé le :

L'équipe projet
Lu et approuvé le :

Encadrant Universitaire
Lu et approuvé le :

Note de cadrage

0.1 Annexe 1 : Diagramme de Gantt du projet



Note de cadrage

C Cahier des charges

Contents

1	Introduction	3
1.1	Contexte et objectifs	3
1.2	Les acteurs	3
1.3	Environnement opérationnel	4
2	Besoins utilisateurs	5
2.1	Besoins fonctionnels	5
2.2	Besoins non fonctionnels	5
3	Besoins système	6
3.1	Besoins système fonctionnels	6
3.2	Besoins système non fonctionnels	9
3.2.1	Performance	9
3.2.2	Sécurité	9

1 Introduction

1.1 Contexte et objectifs

Figeac Aero dispose d'une documentation technique hétérogène qui rend fastidieuse la recherche d'une information particulière. Actuellement, l'utilisateur va avoir tendance à créer un ticket helpdesk ou encore effectuer un appel téléphonique afin de trouver l'information souhaitée, ce qui surcharge les équipes helpdesk qui deviennent donc moins efficaces.

C'est donc de cette problématique qu'est née l'idée du projet proposé par Figeac Aero. Il s'agit d'étudier l'intérêt d'un chatbot pour l'entreprise. Celui-ci doit pouvoir permettre aux utilisateurs d'avoir un accès rapide à l'information. Le but est donc de rendre autonome l'utilisateur et ainsi faire gagner du temps de travail "efficace" aux équipes helpdesk.

Enfin, il pourra éventuellement créer automatiquement des tickets helpdesk avec les informations déjà données par l'utilisateur. Il faudra donc explorer les différentes possibilités au regard du contexte technique et économique, puis choisir la meilleure solution à développer en accord avec l'encadrant industriel. Pour cela, nous devons rédiger un POC (Proof Of Concept) de la solution retenue afin de justifier son intérêt pour l'entreprise.

1.2 Les acteurs

Dans notre système, les acteurs se résumeront aux différents utilisateurs du chatbot ainsi qu'au chatbot lui-même.

Ce chatbot sera développé par l'équipe projet constitué de Guillaume Izart, Malaury Keslick et Théo PONCET, supervisé par l'encadrant académique.

Le projet a été proposé et sera supervisé par les encadrants industriels: Loïc VERNEY et Simon AUROY.

1.3 Environnement opérationnel

Dans le cadre de ce projet, nous allons utiliser les outils suivants :

- Microsoft Azure pour le cloud, qui propose plusieurs API gratuites permettant entre autres d'accéder à SharePoint et Microsoft Teams. Il existera deux environnements Azure : un pour le test et un opérationnel lié à l'entreprise.
- Un environnement Python pour le développement du chatbot.
- Des bases de données utilisées par l'entreprise.
- Une base de données SharePoint pour stocker les échanges avec le chatbot.

2 Besoins utilisateurs

2.1 Besoins fonctionnels

Besoin 1 [Haute] L'utilisateur doit pouvoir écrire un message dans l'IHM du chatbot.

Besoin 2 [Moyenne] L'utilisateur doit pouvoir poser une question au chatbot.

Besoin 3 [Moyenne] L'utilisateur doit pouvoir répondre à une question posée par le chatbot.

Besoin 4 [Moyenne] L'utilisateur doit pouvoir être redirigé vers les équipes helpdesk lorsque sa question n'a pas trouvé de réponse.

2.2 Besoins non fonctionnels

Besoin 5 [Moyenne] L'utilisateur doit avoir accès à un channel privé (ou au moins une anonymisation de la discussion) dédié aux discussions avec le chatbot.

Besoin 6 [Moyenne] L'utilisateur doit avoir accès au chatbot 100% du temps sur la plage ouvrée de l'entreprise.

3 Besoins système

3.1 Besoins système fonctionnels

Besoin 7 [Haute] Le système doit pouvoir se connecter à des directory Windows, SharePoint et Microsoft Teams.

Besoin 8 [Haute] Le système doit pouvoir se connecter à une base de donnée (GLPI), et y rechercher de l'information.(wiki,publication, fichier)

Besoin 9 [Moyenne] Le système doit pouvoir comprendre des synonymes.

Besoin 10 [Moyenne] Le système doit pouvoir comprendre la requête malgré des fautes de frappe.

Besoin 11 [Moyenne] Le système doit pouvoir être sensible à la casse (Ne pas faire de différence entre les lettres majuscules et minuscules, ni entre les lettres avec et sans accents).

Besoin 12 [Moyenne] Le système doit pouvoir poser des questions simples à l'utilisateur afin d'affiner sa requête.

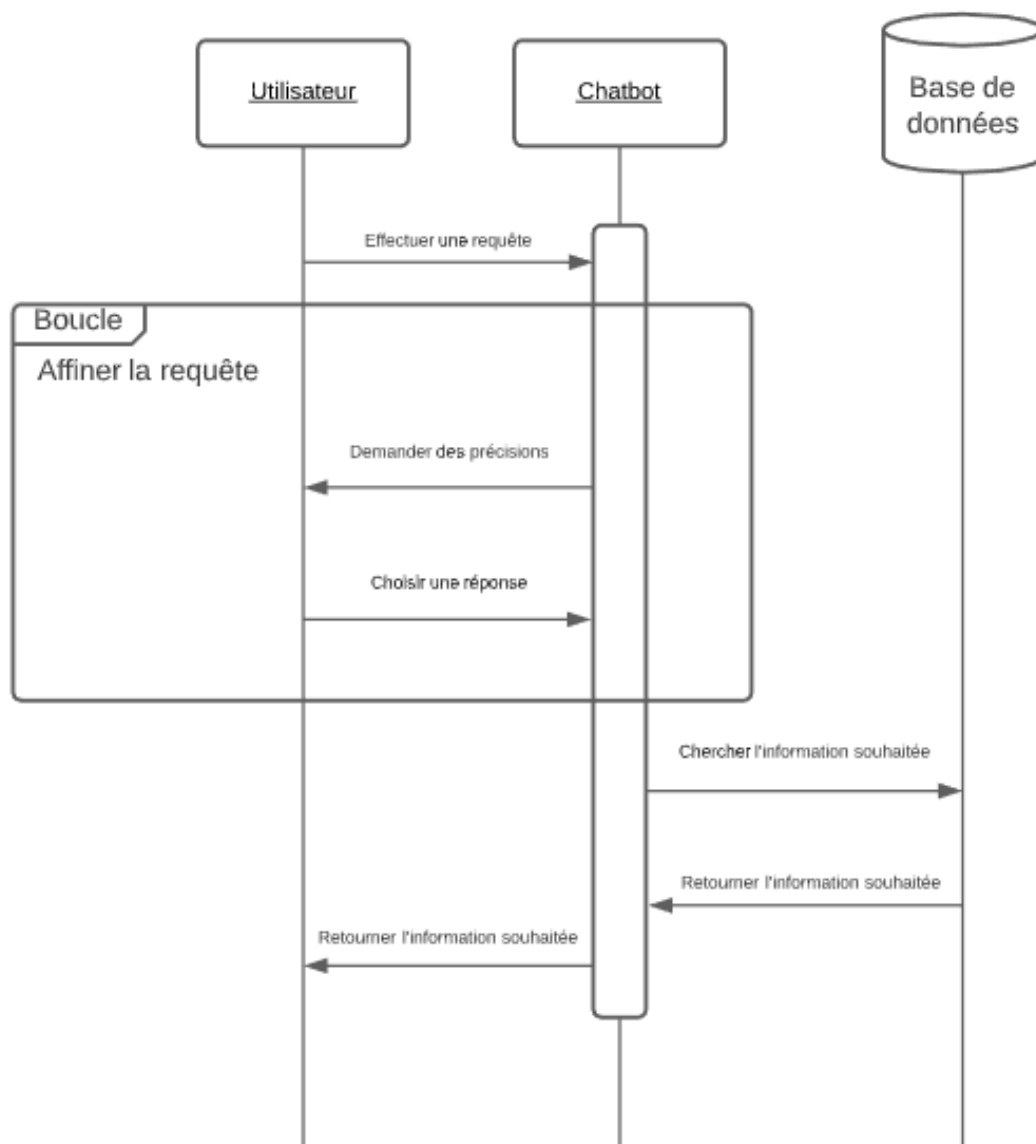


Figure 3.1: Diagramme de séquence d'un dialogue entre l'utilisateur et le chatbot

Besoin 13 [Haute] Le système doit pouvoir renvoyer un lien vers GLPI pour rédiger un ticket

Besoin 14 [Faible] Le système doit pouvoir créer un ticket Helpdesk

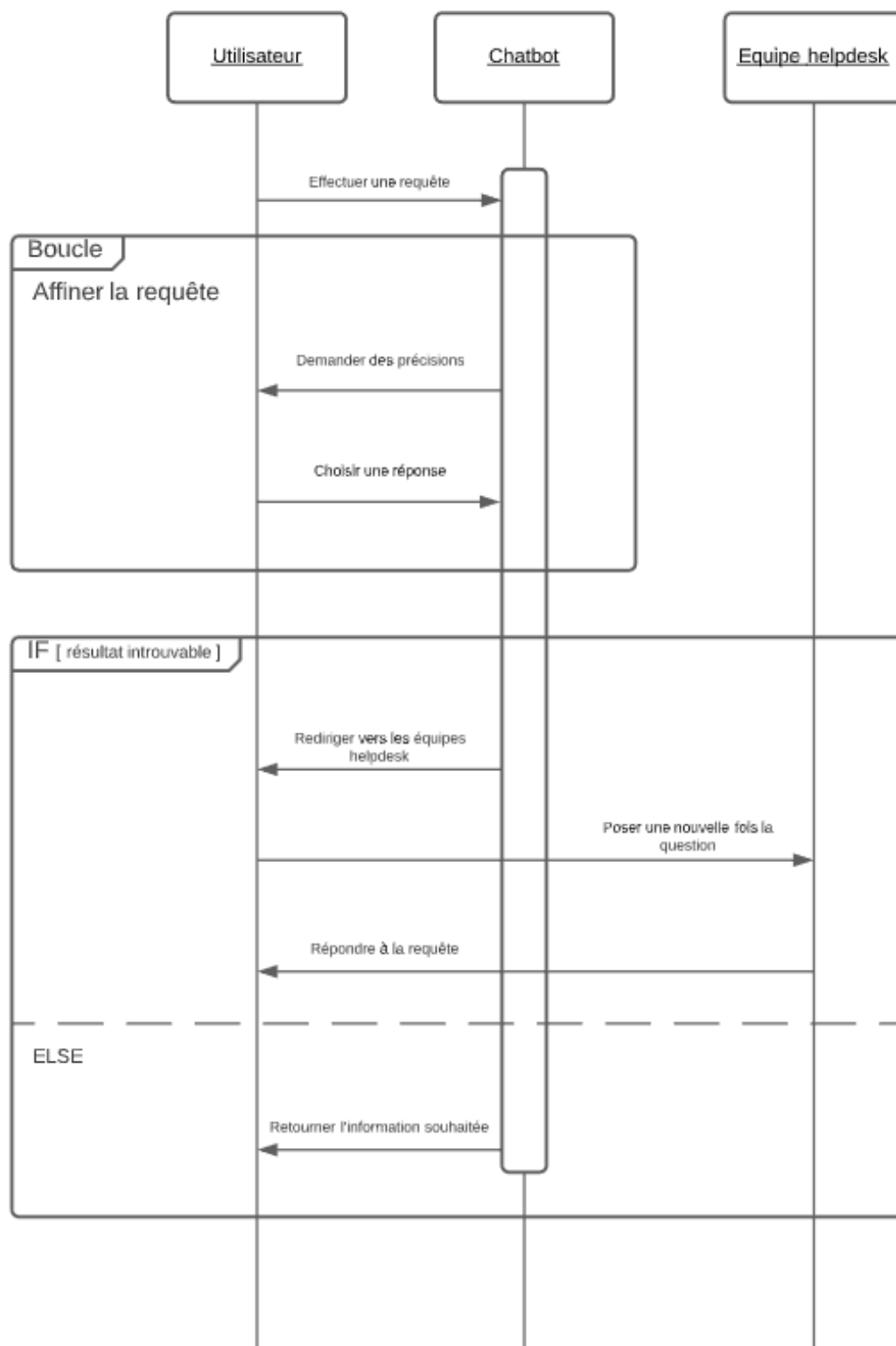


Figure 3.2: Diagramme de séquence d'un cas où le chatbot ne trouve pas de réponse

