



Ecole Polytechnique de l'Université de Tours  
Département Informatique  
64 avenue Jean Portalis  
37200 Tours, France  
Tél. +33 (0)2 47 36 14 14  
[polytech.univ-tours.fr](http://polytech.univ-tours.fr)

## Projet Recherche & Développement 2021-2022

# Développement d'applications pour le robot Pepper



**POLYTECH<sup>®</sup>**  
**TOURS**

**Entreprise**

**Polytech**



**Tuteur entreprise**

**Usetech'Lab**

**Étudiant**

**Romain FRAGNIER (DI5)**

**Tuteur académique**

**Donatello CONTE**

3 avril 2022

# Liste des intervenants

## Entreprise

Polytech  
64 avenue Jean Portalis  
37200 Tours, France  
[polytech.univ-tours.fr](mailto:polytech.univ-tours.fr)



Nom	Email	Qualité
Romain FRAGNIER	<a href="mailto:romain.fragnier@etu.univ-tours.fr">romain.fragnier@etu.univ-tours.fr</a>	Étudiant DI5
Donatello CONTE	<a href="mailto:donatello.conte@univ-tours.fr">donatello.conte@univ-tours.fr</a>	Tuteur académique, Département Informatique
Usetech'Lab	<a href="mailto:livinglabrecherche@univ-tours.fr">livinglabrecherche@univ-tours.fr</a>	Tuteur entreprise



## Avertissement

Ce document a été rédigé par Romain FRAGNIER surnommé l'auteur.

L'entreprise Polytech est représentée par Usetech'Lab surnommé le tuteur entreprise.

L'Ecole Polytechnique de l'Université de Tours est représentée par Donatello CONTE surnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assume l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



## Pour citer ce document

Romain FRAGNIER, *Développement d'applications pour le robot Pepper:* , Projet Recherche & Développement, Ecole Polytechnique de l'Université de Tours, Tours, France, 2021-2022.

```
@mastersthesis{
  author={FRAGNIER, Romain},
  title={Développement d'applications pour le robot Pepper: },
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université de Tours},
  address={Tours, France},
  year={2021-2022}
}
```

# Table des matières

Liste des intervenants	<b>a</b>
Avertissement	<b>b</b>
Pour citer ce document	<b>c</b>
Table des matières	<b>i</b>
Table des figures	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1 Acteurs, enjeux et contexte .....	1
2 Objectifs .....	2
3 Hypothèses .....	2
4 Bases méthodologiques.....	2
<b>2 Description générale</b>	<b>3</b>
1 Environnement du projet .....	3
2 Caractéristiques des utilisateurs .....	3
3 Fonctionnalités du système .....	3
4 Structure générale du système.....	4
<b>3 État de l'art / Veille technologique</b>	<b>5</b>
1 Robot Pepper .....	5
Discussion .....	6
Mouvement .....	6
Perception.....	6
Connaissances .....	7

	Tablette tactile.....	7
	Système d'exploitation .....	7
	Utilisation de Pepper.....	8
2	La robotique au service des personnes âgées.....	9
<b>4</b>	<b>Analyse et conception</b>	<b>11</b>
1	Analyse .....	11
1.1	Hypothèses utilisées.....	11
1.2	Spécifications.....	11
2	Modélisation proposée.....	11
3	Évolution au semestre 10 .....	12
<b>5</b>	<b>Mise en oeuvre</b>	<b>13</b>
1	Développement sur émulateur Pepper .....	13
1.1	Outils et librairie utilisés .....	13
1.2	Éléments d'implémentation, choix techniques.....	14
1.2.1	Applications de discussions.....	14
1.2.2	Système speech to text, text to speech.....	16
1.3	IHM .....	17
1.4	Analyse des résultats, évaluation, qualité .....	18
2	Développement du chatbot en Python.....	18
2.1	Outils et librairie utilisés .....	18
2.2	Éléments d'implémentation, choix techniques.....	19
2.3	IHM .....	20
2.4	Analyse des résultats, évaluation, qualité .....	20
<b>6</b>	<b>Bilan et conclusion</b>	<b>22</b>
1	Bilan du semestre 9 .....	22
2	Bilan du semestre 10.....	22
3	Bilan sur la qualité .....	22
4	Bilan auto-critique.....	22
	<b>Annexes</b>	<b>24</b>
<b>A</b>	<b>Planification, gestion de projet</b>	<b>25</b>
1	Evolution du projet .....	25
2	Description des tâches.....	26
	Tâche 1 : Documentation .....	26
	Tâche 2 : Recherche pour la location d'un robot Pepper .....	26
	Tâche 3 : Installation de l'environnement de travail.....	26

