



Université de la Manouba
École Nationale des Sciences de l'Informatique



Rapport de stage de Fin d'études présenté en vue de l'obtention du titre d'ingénieur en informatique

Botman-webchat full responsive et IHM de suivi et de paramétrage de Botman

Réalisé par :

M^{elle} Nouha BEN GAID HASINE

Organisme d'accueil : Sofrecom



Supervisé par : Mme. Rim Drira

Encadré par : M. Ahmed Hedhili

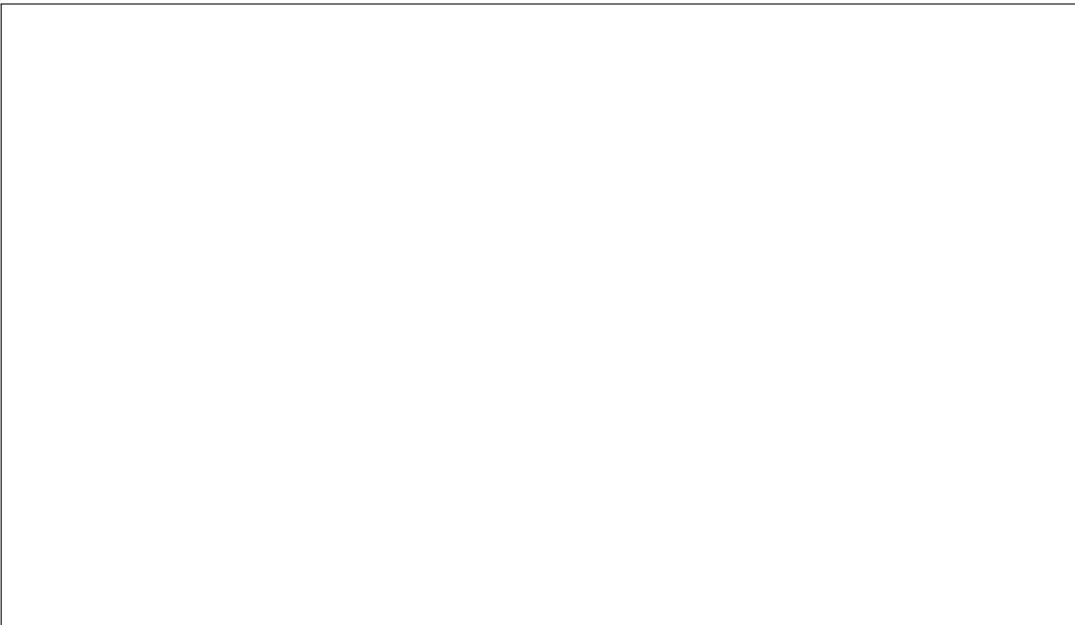
Adresse : Rue du Lac Constance 1053 les Berges du Lac Tunis,Tunisia

Tél : 31 302 749 Fax :71 963 284

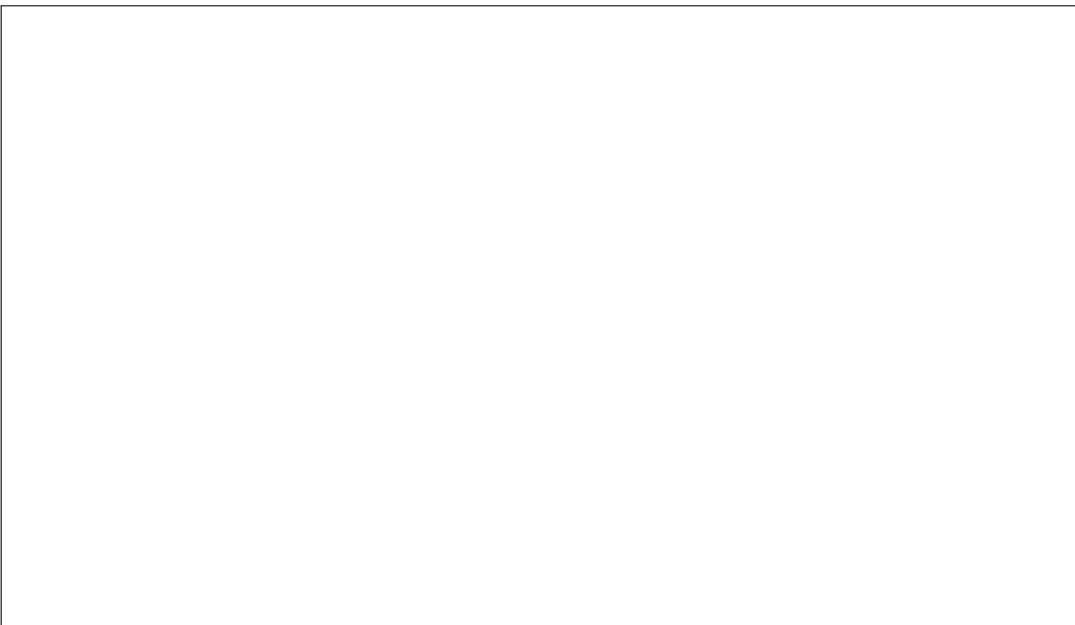
Email : ahmed.hedhili@sofrecom.com

Signatures et Appréciations

École Nationale des Sciences de l'informatique



SOFRECOM



Dédicaces

Je dédie ce travail,

A ma très chère mère Donia le pilier de mes efforts et la source de tendresse qui n'a jamais cessé de m'encourager. Tous les mots du monde ne sauraient exprimer l'immense amour et gratitude que je te porte. J'espère avoir répondu aux espoirs que tu as fondés en moi. Je te rends hommage par ce travail aussi modeste soit-il, en guise de ma reconnaissance éternelle et de mon infini amour. Que dieu te garde ma chère maman.

À la mémoire de mon cher père qui nous a quittés très tôt, qui a été ma source d'inspiration, qui a été toujours dans mon esprit et dans mon cœur. Je ferai toujours de mon mieux pour rester ta fierté. Que dieu, le miséricordieux t'accueille dans son éternel paradis.

À ma chère sœur et mon petit frère, qui ont toujours été à mes côtés, je vous souhaite un avenir plein de bonheur et de réussite.

À mon fiancé mehdi, qui était toujours présent pour m'encourager et qui m'a soutenu tout au long de mon parcours universitaire.

À tous les membres de ma chère famille Ben Gaid Hassine et Ouedfen qui n'ont cessé d'être pour moi des exemples de persévérance, de courage et de générosité.

À mes chers amis, mon autre famille, votre amitié est un honneur et une fierté pour moi. Je vous remercie d'avoir embellie ma vie par des précieux moments de Bonheur.

Remerciements

Avant de présenter mon travail, je réserve ces quelques lignes pour exprimer mes sincères remerciements et ma gratitude aux personnes qui m'ont apporté leur support et leur aide durant mon stage et qui ont contribué au succès de ce travail.

Tout d'abord, je tiens à adresser ma reconnaissance et mes respects les plus distingués à mon encadrant à la société SOFRECOM, Monsieur Ahmed Hedhili, pour son aide, ses conseils, sa disponibilité tout au long de ce stage.

Je tiens aussi à présenter mes sincères remerciements à Madame Rim Drira, mon encadrante à l'École Nationale des Sciences de l'Informatique, pour sa disponibilité et ses remarques constructives et précieuses qui ont donné une valeur ajoutée à mon travail.

Je remercie également le cadre administratif de SOFRECOM et l'équipe Botman pour leur accueil chaleureux et leur support.

J'exprime ma profonde gratitude et ma reconnaissance au corps professoral de l'École Nationale des Sciences de l'Informatique pour la qualité de formation qu'ils nous ont offerte. En effet, nous avons entamé notre première expérience professionnelle avec des bases solides grâce à leurs contributions et leurs conseils.

Je désire aussi remercier les membres du jury qui m'ont fait l'honneur d'examiner et d'évaluer mon modeste travail tout en espérant qu'ils trouveront dans ce rapport la clarté qu'ils espèrent.

Table des matières

Glossaire des acronymes

Introduction Générale	1
1 Cadre Général du projet	3
1.1 Contexte du projet	3
1.2 Organisme d'accueil	3
1.2.1 Groupe Sofrecom	3
1.2.2 Sofrecom Tunisie	4
1.2.3 Domaine d'activité	4
1.3 Étude de l'existant	5
1.3.1 Solution actuelle à Sofrecom et critiques	5
1.3.2 Outils de suivi et de paramétrage des chatbots existantes	7
1.3.3 Solution proposée	9
1.4 Méthodologie adoptée	10
1.4.1 Méthodes agiles	10
1.4.2 Scrum	10
1.4.2.1 Scrum dans l'équipe botman	11
1.5 Outils adoptés pour la gestion de projet	11
2 Sprint 0 : Analyse des besoins	13
2.1 Expression des besoins	13

2.1.1	Identification des acteurs du système	13
2.1.2	Identification des besoins	13
2.1.2.1	Besoins fonctionnels	13
2.1.2.2	Besoins non fonctionnels	14
2.2	Spécifications des besoins	14
2.2.1	Diagrammes de cas d'utilisation	14
2.3	Architecture de l'application	15
2.3.1	Architecture physique	15
2.3.2	Architecture logique	16
2.4	Backlog Produit	18
2.5	Planification des sprints	21
2.6	Environnement de travail	22
2.6.1	Environnement matériel	22
2.6.2	Environnement logiciel	23
2.6.3	Framework et Outils	23
2.6.4	Langages de programmation	24
3	Sprint 1 : Devops-Création d'une chaîne CI/CD	25
3.1	Sprint Backlog	25
3.2	Démarche DevOps	26
3.2.1	Définition	26
3.2.2	Intégration continue	27
3.2.3	Déploiement continu	27
3.3	Éléments clés	28
3.3.1	Outil de gestion de code source : GitLab	28
3.3.2	Outils d'intégration continue : Gitlab CI	28
3.3.3	Outils de qualimétrie logiciels : SonarQube	29

3.3.4	Outils de gestion de configuration et automatisation : Ansible	29
3.3.5	Outil de conteneurisation : Docker	30
3.3.6	Outil d'orchestration : DockerSwarm	30
3.3.7	Cycle de vie d'un User Story dans le projet Botman	31
3.4	Réalisation	31
3.4.1	Configuration et Automatisation de déploiement avec Ansible	31
3.4.2	Botman	33
3.4.2.1	docker-compose.yml	33
3.4.2.2	gitlab-ci.yml	33
3.4.2.3	Dockerfile	34
3.4.2.4	Pipeline CI/CD du projet Botman	34
3.4.3	IHM de suivi et de paramétrage de Botman	34
3.4.3.1	docker-compose.yml	34
3.4.3.2	gitlab-ci.yml	35
3.4.3.3	Dockerfile	35
3.4.3.4	Pipeline CI/CD de l'IHM de suivi et de paramétrage de Botman	36
4	Sprint 2 : Réingénierie du projet botman	37
4.1	Sprint Backlog	37
4.2	Fonctionnement Global du projet Botman	38
4.2.1	Architecture logique du projet Botman	38
4.2.2	Provider JMS	40
4.2.3	RabbitMQ	40
4.3	Réingénierie d'un module backend du projet "Botman"	42
4.3.1	Web service REST	42
4.3.2	Travail élaboré	43
4.4	Réingénierie partie FrontEnd de l'orchestrateur Botman	43

4.4.1	RWD (Responsive Web Design)	44
4.4.2	Les Medias Queries	45
4.4.3	Travail élaboré	45
4.5	Réalisation	46
5	Sprint 3 : Application de suivi et de paramétrage de Botman : Authentification et gestion	49
5.1	sprint Backlog	49
5.2	Spécification	50
5.2.1	Diagramme de cas d'utilisation raffiné du cas «Gérer les utilisateurs»	51
5.2.2	Diagramme de cas d'utilisation raffiné du cas « Gérer les bots »	52
5.3	Conception	53
5.3.1	Diagramme de séquence objet du cas «Ajouter un Bot»	53
5.3.2	Diagramme d'activité du cas «Ajouter un Bot»	54
5.3.3	Diagramme de classes de conception	55
5.4	Réalisation	56
5.5	Test et revue du sprint	60
6	Sprint 4 : Application de suivi et de paramétrage de Botman : Performance des Bots	61
6.1	Sprint Backlog	61
6.2	Spécification	62
6.2.1	Diagramme de cas d'utilisation raffiné	63
6.3	Conception	64
6.3.1	Diagramme de séquence objet du cas «Filtrer les conversations selon plusieurs critères»	64
6.3.2	Diagramme d'activité du cas «Filtrer les conversations selon plusieurs critères	66
6.3.3	Diagramme de classes de conception	66
6.4	Réalisation	68

TABLE DES MATIÈRES

6.5 Test et revue du sprint	71
Conclusion Générale	72
Netographie	74

Table des figures

1.1	Répartition des filiales et clients de Sofrecom[2]	4
1.2	L'ancienne interface de l'application Botman	5
1.3	Interface d'accueil de BotAnalytics [5]	8
1.4	Interface d'accueil de Dashbot [5]	8
1.5	Interface d'accueil de Botmetrics [5]	9
1.6	Mes tâches dans Jira	12
2.1	Diagramme de cas d'utilisation global	15
2.2	Architecture physique de notre système [10]	16
2.3	Architecture logique de notre système [11]	17
2.4	Planification des sprints	22
3.1	DevOps [21]	27
3.2	Utilisation de GitLab	28
3.3	Architecture Ansible [32]	29
3.4	Cycle de vie d'un US	31
3.5	Extrait du fichier Inventory	32
3.6	Extrait du fichier Playbook	32
3.7	extrait du fichier docker-compose.yml du projet Botman	33
3.8	extrait du fichier gitlab-ci.yml du projet Botman	33
3.9	fichier Dockerfile du projet Botman	34

3.10 pipeline CI/CD du projet Botman	34
3.11 extrait du fichier docker-compose.yml de l'IHM de suivi et de paramétrage de Botman	35
3.12 extrait du fichier gitlab-ci.yml de l'IHM de suivi et de paramétrage de Botman	35
3.13 fichier Dockerfile frontend de l'IHM de suivi et de paramétrage de Botman	35
3.14 fichier Dockerfile backend de l'IHM de suivi et de paramétrage de Botman	36
3.15 pipeline CI/CD de l'IHM de suivi et de paramétrage de Botman	36
4.1 Fonctionnement Global de Botman	38
4.2 l'architecture logique du projet Botman	39
4.3 Provider JMS [35]	40
4.4 Communication entre les différents modules de Botman via RabbitMq	41
4.5 Architecture Service Web Rest [39]	42
4.6 Les entêtes des méthodes receive et send	43
4.7 Utilisation mobiles VS ordinateurs [40]	44
4.8 Responsive Web Design [42]	44
4.9 Media Queries de notre projet	45
4.10 Interface de taille 1280*1024	46
4.11 Interface de taille 1024*1024	46
4.12 Interface de taille 768*1024	47
4.13 Interface de taille 360*480	47
4.14 Interface de taille 320*568	47
5.1 Diagramme de cas d'utilisation raffiné du cas « Gérer les utilisateurs »	51
5.2 Diagramme de cas d'utilisation raffiné du cas « Gérer les bots »	52
5.3 Diagramme de séquence objet du cas « Ajouter un Bot»	54
5.4 Diagramme d'activité du cas «Ajouter un Bot»	55
5.5 Diagramme de classes de conception du sprint 3	56
5.6 Interface d'authentification de l'application	57

5.7	Interface d'accueil de l'administrateur	57
5.8	Interface gestion des utilisateurs	58
5.9	Interface modifier/supprimer un utilisateur	58
5.10	Interface gestion des bots	59
5.11	Interface d'ajout d'un bot	59
6.1	Diagramme de cas d'utilisation raffiné du sprint 4	63
6.2	Diagramme de séquence objet du cas «Filtrer les conversations selon plusieurs critères»	65
6.3	Diagramme d'activité du cas «Filtrer les conversations selon plusieurs critères	66
6.4	Diagramme de classes de conception du sprint 2	67
6.5	Interface de filtrages des conversations	68
6.6	Interface résultat de recherche	69
6.7	Interface d'affichage des détails d'une conversation sélectionné	69
6.8	Interface d'ajout des tags	70
6.9	Interface de suivi des KPI	71

Liste des tableaux

1.1	Tableau comparatif de JQuery et Angular	6
1.2	Tableau comparatif de WebSocket et REST [4]	7
1.3	Tableau comparatif des différentes solutions	9
1.4	L'équipe SCRUM	11
2.1	Backlog Produit	20
3.1	Sprint Backlog	26
4.1	Sprint Backlog	37
5.1	Sprint Backlog	50
5.2	Description textuelle du cas «Ajouter un utilisateur»	52
5.3	Description textuelle du cas « Modifier un bot »	53
6.1	Backlog Produit	61
6.2	Backlog Produit	62
6.3	Description textuelle du cas « Filtrer les conversations selon plusieurs critères»	64

Glossaire des acronymes

CI : Continuous Integration

CD : Continuous deployment

MOA : Maître d'ouvrage

MOE : Maître d'œuvre

IHM : Interface Homme-Machine.

JSON : JavaScript Object Notation.

HTTP : Hypertext Transfer Protocol.

API : Application Programming Interface.

REST : Representational State Transfer.

NoSQL : Not Only Structured Query Language.

NLP : Natural Langage Processing

RWD : Responsive Web design

US : User Story

XML : Extensible Markup Language

URL : Uniform Resource Locator

JMS :Java Message Service

KPI :key performance indicators

SCM : Software Configuration Management

Introduction générale

À cours des dix dernières années, de nouvelles tendances ont émergé et la concurrence s'est accrue de manière exponentielle entre les entreprises, en particulier avec le marché en ligne et les réseaux sociaux.

Face à ces défis, les entreprises doivent toujours traiter leurs clients comme un atout essentiel et une valeur centrale pour réussir. Par conséquent, ils doivent offrir un support clientèle de qualité et accessible à tout moment en fidélisant leurs clients et en encourageant d'autres. Pour remplir cet objectif, la communication est fondamentale dans cette perspective.

Dans ce domaine, les plates-formes de messagerie, notamment celles basées sur les robots, sont principalement utilisées par les clients pour leurs interactions avec les entreprises. En effet, 69 % des clients utilisent les robots pour leur rapidité par rapport à l'échange humain selon des statistiques faites par Ubisend – Chatbot Survey en 2017.

Ainsi, un chatbot est un système de dialogue humain. Il est l'outil le plus utilisé et le plus adapté à la communication avec les clients. Ce concept remonte à Alain Turing, son premier inventeur en 1950 [1]. Depuis lors, cette technologie a connu de grands progrès en matière de compréhension, de traitement et d'apprentissage automatique du langage naturel.

De plus, la discussion avec un agent virtuel nécessite des fois de basculer activement entre les différents chatbots pour avoir la réponse pertinente au client, d'où la nécessité de mettre en œuvre des orchestrateurs de chatbots assurant cette fonctionnalité.

Par conséquent, cette technologie crée une grande rivalité entre les grandes et les petites entreprises afin de créer la meilleure plate-forme en langage naturel du marché et offrir ainsi, de meilleures capacités de développement pour réaliser la solution la plus complète en termes de capacités de compréhension et de génération de réponses. Donc, les entreprises doivent mettre en place un mécanisme permettant de surveiller et mesurer les performances du chatbot par rapport à des objectifs prédéfinis. Comme toute technologie, elle fait l'objet d'évaluations périodiques. Après la formation et la mise en œuvre, il faut définir les indicateurs de performance clés (**Key Performance Indicator**) ou les indicateurs nécessaires pour mesurer les performances de chatbot.

C'est dans ce cadre que s'inscrit notre projet de fin d'études à l'École Nationale des Sciences de l'Informatique réalisé à Sofrecom et qui est divisé en trois parties principales : la première consiste à la réingénierie de la partie frontend du projet botman qui représente un orchestrateur de chatbots d'Orange France. Cette partie comporte la mise en place d'une fenêtre de webchat en pleine page et responsive développée en Angular et Typescript, ainsi que la réingénierie d'un module backend de ce projet. La seconde partie consiste à créer une application de suivi et de paramétrage du projet Botman. La troisième partie consiste à créer des chaînes **Continuous Integration/Continuous deployment** pour intégrer les deux développements.

Le présent rapport présente une synthèse des étapes d'élaboration du projet. Le rapport réalisé est organisé comme ceci :

Le premier chapitre sera dédié au cadre général du projet. Nous présenterons au cours de ce chapitre l'organisme d'accueil, le contexte d'élaboration du projet. Nous aborderons aussi dans ce chapitre l'étude et la critique de l'existant. Le deuxième chapitre représentera le sprint 0 qui comportera l'analyse des besoins. Nous commencerons par l'expression et la spécification des besoins de notre projet puis nous définirons l'architecture de l'application. Ensuite nous entamerons la partie planifications des sprints. Nous clôturons ce chapitre par la présentation de l'environnement de travail. Les troisièmes, quatrièmes, cinquièmes et sixièmes chapitres seront dédiés à la réalisation des différents sprints de notre projet. Ce rapport sera clôturé par une conclusion générale qui comportera une évaluation des résultats que nous avons atteints ainsi que les perspectives envisagées pour notre projet.

Chapitre 1

Cadre Général du projet

Ce premier chapitre est constitué de deux grandes parties. Une première partie est dédiée à la présentation de l'organisme d'accueil et à exposer le contexte d'élaboration du projet. Une deuxième partie sera consacrée à l'étude de l'existant qui présente une phase primordiale pour comprendre le système existant dans l'entreprise et bien définir ses objectifs.

1.1 | Contexte du projet

Ce projet a été élaboré dans le cadre d'un stage de fin d'études. Il s'agit d'une étape obligatoire dans le programme officiel de l'École Nationale des Sciences de l'Informatique afin d'obtenir le diplôme d'ingénieur.

Ce projet a été réalisé sur quatre mois et ce du 04 février au 15 juin 2019. Au cours de cette période nous avons intégré Sofrecom Tunisie en tant que stagiaire travaillant sur une mission de développement au sein de l'équipe Botman.

1.2 | Organisme d'accueil

1.2.1 Groupe Sofrecom

Développé depuis plus de 50 ans, Sofrecom[2] est une filiale d'orange et l'un des leaders mondiaux du conseil et de l'ingénierie télécom grâce à un savoir-faire unique dans les métiers du numérique. Sofrecom est présentes dans 11 bureaux sur 4 continents. Pendant ces dernières années, plus de 200 clients dans plus de 100 pays ont confiés leurs projets à Sofrecom.

La carte présentée par la figure 1.1 illustre la répartition des filiales et clients de Sofrecom dans le monde.

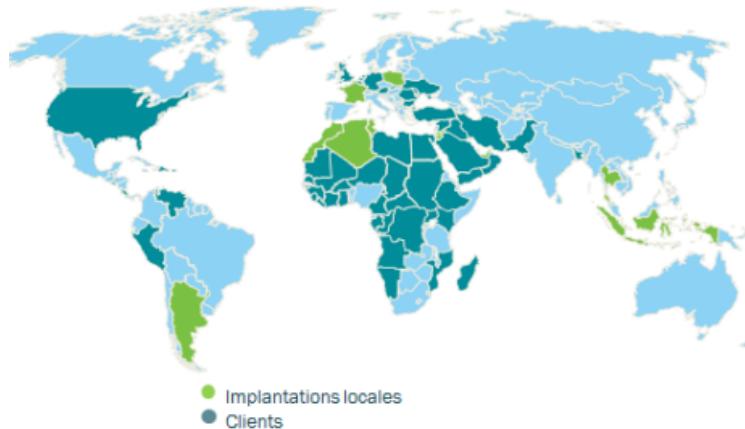


FIGURE 1.1 – Répartition des filiales et clients de Sofrecom[2]

1.2.2 Sofrecom Tunisie

Inaugurée en 2012, Sofrecom Tunisie [2] représente la plus jeune filiale de ce groupe en Moyen-Orient et zone Afrique.

