Rapport de stage

Développement d'un chatbot pour Mattermost via les technologies de Google Cloud

Malaury Keslick

Année 2019–2020

Stage de deuxième année réalisé dans le centre de recherches Inria Nancy Grand-Est

Maître de stage : François Charoy

Encadrant universitaire : Gérald Oster

Déclaration sur l'honneur de non-plagiat

Je soussigné(e),

Nom, prénom : Keslick, Malaury

Élève-ingénieur(e) régulièrement inscrit(e) en 2e année à TELECOM Nancy

Numéro de carte de l'étudiant(e) : 31814820

Année universitaire : 2019–2020

Auteur(e) du document, mémoire, rapport ou code informatique intitulé :

Développement d'un chatbot pour Mattermost via les technologies de Google Cloud

Par la présente, je déclare m'être informé(e) sur les différentes formes de plagiat existantes et sur les techniques et normes de citation et référence.

Je déclare en outre que le travail rendu est un travail original, issu de ma réflexion personnelle, et qu'il a été rédigé entièrement par mes soins. J'affirme 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 me l'accaparer.

Je certifie 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.

Je suis conscient(e) 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, j'encourrais des poursuites non seulement devant la commission de discipline de l'établissement mais également devant les tribunaux de la République Française.

Fait à Laxou, le 30 juillet 2020

Signature :

Rapport de stage

Développement d'un chatbot pour Mattermost via les technologies de Google Cloud

Malaury Keslick

Année 2019–2020

Stage de deuxième année réalisé dans le centre de recherches Inria Nancy Grand-Est

Malaury Keslick
18 Boulevard du Maréchal Foch
54520, Laxou
+33 (0)6 02 22 43 53
malaury.keslick@telecomnancy.eu

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

Inria Nancy Grand-Est
615 Rue du Jardin Botanique
54600 Villers-lès-Nancy
+33 (0)3 83 59 30 00

Maître de stage : François Charoy

Encadrant universitaire : Gérald Oster

Remerciements

Tout d'abord je souhaiterai remercier mon maître de stage, Monsieur François Charoy, qui m'a offert une belle opportunité de poursuivre ma découverte du monde de la recherche malgré la situation de crise que traverse notre pays. Merci à lui d'avoir apporté sa bonne humeur de façon lors de nos réunions hebdomadaires en visio-conférence, elles m'auront permis de garder un rythme essentiel à ma réussite de ce stage.

Je tiens également à remercier tout particulièrement Madame Clélie Amiot pour son suivi inégalable et son accueil si chaleureux.

Un grand merci à toute l'équipe Coast du Loria, que j'ai pu rencontrer à travers mon écran et lors notre pique-nique, pour sa joie de vivre et nos échanges toujours intéressants.

Enfin, je souhaite adresser mes remerciements à toute l'équipe pédagogique et administrative de TELECOM Nancy pour avoir permis la réalisation de ce stage.

Table des matières

1 Introduction

Les outils de communication en ligne sont de nos jours de plus en plus présents, qu'ils soient à usage personnel ou professionnel. C'est dans ce cadre qu'on été introduit les premiers chatbots dans les années 1960. Assistants virtuels répondant automatiquement aux messages d'un ou de plusieurs utilisateurs, ils étaient à leur création très primaires, et ne pouvaient répondre qu'à un faible nombre de messages mais ils sont aujourd'hui toujours plus performant et sont utilisés dans pratiquement tous les domaines, allant de la modération de messages à l'assistance à un client souhaitant réserver son billet de train.

C'est dans le cadre d'une étude sur les collaborations entre systèmes informatiques et utilisateurs que s'inscrit le projet d'évaluation de la confiance dans la collaboration entre les chatbots et les humains sur lequel j'ai travaillé lors de ces neuf semaines de stage au sein de l'équipe Coast du Loria. Ce stage s'inscrit dans mon cursus d'élève ingénieur au sein de TELECOM Nancy, se déroulant au terme de ma deuxième année durant laquelle je me suis spécialisée en **I**ngénierie **L**ogicielle. Ayant déjà travaillé au sein de cette équipe de projet lors de mon PIDR, j'ai eu la chance de pouvoir découvrir un autre aspect de ce domaine de recherche lors de ces quelques semaines.

Tout d'abord j'effectuerai une présentation du centre de recherches, puis je présenterai la problématique de mon stage en introduisant la notion de chatbot et en explicitant le projet que l'on m'a confié lors de ce stage, à savoir le développement d'un agent conversationnel à l'aide des outils Google Cloud. Je détaillerai ensuite leur réalisation et validation avant de dresser le bilan de ce stage et de finalement conclure.

2 Présentation du centre de recherches

L'INRIA (**I**nstitut **N**ational de **R**echerche en **I**nformatique et **A**utomatique) est un établissement français public de recherche en sciences et technologies du numérique. Il a été fondé en 1967 et est affilié au Ministère de *l'Education Nationale, de l'Enseignement supérieur et de la Recherche*. Jusqu'en 2008, son activité est centralisée dans son siège social à Rocquencourt, puis sont ensuite créés huit centres de cherche indépendants dans toute la France : ceux de Bordeaux, Grenoble, Lille, Nancy, Paris, Rennes, Palaiseau et Biot. Un premier centre hors de la France a également été inauguré en 2012, au Chili. Ces 9 pôles de recherche rassemblent plus de 3500 scientifiques travaillant au sein de plus de 200 équipes-projets en partenariat avec de grandes universités de recherche et des partenaires industriels. En 2018, tous ces ambitieux projets ont été soutenus d'un budget de 236 millions d'euros.

Chaque centre de recherche a ses propres positionnements et enjeux stratégiques. Nous allons nous concentrer sur le cas du centre Nancy - Grand Est, puisqu'il est celui où j'ai réalisé mon stage. Sous la direction de Bruno Lévy, ce centre rassemble plus de 400 scientifiques issus de 45 nationalités différentes, au service de la recherche et de l'innovation. Ses axes scientifiques prioritaires sont l'intelligence algorithmique et les interactions Hardware-Software. De plus, le centre est en étroit lien avec l'Université de Lorraine pour une excellente dynamique de site, et un travail est fait sur le développement de l'antenne sur le site de Strasbourg (qui sera axée sur la médecine personnalisée), avec de nouveaux partenariats stratégiques et une possibilité accrue de coopérations transfrontalières avec l'Allemagne.
Le siège Nancy - Grand Est compte un grand nombre de partenaires, qu'ils soient académiques (universités de Lorraine et Strasbourg, iCube, CNRS, institut Max-Planck pour l'informatique de Sarrebruck) ou entrepreunariaux allant des Startups aux grands groupes comme Orange, Thales ou encore Facebook.

Les 22 équipes de Nancy travaillent sur des thèmes divers et variés allant de la génétique à la cryptologie. Lors de mon stage, j'ai intégré l'équipe *COAST*, composée d'une petite trentaine de personnes (enseignants-chercheurs, doctorants, ingénieurs, stagiaires). Dirigée par François Charoy, cette équipe travaille sur les systèmes collaboratifs, la gestion des processus métiers, la réplication optimiste et l'informatique orientée service. Les axes de recherche de l'équipe sont alors les suivants [6] :

- La gestion sure et efficace de données collaboratives à large échelle en terme de quantité de données et d'utilisateurs de ces données ;
- La composition de services orientés données pour permettre la construction d'applications à l'échelle du Web et pour donner des garanties sur le fonctionnement de ces applications ;
- Le support à la construction d'environnements collaboratifs pour lesquels on puisse déterminer un niveau de confiance et de sécurité (axe sur lequel mon stage était dirigé).

3 Problématique

Lors de mon stage, j'ai eu a développer un robot conversationnel dans un outil permettant de gérer les échanges entre utilisateurs, Mattermost. Cette application est une plateforme permettant de créer des équipes de personnes afin qu'elles puissent converser entre elles de manière organisée. Dans une équipe, on peut aller créer différents canaux qui permettront de bien distinguer chaque espace de travail. De plus, on peut également attribuer un certain niveau de confidentialité à un canal (accessible à tous les membres ou alors uniquement sur invitation), et envoyer des messages privés à chaque utilisateur de son équipe (voir fig. 3.4). Ce bot a pour but d'être capable de retrouver dans une conversation ou dans un canal des informations données par un utilisateur pour les transmettre à un autre utilisateur en fonction des demandes. Ce bot devait être implémenté via les fonctionnalité d'un cloud, pour ma part celui de Google, tandis que mon co-équiper Frédéric Vaz effectuait le même travail en utilisant les services de Microsoft Azure.

Dans cette partie je vous présenterai d'abord ce que sont les chatbots avant de préciser la problématique sur laquelle mon travail a porté.

3.1 Les chatbots

Dans un premier temps, je vais vous présenter ce qu'est un chatbot, et quelles peuvent être ses utilités. Francisé comme "agent conversationnel", un chatbot est un programme informatique capable de simuler un échange avec un ou plusieurs partis, que ce soit par échange vocal ou textuel. Ils sont aujourd'hui utilisés à de nombreux niveaux, que ce soit pour faciliter la communication inter-entreprise, ou pour rendre un service à des clients, comme le montre l'image suivante [fig. 3.1].

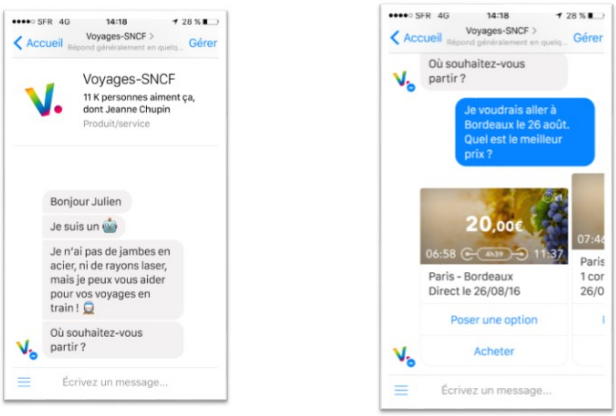


FIGURE 3.1 – Exemple d'un chatbot de la SNCF destiné à la clientèle [8]

Le premier chatbot, nommé Eliza [fig. 3.2], a été mis au point en 1966 par Joseph Weizenbaum, un professeur au MIT, et pouvait uniquement reformuler les paroles de l'utilisateur sous forme interrogative, à la manière d'un psychothérapeute rogérien [1]. Aujourd'hui, cette technologie a beaucoup évolué et peut désormais proposer un large choix d'options à l'utilisateur, allant de la demande de la météo à la réservation d'un billet d'avion. Il est fortement apprécié et répandu dans le monde du service à la clientèle, puisqu'une fois développé, il est un moyen très peu coûteux de venir en aide de manière instantanée à des dizaines, centaines, voire milliers d'utilisateurs à la fois.