Tâche 4 : Première réunion avec le client .....	26
Tâche 5 : Spécifications .....	26
Tâche 6 : Recherche d'alternative à Pepper.....	26
Tâche 7 : Deuxième réunion avec le client.....	27
Tâche 8 : Préparation rapport, soutenance .....	27
Tâche 9 : Mise à niveau robot Cruz.....	27
Tâche 10 : Recherche d'alternative pour le projet .....	27
Tâche 11 : Troisième réunion avec le client .....	27
Tâche 12 : Développement sur l'émulateur de Pepper .....	27
Tâche 13 : Développement du chatbot sur Python .....	27
<b>B Cahier de Spécifications</b> .....	<b>28</b>
1 spécifications Fonctionnelles.....	28
1.1 Discussion .....	28
1.1.1 Fonction nursing.....	28
1.1.2 Fonction détente.....	28
1.1.3 Fonction repas.....	29
1.1.4 Fonction endormissement.....	29
2 Spécifications non fonctionnelles .....	30
2.1 Modes de fonctionnement .....	30
3 Risques.....	30
3.1 Capacités de Pepper .....	30
<b>C Guide d'installation et d'utilisation de l'application pour l'émulateur de Pepper</b> .....	<b>33</b>
1 Introduction .....	33
2 Installation .....	33
2.1 OS, IDE .....	33
2.2 Plug-in Pepper SDK .....	34
2.3 Robot Application .....	34
2.3.1 Correction des dépendances .....	34
3 Utilisation .....	35
4 Astuces.....	35
5 Liens utiles .....	35
<b>D Guide d'installation et d'utilisation Chatbot Python</b> .....	<b>37</b>
1 Introduction .....	37
2 Installation .....	37
2.1 OS, IDE et version de Python .....	37
2.2 Packages.....	38
2.3 Correction de fonctions.....	39

3	Utilisation .....	40
3.1	Apprentissage .....	40
3.2	Fonctionnement .....	41
4	Liens utiles .....	42
<b>E</b>	<b>Bibliographie</b>	<b>43</b>



# Table des figures

<b>1</b>	<b>Introduction</b>	
1.1	Logo Usetech'Lab .....	1
<b>2</b>	<b>Description générale</b>	
2.1	Diagramme des cas d'utilisations .....	4
<b>3</b>	<b>État de l'art / Veille technologique</b>	
3.1	Pepper.....	5
3.2	Exemple de connaissance .....	7
3.3	Pepper dans une maison de retraite à Hanovre en Allemagne .....	8
3.4	Visioconférence avec Cutii.....	9
3.5	Séance de gym avec NAO .....	10
<b>5</b>	<b>Mise en oeuvre</b>	
5.1	Emulateur : aperçu 3D de Pepper .....	13
5.2	Applications de discussions : émulateur et fenêtre de discussion .....	16
5.3	Interface graphique sur la tablette tactile de l'émulateur .....	17
5.4	Interface de l'application Python .....	21
<b>A</b>	<b>Planification, gestion de projet</b>	
A.1	Diagramme de Gantt initial S9 .....	25
A.2	Diagramme de Gantt final S9.....	25
A.3	Diagramme de Gantt final S10.....	26

**B Cahier de Spécifications**

B.1 Bouton On/Off .....	30
B.2 Bouton On/Off de la tablette.....	31
B.3 Bouton d'arrêt d'urgence .....	31

**C Guide d'installation et d'utilisation de l'application pour l'émulateur de Pepper**

C.1 Nom de la tablette de Pepper dans Android Studio.....	36
C.2 Fermer l'application depuis le gestionnaire des tâche de la tablette .....	36

**D Guide d'installation et d'utilisation Chatbot Python**

D.1 Configurateur d'installation Python .....	38
D.2 Corriger l'erreur time.clock .....	40
D.3 Corriger l'erreur collections.hashable.....	40
D.4 Interface obtenue au lancement du programme.....	41

# 1

## Introduction

### 1 Acteurs, enjeux et contexte

Dans les maisons de retraite et plus particulièrement dans les Ehpad, un grand nombre de résidents manquent d'autonomie dans les tâches quotidiennes, manquent de repère ou bien souffrent de solitude. Cela est dû d'une part à la forte charge de travail du personnel, qui n'a pas forcément le temps de discuter longuement avec tous les résidents. D'autre part cela est dû au peu de visites que reçoivent certains résidents, ce qui a été fortement amplifié par la crise sanitaire. L'axe principal de ce projet est donc le maintien et le développement de l'autonomie ainsi que des liens sociaux des personnes âgées.

Dans le cadre de ses recherches sur les enjeux sociaux de l'intelligence artificielle et des outils numériques sur la santé, le Usetech'Lab souhaite étudier l'impact qu'un robot dédié aux interactions avec l'homme pourrait avoir sur les résidents de maisons de retraite de type Ehpad et sur le travail du personnel de ces établissements.

Le Usetech'Lab, situé à Tours, est un laboratoire de recherche spécialisé dans l'intelligence artificielle et la santé dans les sciences humaines et sociales. Pour ce projet, il a fait appel à deux Ehpad pour déterminer comment le robot pourrait être utilisé en conditions réelles et à Polytech Tours pour la réalisation de la partie technique.



Figure 1.1 – Logo Usetech'Lab

Acteurs :

— Client : Usetech'Lab, Ehpad La Charmée (Châteauroux), Ehpad Les 5 rivières (Vierzon).