Aujourd’hui, Sofrecom Tunisie a acquis de la crédibilité et de la force grâce à ces collaborateurs qui travaillent sur des centaines de projets variés à l’échelle internationale et qui ont atteint un nombre qui dépasse les 550.

Elle représente un acteur majeur d’expertise et de développement des plates-formes de services et des systèmes d’information.

1.2.3 Domaine d’activité

L’activité de Sofrecom a pu confirmer son expertise dans trois domaines d’activité [3] :

- **Développement IT** : Ingénierie, Validation et intégration, Gestion de projets et de plates-formes, Architecture, Design de logiciels et services...;
- **Conseil Business** : Transformation digitale, Services financiers mobiles, Capacity building and change Management ;
- **Conseil en IT et Networks** : Conseil EGov et smartGov, sécurité, Design de Réseaux, Transformation de Réseaux, Optimisation Coûts et Investissement Réseaux, E-Santé ...

1.3 | Étude de l'existant

1.3.1 Solution actuelle à Sofrecom et critiques

Botman est la solution actuelle à Sofrecom. Il s'agit d'un orchestrateur de chatbots qui représente un méta-bot permettant de basculer facilement entre plusieurs bots au sein d'une seule interface. En effet, Botman est une solution qui permet de centraliser et d'orchestrer les agents conversationnels multicanaux alimentés avec différents NLU (RASA, IBM Watson, etc...).

Il offre aux clients d'Orange ainsi qu'à ses employés la possibilité d'avoir des informations ou des instructions concernant Orange via des bots.

Actuellement, l'application Botman met à disposition une fenêtre de webchat qui est seulement utilisable sur des devices desktop dans le coin inférieur droit de la page grâce à l'insertion d'un script Js qui déploie cette fenêtre réalisée en Javascript et JQuery.

La figure 1.2 illustre l'ancienne interface de l'application Botman.

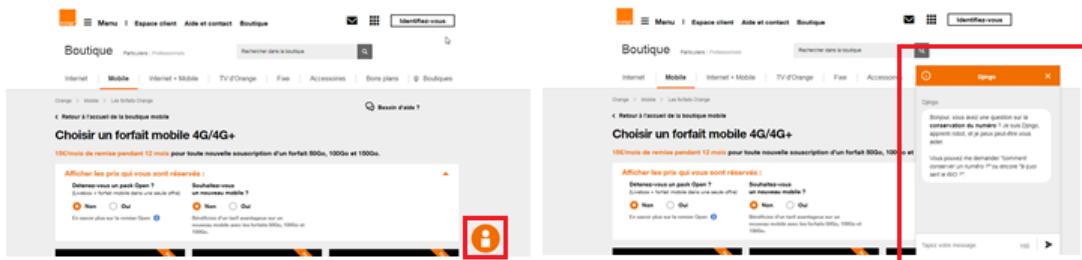


FIGURE 1.2 – L'ancienne interface de l'application Botman

Cependant, le client présente des nouveaux besoins côté frontend auxquels Botman doit répondre :

- Réingénierie du code (passage de javascript et JQuery à Angular et TypeScript)
- L'affichage de ce webchat sur tous type d'écran, en full screen, avec la prise en compte des spécificités UX et UI de ces devices. (webchat responsive)

En effet, une migration du code de la partie Frontend vers Angular et typeScript s'avère intéressante dans le cas de ce projet.

Le tableau 1.1 illustre les avantages que présente le développement de la partie frontend avec Angular par rapport à celle avec JQuery.

	JQuery	Angular
Manipulation du DOM	✓	✓
Restful API	X	✓
Supporte l'animation	✓	✓
Deep linking routing	X	✓
Validation de la form	X	✓
2 way data-binding	X	✓
AJAX/JSONP	✓	✓

TABLE 1.1 – Tableau comparatif de JQuery et Angular

Également, le module webchat de la partie backend du projet botman qui se base sur le Websocket présente des problèmes :

- Mise en production avec coupure (car le Websocket est Statefull) ;
 - Ne permet pas d'établir des statistiques sur les Bots et de mesurer leurs KPI's.
- Ainsi, un remplacement du Websocket dans le module webchat par un protocole plus efficace s'avère très important.

Le tableau 1.2 représente une comparaison entre le WebSocket et le REST.

	WebSocket	REST
Protocole	FTP	HTTP
Communication	Bidirectionnelle	Unidirectionnelle
La nature	Basé sur les sockets	basé sur les ressources
Dépendance	Adresse IP + Port	utilise les méthodes HTTP
State	Statefull	Stateless
Scalabilité	verticale (ajouter des ressources à un unique serveur)	Horizontale(ajouter de nouveaux serveurs)

TABLE 1.2 – Tableau comparatif de WebSocket et REST [4]

Puisque le REST est stateless, il permet de remédier aux problèmes de la mise en production avec coupure et l'établissement des statistiques. En effet, nous devons gérer le comportement du REST pour l'adapter exactement aux besoins du projet. Nous détaillons cette partie dans le deuxième sprint.

Par ailleurs, le projet Botman n'a aucun outil de suivi et de paramétrage. Dans la section suivante, nous allons présenter quelques outils existants sur le marché permettant le suivi et le paramétrage des chatbots.

1.3.2 Outils de suivi et de paramétrage des chatbots existantes

Botanalytics : c'est une solution [5] qui permet de consulter les dernières conversations, leur durée et le nombre de sessions avec le bot. Cette solution permet aussi de donner des statistiques concernant les utilisateurs les plus engagés et les plus actifs.

Elle est compatible avec plusieurs chatbots tels que : Messenger, Skype, Viber, Wechat, Kik ... La figure 1.3 illustre l'interface d'accueil de BotAnalytics.

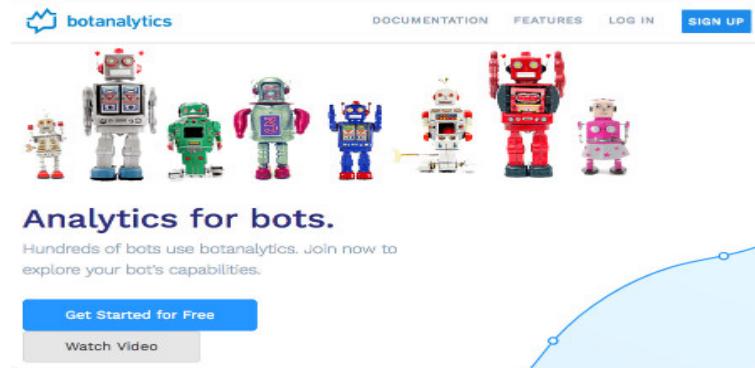


FIGURE 1.3 – Interface d'accueil de BotAnalytics [5]

Dashbot : c'est un outil [5] permettant de faire des analyses approfondies concernant les conversations par utilisateur, et qui grâce à certains indicateurs estime l'engagement du public vis à vis du bot.

Cette solution propose aussi un aperçu sur la conversation typique entre l'utilisateur et le bot afin de détecter les erreurs commises et pouvoir les corriger.

Cet outil est compatible avec Slack, Messenger, Amazon Alexa, Google Assistant.

la figure 1.4 illustre l'interface d'accueil de DashBot.

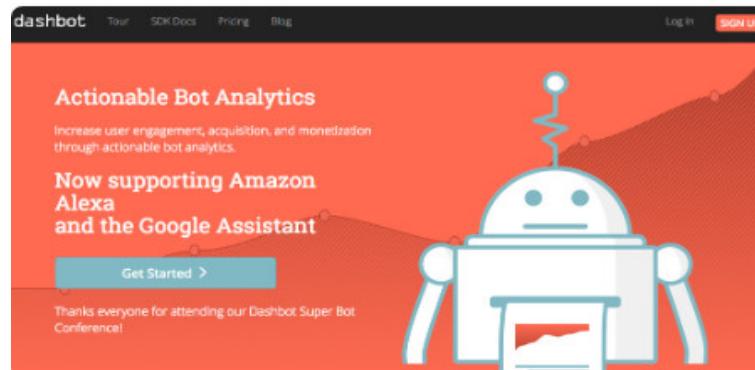


FIGURE 1.4 – Interface d'accueil de Dashbot [5]

Botmetrics : c'est un outil [5] modulable permettant des analyses sur les chatbots. Cette solution permet de personnaliser le tableau de bord en se focalisant sur les informations qui vous intéressent et avoir une vue globale sur les indicateurs clés du bot.

Botmetrics est compatible avec Google Assistant, Messenger, Kik, Slack, Microsoft Bot.

La figure 1.5 illustre l'interface d'accueil de Botmetrics.

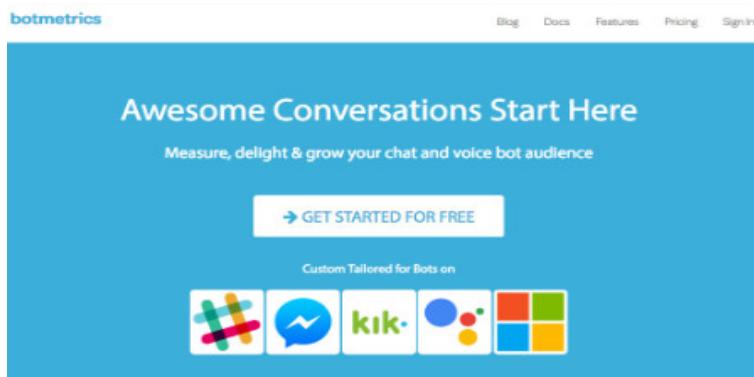


FIGURE 1.5 – Interface d'accueil de Botmetrics [5]

1.3.3 Solution proposée

Lors de notre étude, nous avons identifié des défaillances de la solution Botman. Par conséquent, et afin de remédier ce problème, nous proposons une solution qui consiste à mettre en œuvre une fenêtre de webchat en Angular. Cette fenêtre est utilisable sur tous type d'écran en pleine page avec la prise en compte des spécificités UX et UI des devices.

Aussi, nous allons résoudre le problème du module webchat de la partie backend du projet Botman en passant du websocket au style d'architecture REST.

Nous avons, aussi, identifier des limites pour les solutions de suivi et de paramétrage des chatbots existantes sur le marché.

Le tableau 1.2 présente une comparaison entre les différentes solutions citées :

	Suivi des conversations	Annoter les conversations	Suivi des KPI de lecture	Paramétrage des bots	Filtrer les conversations
BotAnalytics	X				
Dashbot	X				
Botmetrics	X		X		

TABLE 1.3 – Tableau comparatif des différentes solutions

Il est évident que les solutions citées présentent plusieurs fonctionnalités. Cependant, elles ne répondent pas à toutes les attentes de la sociétés Orange. Aucune d'entre elles ne présente une

combinaison de toutes les fonctionnalités d'où la nécessité de la mise en place d'une application dédiée à Orange qui répond à tous ses besoins. C'est dans cet objectif que nous avons abordé ce sujet afin de mettre en oeuvre une application qui englobe toutes les fonctionnalités non disponibles dans les solutions existantes sur le marché.

De plus, nous allons créer des chaîne CI/CD afin d'y intégrer les deux développement.

1.4 | Méthodologie adoptée

Dans cette section, nous présenterons la méthodologie du travail adoptée. Nous introduirons d'abord les méthodes agiles et scrum, puis nous justifions la méthodologie et les outils choisis qui ont contribué à l'avancement du projet.

1.4.1 Méthodes agiles

Les méthodes agiles peuvent être définies comme de nouvelles approches basées sur un développement itératif adapté et souple, où les exigences et les solutions évoluent tout au long du projet selon[6]. Les méthodes agiles utilisent les meilleures pratiques d'ingénierie permettant la livraison rapide d'un produit de haute qualité et de réduire le taux d'erreur puisque les besoins du client représentent le centre des priorités du projet.

Selon [7], les valeurs des méthodes agiles sont :

- Accepter les changements ;
- Collaborer avec le client ;
- Basées sur des fonctionnalités opérationnelles ;
- Fondées sur les interactions des individus.

1.4.2 Scrum

La méthode Scrum est une sous-catégorie des méthodes agiles. C'est aujourd'hui l'approche la plus utilisée pour le développement de logiciels d'après le rapport annuel réalisé par "VersionOne" qui affirme que 56 % des équipes agiles utilise Scrum. En effet, il est utilisé pour développer des produits logiciels complexes basé sur des processus itératifs et incrémentaux [8].

Par conséquent, nous avons opté pour l'approche Scrum parmi d'autres méthodes, car cela nous aidera à livrer rapidement un produit de haute qualité et plus flexible aux environnements changeants.

Une méthode scrum regroupe généralement trois acteurs principaux :

- Product Owner : Il représente le client ;
- Scrum Master : Personne (s) responsable (s) de la direction et de l'équipe de direction ;
- Team : Composé de développeurs, ingénieurs, concepteurs, etc ;

1.4.2.1 Scrum dans l'équipe botman

Le tableau 1.4 illustre L'équipe SCRUM dans le projet Botman.

Rôle	Acteur	Mission
Product Owner	Christophe Gautherat	client
Scrum master	Khouloud Zaiem	responsable technique
Team	Nouha Ben Gaid Hassine, Ahmed Hedhili, Takoua Louhichi, Samira Hellab	développement

TABLE 1.4 – L'équipe SCRUM

1.5 | Outils adoptés pour la gestion de projet

Pour le bon déroulement du projet et afin de réussir à livrer notre produit, nous avons adopté certains outils de gestion de projet tels que :

- **Git** : un système de contrôle de version permettant de suivre les modifications apportées aux fichiers de codes et de coordonner le travail sur ces fichiers entre plusieurs personnes [9].
- **Gitlab** : outil de gestion de dépôts utilisant Git.

- **lynx** : Plateforme collaborative de communication entre équipes.
- **Jira** : Outil open source de gestion de projet.

La figure 1.6 représente mes tâches sur jira.

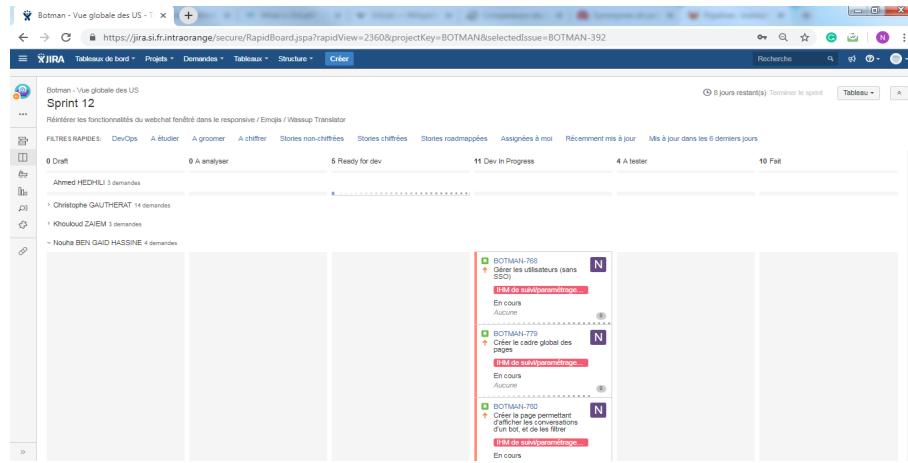


FIGURE 1.6 – Mes tâches dans Jira

Conclusion

Au cours de ce premier chapitre, nous avons présenté le cadre général du projet. Nous avons commencé par exposer le cadre de stage en premier lieu.

En second lieu, nous avons présenté l'organisme d'accueil. La dernière partie a été consacrée à l'étude de l'existant. Dans le chapitre suivant, nous aborderons la partie analyse des besoins.

Chapitre 2

Sprint 0 : Analyse des besoins

Ce premier sprint est dédié principalement à l'analyse des besoins de notre projet. Le sprint 0 était une occasion pour s'intégrer dans l'équipe Botman. Les réunions organisées durant ce sprint nous ont permis d'avoir une idée claire sur le projet existant et les anomalies qu'il comporte ainsi que de capturer les besoins fonctionnels et les besoins non fonctionnels. Puis, nous avons élaboré l'architecture de notre solution. Nous avons aussi recensé les cas d'utilisation afin d'élaborer le diagramme de cas d'utilisation global. Nous clôturons ce chapitre avec le Product Backlog pour permettre la planification des sprints à venir.

2.1 | Expression des besoins

2.1.1 Identification des acteurs du système

Dans le cas de ce projet, l'application s'adresse à deux types d'acteurs à savoir :

- Administrateur : représente le MOA et le MOE du projet Botman.
- Utilisateur : représente les membres des équipes projet et des équipes de support technique.

2.1.2 Identification des besoins

2.1.2.1 Besoins fonctionnels

Notre projet consiste à :

- Développer la partie frontend responsive (utilisable sur tous types d'écrans) et en pleine page avec Angular et typeScript. La réingénierie du module webchat de la partie backend du projet botman en remplaçant l'utilisation du protocole Websocket par le REST.
- Développer une application de suivi et de paramétrage de Botman qui présente les besoins suivant :

- Authentification
 - Gestion des utilisateurs
 - Gestion des bots
 - filtrage des conversations selon plusieurs critères
 - Ajout des tags aux conversations
 - Suivi des KPI des bots
- Créer des chaîne CI/CD pour y intégrer les deux développement.

2.1.2.2 Besoins non fonctionnels

En plus des besoins fonctionnels, notre système doit respecter quelques contraintes et critères à savoir :

- **La sécurité** : L'accès à l'application doit être sécurisé car les données manipulées par cette dernière sont confidentielles ;
- **Ergonomie** : Les interfaces graphiques de la solution doivent être responsives, compréhensibles, faciles à utiliser et conformes à la charte graphique d'Orange ;
- **La maintenance** : Afin de se conformer aux nouveaux besoins le système doit être souple et évolutif.

2.2 | Spécifications des besoins

Dans cette partie, nous exprimerons les besoins fonctionnels de l'application de suivi et de paramétrage de Botman que nous avons exposé dans la section précédante sous forme de diagrammes UML pour mieux comprendre et structurer les interactions entre notre application et l'utilisateur.

2.2.1 Diagrammes de cas d'utilisation

La figure 2.1 illustre le diagramme de cas d'utilisation qui permet de représenter l'ensemble de fonctionnalités que dispose l'utilisateur de notre application.

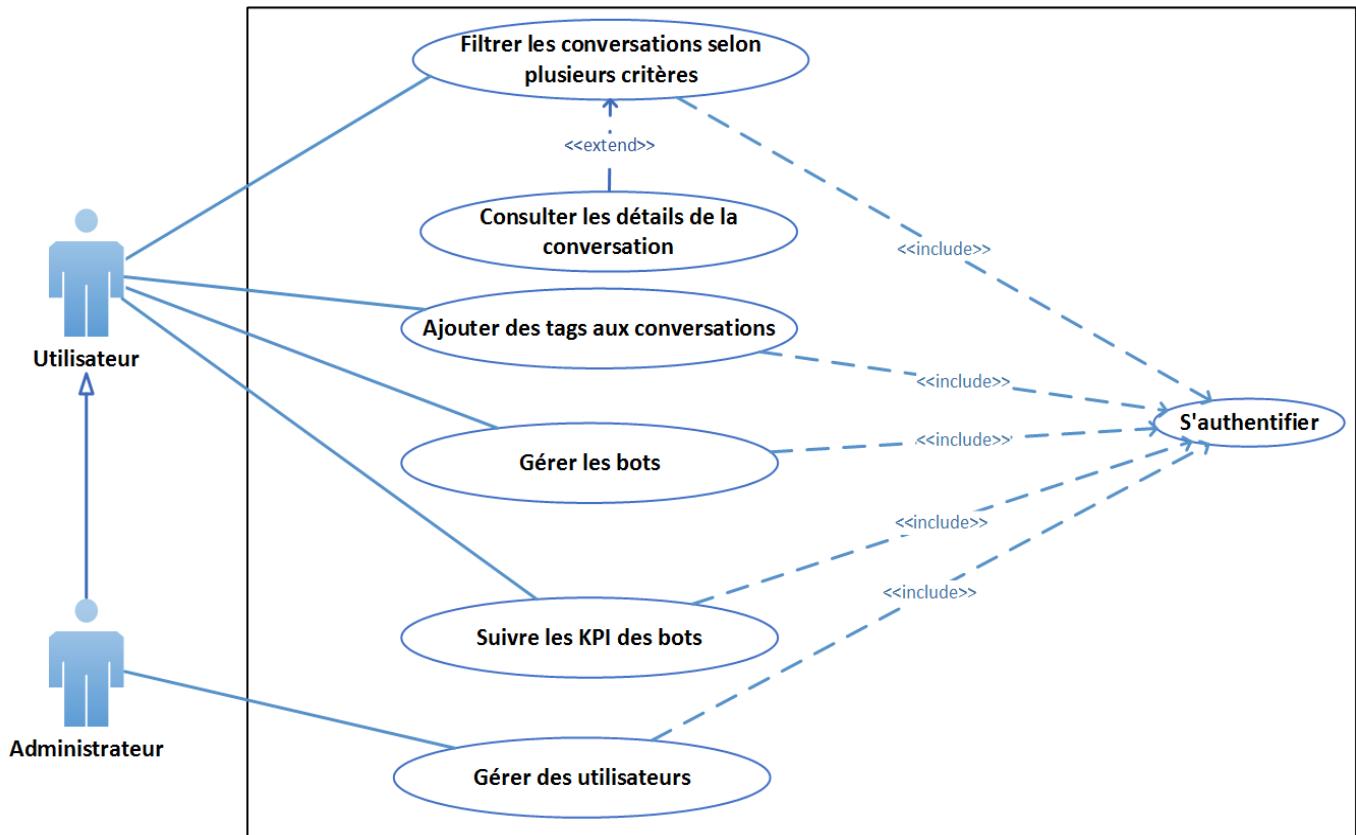


FIGURE 2.1 – Diagramme de cas d'utilisation global

Dans l'application de suivi et de paramétrage de Botman, l'utilisateur peut suivre les conversations de production, peut ajouter des tags et peut gérer les bots et suivre ses KPI. Mis à part les fonctionnalités qu'il hérite de l'acteur "utilisateur", l'administrateur peut gérer les comptes.

2.3 | Architecture de l'application

Dans cette sections, nous allons présenter l'architecture de l'application de suivi et de paramétrage de Botman à développer. Il est important de choisir une architecture qui nous permettra de répondre aux besoins définis précédemment.

2.3.1 Architecture physique

Afin de garantir une bonne performance et une réutilisation de notre système nous avons opté à suivre une architecture 3 tiers.

La figure 2.2 présente l'architecture physique de notre système.

