Documentation de conception

Projet COB

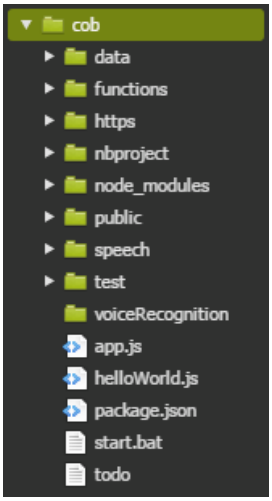
BONNARDEL Julien

Lundi 22 août 2016

Table des matières

I- Architecture

Voici ci-contre les différents fichiers et sous-dossiers de l'architecture de COB :

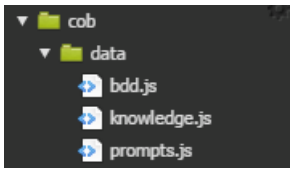


1- Fichiers racines

- Le **todo.txt** retranscrit le savoir-faire de COB, les différentes fonctionnalités implémentées pour le bon fonctionnement de COB ainsi que des indications d'améliorations possible.
- **start.bat** permet de lancer app.js (node app.js).
- **package.json** regroupe la description ainsi que les dépendances nécessaires pour le bon fonctionnement de COB.
- **helloWorld.js** est un fichier hello world de connexion au serveur. Le bot renvoi simplement « Hello World » à toutes demandes.
- **app.js** est le fichier central de COB. Il regroupe la fonction de connexion au serveur et les différents 'matchs' possible entre les dires de l'utilisateur et la technologie LUIS utilisé.

2- Data

Le sous dossier data regroupe les connaissances de COB (**knowledge.js** et **prompts.js**) et une fausse base de données (**bdd.js**).



- Les connaissances de COB sont représentées de la manière suivante :

‘theme’ : {

‘intitulé’ : ‘Bonjour’,

‘intitulé_2’ : ‘Il fait beau aujourd’hui’

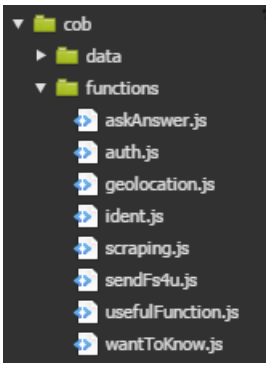
}

```
module.exports = {
  // Description d'entreprise
  'cgi': {
       description: 'Groupe CGI est un groupe canadien actif dans le domaine des services en technologies de l\'information et de l
       fondateur: 'Michael E. Roach',
       localisation: 'Le siège France se trouve à la défense ! (sinon c\'est au Canada)',
       website: 'www.cgi.fr',
       activité: 'Finance ...',
       solution: 'Pleins'
   },
   //---------------------------------------//

   // Paroles de bases
   'salutation': {
       matin : 'Bonjour, tu es matinal dis donc. Ou tu te couche tard, au choix, hein',
       midi: 'Salut ! J\'espère que tu as mangé',
       soir: 'Bonsoir',
       default: 'Bonjour'
   },

   'remerciment': {
       1: 'No problem !',
       2: 'Je suis gentil par nature donc je te réponds ...',
       3: 'Je t\'en pris !',
       4: 'Pas de soucis !'
   },

```

- Le fichier **prompts.js** a la même philosophie, il sert à répertorier des dires de fonctionnement.
- La base de données (**bdd.js)** est un fichier factice qui permet de tester COB en imaginant qu’il est connecté à une base de données.

3- Functions

Le sous dossier functions regroupe toutes les fonctions nécessaires au fonctionnement de traitement des demandes de l’utilisateur.



Toutes les fonctions (sauf usefulFunctions.js) sont développées sur le même principe :

```
var nomDeLaFonction = function(parametres){
    return function(session, results, next){
        //Votre code ici
    };
}
```

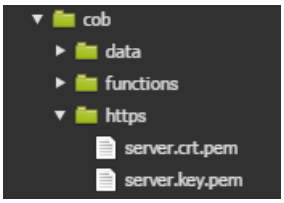
Cette structure est dû à la structure même du framework de microsoft utilisé. C'est-à-dire que le traitement d'une demande de l'utilisateur passe par une succession de function(session, results, next). Le terme ***session*** correspond à la session qui est active entre le bot et l'utilisateur, le ***results*** récupère les données (ou mots clé déterminé par LUIS) écrite par l'utilisateur. Le ***next*** (optionnel) permet d'envoyer une donnée spécifique à la fonction suivante.