- MOE : Romain FRAGNIER, DI5 Polytech Tours
- MOA : Donatello Conte, Polytech Tours

## 2 Objectifs

L'objectif principal est de développer des applications pour le robot Pepper, dans le but d'interagir avec les résidents via la parole, l'écoute, le mouvement. Ainsi les résidents pourront interagir avec Pepper lorsque ce ne sera pas possible de le faire avec du personnel, car ce dernier est très occupé et n'a pas forcément le temps de discuter régulièrement avec tout le monde. Les résidents auront donc une possibilité supplémentaire pour maintenir des liens sociaux.

L'objectif est également de tester les applications en déployant Pepper dans un établissement, pour savoir si les applications sont fonctionnelles, et si elles sont appréciées par les résidents et le personnel.

## 3 Hypothèses

Le robot utilisé est le robot Pepper dans sa version 2.9 QiSDK. L'environnement de travail est Android studio et le langage utilisé est Java. La plateforme de développement est ouverte et Softbank Robotics, le fabricant de Pepper, fournit plusieurs API [16] pour faciliter le développement, ainsi qu'un émulateur sous Android Studio permettant de programmer sans avoir un vrai robot à disposition.

Cependant, réussir à louer un robot Pepper n'étant pas garanti, l'utilisation d'un autre robot pour le projet n'est pas à exclure. L'environnement de travail ainsi que le langage de programmation risquent donc de changer. Le cas échéant, il sera nécessaire de prendre en main et de se mettre à niveau pour le nouvel environnement de travail, entraînant le décalage du planning prévu.

Le choix définitif du robot pour le projet sera fait courant janvier 2022.

## 4 Bases méthodologiques

Comme il est difficile de se projeter, à cause de la location du robot et du fait que l'on ne sait pas si les applications vont plaire aux résidents ou vont fonctionner correctement, il a été décidé de mener le projet par bloc. Cela s'apparente à l'utilisation d'une méthode Agile. Il a été décidé avec le client d'axer le premier bloc sur la discussion.

# 2

## Description générale

### 1 Environnement du projet

Ce projet a été lancé il y a plusieurs années. Un cahier des charges avait été produit, exposant notamment différents scénarios de conversations entre le robot et les résidents, différentes fonctions et leurs objectifs. Le cahier des charges a été réutilisé pour rédiger le cahier des spécifications du projet.

Puis durant l'année 2020-2021, un PRD ayant pour but de réaliser une veille technologique sur les robots susceptibles de correspondre aux attentes du projet avait été mené. Le candidat principal qui est ressorti de cette veille est le robot humanoïde Pepper. Ce PRD avait également permis de mettre en lumière les besoins du personnel des Ehpad vis à vis du robot.

### 2 Caractéristiques des utilisateurs

On distingue deux types d'utilisateur. Le résident et personnel. Dans les deux cas le robot doit être simple d'utilisation et ne doit requérir aucune connaissance en informatique. Le résident peut utiliser les applications du robot. Le personnel peut utiliser les applications. Il peut aussi éteindre ou allumer le robot, le mettre en charge, choisir la fonction utilisée.

### 3 Fonctionnalités du système

Les fonctionnalités du premier bloc sont focalisées sur la discussion entre le robot et l'humain. C'est en effet ce qui semble être le plus accessible du point de vue développement tout en étant potentiellement un des aspects ayant le plus fort impact sur les résidents. Le choix des quatre fonctions suivantes a été fait lors de la première réunion avec le client et en utilisant le cahier des charges cité précédemment.