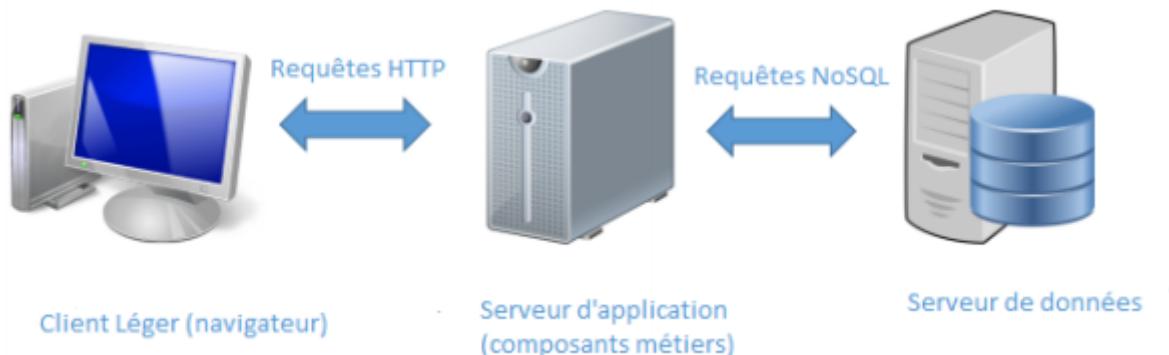


FIGURE 2.2 – Architecture physique de notre système [10]

- Client léger : Il comporte les interfaces utilisateur accessible par un navigateur web ;
- Serveur d'application : Il comporte la logique métier et il est chargé de fournir les ressources demandées par le client ;
- Serveur de données : Il contient le SGBD et il permet la centralisation ,le stockage et l'accès aux données ;

2.3.2 Architecture logique

L'architecture logique concerne la division logique du système et le regroupement des composants selon le traitement effectué. Nous avons adopté une architecture multi-couches pour le cas de notre solution de suivi.

La figure 2.3 illustre l'architecture logique de notre système.

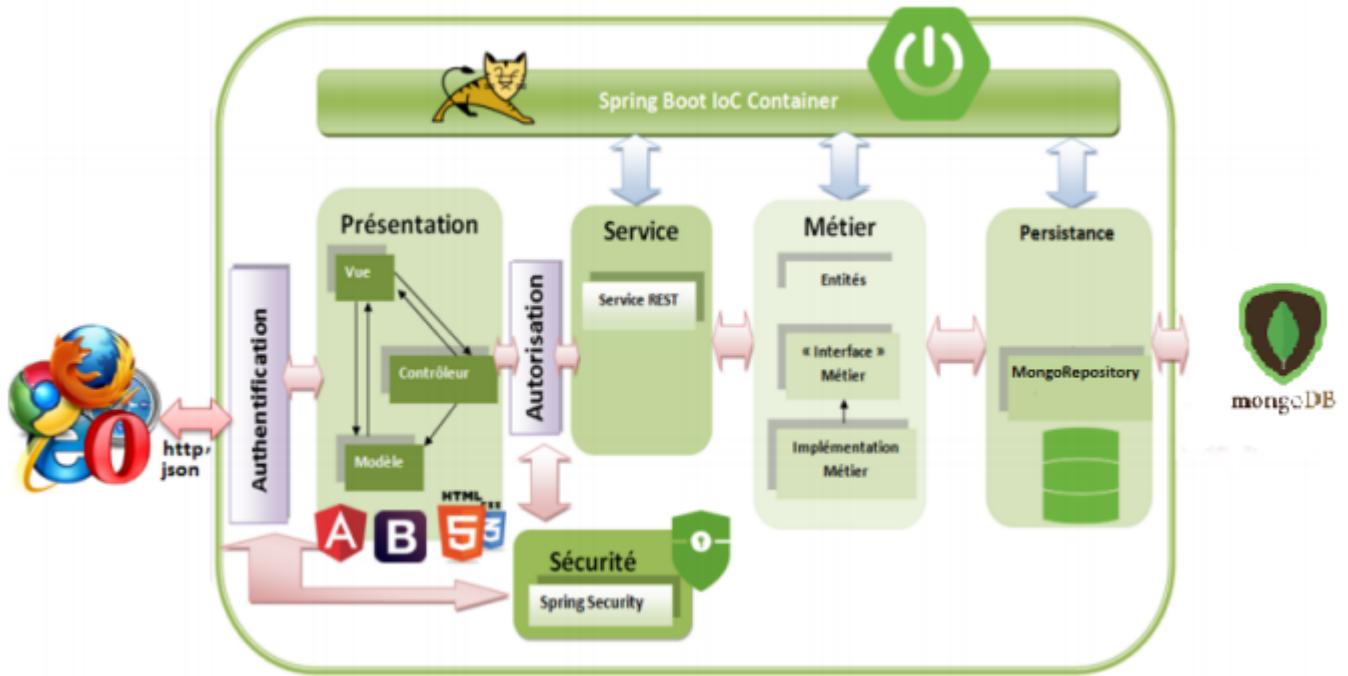


FIGURE 2.3 – Architecture logique de notre système [11]

Les différentes couches de l'architecture logiques sont :

- La couche de présentation : Elle comporte les composants Angular assurant la représentation graphiques. Cette couche permet d'assurer la logique de navigation et interagit avec à la couche service afin de répondre aux requêtes clients ;
- La couche métier : Elle englobe les traitements métiers de système, fait appel à la couche persistante pour répondre aux requêtes envoyés par la couche présentation.
- La couche sécurité : Elle est chargée de la sécurité de l'application ;
- La couche service : Elle contient l'implémentation des traitements sous forme de services REST et elle interagit avec la couche présentation via la couche métier ;
- La couche persistante : Elle permet le sauvegarde et l'accès aux données persistantes. Cette couche communique avec MongoDB qui est une base de données NoSQL en utilisant la robustesse du Framework Spring Data ;

2.4 | Backlog Produit

Pour assurer la conformité du produit final aux exigences du client, toutes les fonctionnalités sont rassemblées dans le Backlog produit qui représente le principal artefact dans chaque projet. Le tableau 2.1 présente le Backlog Produit.

id-F	Features	Id-Us	User Stories
1	Création d'un pipeline CI/CD pour le projet Batman	1.1	En tant que développeur je peux avoir une création d'un pipeline et un lancement du build de façon automatique suite à un "commit" du code
		1.2	En tant que développeur je peux déployer le code sur une machine
		1.3	En tant que développeur je peux lancer les tests fonctionnels sur un environnement
2	Création d'un pipeline CI/CD pour l'application de suivi et de paramétrage de Batman	2.1	En tant que développeur je peux avoir une création d'un pipeline et un lancement du build de façon automatique suite à un "commit" du code
		2.2	En tant que développeur je peux déployer le code sur une machine
3	Réingénierie d'un module backend du projet Batman	3.1	En tant que développeur je ne vais pas avoir des coupures dans la mise en production.

4	Réingénierie de la partie frontend du projet Botman	4.1	En tant que utilisateur je peux accéder à Botman à partir de tous type de device
5	Authentification	5.1	En tant que utilisateur je dois m'authentifier pour accéder à mon compte au sein de l'application
6	Gestion des utilisateurs	6.1	En tant que administrateur je peux ajouter un utilisateur
		6.2	En tant que administrateur je peux consulter un utilisateur
		6.2	En tant que administrateur je peux modifier un utilisateur
		6.2	En tant que administrateur je peux supprimer un utilisateur
7	Gestion des bots	7.1	En tant que utilisateur je peux ajouter un bot
		7.2	En tant que utilisateur je peux consulter un bot
		7.3	En tant que utilisateur je peux modifier un bot
		7.4	En tant que utilisateur je peux supprimer un bot

8	Filtrer les conversations selon plusieurs critères	8.1	En tant que utilisateur je peux effectuer une recherche sur les conversations selon plusieurs critères
		8.2	En tant que utilisateur je peux consulter les détails d'une conversation
9	Ajout des tags aux conversations	9.1	En tant que utilisateur je peux ajouter des tags aux conversations
10	Suivi des KPIs des bots	10.1	En tant que utilisateur je peux consulter le nombre des messages incompris par les bots
		10.2	En tant que utilisateur je peux consulter le nombre de conversations contenant des erreurs techniques
		10.2	En tant que utilisateur je peux consulter le nombre de conversations contenant des erreurs fonctionnelles
		10.2	En tant que utilisateur je peux consulter le nombre de conversations par jours pendant les 15 derniers jours

TABLE 2.1 – Backlog Produit

2.5 | Planification des sprints

La planification des sprints fait l'objet de l'une des réunions les plus importantes d'un projet Scrum.

La figure 2.6 illustre la planification des sprints de notre application.

Nous avons organisé de répartir le projet en quatre sprints :

— **Sprint 1 : DevOps - Création d'une chaîne de CI/CD**

Ce premier sprint sera consacré à création d'une chaîne CI/CD et d'y intégrer les deux développement.

— **Sprint 2 : La réingénierie du projet Botman**

Ce sprint consiste à développer une nouvelle fenêtre de webchat en pleine page et responsive en Angular et typescript. Aussi, au cours de sprint, nous allons modifier le module webchat de la partie backend du projet "Botman" existant en passant du protocole WebSocket au REST afin de remédier aux problèmes actuels.

— **Sprint 3 : Application de suivi et de paramétrage de Botman- Authentification et gestion**

Au cours de ce troisième sprint, nous allons commencer à développer l'application de suivi et de paramétrage de Botman.

Ce sprint consiste à développer l'authentification, la gestion des utilisateurs (ajout, suppression, modification ...) et la gestion de bots afin de passer d'une configuration gérée dans des fichiers JSON à un paramétrage par IHM.

Sprint 4 : Application de suivi et de paramétrage de Botman- Performance des bots

Dans ce sprint nous allons nous focaliser sur les conversations établies entre les bots et les utilisateurs pour pouvoir mesurer leurs performances.

Nous allons ajouter la fonctionnalité permettant de suivre les conversations de production et les filtrer en utilisant différents critères. Dans ce sprint, nous allons mettre en place aussi la fonctionnalité permettant le suivi des KPI des bots et finalement permettre à l'utilisateur d'ajouter des tags aux conversations.

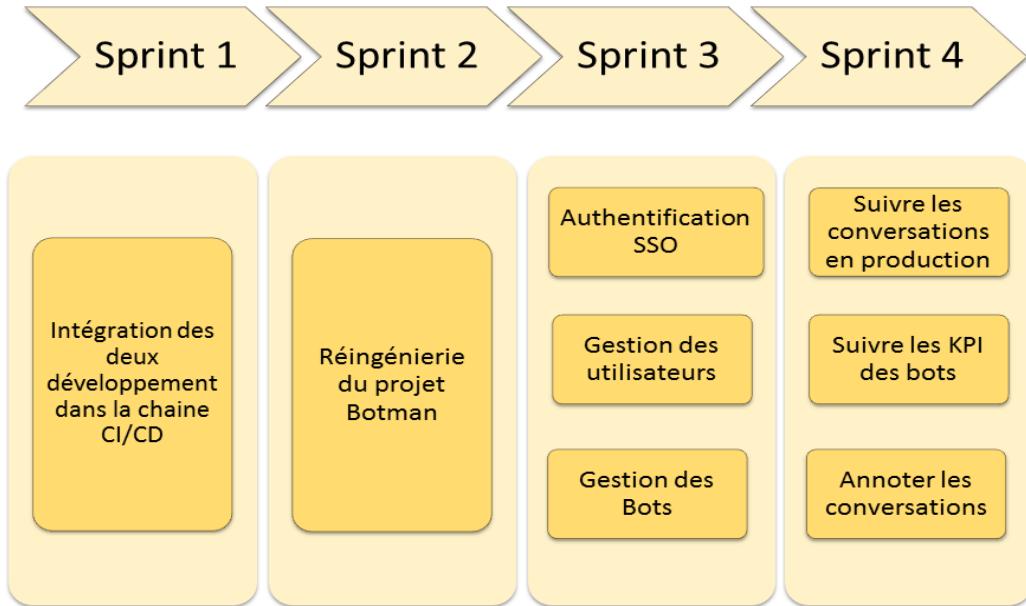


FIGURE 2.4 – Planification des sprints

2.6 | Environnement de travail

Cette partie est dédiée à la présentation de l'environnement matériel et logiciel qui nous permettront de réaliser notre projet.

2.6.1 Environnement matériel

Pour mettre en oeuvre notre système, nous utiliserons un PC possédant les caractéristiques suivantes :

- **Marque** : Dell
- **Processeur** : Intel(R) Core(TM) i5-4210 CPU @ 2.40GHz
- **RAM** : 8.0 Go
- **Système d'exploitation** : windows 7 Entreprise 64 bits

2.6.2 Environnement logiciel

Pour mettre en place ce projet nous avons eu recours à un ensemble de logiciels qui sont :

- **STS** (Spring Tool Suite) c'est un IDE utilisé pour le développement de la partie backend de l'application. Basé sur Eclipse, STS est spécialisé dans le développement des applications spring.
- **VSC** (Visual Studio code) est un éditeur [12] de code gratuit développé par Microsoft, opensource et fonctionne sur différentes plateformes. Il est très souvent mis à jour afin de corriger les bugs et ajouter des nouvelles fonctionnalités. Nous l'avons utilisé pour développer la partie frontend de notre système. Il Permet de faire des tests d'unités en direct. Il est conçu, principalement, pour le développement des applications codées en Typescript, JavaScript et nodeJs.
- **MongoDb** est un serveur de base de données NoSQL où les données sont stockées dans des documents JSON. Il assure une haute disponibilité, une distribution géographique et une mise à l'échelle puisque la base de données est distribuée [13].
- **Redis** (Remote Dictionary Server) est un système de stockage de données open source et rapide. Il est utilisé dans le cas de notre projet en tant que cache [14].
- **Apache Maven** Cet outil permet d'assurer la gestion et la compréhension de projets logiciels. À partir d'une information centrale, il peut gérer la construction, la documentation ainsi que les rapport d'un projet [15].

2.6.3 Framework et Outils

- **Angular** (version 7) est un Framework [16] développé par Google et basé sur le langage TypeScript. Ce framework décompose l'application en composants, modèles, modules et directives. Angular facilite l'évolution de l'application en favorisant la réutilisation des modules et en garantissant la modularité de l'application.

Nous optons pour l'utilisation de ce Framework pour le développement de la partie frontend de l'application.

- **Spring Boot** est un framework [17] libre utilisé pour faciliter le développement des applications basées sur Spring.

Spring Boot simplifie :

- la création des application Spring en offrant des outils permettant le développement des applications autonomes packagées en jar, war ... ;
- La configuration en fournissant des dépendances de démarreur ;
- Le développement en offrant des fonctionnalités prêtées à la production (la configuration externalisée, les mesures ...).

2.6.4 Langages de programmation

- **Java** est un langage de programmation orienté objet de haut niveau. Ce langage [18] favorise la réutilisation du code.
- **TypeScript** est un langage [19] conçu afin de faciliter le développement d'applications larges écrites en JavaScript. TypeScript représente une surcouche de javascript qui ajoute des concepts classiques tels que les modules, les interfaces, les classes ...
Le code TypeScript est un code javascript valide.
- **HTML5** (HyperText Markup Language) Il s'agit d'une version du langage de balisage HTML utilisé pour concevoir des pages web.
- **Sass** (Syntactically Awesome Stylesheets) Basée sur le CSS, Sass [20] est un langage dynamique de mise en forme et de design des pages web. Sass granatit la création des feuilles de styles plus modulaires. Nous avons opté à l'utilisation de Sass pour la réingénierie de la partie frontend du projet Botman pour simplifier la créations des interfaces responsives.

Conclusion

Au cours de ce chapitre, nous avons défini les acteurs de notre système, les besoins fonctionnels et non fonctionnels. Ceci nous a permis d'élaborer le diagramme de cas d'utilisation global. Également, nous avons présenté la répartition des sprints de notre projet. Nous avons finalement clôturé ce chapitre par décrire notre environnement de travail matériel et logiciel suivant les besoins de notre projet. Dans le prochain chapitre nous présenterons le sprint 1.

Chapitre 3

Sprint 1 : Devops-Création d'une chaîne CI/CD

Au cours de ce chapitre, nous allons entamer la partie réalisation du projet. Ce chapitre sera dédié au sprint 1 qui a pour objectif la création des chaînes CI/CD pour y intégrer les deux développements. Nous allons commencer ce chapitre par une définition de DevOps. Puis, nous allons présenter les outils Devops utilisés pour la mise en place du pipeline et nous allons clôturer ce chapitre par la partie réalisation où nous présentons le fonctionnement des pipelines illustré par des captures d'écrans explicatives.

3.1 | Sprint Backlog

Le tableau 3.1 présente le Sprint Backlog.

id-F	Features	Id-Us	User Stories
1	Création d'un pipeline CI/CD pour le projet Botman	1.1	En tant que développeur je peux avoir une création d'un pipeline et un lancement du build de façon automatique suite à un "commit" du code
		1.2	En tant que développeur je peux déployer le code sur une machine
		1.3	En tant que développeur je peux lancer les tests fonctionnels sur un environnement
2	Création d'un pipeline CI/CD pour l'application de suivi et de paramétrage de Botman	2.1	En tant que développeur je peux avoir une création d'un pipeline et un lancement du build de façon automatique suite à un "commit" du code
		2.2	En tant que développeur je peux déployer le code sur une machine

TABLE 3.1 – Sprint Backlog

3.2 | Démarche DevOps

3.2.1 Définition

Actuellement, les applications existantes évoluent rapidement et doivent s'adapter aux nouveaux besoins des clients. Dans ce contexte, le développement traditionnel présente une séparation entre les équipes de développement et les équipes des opérations. Chaque équipe a des objectifs qui peuvent souvent être opposés l'une à l'autre. Dans la majorité des cas, ce fait peut engendrer l'insatisfaction des clients.

En 2009 [21], le mouvement DevOps est apparu pour remédier à ces problèmes et ceci en réunissant les équipes chargées de développement et les équipes chargées des infrastructures (les OPS).

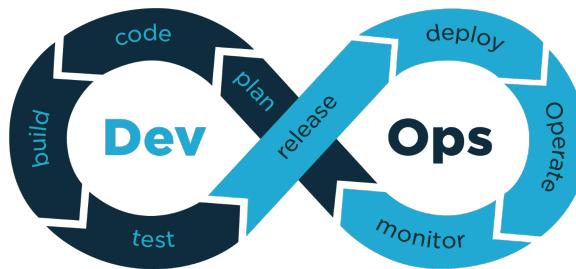


FIGURE 3.1 – DevOps [21]

Dans ce qui suit, nous allons définir l'intégration continue et le déploiement continu qui représentent des méthodes des développements logiciel DevOps.

3.2.2 Intégration continue

L'intégration continue représente un ensemble de pratiques qui consiste à s'assurer que chaque modification de code source effectuée par un développeur ne produit pas de conflits dans l'application [22]. L'intégration des changements apportés par un développeur dans le référentiel centralisé lance automatiquement des opérations de création et de test.

Selon [23] l'intégration continue aide à :

- Promouvoir la productivité des développeurs ;
- Déetecter et corriger rapidement les bogues ;
- Livrer les mises à jour dans des plus brefs délais.

3.2.3 Déploiement continu

Le déploiement continu représente le processus qui permet de transférer automatiquement dans l'environnement de production toute modification du code validé par des vérifications appropriées ainsi que des tests automatisés [24]. Le déploiement continu permet de :

- Baisser la durée et l'effort de déploiement ;
- Réduire les délais de livraison des bugs et des fonctionnalités ;
- Libère également des ressources.

3.3 | Éléments clés

Nous avons besoin d'un ensemble d'outils DevOps pour la mise en place d'un pipeline d'intégration continue et de livraison continue.

Les outils sur le marché sont très nombreux, nous avons choisi ceux qui répondent mieux aux besoins de notre projet.

Dans cette section, nous présentons l'ensemble des éléments clés utilisés pour l'intégration des développements dans une chaîne CI/CD.

3.3.1 Outil de gestion de code source : GitLab

Un Software Configuration Management permet une meilleure visibilité et un suivi de code source en offrant une collaboration entre les développeurs travaillant sur un même projet.

GitLab est un système de gestion de code gratuit et open source écrit en Ruby basé sur Git. Cet outil propose plusieurs fonctionnalités de suivi des problèmes liés au projet et de revues de code, il inclue un wiki, donne à ces utilisateurs un contrôle entier sur leurs projets ou référentiels [25].

la figure 3.2 illustre l'utilisation de GitLab dans le cas de notre projet.

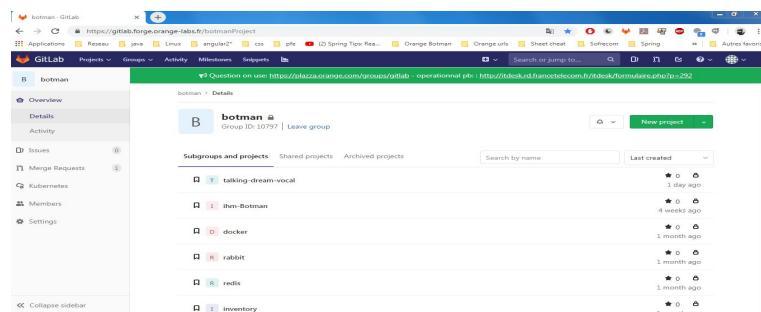


FIGURE 3.2 – Utilisation de GitLab

3.3.2 Outils d'intégration continue : Gitlab CI

La principale fonctionnalité des outils d'intégration continue est de permettre aux développeurs de commiter leurs codes vers un référentiel [26].

GitLab CI représente un outil d'intégration continue intégré à GitLab avec la sortie de la version 8.0 en septembre 2015. Dans cet outil , le processus CI/CD (déployer, tester, concevoir et superviser le code) est défini dans un fichier YAML [27].

3.3.3 Outils de qualimétrie logiciels : SonarQube

Les outils de Qualimétrie sont utilisés pour couvrir les tests et mesurer la qualité de code en se basant sur une analyse des données [28].

Développée par SonarSource, SonarQube est un logiciel de qualimétrie open source permettant de mesurer la qualité du code source et effectuer des tests très complets de façon continue. Il se base sur une analyse statique du code pour déterminer les bugs, la duplication de code, les failles de sécurité sur plusieurs langages de programmation ...[29]

3.3.4 Outils de gestion de configuration et automatisation : Ansible

Nous avons eu recours aux outils de configuration afin de contrôler l'infrastructure de l'entreprise. Ils garantissent l'uniformité des mises à jours, éliminent les problèmes de contrôle de versions et évitent les variations entre les serveurs et l'infrastructure [30].

Ansible est un outil de gestion de configuration et automatisation Open Source. Ce moteur d'automatisation met fin aux tâches répétitives, facilite l'orchestration, le déploiement applicatif et la gestion de configuration. Il est conçu avec une architecture sans agents garantissant la sécurité et réduisant les frais de réseau. Cet outil est fiable, simple et contient un minimum de dépendance[31].

La figure 3.3 représente l'architecture de l'outil Ansible.

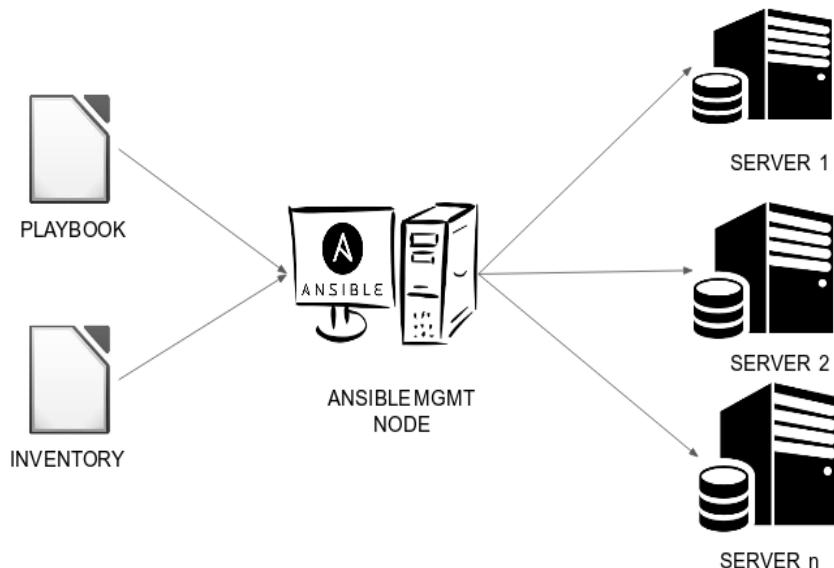


FIGURE 3.3 – Architecture Ansible [32]

Pour fonctionner, ansible se connecte aux noeuds et pousse les "Ansible Modules".