Voici une description des fonctions implémentées.

- **askAnswer.js** regroupe deux fonctions, question(type) et reponse(champInfo). La première fonction permet une première analyse de la demande de l'utilisateur et regarde si COB en a la connaissance (knowledge.js). Cette fonction est en lien direct avec LUIS. La deuxième est une fonction réponse basique de COB si il possède la connaissance. Cette dernière n'est utile pour comprendre le fonctionnement de LUIS.
- **auth.js** permet l'authentification d'une personne par un code variable. (A améliorer par l'envoi du mot de passe sur l'appli FS4U)
- **geolocation.js** doit renvoyer la géolocalisation de la personne. (Non implémenté)
- **ident.js** permet l'identification de l'utilisateur. COB demandera le prénom et le nom de ce dernier.
- **scraping.js** permet de récupérer les news de google news. A améliorer pour la rendre plus fonctionnelle comme pouvoir choisir le site que l'on veut, ….
- **sendFs4u.js** (Non implémenté)
- **wantToKnow.js** permet à COB de vouloir apprendre des choses sur l'utilisateur si ce dernier pose plusieurs fois la même demande. Seule la partie news est implémentée.
- **usefulFunction.js** regroupe des fonctions utiles pour le débugger, voir si une date est valide, ….

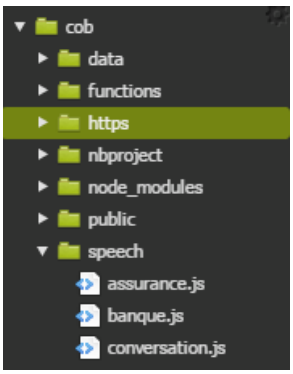
4- *https*

Ce dossier regroupe le certificat et la clé pour garantir du https et pouvoir notamment utiliser l'interface de microsoft (bot framework).



5- *Speech*

Ce dossier regroupe les fonctions de traitement des dires de l'utilisateur.



- **assurance.js** permet de traiter les demandes « Je suis en panne » (ainsi que ces variantes)
- **banque.js** traite les demandes « je peux faire un virement », « je veux voir mes comptes »
- **conversation.js** traite les demandes liée au news, au remerciement, à la santé, à l'entreprise CGI, à la salutation, etc. (voir todo.txt).

6- *Autres fonctionnalités*

Il y a la présence de quelles que autres fonctionnalités qui ne sont pas complètes ou désactivées dans le code principale.

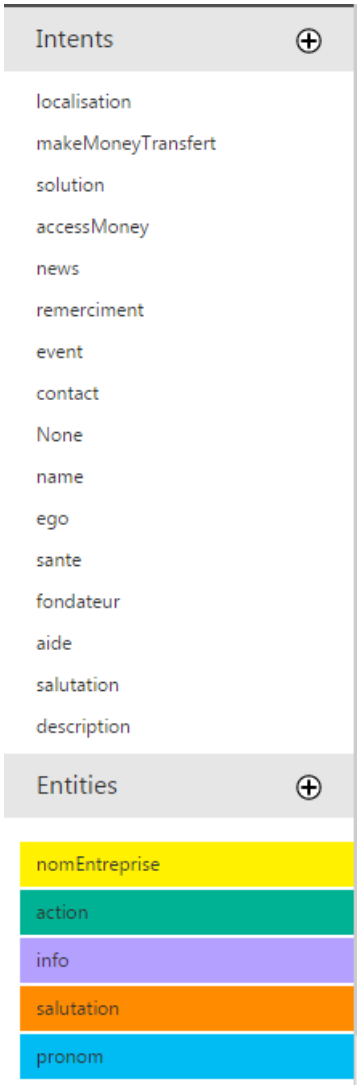
Une ébauche primitive de géolocalisation utilisant l'api de google se trouve dans le fichier **geolocalisation.js.** Cependant elle n'est pas fonctionnelle et nécessite un travail plus approfondie. De plus, une fonctionnalité qui vérifie la validité d'une date est opérationnelle, elle se trouve dans le fichier **usefulFunction.js** sous le nom de *isValidDate(string)*. Mais cette dernière est désactivé dans le code principal.