Besoin 15 [**Haute**] Le système doit demander si l'utilisateur est satisfait de la réponse.

Besoin 16 [**Moyenne**] Le système doit tager l'utilisateur qui a posé la question lorsqu'une réponse est fournie (et afficher/stocker le nombre d'interactions et le temps de réponse).

3.2 Besoins système non fonctionnels

3.2.1 Performance

Besoin 17 [**Haute**] Le système doit pouvoir gérer les requêtes de plusieurs utilisateurs connectés en même temps.

Besoin 18 [**Moyenne**] Le système doit fournir une réponse rapide à l'utilisateur (de l'ordre de 3-4 secondes maximum).

3.2.2 Sécurité

Besoin 19 [**Faible**] Le système doit respecter la RGPD (Règlement Général sur la Protection des Données).

Besoin 20 [**Moyenne**] Le système doit prendre en compte les droits d'accès des différents utilisateurs .

D RoadMap

RoadMap

izart2u, keslick1u, poncet15u

November 2020

1 Listing des tâches par ordre chronologique

1.1 Planification du projet

Rédaction d'une RoadMap du projet ChatBot :

- lister les différentes tâches

1.2 Mise en place de l'architecture du ChatBot

Mise en place d'un ChatBot en local:

- Test d'un ChatBot basique avec du code provenant de Microsoft
- Compréhension de son fonctionnement
- Rétro-ingénierie des librairies et APIs utilisée pour les ré-employer

Début de la rédaction du rapport de projet

Déploiement du ChatBot en ligne :

- Export du ChatBot de local vers Azure
- Mise en place du ChatBot pour Teams depuis Azure
- Connexion du ChatBot au Teams et à la base de données SharePoint

1.3 Mise en place des interactions avec l'utilisateur

1.3.1 Réception d'informations

Mise en place d'une récupération des messages envoyés par l'utilisateur au ChatBot :

Mise en place d'une logique de décision pour répondre aux questions fréquentes:

- Inventaire des questions fréquentes dans l'entreprise

Mise en place d'un système permettant une plus grande souplesse pour la reconnaissance des questions de l'utilisateur :

- Implémenter une ou des solutions permettant au bot de comprendre des requêtes malgré des fautes d'orthographe ou de frappe
- Si le temps le permet, intégrer une solution d'intelligence artificielle permettant au bot de comprendre des requêtes ne reprenant pas le pattern attendu

1.3.2 Envoi d'informations

Mise en place d'un système permettant d'envoyer des messages à l'utilisateur :

- Envoi de messages simples dans un premier temps
- Envoi de messages personnalisés
- Envoi de fichiers, voire au mieux de liens vers les fichiers pour éviter le transfert direct

Mise en place de la logique Q&A

1.4 Features

Mise en place d'un système de tickets HelpDesk :

- Renvoyer un lien vers GLPI pour créer un ticket Helpdesk si la réponse fournie par le ChatBot ne convient pas à l'utilisateur.

Déploiement du ChatBot sur les serveurs Figeac.

Réaliser une phase de tests.

- Tester chaque fonctionnalité du ChatBot une fois déployé sur les serveurs Figeac
- Effectuer les corrections nécessaires si besoin.

1.5 Clôture du projet

Création d'une démonstration du ChatBot pour l'industriel :

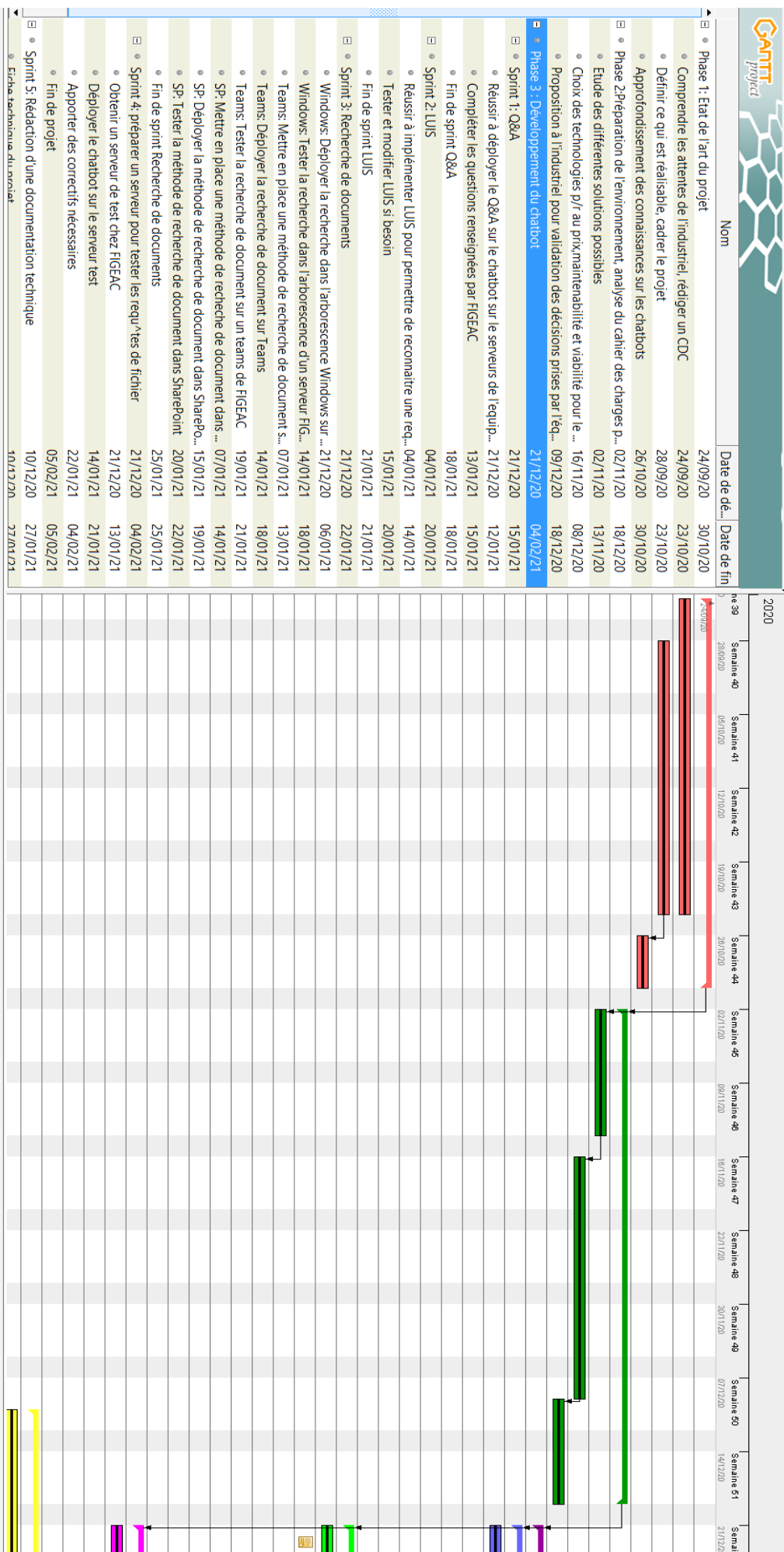
- Montrer les capacités du ChatBot
- Montrer les limites du ChatBot

Rédaction de la documentation du ChatBot pour permettre une poursuite de projet si besoin :

- Documenter les méthodes du code
- Rédiger une documentation générale présentant le projet et son contexte
- Rédiger une documentation technique reprenant les concepts, librairies, API, techniques utilisés pour la conception du ChatBot