- **Ansible Modules** sont des petits programmes qui définissent des modèles de ressources représentant un état souhaité du système.
- **Inventory** ce fichier définit la liste des machines sur lesquelles Ansible exécute l'ensemble de tâches. Dans ce fichier on peut regrouper les machines et utiliser ces groupes pour la configuration.
- **Playbooks** Ce sont des fichiers écrit en YAML utilisés pour :
 - effectuer les tâches et les étapes de configuration sur les noeuds ;
 - suivre et répartir les charges sur les serveurs ;
 - réaliser le déploiement.

3.3.5 Outil de conteneurisation : Docker

Docker est une plateforme logicielle de conteneurisation open source. Cet outil permet la création, le déploiement et la gestion des conteneurs des applications virtualisées sur un système d'exploitation. Docker présente plusieurs avantages, on peut citer le fait que :

- Il utilise peu de ressources ;
- Il offre un déploiement rapide ce qui implique un développement plus efficient [33].

3.3.6 Outil d'orchestration : DockerSwarm

DockerSwarm est une plateforme open-source permettant l'orchestration d'un ensemble de conteneurs répartis sur plusieurs serveurs [34]. Cet outil constitue le moteur de clusternig. L'utilisation et l'installation de dockerSwarm est très simple. Dans le cas de notre projet nous utilisons uniquement une infrastructure avec Docker. Donc,l'utilisation de DockerSwarm se révèle la plus adapté puisque son intégration à Docker est facile[34].

3.3.7 Cycle de vie d'un User Story dans le projet Botman

La figure 3.4 représente le cycle de vie d'un US dans l'application Botman.

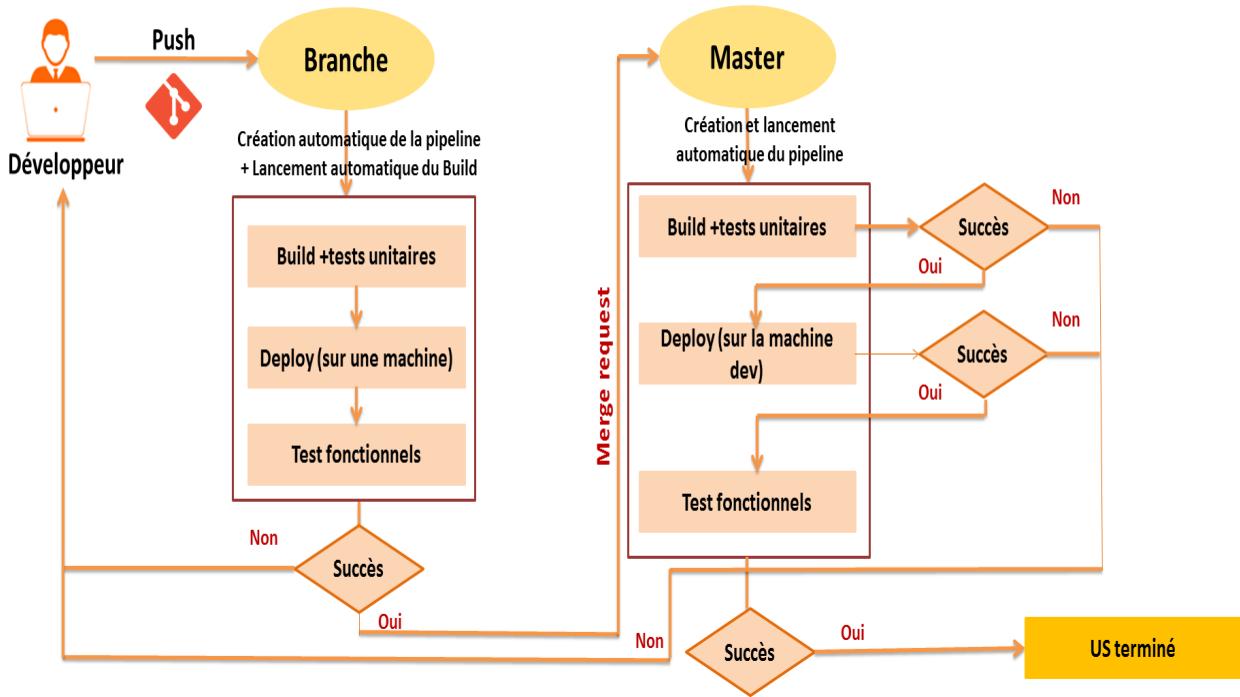


FIGURE 3.4 – Cycle de vie d'un US

Le développeur fait un "push" de son code sur Git, un pipeline est créé automatiquement et lance un build. Une fois le build terminé, le développeur peut lancer le deploy et les tests fonctionnels. Si les tests fonctionnels ne comportent pas d'erreurs, le développeur peut effectuer un merge request sur le master sinon il doit vérifier son environnement et son code et procéder aux corrections avant de re-committer. Le merge request crée et lance la pipeline. Si tout se passe bien l' US est terminé sinon le développeur doit corriger les erreurs.

3.4 | Réalisation

3.4.1 Configuration et Automatisation de déploiement avec Ansible

Dans cette section nous présentons des extraits des fichiers Inventory et Playbook utilisés pour gérer la configuration et l'automatisation de déploiement.

La figure 3.5 représente un extrait du fichier Inventory qui contient l'ensemble de machines de notre projet.

```

# inventory_dev_e
1 [manager_launcher]
2 dvfwcws0e ansible_host=XXXXXXXXXX ansible_user=adfvcwas
3 [as]
4 dvfwcws0e ansible_host=XXXXXXXXXX ansible_user=adfvcwas
5 [rasa_node]
6 dvfwcws0e ansible_host=XXXXXXXXXX ansible_user=adfvcwas
7 [manager_launcher:vars]
8 buildPath=
9 registryAddress=XXXXXXXXXX
10 confPath=/opt/application/fwcmas/current/properties/
11 webchatReplicas=1
12 smartlyReplicas=1
13 restApiReplicas=1
14 watsonReplicas=1
15 skypeReplicas=0
16 rasaReplicas=1
17 livepersonReplicas=1
18 enrichmentReplicas=1
19 dyduReplicas=1
20 dispatcherReplicas=1
21 rabbitReplicas=1
22 redisReplicas=1
23 asConstraints- node.labels.as == true
24 asConstraints2-
25 rabbitConstraints- node.labels.rabbit == true
26 redisConstraints- node.labels.redis == true
27 botmanPath=/home/docker/botman
28
29 [as:vars]
30 logPath=/opt/application/fwcmas/current/LOGS/
31

```

FIGURE 3.5 – Extrait du fichier Inventory

La figure 3.6 représente un extrait du fichier Playbook responsable de déploiement automatique. Ce fichier fait un appel aux rôles Ansible qui définissent l'ensemble des opérations et des tâches.

```

! swarm_botanyml !
1 - name: serveur AS
2 hosts: cluster
3 become: yes
4 tasks:
5 #####
6 # CREATE logs Directory
7 ######
8 #####
9   - name: create log directory if not exists
10   file:
11     path: "{{ logPath }}"
12     state: directory
13     mode: 0755
14     recurse: yes
15     owner: adfvcwas
16     group: gpfcwmas
17
18   #- name: Log into private registry and force re-authorization
19   #docker_login:
20     #registry: registry.forge.orange-labs.fr
21     #username: gitlab+deploy-token-741
22     #password: mX1BpxyXBRXLf2YCs-v
23     #reauthorize: no
24     #when: "dev" not in inventory_file'
25
26   - name: manager launcher
27     hosts: manager_launcher
28     become: yes
29     tasks:
30

```

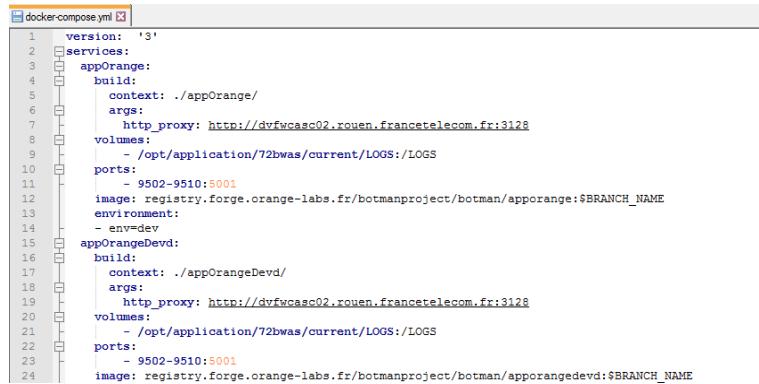
FIGURE 3.6 – Extrait du fichier Playbook

3.4.2 Botman

Dans cette section nous présentons les fichiers ajoutés au projet Botman pour aboutir à la création d'un pipeline CI/CD.

3.4.2.1 docker-compose.yml

La figure 3.7 représente un extrait du fichier docker-compose.yml du projet Botman. Ce fichier permet l'exécution et la mise en relation des différentes images docker (permet de représenter de manière statique un service ou une application ainsi que leurs dépendances et leurs configurations) dont le projet Botman a besoin.



```

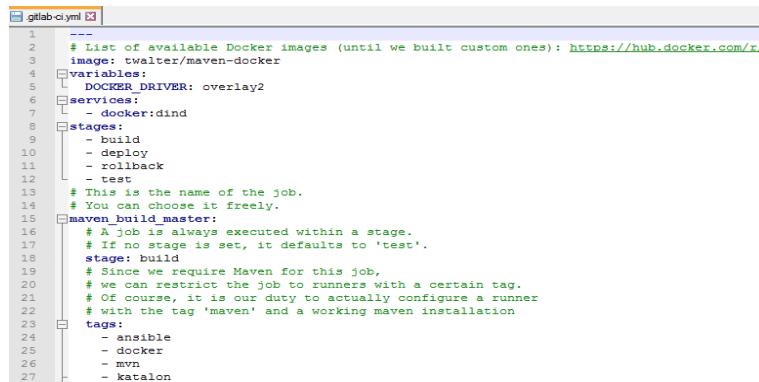
version: '3'
services:
  appOrange:
    build:
      context: ./appOrange/
      args:
        http_proxy: http://dsvfcasc02.rouen.francetelecom.fr:3128
    volumes:
      - /opt/application/72bwas/current/LOGS:/LOGS
    ports:
      - 9502-9510:5001
    image: registry.forge.orange-labs.fr/botmanproject/botman/apporange:$BRANCH_NAME
    environment:
      - env=dev
  appOrangeDevd:
    build:
      context: ./appOrangeDevd/
      args:
        http_proxy: http://dsvfcasc02.rouen.francetelecom.fr:3128
    volumes:
      - /opt/application/72bwas/current/LOGS:/LOGS
    ports:
      - 9502-9510:5001
    image: registry.forge.orange-labs.fr/botmanproject/botman/apporangedeved:$BRANCH_NAME

```

FIGURE 3.7 – extrait du fichier docker-compose.yml du projet Botman

3.4.2.2 gitlab-ci.yml

La figure 3.8 représente un extrait du fichier gitlab-ci.yml qui définit le processus CI/CD.



```

# List of available Docker images (until we built custom ones): https://hub.docker.com/r/
variables:
  DOCKER_DRIVER: overlay2
services:
  - docker:dind
stages:
  - build
  - deploy
  - rollback
  - test
# This is the name of the job.
# You can choose it freely.
maven_build_master:
  # A job is always executed within a stage.
  # If no stage is set, it defaults to 'test'.
  stage: build
  # Since we require Maven for this job,
  # we can restrict the job to runners with a certain tag.
  # Of course, it is our duty to actually configure a runner
  # with the tag 'maven' and a working maven installation
  tags:
    - ansible
    - docker
    - mvn
    - katalon

```

FIGURE 3.8 – extrait du fichier gitlab-ci.yml du projet Botman

3.4.2.3 Dockerfile

La figure 3.9 représente le Dockerfile du projet Botman. Ce fichier permet de définir tous les étapes nécessaires à la fabrication d'une image.

```

1 FROM openjdk:8u171-jdk-slim-stretch
2 VOLUME /tmp
3 ADD target/appWebChatng-0.0.1-SNAPSHOT.war appWebChatng-0.0.1-SNAPSHOT.war
4 ADD botman-config botman-config
5 ADD botman-config-INIT botman-config-INIT
6 ENTRYPOINT ["java","-jar","-Dmodule=webchat","-Dext.prop=/botman-config/",
7 "-Dlogging.config=/botman-config/logConf/logback.xml","appWebChatng-0.0.1-SNAPSHOT.war"]

```

FIGURE 3.9 – fichier Dockerfile du projet Botman

3.4.2.4 Pipeline CI/CD du projet Botman

la figure 3.10 représente le pipeline CI/CD du projet Botman. Le pipeline contient trois stages :

- Build ;
- Deploy ;
- Tests fonctionnels .

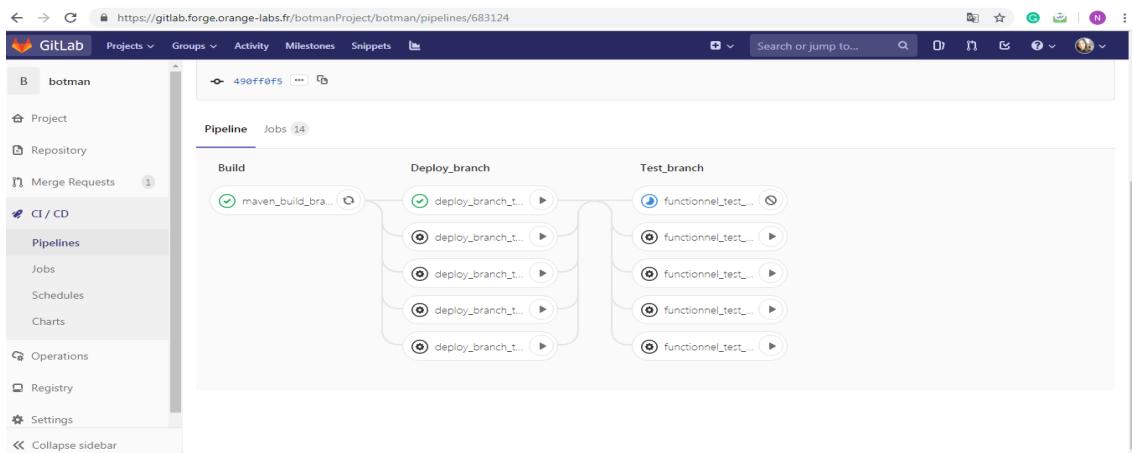


FIGURE 3.10 – pipeline CI/CD du projet Botman

3.4.3 IHM de suivi et de paramétrage de Botman

3.4.3.1 docker-compose.yml

La figure 3.11 représente un extrait du fichier docker-compose.yml de l'IHM de suivi et de paramétrage de Botman.

```

version: '3'
services:
  ihmback:
    build:
      context: ./back/
      image: registry.forge.orange-labs.fr/botmanproject/ihm-botman/ihmback:$BRANCH_NAME
  ihmfront:
    build:
      context: ./front/
      image: registry.forge.orange-labs.fr/botmanproject/ihm-botman/ihmfront:$BRANCH_NAME
  
```

FIGURE 3.11 – extrait du fichier docker-compose.yml de l'IHM de suivi et de paramétrage de Botman

3.4.3.2 gitlab-ci.yml

La figure 3.12 représente un extrait du fichier gitlab-ci.yml de l'IHM de suivi et de paramétrage de Botman.

```

---
# List of available Docker images (until we built custom ones): https://hub.docker.com/r/weldpua2009/docker-ansible/
image: walter/maven-docker
variables:
  DOCKER_DRIVER: overlay2
services:
  docker:dind
stages:
  - build
  - test
  - deploy_branch
# This is the name of the job.
# You can choose it freely.
maven_build_branches:
# A job is always executed within a stage.
# If no stage is set, it defaults to 'test'.
stage: build
# Since we require Maven for this job,
# we can restrict the job to runners with a certain tag.
# Of course, it is our duty to actually configure a runner
# with the tag 'maven' and a working maven installation
tags:
  - botman
  - ihm
  - ansible
  - docker
  
```

FIGURE 3.12 – extrait du fichier gitlab-ci.yml de l'IHM de suivi et de paramétrage de Botman

3.4.3.3 Dockerfile

La figure 3.13 représente le Dockerfile de la partie frontend de l'IHM de suivi et de paramétrage de Botman.

```

FROM openjdk:8u171-jdk-slim-stretch
VOLUME /tmp
ADD target/appWebChatng-0.0.1-SNAPSHOT.war appWebChatng-0.0.1-SNAPSHOT.war
ADD botman-config botman-config
ADD botman-config-INIT botman-config-INIT
ENTRYPOINT ["java","-jar","-Dmodule=webchat","-Dext.prop=/botman-config/","-Dlogging.config=/botman-config/logConf/logback.xml","appWebChatng-0.0.1-SNAPSHOT.war"]
  
```

FIGURE 3.13 – fichier Dockerfile frontend de l'IHM de suivi et de paramétrage de Botman

La figure 3.14 représente le Dockerfile de la partie backend et l'IHM de suivi et de paramétrage de Botman.

```

1 FROM openjdk:8u171-jdk-slim-stretch
2 VOLUME /tmp
3 ADD target/appWebChatng-0.0.1-SNAPSHOT.war appWebChatng-0.0.1-SNAPSHOT.war
4 ADD botman-config botman-config
5 ADD botman-config-INIT botman-config-INIT
6 ENTRYPOINT ["java","-jar","-Dmodule=webchat","-Dext.prop=/botman-config/",
7 "-Dlogging.config=/botman-config/logConf/logback.xml","appWebChatng-0.0.1-SNAPSHOT.war"]

```

FIGURE 3.14 – fichier Dockerfile backend de l'IHM de suivi et de paramétrage de Botman

3.4.3.4 Pipeline CI/CD de l'IHM de suivi et de paramétrage de Botman

La figure 3.15 représente le pipeline CI/CD réalisé pour l'IHM de suivi et de paramétrage de Botman.

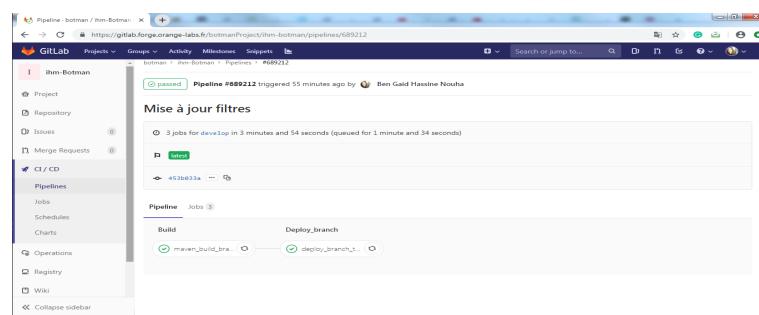


FIGURE 3.15 – pipeline CI/CD de l'IHM de suivi et de paramétrage de Botman

Conclusion

Ce premier sprint nous a permis de nous familiariser avec le mouvement DevOps et de créer un pipeline CI/CD pour le projet Botman et l'IHM de suivi et paramétrage de Botman. Dans le chapitre suivant nous présenterons le sprint 2 qui a pour objectif la réingénierie de la partie frontend et la partie backend du module webchat du Projet Botman.

Chapitre 4

Sprint 2 : Réingénierie du projet botman

Ce chapitre sera consacré au sprint 2 qui a pour objectif la réingénierie de la partie frontend et le module webchat de la partie backend du projet botman. Nous allons commencer ce chapitre par définir le fonctionnement global du projet botman. Puis nous allons entamer la partie réingénierie de la partie frontend et backend. Nous allons clôturer ce chapitre par présenter la réalisation de ce sprint.

4.1 | Sprint Backlog

Le tableau 4.1 présente le Backlog Produit.

id-F	Features	Id-Us	User Stories
3	Réingénierie d'un module backend du projet Botman	3.1	En tant que développeur je ne vais pas avoir des coupures dans la mise en production.
4	Réingénierie de la partie frontend du projet Botman	4.1	En tant que utilisateur je peux accéder à Botman à partir de tous type de device

TABLE 4.1 – Sprint Backlog

4.2 | Fonctionnement Global du projet Botman

La figure 4.1 illustre le fonctionnement global du projet Botman.

Dans le projet Botman, c'est l'utilisateur qui pose une question. Puis, l'orchestrateur reçoit la requête, la traite et la redirige vers le bot approprié. Puis, l'orchestrateur obtient la réponse pour la transmettre à l'utilisateur.

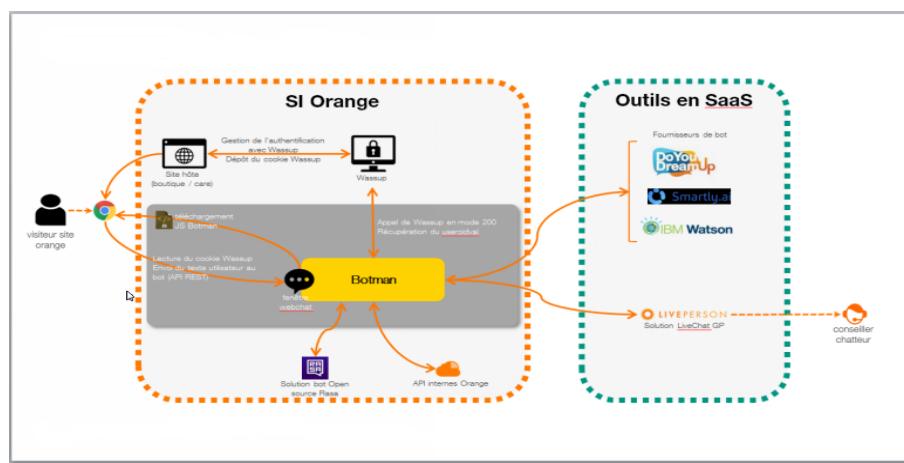


FIGURE 4.1 – Fonctionnement Global de Botman

4.2.1 Architecture logique du projet Botman

Le projet Botman repose sur une architecture multi-modules. Cette architecture nous permet de séparer les différents modules (les modules ne communiquent pas directement entre eux).

La figure 4.2 représente l'architecture logique du projet Botman.