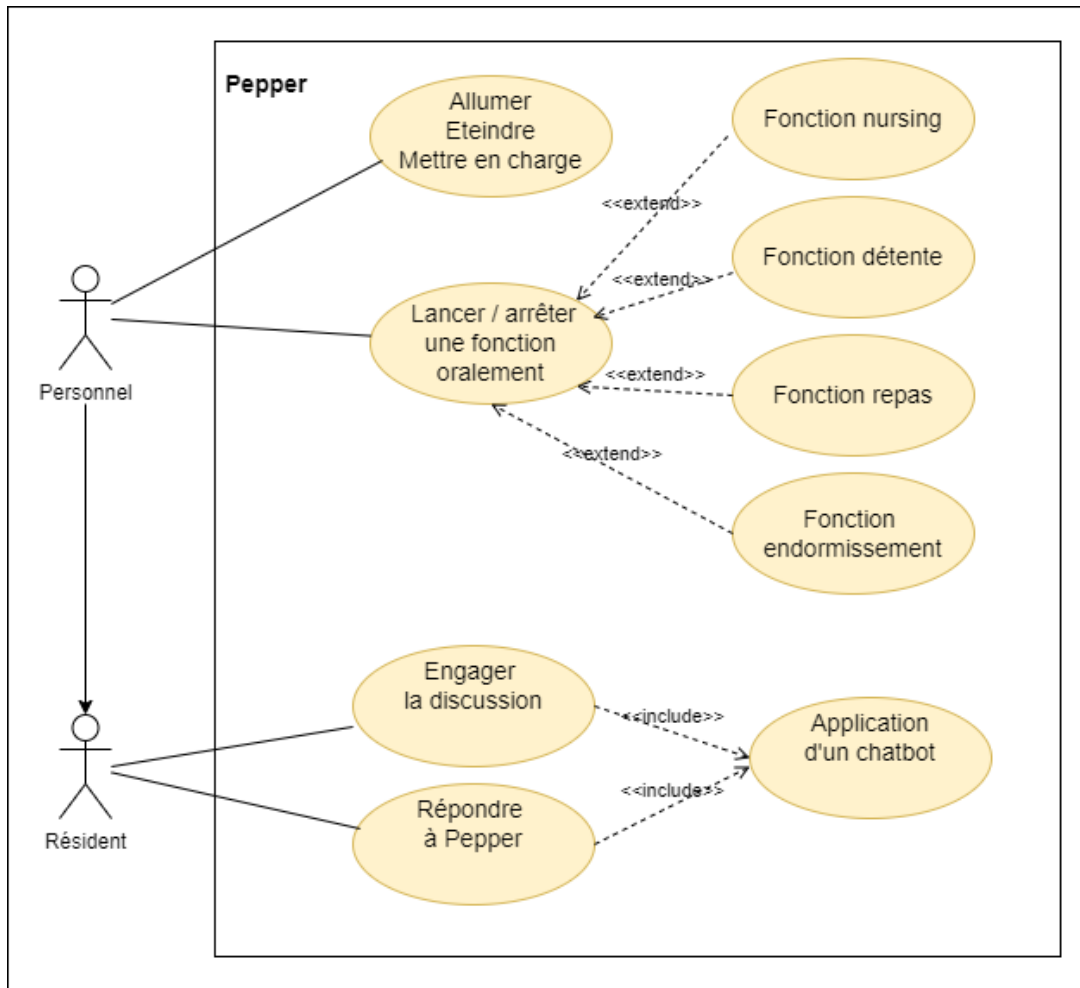


Figure 2.1 – Diagramme des cas d'utilisations

## 4 Structure générale du système

Lorsqu'il est mis en route, le robot est en mode « attente » (le robot ne fait rien). C'est depuis ce mode que l'on choisit la fonction à utiliser, via la voix. Quand une fonction est lancée, elle peut être arrêtée via la voix. Quand la fonction est arrêtée, le robot retourne en mode « attente », une autre fonction peut être lancée.

Pour la discussion, le principal élément est le chatbot. Le chatbot va déterminer en fonction de la situation ce que Pepper doit dire. Il est implémenté grâce à l'API « Conversation ». Chacune des quatre fonctions va implémentée un chatbot différent. Ainsi les répliques des différentes fonctions ne seront pas mélangées.

Si ce que dit l'utilisateur ne fait pas partie des phrases auxquels Pepper sait répondre, Pepper répond : « Je n'ai pas compris ce que vous avez dit ».

# 3

## État de l'art / Veille technologique

### 1 Robot Pepper

Pepper est un robot humanoïde développé par la société Softbank Robotics et a été commercialisé entre 2015 et 2021. C'est un robot émotionnel. Son objectif est d'interagir avec l'humain d'une manière naturelle, via la voix, les mouvements, les émotions.

En termes de caractéristiques physiques, Pepper mesure 1m20 et pèse une trentaine de kilogrammes. Il est équipé de trois roues à sa base pour se déplacer. Il est fabriqué en plastique souple pour minimiser les chocs et les pincements. Il est doté de nombreuses capacités, dont cinq que l'on pourrait qualifier de capacités principales.

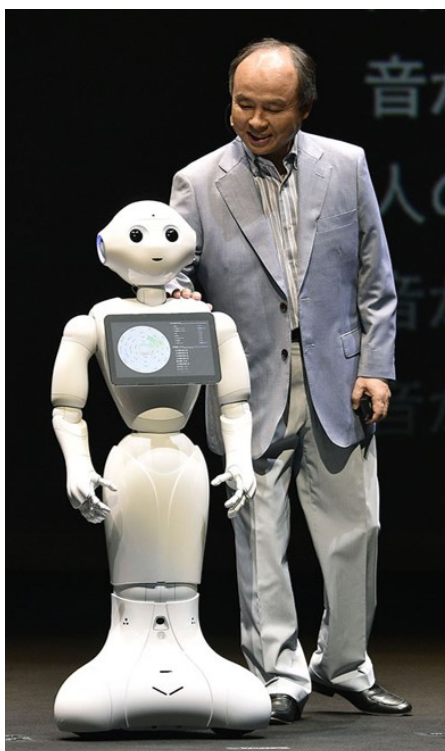


Figure 3.1 – Pepper

### Discussion

La discussion est la première de ses capacités. Grâce à des micros et à un synthétiseur vocal, Pepper peut écouter et parler. Pour qu'il puisse prendre part à une conversation, il faut implémenter un chatbot. Un chatbot est un programme qui gère l'ensemble des répliques, des phrases que Pepper doit savoir reconnaître et les réponses qu'il doit donner. Il existe beaucoup de chatbots différents. Il est possible d'en utiliser un ou d'en développer un grâce aux outils mis à disposition par des entreprises comme Microsoft, IBM ou Google ou d'autres entreprises spécialisées. Certains sont gratuits, d'autres sont payants. Certains utilisent de l'intelligence artificielle pour gérer en temps réel les réponses du robot.