Préparation de la soutenance de fin de projet

E Diagramme de Gantt globale effectif



54
FIGURE E.1 – Première partie diagramme de gantt effectif du projet PI

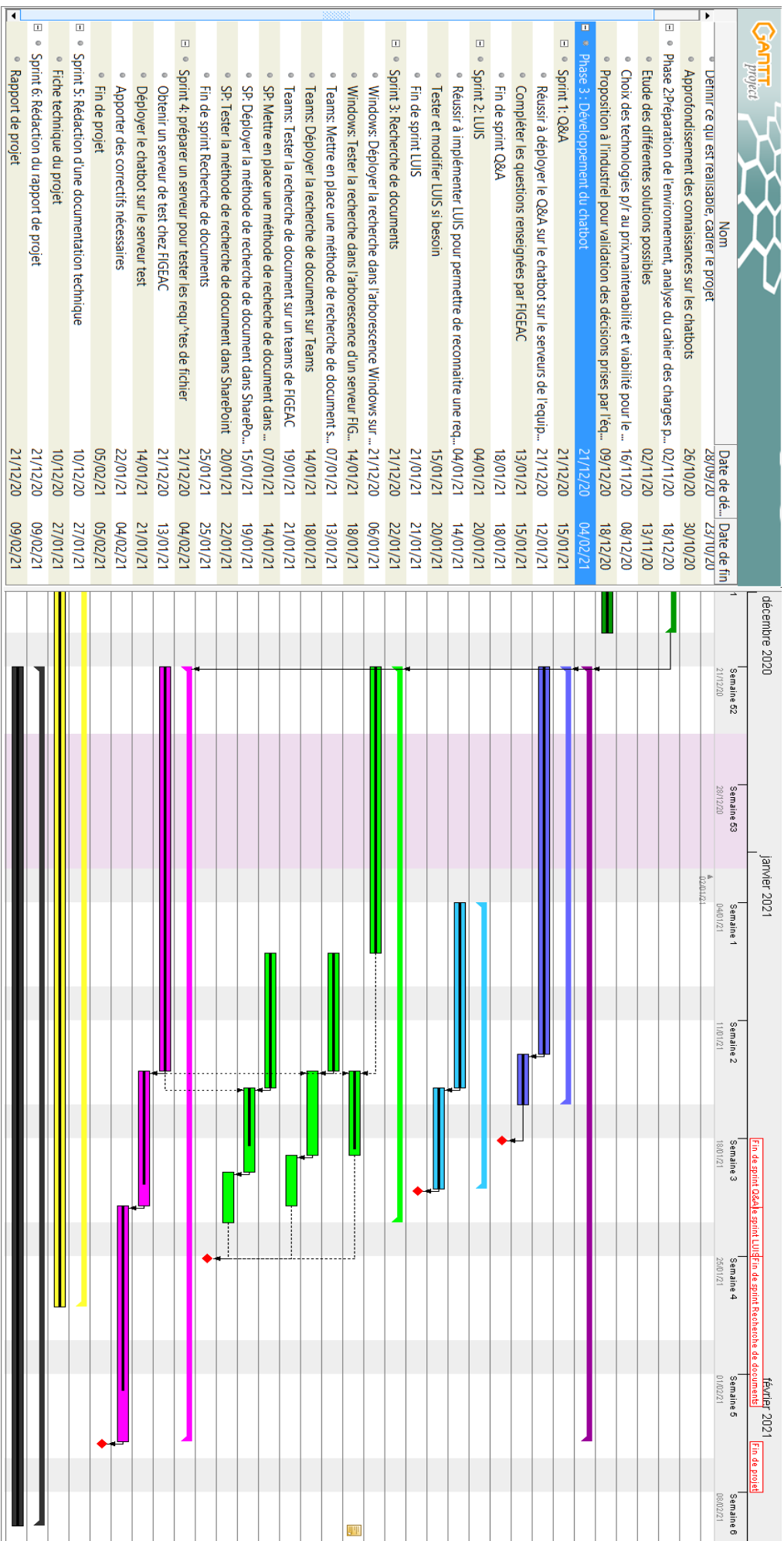


FIGURE E.2 – Seconde partie diagramme de gantt effectif projet PI

F Fichie technique

Nous avons choisi de réaliser ce chatbot dans le langage Python car ce dernier est répandu, populaire, et dispose d'une documentation très conséquente.

De plus, il est important de créer le bot sur un noyau Windows et non Linux afin de ne pas générer de conflits pendant le déploiement (en effet, il semble que le cloud de Microsoft n'accepte que des chatbots ayant été créés sur un noyau Windows). Il faudra donc utiliser le cmd de Windows pour lancer les différentes commandes.

CRÉATION ET TEST EN LOCAL

Dans un premier temps, ouvrez votre invite de commande Windows, et vérifiez que Python est bien installé. Attention, les dernières versions ne fonctionneront pas avec ce produit, il faut donc que vous ayez [la version 3.6](#) installée. Lors de l'installation, cochez la case "Ajouter au path". Vous pouvez poursuivre dès que l'installation est achevée.

Plusieurs packages Python vont être nécessaires pour créer le bot. Voici donc la liste des installations à effectuer :

```
pip install botbuilder-core
pip install asyncio
pip install aiohttp
pip install cookiecutter==1.7.0
```

Il est conseillé d'écrire tous les packages requis dans le fichier requirements.txt. Cela sera indispensable lors du déploiement car le bot ne sera plus sur votre machine où tous les packages sont déjà installés, il devra donc savoir de quels packages il a besoin.

Si vous effectuez ces commandes via le cmd, il faut trouver un moyen de lancer des commandes Python. Pour ma part, j'utilise le préfixe `python -m` au début de chaque

commande.

Enfin, il ne vous reste plus qu'à récupérer un template de bot (EchoBot dans le cas présent) et de le télécharger en local grâce à la commande suivante :

```
python -m cookiecutter  
https://github.com/microsoft/botbuilder-python/releases/download/Templates/echo.zip
```

Vous n'aurez plus qu'à renseigner le nom souhaité pour le chatbot et le tour est joué. Il ne reste plus qu'à tester le bot en local pour vérifier que tout fonctionne correctement. Pour cela, vous aurez besoin d'installer le Bot Framework Emulator (C'est ici que ça se passe : <https://github.com/Microsoft/BotFramework-Emulator/blob/master/README.md>).

Une fois téléchargé, vous n'avez plus qu'à démarrer le chatbot en lançant le app.py :

```
python app.py (ou simplement en lançant via l'IDE si vous en utilisez un.)
```

Le terminal vous affiche alors sur quel port a été lancé votre bot. Il ne reste plus à présent qu'à se rendre sur le Bot Framework Emulator et renseigner les informations nécessaires pour voir la magie opérer.

Open a bot

Bot URL
http://localhost:3978/api/messages **Browse**

Microsoft App ID Optional Microsoft App password Optional

Direct Line Speech Region Optional Direct Line Speech Key Optional

☐ Open in debug mode
☐ Azure for US Government [Learn more.](#)

Cancel **Connect**

DÉPLOIEMENT

1) Premier déploiement

Dans cette partie, nous allons utiliser le cloud de Microsoft : Microsoft Azure. Afin de communiquer avec lui en ligne de commandes, il va être nécessaire d'installer Azure CLI (Command Line Interface). Pour cela, direction le lien suivant :

<https://docs.microsoft.com/fr-fr/cli/azure/install-azure-cli-windows?tabs=azure-cli>

Un fois téléchargé, vous pourrez effectuer toutes les commandes Azure dans le cmd (il faut bien entendu avoir un compte Azure pour cela). Ensuite plus qu'à se connecter à son compte Azure grâce à la commande suivante :

```
az login
```

Attention : il faut que sa session Azure soit active pour que la manœuvre fonctionne, il faut donc au préalable se connecter au portail Azure.

Cette commande vous emmènera sur l'interface web de Microsoft où il faudra vous connecter à votre compte. Une fois de retour sur le terminale de commande, vous pourrez accéder à la liste des abonnements (subscription) dont vous disposez sur votre compte avec la commande suivante :

```
az account list
```

Attention : notez bien l'id renvoyé par cette commande, celui-ci sera important pour la prochaine étape. Dans la plupart des cas, votre compte ne disposera que d'un seul abonnement.

Il faut à présent définir l'abonnement que nous souhaitons utiliser par défaut. Cela se fait grâce à la commande suivante (<azure-subscription> étant l'id obtenu lors de l'étape précédente) :

```
az account set --subscription "<azure-subscription>"
```

Vous devez à présent créer une "app registration" (enregistrer une application) sur Azure. C'est à partir de maintenant que les commandes deviennent légèrement plus complexes.

Voici comment procéder pour créer une "registration" :

```
az ad app create --display-name "<display-name>" --password "<password>"  
--available-to-other-tenants
```

Ici, il faut choisir ce qu'on veut pour display name et password, le displayName étant le nom de l'application.

Cette commande renvoie un Json : il faut bien noter le appId qui s'y trouve car il sera utile par la suite (de même que le password que l'on vient de renseigner)

Courage, on touche presque au but !

Azure fonctionne avec des groupes de ressources qui permettent d'organiser ses entités dans le cloud. Il va donc à présent s'agir de placer cette application nouvellement créée dans un groupe de ressources. Pour cela, il faut donc créer au préalable ce groupe de ressources sur Azure, puis lancer la commande suivante :

```
az deployment group create --resource-group "<name-of-resource-group>"  
--template-file "<path-to-template-with-preexisting-rg.json>"  
--parameters appId="<app-id-from-previous-step>"  
appSecret="<password-from-previous-step>" botId="<id or  
bot-app-service-name>" newWebAppName="<bot-app-service-name>"  
newAppServicePlanName="<name-of-app-service-plan>"  
appServicePlanLocation="<region-location-name>" --name  
"<bot-app-service-name>"
```

Pas de panique, voici à quoi correspondent les différents arguments présents dans la commande :

- `<name-of-resource-group>` : le nom du groupe de ressources créé au préalable
- `<path-to-template-with-preexisting-rg.json>` : il s'agit du chemin vers le fichier `template-with-preexisting-rg.json` qui se situe dans le dossier `deploymentTemplates` de votre bot.
- `<app-id-from-previous-step>` : tout simplement le appId obtenu lors des précédentes étapes
- `<region-location-name>` : dans notre cas, il faut choisir "westeurope"
- On peut choisir le même nom pour tous les autres paramètres, cela ne pose pas de problèmes. Personnellement, j'ai choisi le même nom que le `<display-name>` de l'app registration.

Enfin, dernière étape. Il faut maintenant compresser son code pour l'envoyer sur le cloud.




Attention : lors de la compression, faites bien attention à où se situent vos programmes python. En effet, il faut que ceux-ci soient à la racine de l'archive et non un cran plus bas.

Maintenant, plus qu'à lancer la commande suivant pour que le chatbot se déploie en quelques minutes :

```
az webapp deployment source config-zip --resource-group
"<resource-group-name>" --name "<name-of-web-app>" --src
"<project-zip-path>"
```

- `<name-of-resource-group>` : toujours le même nom du groupe de ressource créé précédemment
- `<name-of-web-app>` : le nom de la web app créée précédemment, à savoir le `<bot-app-service-name>` de la commande précédente
- `<project-zip-file>` : enfin, il faut mettre ici le chemin vers l'archive contenant votre code compressé.

L'opération prend alors quelques temps mais devrait se terminer sans encombre. Une fois fini, vous pouvez tester votre bot directement sur Azure. Pour cela, vous devrez vous rendre dans le groupe de ressources. Si tout s'est bien passé, 3 éléments ont été créés :

<input type="checkbox"/> Nom ↑↓	Type ↑↓	Emplacement ↑↓	
<input type="checkbox"/>  chatbotFigeac	Inscription de chaînes de bot	Global	...
<input type="checkbox"/>  chatbotFigeac	Plan App Service	Europe occidentale	...
<input type="checkbox"/>  chatbotFigeac	App Service	Europe occidentale	...

C'est dans le premier que vous devrez vous rendre pour tester votre bot à l'aide de la fonctionnalité "Tester dans une discussion web" qui s'y trouve.

Et voilà, votre chatbot est maintenant déployé et prêt à être utilisé sur différentes applications !

2) Redéploiement

Pour redéployer un chatbot sur Azure, rien de plus simple. Vérifiez tout d'abord que vous êtes bien connectés à Azure (en ligne puis avec la commande `az login`). Enfin il ne reste plus qu'à compresser à nouveau le code et à l'injecter sur le cloud avec la dernière commande :

```
az webapp deployment source config-zip --resource-group
"<resource-group-name>" --name "<name-of-web-app>" --src
"<project-zip-path>"
```

Il faut parfois attendre quelques minutes (1 ou 2 pour moi mais parfois plus) pour que le redéploiement soit effectif, puis vous pourrez tester le chatbot dans le chat web sur Azure pour vérifier que tout fonctionne correctement.