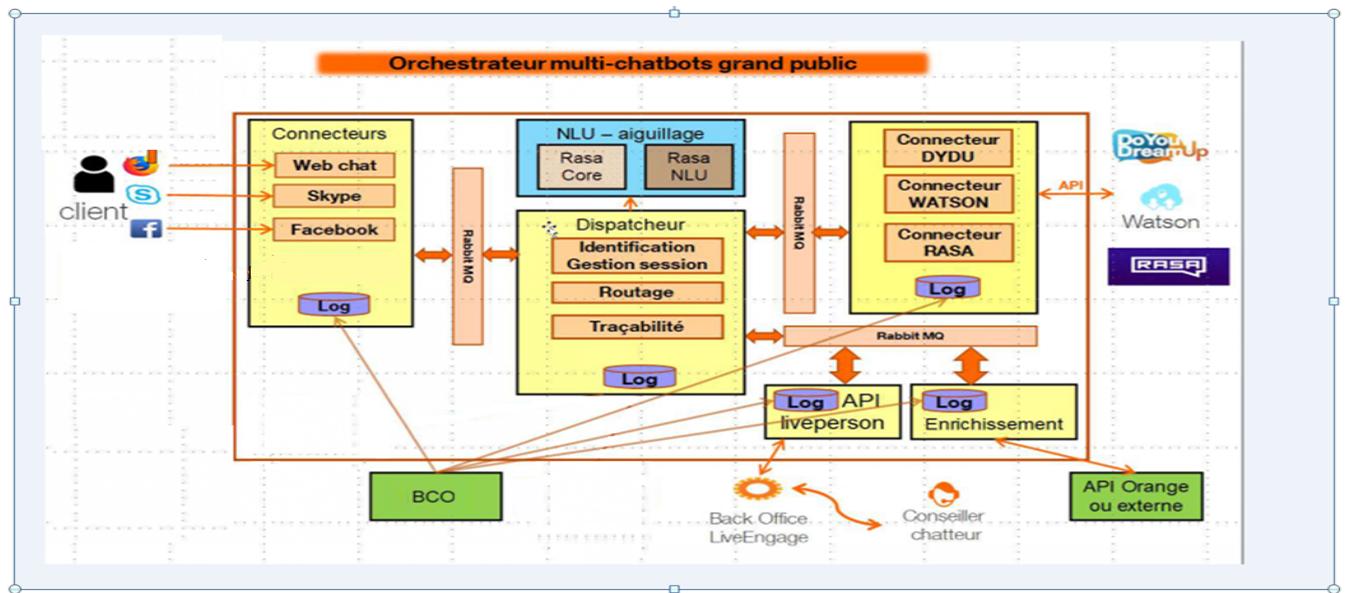


FIGURE 4.2 – l'architecture logique du projet Botman

les modules présents dans notre projet sont :

- **appDispatcher** : C'est le module qui s'occupe de l'orchestration des messages.
- **appCommon** : Ce module contient tous les fichiers qui peuvent être utilisés par plus qu'un module.
- **appDydu** : Ce module reçoit le message du Dispatcher et initie l'API du DYDU et renvoie le message au Dispatcher.
- **appEnrichment** : Ce module contient les appels des API en interne d'orange.
- **appLiveperson** : Ce module initie les appels à l'API liveEngage pour permettre les chats avec des agents réels.
- **appMessenger** : C'est un prototype réalisé pour lancer un bot Messenger.
- **appMocks** : Ce module contient les Mocks (simulateurs) que nous avons besoin pour les autres modules.
- **appOrange** : Ce module fournit un fonctionnement similaire aux pages où on peut intégrer Botman via un script d'intégration.
- **appRasa** : Ce module initie les API de Rasa.
- **appRestApi** : Ce module contient les différents traitements qui permettent d'initier l'API Botman.
- **appSkype** : Permet d'initier l'api REST "UCWA" pour communiquer avec skype entreprise.

- **appSmartly** : Ce module initie l'API smartly.
- **appWatson** : Ce module initie l'API Watson d'IBM.
- **appWebChatng** : présente l'un des points d'entrée du projet Botman, ce module interprète le message du client et l'envoie au dispatcher.

En effet, la réingénierie de la partie backend du projet concerne que le module appWebChatng.

Par ailleurs, les modules communiquent entre eux par des messages JMS à travers un provider JMS que nous définissons dans la section suivante.

4.2.2 Provider JMS

Il s'agit d'un outil qui permet d'échanger les messages en implémentant l'API JMS.

La figure 4.3 représente le fonctionnement d'un provider JMS. En effet, le producteur de message envoie son message au JMS provider en l'intégrant dans sa file et le consommateur lit le message à partir de cette même file.

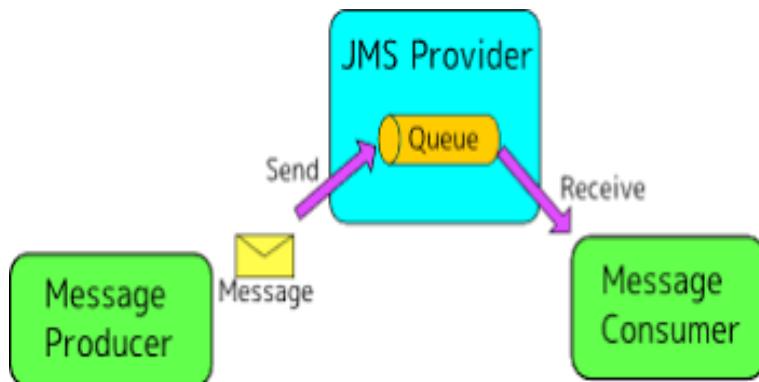


FIGURE 4.3 – Provider JMS [35]

Dans le cas de notre projet nous avons opté à l'utilisation du provider JMS RabbitMQ que nous définissons dans ce qui suit.

4.2.3 RabbitMQ

RabbitMQ représente une solution logicielle permettant à des applications de communiquer en offrant un réseau d'échange d'informations. RabbitMQ utilise l'échange de messages pour communiquer les informations. Il implémente le protocole AMQP (Advanced Message Queuing Protocol) qui définit le transport de messages entre les différentes applications. Ce protocole est wire-level (similaire aux protocoles TCP et HTTP) et permet un transport

asynchrone. La connexion des applications se fait grâce à un broker par le biais de messages routés.

Le broker représente l'intermédiaire, il reçoit un message d'une application et le transmet à une autre [36].

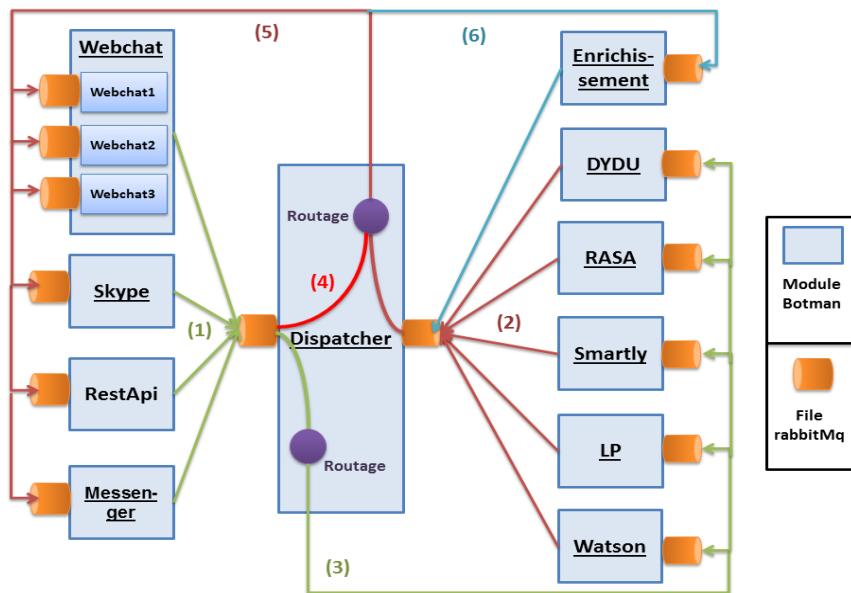


FIGURE 4.4 – Communication entre les différents modules de Botman via RabbitMq

La figure 4.4 représente un schéma explicatif de la communication entre les différents modules de Botman via RabbitMq :

- Tous les modules des connecteurs (entrypoints) envoient leurs messages sur une file d'entrée commune pour les entrypoints du module dispatcher.(1)
- Tous les modules des NLU / enrichissement / LP envoient leurs messages sur une file d'entrée commune pour les endpoints du module dispatcher.(2)
- Un message provenant d'un endpoint (webchat, skype...) sera redirigé lors du routage du module dispatcher vers une file spécifique à l'endpoint dédié (dydu, rasa, smartly, lp ...) tant que la réponse n'est pas fournie par le dispatcher (3) sinon la réponse du dispatcher sera envoyée sur une file spécifique au connecteur dédié(exemple dans le cas de initiation de conversation ou BotmanInfo) (4) puis (5).
- Un message provenant d'un endpoint sera redirigé lors du routage du module dispatcher vers une file spécifique au connecteur dédié (5) ou vers la file de appEnrichissement s'il s'agit d'un appel à une API(6).

- Un message provenant du module enrichissement sera redirigé vers une file spécifique au connecteur dédié (5).

4.3 | Réingénierie d'un module backend du projet "Botman"

Dans cette section, nous présentons la réingénierie du module backend "appWebChatng" du projet botman. Nous commençons par définir les Web Services REST puis nous présentons le travail élaboré pour passer du WebSocket vers le REST.

4.3.1 Web service REST

Les services Web permettent à des applications qui utilisent des technologies différentes de communiquer en se basant sur les standards XML et HTTP [37].

REST (Representational State Transfer) représente un type de service web. C'est un style d'architecture qui repose sur le protocole HTTP, il expose les ressources à accéder à travers les URLs (Uniform Ressource Locator) et y effectue des opérations par le biais des méthodes HTTP (GET, POST, PUT, UPDATE, DELETE ...)[38].

La figure 4.5 représente une architecture web service Rest.

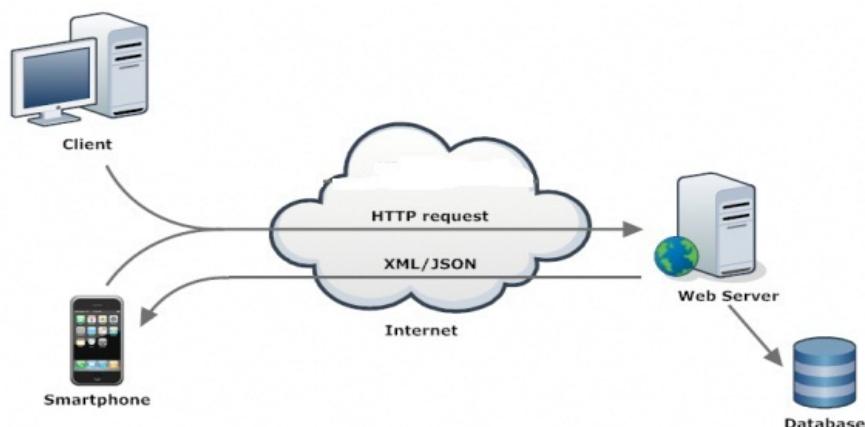


FIGURE 4.5 – Architecture Service Web Rest [39]

Comme le montre cette figure, le client utilise une URL HTTP pour consommer un service Web REST et le serveur envoie en format JSON sa réponse.

4.3.2 Travail élaboré

Comme nous l'avons déjà mentionné dans l'étude de l'existant, nous devons gérer le comportement du REST pour l'adapter aux besoins de notre projet. Tout d'abord nous avons créé un nouveau module qui contient la nouvelle partie Frontend et la transformation du module "appWebChatng" en REST. Dans cette section, on va s'intéresser à la transformation du module backend.

En effet, l'ancienne version du module est basé sur les Sockets qui présente une communication Bidirectionnelle. Dans le cas du REST, la communication est unidirectionnelle. Donc pour remédier à ça, nous avons créer deux nouvelles méthodes send et receive.

```
@GetMapping("/receive")
public ResponseEntity<MessageChat> receiveMsg(@RequestHeader("token") String token) {
```



```
@PostMapping("/send")
public String sendMsg(@RequestBody String message, @RequestHeader("messageType") String messageType,
    @RequestHeader(name = "token", defaultValue = "") String token) {
```

FIGURE 4.6 – Les entêtes des méthodes receive et send

La figure 4.6 représente les entêtes des méthodes receive et send du module "appWebChatng". Comme le websocket est statefull, il présente la notion de sessions (une session par utilisateur). Le REST est stateless donc nous avons ajouté la notion de token par utilisateur pour remédier à ça. En effet, dans la méthode send, on doit vérifier si le token existe ou pas. S'il existe on doit vérifier sa validité sinon on doit en créer un et l'ajouter dans Redis et puis créer une file (queue) dans RabbitMQ. Dans la méthode receive, aussi, on doit procéder à la vérification de l'existence de token et de la file pour poursuivre le traitement.

4.4 | Réingénierie partie FrontEnd de l'orchestrateur Botman

La partie frontend de la solution actuellement existante est réalisée en Javascript et Jquery. Comme nous avons discuté au premier chapitre, le développement avec Angular et TypeScript présente un grand nombre d'avantages. Dans ce cadre, Orange a décidé de migrer vers cette technologie pour le développement de son orchestrateur de chatbots.

En effet, l'application existante met à disposition une fenêtre de webchat intégré dans une page dans le coin droit inférieur utilisable que sur des écrans desktop. Le nouveau besoin de

cette application est de développer une interface full screen responsive conforme dans tous les navigateurs.

4.4.1 RWD (Responsive Web Design)

De nos jours, le nombre de personnes qui accèdent à Internet et aux applications à partir de leurs smartphones et tablettes devient de plus en plus important.

La figure 4.7 compare l'utilisation des mobiles par rapport aux ordinateurs pour accéder à Internet entre les années 2006 et 2016.

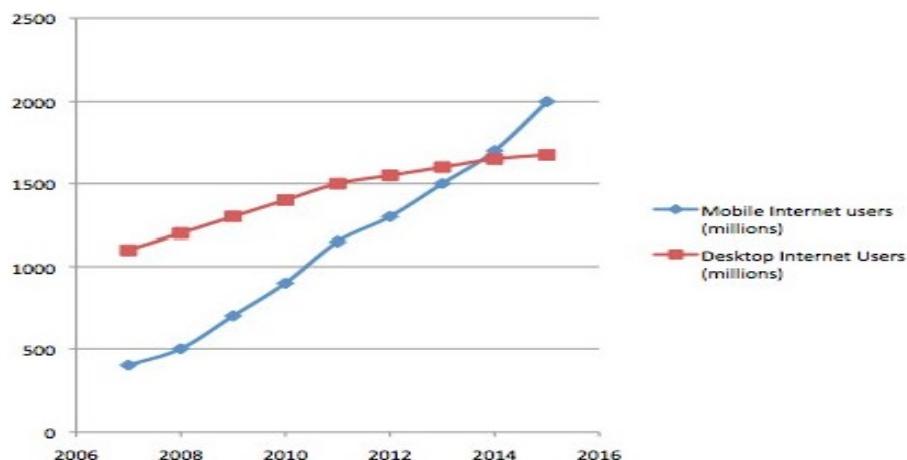


FIGURE 4.7 – Utilisation mobiles VS ordinateurs [40]

Or les conceptions classiques des applications Web ne s'adaptent pas à la visualisation sur des écrans des smartphones et tablettes (des écrans à une petite résolution).

D'où l'apparition de RWD qui a été introduit par Ethan Marcotte en mai 2010 dans un article de "A LIST APART"[41].

RWD représente une approche de conception Web qui permet d'ajuster d'une manière automatique l'affichage des interfaces web à la taille d'écran de l'appareil utilisé (mobiles, tablettes, ordinateur...) de manière transparente pour l'utilisateur.

La figure 4.8 illustre la visualisation d'une même applications web sur différentes tailles d'écrans.



FIGURE 4.8 – Responsive Web Design [42]

L'utilisation de RWD permet de faciliter la maintenance de notre projet et d'effectuer des mise à jour transparentes et un déploiement multi-plateformes.

Pour mettre en place des interfaces responsives, nous avons utilisé les règles CSS appelées MediaQueries.

4.4.2 Les Medias Queries

Les Media Queries représentent l'une des nouveautés de Css3.

Ils représentent des règles qui s'appliquent pour modifier le design des interfaces en se basant sur les caractéristiques de l'écran utilisée.

Dans le cas de notre projet nous avons définis cinq Medias Queries comme le présente la figure 4.9 pour exprimer les règles à suivre pour cinq tailles d'écrans différentes.

```
@media screen and (max-width: 4000px) and (min-width: 1025px) {  
  
    @media screen and (max-width: 1024px) and (min-width: 769px) {  
  
        @media screen and (max-width: 768px)and (min-width: 500px){  
  
            @media screen and (width: 360px) and (height:480px ){  
  
                @media screen and (max-width: 499px){
```

FIGURE 4.9 – Media Queries de notre projet

4.4.3 Travail élaboré

Mis à part la création des MediaQueries pour aboutir à des interfaces responsives, nous avons géré quelques anomalies que présente le REST pour l'adapter à nos besoins. En effet, le websocket présente une seule file RabbitMQ d'où la diffusion automatique des messages pour tous les listeners.

Comme REST utilise une file pour chaque session, nous devons donc gérer la diffusion des messages. Pour se faire, nous avons créer une tâche planifié chaque 1 seconde qui permet de comparer et mettre à jour le contenu des local Storage des listeners. Si l'un des listeners présente plus de messages dans son local Storage, ils seront propagés aux autres afin de garder un contenu identique pour tous les listeners.

4.5 | Réalisation

Au cours de cette section, nous allons présenter la réalisation relative à ce sprint. les figures 4.10, 4.11, 4.12, 4.13 et 4.14 présentent une interaction entre un utilisateur et un bot et illustrent l'interface Botman pour différentes tailles d'écrans.

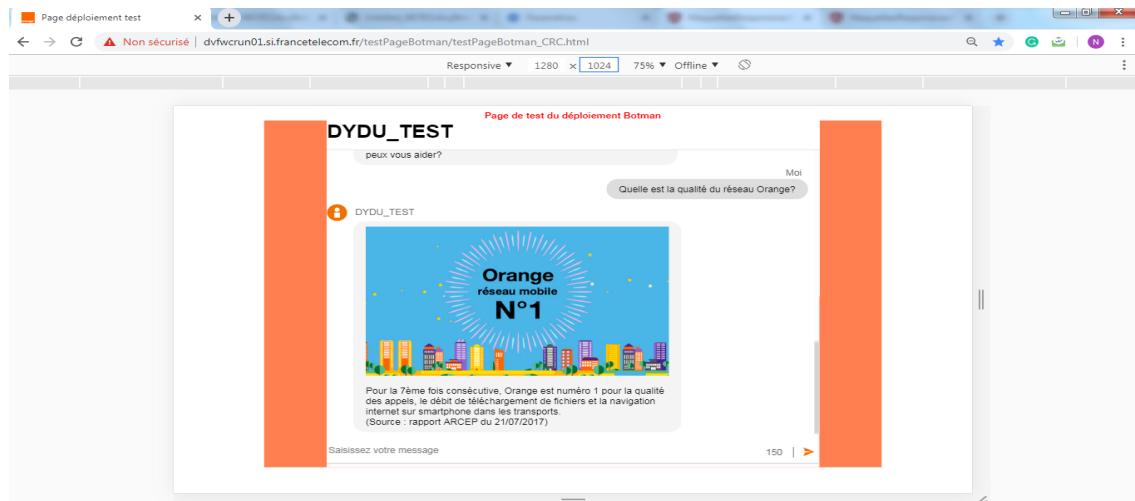


FIGURE 4.10 – Interface de taille 1280*1024

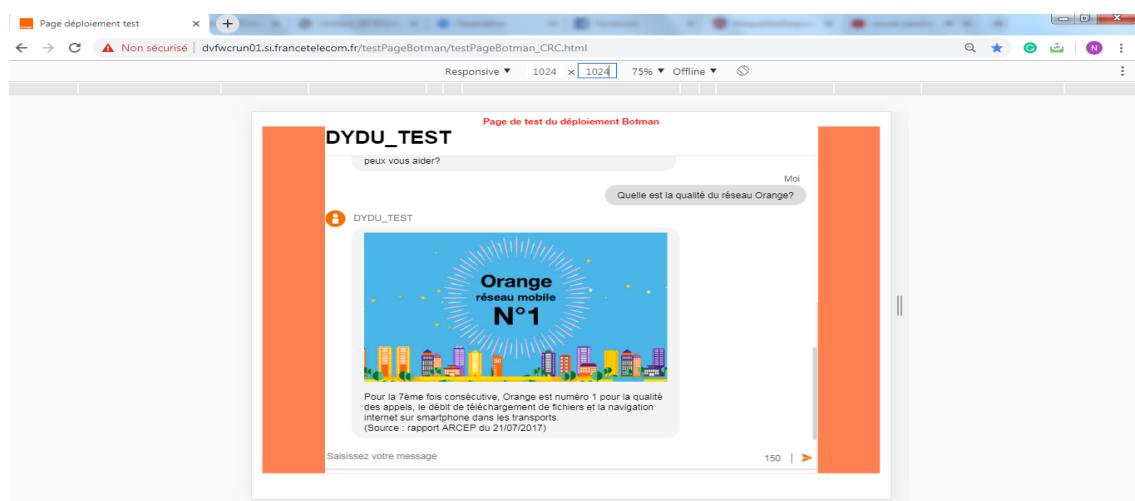


FIGURE 4.11 – Interface de taille 1024*1024

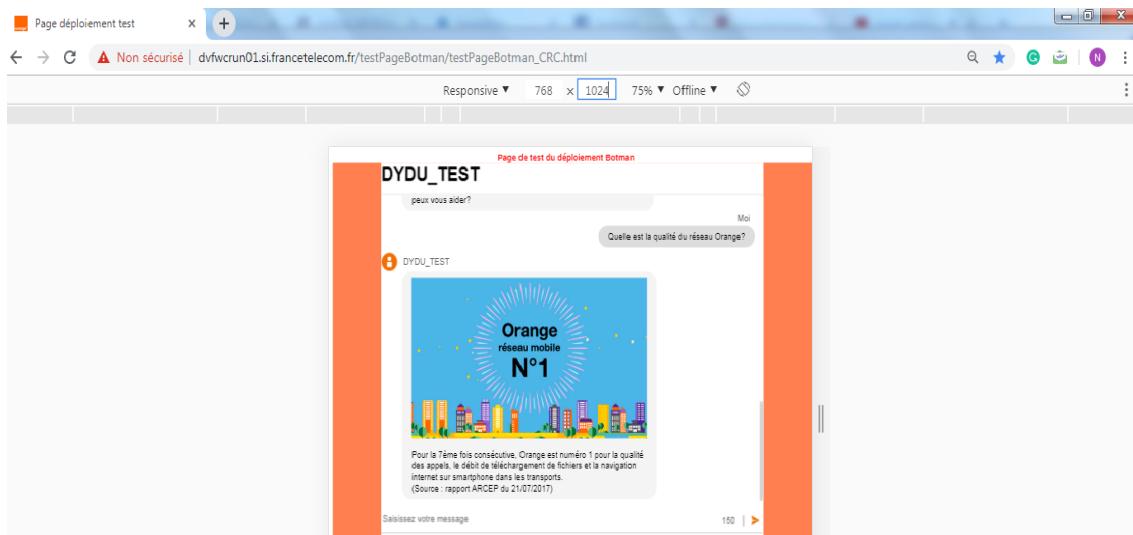


FIGURE 4.12 – Interface de taille 768*1024

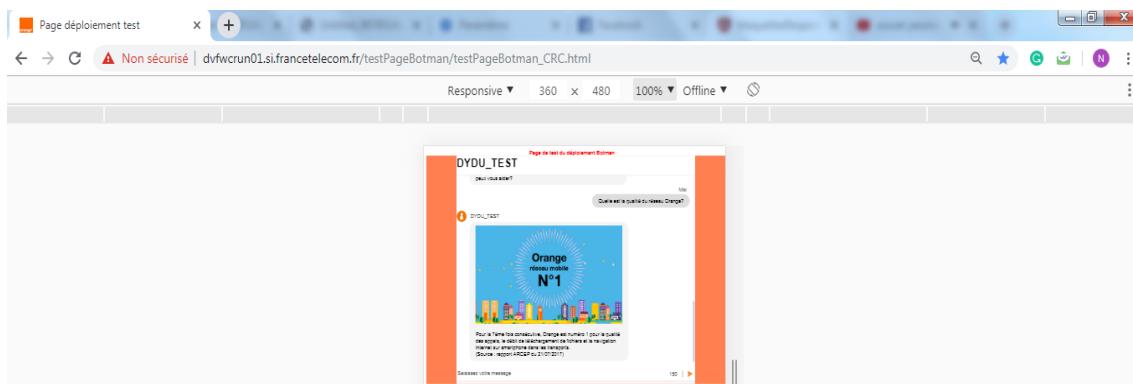


FIGURE 4.13 – Interface de taille 360*480

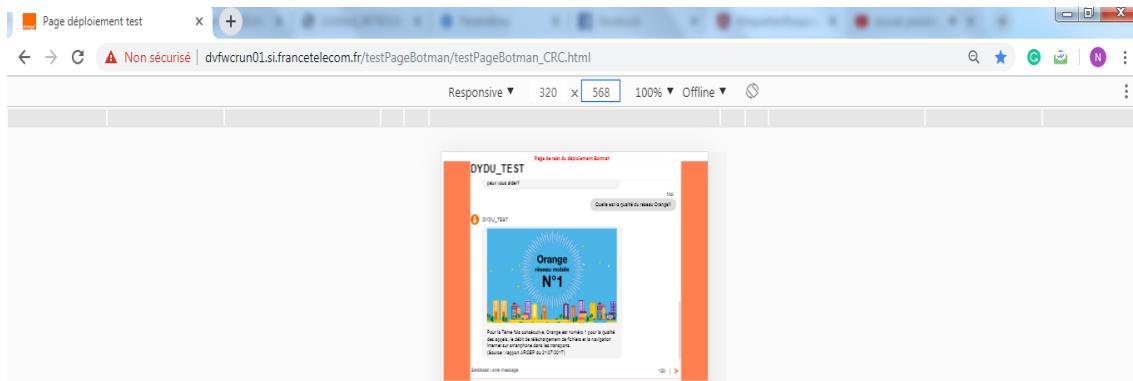


FIGURE 4.14 – Interface de taille 320*568

Conclusion

Ce deuxième sprint nous a permis d'effectuer la réingénierie de la partie frontend et backend du projet Botman et produire le premier livrable de notre projet. Le chapitre suivant sera dédié au sprint 3 qui a pour objectif la mise en place de l'authentification et la gestion des utilisateurs et des bots de l'application de suivi et de gestion Botman.