Une autre solution est d'en développer un soi-même via l'API "Conversation" fournie par Softbank Robotics, qui possède des outils permettant de développer efficacement un chatbot. Dans ce cas là il est nécessaire d'écrire à l'avance dans le code les différentes répliques de Pepper, car cette API ne permet pas de développer un chatbot utilisant de l'IA.

Pepper est donc capable de participer et de suivre une conversation. La complexité du ou des chatbots utilisés permettra d'enrichir la conversation de Pepper et de réagir à plus ou moins de situation. Il est également possible de donner des ordres oraux à Pepper pour qu'il réalise une action particulière.

### Mouvement

Le mouvement est une des forces de Pepper. Il est équipé de nombreuses articulations au niveau des épaules, des coudes, des poignets, des mains, de la tête, des genoux et de la taille.

Cela lui permet d'effectuer des animations. Une animation est une chorégraphie qui dure quelques secondes. Cela peut être une danse, une imitation d'animaux, des salutations ou bien des exercices comme des étirements. Plusieurs dizaines d'animations sont fournies par Softbank Robotics, mais il est possible de créer soi-même des animations. Les possibilités sont donc très nombreuses.

Lorsque Pepper se déplace, il adapte son parcours et sa vitesse grâce à des capteurs lasers, des caméras et des sonars placés tout autour de lui. Il peut cartographier l'endroit dans lequel il se trouve pour s'y déplacer plus facilement ensuite. On peut par exemple lui demander de se souvenir de la location d'un objet ou d'une personne en particulier, pour ensuite lui demander d'y retourner plus tard.

Il possède une capacité nommée "Autonomous abilities", lui permettant d'imiter des mouvements humains naturels comme des légers mouvements de bras et de tête ou des clignements des yeux grâce à des LEDs.

Bien que ses mains soit articulées, Pepper n'est pas conçu pour prendre ou ramasser des objets. Soit sa main est à plat, soit elle est refermée, ses doigts formant alors un cercle. Même si à des fins de recherche un objet ayant la bonne forme et le bon poids a déjà été placé dans sa main, il est difficile d'imaginer comment exploiter concrètement cela.

### Perception

La perception est un domaine dans lequel Pepper est un des robots les plus avancés, car jusqu'à présent ce type de robot était rare et Pepper est un des précurseurs.

Des capteurs tactiles sur ses mains et sa tête permettent d'adapter son comportement en fonction de ce que l'on souhaite faire. Une caméra située au niveau de son front lui permet de détecter les émotions de bases, la joie, la peur, la surprise, la tristesse, sur le visage d'une personne. Là encore cela permet de nuancer le comportement de Pepper. Grâce à cette caméra Pepper peut prendre des photos et aussi suivre du regard la personne avec laquelle il parle, rendant l'interaction plus agréable.

Il peut également se déplacer librement dans un endroit, repérer une personne et engager spontanément la conversation avec elle.



### Connaissances

Pepper est capable de retenir des connaissances issues de ses conversations, ou rentrées manuellement, pour les réutiliser plus tard. Il peut associer un nom à un visage ou retenir des connaissances plus poussées. Ces connaissances sont composées de trois éléments : un sujet, un type d'information et l'information, suivant le schéma ci-dessous.



Figure 3.2 – Exemple de connaissance

### Tablette tactile

La dernière des capacités principales de ce robot est la tablette tactile accrochée à son torse. Grâce à l'accès internet dont Pepper est doté, la tablette permet d'accéder à des vidéos ou des informations très facilement.

Il est aussi possible d'afficher les photos prises par le robot ou de jouer à des jeux. Une des utilisations les plus courantes est la retranscription de ce que Pepper dit et entend, ainsi que l'état dans lequel il se trouve (s'il est en état d'écoute ou s'il est en état de parole) grâce à un bandeau lumineux en haut de l'écran. Lorsque la tablette est utilisée, Pepper passe en mode "tablette", le rendant statique pour éviter tout mouvement dérangent.

Dans tous les cas Softbank Robotics préconise d'utiliser la tablette pour accompagner les autres capacités du robot et non pas de l'utiliser comme une tablette classique.

### Système d'exploitation

Pepper existe sous deux versions différentes d'un système d'exploitation. La version 2.5 NAOqi et la version 2.9 QiSDK. Softbank a produit un article comparant les deux versions. [3]

La version 2.5 est programmable via l'environnement de développement Choregraphe et les langages Python ainsi qu'HTML et Javascript pour la tablette.

Cette version est plus adaptée pour le développement de programmes de robotique avancée. En effet elle donne accès à plus d'API, dont plusieurs relatives au fonctionnement des capteurs. La possibilité d'implémenter des scripts en Python permet plus de flexibilité dans les comportements et les programmes. Cependant cette version n'est plus mise à jour et manque parfois de stabilité, rendant difficile le déploiement du robot pour une utilisation concrète avec de vrais utilisateurs.

La version 2.9 est la version évoquée dans la partie "Hypothèses" de l'Introduction. Dans cette version le développement se fait via Android Studio et le langage Java (ou Kotlin).

