Auteur : Julien Margarido 29/11/2016

Equipe : Anna Kostrikova, Félix Lima Gorito, Julien Margarido

Type : **Spécification Technique des Besoins et Exigences**

Chemin : github.com/JulienMrgrd/lab-bot/docs Statut : Final

Destinataire(s) : Jonathan Alimi, Cédric Besse, Mikael Gibert, Lamia Laraqui, Bruno Lesueur Page(s) : 33

STBE du projet “Hello Suribot”

Table des matières

1. Définition du projet

1.1. Les objectifs

Notre projet “Hello Suribot” est un bot conversationnel, qui doit pouvoir communiquer avec un client sur divers chats, analyser ses diverses demandes et lui répondre en fournissant les informations demandées.

Le bot reçoit une demande d’un utilisateur, qui est une phrase dans un langage humain, via Slack, Skype ou un autre chat. Cette demande est analysée par un moteur d’intelligence (de type *Recast.ai*), qui retourne la demande sous une forme qui permet au bot de faire une requête vers une ou plusieurs API externes. Ayant reçu une ou plusieurs réponses, le bot les traduits dans un langage humain et les transmets à l’utilisateur via le chat.

Hello Suribot est destiné, par exemple aux compagnies d’assurances et aux banques, et par extension, à leurs clients. Les objectifs de ce projet pour un client de Suricats Consulting (compagnie d’assurance, banque, …) sont de :

- diminuer les dépenses sur le personnel (prestation des services sans interaction humaine).
- augmenter la satisfaction des clients grâce à l’accès simplifié aux services 24h/24, sur diverses plateformes.

De plus, ce projet a pour objectifs pour Suricats Consulting de :

- attirer de nouveaux partenaires (possiblement dans d’autres secteurs d’activités).
- obtenir une réputation d’une entreprise technologiquement avancée.
- de satisfaire son propre intérêt dans les sphères de technologiques et développement d’entreprises.

1.2. Le contexte

Actuellement, il existe divers moyens de communication en ligne pour les utilisateurs, comme les applications Slack, Facebook Messenger, Skype, etc…

Récemment, Microsoft a mis en ligne un service, Microsoft Bot Connector (aussi appelé MBC), permettant de configurer rapidement et simplement un système de routage des messages.
En effet, il suffit de configurer sur leur interface un nouveau bot, de le connecter aux différents channels (Messenger, Slack, …), et de définir le point d’arrivée des messages (un “end-point”) qui correspond à l’URI du serveur où sera effectué le traitement des messages.
Ainsi, sur les différents channels configurés, un bot sera visible et utilisable, chaque message lui étant adressé sera reçu par Microsoft Bot Connector, et redirigé vers le serveur où se trouve le bot.
Nous allons donc utiliser ce service pour notre bot, celui-ci permettant :

- de faire une application dont l’implémentation ne dépend pas des applications de communication des utilisateurs.
- une grande modularité, puisque l’ajout d’une nouvelle application de chat se fait en quelques minutes, simplement avec de la configuration sur Bot Connector.
- de pouvoir utiliser dans le futur un autre système équivalent à Bot Connector, sans devoir faire une refonte complète du projet.

Enfin, nous utiliserons Recast.ai, une plateforme (développée récemment par des anciens étudiants de l'école 42) mise à disposition des développeurs permettant de créer des bots conversationnels, auto-apprenants et personnalisables. De plus, Recast permet d'analyser du texte, et d'en retourner un contexte, avec des tags ou des mots (sous forme d' "intents").

Exemples d'objets métiers d'une API externe d'assurance de "Hello suribot":

- Personne : identifiant, nom, prénom, code postal, …
- Contrat : identifiant, objets couverts, billings, party rôles, ...
- Billing : identifiant, la somme, périodicité, date prochaine, moyen de paiement, ...

Exemples de procédures utilisables sur Hello Suribot :

- Consultation des informations concernant les objets couverts d'un utilisateur
- Obtention des prélèvements liés à un contrat d'un utilisateur

1.3. Le périmètre du système

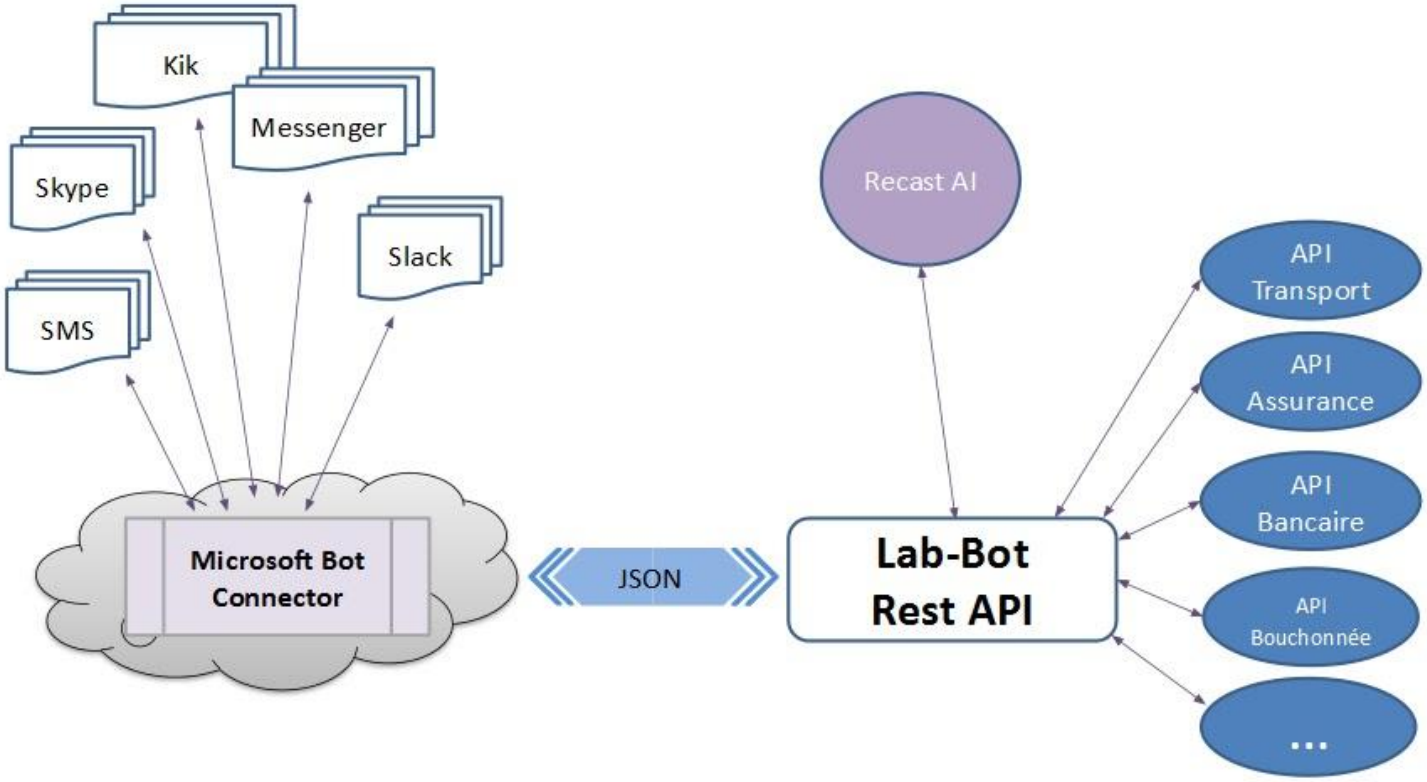
Voici les fonctionnalités que "Hello Suribot" devra fournir :

- Renseigner un client sur des éléments de ses différents contrats.
- Configuration d'une ou plusieurs analyses sur Recast.ai.
- L'application doit être facilement modulable (ajout d'une nouvelle API, changement de moteur d'intelligence artificielle,…) et configurable (autre plateforme que MBC, ajout d'un nouveau canal de communication,…).
- Une API REST bouchonnée, que l'on pourra interroger à l'aide de plusieurs méthodes (et paramètres) et qui nous retournera des fausses données de contrat d'assurance, sous format JSON.

Voici les fonctionnalités que « Hello Suribot » ne pourra pas fournir :

- Souscrire à des contrats.
- Créer ou modifier des données (contrats, comptes bancaires, …).
- Fournir une réponse correcte et cohérente par rapport à la question de l'utilisateur avec une fiabilité de 100%.