Chapitre 5

Sprint 3 : Application de suivi et de paramétrage de Botman : Authentification et gestion

Au cours de ce chapitre, qui représente le sprint 3, nous allons commencer la partie réalisation de l'application de suivi et de paramétrage de Botman. Ce sprint a pour objectif la mise en place de l'authentification, la gestion des utilisateurs ainsi que la gestion des bots. En premier lieu, nous allons présenter la spécification où nous analyserons les user story de ce sprint. Puis, nous allons présenter la partie conception en s'appuyant sur des diagrammes conceptuels. Nous allons clôturer ce chapitre par des captures explicatives relatives à la réalisation de ce sprint.

5.1 | sprint Backlog

Le tableau 5.1 présente le Sprint Backlog.

5	Authentification	5.1	En tant que utilisateur je dois m'authentifier pour accéder à mon compte au sein de l'application
6	Gestion des utilisateurs	6.1	En tant que administrateur je peux ajouter un utilisateur
		6.2	En tant que administrateur je peux consulter un utilisateur
		6.2	En tant que administrateur je peux modifier un utilisateur
		6.2	En tant que administrateur je peux supprimer un utilisateur
7	Gestion des bots	7.1	En tant que utilisateur je peux ajouter un bot
		7.2	En tant que utilisateur je peux consulter un bot
		7.3	En tant que utilisateur je peux modifier un bot
		7.4	En tant que utilisateur je peux supprimer un bot

TABLE 5.1 – Sprint Backlog

5.2 | Spécification

Dans cette section, nous détaillons les user story de ce sprint en présentant les diagrammes de cas d'utilisation relatif suivi d'une description textuelle.

5.2.1 Diagramme de cas d'utilisation raffiné du cas «Gérer les utilisateurs»

La figure 5.1 représente le diagramme de cas d'utilisation raffiné du cas «Gérer les utilisateurs». Comme nous avons spécifié au début, la fonctionnalité «Gérer les utilisateurs» est réalisée que par l'administrateur.

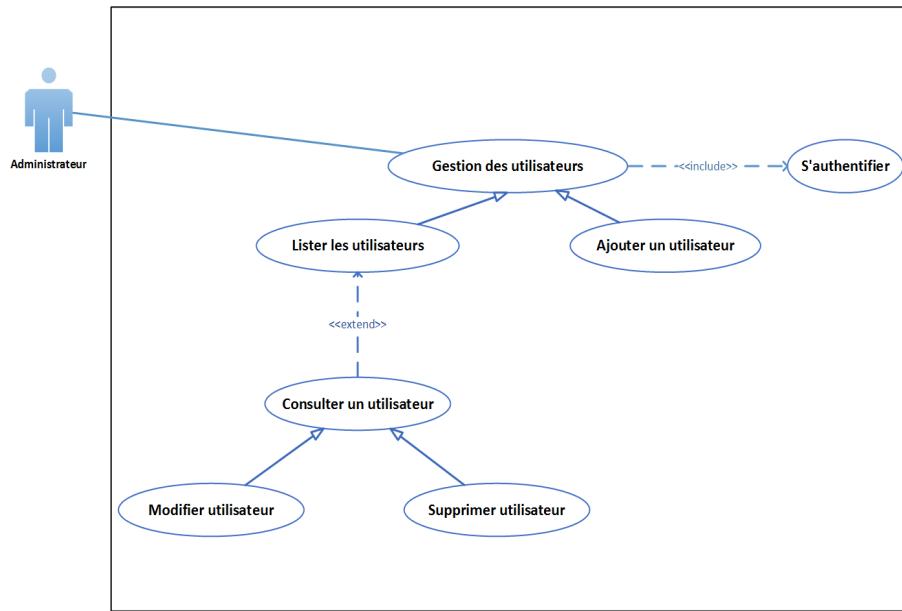


FIGURE 5.1 – Diagramme de cas d'utilisation raffiné du cas « Gérer les utilisateurs »

Description textuelle

Le tableau 5.1 définit une description textuelle du scénario de l'action «Ajouter un utilisateur» du cas d'utilisation «Gérer les utilisateurs».

Nom	— Ajouter un utilisateur
Acteur	— Administrateur
Précondition	— Administrateur authentifié
Postcondition	— Utilisateur ajouté
Scénario principal	<ul style="list-style-type: none"> — L'administrateur clique sur le bouton «Gestion des utilisateur» puis sur le bouton «Créer». — Il remplit le formulaire. — Il clique sur le bouton «Ajouter». — Le système enregistre l'utilisateur et met à jour la liste des utilisateur affichée.
Scénario alternatif	<ul style="list-style-type: none"> — S'il existe un ou plusieurs champs vides ou invalides le système affiche un message d'erreur.

TABLE 5.2 – Description textuelle du cas «Ajouter un utilisateur»

5.2.2 Diagramme de cas d'utilisation raffiné du cas « Gérer les bots »

La figure 5.2 représente le diagramme de cas d'utilisation raffiné du cas « Gérer les bots».

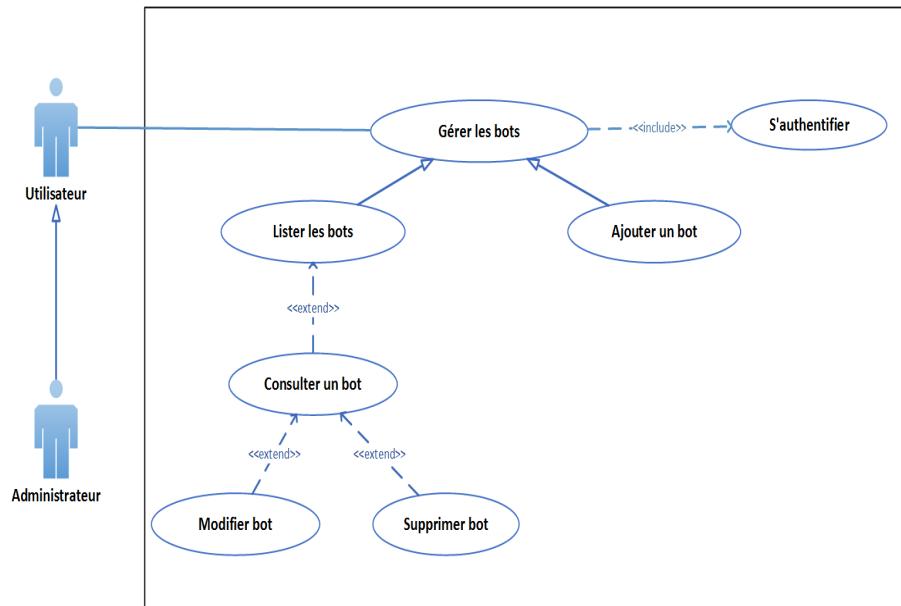


FIGURE 5.2 – Diagramme de cas d'utilisation raffiné du cas « Gérer les bots »

Description textuelle

Le tableau 5.2 définit une description textuelle du scénario de l'action «Modifier un Bot» du cas d'utilisation «Gérer les Bots».

Nom	— Modifier un bot
Acteur	— Utilisateur
Précondition	— Utilisateur authentifié
Postcondition	— Bot modifié
Scénario principal	<ul style="list-style-type: none"> — L'utilisateur choisit le bot à modifier en le sélectionnant. — Le système affiche un formulaire contenant les informations du bot à modifier. — L'utilisateur modifie les informations liées au bot. — Il confirme ses modifications en cliquant sur le bouton «Modifier». — Le système affiche un message de validation.
Scénario alternatif	<ul style="list-style-type: none"> — Si il existe un ou plusieurs champs vides ou invalides le système affiche un message d'erreur.

TABLE 5.3 – Description textuelle du cas «Modifier un bot»

5.3 | Conception

5.3.1 Diagramme de séquence objet du cas «Ajouter un Bot»

Le diagramme de séquence objet permet d'illustrer les interactions entre les différents objets de notre système. La figure 5.3 présente le diagramme de séquence objet du cas «Ajouter un Bot».

CHAPITRE 5. SPRINT 3 : APPLICATION DE SUIVI ET DE PARAMÉTRAGE DE BOTMAN : AUTHENTIFICATION ET GESTION

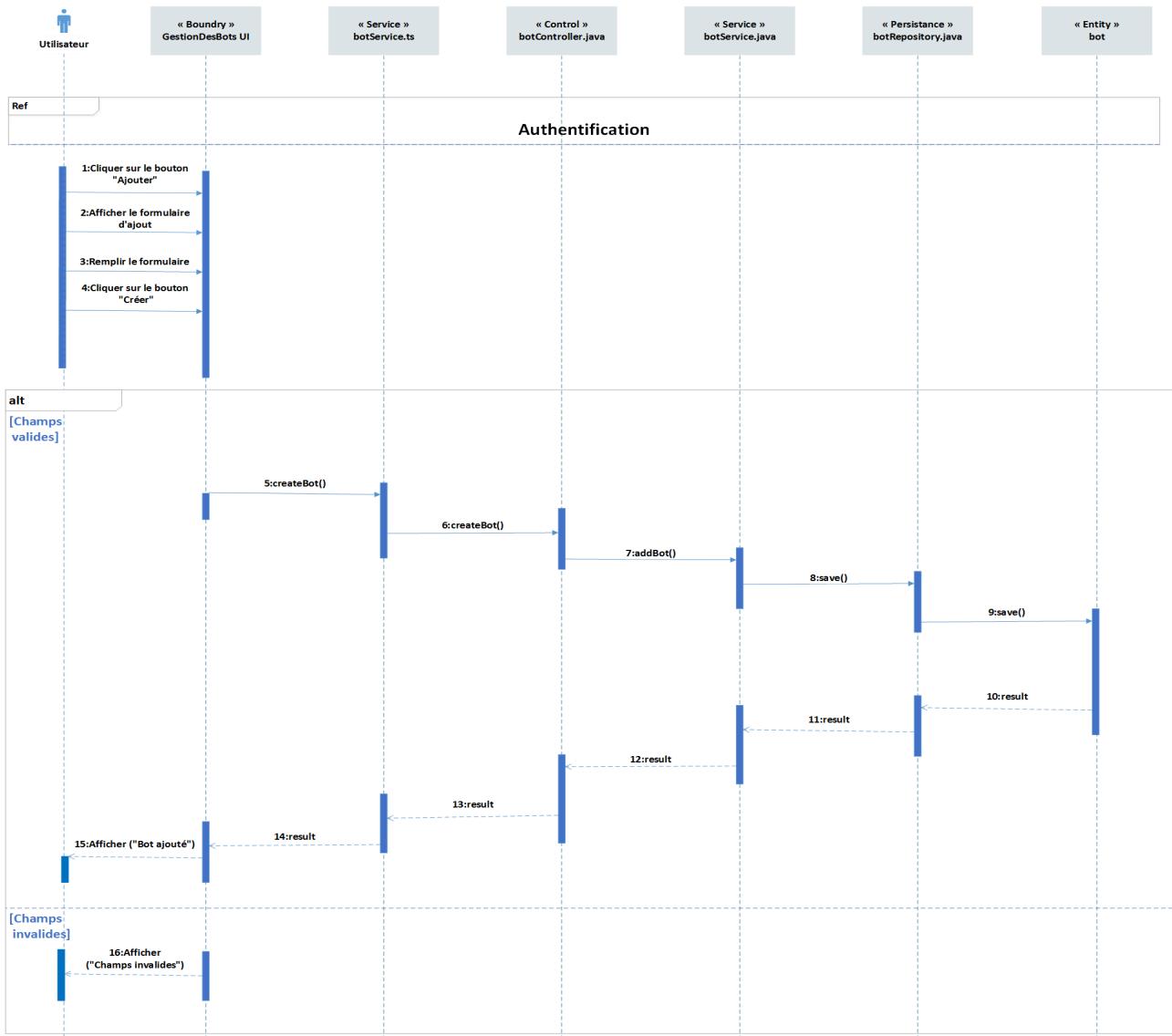


FIGURE 5.3 – Diagramme de séquence objet du cas «Ajouter un Bot»

5.3.2 Diagramme d'activité du cas «Ajouter un Bot»

Le diagramme d'activité modélise l'enchaînement d'une fonctionnalités ainsi que ses conditions d'exécution.

La figure 5.4 illustre le digramme d'activité du cas «Ajouter un Bot».

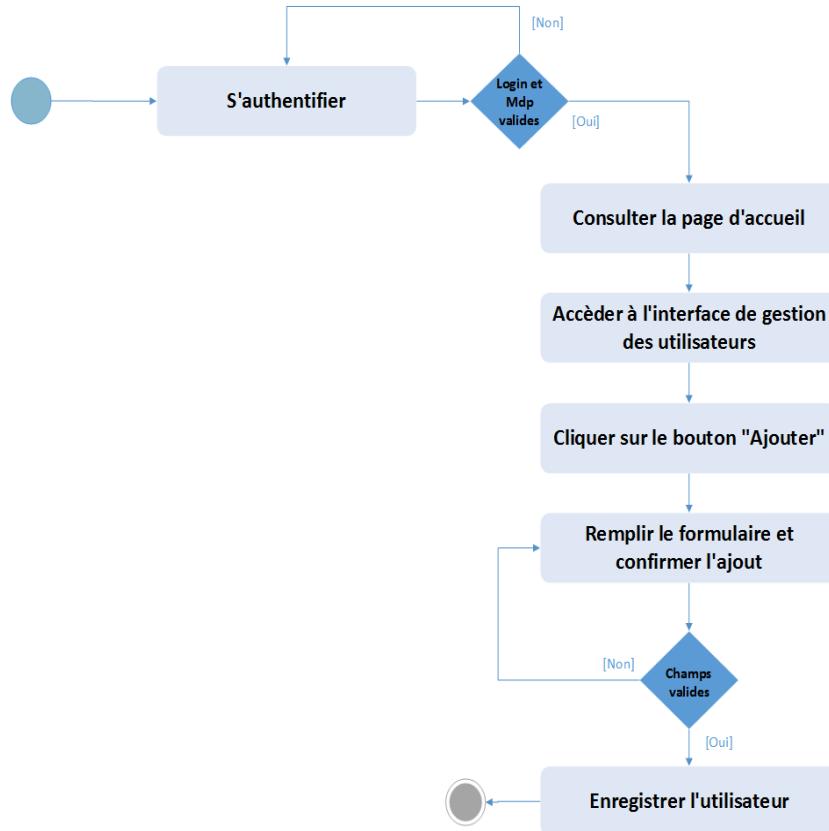


FIGURE 5.4 – Diagramme d'activité du cas «Ajouter un Bot»

5.3.3 Diagramme de classes de conception

Le diagramme de classes de conception illustre les interactions des classes du système impliquées dans ce sprint.

Ce diagramme est représenté par la figure 5.5.

Description

- **Utilisateur** : représente l'entité qui comporte les données relatives à l'utilisateur de notre application. Chaque utilisateur possède un seul rôle.
- **Rôle** : cette entité représente les données relatives au rôle.
- **Bot** : c'est l'entité qui regroupe les informations concernant un bot.
- **CommitmentBubble** : cette entité représente les informations relatives aux bulles d'engagement d'un bot.

- **ErrorMsg** : cette entité comporte les messages d'erreurs relatifs à un bot.
- **Tag** : cette entité permet de représenter les tags d'un bot.

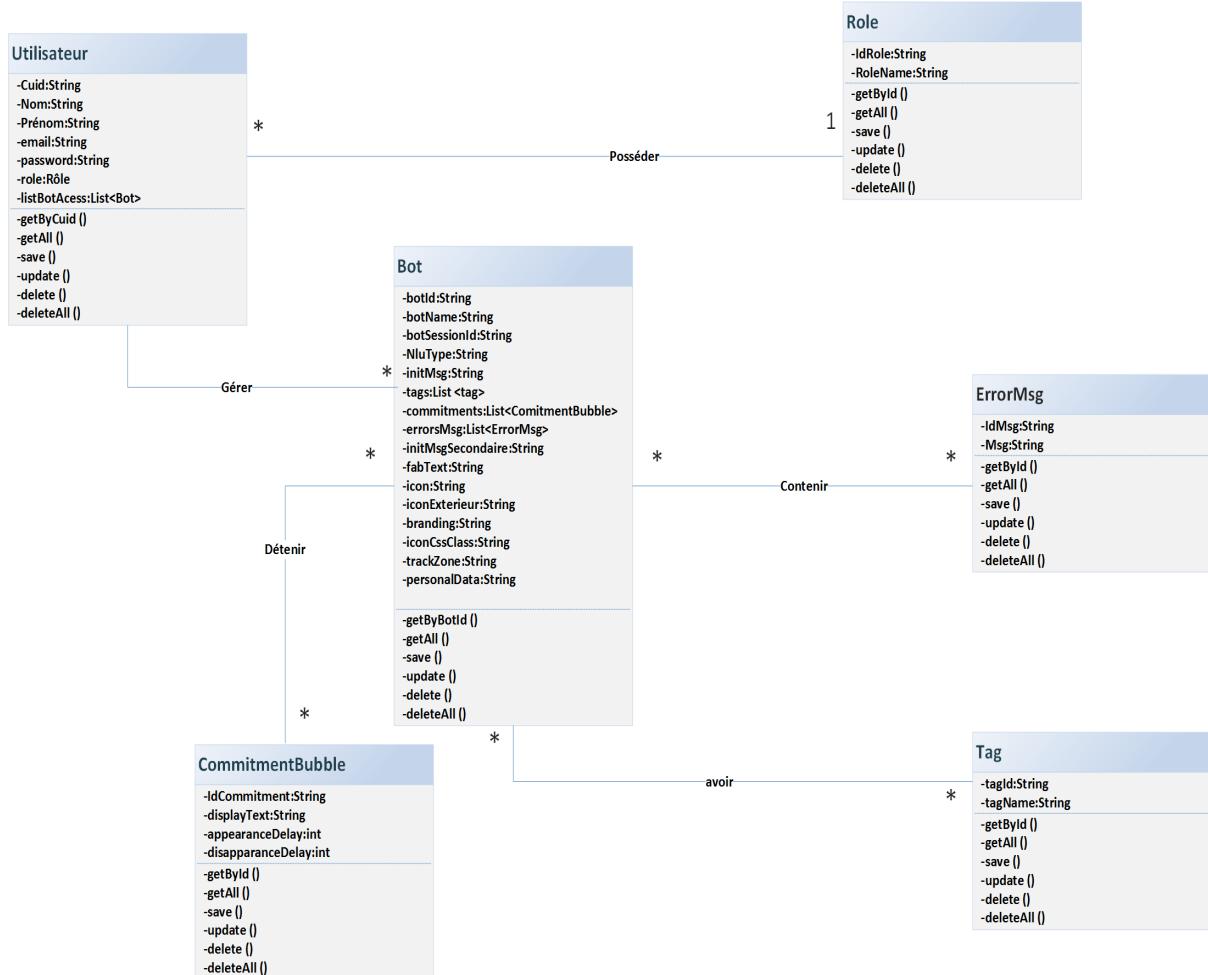


FIGURE 5.5 – Diagramme de classes de conception du sprint 3

5.4 | Réalisation

Durant cette section, nous allons présenter quelques interfaces illustrant des fonctionnalités de notre application.

La figure 5.6 représente l'interface d'authentification de l'application.

CHAPITRE 5. SPRINT 3 : APPLICATION DE SUIVI ET DE PARAMÉTRAGE DE BOTMAN : AUTHENTIFICATION ET GESTION

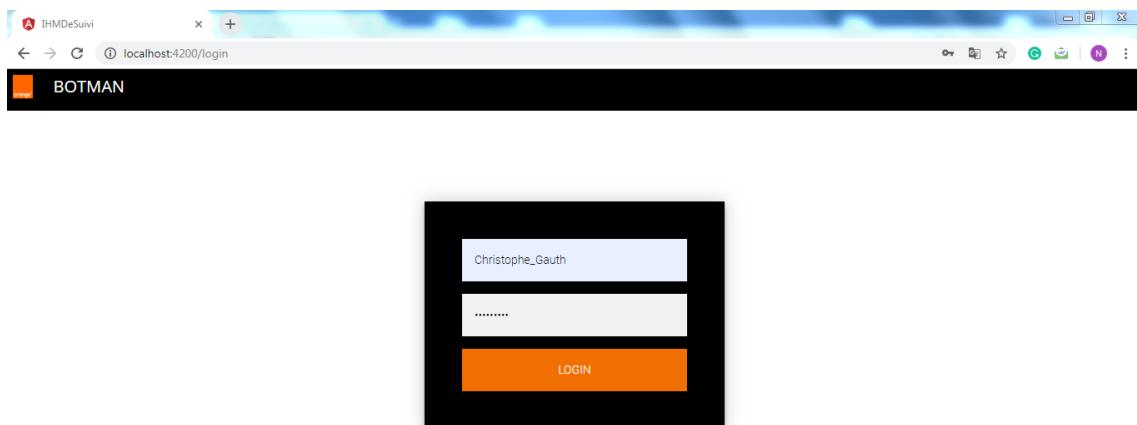


FIGURE 5.6 – Interface d’authentification de l’application

la figure 5.7 représente l’interface d’accueil de l’administrateur de notre application. À partir de cette interface, l’administrateur peut accéder à toutes les fonctionnalités offertes par notre système.