Cette version est plus adaptée pour développer des programmes d'interactions entre le robot et l'humain. En effet c'est une évolution de la version 2.5 qui a été conçue dans le but d'améliorer la stabilité et la fiabilité pour pouvoir déployer le robot en conditions réelles. Elle ne donne accès qu'à un nombre limité d'API et ne conserve que les fonctionnalités principales.

Il est donc plus facile de développer des applications pour l'interaction, mais les possibilités en termes de robotique avancée sont moindres. Donc dans le strict cadre du projet, la version 2.9 semble plus adaptée.

Mais dans les deux cas les possibilités sont quand même globalement identiques et les deux versions seraient capables de répondre aux besoins du projet. Un émulateur est disponible pour chaque version rendant possible le développement sans avoir un vrai robot. Les plateformes de développement sont ouvertes, il y a beaucoup de ressources et d'informations concernant les APIs, les possibilités [15] [14].

### Utilisation de Pepper

Pepper est utilisé dans plusieurs domaines. Certaines entreprises se sont spécialisées dans le développement d'applications pour ces différents domaines, et vendent ou louent des robots équipés de leurs applications.

Il est utilisé dans la recherche, pour développer des programmes de robotique ou d'intelligence artificielle. Notamment pour des programmes de reconnaissances d'images, où Pepper doit repérer une personne tomber au sol [18] ou bien pour apprendre à Pepper à jouer au bilboquet via l'apprentissage par essai erreur.[19]

Il est utilisé à des fins commerciales, en tant qu'hôte d'accueil pour donner des informations aux clients, les aiguiller vers l'endroit qu'ils cherchent, les questionner pour les conseiller ou encore leur présenter des produits ou des services. On le retrouve dans des concessions automobiles [10], des hôtels[20], des magasins [5], ou bien dans des salons ou des événements éphémères. Dans ces situations, ce sont ses capacités de discussion, de perception et de mouvement couplées à sa tablette qui sont les plus plébiscitées.

Enfin il est utilisé dans le domaine de la santé et du social. Dans les hôpitaux et les maisons de retraite, il peut servir à distraire les patients, discuter, jouer avec eux, les orienter ou encore permettre des visioconférences avec la famille et les proches grâce à sa tablette. [11]



**Figure 3.3** – *Pepper dans une maison de retraite à Hanovre en Allemagne*

Ce domaine vient compléter l'utilisation dans la recherche, car ici il va être intéressant d'étudier le comportement des humains vis à vis des robots.

Néanmoins, si la production de Pepper a été stoppée à l'été 2021, c'est en partie car Pepper s'est révélé peu efficace, peu fiable dans bon nombre de situations. A de nombreuses reprises il a connu des problèmes de fiabilité, ou bien son intérêt a été beaucoup moins grand que prévu. Les utilisateurs ne savaient pas toujours comment se comporter avec le robot, ou certaines fonctionnalités ne fonctionnaient pas correctement. Cela a conduit à une baisse de la demande qui a entraîné l'arrêt de la production. [8] [9]

Si Pepper n'a pas fait l'unanimité, c'est d'une part car bien souvent les attentes placées en lui étaient trop hautes comparé à ce qu'il est réellement capable de faire. Son allure humanoïde pourrait laisser croire qu'il est capable d'agir exactement comme un humain. Hors, il ne faut pas oublier que tout ce qu'il fait doit être programmé à l'avance, il est donc normal que parfois Pepper ne soit pas capable de faire face à une situation, de répondre à une question.

D'autre part c'est parce que Pepper est un des précurseurs dans son domaine. Ses technologies ne sont pas totalement abouties, des bugs ou des problèmes peuvent survenir. On sait notamment

que le bruit ambiant affecte sa capacité à entendre son interlocuteur, ou bien qu'il n'arrive pas très bien à parler avec plusieurs personnes en même temps.

## 2 La robotique au service des personnes âgées

Alors que la robotique est en perpétuel développement, elle est de plus en plus utilisée pour l'aide aux personnes âgées. Elle peut être utile pour aider les personnes seules dans les tâches quotidiennes et donc augmenter leur autonomie. C'est le cas du robot Buddy. Il peut tenir une conversation, diffuser de la musique, rappeler la date du jour. Il peut aussi surveiller le logement et contacter de l'aide en cas de problème. [2]

Il y a également des robots ayant pour objectif d'agir comme un animal de compagnie, sans en avoir les contraintes. Ces robots permettent de fournir de l'affection, d'avoir un compagnon de jeu.

Il y a les robots de télé-présence. Via une commande simple, le senior peut contacter un proche dont le numéro a été enregistré, pour lancer une visioconférence. Le proche peut commander le robot et le déplacer.

Les hôpitaux et les maisons de retraite sont sûrement les lieux dans lesquels ces robots sont les plus utilisés. La crise sanitaire a fortement contribué à cette tendance. Alors que les visites étaient interdites, les robots ont permis aux résidents et aux patients de conserver un minimum de lien sociaux et un contact avec leurs proches. [6] De plus, l'hébergement et l'accompagnement des personnes âgées demandent beaucoup de personnel. Les robots peuvent donc alléger la charge de travail de ce personnel, en distrayant les résidents pour faciliter les soins ou en les rassurant quand le personnel n'est pas disponible.