1.4. Interactions entre le système à réaliser et l'organisation existante



2. Définition du système à réaliser

2.1. Architecture fonctionnelle du système

Pour la réalisation de "Hello Suribot", nous avons fait le choix de diviser notre architecture en 5 sous-systèmes.
Le premier que nous présenterons est le sous-système communiquant avec Microsoft Bot Connector, le deuxième sera celui communiquant avec Recast.ai.
Le troisième aura pour but d'analyser les réponses de Recast.ai et pourra éventuellement (suivant l'analyse de la réponse) récupérer une ou plusieurs données d'une ou plusieurs API externes, par l'intermédiaire du quatrième sous-système.
De plus, il obtiendra une réponse générée par le cinquième sous-système, puis pourra l'envoyer au premier sous-système permettant de le transférer à l'utilisateur.

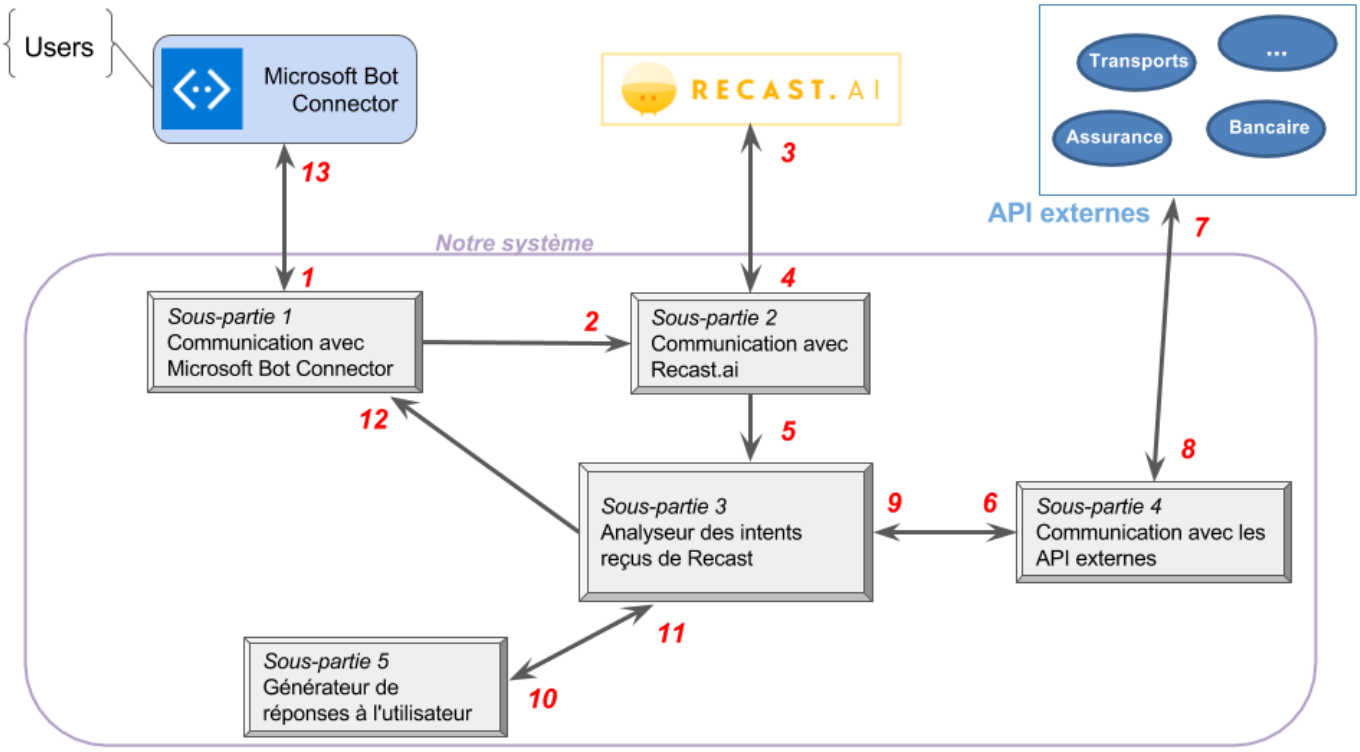


Schéma décrivant l'architecture fonctionnelle
(le chemin de traitement d'une demande est indiqué par les nombres)

2.2. Sous-systèmes

Différents services :

- IJSonDecoder : permet d'analyser et de récupérer des données d'un texte au format JSON
- IJSonCreator : permet de créer et d'injecter des données dans un texte au format JSON
- IHttpSender : permet d'envoyer des requêtes HTTP

2.2.1. S.S.1 Communication avec Microsoft Bot Connector

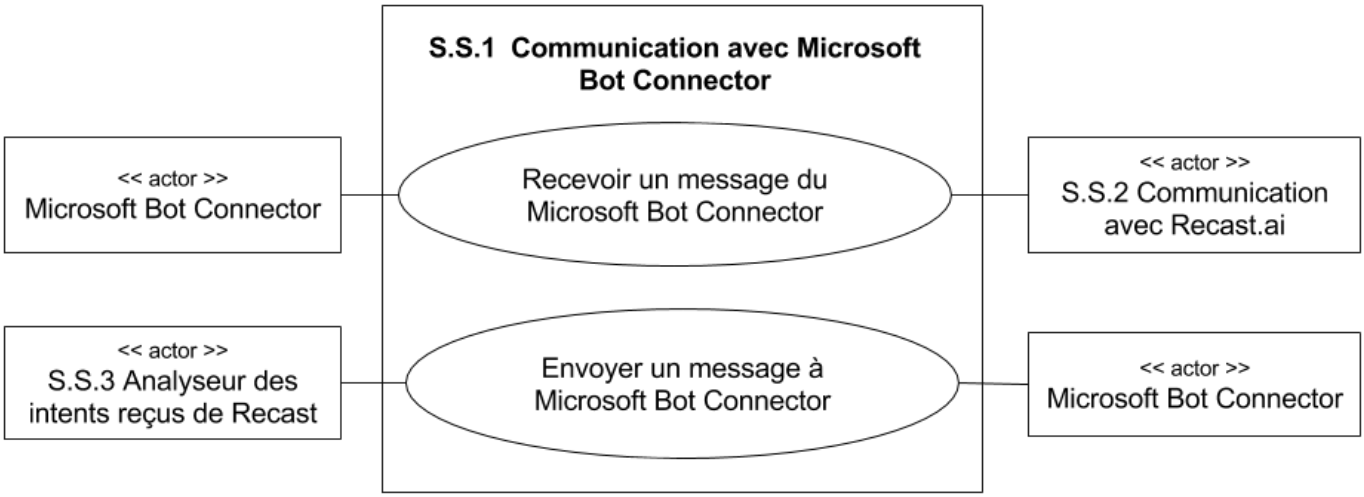
2.2.1.1. Architecture du sous-système

Le sous-système se trouve dans le package "communication.mbc".
Il possèdera comme interfaces offertes : IHttpSender.
Il possèdera comme interfaces requises : IHttpSender.

Le sous-système aura pour rôle de communiquer avec Microsoft Bot Connector.
Ce composant devra :

- Récupérer un message de l'utilisateur et le faire parvenir au S.S.2.
- Récupérer un message du S.S.3 et le faire parvenir à Microsoft Bot Connector.

2.2.1.2. Diagramme de cas d'utilisations



2.2.1.3. Acteurs

2.2.1.3.1. Acteur « Microsoft Bot Connector »

Nom	Microsoft Bot Connector
Type	Système
Rôle	Transmettre la requête de l'utilisateur au système

2.2.1.3.2. Acteur « S.S.2 Communication avec Recast.ai »

Nom	S.S.2 Communication avec Recast.ai
Type	Système
Rôle	Faire traiter la requête par Recast.ai

2.2.1.3.3. Acteur « S.S.3 Analyseur des intents reçus de Recast»

Nom	S.S.3 Analyseur des intents reçus de Recast
Type	système
Rôle	Ce composant a pour rôle de traiter la réponse transmise par Recast.ai puis d'appeler S.S.4 avec les bonnes données, et S.S.5 pour générer la réponse.