Si une erreur survient avec pour description "ImportError : no module named decorator", il vous faudra [désinstaller puis réinstaller le client azure](#).

CREATION D'UN SERVICE DE QUESTION AND ANSWER (QNA MAKER)

La création d'un service QnA se fait en plusieurs temps. Il va d'abord falloir créer un "QnA maker service" associé à un groupe de ressources. Ce dernier permettra de créer une ou plusieurs bases de connaissances qui pourront être reliées au bot et dans lesquelles seront renseignées tous les couples de questions/réponses.

1) Création du QnA Maker service

La première étape sera de créer un QnA Maker service en se rendant sur ce lien : <https://portal.azure.com/#create/Microsoft.CognitiveServicesQnAMaker>. Une fois sur cette page, une multitude d'informations vous seront demandées. Tout d'abord, plusieurs options s'offrent à vous : vous pouvez choisir de créer un QnA Maker service managé ou non. La seule différence réside dans le fait que Azure gérera automatiquement certains paramètres ou non. Personnellement, j'ai expérimenté l'option non managée et elle

fonctionne très bien.

Détails du projet

Sélectionnez l'abonnement pour gérer les coûts et les ressources déployées. Utilisez les groupes de ressources comme les dossiers pour organiser et gérer toutes vos ressources.

Abonnement *	Essai gratuit
Groupe de ressources *	QnAMakerBis
	Créer nouveau
Nom *	figeacChatbotQnAMakerBis
Niveau tarifaire (En savoir plus) *	Gratuit F0 (3 documents gérés par mois, 3 transactions par seconde, ...)

Détails de la recherche Azure, pour les données

Quand vous créez une ressource QnAMaker, vous hébergez les données dans votre abonnement Azure. Le service Recherche Azure est utilisée pour indexer vos données.

Emplacement Recherche Azure *	(Europe) Europe occidentale
Niveau tarifaire Recherche Azure *	Gratuit F (3 Indexes)

Détails App Service, pour le runtime

Quand vous créez une ressource QnAMaker, vous hébergez le runtime dans votre abonnement Azure. App Service est le moteur de calcul qui exécute les requêtes QnA Maker pour vous.

Nom de l'application *	figeacChatbotQnAMakerBis
Emplacement du site web *	(Europe) Europe occidentale

i Le plan App Service est actuellement défini par défaut avec le niveau Standard (S1) (Prix). Pour le modifier, visitez la page de ressources du plan App Service une fois la ressource créée.

Détails App Insights, pour la télémétrie et les journaux de conversation

QnAMaker provisionne éventuellement une instance d'Application Insights et s'affiche dans votre abonnement Azure. Les données de télémétrie et les chatlogs sont stockés ici.

App Insights	<button>Activer</button> <button>Désactiver</button>
Emplacement d'App Insights *	(Europe) Europe occidentale

Voici donc la liste de toutes les options que vous devrez remplir. Ici, j'ai choisi de placer le QnA Maker service dans un autre groupe de ressources, mais cela n'a pas vraiment d'importance pour la suite. Vous devrez donc choisir un nom pour votre QnA Maker puis renseigner les différentes options de votre abonnement (niveau tarifaire, emplacement géographique, ...). Pour nous, tout est au niveau tarifaire le plus bas et en

Europe occidentale.

Enfin, Azure vous propose le choix d'activer ou non les app insights. Nous ne savons pas encore trop à quoi cela correspond mais nous avons gardé cette option activée dans le doute.

2) Création d'une base de connaissances

Une fois le QnA Maker créé, Azure vous renverra vers l'interface d'édition des knowledge bases. C'est ici que vous pourrez créer vos bases de connaissances et les remplir de questions/réponses. C'est donc ce que nous allons faire à présent : créez une base de connaissance et rentrez autant de questions/réponses que vous le souhaitez (une ou deux suffiront pour l'instant). Vous remarquerez que vous pouvez également renseigner des formulations alternatives à vos questions afin que le chatbot puisse mieux les comprendre.

Une fois cette étape effectuée, notre attention va se porter sur les deux boutons situés en haut à droite : "Save and train" et "Test". Tout d'abord, il vous faudra sauvegarder votre base de connaissance dans le cloud. Vous pourrez alors la tester pour vérifier que tout fonctionne correctement.

Une fois que vous êtes satisfait de votre base de connaissances, vous pourrez alors la publier (en cliquant sur l'onglet "Publish"). Après avoir confirmé la publication, vous verrez s'afficher un cadre contenant de précieuses informations : notez les bien ! Elles seront très utiles pour la suite.

Postman Curl

```
POST /knowledgebases/00f5b512-0805-41d6-bff3-7af4fc1316a6/generateAnswer
Host: https://figeacchatbotqnamakerbis.azurewebsites.net/qnamaker
Authorization: EndpointKey 4080be5f-b3ba-45c0-95eb-e166efc18cec
Content-Type: application/json
{"question": "<Your question>"}
```

Ces informations sont de la forme :

```
POST /knowledgebases/<knowledge-base-id>/generateAnswer
Host: <your-hostname> // NOTE - this is a URL ending in /qnamaker.
Authorization: EndpointKey <qna-maker-endpoint-key>
```

Par soucis de sûreté, nous vous invitons à vérifier que la souscription à ce service est

bien passé en gratuite. Pour cela, rendez vous sur le groupe de ressources de votre chatbot via le portail Azure, et ouvrez le *App Service* créé pour le QnA. Dans la partie Overview, vous aurez accès à la tarification. Si elle est sur F1, alors très bien. Sinon, rendez-vous dans la section "Change App Service Plan" dans la barre déroulante de gauche, puis créez un nouveau *App Service Plan* à la bonne tarification. Allez ensuite supprimer celui qui n'est plus utilisé.

3) Liaison entre cette base de connaissances et le chatbot

Il va maintenant falloir modifier le code de votre bot afin que celui-ci intègre la base de connaissances :

- Première étape : installer les packages suivants (avec pip comme au lors de la création du chatbot) : botbuilder-ai et flask
- Deuxième étape : ajouter ces lignes dans votre config.py

```
QNA_KNOWLEDGEBASE_ID = os.environ.get("QnAKnowledgebaseId", "<knowledge-base-id>")
QNA_ENDPOINT_KEY = os.environ.get("QnAEndpointKey", "<qna-maker-endpoint-key>")
QNA_ENDPOINT_HOST = os.environ.get("QnAEndpointHostName", "<your-hostname>")
```

- Troisième étape : modifier le app.py en ajoutant l'argument "CONFIG" (ligne 60)

```
BOT = MyBot(CONFIG)
```

- Quatrième étape : modifier le bot.py.
D'abord ajouter quelques imports au début du fichier (pensez également à ajouter les packages botbuilder-ai et flask dans le requirements.txt)

```
from flask import Config

from botbuilder.ai.qna import QnAMaker, QnAMakerEndpoint
from botbuilder.core import MessageFactory
```

Puis ajouter une fonction `__init__` au début de la classe :

```
def __init__(self, config: Config):
    self.qna_maker = QnAMaker(
        QnAMakerEndpoint(
```

```

        knowledge_base_id=config.QNA_KNOWLEDGEBASE_ID,
        endpoint_key=config.QNA_ENDPOINT_KEY,
        host=config.QNA_ENDPOINT_HOST,
    )
)

```

- Enfin, cinquième et dernière étape, il faut modifier légèrement la fonction `on_message_activity` de la manière suivante :

```

async def on_message_activity(self, turn_context: TurnContext):
    # The actual call to the QnA Maker service.
    response = await self.qna_maker.get_answers(turn_context)
    if response and len(response) > 0:
        await turn_context.send_activity(MessageFactory.text(response[0].answer))
    else:
        await turn_context.send_activity("No QnA Maker answers were found.")

```

C'est à priori dans la dernière ligne que sera effectué le code si aucune réponse n'est trouvée dans la base de connaissances.

Il ne reste maintenant plus qu'à redéployer le chatbot pour que ces changements soient effectifs.

LA RECHERCHE DE FICHIERS

1) Utilisation de Luis

Dans cette partie nous allons dérouler le processus de création et d'utilisation d'une ressource Luis.

Dans un premier temps, rendez-vous sur luis.ai. Vous pourrez alors cliquer sur "nouvelle ressource de création". Choisissez alors l'abonnement, le groupe de ressource associé (celui du chatbot, créé lors du déploiement), et le nom de la ressource.

Il est probable que vous rencontriez l'erreur suivante :

✖ Échec de la création d'un compte Az... 5:31 PM ✕

MissingSubscriptionRegistration: The subscription is not registered to use namespace 'Microsoft.CognitiveServices'. See <https://aka.ms/rps-not-found> for how to register subscriptions.

Dans ce cas, rendez-vous sur le portail Azure, et ouvrez le menu en haut à gauche. Vous trouverez dans la barre déroulante l'accès aux *Subscriptions*. Cliquez sur la souscription avec laquelle vous travaillez depuis le début, puis cliquez sur *Resource Providers* dans la barre déroulante à gauche de votre écran. Filtrez ensuite sur "Microsoft.CognitiveServices", et cliquez sur *Register*. Il est probable que vous deviez cliquer plusieurs fois pour que la modification soit prise en compte. Une fois cela fait, vous pouvez retourner sur l'onglet Luis, et cliquer sur terminer.

Votre ressource ayant été créée, vous pouvez maintenant créer une nouvelle application. Ouvrez l'onglet *Gérer* en haut à droite de l'écran. Dans la barre de gauche, cliquez sur *Versions*, puis importez le fichier .lu que nous vous avons fourni. Cliquez ensuite sur *Entraîner* en haut à droite, puis sur *Publier* juste à côté. Rendez vous ensuite dans la partie *Ressources Azure* dans la barre à gauche de votre écran. Dans la partie *Ressources de Prévion*, créez une nouvelle ressource de prévision s'il n'y en a pas déjà une.

Vous pourrez ensuite récupérer l'exemple de requête fourni, puisqu'il contient des clés essentielles pour relier Luis au chatbot. Elle sera de la forme :

```
https://<LUIS_API_HOSTNAME>/luis/prediction/v3.0/apps/<LUIS_APP_ID>/slots/production/prediction?subscription-key=<LUIS_API_KEY>&verbose=true&show-all-intents=true&log=true&query=YOUR_QUERY_HERE
```

Rendez vous dans le fichier config.py du chatbot, puis mettez à jour les valeurs suivantes avec celles que vous venez d'obtenir.

```
LUIS_APP_ID = os.environ.get("LuisAppId", "<LUIS_APP_ID>")
LUIS_API_KEY = os.environ.get("LuisAPIKey", "<LUIS_API_KEY>")
# LUIS endpoint host name, ie "westus.api.cognitive.microsoft.com"
LUIS_API_HOST_NAME = os.environ.get("LuisAPIHostName", "<LUIS_API_HOST_NAME>")
```

Finalement, re-déployez votre chatbot, et vous pourrez alors utiliser la fonctionnalité de Luis.