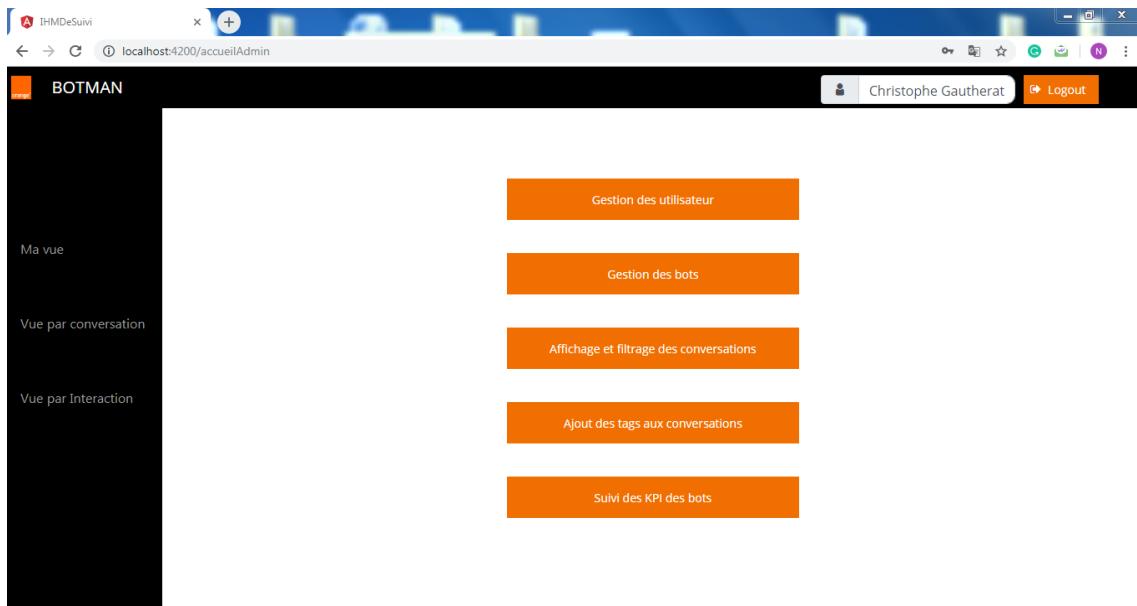


FIGURE 5.7 – Interface d’accueil de l’administrateur

CHAPITRE 5. SPRINT 3 : APPLICATION DE SUIVI ET DE PARAMÉTRAGE DE BOTMAN : AUTHENTIFICATION ET GESTION

L’interface présenté par la figure 5.8 apparaît suite à un clique sur le bouton «Gestion des utilisateurs». L’administrateur peut ainsi ajouter, supprimer et modifier les utilisateurs.

The screenshot shows a web application window titled 'BOTMAN'. At the top right, there is a user session indicator for 'Christophe Gautherat' and a 'Logout' button. Below the header, a sidebar on the left lists navigation options: 'Ma vue', 'Vue par conversation', and 'Vue par Interaction'. The main content area is titled 'Liste des utilisateurs' and displays a table with the following data:

Cuid	Mdp	Nom	Prénom	Email	ListBotAccess	Rôle
NVDQ1941	ben gaid hassine	nouha	nouha.bgh@hotmail.com	▼	user
FQAD2047	Hedhili	Ahmed	ahmed.hedhili@sofrecom.com	▼	user
MGHJ1598	Zaiem	Khouloud	khouloud.zaiem@sofrecom.com	▼	user
FKDF6589	Louhichi	Takoua	takous.louhichi@sofrecom.com	▼	user
GAUTH6985	Gautherat	Christophe	christophe.gautherat@orange.com	▼	admin

At the bottom right of the table area is a red '+ Créer' button. Below the table, there are navigation arrows and a page number '1'.

FIGURE 5.8 – Interface gestion des utilisateurs

La figure 5.9 représente l’interface qui apparaît suite à la sélection d’un utilisateur à partir de la liste affichée. Cette interface permet à l’administrateur de modifier ou supprimer l’utilisateur sélectionné.

The screenshot shows the same web application window as Figure 5.8. A modal dialog box is open over the user list, titled 'Utilisateur Détails'. This dialog contains the following fields for the user with Cuid 'NVDQ1941':

- Mot de passe: (password field)
- Nom: ben gaid hassine
- Prénom: nouha
- Email: nouha.bgh@hotmail.com
- Rôle: user
- Liste des bots accessibles: ERRORBOT X (dropdown menu)

At the bottom of the dialog are two buttons: a red '✓ Modifier' button and a black '✗ Supprimer' button. To the right of the dialog, the original user list table is visible, showing the other four users listed in Figure 5.8.

FIGURE 5.9 – Interface modifier/supprimer un utilisateur

CHAPITRE 5. SPRINT 3 : APPLICATION DE SUIVI ET DE PARAMÉTRAGE DE BOTMAN : AUTHENTIFICATION ET GESTION

La figure 5.10 illustre l'interface qui apparaît suite à un clique sur le bouton «Gestion des bots». L'utilisateur peut ainsi ajouter, supprimer et modifier les différents bots auxquels il a accès.

Liste des Bots						
BotId	Name	NLUType	InitMessage	Error Messages	Tags	track zone
ERRORBOT	ERRORBOT	WATSON	Bonjour \$cuid, je suis le bot WATSON aliment´ par Watson comment je peux vous aider?	D´acute;sol´, suite ` un problème interne, je n'arrive pas ` vous comprendre.		botCommerce
DYDU_TEST	DYDU_TEST	DYDU	Bonjour \$cuid, je suis le bot assistance aliment´ par DYDU comment je peux vous aider?	D´acute;sol´, suite ` un problème interne, je n'arrive pas ` vous comprendre.		test_botman
ATHRASA	Djingو	DYDU	Bonjour, je suis Djingo, le chatbot d'Orange. Je suis ` pour vous aider sur vos questions techniques. Au besoin, je pourrai vous mettre en relation avec un conseiller (entre 8h et 22h).	D´acute;sol´, suite ` un problème interne, je n'arrive pas ` vous comprendre.		djingو_AT
Watson	Djingو	Watson	Bonjour \$cuid, je suis le bot WATSON aliment´ par Watson comment je peux vous aider?	D´acute;sol´, suite ` un problème interne, je n'arrive pas ` vous comprendre.		test_watson
			Bonjour \$userName ! Je m'appelle Djingo et je			

FIGURE 5.10 – Interface gestion des bots

De même, l'utilisateur peut ajouter un bot à partir de l'interface de «Gestion des bots» comme le montre la figure 5.11.

'."/>

Bot Details

icon_externer	<input type="text"/>					
branding	<input type="text"/>					
fabText	<input type="text"/>					
iconCssClass	<input type="text"/>					
errorMessages	<input type="text"/>					
trackzone	<input type="text"/>					
PROTO_RASA	personalData	<input type="text"/>				

Ajouter

>Géacute;acute;rer votre profil Q/R

FIGURE 5.11 – Interface d'ajout d'un bot

5.5 | Test et revue du sprint

Après avoir terminé la réalisation de ce sprint, nous avons eu une réunion "rétrospective de sprint" avec l'équipe scrum formée par notre "product owner" monsieur Christophe GAUTHEART et notre "scrum Master" khouloud zaiem afin de valider le travail réalisé. Si les objectifs sont atteints, le "product owner" définit les buts du sprint suivant.

Conclusion

Ce troisième sprint nous a permis de produire le premier livrable de l'application de suivi et de paramétrage de Botman. Nous avons en premier lieu passé par une analyse et conception de nos besoins. En second lieu, nous avons illustrée la partie réalisation avec quelques interfaces de notre application.

Le chapitre suivant présente le dernier sprint de notre projet.

Chapitre 6

Sprint 4 : Application de suivi et de paramétrage de Botman : Performance des Bots

Ce chapitre présente le dernier sprint de notre projet qui a pour objectif le suivi et le filtrage des conversations de production, le suivi des KPI des bots ainsi que l'annotation des conversations en y ajoutant des tags libres. Similaire au chapitre qui le précède, nous allons commencer ce chapitre par présentons la spécification des user story. Puis nous allons entamer la partie conception en présentant quelques diagrammes UML. Finalement, nous allons illustrer la réalisation de ce sprint par des captures explicatives.

6.1 | Sprint Backlog

Le tableau 6.1 présente le Backlog Produit.

8	Filtrer les conversations selon plusieurs critères	8.1	En tant que utilisateur je peux effectuer une recherche sur les conversations selon plusieurs critères
		8.2	En tant que utilisateur je peux consulter les détails d'une conversation

TABLE 6.1 – Backlog Produit

9	Ajout des tags aux conversations	9.1	En tant que utilisateur je peux ajouter des tags aux conversations
10	Suivi des KPIs des bots	10.1	En tant que utilisateur je peux consulter le nombre des messages incompris par les bots
		10.2	En tant que utilisateur je peux consulter le nombre de conversations contenant des erreurs techniques
		10.2	En tant que utilisateur je peux consulter le nombre de conversations contenant des erreurs fonctionnelles
		10.2	En tant que utilisateur je peux consulter le nombre de conversations par jours pendant les 15 derniers jours

TABLE 6.2 – Backlog Produit

6.2 | Spécification

Dans cette section, nous présentons le diagramme de cas d'utilisation raffiné relatif à ce sprint, puis nous détaillons un cas d'utilisation en définissant sa description textuelle.

6.2.1 Diagramme de cas d'utilisation raffiné

La figure 6.1 représente le diagramme de cas d'utilisation raffiné relatif à ce sprint.

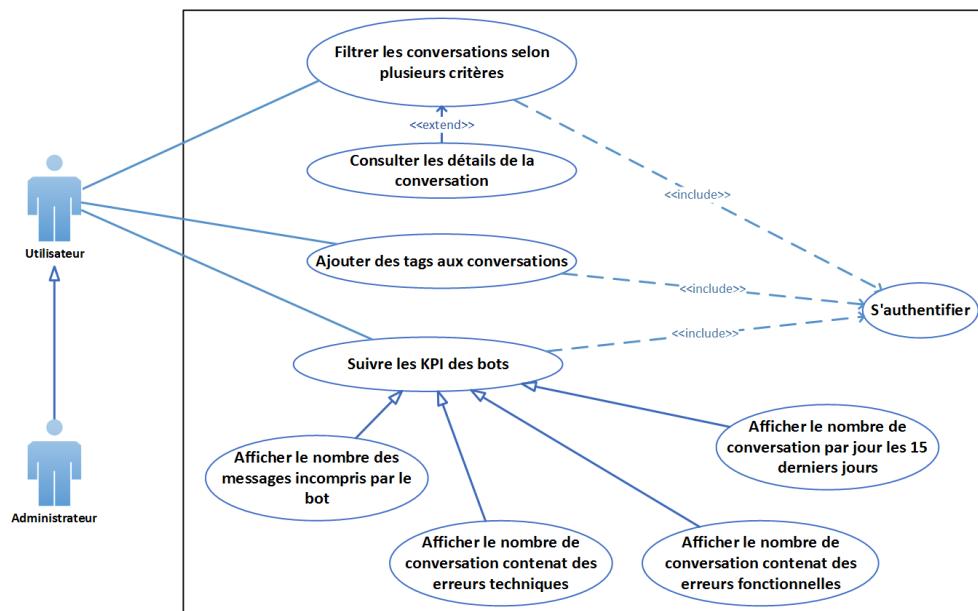


FIGURE 6.1 – Diagramme de cas d'utilisation raffiné du sprint 4

Description textuelle du cas d'utilisation «Filtrer les conversations selon plusieurs critères»

Le tableau 6.1 définit une description textuelle du scénario du cas d'utilisation «Filtrer les conversations selon plusieurs critères».

Nom	— Filtrer les conversations selon plusieurs critères
Acteur	— Utilisateur
Précondition	— Utilisateur authentifié
Postcondition	— Conversations filtrées
Scénario principal	<ul style="list-style-type: none"> — L'utilisateur choisit les filtres appropriés à sa recherche. — Le système cherche et affiche les Ids des conversations qui répondent à la recherche . — L'utilisateur peut afficher les détails d'une conversation en sélectionnant son Id.
Scénario alternatif	<ul style="list-style-type: none"> — S'il n'existe pas des conversations qui correspondent à la recherche le système affiche un message d'erreur.

TABLE 6.3 – Description textuelle du cas «Filtrer les conversations selon plusieurs critères»

6.3 | Conception

6.3.1 Diagramme de séquence objet du cas «Filtrer les conversations selon plusieurs critères»

La figure 6.2 présente le diagramme de séquence objet du cas «Filtrer les conversations selon plusieurs critères» qui illustre la séquence d'actions se produisent lorsque un utilisateur veut filtrer les conversations.

CHAPITRE 6. SPRINT 4 : APPLICATION DE SUIVI ET DE PARAMÉTRAGE DE BOTMAN :
PERFORMANCE DES BOTS

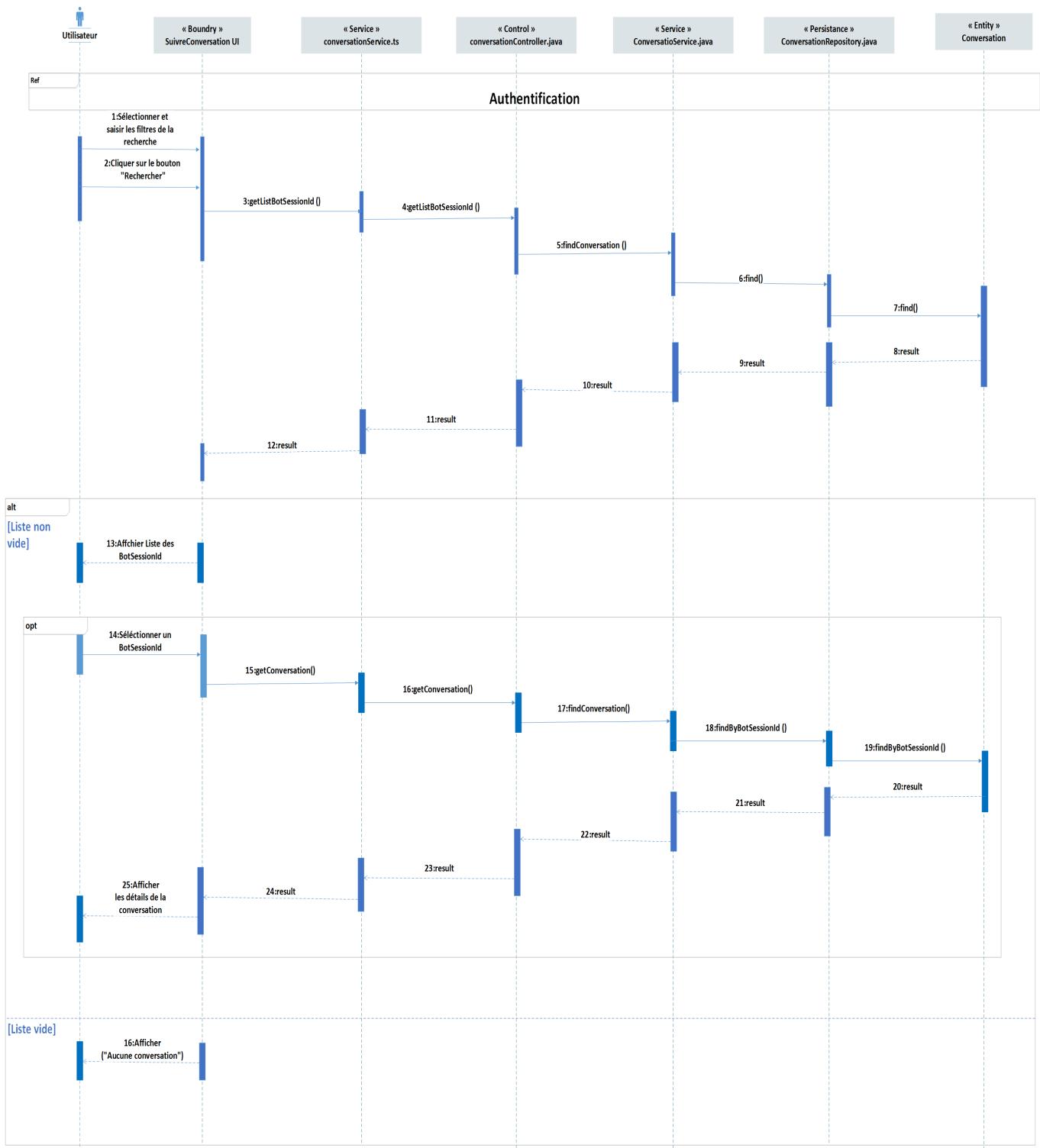


FIGURE 6.2 – Diagramme de séquence objet du cas «Filtrer les conversations selon plusieurs critères»

6.3.2 Diagramme d'activité du cas «Filtrer les conversations selon plusieurs critères

La figure 6.4 illustre le diagramme d'activité du cas «Filtrer les conversations selon plusieurs critères».

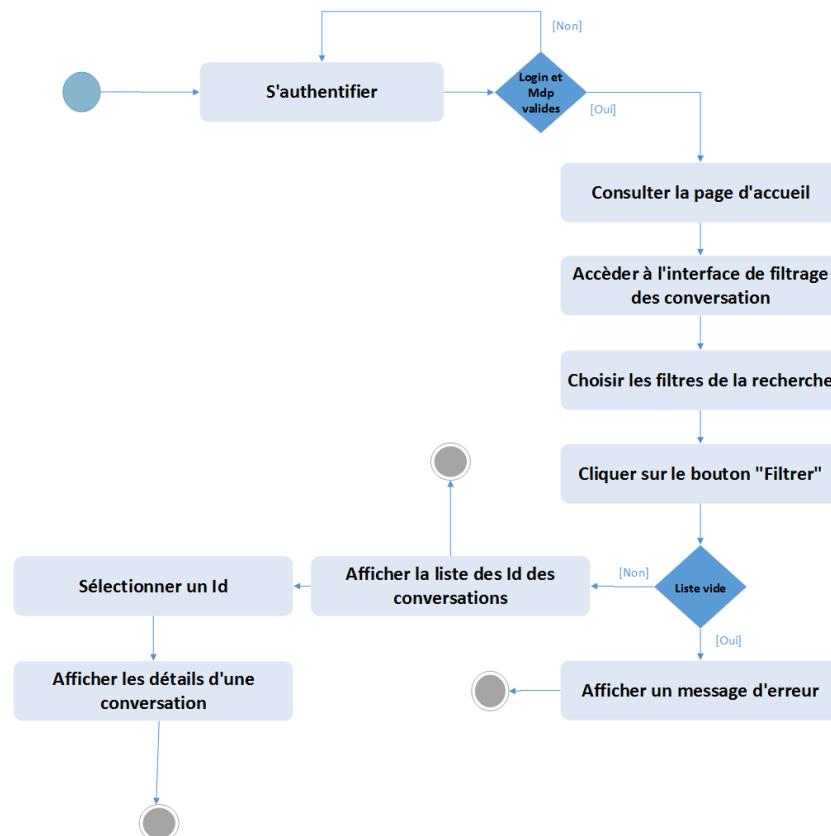


FIGURE 6.3 – Diagramme d'activité du cas «Filtrer les conversations selon plusieurs critères

6.3.3 Diagramme de classes de conception

La figure 6.4 représente le diagramme de classes de conception relatif à ce sprint.

Description

- **Conversation** : représente l'entité qui encapsule toutes les informations qui définissent une conversation dans notre projet.
- **Media** : Cette entité contient une liste de mediaNodes.
- **MediaNodes** : cette entité représente les messages des bots.

- **MediaTypeContainer** : c'est une énumération qui définit les différents types de medias.
- **MediaNodeAction** : c'est une énumération qui définit les différents types de medias-nodes.

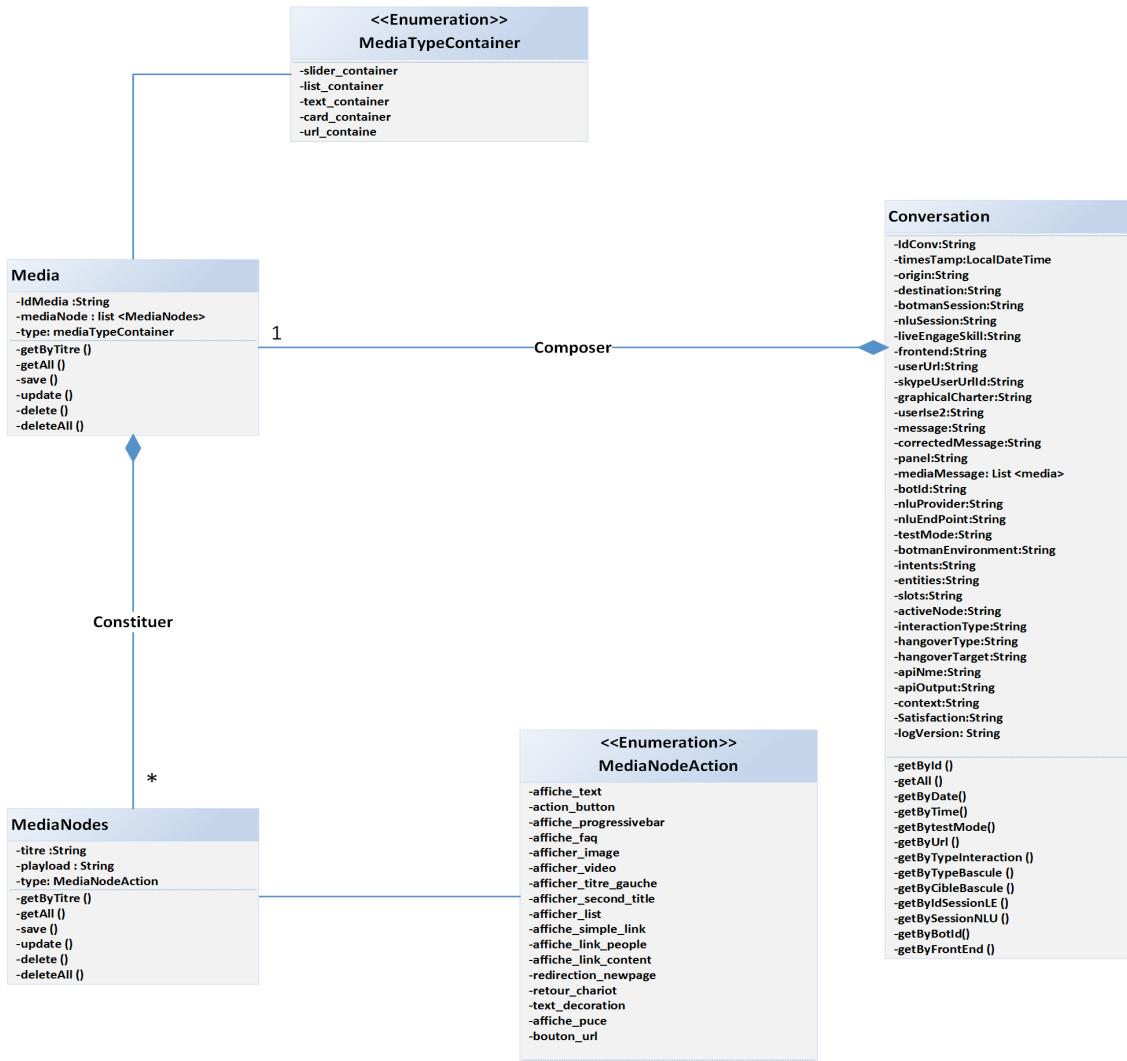


FIGURE 6.4 – Diagramme de classes de conception du sprint 2

6.4 | Réalisation

Dans cette section, nous présentons la phase de réalisation à l'aide des principales captures d'écran qui décrivent les fonctionnalités réalisées au cours de ce sprint.

la figure 6.5 représente l'interface à partir de laquelle l'utilisateur peut filtrer les conversations. Le bouton "Réinitialiser" permet de remettre les filtres à des valeurs par défaut. Le bouton "plus de filtres" permet d'afficher d'autres filtres de recherche.