II- LUIS

1- *Description*

Luis est une manière intelligente d'analyser le langage humain et d'en comprendre le sens par un calcul de probabilité entre les différentes intentions (voir en annexe). Cette technique est développée par microsoft accessible sur le lien suivant : https://www.luis.ai. Lors de ce développement, il n'y a pas eu de mise en place d'une vraie structure. Le « modèle » exposé par la suite est une juxtaposition de scénarii envisagés lors de la découverte de cette API.

Le modèle de COB est le suivant :



Il se compose de différentes intentions liées à des situations auxquelles COB sera capable de répondre.

- Description d'une entreprise, ici CGI :

    * **description**

"Connais tu cgi ?", "Tu connais cgi ?", "qu'est ce que cgi ?", "cgi ?"

    * **fondateur**

"Qui a fondé cgi ?", "Qui est le fondateur de cgi ?", "le fondateur de cgi ?"

    * **localisation**

"Ou se trouve cgi ?", "Ou est cgi ?"

    * **contact**

"Comment puis je contacter cgi ?", "ou contacter cgi ?", "comment contacter cgi ?", "Comment je prend contact avec cgi ?"

    * **solution**

"Quelles solutions propose cgi ?", "Quelles sont les solutions de cgi ? "

- Salutation

    * **salutation**

"Bonjour", "Yo", "Yop", "Coucou", "wesh", "hello", "salut", ...

- Demande de nom

* **name**

"Comment je m'appelle ?", "Comment tu t'appelles ?", "Quel est mon nom/prénom", "Quel est ton nom/prénom"

Rq: Il ne fait pas la distinction entre nom et prénom

- Sante

* **santé**

"La forme ?", "Comment tu vas ?", "Comment vas tu ?", "Ca/ça va bien ?"

Rq: il répond et renvoi et toi ? Il ne comprendra pas la réponse comme ca va bien.

- Remerciment

* **remerciment**

"Merci", "Merci beaucoup", ...

Rq: Certaine tournure comme "Merci milles fois" il ne les comprend pas

- News

* **news**

"Quelles sont les news ?", "Quelles sont les infos ?", "Quelles sont les informations ?"

- Virement

* **makeMoneyTransfert**

"Je peux faire un virement", "Puis-je faire un virement ?", "Est ce que je peux faire un virement ?"

- Consultation de ces comptes

* **accesMoney**

"Je peux voir mes comptes", "Puis-je voir mes comptes ?", "Est ce que je peux voir mes comptes ?"

- Assurance

* **event**

"Je suis en panne"

2- Remarques

Lors de la simulation et du faite de l'organisation de ce « modèle », il y a un conflit entre l'intention de santé/none et la description. Autrement dit la description est désactivé dans le code. De plus l'intention aide est à revoir et est utile pour l'utilisateur lorsqu'il demande « Peux-tu m'aider ? » ainsi que ces variantes.

La notion d'entities a été très peu utilisée. Cette notion doit être développée pour utliser au mieux l'API LUIS de microsoft. Prenons un exemple :

« Connais-tu CGI ? »

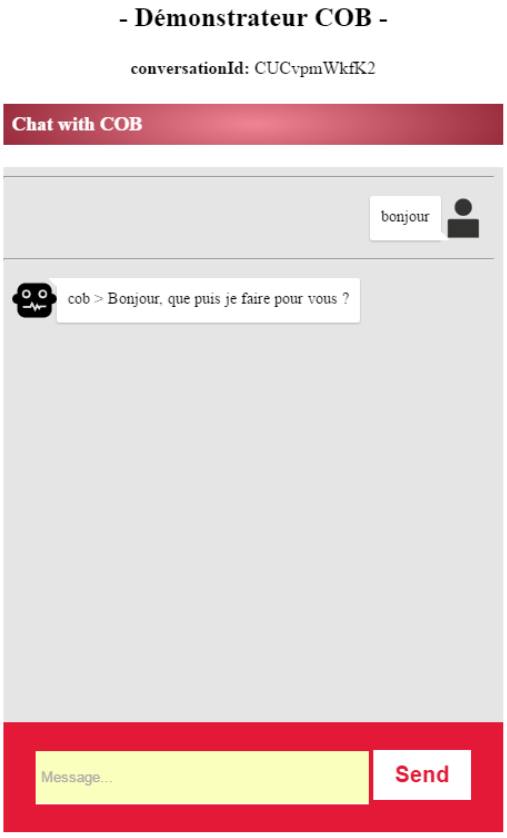
Intention de connaissance de quelque chose