2.2.1.4. Cas d'utilisations

2.2.1.4.1. Envoyer un message à MBC

Fiche de description détaillée du cas d'utilisation

Nom du cas d'utilisation	Envoyer un message à MBC
Acteur principal	S.S.3 Analyseur des intents reçus de Recast
Acteurs secondaires	Microsoft Bot Connector
Objectif	Transmettre la réponse pour l'utilisateur

Pré-condition(s)	Une demande a transitée par S.S.1
Hypothèses	Aucune
Scénario	Le message à envoyer par l'utilisateur a été traité par tous les sous-systèmes
Post-condition(s)	Le message reçu par MBC et le même que celui généré par S.S.5
Exigences non fonctionnelles	Configuration de MBC OK
Exceptions, alternatives	Aucune

Tests de validation du cas d'utilisation ("nX" signifie "nominal numéro X" et "aX" signifie "alternative numéro X")

Identifiant du test	**SS1n1**
Titre	Tester l'envoi du message par S.S.1 - Nominal
Use Case correspondant	Envoyer un message à MBC
Contexte	Une demande a été traitée par le système
Pré-condition(s)	Aucune

Entrée(s)	Le message (JSON)
	Exemple :
	{
	"sender" : [type : "Slack", user : "Félix", ...],
	"receiver" : [type : "Slack", user : "Jonathan", ...],
	"text" : "31 rue Jussieu 75005 Paris"
	...
	}

Scénario	Le message envoyé par l'utilisateur est traité par tous les sous-systèmes. S.S.5 fourni la réponse à S.S.1 S.S.1 envoie la réponse fournie par S.S.3 à MBC.
Résultat attendu	L'utilisateur reçoit la réponse
Moyen de vérification	Le message reçu par l'utilisateur est identique au message généré par S.S.3

2.2.1.4.2. Recevoir un message de MBC

Fiche de description détaillée du cas d'utilisation

Nom du cas d'utilisation	Recevoir un message de MBC
Acteur principal	Microsoft Bot Connector
Acteurs secondaires	S.S.2 Communication avec Recast.ai
Objectif	Recevoir le message d'un utilisateur
Pré-condition(s)	Un message a été écris par un utilisateur. MBC transmet le message à S.S.1.

Hypothèses	Aucune
Scénario	S.S.1 reçoit un message (JSON) de MBC et le transmet à S.S.2, sans appliquer de modifications. MBC envoie un message à S.S.1. S.S.1 reçoit le message de MBC, et le transmet à S.S.2.
Post-condition(s)	S.S.2 peut traiter le message fourni par S.S.1
Exigences non fonctionnelles	Configuration de MBC OK
Exceptions, alternatives	Aucune

Tests de validation du cas d'utilisation

Identifiant du test	**SS1n2**
Titre	Tester la réception du message par S.S.1 - Nominal
Use Case correspondant	Recevoir un message de MBC
Contexte	Un message a été écris par un utilisateur. MBC transmet le message à S.S.1.
Pré-condition(s)	Aucune
Entrée(s)	Le message (JSON) Exemple : { "sender" : [type : "Slack", user : "Félix", ...], "receiver" : [type : "Slack", user : "Jonathan", ...], "text" : "Y'a-t-il une bnp proche de Jussieu à Paris ?" ... }
Scénario	S.S.1 reçoit le message (JSON) de MBC et le transmet à S.S.2, sans appliquer de modifications. MBC envoie le message à S.S.1 S.S.1 reçoit le message de MBC, et le transmet à S.S.2.
Résultat attendu	S.S.2 reçoit le message (paramètre « text »)
Post-condition(s)	Aucune
Moyen de vérification	Le message reçu par S.S.1 est le même que celui envoyé par l'utilisateur

2.2.2. S.S.2 Communication avec Recast.ai

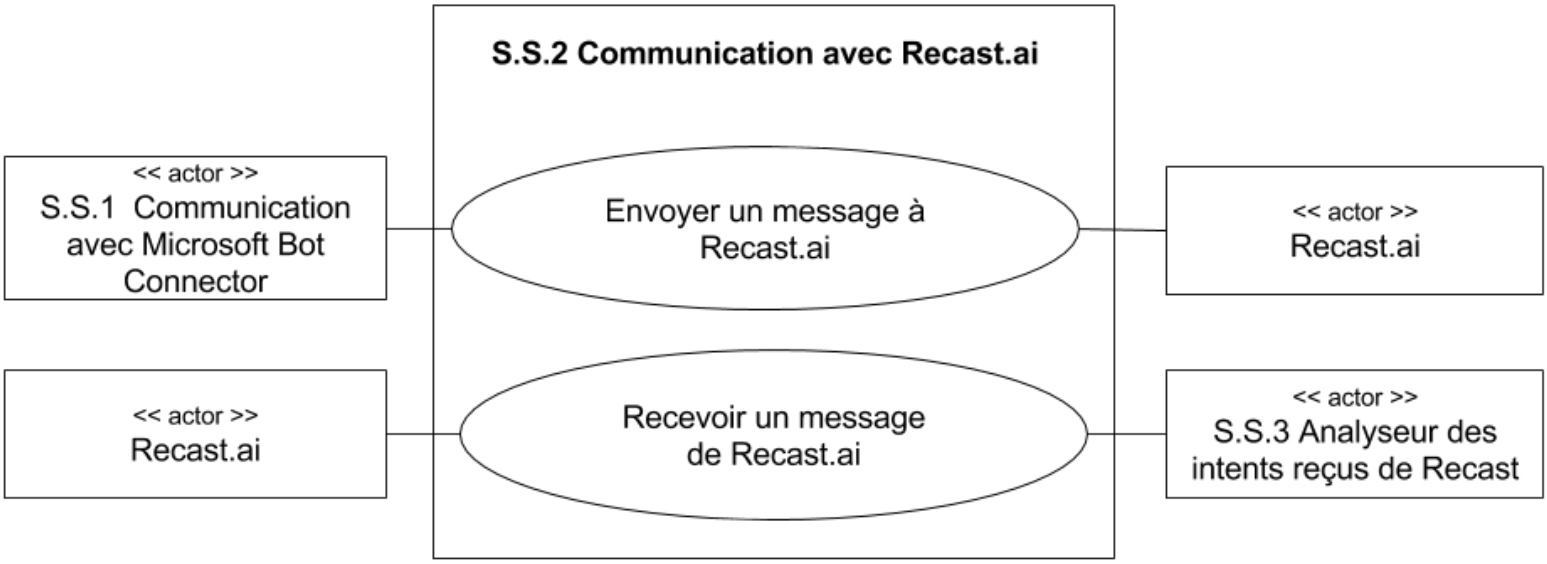
2.2.2.1. Architecture du sous-système

Le sous-système se trouve dans le package “communication.recast".
Il possèdera comme interfaces offertes : IHttpSender.
Il possèdera comme interfaces requises : IJSonDecoder.

Le sous-système aura pour rôle de communiquer avec Recast.ai.
Ce composant devra:

- récupérer un message du S.S.1 et le faire parvenir à Recast.ai.
- récupérer un message de Recast.ai et le transmettre à S.S.3.

2.2.2.2. Diagramme de cas d’utilisations



2.2.2.3. Acteurs

2.2.2.3.1. Acteur « S.S.1 Communication avec Microsoft Bot Connector »

Nom	S.S.1 Communication avec Microsoft Bot Connector
Type	Système
Rôle	Réceptionner les messages de l’utilisateur et envoyer des messages à l’utilisateur

2.2.2.3.2. Acteur « S.S.3 Analyseur des intents reçus de Recast »

Nom	S.S.3 Analyseur des intents reçus de Recast
Type	Système
Rôle	Ce composant a pour rôle de traiter la réponse transmise par Recast.ai puis d'appeler S.S.4 avec les bonnes données, et S.S.5 pour générer la réponse à l'utilisateur.