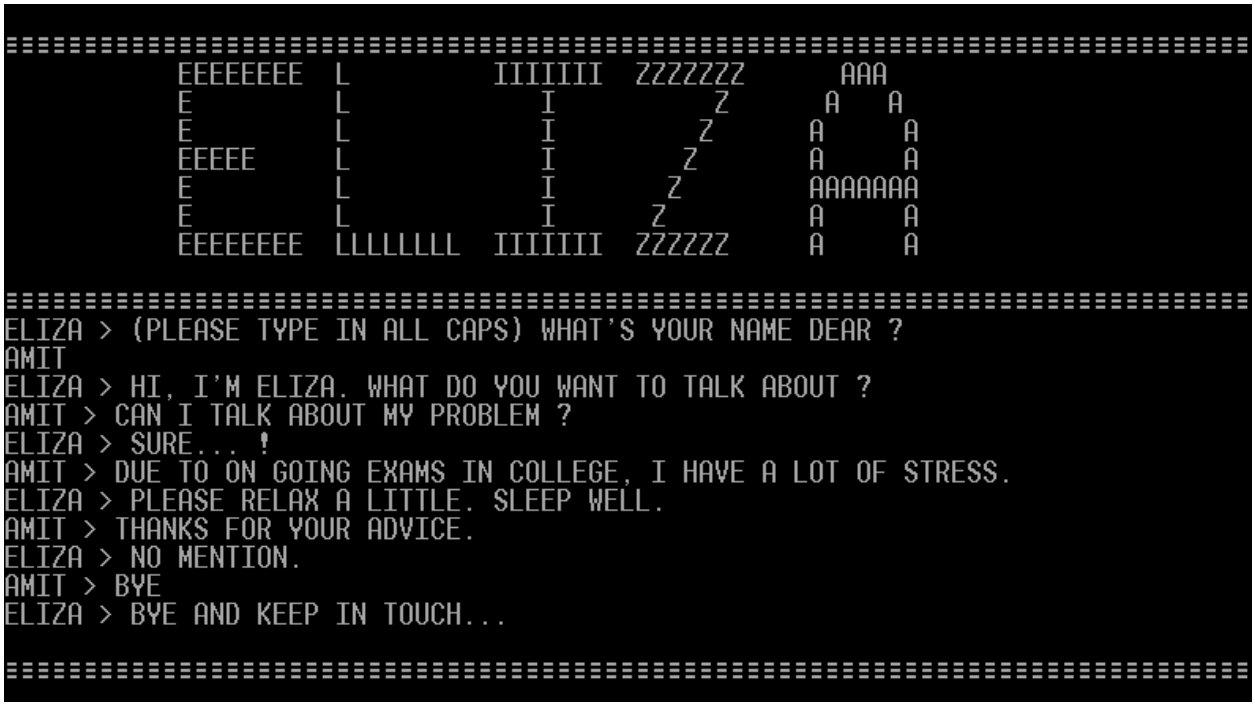


FIGURE 3.2 – Exemple de dialogue avec Eliza, le tout premier chatbot [4]

Il existe aujourd'hui deux types d'agent conversationnel. Le premier repose sur un set de règles précises uniquement, et a donc un intérêt très limité puisqu'une seule formulation un peu exotique perdrait rapidement le bot. En revanche, le second type repose sur deux choses : des règles

et l'intelligence artificielle. Tout d'abord, on inculque à notre robot une série de règles lui permettant d'identifier et de réagir à certains comportements précis de l'utilisateur. Ensuite, c'est la part d'intelligence artificielle qui pourra reconnaître d'autres comportements, en les comparant avec tout ce qu'elle connaît déjà, pour en déduire la réponse adaptée. Ce nouvel échange sera ajouté aux règles (*machine learning*), qui seront donc de plus en plus fiables à chaque apprentissage de l'agent. Ce type de bot devient donc meilleur à chaque nouvelle interaction, et c'est pourquoi il est privilégié de manière générale. Vous pouvez trouver un schéma explicatif du processus de *machine learning* ci-dessous [fig. 3.3].

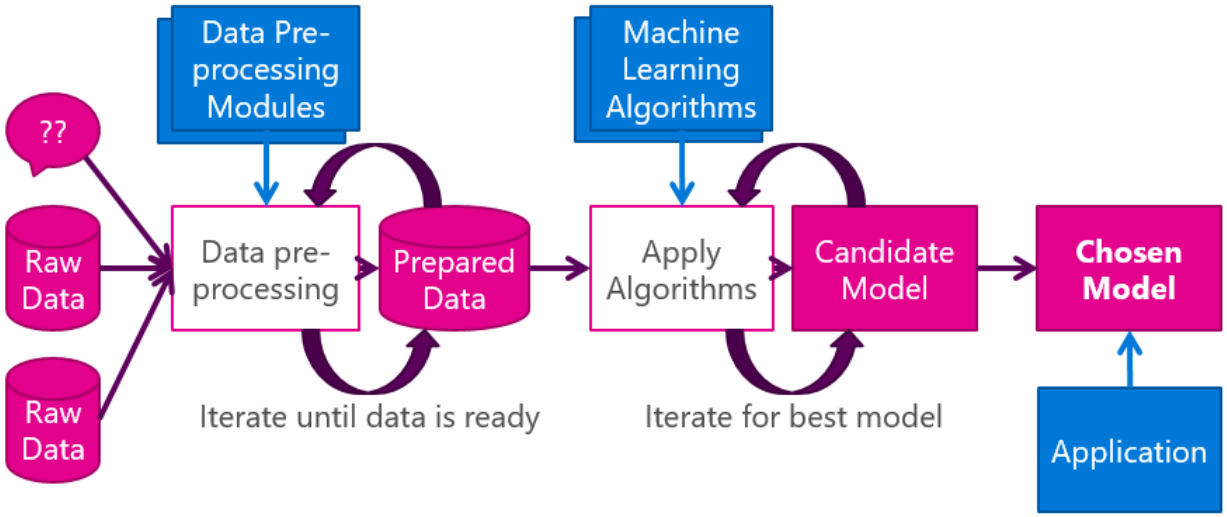


FIGURE 3.3 – Schéma explicatif du processus de *machine learning* [5]

3.2 Projet confié pendant le stage

Lors de ce stage, ma tâche était donc de développer un agent conversationnel intégré dans l'application Mattermost [fig. 3.4]. Mon bot devait pouvoir s'intégrer dans un canal avec plusieurs utilisateurs et se manifester lorsque son aide peut s'avérer utile, comme pour transmettre des informations précédemment données, rappeler une horaire de réunion,...

Le développement était idéalement demandé en NodeJS ou en Python, et j'ai pour ma part choisi la première option. Finalement, comme expliqué précédemment, j'ai pour ma part utiliser les services cognitifs de Google Cloud pour mener à bien mon projet.

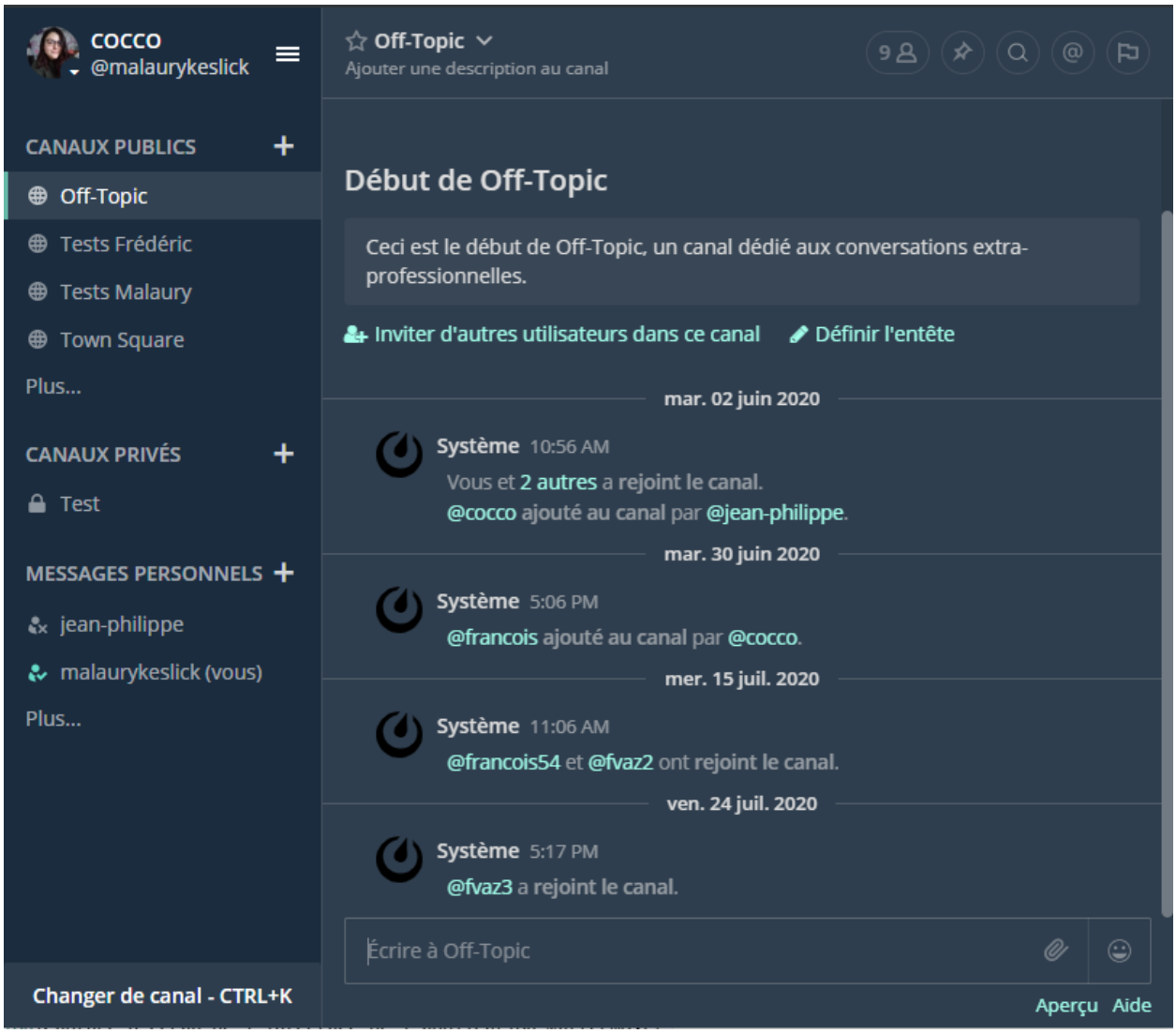


FIGURE 3.4 – Capture d'écran de l'interface de l'application Mattermost

4 Réalisation et validation

4.1 Environnement matériel et logiciel

Sur un total de neuf semaines de stage, j'en ai passé sept et demi en télétravail depuis mon domicile étudiant, à cause de la situation encore compliquée que présentait la crise de la COVID-19. J'ai choisi d'aller sur site pour la fin de mon stage afin de récupérer une ambiance de travail et un cadre sérieux, ce qui commençait à me manquer chez moi. J'ai effectué la totalité de mon travail depuis mon ordinateur personnel, un Dell Inspiron 5000 17 pouces, ce grand écran s'étant avéré très pratique dans des conditions où je ne pouvais pas faire de double-écran.