2) Recherche dans les répertoires partagés Windows

Pour le moment, la recherche de fichiers Windows ne se fait que sur les fichiers locaux à l'archive envoyée lors du déploiement. Si le projet est repris, il suffirait alors de programmer l'accès à une base de dossiers partagés Windows, et le reste du code serait toujours fonctionnel.

3) Rechercher dans SharePoint

L'accès à la base SharePoint se fait à l'aide de son URL, et d'un compte utilisateur sur le domaine de Figac Aero, avec les accès associés. Il doit impérativement avoir été désactivé la double authentification pour ce compte, car cela bloque l'accès depuis la fonction Python sinon. Si le compte d'accès ou le lien du SharePoint souhaite être modifié, il faudra alors se rendre dans le fichier config.py, et actualiser la partie ci-dessous.

```
SHAREPOINT_USERNAME = os.environ.get("SharepointId","<email-compte-service>")
SHAREPOINT_PASSWORD = os.environ.get("SharepointPassword","<password>")
SHAREPOINT_SERVER_URL = os.environ.get("SharepointServerUrl","https://fga01.sharepoint.com/sites/DSI-FigeacAero-ProjetChatBotavecTelecomNancy")
```

Il est important de noter que pour le moment, le chatbot ne différencie pas les droits d'accès de l'utilisateur des siens, et par conséquent peut renvoyer des fichiers auxquels l'utilisateur ne devrait pas avoir accès.

4) Recherche dans Teams

La recherche dans les fichiers Teams fonctionnant de la même manière que dans un SharePoint classique, il faudra alors seulement récupérer l'adresse du SharePoint associé au Teams en question, et le fournir dans le fichier config.py.

```
TEAMS_SP_URL = os.environ.get("TeamsSharepointUrl","<URL>")
```

L'URL de ce SharePoint peut être trouvée en ouvrant un fichier dans le Teams voulu, puis en cliquant sur les ... en haut à droite puis sur "Ouvrir dans SharePoint".

G Comptes-rendus de réunion

G.1 Première réunion

Compte-rendu de réunion

Jeudi 24 Septembre 2020 - Meeting sur Microsoft Teams

Réunion de lancement - Durée : 1h00

Président de séance : Théo Poncet

Secrétaire de séance : Malaury Keslick

Présents :

AUROY Simon
CHAROY François
IZART Guillaume
KESLICK Malaury
PONCET Théo
VERNEY Loïc

Ordre du jour

- Prise de contact
- Cadrage du projet
- Propriété intellectuelle

Informations échangées

Prise de contact

La réunion débute par un tour de table où chacun se présente.

Cadrage du projet

Loïc VERNEY nous fait un rappel sur le but du projet. L'idée est d'optimiser le travail au sein de Figeac Aero à l'aide d'un chatbot que les employés pourraient solliciter à la place d'une aide humaine lorsqu'ils recherchent une information, un document, etc.. Notre rôle serait alors d'explorer les différentes possibilités afin de leur proposer un POC.

Ce projet pourrait alors être découpé en trois grandes étapes :

- une première partie lors de laquelle on cadrerait réellement le projet : approfondissement de nos connaissances sur les chatbots, comprendre ce qui est réalisable ou non, ainsi que cerner précisément les attentes de l'entreprise pour enfin établir un cahier des charges du projet ;

- nous devons ensuite étudier les différentes technologies existantes qui pourraient nous permettre de mener à bien ce projet, pour extraire celles que nous considérons les plus avantageuses en terme de prix, d'intégration aux technologies déjà utilisées par l'entreprise (partenariat Microsoft) et de facilité de reprise/maintenance du projet. C'est à ce moment que nous établirons le POC que nous confronterons à l'avis de Loïc VERNEY ;

- enfin nous pourrions passer à la phase de développement de l'agent conversationnel tout en nous assurant en parallèle de la rédaction de la documentation technique qui permettra la bonne reprise du projet.

Plus clairement, ce projet est un travail d'exploration des différentes options qu'ont l'entreprise. Ils ne sont absolument pas experts du domaine et se retrouvent alors dans la position de client face à nous. Loïc VERNEY appuie sur le fait que lui et Simon AUROY seraient intéressés par une présentation sur le concept des chatbots. Malaury KESLICK répond que cela devrait pouvoir se faire assez rapidement puisqu'elle est déjà familière avec ceux-ci, en ayant développé un lors de son stage de deuxième année cet été.

Après ces discussions sur la partie "cernage" du projet, les membres se mettent d'accord pour garder ce créneau de réunion du jeudi de 17h à 18h sur une base pour le moment hebdomadaire. La prochaine réunion aura donc lieu juste après le rendu de la note de cadrage, nous allons donc essayer d'en faire parvenir une première version à Loïc VERNEY d'ici lundi soir au plus tard afin d'avoir le temps d'en corriger les défauts.

Propriété intellectuelle

L'entreprise souhaiterait pouvoir garder le code que nous produirons pour pouvoir ensuite reprendre le projet en interne. Pour autant, si nous souhaitons ré-utiliser ce code dans notre vie future, il n'y aurait aucun problème. Malaury KESLICK va se renseigner auprès de Mme. CRUGNOLA afin de connaître les modalités du contrat à rédiger dans ce cas de figure.

Loïc VERNEY précise également que nous devons signer un accord de non-divulgaration puisque nous aurons l'occasion de manipuler des données privées de l'entreprise.

Todolist

Description	Respo	Délai	Livrable
Rédaction du CR	Malaury	25/09	CR
Rédaction de la première version de la note de cadrage	Malaury, Guillaume, Théo	28/09	Note de cadrage
Contacter Mme. Crugnola pour la propriété intellectuelle	Malaury	01/10	
Initier Guillaume et Théo aux chatbots	Malaury	01/10	

Prochaine réunion : Jeudi 1er octobre à 17h.

G.2 Troisième réunion

Compte-rendu de réunion

Jeudi 15 Octobre 2020 - Meeting sur Microsoft Teams

Réunion d'avancement - Durée : 30min

Président de séance : Théo Poncet

Secrétaire de séance : Malaury Keslick

Présents :

AUROY Simon
IZART Guillaume
KESLICK Malaury
PONCET Théo
VERNEY Loïc

Ordre du jour

- Note de cadrage
- Conférence Jellybot
- Prochaine deadlines

Informations échangées

Note de cadrage

Nous avons reçu notre note de cadrage corrigée par Anne-Claire HEURTEL. Nous allons donc faire les modifications en conséquence, et en particulier retravailler le Gantt pour qu'il soit plus précis, détaillé et visuel. Cette nouvelle version de la note de cadrage sera envoyée au plus tard en début de semaine prochaine à l'encadrant industriel, afin qu'elle puisse être validée et signée avant le rendu académique du 23 octobre.

Conférence Jellybot

Malaury présente succinctement la conférence sur Jellybot qu'elle a pu suivre dernièrement. Elle note que bien qu'elle connaissait la plupart des mécanismes présentés, quelques nouvelles idées sont intéressantes à développer dans le cadre d'options. Ces idées feront partie du cahier des charges.

Prochaines deadlines

Actuellement, chaque membre du groupe est sur une tâche différente :

- Théo se charge d'apporter les correctifs nécessaires à la note de cadrage avant de l'envoyer à Loïc VERNEY en début de semaine prochaine pour validation. Il rejoindra ensuite la tâche de Guillaume ;
- Guillaume, ensuite rejoint par Théo, se charge de la rédaction du cahier des charges du projet, afin de poser noir sur blanc quelles sont précisément les attentes vis-à-vis du chatbot, et quels seront les axes optionnels. Ce cahier des charges devra être déposé sur Teams au plus tard le 1er novembre ;
- Malaury se charge d'étudier l'option du cloud Microsoft Azure pour développer le chatbot, le but étant de préparer une présentation de l'option complète, du processus de création aux ressources nécessaires.

Ces deux derniers points seront présentés et discutés lors de la réunion du **mercredi 4 septembre**.

Todolist

Description	Respo	Délai	Livrable
Correction et envoi de la note de cadrage	Théo	20/10	Note de cadrage
Rédaction du cahier des charges	Théo et Guillaume	01/11	Cahier des charges
Présentation de l'option cloud Azure pour développer et intégrer le chatbot	Malaury	04/11	Powerpoint

Prochaine réunion : Jeudi 22 octobre à 17h.

G.3 Cinquième réunion

Compte-rendu de réunion

Jeudi 19 Novembre 2020 - Meeting sur Microsoft Teams

Réunion d'avancement - Durée : 1 heure

Président de séance : Théo Poncet

Secrétaire de séance : Malaury Keslick

Présents :

AUROY Simon
IZART Guillaume
KESLICK Malaury
PONCET Théo
VERNEY Loïc

Ordre du jour

- Cahier des charges
- Roadmap
- Début de mise en place du Chatbot
- Prochaines étapes

Informations échangées

Cahier des charges

La réunion commence par quelques remarques de Loïc VERNEY sur le cahier des charges :

- la fonctionnalité permettant à un utilisateur de gérer ses tickets helpdesk n'est pas nécessaire. Si le bot n'est pas capable d'aider l'utilisateur, un simple envoi de lien vers le site est suffisant.
- le chatbot doit être accessible par conversation privée pour chaque utilisateur et non via un canal public.
- reformuler le besoin concernant la disponibilité du chatbot : 100% de disponibilité sur les horaires de travail (6h-20h).
- l'ordre de priorité des plateformes où chercher les informations est : arborescence Windows, Teams (et la base SharePoint qui est cachée derrière celui-ci), GLPI.
- utiliser le terme "information" plutôt que "fichier" pour couvrir un panel plus large de fonctionnalités.
- la priorité du besoin "le système doit demander à l'utilisateur s'il est satisfait" doit être plus haute.

Roadmap

Le groupe projet présente la roadmap à Loïc VERNEY et Simon AUROY. Elle est confrontée au diagramme de Gantt présentée en début de projet, sur lequel nous sommes un peu en avance. Nous devons simplement rajouter une phase de déploiement sur les serveurs de l'entreprise et une phase de test en fin de projet.

Début de mise en place du Chatbot

Ensuite, Malaury présente le bot basique qui a été implémenté en Python, déployé sur Azure et intégré à une équipe Teams créée pour les tests [fig. 1]. Ainsi la structure et les méthodes de base du Chatbot sont implémentées. Les fonctionnalités décidées et présentées dans le cahier des charges seront ajoutées à ce squelette au fur et à mesure.