2.2.2.3.3. Acteur « Recast.ai »

Nom	Recast.ai
Type	Système
Rôle	Analyse un message pour en donner un contexte, sous forme d'intents (format JSON)

2.2.2.4. Cas d'utilisations

2.2.2.4.1. Envoyer un message à Recast.ai

Fiche de description détaillée du cas d'utilisation

Nom du cas d'utilisation	Envoyer un message à Recast.ai
Acteur principal	S.S.1 Communication avec Microsoft Bot Connector
Acteurs secondaires	Recast.ai
Objectif	Transmettre le message de l'utilisateur à Recast.ai
Pré-condition(s)	S.S.2 a reçu un message de S.S.1
Hypothèses	Aucune
Scénario	S.S.2 reçoit un message (JSON) du S.S.1, construit une requête à Recast.ai et l'envoie
Post-condition(s)	Recast.ai traite le message transmis par S.S.2
Exigences non fonctionnelles	Connexion et configuration avec Recast OK
Exceptions, alternatives	Aucune

Tests de validation du cas d'utilisation

Identifiant du test	**SS2n1**
Titre	Tester l'envoi du message par S.S.2 - Nominal
Use Case correspondant	Envoyer un message à Recast
Contexte	Aucun
Pré-condition(s)	S.S.2 a reçu un message JSON de S.S.1

Entrée(s)	Le message (JSON)
	Exemple :
	{
	“sender” : [type : “Slack”, user : “Félix”, ...],
	“receiver” : [type : “Slack”, user : “Jonathan”, ...],
	“text” : “Y’a-t-il une bnp proche de Jussieu à Paris ?”
	...
	}

Scénario	S.S.2 envoie un message (JSON) de S.S.1, construit une requête pour Recast, et lui envoie
Résultat attendu	Recast.ai reçoit le message de l’utilisateur et génère des intents
Moyen de vérification	Sur la page de Recast, il faut vérifier que le message envoyé fait bien parti de l’historique du compte

2.2.2.4.2. Recevoir un message de Recast.ai

Fiche de description détaillée du cas d’utilisation

Nom du cas d’utilisation	Recevoir un message de Recast.ai
Acteur principal	Recast.ai
Acteurs secondaires	S.S.3 Analyseur d’intents reçu de Recast.ai
Objectif	Recevoir les intents de Recast
Pré-condition(s)	Un message avait été envoyé à Recast par S.S.2
Hypothèses	Aucune
Scénario	S.S.2 envoie un message à Recast.ai. Recast génère une réponse sous forme d’intents (JSON) à S.S.2. S.S.2 reçoit ces intents, et les transmets à S.S.3.
Post-condition(s)	S.S.3 reçoit les intents de Recast.ai
Exigences non fonctionnelles	Connexion et configuration de Recast OK
Exceptions, alternatives	Aucune

Tests de validation du cas d’utilisation

Identifiant du test	**SS2n2**
Titre	Tester la réception du message de Recast.ai - Nominal
Use Case correspondant	Recevoir un message de Recast.ai
Contexte	Aucun
Pré-condition(s)	S.S.2 a envoyé un message à Recast.ai, et celui-ci a répondu des intents (JSON)
Entrée(s)	Le message de Recast (JSON)

	Exemple :
	{
	‘’sources’’ : ‘’Y’a-t-il une BNP à Jussieu ?’’,
	“type” : “question”,
	“bank” : [
	{
	“value” : “BNP Paribas”,
	“raw” : “bnp”
	}
	] ,
	“location” : [
	{
	“value” : “Jussieu”,
	“raw” : “Jussieu”,
	“lat” : 48.856762,
	“lng” : 2.356762
	}
	] }

Scénario	S.S.2 envoie un message à Recast.ai. Recast génère la réponse sous forme d’intents (JSON) à S.S.2. S.S.2 reçoit ces intents, et les transmets à S.S.3.
Résultat attendu	S.S.3 reçoit les intents de Recast.ai
Moyen de vérification	Les intents contiennent le message d’origine (paramètre « source »)

2.2.3. S.S.3 Analyseur des intents reçus de Recast

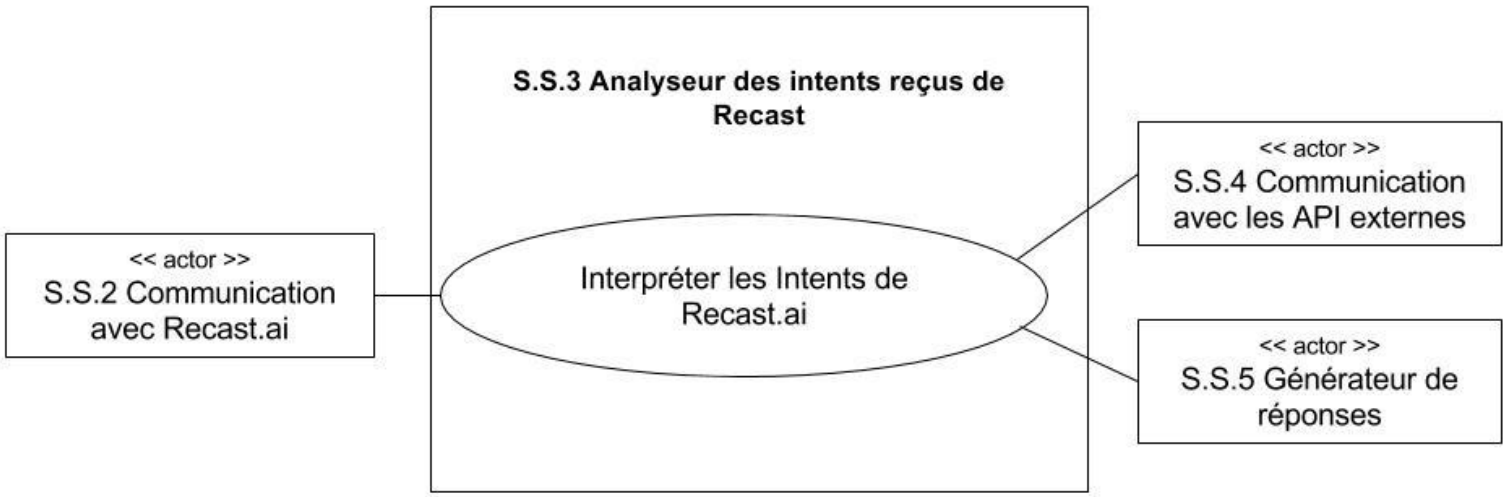
2.2.3.1. Architecture du sous-système

Le sous-système se trouvera dans le package “analyze".
Il possèdera comme interfaces offertes : IJsonDecoder.
Il possèdera comme interfaces requises : IHttpSender.

Ce composant aura pour rôle d’analyser les messages JSON reçus par le sous-système S.S.2 “Communication avec Recast”, et de faire parvenir au sous-système 2.2.4 “Communication avec les API externes” la (ou les) donnée(s) à récupérer d’une (ou plusieurs) API externe(s), en passant le nom de l’API, la méthode à appeler, ainsi que les éventuels paramètres.

De plus, il devra demander au sous-système 2.2.5 “Formateur de réponses à l'utilisateur” de formuler une réponse (suivant le nombre de paramètres reçus).

2.2.3.2. Diagramme de cas d'utilisations



2.2.3.3. Acteurs

2.2.3.3.1. Acteur « S.S.2 Communication avec Recast.ai »

Nom	S.S.2 Sous-système de communication avec Recast
Type	Système
Rôle	Ce sous-système est chargé d'envoyer les messages des utilisateurs à Recast, et de récupérer les réponses de celui-ci. Cette réponse est sous format JSON, et contient des intents.

2.2.3.3.2. Acteur « S.S.4 Sous-système de communication avec les API externes »

Nom	S.S.4 Sous-système de communication avec les API externes
Type	Système
Rôle	Ce sous-système a pour rôle de communiquer avec les API externes.

2.2.3.3.3. Acteur « S.S.5 Sous-système formateur de réponses à l'utilisateur »

Nom	S.S.5 Sous-système formateur de réponses à l'utilisateur
Type	Système
Rôle	Ce sous-système a pour rôle de formater les réponses sous format JSON destinées à être envoyées à l'utilisateur.

2.2.3.4. Cas d'utilisations

2.2.3.4.1. Interpréter les intents de Recast.ai

Fiche de description détaillée du cas d'utilisation

Nom du cas d'utilisation	Interpréter les intents de Recast.ai
Acteur principal	S.S.2 Sous-système de communication avec Recast
Acteurs secondaires	S.S.4 Communication avec les API externes S.S.5 Formateur de réponses à l'utilisateur
Objectif	Générer une réponse adaptée à la demande de l'utilisateur
Pré-condition(s)	S.S.2 fourni à S.S.3 les intents envoyés à Recast.ai
Hypothèses	Aucune
Scénario	S.S.2 transmet les intents fournis par Recast.ai à S.S.3. S.S.3 analyse les intents. S.S.3 transmet à S.S.4 la bonne API à appeler, ainsi que la méthode et les paramètres à fournir. S.S.3 récupère les données fournies par S.S.4. S.S.3 demande à S.S.5 de générer une réponse pour l'utilisateur. S.S.3 transmet à S.S.1 la réponse.
Post-condition(s)	S.S.1 reçoit la réponse de l'utilisateur
Exigences non fonctionnelles	Aucune
Exceptions, alternatives	Alternative : La réponse de Recast.ai fournie par S.S.2 est vide ou ne contient pas de tags connus par notre système concernant les API externes. S.S.4 n'est donc pas appelé, et on demande à S.S.5 de générer une réponse de non compréhension de la demande à l'utilisateur.