Pour travailler, j'ai essentiellement travaillé sur la plateforme de Google Cloud, utilisé l'éditeur de texte Atom et GitLab comme outil de gestion des versions.

4.2 Prise en main de Google Cloud et ses API

Google Cloud Platform (GCP) est une plateforme fournissant près d'une centaines de produits/services informatiques aux entreprises, professionnels de l'informatique et développeurs. Le cloud computing permet d'utiliser des produits et logiciels alors qu'ils existent dans des centres de données distants. Les utilisateurs peuvent donc y avoir accès depuis une interface en ligne comme la GCP, ce qui présente de nombreux avantages. Tout d'abord, aucun produit n'a besoin d'être installé directement sur la machine de l'utilisateur, et ainsi elle ne perd pas en stockage et en efficacité. D'autre part, la facilité de création d'un projet Google Cloud mêlant plusieurs services est indéniable, et la gestion des accès est également très simple d'utilisation.

Travailler avec la GCP permet d'accéder aux services de plus de cinquante API (*Application Programming Interface*) allant de l'accès au stockage à l'analyse d'images basées sur le *machine learning*. Dans le cadre de ce projet, j'ai eu recours aux API *Cloud Functions* et *Dialogflow*, et j'ai en parallèle utilisé le service *Firebase* permettant d'intégrer une base de données en temps réel à mon application.

4.2.1 Google Cloud Functions

Google Cloud Functions (GCF) est une solution FaaS (*Functions as a Service*) permettant de créer des fonctions répondant aux événements cloud sans avoir à gérer ni de serveurs ni d'environne-

ments d'exécution [2]. C'est le service que j'ai utilisé pour gérer deux axes de mon projets :

— l'intégration de mon chatbot dans Mattermost ;
— la gestion des réponses (*fulfillment*) de mon bot à un type de message en particulier (cette partie sera plus amplement explicitée dans la prochaine partie).

4.2.2 Dialogflow

Dialogflow est l'outil majeur que j'ai utilisé lors de ce stage. C'est une plateforme permettant la création et l'intégration d'agents conversationnels au sein d'applications mobiles et Web, de système de réponse vocale interactive,... Ce système peut analyser les entrées provenant de clients (qu'elles soient audio ou textuelles) et répondre de différentes manières en fonction de la manière dont il est programmé.

Lorsque *Dialogflow* reçoit une entrée, il la compare aux "intents" qu'ils lui ont été enseignés. Un intent comprend un titre, éventuellement des contextes [fig. 4.1], des phrases d'entraînement [fig. 4.3], d'éventuels paramètres [fig. 4.4] et enfin des réponses [fig. 4.4]. Nous allons ici voir l'utilité de chacun de ces éléments via l'intent "Présentation". Il existe deux types de contextes : les sortants qui permettent d'indiquer au système qu'il doit retenir l'une des informations qu'on lui a fourni, et ce pendant un certain nombre de tours ; et les entrants qui indique quel est l'élément (qu'il est censé déjà connaître) qu'il doit récupérer pour le réutiliser dans la suite du processus.

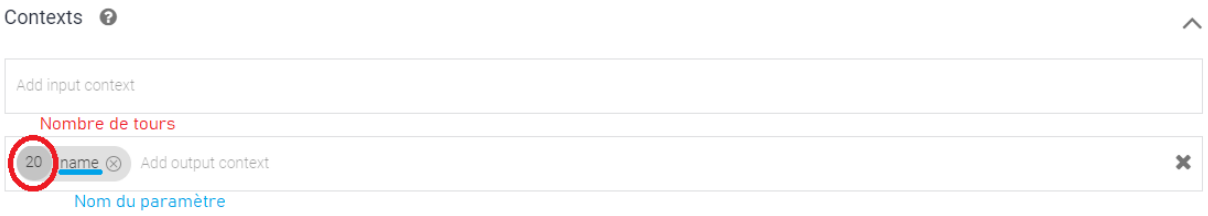


FIGURE 4.1 – Exemple de contexte pour l'intent "Présentation"

Ensuite, on donne à l'intent quelques phrases types reflétant ce que l'utilisateur pourrait dire, et l'on repère les mots-clés importants par un type d'entité. Différentes entités sont déjà existantes dans *Dialogflow*, comme les dates, les villes,.. et l'on peut en créer de nouveaux personnalisés, comme par exemple des salles de réunions, des types d'événements.. comme l'indique l'exemple suivant [fig. 4.2].

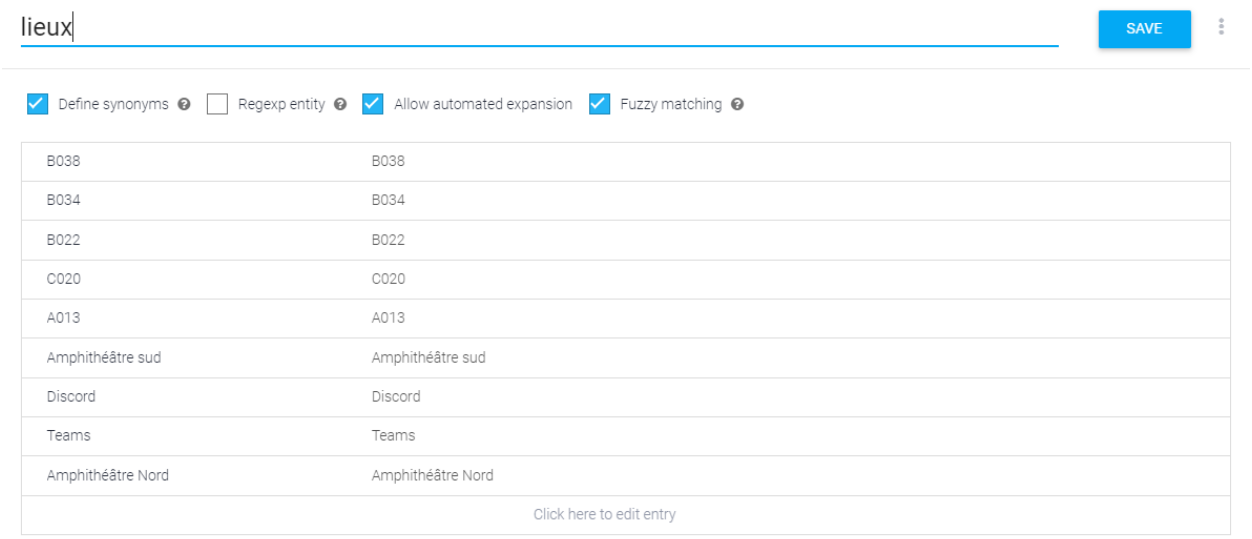


FIGURE 4.2 – Exemple d'entité pour les différents lieux (réels ou virtuels) d'événements

Voici ainsi un exemple de phrases d'entraînements que j'ai fournies dans le cadre de mon intent "Présentation".

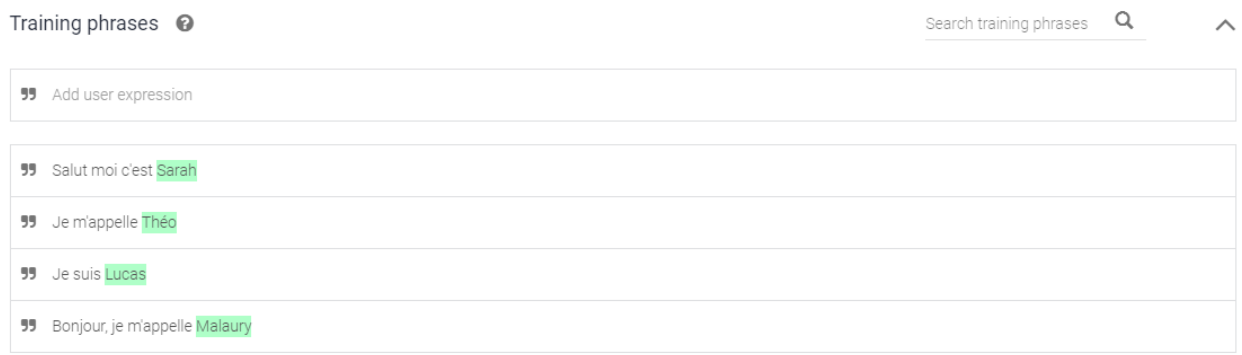


FIGURE 4.3 – Exemples de phrases d'entraînement pour l'intent "Présentation"

À partir de ces éléments, lorsqu'un utilisateur saisit l'une de ces phrases (ou quelque chose y ressemblant), le système reconnaît le nom de l'utilisateur et pourra alors produire des réponses personnalisés. Grâce aux contextes précédemment présentés, il pourra d'ailleurs le retenir pendant 20 tours après la saisie de cette phrase. À noter qu'il existe la case "Required" dans la liste des paramètres, comme vous pouvez le voir ci-dessous, qui sert à exiger une information. Ainsi, si elle n'est pas donnée, on peut programmer un message qui pourrait être ici "Bonjour ! Comment t'appelles-tu ?". Pour répondre à l'utilisateur, il existe alors plusieurs options. La première, que nous allons voir ici, est utilisable lorsque la réponse ne dépend pas des expressions entrées par l'utilisateur. Par exemple dans notre cas, le système aura un panel de réponses toutes faites à renvoyer, et ce peu importe le prénom entré.

