

# Développement d'une Application d'Assistance aux Personnes Agées pour le Robot Humanoïde NAO

Mohamed Adibe Chemaou, Xhersika Kenga, Sebastien Roach, Marie Legrand

Mai 2017

## Sommaire

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Le contexte et problème</b>	<b>2</b>
<b>3</b>	<b>La solution</b>	<b>2</b>
3.1	Définition des scénarios . . . . .	2
3.2	Architecture global du système . . . . .	2
3.3	Réalisation scénario d'interaction . . . . .	2
3.3.1	Appel météo . . . . .	2
<b>4</b>	<b>La gestion du projet</b>	<b>3</b>
4.1	Planning . . . . .	3
4.2	Organisation . . . . .	3
4.3	Difficultés et solutions . . . . .	3
<b>5</b>	<b>Conclusion</b>	<b>4</b>

# 1 Introduction

## 2 Le contexte et problème

## 3 La solution

### 3.1 Définition des scénarios

### 3.2 Architecture global du système

### 3.3 Réalisation scénario d'interaction

#### 3.3.1 Appel météo

Ce scénario était un des premiers scénarios réalisés. Le principe de ce fonctionnalite est très simple : Chaque fois que la personne âgée demande à NAO de savoir les conditions météorologiques actuelles, NAO lui donne toutes les informations nécessaires. En plus il donne des conseils pour la manière dont la personne doit s'habiller, par rapport au météo.

L'aspect technique de ce scénario est le suivant : Dans un box Python, on lit un fichier json qui contient toutes les donnes de météo courantes. Ce fichier est lu depuis un service météo en ligne , l'OpenWeatherAPI. En premier lieu, avant d'avoir commencé réaliser notre code, on a fait une inscription sur le site du notre fournisseur des donnes. Après cette inscription on récupère un user id qui est indispensable pour les prochaines étapes de réalisation.

Dans un premier temps a cree une fonction *url\_builder(cityid)*. Dans cette fonction on fait la génération d'une URL valide pour accéder nos donnes météo qui se trouve dans le site du OpenWeatherAPI. Notre lien complet pour accéder au fichier json est constitué de l'adresse api, ou on ajoute le mode de récupération des donnes(dans notre cas json) et notre user id.

Après avoir généré notre URL, il était nécessaire de récupérer notre fichier json. Pour faire cela on a créé une fonction *data\_fetch(URL)*, qui prenant en paramètre l'URL de notre fichier, retourne toutes les données que ce fichier contient. Les modules python nécessaires pour réaliser cette tâche étant : json et urllib2.

On utilise une fonction du module json appelée json.loads qui sert à decoder notre fichier json et le charger en mémoire.

La prochaine étape de réalisation de notre scénario était d'organiser toutes les données brutes récupérées de notre fichier json, pour ensuite être capable de les transmettre à notre robot, de manière qu'il puisse les communiquer sous forme orale à la personne. La fonction qui sert à faire l'organisation des données est *data\_organizer* qui prend en paramètre les données retournées par notre fonction précédente *data\_fetch*. Il s'occupe de la création d'un dictionnaire python, qui a comme clés la ville, température maximale, minimale etc. Il renvoie ce dictionnaire à la fin.

L'étape suivante, qu'on peut qualifier comme étant la plus essentielle, consiste à appeler une fonctionnalité de NAO et Coregraphe qui est très importante et fréquemment utilisée dans notre projet, le module ALTextToSpeech. Il renvoie du texte au moteur d'articulation de NAO. Pour utiliser ce module on se sert d'une ligne très simple : `tts = ALProxy("ALTextToSpeech", adresseDuRobot, portDuRobot)`. TTS(text-to-speech) est une variable instance de la classe ALProxy, fournie par Coregraphe. Cette classe permet de se connecter à un robot NAO pour notamment lui envoyer des chaînes de caractères à dire oralement.

Notre fonction *data\_output* prend alors en entrée notre dictionnaire créé précédemment. On utilise la variable tts qu'on appelle avec `tts.say`, la fonction du module ALTextToSpeech qui prend en paramètre un texte et traduit ce dernier en speech de notre robot.

Dans notre fonction de classe *onInput\_nStart* on appelle : `(data_organizer(data_fetch(url_builder())))`. Cette ligne permet de lier toutes les fonctions créées dans notre petit box python. À ce stade NAO est capable de dire clairement la météo actuelle quand la personne lui demande !

## 4 La gestion du projet

### 4.1 Planning

### 4.2 Organisation

### 4.3 Difficultés et solutions

## 5 Conclusion