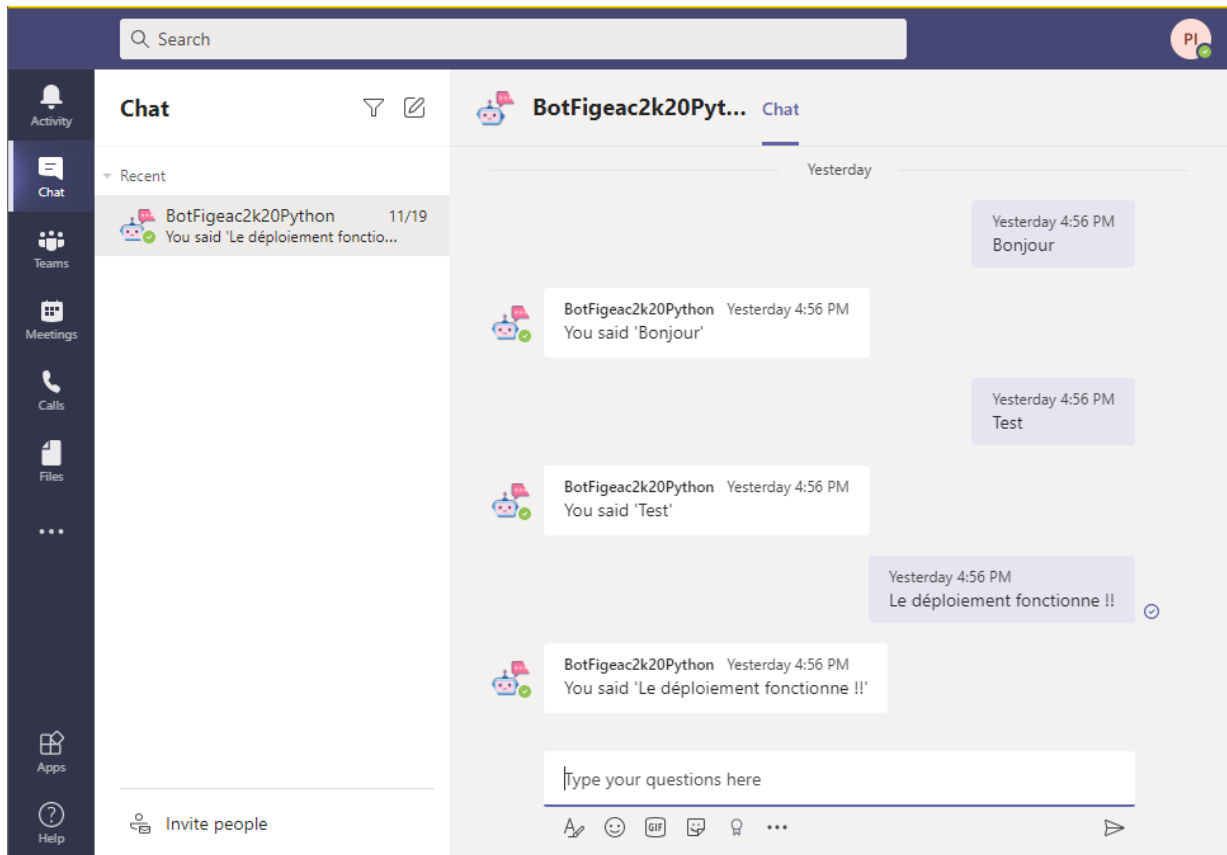


Figure 1: Mise en place d'un EchoBot basique

Prochaines étapes

Todolist

Description	Responsable	Délai
Correctifs cahier des charges et roadmap	Guillaume et Théo	22/11
Mise en place de la base QnA	Guillaume	26/11
Renseignements sur le lien bot/base SharePoint de Teams	Théo	26/11
Renseignements sur le lien bot/arborescence de fichiers Windows	Malaury	26/11
Commencer la rédaction du rapport	Guillaume, Malaury et Théo	-

Prochaine réunion : Jeudi 26 Novembre à 17h.

G.4 Réunion de confrontation des attentes

Compte-rendu de réunion

Jeudi 17 Décembre 2020 - Meeting sur Microsoft Teams

Réunion de confrontation des attentes - Durée : 1 heure

Président de séance : Théo Poncet

Secrétaire de séance : Malaury Keslick

Présents :

AUROY Simon
CHAROY François
IZART Guillaume
KESLICK Malaury
PONCET Théo
VERNEY Loïc

Ordre du jour

- Retour sur les formulaires ;
- Point sur l'avancement du projet ;
- Plan d'action et mise à jour du programme de travail.

Informations échangées

1 Retour sur les formulaires

L'objectif de cette réunion était de faire un point sur les premiers mois du Projet Industriel. Chaque acteur du projet (les élèves, les encadrants industriels ainsi que l'encadrant académique) devait donc remplir au préalable un formulaire ou il expliquait son ressenti sur le début de ce projet.

1. Les encadrants industriels ont montré un avis assez positif par rapport à l'opinion générale sur le projet et son avancement. Ils estiment que les élèves ont bien appréhendé le projet mais ne se rendent tout de même pas bien compte de leur capacité à gérer leur temps en parallèle de leur travail à l'école. Eux aussi sont un peu frustré par leur contexte économique qui les force à parfois avoir d'autres priorités que notre projet. Théo PONCET souligne les difficultés à se motiver et à travailler efficacement en ces temps de confinement difficiles, mais garde quand même une opinion générale positive sur ce projet. Pour les mêmes raisons, qui affectent plus Malaury KESLICK, elle avoue un point de vue mitigé sur ce point. Guillaume IZART quant à lui se montre un peu plus mitigé : en effet, bien que l'équipe soit en avance sur le planning prévisionnel, il est frustré de ne pas avoir pu concilier projet et travail à l'école assez efficacement. Cela est cependant voué à s'améliorer dans les semaines à venir avec l'allègement des cours.
2. Toute l'équipe est d'accord pour dire que l'objectif du projet a été plutôt bien appréhendé dès le début, même si certains points ont dû être rediscutés au fur et à mesure.
3. Concernant les conditions de démarrage du projet, tout le monde s'accorde à dire qu'elles ont été satisfaisantes malgré les conditions difficiles dues à la distance puis au confinement qui ont rendu la communication et la compréhension plus difficiles entre les différents partis.

4. Pour ce qui est du planning prévisionnel, les élèves estiment qu'il est adapté et respecté même s'il ne prend pas encore tout en compte (comme expliqué plus haut). Les encadrants industriels quant à eux sont partiellement satisfaits car ils craignent qu'il ne puisse pas être tenu. Ils ont du mal à se projeter sur ce qu'il reste à faire dans le peu de temps restant, surtout par rapport au déploiement côté entreprise.
5. Globalement, les élèves ne sont que partiellement satisfaits de leur utilisation du temps dédié au projet. Malaury KESLICK souligne à nouveau qu'elle n'arrive pas à se sentir satisfaite à cause du distanciel qui ne lui convient pas du tout, puisqu'elle a l'impression d'être de moins en moins motivée et efficace de manière générale. Guillaume IZART quant à lui estime que le groupe aurait pu s'investir plus à certains moments, mais reste tout de même satisfait de l'efficacité générale du groupe.
6. C'est à présent au tour de la qualité des documents produits d'être évaluée. Les élèves sont satisfaits de ceux-ci mais c'est moins le cas pour les encadrants industriels. En effet, ceux-ci ont parfois du mal à les comprendre avec leur point de vue plus externe. Il faudra donc faire des efforts pour rendre le travail plus compréhensible pour tous les partis, une fiche technique et des infographies pourront par exemple être utiles.
7. L'ensemble des partis est satisfait sur le point de la qualité de la communication. L'équipe projet semble rapidement comprendre et rebondir dans la bonne direction lorsque c'est nécessaire.
8. Tout le monde est unanime : le groupe dispose des compétences et connaissances nécessaires pour mener le projet à bien. A l'origine, seule Malaury KESLICK possédait une expérience et des compétences dans le domaine des chatbots, mais le reste du groupe tend à les acquérir également.
9. Le groupe estime également disposer des ressources nécessaires à la bonne réalisation du projet. La seule chose qui pourrait éventuellement venir à manquer est un serveur fourni par l'entreprise pour tester la recherche de fichiers, mais ce sujet a été abordé et le serveur devrait être fourni prochainement. De même, les droits d'accès aux différentes plateformes nécessaires à l'intégration du bot à l'architecture numérique de l'entreprise seront donnés en temps voulu.
10. Tous sauf Malaury KESLICK s'accordent sur leur satisfaction. Cette dernière n'engage que sa propre motivation dans ses propos, et annonce qu'elle a prit la décision d'aller travailler sur site à TELECOM Nancy en janvier même si les cours ne reprennent pas en présentiel. Elle espère ainsi pouvoir pallier à sa perte de motivation passagère.
11. Le planning des réunions du jeudi ayant réussi à être tenu quasi-systématiquement, tous les partis sont satisfaits de la disponibilité des autres.
12. Tous les acteurs s'accordent et sont parfaitement satisfaits sur le point de l'esprit d'équipe, de la cohésion et de la bienveillance pendant ce début de projet.

2 Point sur l'avancement du projet

A l'heure actuelle, l'équipe projet est en avance sur le GANTT prévisionnel, bien que celui-ci ne prenne pas en compte le déplacement de la semaine bloquée pour le projet (à cause des conditions sanitaires) ainsi que la phase de déploiement sur les serveurs de Figeac Aero. Le GANTT devra donc être refait pour prendre en compte ce changement.

Certaines fonctionnalités fonctionnent d'ores et déjà mais n'ont pas encore été assemblées et fonctionnent pour l'instant indépendamment les unes des autres. L'équipe a donc développé un bot avec système de QnA ainsi que des fonctions de recherches dans une arborescence de fichiers (il ne manque plus qu'un accès à un serveur afin de tester cette fonctionnalité). Malaury KESLICK a également commencé à travailler sur l'utilisation du service LUIS : celui-ci permet de rendre un bot plus "intelligent" en lui permettant de mieux comprendre un utilisateur.

3 Plan d'action et mise à jour du programme de travail

Afin de maintenir le cap vers une bonne réalisation de ce projet, l'équipe fournira lors la réunion de la rentrée une présentation point par point de ce qui est déjà traité par le bot et de qu'il reste à réaliser. Ainsi l'encadrant industriel pourra mieux se projeter sur les problématiques en suspens. De plus, le GANTT sera remis à jour afin d'être plus précis sur ces dernières semaines, et d'inclure la phase de déploiement côté entreprise.

Todolist

Description	Responsable	Délai
Retravailler le GANTT	Théo PONCET	04/01
Rédiger une documentation	Théo PONCET	04/01
Préparer un schéma récapitulatif de l'avancement	Malaury KESLICK	04/01
Déployer un bot avec QnA sur le cloud puis sur Teams	Guillaume IZART	04/01
Continuer à mettre en place LUIS	Malaury KESLICK	04/01
Préparer un serveur pour tester la recherche de fichiers	Loïc VERNEY	04/01

Prochaine réunion : Jeudi 7 Janvier à 17h.

G.5 Septième réunion

Compte-rendu de réunion

Jeudi 7 Janvier 2021 - Meeting sur Microsoft Teams

Réunion d'avancement - Durée : 1 heure

Président de séance : Théo Poncet

Secrétaire de séance : Malaury Keslick

Présents :

AUROY Simon
IZART Guillaume
KESLICK Malaury
PONCET Théo
VERNEY Loïc

Ordre du jour

- Illustration de la construction du chatbot
- Point sur le GANTT pour la fin de ce projet
- Démonstration live des blocks Luis et QnA
- Prochaine étapes

Informations échangées

Illustration de la construction du chatbot

La réunion débute par la présentation d'un schéma fait par Malaury KESLICK, illustrant la construction du chatbot et l'avancement des différents blocs. Celui-ci répond au besoin de visualisation globale exprimé par l'encadrant industriel lors de la réunion de confrontation des points de vues le 17 décembre 2020.