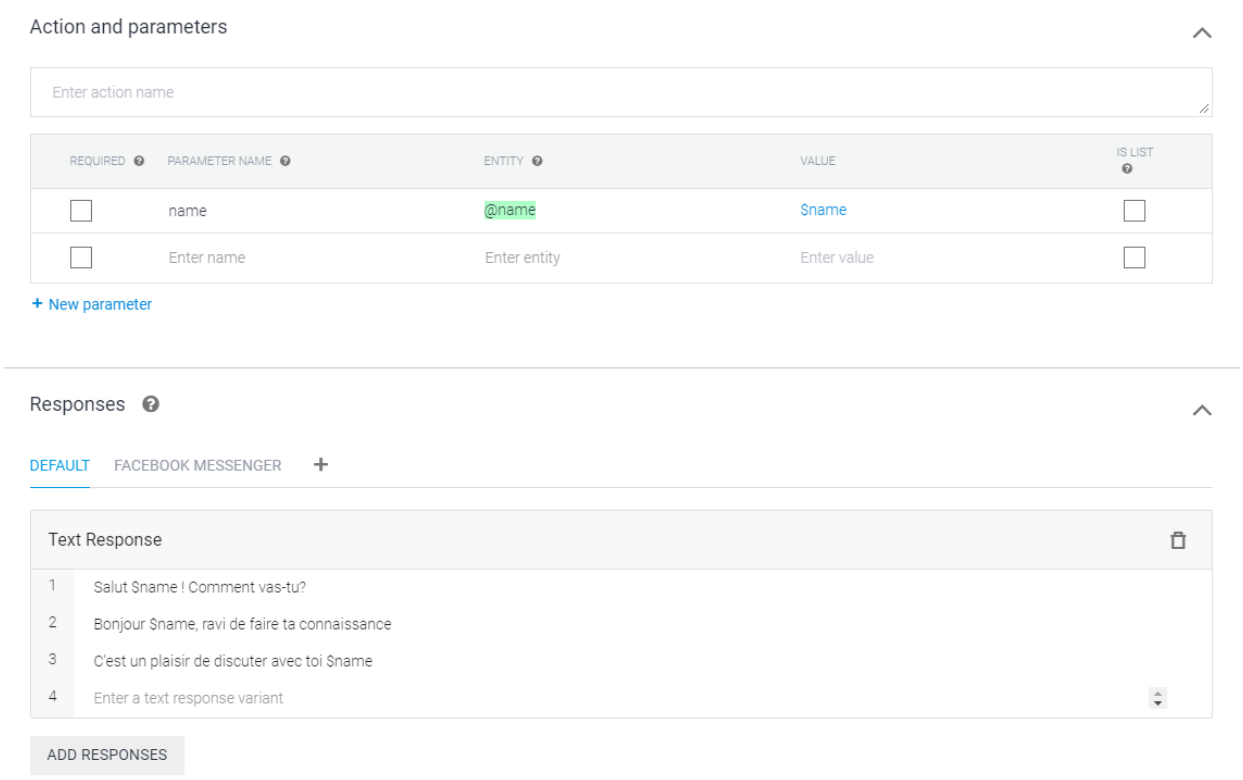


FIGURE 4.4 – Démonstration de l'utilisation des expressions dans le cadre de réponses simples pour l'intent "Présentation"

Dans le cas où l'on doit travailler sur les différents paramètres avant de pouvoir produire une réponse (ou si l'on doit faire une entrée dans la base de données *Firebase* par exemple), il faut alors autoriser les appels au webhook disponible dans la partie fulfillment de *Dialogflow*. Le webhook en question est une *Google Cloud Function*, qui reçoit une requête HTTP provenant du client avec les éléments entrés par l'utilisateur sous format JSON. C'est ce qui va permettre par exemple, de déterminer si la date entrée est valide, si la salle de réunion demandée est disponible à l'instant t, d'entrer des informations dans la base de données..

Je vais maintenant vous présenter les différents intents que j'ai créés dans le cadre de ce stage, et les éventuelles fonctions de fulfillment associées.

Programmation d'un événement

Pour ce projet, je me suis concentrée sur des échanges d'informations à propos d'évènements au sein d'une équipe ou d'une entreprise. Ainsi, j'ai créé une entité *event* [fig. 4.5], afin que mon système puisse reconnaître les différentes catégories pouvant exister. Un utilisateur de mon bot peut donc choisir de programmer une réunion, une conférence ou une visio-conférence.

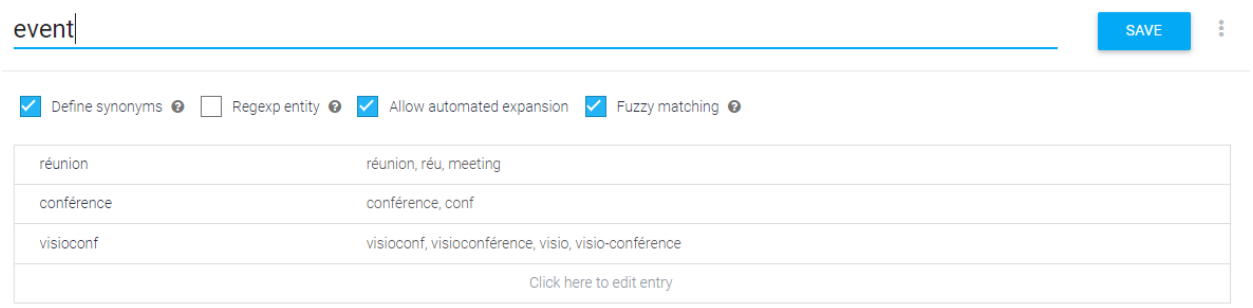


FIGURE 4.5 – Entité *event* créée pour la programmation d'évènement

Afin de pouvoir programmer ces différentes catégories d'évènements, j'ai eu besoin de créer deux autres entités, les lieux et le nom du bot. La première sert à ce que mon robot reconnaisse un nom de salle, d'amphithéâtre, ou même la plateforme utilisée si l'évènement se produit à distance. J'ai choisi d'utiliser des exemples réalistes dans le cadre de mon équipe au Loria (notamment au niveau des noms de salle) et de la situation actuelle (réunion en visio), mais c'est quelque chose qui peut être très facilement ajustable en ajoutant de nouveaux exemples dans cet entité. Cet entité a déjà été présentée un peu plus haut [fig. 4.2].

Ensuite, l'entité responsable du nom du bot est celle qui permettra au bot de comprendre si l'on parle de lui a un moment donné dans une phrase, ou si l'on cherche à le mentionner pour s'adresser directement à lui [fig. 4.6]. Dans le cadre de ce stage, j'ai choisi de lui donner un nom générique (*chatbot*) mais il est bien sûr personnalisable en allant le modifier directement dans l'entité et dans les paramètres d'intégration.

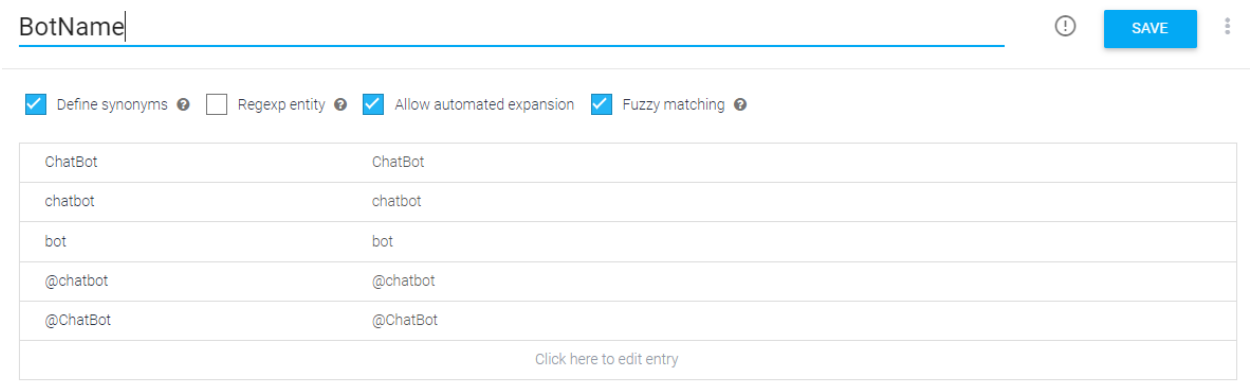


FIGURE 4.6 – Entité *BotName* créée pour le dialogue avec le bot

Ces entités créées de toute pièce, plus l'entité déjà connue par *Dialogflow*, *@sys.date-time* responsable de la reconnaissance de dates et heures, m'ont permis de créer l'intent *EventHandling*. Cet intent est celui qui me permet de reconnaître la volonté d'un utilisateur de programmer un évènement. Si au départ je n'avais pris en compte que le cas où l'on s'adressait directement au Bot, j'ai ensuite rajouté le cas où ces informations se glissent au sein d'une conversation entre plusieurs utilisateurs. Voici un aperçu des phrases d'entraînement que j'ai soumises à mon système [fig. 4.7] :