Tests de validation du cas d'utilisation

Identifiant du test	**SS3n1**
Titre	**Vérifier la bonne interprétation des intents - Nominal**
Use Case correspondant	Interpréter les intents de Recast.ai
Contexte	Aucun
Pré-condition(s)	Un message d'un utilisateur a été envoyé à Recast.ai, et des intents ont été transmis par S.S.2 à S.S.3.

Entrée(s)	La réponse de Recast.ai, sous format JSON.
	Exemple :
	
	{
	"type" : "question",
	"bank" : [
	{
	"value" : "BNP Paribas",
	"raw" : "bnp"
	}
	] ,
	"location" : [
	{
	"value" : "Jussieu",
	"raw" : "Jussieu",
	"lat" : 48.856762,
	"lng" : 2.356762
	}
	]
	}

Scénario	(exemple dans le cas d'intents concernant une demande en rapport avec une banque) S.S.3 analyse les intents. Il détecte que les intents concernent une demande de position d'une banque (BNP). De plus, il interroge donc S.S.4, en lui fournissant le nom de l'API de la BNP, ainsi que la méthode et les paramètres pour récupérer l'éventuelle position d'une banque à proximité de la position donnée par l'utilisateur. S.S.4 retourne un résultat. On demande à S.S.5 de générer une réponse à partir de ce résultat.
Résultat attendu	S.S.1 récupère la réponse à fournir à l'utilisateur.
Moyen de vérification	S.S.1 récupère la même réponse que celle générée par S.S.5, et la réponse est cohérente à la demande de l'utilisateur.

Identifiant du test	**SS3a1**
Titre	**Vérifier la bonne interprétation d'intents vides - Nominal**
Use Case correspondant	Interpréter un message formé d'intents
Contexte	Aucun
Pré-condition(s)	Un message d'un utilisateur (exemple: Yatil op bnnpp proxhe de Juzzieu ?") a été envoyé à Recast.ai par S.S.2, puis fourni à S.S.3 les intents de Recast.
Entrée(s)	La réponse de Recast.ai, sous format JSON. Exemple : { }
Scénario	S.S.3 analyse les intents. Il détecte que les intents sont vides. Il n'interroge pas S.S.4. Il demande à S.S.5 de générer une réponse de non compréhension de la demande. Il fournit cette réponse à S.S.1.
Résultat attendu	S.S.1 récupère une réponse de non compréhension de la demande.
Moyen de vérification	Lorsque les intents retournés par Recast sont vides, S.S.1 récupère un message de non compréhension de la demande (le même que celui paramétré par défaut dans S.S.5)

2.2.4. S.S.4 Sous-système de communication avec les API externes

2.2.4.1. Architecture du sous-système

Ce composant se trouvera dans le package "communication.api".

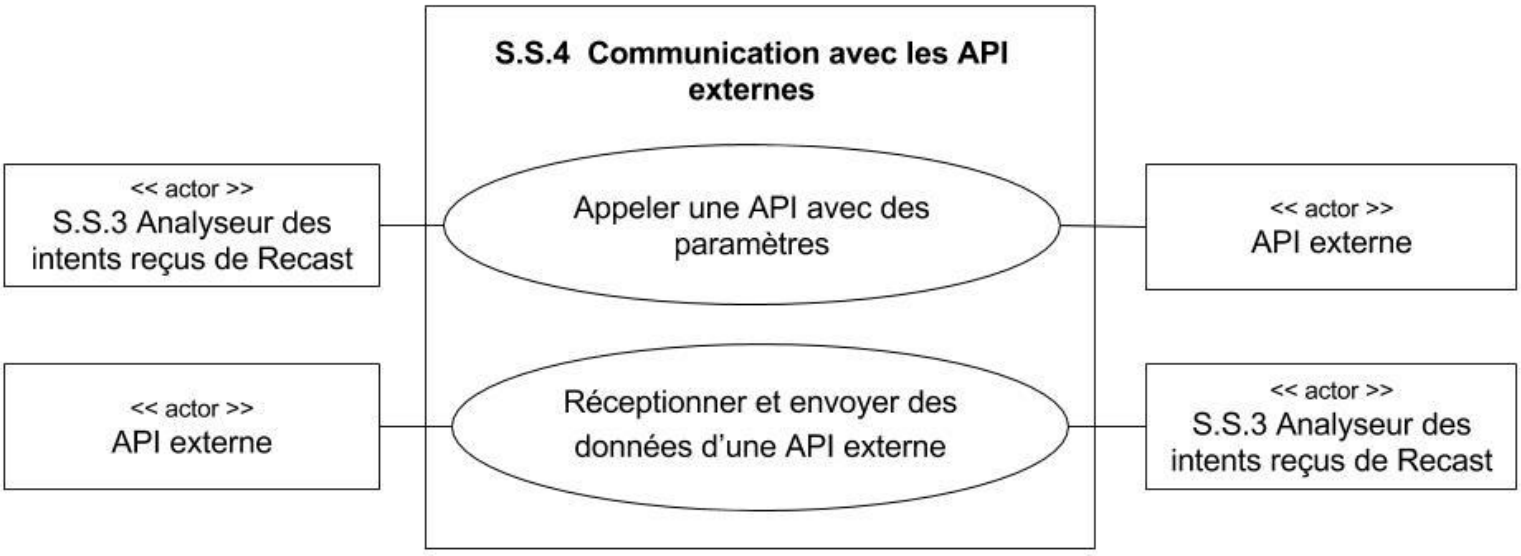
Il possèdera comme interfaces offertes : IHttpSender.

Il possèdera comme interfaces requises : IHttpSender.

Le sous-système aura pour rôle de communiquer avec les API externes après avoir reçu le message analysé du sous-système 2.2.3 "Analyseur des intents reçus de Recast".
Ce composant devra:

- appeler la bonne API externe avec la (les) donnée(s) en paramètres.
- récupérer la réponse de l'API externe et la faire parvenir au sous-système 2.2.5.

2.2.4.2. Diagramme de cas d'utilisations



2.2.4.3. Acteurs

2.2.4.3.1. Acteur « API externe »

Dans le cadre de ce projet, il est obligatoire que cette API soit une API REST, fournissant une liste de méthodes dont l'appel permet d'obtenir une réponse au format JSON.
Pour les besoins du développement, une API REST bouchonnée (disponible sur github.com/suricats/lab-bot-api) sera créé, et permettra d'interroger une API externe en simulant un retour de données.

Nom	API externe
Type	Système
Rôle	Ce composant a pour rôle de traiter la requête envoyée par S.S.4

2.2.4.3.2. Acteur « S.S.3 Analyseur des intents reçus de Recast »

Nom	S.S.3 Analyseur des intents reçus de Recast
Type	Système
Rôle	Ce composant a pour rôle de traiter la réponse transmise par Recast.ai puis d'appeler S.S.4 avec les bonnes données, et S.S.5 pour générer la réponse.

2.2.4.4. Cas d'utilisations

2.2.4.4.1. Appeler une API externe avec des paramètres

Fiche de description détaillée du cas d'utilisation

Nom du cas d'utilisation	Appeler une API externe avec des paramètres
Acteur principal	S.S.3 Analyseur des intents reçus de Recast
Acteurs secondaires	API externe
Objectif	L'objectif est d'appeler une API avec des paramètres
Pré-condition(s)	L'API indiquée existe. L'API est disponible.
Hypothèses	Aucune
Scénario	S.S.3 a formé une URL, avec le nom de l'API, la méthode et les éventuels paramètres, puis l'a transmise à S.S.4. S.S.4 émet la requête à l'API externe.
Post-condition(s)	L'API traite la demande reçue
Exigences non fonctionnelles	L'API externe doit être accessible par S.S.4
Exceptions, alternatives	Aucune

Tests de validation du cas d'utilisation

Identifiant du test	**SS4n1**
Titre	Tester l'appel à l'API externe - Nominal
Use Case correspondant	Appeler une API externe avec des paramètres
Contexte	S.S.3 a généré une URL avec un nom de méthode et des paramètres
Pré-condition(s)	Aucune
Entrée(s)	L'URL à appeler. Exemple: https://api.bnp.fr/dateDeFinContrat?contrat=1686952354
Scénario	S.S.3 génère l'URL à appeler et la transmet à S.S.4 S.S.4 fait une requête avec l'URL
Résultat attendu	L'API reçoit une requête
Moyen de vérification	L'envoi de la requête provoque l'appel à la bonne API et méthode avec les bons paramètres, les mêmes que ceux indiqués dans la requête fournie par S.S.3.