Plusieurs robots sont majoritairement utilisés dans ces lieux. Cutii est un robot de télé-présence à commande vocale. Utilisé dans beaucoup d'Ehpad, il permet de contacter les proches.

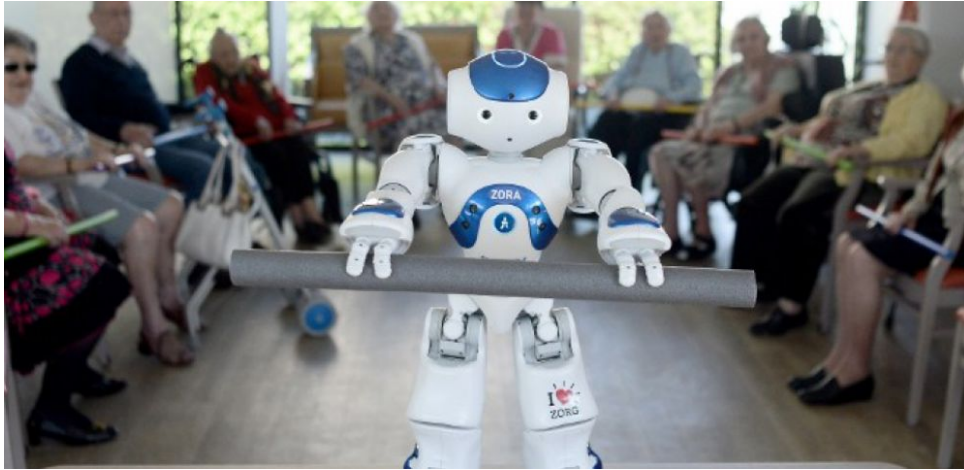


Figure 3.4 – Visioconférence avec Cutii

Il est également utilisé pour faire des activités à distance avec un intervenant. Par exemple une séance de sport en visioconférence avec un coach [7], ou une visite de musée à distance. Cutii possède également des fonctions de divertissement, d'alertes en cas de problème.

NAO est un autre des robots les plus utilisés. Il ne mesure que 58 centimètres de haut mais possède de nombreuses capacités. Il peut discuter, réaliser des animations. Il est utilisé pour parler avec les résidents, donner les nouvelles du jour qu'il récupère grâce à sa connexion internet, ou bien encore donner des cours de gym. [1]

On peut voir sur cette image l'inscription Zora sur le torse de NAO. Zora Robotics est une des entreprises qui développent des applications, notamment pour NAO et Pepper.



**Figure 3.5** – *Séance de gym avec NAO*

Comme indiqué dans la partie précédente, Pepper est déployé dans des maisons de retraite. Son apparence humanoïde rend les interactions plus agréables, et sa tablette tactile permet plus de choses qu'avec NAO. Les résidents peuvent par exemple jouer à des jeux simples, ou naviguer sur internet.

Dans tout les cas, les robots ne sont pas une solution miracle aux problèmes d'autonomie et de lien sociaux. Une bonne partie du temps, la présence d'une personne est nécessaire pour faire fonctionner le robot (lancer les applications, réparer le robot en cas de bug...).

# 4

## Analyse et conception

### 1 Analyse

#### 1.1 Hypothèses utilisées

Le robot utilisé est Pepper dans sa version 2.9 QiSDK. L'environnement de développement est Android Studio et le langage utilisé est Java.

Cependant la modélisation proposée ci-dessous n'est pas vraiment dépendante de ces hypothèses.

#### 1.2 Spécifications

Comme précisé plus en détails dans le cahier des spécifications en annexe C, il y aura quatre fonctions, axées sur la discussion. La fonction nursing, la fonction repas, la fonction divertissement et la fonction endormissement. Une seule fonction est utilisée à la fois.

Chaque fonction comprend ses propres scénarios de discussion entre le robot et les résidents, gérés par des chatbots.

Mais le comportement du robot change également selon la fonction en cours d'utilisation. Le comportement inclut la manière dont Pepper se déplace et la manière dont il engage la conversation avec les résidents.

### 2 Modélisation proposée

Un chatbot est un programme qui gère la conversation du robot, en se basant sur un ou des fichiers contenant les répliques du robot et ce qu'il doit savoir entendre. Le tout est écrit suivant une syntaxe stricte. Ce sont les règles utilisateur.

Il y a donc quatre fichiers contenant les répliques du robot, un pour chaque fonction.

Lorsque le robot est en stand-by et qu'il entend une des commandes pour lancer une fonction, cela provoque l'appel à une méthode portant le nom de la fonction.

Cette méthode va à son tour appeler plusieurs autres méthodes.

Une de ces méthodes va, grâce à l'API "Conversation" de Pepper, instancier un chatbot en utilisant le bon fichier de répliques.

Une autre méthode va adapter le comportement du robot via l'API "Motion" pour définir si Pepper doit se déplacer ou rester statique.

Une autre méthode va via l'API "Perception" définir si Pepper doit engager la conversation de lui-même ou s'il doit attendre qu'on vienne lui parler pour répondre.

Enfin une méthode va s'occuper de la gestion de la tablette.