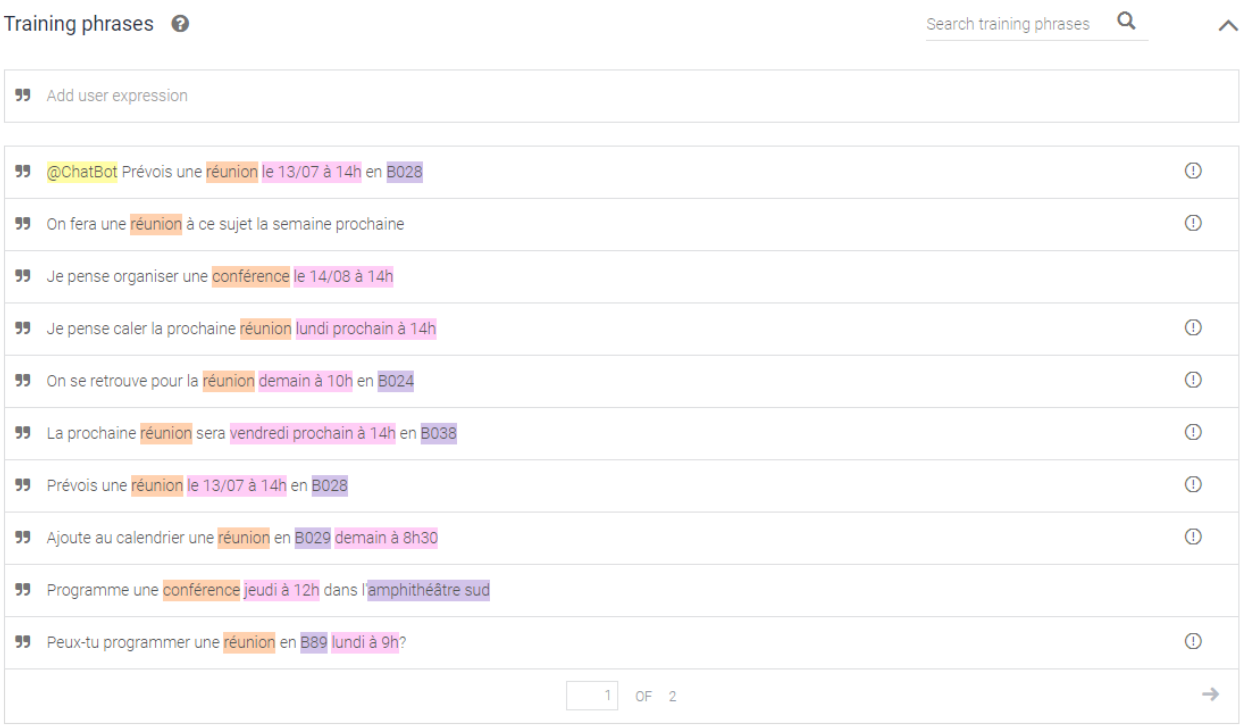


FIGURE 4.7 – Phrases d'entraînement pour l'intent *EventHandling*

Il s'agit ensuite pour le bot de s'assurer qu'il a bien recueilli toutes les informations nécessaires pour discriminer un évènement, c'est-à-dire son type, son lieu, et sa date (jour et heure, sachant que si l'horaire n'est pas précisée, elle sera automatiquement à 12h). Si le système se rend compte qu'il lui manque l'une de ces informations, une *prompt phrase* est alors envoyée à l'utilisateur, lui demandant le paramètre manquant [fig 4.8].

Une fois tous les paramètres reçus, ces informations sont envoyées sous format JSON à la partie *fulfillment* de l'application. Ma *Google Cloud Function* responsable de cette partie est écrite en NodeJS, et repose essentiellement sur le package *dialogflow-fulfillment*, qui permettra d'envoyer une réponse au système. J'utilise également le service Google *Firebase* afin d'avoir une base de données en temps réel des évènements ajoutés au calendrier. Cette partie sera détaillée un peu plus tard [section 4.2.3].

Ainsi, une fois les informations reçues dans la requête, une nouvelle entrée est créée dans la base de données. J'aurais aimé que le système vérifie si la date est valide, que l'évènement en question n'ait pas déjà été programmé et enfin que la salle demandée soit disponible à ce moment-là avant d'écrire dans la base, mais j'ai rencontré quelques problèmes en Javascript qui m'en ont empêché.

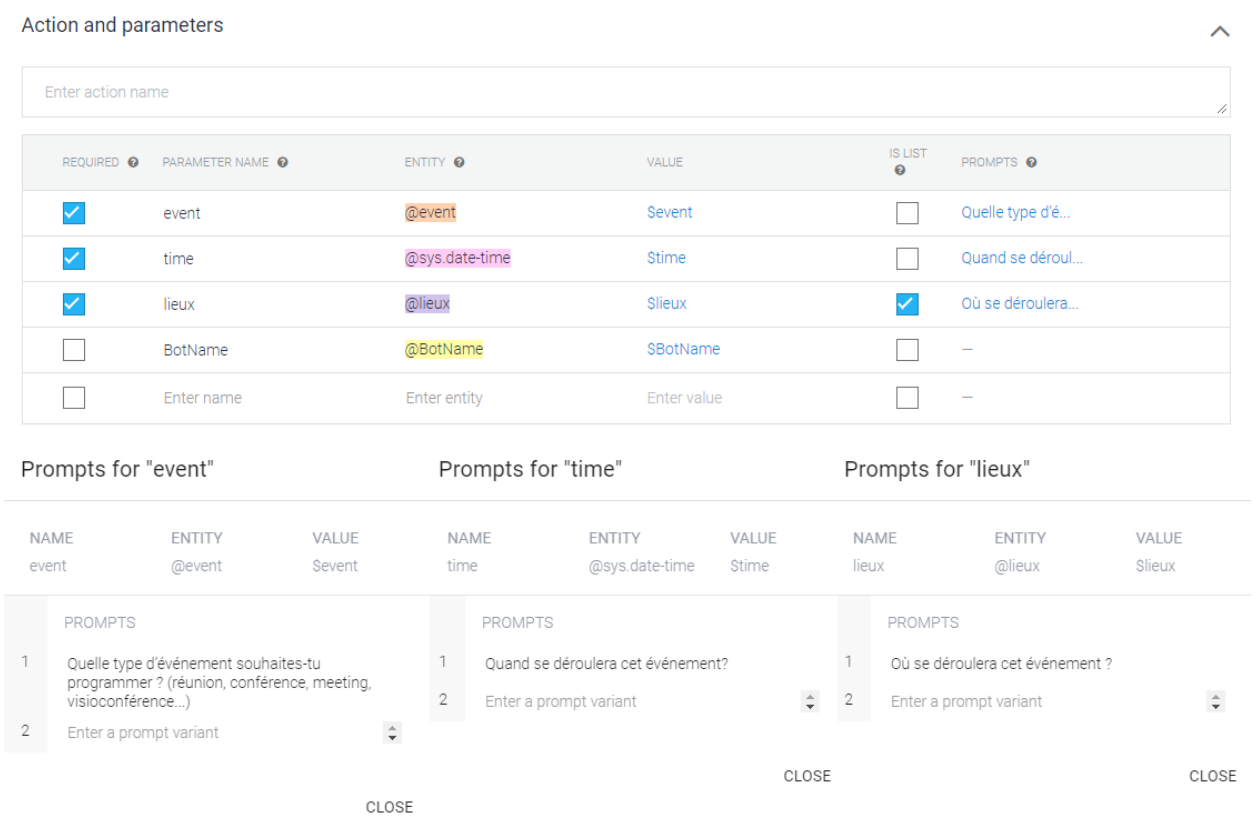


FIGURE 4.8 – Paramètres et *prompt phrases* pour l'intent *EventHandling*

Je présenterai ce problème dans mon bilan [section 5].

Demande d'informations à propos d'un événement

Une fois des évènements ajoutés à la base de données, nous voulons naturellement pouvoir en lire les informations. Ainsi, un utilisateur peut se poser diverses questions, par exemple sur l'heure d'une réunion, sur la disponibilité d'une salle, sur le programme d'une journée en particulier... C'est ce que contrôle mon intent *ReadFromDB*, dont les phrases d'entraînement vous sont présentées ci-dessous [fig. 4.9].

Dans ce cas de figure, aucun paramètre n'est requis, puisque l'analyse de l'information voulue se fera sur le nombre et le type de paramètres proposés par l'utilisateur. J'ai ici rajouté un paramètres de l'entité système *@sys.any* qui me permet de reconnaître quand certaines informations très précises sont demandées, par exemple une heure ou un lieu. Comme le montre la correspondance du tableau suivant [fig. 4.10], ces paramètres sont surlignés en vert dans les phrases d'entraînement précédentes. Ce nouveau paramètre me servira dans la distinction faite dans la partie fulfillment, lorsque le bot cherchera à comprendre quelle est l'information recherchée.