2.2.4.4.2. Réceptionner et envoyer des données d'une API externe

Fiche de description détaillée du cas d'utilisation

Nom du cas d'utilisation	Réceptionner et envoyer des données d'une API externe
Acteur principal	API externe
Acteurs secondaires	S.S.3 Analyseur des intents reçus de Recast.ai
Objectif	L'objectif est de réceptionner les données de l'API externe et de les envoyer à S.S.3
Pré-condition(s)	Une requête a été envoyée à une API externe.
Hypothèses	Aucune
Scénario	Une requête a été envoyée à l'API externe par S.S.4. L'API externe a générée une réponse au format JSON. S.S.4 récupère cette réponse au format JSON et la transmet à S.S.3
Post-condition(s)	S.S.3 récupère le résultat fourni par l'API.
Exigences non fonctionnelles	Aucune
Exceptions, alternatives	Aucune

Tests de validation du cas d'utilisation

Identifiant du test	**SS4n2**
Titre	Tester la réception des données de l'API externe - Nominal
Use Case correspondant	Réceptionner et envoyer des données d'une API externe
Contexte	S.S.4 a envoyé une requête à l'API puis attend sa réponse.
Pré-condition(s)	L'API externe a reçue la requête

Entrée(s)	La réponse (JSON)
	Exemple :
	{
	"contrat" : [
	{
	"identifiant" : "1686952354",
	"dateDeFin" : "25/11/2016"
	}
	]
	}

Scénario	Une requête a été envoyée à l'API externe par S.S.4. L'API externe a générée la réponse au format JSON. S.S.4 récupère cette réponse au format JSON et la transmet à S.S.3
Résultat attendu	S.S.3 reçoit la même donnée que celle fournie par l'API externe.
Moyen de vérification	Vérifier que le message reçu par S.S.3 et le résultat obtenu par l'appel de l'API soient identiques.

Identifiant du test	SS4e2
Titre	Tester la réception des données de l'API externe - Erreur
Use Case correspondant	Réceptionner et envoyer des données d'une API externe
Contexte	L'appel de l'API externe a été fait avec une URL incorrect, par conséquent l'API retourne une erreur 404.
Pré-condition(s)	Aucune
Entrée(s)	Aucune
Scénario	S.S.4 appelle une API avec des paramètres ou des noms de méthode erronés. L'API retournera une erreur 404. S.S.4 retournera une réponse vide à S.S.3.
Résultat attendu	Un message vide.
Post-condition(s)	Aucune
Moyen de vérification	Le S.S.3 va recevoir un JSON vide.

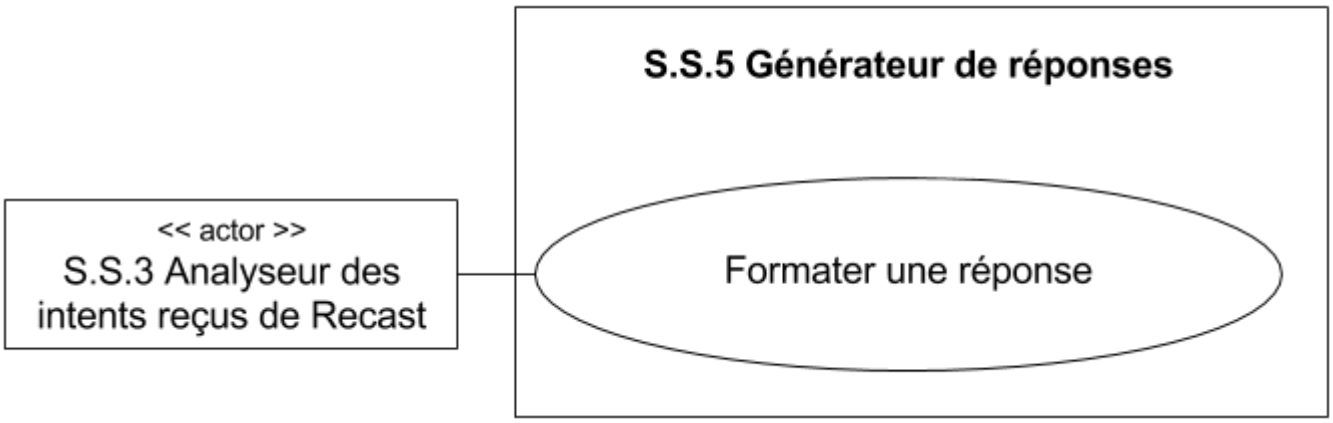
2.2.5. S.S.5 Générateur de réponses à l'utilisateur

2.2.5.1. Architecture du sous-système

Ce composant se trouvera dans le package "response".
Il possèdera comme interfaces offertes : IJsonDecoder, IJsonCreator.
Il possèdera comme interface requise : IHttpSender.

Le sous-système aura pour rôle de recevoir des instructions du S.S.3 (avec d'éventuels paramètres) dans le but de générer une réponse au format JSON à donner à l'utilisateur.

2.2.5.2. Diagramme de cas d'utilisations



2.2.5.3. Acteurs

2.2.5.3.1. Acteur « S.S.3 Analyseur des intents reçus de Recast »

Nom	S.S.3 Analyseur des intents reçus de Recast
Type	Système
Rôle	Ce composant a pour rôle de traiter la réponse transmise par Recast.ai puis d'appeler S.S.4 avec les bonnes données, et S.S.5 pour générer la réponse.

2.2.5.4. Cas d'utilisations

2.2.5.4.1. Formater une réponse

Fiche de description détaillée du cas d'utilisation

Nom du cas d'utilisation	Formater une réponse
Acteur principal	S.S.3 Analyseur des intents reçus de Recast
Objectif	Générer une réponse dans une forme compréhensible par l'utilisateur
Pré-condition(s)	S.S.3 envoie une instruction avec S.S.5, avec d'éventuels paramètres
Hypothèses	Aucune
Scénario	S.S.5 reçoit une instruction de S.S.3 (un appel de méthode particulier). S.S.5 génère une réponse, en y incluant d'éventuelles données fournies par S.S.3. S.S.5 fournit la réponse générée à S.S.3.