Le schéma [fig. 1 en annexe] fait émerger quelques questions :

"La demande de précisions permet-elle de rendre le système "récurssif" ?" Oui, si une précision est apportée, alors le même chemin sera à nouveau emprunté par l'information complétée.

"Est-ce que l'on renvoie le premier document trouvé correspondant, ou alors tous, sur toutes les plateformes ?" C'est un choix de conception. On donnera donc tous les fichiers correspondants, sur toutes les plateformes, conformément à l'avis des industriels.

"Une fois le chatbot déployé sur nos serveurs, notre seul travail sera d'alimenter la base de connaissances du QnA si l'on souhaite ajouter des options ?" Exactement. Nous avons choisi de garder ce block à part, même s'il rend la phase de déploiement un peu plus compliquée, afin que lorsque tout sera prêt, vous n'ayez pas à toucher le code en lui-même, puis à tout redéployer. Il suffira simplement d'utiliser l'interface Web du QnA, très intuitive, selon la fiche technique qui sera fournie.

Point sur le GANTT pour la fin de ce projet

Théo PONCET présente ensuite le GANTT qu'il a réalisé pour la fin de ce projet [fig. 2 en annexe]. Notons qu'entre temps, la partie Luis a beaucoup avancé, et il ne reste que quelques petites erreurs à fix pour qu'elles soit parfaitement prête. Nous sommes ainsi en avance sur ce bloc et un tout petit peu en retard sur le QnA. Les parties SharePoint et Windows suivent leur cours et devraient respecter le timing imposé par le document sans problème.

Nous avons choisi de garder une bonne semaine de marge sur la fin du projet pour anticiper d'éventuels petits problèmes de déploiement, et pour ne pas nous surestimer pendant cette période qui sera aussi consacrée aux partiels.

La soutenance ayant lieu le 18 février, nous actons avec l'encadrant industriel d'effectuer une soutenance blanche le jeudi 11 février afin de pouvoir rendre notre présentation la meilleure possible.

Démonstration live des blocks Luis et QnA

La partie démonstration commence par la partie Luis (à laquelle la recherche sur l'arborescence Windows locale à été ajoutée), pratiquement terminée à quelques détails près. Le chatbot peut donc :

- reconnaître une intention de recherche de fichier : à terme il recherchera donc parmi les fichiers Windows partagés, parmi les fichiers Teams et parmi ceux du SharePoint de l'entreprise ;
- reconnaître une salutation et un remerciement ;
- rappeler à l'utilisateur ce à quoi il peut lui servir ;
- rediriger l'utilisateur lorsque l'entrée saisie n'a pas été comprise.

Vous trouverez en annexe [fig. 3] une capture d'écran de ce qui est possible pour le moment (les réponses étant très primaires, elles seront étoffées lorsque tout sera fin prêt).

La démonstration se poursuit par Guillaume IZART, qui présente la partie QnA du chatbot. Il en expose la souplesse et la facilité d'utilisation, notamment au niveau de l'interface web permettant d'entrer de nouveaux couples de questions-reponses. Cette base de connaissance peut d'ailleurs être importée depuis des fichiers la contenant intégralement si nécessaire. Il expose également la facilité de redirection du chatbot dans la bonne direction, en jouant manuellement sur les indices de confiance lors de la phase de test.

Vous trouverez en annexe [fig. 4] une capture d'écran de conversation avec le chatbot reposant sur le QnA.

Prochaines étapes

Nous allons donc continuer de suivre le GANTT présenté plus haut jusqu'à la fin du projet.

Todolist

Description	Responsable
Ignorer le bloc Luis et préparer la fiche technique	Malaury
Terminer le bloc QnA	Guillaume
Travailler sur le bloc SharePoint	Théo
Continuer le rapport	Groupe projet

Prochaine réunion : Jeudi 14 Janvier à 17h.