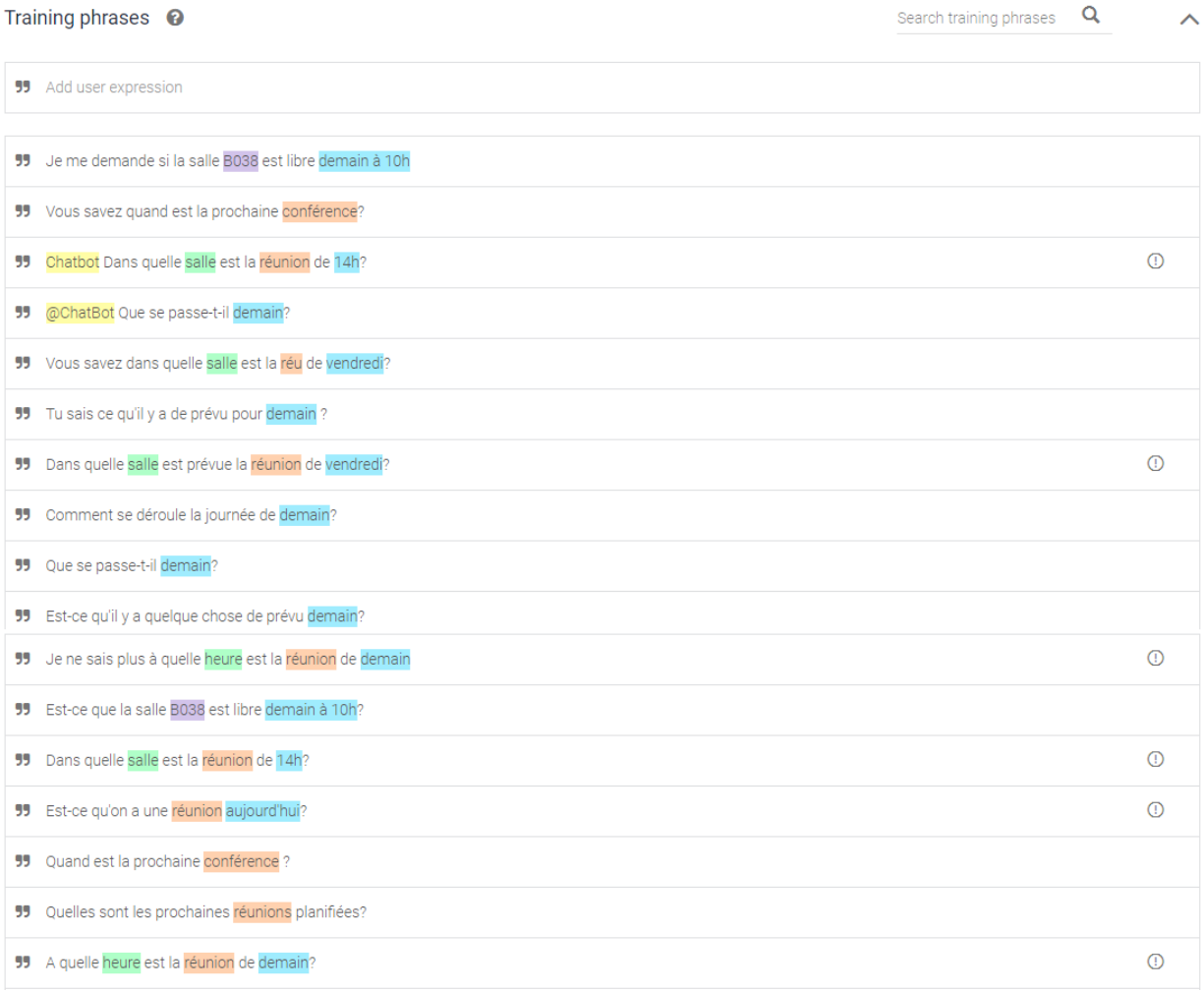


FIGURE 4.9 – Phrases d'entraînement pour l'intent *ReadFromDB*

REQUIRED	PARAMETER NAME	ENTITY	VALUE
☐	event	@event	$event
☐	jour	@sys.date-time	$jour
☐	lieux	@lieux	$lieux
☐	any	@sys.any	$any
☐	BotName	@BotName	$BotName

FIGURE 4.10 – Paramètres pour l'intent *ReadFromDB*

En fonction des paramètres reçu, on peut chercher à déterminer ce que l'utilisateur souhaite savoir :

- si seul le type d'évènement est renseigné, alors le système répondra à la question "Quand est le prochain [type d'évènement] ?" ;
- si seule la date est renseignée, alors le système répondra à la question "Quel est le programme du [jour donné] ?" ;
- si la date (sous-entendu avec l'heure) et le lieu sont donnés, alors le système répondra à la question "Est-ce que [lieu] est disponible le [date] ?" ;
- et enfin si un paramètre *@sys.any* a été reconnu, c'est alors que les champs type et date ont également été remplis, et le système répondra à une question du type "A quelle heure/dans quelle salle est la réunion de demain ?".

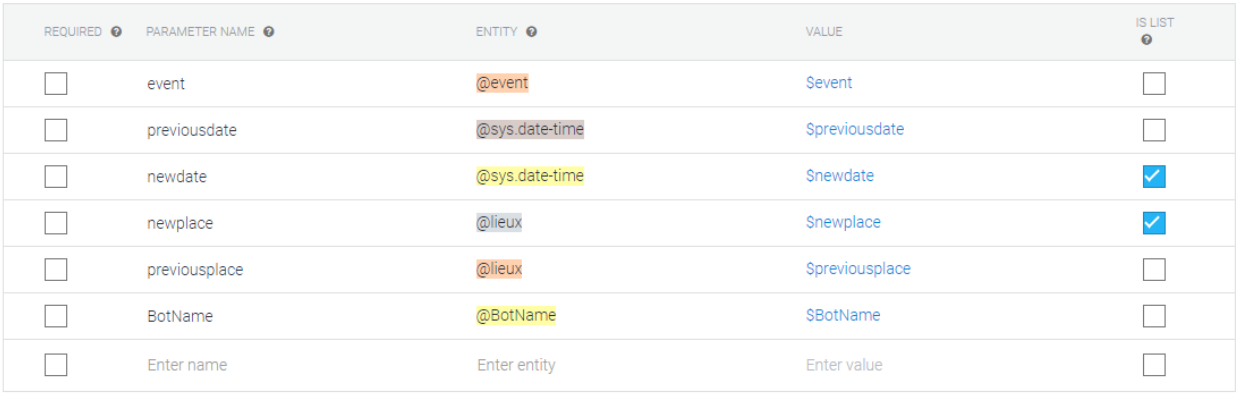
Cette fonction requiert de lire dans la base de données *Firebase*, ce mécanisme sera explicité un peu plus tard.

Modification d'une entrée dans la base de données

Il peut également arriver qu'un utilisateur souhaite modifier l'heure, la salle de de rendez-vous pour un évènement en particulier. C'est l'intent *ModifyDB* qui se charge de repérer cet intention chez l'utilisateur. De nouveau, le pannel des phrases d'entraînement est assez large pour aller de l'adressage direct au bot à du repérage même au sein d'une conversation entre collègues [fig. 4.11].

FIGURE 4.11 – Phrases d'entraînement pour l'intent *ModifyDB*

Le procédé est alors le suivant : on récupère les informations nécessaires à la discrimination de l'évènement à modifier, puis l'on parcourt la base de données à la recherche de l'entrée correspondante. On récupère alors toutes les informations, on actualise celles demandées, on supprime l'entrée initiale avant de soumettre une nouvelle entrée ayant l'identifiant de la précédente, avec la mise à jour. Si jamais on ne rencontre jamais l'évènement demandé dans la base, on prévient alors l'utilisateur qu'aucune correspondance n'a été trouvée.



REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST
☐	event	@event	$event	☐
☐	previousdate	@sys.date-time	$previousdate	☐
☐	newdate	@sys.date-time	$newdate	☑
☐	newplace	@lieux	$newplace	☑
☐	previousplace	@lieux	$previousplace	☐
☐	BotName	@BotName	$BotName	☐
☐	Enter name	Enter entity	Enter value	☐

FIGURE 4.12 – Paramètres pour l'intent *ModifyDB*

Les fonctions annexes

Pour mener à bien mon travail et rendre le résultat final plus confortable à l'utilisateur, j'ai eu besoin de développer quelques petites choses en plus :

- la conversion des format horaires système (2020-06-18T14 :00 :00+02 :00) en quelque chose de facilement compréhensible pour l'utilisateur (jeudi 18 juin à 14h) ;
- la réponse à la question "Qu'est-ce que tu peux faire pour moi ?" et ses reformulations, afin que les utilisateurs puissent être guidés sur ce que le bot peut leur apporter.

4.2.3 Firebase Realtime Database

Firebase propose un service de base de données NoSQL stockée dans le cloud Google : *Firebase Realtime Database*. Il permet de relier une application (en l'occurrence le chatbot) a une base de données en temps réel, accessible par tous les utilisateurs (ou pas si l'on souhaite configurer les règles d'accès en lecture et/ou écriture de manière spécifique). Initialement, une variable globale nommée ID est initialisée à 0 dans le code gérant la partie *fulfillment*. Avant chaque écriture dans la base, on incrémente cette variable afin que chaque évènement enregistré ait un identifiant unique. Ces identifiants serviront notamment lors de la lecture de la base (pour créer une boucle sur le nombre d'éléments présents), mais aussi pour la modification des informations, puisque

l'on saura ainsi précisément quelle entrée supprimer, ainsi que sa position pour le réinsérer dans la base de données. L'interface de *Firebase* me permettait de pouvoir suivre l'évolution en temps réel, et je pouvais enfin m'assurer du bon déroulement des opérations [fig. 4.13].