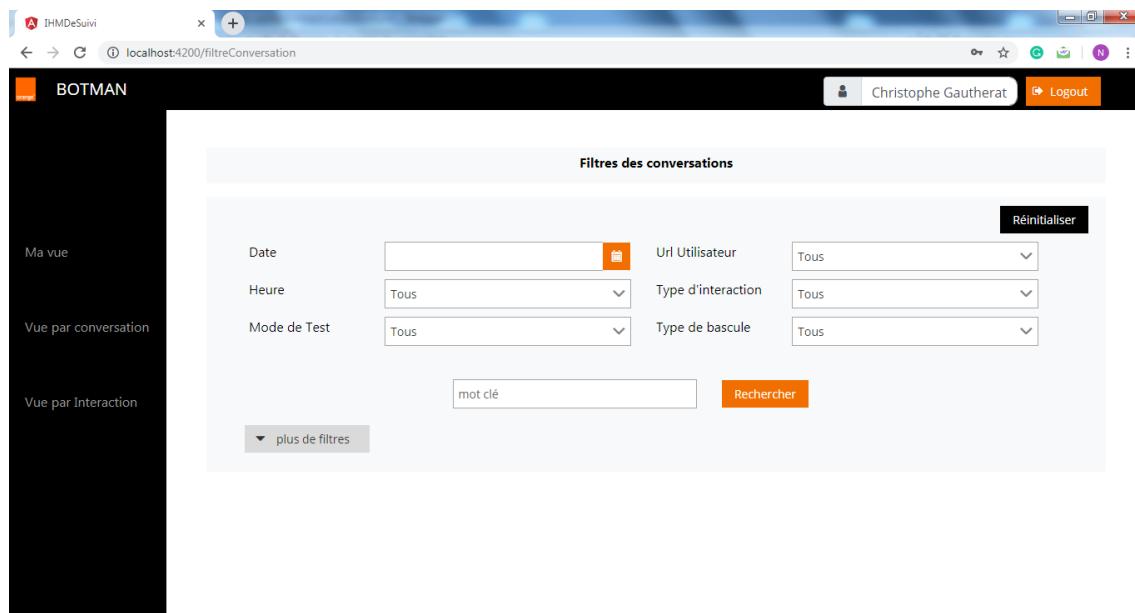


FIGURE 6.5 – Interface de filtrages des conversations

La figure 6.6 représente l'interface qui apparaît suite à un clique sur le bouton "Rechercher". Le système affiche la liste des Id des conversations qui répondent à la recherche.

CHAPITRE 6. SPRINT 4 : APPLICATION DE SUIVI ET DE PARAMÉTRAGE DE BOTMAN : PERFORMANCE DES BOTS

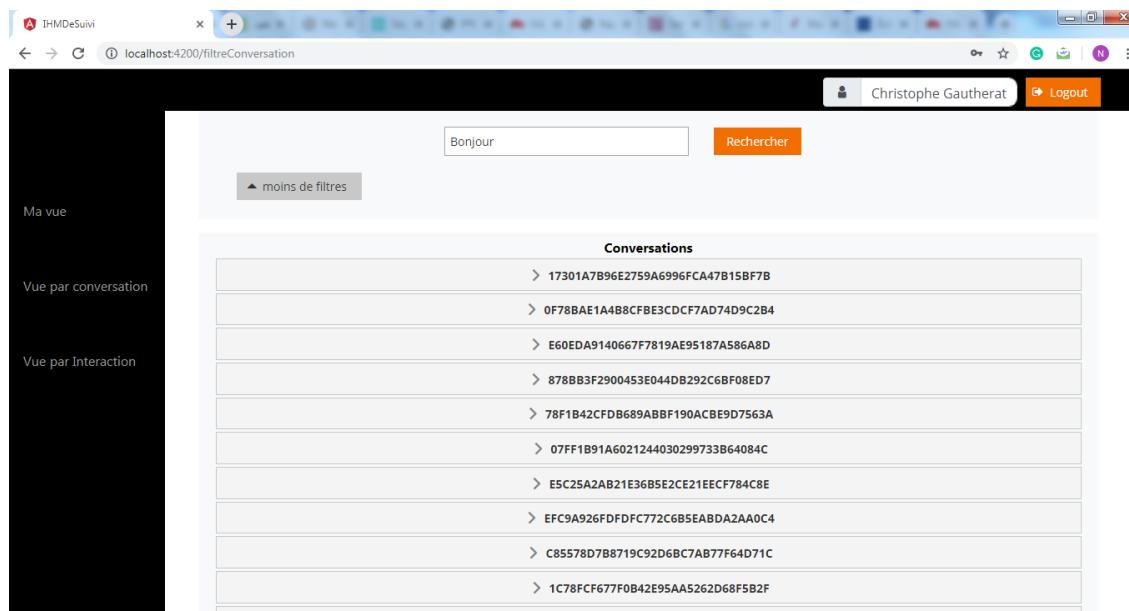


FIGURE 6.6 – Interface résultatat de recherche

L’interface présenté par la figure 6.7 apparaît lorsque l’utilisateur sélectionne un Id des conversations. Cette fonctionnalité permet d’afficher les détails de l’échange entre le bot et l’utilisateur.

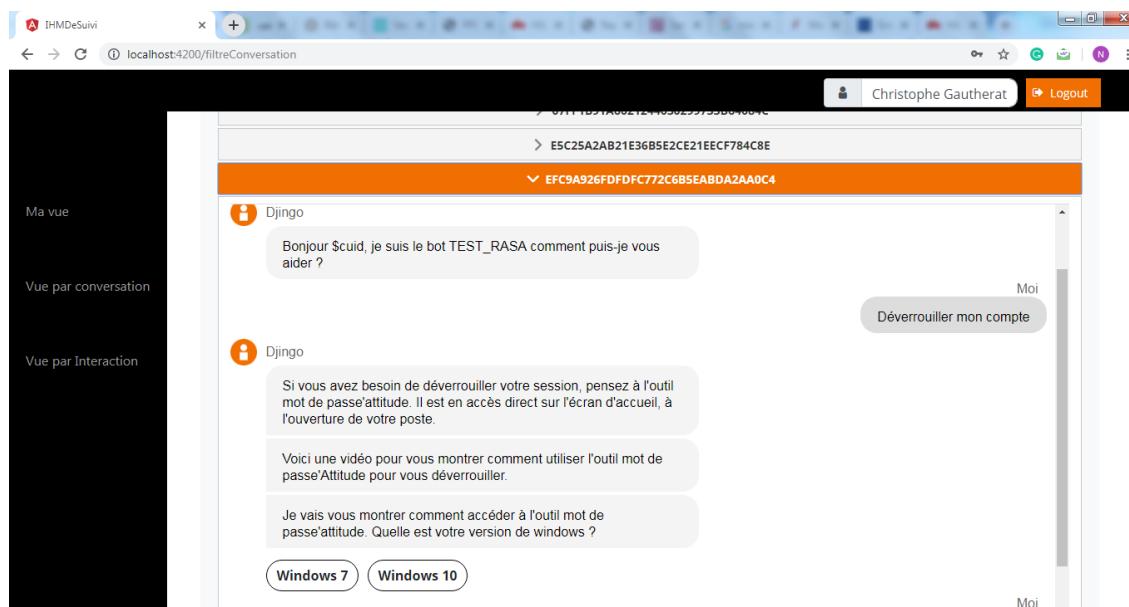


FIGURE 6.7 – Interface d'affichage des détails d'une conversation sélectionné

CHAPITRE 6. SPRINT 4 : APPLICATION DE SUIVI ET DE PARAMÉTRAGE DE BOTMAN : PERFORMANCE DES BOTS

La figure 6.8 illustre l’interface qui apparaît suite à un clique sur le bouton «Ajout des tags aux conversations». L’utilisateur peut ainsi sélectionner l’Id de la conversation concernée et le tag approprié.

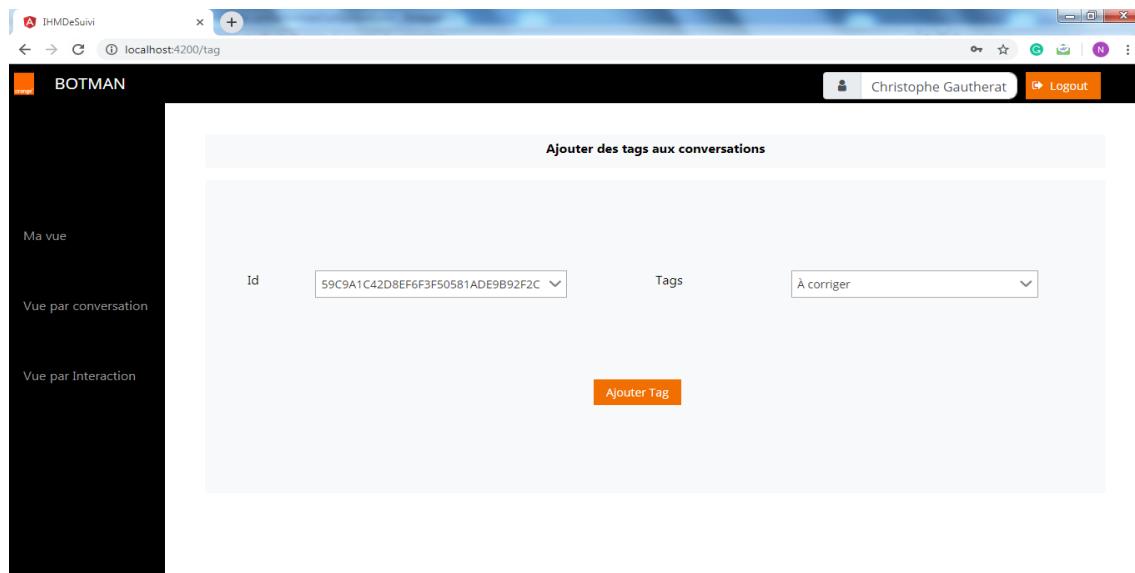


FIGURE 6.8 – Interface d’ajout des tags

La figure 6.9 présente les KPI affichés à l’utilisateur suite à un clique sur le bouton «Suivi des KPI des bots». Cette fonctionnalité permet de calculer et afficher le nombre et le pourcentage des messages incompris par le bot, les conversations contenant des erreurs techniques et celle contenant des erreurs fonctionnelles. Aussi, cette interface affiche un diagramme en bâtons qui présente le nombre de conversation par jour les 15 derniers jours.

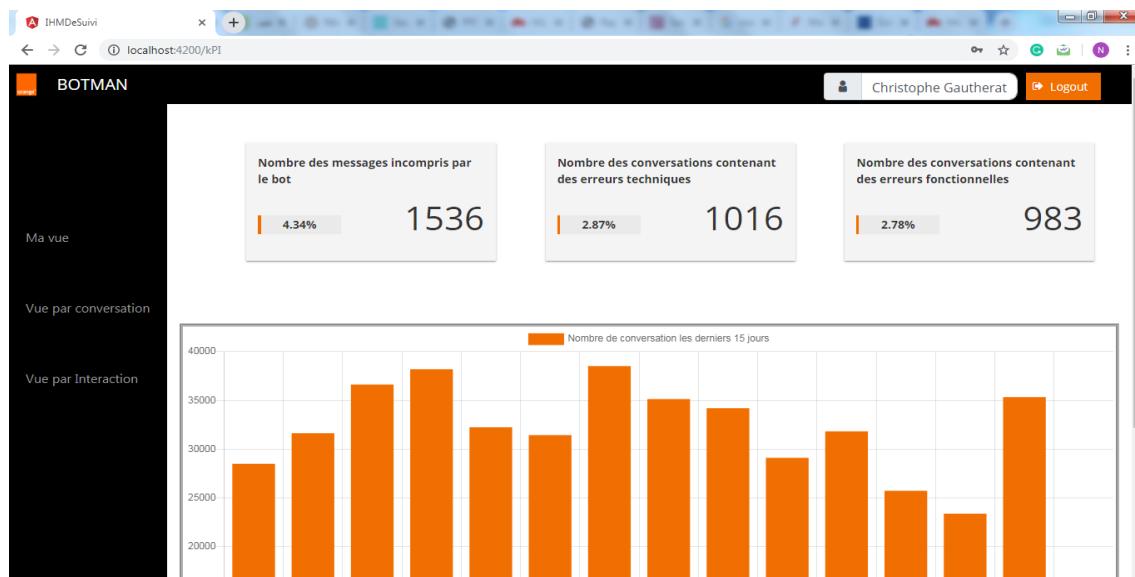


FIGURE 6.9 – Interface de suivi des KPI

6.5 | Test et revue du sprint

Comme le sprint précédent, après avoir terminé la phase de réalisation, nous avons eu une réunion "rétrospective de sprint" avec l'équipe scrum. Comme tous les objectifs sont atteints, le "product owner" Monsieur GAUTHEART a validé le dernier sprint de notre projet.

Conclusion

Par ce chapitre, nous clôturons la réalisation du sprint 4 qui représente le dernier sprint de notre projet. Ce sprint nous a permis de se focaliser sur les conversations pour pouvoir principalement mesurer les performances des divers bots.

Conclusion Générale

La satisfaction de la clientèle par le biais des canaux numériques présente un grand problème pour les entreprises. En effet, les entreprises recherchent toujours des retours positifs de leurs clients pour réussir sur le marché en ligne.

Ces dernières années, les entreprises informatiques ont commencé à investir dans des agents virtuels, également appelés chatbots. Ces agents aident les entreprises à mieux s'approcher de leurs clients grâce à la messagerie en langage naturel.

Les chatbots ont connu des avancées notables ces dernières années grâce au développement croissant de l'intelligence artificielle, de l'apprentissage automatique et de la compréhension du langage naturel. Dans ce domaine, la concurrence entre les entreprises s'est accrue pour créer la solution la plus robuste pour la conception et le développement de chatbots.

Notre projet s'inscrit dans cette perspective, il est divisé en trois parties principales : la première consiste à la réingénierie de la partie frontend qui comporte la mise en place d'une fenêtre de webchat en pleine page et responsive développée en Angular et Typescript, ainsi que la réingénierie d'un module backend du projet botman. La seconde partie consiste à créer une application de suivi et de paramétrage du projet Botman. La troisième partie consiste à intégrer les deux développements dans une chaîne CI/CD.

Dans ce rapport, nous avons détaillé les différentes étapes de la réalisation de ce projet en six chapitres. Nous avons consacré le premier chapitre à présenter le contexte général, académique et professionnel, à faire une étude et critique de l'existant et à présenter notre méthodologie de travail adoptée ainsi que les outils utilisés pour la gestion de notre projet. Dans le deuxième chapitre, nous avons présenté le sprint 0 dans lequel nous avons commencé par l'expression des besoins. Ensuite nous avons présenté l'architecture de l'application de suivi de paramétrage de Botman. Aussi, nous avons consacré une partie pour la spécification des besoins. Nous avons clôturé ce sprint par définir le backlog produit et la planification des sprints et présenter l'environnement de travail. Nous avons consacré le chapitre suivant pour le sprint 1 qui a pour objectif la mise en place d'une chaîne CI/CD. Dans le quatrième chapitre, nous avons présenté le sprint 2 qui a pour objectif la réingénierie du projet Botman. Le cinquième chapitre a été consacré pour le sprint 3 dans lequel nous avons développé la première partie de l'application de suivi et de paramétrage de Botman.

CHAPITRE 6. SPRINT 4 : APPLICATION DE SUIVI ET DE PARAMÉTRAGE DE BOTMAN : PERFORMANCE DES BOTS

Le dernier chapitre présente le dernier sprint dans lequel nous avons développé les fonctionnalités permettant le suivi de performance des bots.

Indépendamment des contraintes techniques et des défis rencontrés, nous avons atteint les objectifs et nous avons répondu aux exigences du projet. Mais, dans les étapes ultérieures, nous avons l'intention d'améliorer le suivi de performance des bots et d'ajouter plus de KPI pour avoir une vue plus approfondie sur les bots. Ceci va permettre d'améliorer le rendement et la qualité des réponses.

Notre stage chez Sofrecom a été une expérience extrêmement enrichissante et intéressante. Il a été pleinement instructif, tant sur le plan académique que professionnel. Nous avons eu le privilège de travailler avec une équipe collaborative et hautement qualifiée.

Netographie

- [1] <https://en.wikipedia.org/wiki/Chatbot> (consulté le 15/03/2019)
- [2] <https://www.sofrecom.com/fr> (consulté le 15/03/2019)
- [3] <https://www.sofrecom.com/fr/business-sectors> (consulté le 15/03/2019)
- [4] <https://www.educba.com/websocket-vs-rest/> (consulté le 15/03/2019)
- [5] <https://www.codeur.com/blog/analytics-chatbot/> (consulté le 04/04/2019)
- [6] <https://www.cprime.com/resources/what-is-agile-what-is-scrum/>
(consulté le 16/04/2019)
- [7] <https://www.qualitystreet.fr/2010/01/13/manifeste-agile/> (consulté le 16/04/2019)
- [8] <https://agiliste.fr/introduction-methodes-agiles/> (consulté le 16/04/2019)
- [9] <https://en.wikipedia.org/wiki/Git> (consulté le 19/04/2019)
- [10] <https://www.jinfonet.com/resources/bi-defined/3-tier-architecture-complete-overview/> (consulté le 21/04/2019)
- [11] <https://fr.slideshare.net/mohamedyoussfi9/support-de-cours-jsf-2-premire-partie>
(consulté le 22/04/2019)
- [12] <https://github.com/Microsoft/vscode> (consulté le 24/04/2019)
- [13] MongoDB <https://www.mongodb.com/what-is-mongodb> (consulté le 24/04/2019)
- [14] <https://aws.amazon.com/fr/redis/> (consulté le 24/04/2019)
- [15] <https://maven.apache.org/> (consulté le 24/04/2019)
- [16] <https://angular.io/> (consulté le 24/04/2019)
- [17] <https://spring.io/projects/spring-boot> (consulté le 24/04/2019)
- [18] <https://www.thoughtco.com/what-is-java-2034117> (consulté le 24/04/2019)

- [19] https ://www.wakefly.com/blog/what-is-typescript-and-why-should-you-use-it/ (consulté le 24/04/2019)
- [20] https ://sass-lang.com/ (consulté le 24/04/2019)
- [21] https ://www.fullipcom.com/extranet/announcements/view/2018-05 (consulté 01.05.2019)
- [22] https ://putaindecode.io/articles/qu-est-ce-que-l-integration-continue/ (consulté 01.05.2019)
- [23] https ://aws.amazon.com/fr/devops/continuous-integration/ (consulté 01.05.2019)
- [24] https ://www.lemagit.fr/definition/Deploiement-continu (consulté 01.05.2019)
- [25] https ://assessment-tools.ca.com/tools/continuous-delivery-tools/fr/source-control-management-tools (consulté 01.05.2019)
- [26] https ://assessment-tools.ca.com/tools/continuous-delivery-tools/fr/continuous-integration (consulté 03.05.2019)
- [27] https ://www.supinfo.com/articles/single/6569-comparaison-outils-ci-cd-jenkins-gitlab-ci-buildbot-drone-concourse ?fbclid=IwAR1_8wCIjjXpuZm6iDOVZDdreV4niqXtZu17VysBTKuxhQHcmXjU7O1KVaQ (consulté 03.05.2019)
- [28] http ://www.pipobimbo.info/blog/techno/integration-continue/outil-de-qualimetrie/ (consulté 05.05.2019)
- [29] https ://www.supinfo.com/articles/single/6418-qu-est-ce-que-sonarqube (consulté 05.05.2019)
- [30] https ://assessment-tools.ca.com/tools/continuous-delivery-tools/fr/configuration-management-tools (consulté 05.05.2019)
- [31] https ://assessment-tools.ca.com/tools/continuous-delivery-tools/fr/configuration-management-tools/ansible (consulté 05.05.2019)
- [32] https ://medium.com/formcept/configuration-management-and-continuous-deployment-cd0892dce998 (consulté 05.05.2019)
- [33] https ://www.lebigdata.fr/docker-definition (consulté 05.05.2019)
- [34] https ://www.it-wars.com/posts/virtualisation/docker-swarm-par-lexemple/ (consulté 05.05.2019)

- [35] https://www.ibm.com/blogs/bluemix/?p=163311&preview=true&preview_id=163311
(consulté 07.05.2019)
- [36] <https://les-experts.tech/metabot-un-chatbot-pour-les-gouverner-tous/>
(consulté 07.05.2019)
- [37] <https://javainsider.wordpress.com/tag/jms-with-activemq-sample-example/>
(consulté 09.05.2019)
- [38] <https://www.supinfo.com/articles/single/3470-rabbitmq> (consulté 09.05.2019)
- [39] <https://openclassrooms.com/fr/courses/219329-les-services-web> (consulté 09.05.2019)
- [40] <https://www.lemagit.fr/conseil/SOAP-ou-REST-comment-bien-choisir>
(consulté 09.05.2019)
- [41] <https://www.phpflow.com/php/web-service-types-soapxml-rpcrestful/>
(consulté 09.05.2019)
- [42] https://www.mightyminnow.com/2013/11/what-is-mobile-first-css-and-why-does-it-rock/mobile_v_s_desktop_internet-xlsx-1/ (consulté 11.05.2019)
- [43] <https://www.alsacreations.com/article/lire/1615-cest-quoi-le-responsive-web-design.html> (consulté 11.05.2019)
- [44] <https://dsp-interface.be/site-web/responsive/> (consulté 11.05.2019)

ملخص

يلخص هذا التقرير العمل الذي تم تطويره داخل شركة سفركوم كجزء من مشروع التخرج من أجل الحصول على شهادة الهندسة في علوم الإعلامية من المدرسة الوطنية للعلوم الإعلامية. الهدف من هذا المشروع هو إعادة تصميم مشروع بوتمن الحالي وتطوير تطبيق يسمح لبوتمن بالمتابعة والتكونين، وإنشاء سلسلة س.أ / س.د لدمج كل من التطبيقين.

الكلمات المفتاحية: متجاوية، أنقلار، قيتبلاب س.أ، دوكار، سوارم، منقوذب، الأوركستراتور، ك.ب.إ....

Résumé

Le présent rapport résume le travail élaboré au sein de l'entreprise Sofrecom dans le cadre du projet de fin d'études afin d'obtenir le diplôme d'ingénieur en sciences de l'informatique de l'École Nationale des Sciences de l'Informatique. L'objectif de ce projet est de faire une refonte du projet Botman existant, de développer une application web permettant le suivi et le paramétrage de Botman et de créer une chaîne CI /CD pour y intégrer les deux développements.

Mots-clés: Responsive, Angular, SpringBoot, Gitlab CI, Docker, Swarm, MongoDB, Orchestrateur, KPI ...

Abstract

This report summarizes the work developed within the company Sofrecom as part of the graduation project in order to obtain the engineering degree in computer science from the National School of Computer Sciences. The objective of this project is to redesign the existing Botman project, to develop a web application allowing to follow and configure Botman and to create a CI / CD chain to integrate both developments.

Keywords: Responsive, Angular, SpringBoot, CI Gitlab, Docker, Swarm, MongoDB, Orchestrator, KPI ...