Annexe

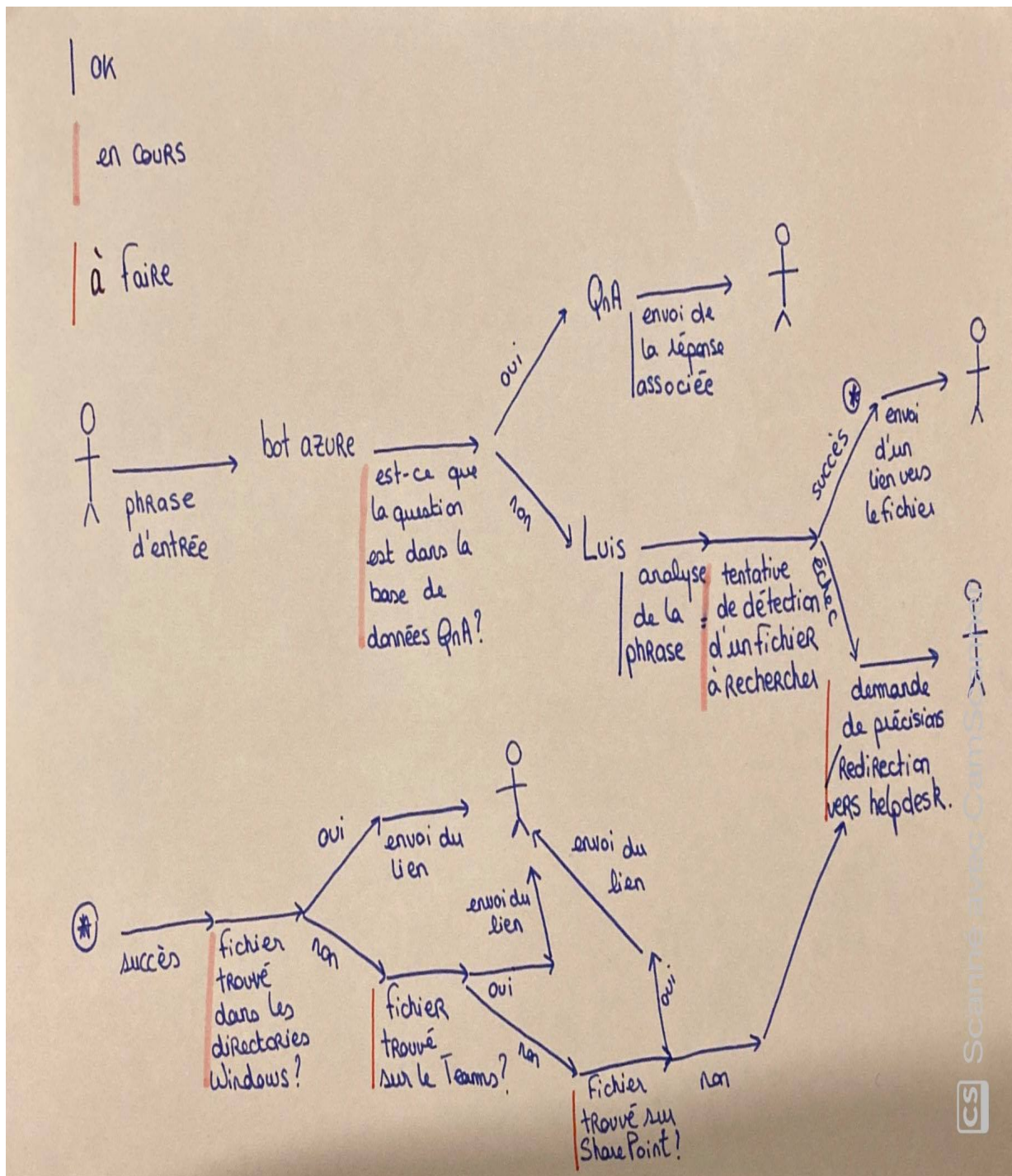


Figure 1: Schéma illustrant la construction du chatbot

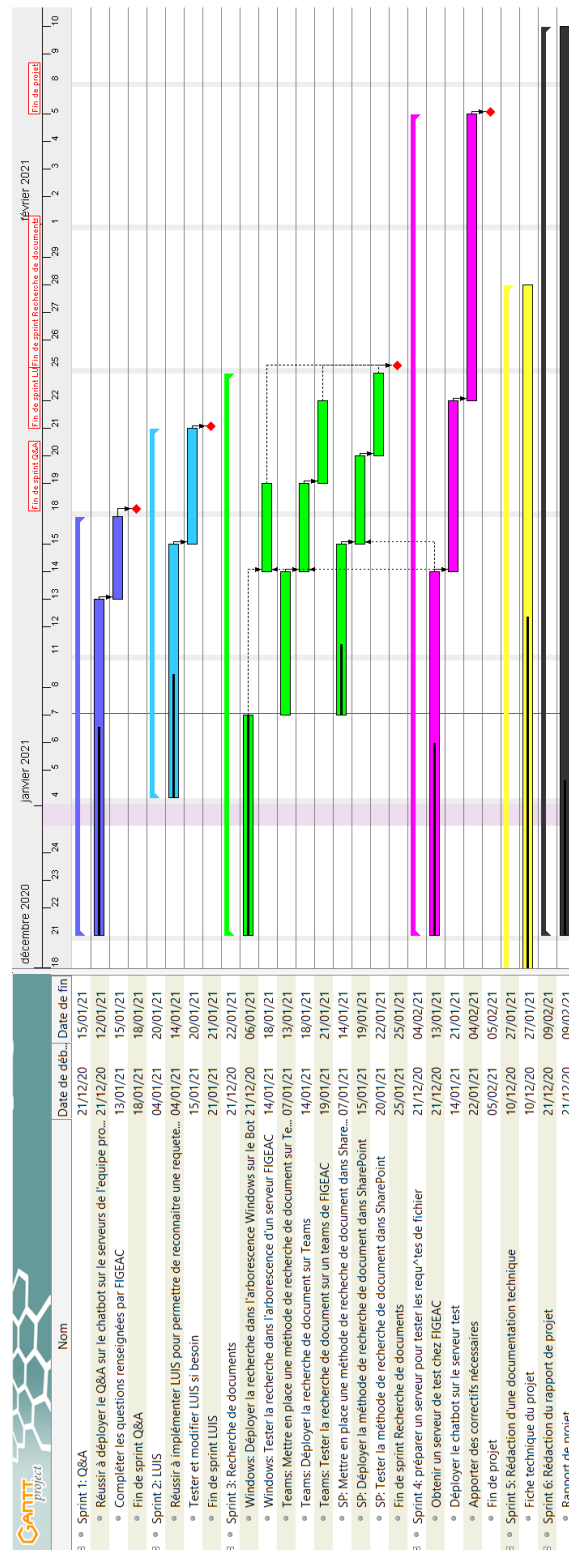


Figure 2: Gantt pour le temps imparti restant

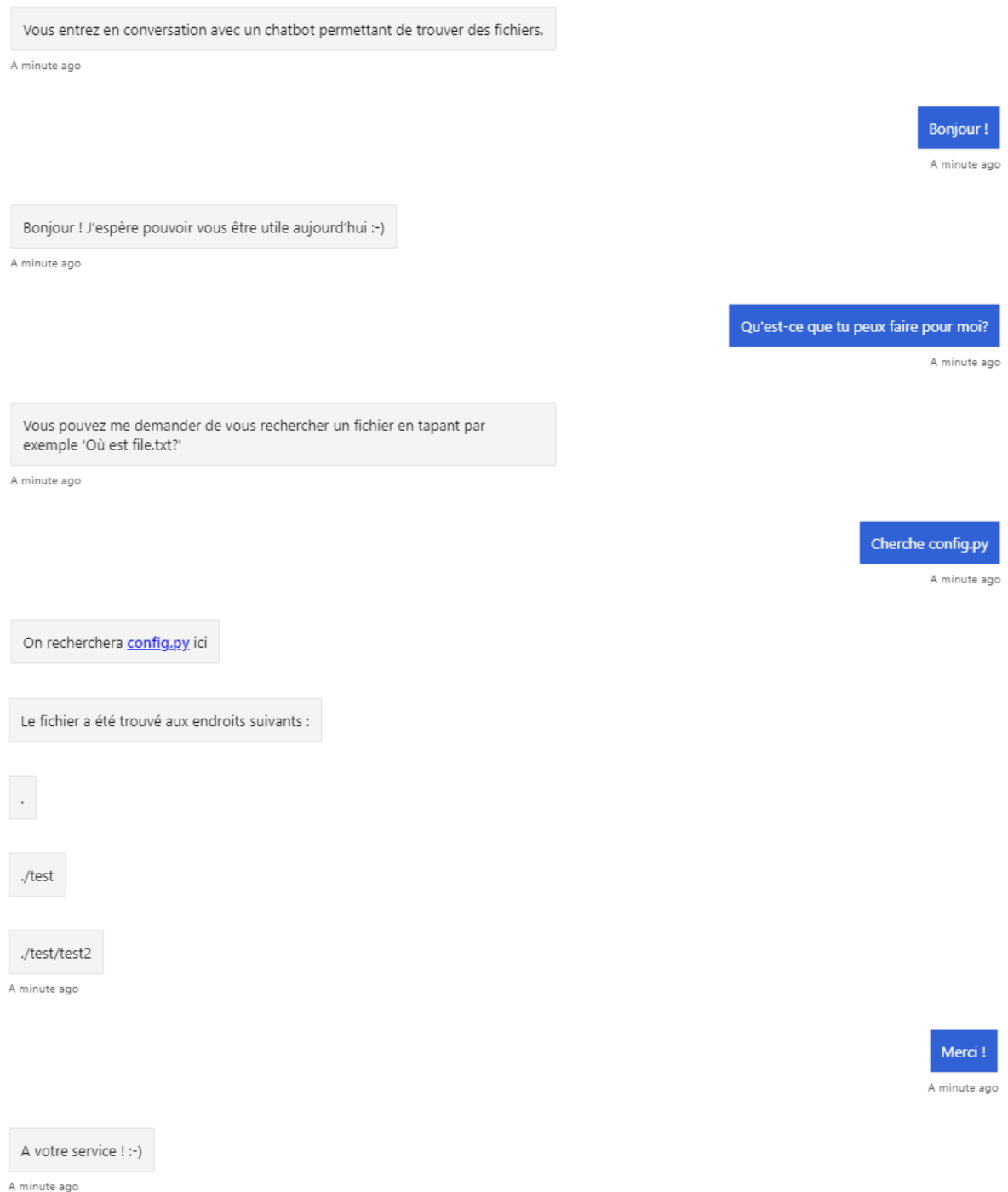


Figure 3: Scénario de discussion avec le bot (Luis + recherche de fichiers locaux Windows)

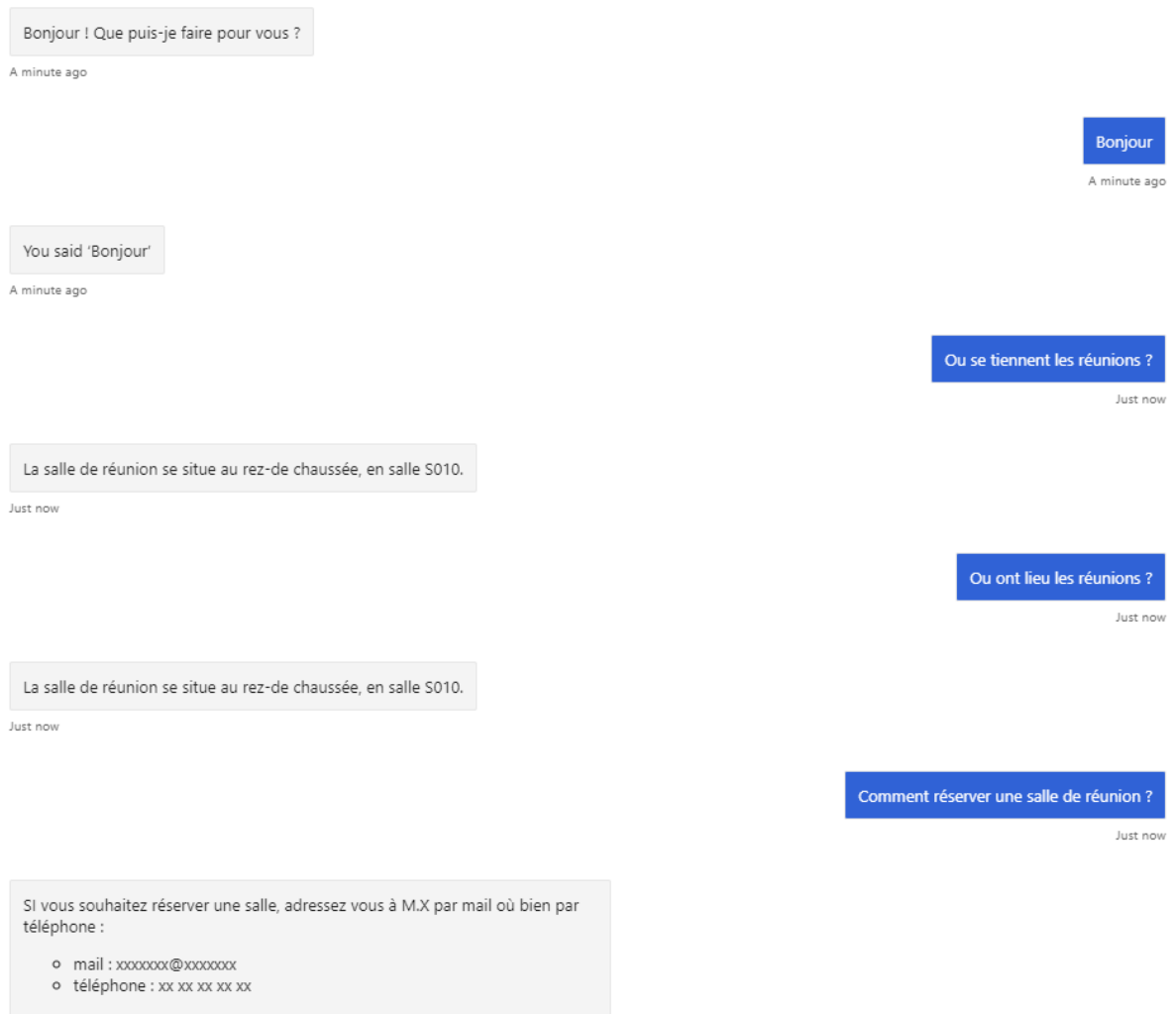


Figure 4: Scénario de discussion avec le bot (QnA)

Résumé

Ce projet en collaboration avec Figeac Aero a pour but la réalisation d'un chatbot évoluant sur un canal Microsoft Teams.

Ce chatbot doit être capable de répondre à des questions couramment posées au support pour permettre de diminuer le nombre de requêtes d'assistance.

Aussi, il doit pouvoir réaliser des recherches de documents dans différentes bases de données comme un serveur Windows, un SharePoint, ou encore dans des documents partagés sur Microsoft Teams.

La demande de l'utilisateur pour une recherche de document doit être comprise avec plusieurs tournures de phrases. Une laxité suffisante doit également être permise pour pallier les éventuelles fautes de frappe et d'orthographe. Notre projet a tout d'abord débuté par une importante phase de recherche sur l'état de l'art des chatbots, ainsi que les différentes technologies potentiellement utilisables pour répondre à notre besoin. Après discussion avec notre encadrant industriel, nous avons choisi d'utiliser la solution Cloud Microsoft Azure.

Cette solution inclut entre autres une partie Question-réponse permettant de gérer les demandes fréquemment soumises au support de l'entreprise, ainsi qu'une partie d'intelligence artificielle permettant de détecter une demande de recherche de document par l'utilisateur.

Dans le cas du Question-Réponse, le chatbot renvoie une réponse prédéfinie, ou peut poser une question pour affiner la demande de l'utilisateur. Si l'utilisateur ne trouve pas la réponse qu'il attend, il sera redirigé vers le support.

Pour la partie apprentissage automatique, le chatbot dispose d'un outil qui lui permet d'analyser la sémantique de la phrase et de déterminer les intentions de l'utilisateur. S'il détecte une demande de recherche de document, il va donc faire une recherche dans les bases de données pour trouver des documents dont le nom correspond à la demande. Le bot est donc capable de chercher des documents dans des serveurs Windows et SharePoint suite à une demande utilisateur.

Mots-clés : "ChatBot", "Cloud Azure", "Recherche de fichier", "Question-réponse"

Abstract

This project in collaboration with Figeac Aero aims to create a chatbot evolving on a Microsoft Teams channel.

This chatbot must be able to answer questions commonly asked to the support team in order to reduce the number of support requests.

Also, it must be able to search for documents in different databases such as a Windows server, SharePoint, or in documents shared on Microsoft Teams.

The user's request for a document search must be understood with several turns of phrase. Sufficient laxity must also be allowed to compensate for possible typing and spelling mistakes. Our project began with an important research phase on the state of the art of chatbots, as well as the different technologies that could be used to meet our needs. After discussions with our industrial manager, we chose to use the Microsoft Azure Cloud solution.

This solution includes, among other things, a question-answer section to manage the requests frequently submitted to the company's support, as well as an artificial intelligence section to detect a document search request by the user.

In the case of the Question-Answer, the chatbot returns a predefined answer, or can ask a question to refine the user's request. If the user does not find the answer they are looking for, they will be redirected to support.

For the machine learning part, the chatbot has a tool that allows it to analyse the semantics of the sentence and to determine the user's intentions. If it detects a document search request, it will then search the databases to find documents whose name corresponds to the request. The bot is therefore able to search for documents in Windows and SharePoint servers following a user request.

Keywords : "ChatBot", "Azure cloud", "File search", "Question and answer"