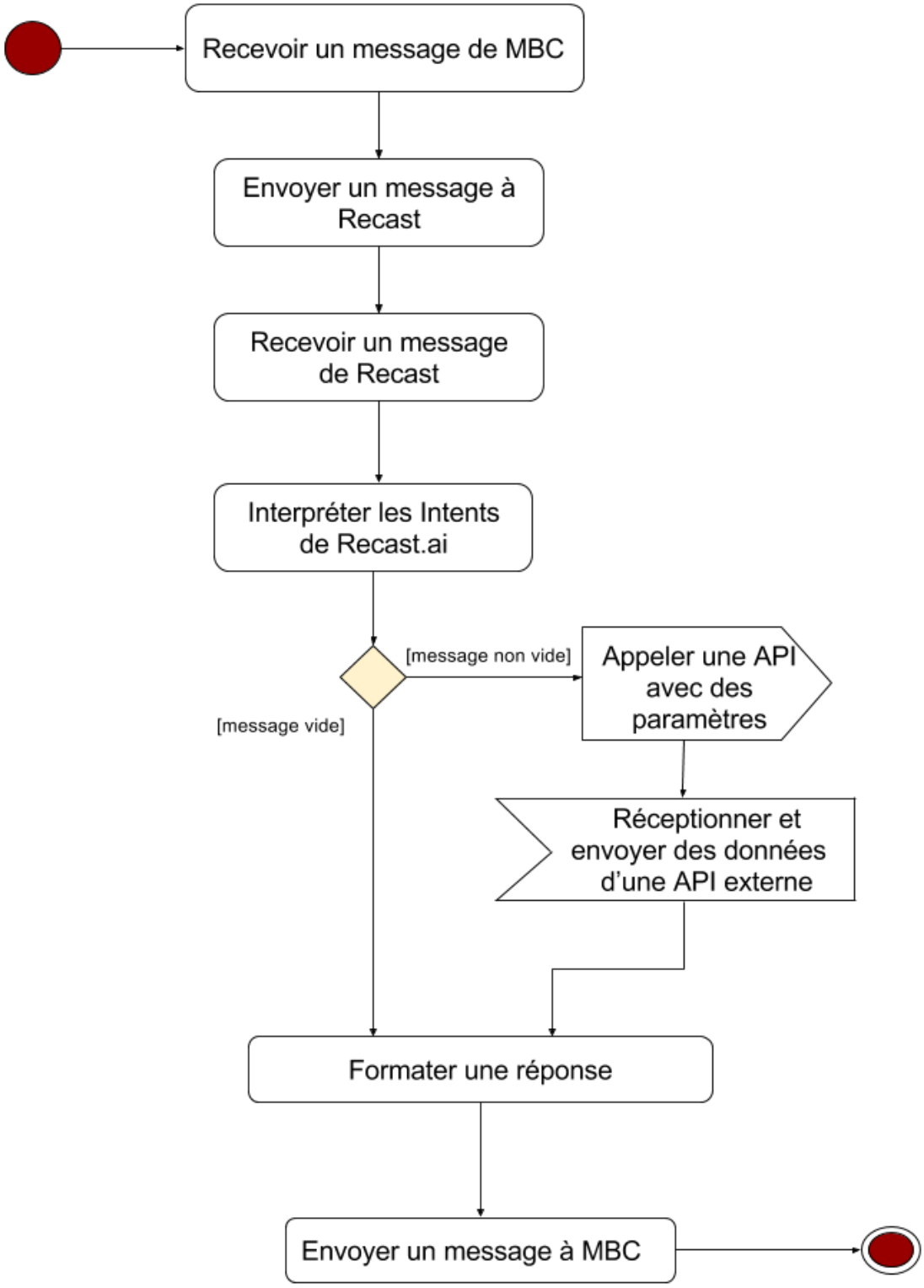
Post-condition(s)	Aucune
Exigences non fonctionnelles	Aucune
Exceptions, alternatives	Alternative : S.S.3 envoie à S.S.5 une instruction de non-compréhension du message. S.S.5 fournit un message « Veuillez reformuler votre demande » à S.S.3.

Tests de validation du cas d'utilisation

Identifiant du test	**SS5n1**
Titre	Tester le formatage d'une réponse pour S.S.3 - Nominal
Use Case correspondant	Formater une réponse
Contexte	S.S.3 génère une instruction à S.S.5
Pré-condition(s)	Aucune
Entrée(s)	Une donnée « 12474762 »
Scénario	S.S.5 reçoit comme instruction de S.S.3 de générer une réponse, avec en paramètre la donnée. S.S.5 génère à S.S.3 la réponse désirée.
Résultat attendu	S.S.3 reçoit la réponse « La réponse à votre demande est : 12474762 »
Moyen de vérification	La réponse fournie par S.S.5 à S.S.3 est non-vide, et contient bien la donnée fournie

Identifiant du test	**SS5a1**
Titre	Tester le formatage d'une réponse pour S.S.3 - Alternative
Use Case correspondant	Formater une réponse
Contexte	S.S.3 génère une instruction de non-compréhension à S.S.5
Pré-condition(s)	Aucune
Entrée(s)	Aucune
Scénario	S.S.5 génère une réponse de non-compréhension, et la fournie à S.S.3
Résultat attendu	S.S.3 reçoit la réponse « Veuillez reformuler votre demande. »
Post-condition(s)	Aucune
Moyen de vérification	Vérifier que S.S.3 reçoit bien la réponse « Veuillez reformuler votre demande. »

2.3. Détails d'interactions entre cas d'utilisations



2.4. Prototypage

Aucun prototype n'est à effectuer dans le cadre de ce projet.

Le développement de ce projet est basé sur une implémentation d'une nouvelle fonctionnalité (sous forme de POC) chaque semaine.

3. Contraintes

3.1. Les types de contraintes

3.1.1. Techniques

Suricats Consulting ne nous a définis aucune contrainte technique concernant l'utilisation d'un langage de programmation en particulier. Nous avons donc décidé d'utiliser Java. Par conséquent, la machine distante fournis par Suricats Consulting devra pouvoir faire fonctionner les applications java.

Une contrainte imposée est le stockage de l'ensemble du code et de la documentation en lien avec ce projet sur github dans un répertoire public.

3.1.2. Fonctionnelles

Notre projet étant entièrement Open source, la contrainte fonctionnelle que nous avons repérée (et par conséquent eu à gérer) est le cryptage de l'ensemble des données dites "sensibles" à notre projet, comme les identifiants de connexion, l'adresse IP du serveur ou encore les variables d'environnement.

Pour crypter ces données sensibles, nous avons décidé d'utiliser l'outil Travis, qui s'occupe de crypter et de décrypter les données voulus mais également les fichiers si nécessaire.

3.1.3. Performances et planning

Pour que notre bot soit exploitable par les différents clients de Suricats Consulting (et par extension aux utilisateurs communiquant avec le bot), nous avons identifié 2 contraintes liées à la performance.

La première est la disponibilité du système : en effet la principale utilitée d'un bot est qu'il soit disponible sans interruption, contrairement aux humains qui ont besoin de repos.

La seconde est le délai de réponse de notre bot à l'utilisateur final. Nous avons défini ce seuil à une durée de 2 secondes, pour éviter une attente trop longue de l'utilisateur et par conséquent une opinion défavorable vis-à-vis du robot qui mettrait trop de temps à lui répondre.

3.1.4. Ressources

Ce projet est réalisé par une équipe de 3 développeurs.
L'entreprise ne nous a pas définit de budget propre au projet. S'il est nécessaire d'utiliser des API ou certain logiciel payant, il faut en discuter avec Suricats Consulting pour voir avec eux la décision à prendre.
Notre bot "Hello Suribot" tournera sur une machine Ubuntu fournie par la société Suricats Consulting.

3.2. Analyse des risques

3.2.1. Risques liés aux contraintes techniques

Mauvaise assimilation d'une nouvelle technologie	
Description	Le groupe de développeur n'arrive pas à comprendre une nouvelle technologie suffisamment rapidement.
Causes	Mauvaise documentation technique de la nouvelle technologie. Manque de motivation de la part des développeurs pour l'assimiler le plus rapidement. Complexité importante de cette technologie.
Conséquences	L'utilisation de la nouvelle technologie prendra du retard comparée au planning et par conséquent entraînera un retard du projet.
Moyens de prévention	Bien lire la documentation disponible Poser des questions aux personnes connaissant mieux la technologie.
Gravité	Moyen

3.2.2. Risques liés aux fonctionnalités

Aucun.

3.2.3. Risques liés aux performances

Recast.ai ne traite plus les requêtes	
Description	Recast.ai ne fournit plus de réponses suite à l'envoi de demande
Causes	Changement de politique de Recast.ai L'entreprise de Recast.ai dépose le bilan (possibilité car entreprise jeune) Serveur de Recast.ai devient hors service (suite à un hack ou à un dysfonctionnement, ...)
Conséquences	La demande de l'utilisateur ne peut plus être analysé, et par conséquent traitée
Moyens de prévention	Discuter régulièrement avec les responsables de Recast.ai, pour prévoir un éventuel changement de politique. Prévoir un message de réponse automatique à l'utilisateur pour le prévenir d'un problème technique. Prévoir des utilisations de concurrents comme Craft.ai
Gravité	Haut

Aucun moyen de communiquer avec le serveur hébergeant "Hello Suribot"	
Description	Le serveur est inaccessible
Causes	Composant électronique du serveur tombe en panne Coupure de la connexion internet avec le serveur Coupure électrique
Conséquences	Plus aucun lien possible entre l'utilisateur et "Hello Suribot"
Moyens de prévention	Vérifier l'état du serveur et des installations hébergeant le serveur régulièrement. Prévoir un serveur de secours.
Gravité	Moyen