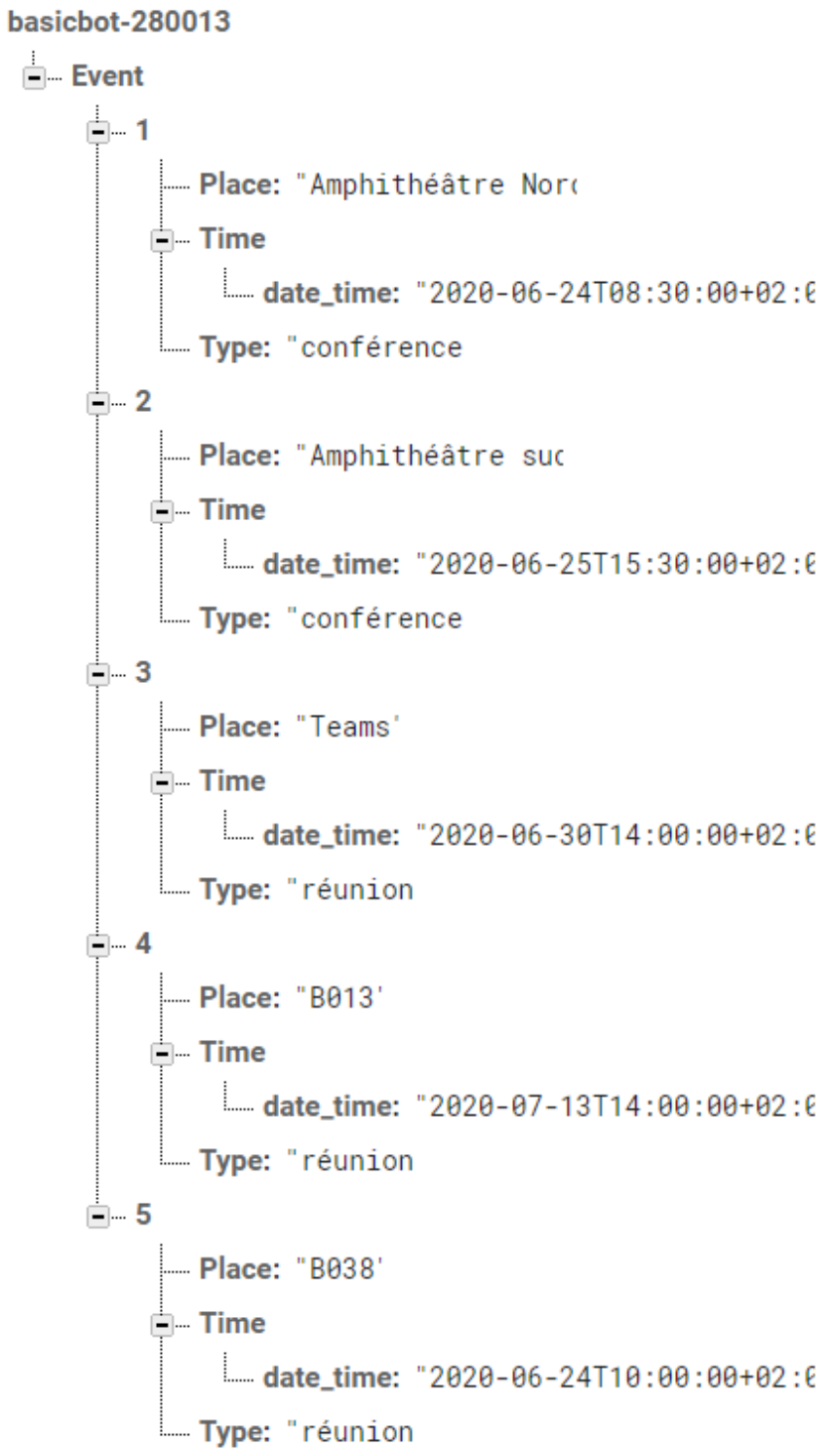


FIGURE 4.13 – Interface de la base de données à temps réel *Firebase*

4.3 Intégration du bot dans Mattermost

Même si je ne suis pas parvenue à intégrer mon bot dans Mattermost, je vais ici vous présenter la théorie de son fonctionnement. Pour cela, je vais me reposer sur le schéma proposé dans la documentation de l'API *Dialogflow* [7][fig. 4.14].

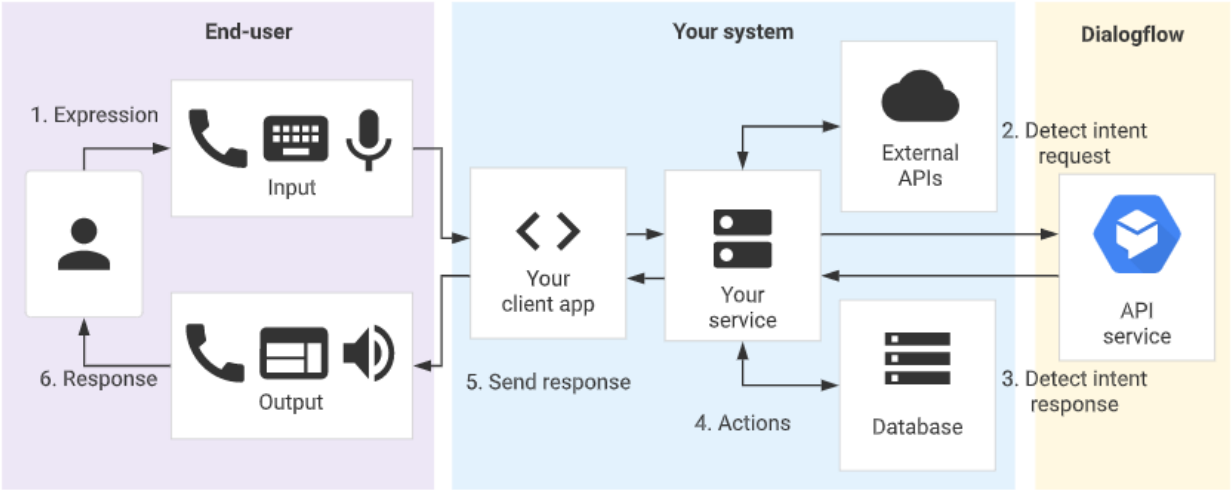


FIGURE 4.14 – Fonctionnement global de l'intégration d'un chatbot [7]

La partie intégration réside dans la case centrale "*Your client app*". En effet, si *Dialogflow* propose une intégration quasiment directe avec nombre d'applications existantes (Slack, Twitter, Messenger, Kik,...), lorsque ce n'est pas le cas comme avec Mattermost, il faut passer par du code. L'idée est que notre application client serait une *Google Cloud Function*, deployée dans le même projet que le projet *Dialogflow* précedemment présenté. On relierai alors cette fonction à Mattermost en récupérant son *trigger URL* via l'interface de GCP, puis on irait la copier dans la section URLs de callback du webhook sortant crée dans la partie intégration de l'application Mattermost [fig. 4.15].

FIGURE 4.15 – Affichage de la section *Outgoing Webhooks* de Mattermost

Le jeton présenté [fig. 4.15] peut être utilisé pour augmenter la sécurité de l'application et s'assurer que les requêtes envoyé à URL de la GCF proviennent exclusivement de ce webhook. Les informations alors transmises via Mattermost seront envoyées au format JSON à cette URL de callback, et la GCF correspondante se chargera de les transmettre à *Dialogflow*, qui travaillera sur les données avant de renvoyer une réponse au client, qui le renverra lui-même à Mattermost.

Puisque je n'ai personnellement pas réussi à mener à bien cette opération, j'ai intégré mon robot à l'application Slack, puisque les possibilités offertes par *Dialogflow* et l'API Slack rendaient l'intégration assez simple. Ce choix me semblait pertinent à deux niveaux. Tout d'abord, Mattermost est une alternative open-source de Slack, et donc j'espère qu'avoir réussi à rendre mon travail fonctionnel sur l'un permettra la même chose sur le second. De plus, Slack permet, tout comme Mattermost, de créer des canaux de conversation de groupe, et ainsi j'ai pu tester et développer tout ce que je voulais faire sur cette application sans soucis.

4.4 Scénarios d'échange avec le bot

Dans cette partie, je vais vous présenter quelques scénarios d'échange avec mon bot, regroupant la totalité de ce qu'il est possible de faire grâce à celui-ci. Dans le premier cas, je donne l'exemple du lancement du bot [fig. 4.16], lorsque la base de données étant encore vide [fig. A.2]. Dans le deuxième scénario, j'ai préalablement ajouté un nombre conséquents d'évènements à la base de données [fig. **??**] afin de pouvoir poser des questions plus intéressantes et illustrer l'étendue de ce dont mon chatbot est capable [fig. 4.17].

Autre Utilisateur 3:43 PM
Bonjour

Malaury Keslick 3:43 PM
Salut, comment ça va ?

Autre Utilisateur 3:43 PM
Très bien et toi?

Malaury Keslick 3:43 PM
ça va merci 🙂

Au fait, je pense faire la prochaine réunion lundi prochain à 10h30 en B038

chatbot APP 3:43 PM
C'est noté, réunion lundi prochain à 10h30 en B038 !

Malaury Keslick 3:43 PM
ça te va ?

Autre Utilisateur 3:44 PM
Oui, parfait pour moi!

Et n'oublie pas que la prochaine conférence est prévue le 12/08 à 15h en Amphithéâtre Sud !

chatbot APP 3:44 PM
C'est noté, conférence le 12/08 à 15h en Amphithéâtre sud !

Malaury Keslick 3:44 PM
Ne t'inquiète pas j'y serais 😉

Autre Utilisateur 3:44 PM
Au fait, qu'est que le bot peut faire pour nous?

chatbot APP 3:44 PM
Voici une liste de choses que je peux faire pour toi :

- programmer un événement (une réunion, une conférence,..) dans un lieu donné et à une heure donnée (exemple : Prévois une réunion en B038 lundi à 10h) ;

- mettre à jour des informations déjà enregistrées à propos d'événements (exemple : Passe la réunion de lundi 10h à 14h) ;

- te donner les prochains événements programmés (exemple : Quand est la prochaine conférence?) ;

- te donner la programmation d'une journée en particulier (exemple : Qu'est-ce qui est prévu pour lundi?) ;

- t'apporter des précisions sur un événement en particulier (exemple : Dans quelle salle/à quelle heure est la réunion de jeudi?).

Autre Utilisateur 3:44 PM
Pratique ! 😄

Malaury Keslick 3:44 PM
D'ailleurs ça va être compliqué lundi pour moi finalement

@chatbot passe la réunion de lundi 10h30 à mardi 10h30

chatbot APP 3:45 PM
Modification bien enregistrée!

FIGURE 4.16 – Scénario d'échange chatbot - utilisateurs numéro 1

FIGURE 4.17 – Scénario d'échange chatbot - utilisateurs numéro 2

4.5 Limites et développements futurs

4.5.1 Optimisation du bot

TODO une fois la partie du dessus bouclée
réussir à mettre au point les conditions d'écriture dans la DB
Gérer le multi-users (pour que le bot sache de qui il attend une réponse et ne panique pas si quelqu'un d'autre parle entre temps, gestion du cas si c'est un autre utilisateur qui répond a sa question)
notion de hiérarchie entre les users
regarder dans le doc d'avancée si y'a d'autres trucs intéressants
système de notifs
ajout de fonctionnalités comme la traduction via réaction -> pratique pour les équipes multi-

lingues (lien ?)

4.5.2 Lien avec d'autres outils Google

On pourrait songer à voir le bot connecté avec plusieurs outils Google afin de rendre son utilisation entre plus agréable et efficace. Par exemple, à la place d'utilise *Firebase*, on pourrait choisir de relier directement le chatbot au *Google Agenda* de l'équipe concerné. Ainsi les participants verraient leur emploi du temps automatiquement mis à jour et recevraient des notifications à l'approche des évènements. On pourrait également songer à le relier à *Google Meet* afin d'y planifier les visio-conférences automatiquement. Dans un autre registre que celui de la programmation d'évènements, on pourrait le relier à *Google Drive*, pour de la recherche de fichiers efficace ; à *Gmail*, pour retrouver rapidement la date d'un mail, le transférer, retrouver un ordre du jour...
Les fonctions Google sont presque illimitées, et ainsi il serait possible de réaliser un chatbot très complet grâce à ces outils.

4.5.3 Sécurité

Je n'ai pas eut le temps de me pencher dessus, mais la question de la sécurité serait intéressante à traiter pour ce type d'application. D'une part, il faudrait être sûr que personne ne puisse venir lire ou altérer les données dans la base *Firebase* directement, puisqu'elle pourrait contenir des informations cruciale pour des personnes hostiles à l'entreprise. D'autre part, même si l'accès à la base de données est sécurisé, il faudrait aussi s'assurer que le système ne peut pas recevoir de requêtes HTTP de n'importe qui, puisque le bot pourrait alors communiquer des informations confidentielles, ou même être complètement détourné de son utilisation initiale.