Entities : nomEntreprise

Ceci permet de demander à COB « Connais-tu EBAY ? » et si COB a des connaissances sur EBAY il répondra.

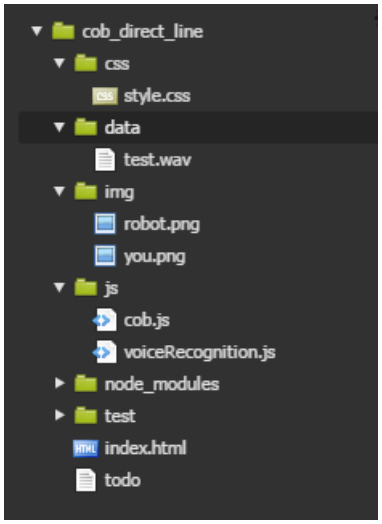
III- IHM

Une petit IHM a été développée. Cette dernière est simple et nécessite certains ajustements.



1- *Architecture*

Voici ci-contre les différents fichiers et sous-dossiers de l'architecture de l'IHM :



Cette ihm est réalisé sur une technologie HTML/CSS et javascript. Le fichier **index.html** est le fichier d'entrée. Ce dernier est lié au fichier **style.css** qui façonne le design de la page html et au fichier **cob.js** qui est le fichier de traitement. De plus le fichier **voiceRecognition.js** permet la mise en place de la reconnaissance vocale (text to speech et speech to text) en utilisant les APIs Bing de chez microsoft. Cependant lors de la rédaction de ce rapport, les services Bing de microsoft étaient indisponibles. La reconnaissance n'est donc pas fonctionnelle.

Le dossier *img* est un dossier classique qui loge les images nécessaires à l'ihm. Le dossier *data* permet de stocker les données liées à la reconnaissance vocale.

2- *Remarques*

L'ihm n'est pas parfaite esthétiquement notamment le bouton send qui n'est pas aligné. Le css mérite d'être retravaillé.

Annexes

Simulation de LUIS

Question : « Je veux faire un virement »


```
{
  "query": "je veux faire un virement ?",
  "intents": [
    {
      "intent": "makeMoneyTransfert",
      "score": 0.8873465
    },
    {
      "intent": "None",
      "score": 0.05921297
    },
    {
      "intent": "description",
      "score": 0.05428754
    },
    {
      "intent": "news",
      "score": 0.0441132523
    },
    {
      "intent": "name",
      "score": 0.0294190589
    },
    {
      "intent": "sante",
      "score": 0.02921893
    },
    {
      "intent": "contact",
      "score": 0.0184604134
    },
    {
      "intent": "ego",
      "score": 0.00661635026
    },
    {
      "intent": "solution",
      "score": 0.004537115
    },
    {
      "intent": "accessMoney",
      "score": 0.00397271337
    },
    {
      "intent": "salutation",
      "score": 0.00275831833
    },
    {
      "intent": "fondateur",
      "score": 1.38797631E-07
    },
    {
      "intent": "event",
      "score": 9.621711E-09
    },
    {
      "intent": "localisation",
      "score": 8.302712E-09
    },
    {
      "intent": "remerciment",
      "score": 3.24653726E-09
    },
    {
      "intent": "aide",
      "score": 1E-15
    }
  ],
  "entities": [
    {
      "entity": "virement",
      "type": "action",
      "startIndex": 17,
      "endIndex": 24,
      "score": 0.740783
    }
  ]
}
```