3.2.4. Risques liés aux ressources

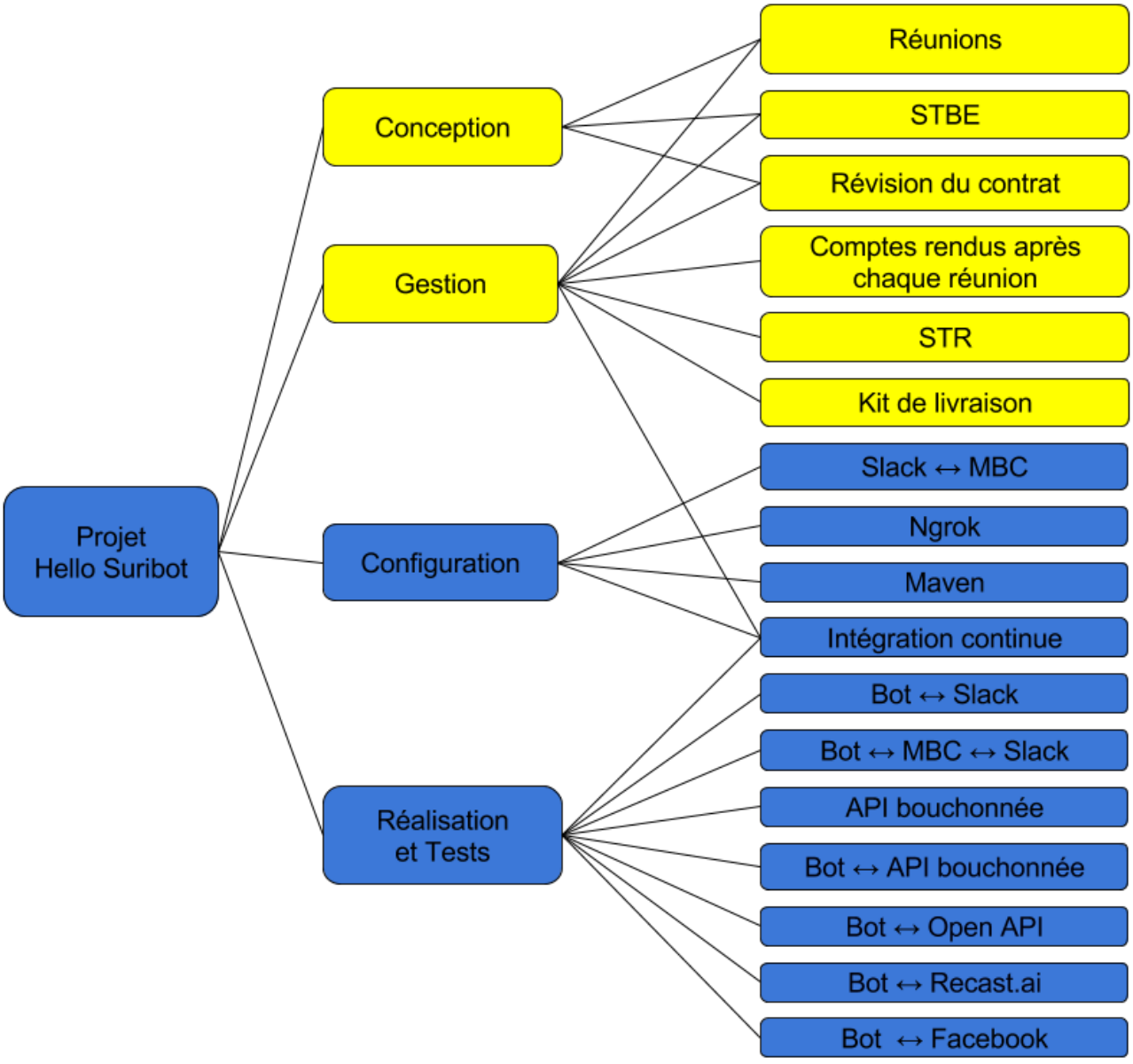
Abandon d'un ou plusieurs développeurs	
Description	Un ou plusieurs développeurs de l'équipe ont décidés de quitter le projet en cours de route.
Causes	Manque de motivation Lassitude du projet Changement de voie professionnelle Problème personnel
Conséquences	Les 2 autres membres de l'équipe doivent fournir le travail pour 3, causant d'éventuels retards.
Moyens de prévention	Discuter régulièrement entre les différents membres du groupe. Prévoir d'éventuels changements de contrat avec Suricats Consulting.
Gravité	Haut

Peu de temps consacré	
Description	Un (des) développeur(s) ne consacre(nt) pas suffisamment de temps
Causes	Manque de motivation Lassitude du projet Sous-estimation du travail à cause du manque d'expérience Problèmes personnels
Conséquences	Surcharge de l'équipe pendant certaines périodes, causant d'éventuels retards
Moyens de prévention	Travailler, discuter et réfléchir régulièrement. Se donner des marges.
Gravité	Haut

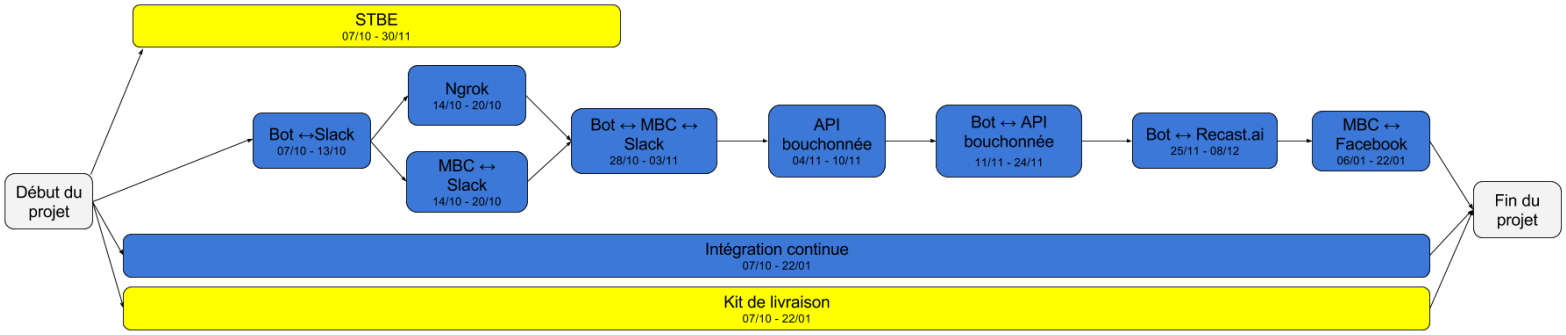
4. Planification

Légende :
↔ connexion entre éléments, MBC - Microsoft Bot Connector
Tâches en bleu - création et mise en marche du produit
Tâches en jaune - encadrement du travail

4.1. WBS



4.2. PERT



4.3. GANTT

	début	fin	durée	7-13 oct	14-20 oct	21-27 oct	28 oct - 3 nov	4-10 nov	11-17 nov	18-24 nov	25 nov - 1 déc	2 - 8 déc	9-16 déc	17 déc - 5 jan	6 - 12 jan	13 - 22 jan
Bot ↔ Slack	7 oct	13 oct	1 sem	100%												
Ngrok	14 oct	20 oct	1 sem		100%											
MBC ↔ Slack	14 oct	20 oct	1 sem		100%											
Intégration continue	7 oct	22 jan	13 sem	70%												
Bot ↔ MBC ↔ Slack	28 oct	3 nov	1 sem				100%									
API bouchonnée	4 nov	10 nov	1 sem					100%								
Bot ↔ API bouchonnée	11 nov	24 nov	2 sem						100%							
Bot ↔ Recast.ai	25 nov	8 déc	2 sem								60%					
MBC ↔ Facebook	6 jan	22 jan	2 sem												0%	
STBE	7 oct	30 nov	8 sem	100%												
Kit de livraison	7 oct	22 jan	13 sem	30%												

L'avancement se base sur la date du 30/11/2016. A cette date, tous les sprints n'ont pas encore été définis, la période de Décembre à Janvier est donc incomplète.

5. Annexes

5.1. Lexiques

1- **Un moteur d'analyse** est un programme qui traduit le message d'un utilisateur dans un message compréhensible par un bot (un arbre).

2- **Intent:** "Intents are boxes including expressions that mean the same thing but are constructed differently. Each intent correspond to one action your user want to perform".

3- **Un bot** est un agent conversationnel pouvant dialoguer avec un utilisateur.

4- **JSON** : ou JavaScript Object Notation, est un format de données textuelles permettant de représenter de l'information structurée**.**

5- **Une API** est un ensemble normalisé de classes, de méthodes ou de fonctions qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels. Elle est offerte par une bibliothèque logicielle ou un service web, le plus souvent accompagnée d'une description qui spécifie comment des programmes consommateurs peuvent se servir des fonctionnalités du programme fournisseur.

5.2. Bibliographie

- https://blog.recast.ai/nodejs-bot-tutorial/
- https://chatbotsmagazine.com/the-complete-beginner-s-guide-to-chatbots-8280b7b906ca
- https://docs.botframework.com/en-us/