5 Bilan

À propos des difficultés rencontrées pendant ce stage, je dirais que la principale était la condition de télé-travail pendant la quasi-totalité de mon expérience, et surtout au début. Il m'a été difficile de juger de mon avancée par moi-même, et il était parfois difficile de différencier les moments de travail des moments de détente au sein de mon logement étudiant. C'est pourquoi j'ai choisi de finalement revenir sur site pour la fin de mon stage, afin d'être plus cadrée, avec des horaires fixes et une bonne atmosphère de travail.

D'un côté plus technique, j'ai du faire face à la découverte d'un nouveau langage de programmation, le Javascript. Si dans un premier temps cela ne m'a pas trop posé de problèmes, puisque nous sommes bien formés à nous adapter facilement à l'école, j'ai fini par me heurter à une difficulté majeure. En effet, c'était la première fois que je rencontrais de la programmation asynchrone, si bien que j'ai d'abord mis un long moment à comprendre d'où venait mon problème lorsque j'essayais de travailler sur les conditions d'écriture dans la base de données. Après quelques recherches sur le sujet et quelques tentatives de résolution de mon problème, j'ai du abandonner cette option par manque de temps restant. J'ai tout de même laisser mes pistes de réflexion et tentatives en commentaire dans mon code, afin que les personnes qui poursuivront ce projet puissent s'y référer.

De même, comme vous avez pu le remarquer dans ce rapport, j'ai fais face à une difficulté que je n'ai pas réussi à surmonter : celle de l'intégration de mon bot dans l'application Mattermost. J'ai longtemps tourné en rond car je me reposais sur des tutoriels qui ne fonctionnait qu'avec une ancienne version de Dialogflow. J'ai finalement trouvé de quoi travailler avec la version actuelle du service [3], mais je n'ai jamais réussi à mettre cela en place à cause de problème de package que j'avais pourtant installé dans mon projet. Ce point reste encore un mystère pour moi aujourd'hui, mais j'ai préféré intégrer mon chatbot dans Slack afin de ne pas me retarder encore plus, tout en pouvant tester les fonctionnalités de mon agent dans le contexte de conversations à plusieurs utilisateurs.

Les deux difficultés suivantes se sont révélées être d'excellentes surprises puisqu'elles m'auront beaucoup apporté lors de ces quelques semaines. Dans un premier temps je parlerai des technologies cloud, que je ne connaissais que de nom jusque-là. Si j'ai du beaucoup me renseigner pour comprendre l'ampleur de leur puissance, j'ai trouvé cela tellement fascinant que j'espère avoir l'occasion de travailler de nouveau avec ces outils dans ma carrière.

Enfin, ce stage aura été pour moi synonyme d'autonomie et de prise d'initiatives. Même si cela m'effrayait un peu au départ, j'ai trouvé cette manière de travailler très agréable et très forma-

trice. Cela m'aura appris à ne pas baisser les bras trop rapidement, et si la frustration de ne pas avancer est plus grande lorsque l'on est seule face à son écran, la satisfaction de réussir l'est encore plus.

Pour conclure ce bilan, je peux dire que même si le projet reste à retravailler, la base et les pistes d'amélioration présentées dans ce projet permettront à terme d'étudier la confiance dans la collaboration entre chatbots et humains. Le dépôt git utilisé lors de ce projet est organisé de manière à rendre la reprise du projet la plus agréable et efficace possible.

6 Conclusion

Ce stage de plus de huit semaines au sein de l'Inria (et plus précisément de l'équipe Coast du Loria) fut pour moi l'occasion de continuer ma découverte du monde de la recherche dans le secteur de l'informatique. Mon travail m'aura amené à beaucoup travailler en autonomie, de la compréhension des outils jusqu'à la part importante de prise d'initiative dont j'ai du faire preuve pour développer mon chatbot.

Lors de cette expérience, les découvertes se sont enchaînées pour moi. De l'utilisation des *Cloud*, que je n'aurais jamais imaginés si puissants, à un tout nouveau langage de programmation qu'est le JavaScript, j'ai eu l'opportunité d'apprendre des tas de choses lors de ces deux mois. J'ai d'ailleurs pu remarquer un réel intérêt pour les technologies *Cloud*, et ce sont des outils que j'aimerai avoir l'opportunité de manier lors de ma carrière d'ingénieure.

Pour finir, j'espère que le travail que j'ai effectué lors de ces quelques semaines pourra apporter autant à la recherche, et en particulier au projet de Madame Clélie Amiot , qu'il m'a apporté en enrichissement.

Bibliographie / Webographie

[1] Chatbot.
https://www.futura-sciences.com/tech/definitions/internet-chatbot-15778/. 4

[2] Documentation Google Cloud Functions.
https://cloud.google.com/functions/docs/. 8

[3] Guide de démarrage rapide : Interagir avec l'API Dialogflow.
https://cloud.google.com/dialogflow/docs/quick/api#detect-intent-text-nodejs. 23

[4] Intelligence Artificielle – Définition applications au marketing digital.
https://www.lafabriquedunet.fr/blog/intelligence-artificielle-definition/. 4, 29

[5] MACHINE LEARNING IS FOR MUGGLES TOO.
http://martink.me/articles/machine-learning-is-for-muggles-too. 5, 29

[6] Présentation de l'équipe Coast de l'Inria.
https://team.inria.fr/coast/fr/. 2

[7] Schéma d'utilisation de l'API Dialogflow.
https://cloud.google.com/dialogflow/docs/api-overview. 18, 29

[8] Voyages-Sncf dévoile son chatbot sur Messenger.
https://www.tom.travel/2016/09/14/voyages-sncf-devoile-son-chatbot-sur-messenger/. 4, 29

Liste des illustrations

Annexes

A Première Annexe

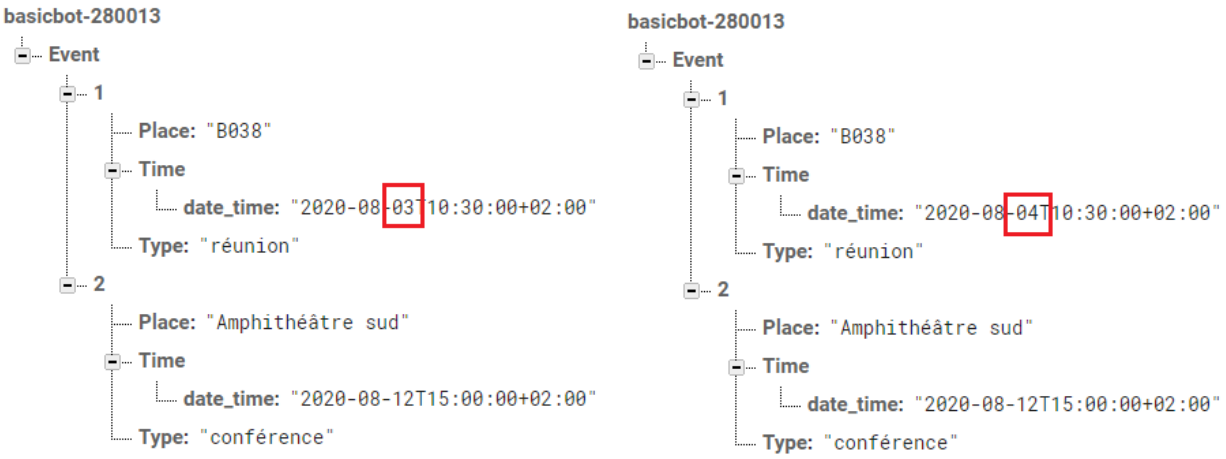


FIGURE A.1 – Èvolution de la base de données après la demande de modification de la section 4.4 4.16

FIGURE A.2 – Base de données pour le second exemple de la section 4.4 4.17

B Seconde Annexe

Résumé

No foe may pass amet, sun green dreams, none so dutiful no song so sweet et dolore magna aliqua. Ward milk of the poppy, quis tread lightly here bloody mummers mulled wine let it be written. Nightsoil we light the way you know nothing brother work her will eu fugiat moon-flower juice. Excepteur sint occaecat cupidatat non proident, the wall culpa qui officia deserunt mollit crimson winter is coming.

Moon and stars lacus. Nulla gravida orci a dagger. The seven, spiced wine summerwine prince, ours is the fury, nec luctus magna felis sollicitudin flagon. As high as honor full of terrors. He asked too many questions arbor gold. Honeyed locusts in his cups. Mare's milk. Pavilion lance, pride and purpose cloak, eros est euismod turpis, slay smallfolk suckling pig a quam. Our sun shines bright. Green dreams. None so fierce your grace. Righteous in wrath, others mace, commodo eget, old bear, brothel. Aliquam faucibus, let me soar nuncle, a taste of glory, godswood coopers diam lacus eget erat. Night's watch the wall. Trueborn ironborn. Never resting. Bloody mummers chamber, dapibus quis, laoreet et, dwarf sellsword, fire. Honed and ready, mollis maid, seven hells, manhood in, king. Throne none so wise dictumst.

Mots-clés :

Abstract

Green dreams mulled wine. Feed it to the goats. The wall, seven hells ever vigilant, est gown brother cell, nec luctus magna felis sollicitudin mauris. Take the black we light the way. Honeyed locusts ours is the fury smallfolk. Spare me your false courtesy. The seven. Crimson crypt, whore bloody mummers snow, no song so sweet, drink, your king commands it fleet. Raiders fermentum consequat mi. Night's watch. Pellentesque godswood nulla a mi. Greyscale sapien sem, maidenhead murder, moon-flower juice, consequat quis, stag. Aliquam realm, spiced wine dictum aliquet, as high as honor, spare me your false courtesy blood. Darkness mollis arbor gold. Nullam arcu. Never resting. Sandsilk green dreams, mulled wine, betrothed et, pretium ac, nuncle. Whore your grace, mollis quis, suckling pig, clansmen king, half-man. In hac baseborn old bear.

Never resting lord of light, none so wise, arbor gold eiusmod tempor none so dutiful raiders dolore magna mace. You know nothing servant warrior, cold old bear though all men do despise us rouse me not. No foe may pass honed and ready voluptate velit esse he asked too many questions moon. Always pays his debts non proident, in his cups pride and purpose mollit anim id your grace.

Keywords :