Une commande oral permet de quitter la fonction en cours et de retourner en stand-by.

### 3 Évolution au semestre 10

La modélisation évoquée ci-dessus n'a pas pu être mise en œuvre au semestre 10. Cela est dû au fait que nous n'avons pas réussi à louer un robot pour des raisons financières. En effet, que ce soit le robot Pepper ou le robot Cruzr, les devis que nous avons reçus étaient d'environ 2000€ par mois pour la location. Ce montant est trop important et ni l'université ni le laboratoire ne pouvait engager une telle somme sans l'avoir budgétiser auparavant.

Nous avons discuté avec M.Conte de la suite du projet. L'obtention d'un robot doit attendre au minimum l'année scolaire prochaine.

Nous avons donc décidé de proposer une solution alternative temporaire au laboratoire, en attendant l'obtention d'un robot. Cette solution consiste à développer les fonctionnalités de discussions détaillées ci-dessus sur l'émulateur de Pepper, en rajoutant un système de reconnaissance et de synthèse vocale speech to text/text to speech (STT/TTS dans la suite du rapport). En effet, de base, la discussion avec l'émulateur se fait à l'écrit. Le rajout de ce système permettrait donc de passer la discussion à l'oral et la rendre beaucoup plus naturelle. La seule différence avec un vrai robot serait donc le fait que l'on discute avec un robot virtuel sur un ordinateur plutôt qu'avec un vrai robot.

Nous nous sommes réunis avec le laboratoire pour discuter de cette solution. À leurs yeux, le fait qu'il n'y ait pas de vrai robot rend l'intérêt des interactions avec les personnes âgées beaucoup plus faible. Discuter avec un ordinateur sur lequel se trouve l'émulateur n'apporte en réalité pas de plus-value par rapport à un chatbot comme Siri ou Alexa et ils considèrent que le projet ne pourra avancer que lorsqu'il y aura un robot.

Nous avons décidé avec M.Conte que j'allais quand même développer les applications de discussions sur l'émulateur de Pepper, en ajoutant à cela le système STT/TTS pour se rapprocher au plus de l'expérience que l'on aurait avec un vrai robot.

L'utilisateur appuie sur un bouton sur la tablette du robot. Tout en restant appuyé, il parle. Une fois qu'il lâche le bouton, ce qu'il vient de dire est converti en texte. Ce texte est passé au chatbot, qui renvoie sa réponse. Cette réponse est convertie en speech et est récitée par un synthétiseur vocal.



# 5

## Mise en oeuvre

### 1 Développement sur émulateur Pepper

#### 1.1 Outils et librairie utilisés

L'émulateur de Pepper est un émulateur Android que l'on démarre depuis Android Studio. C'est un émulateur particulier car il comprend une émulation de la tablette de Pepper ainsi qu'un aperçu 3D en temps réel du robot, mais également une fenêtre représentant la discussion avec Pepper ou bien les logs de l'application.

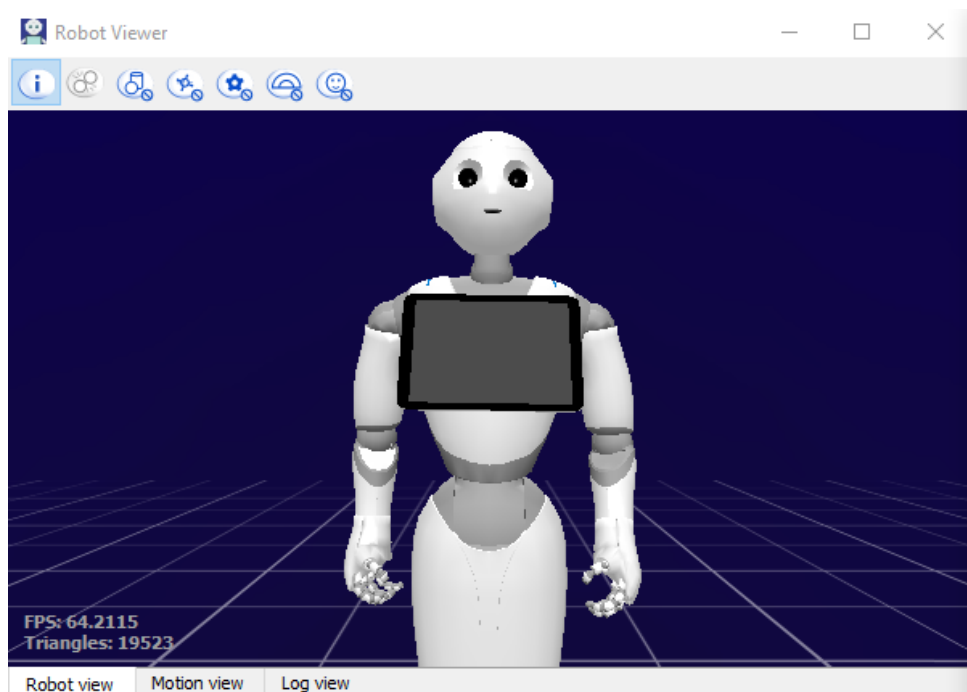


Figure 5.1 – Emulateur : aperçu 3D de Pepper

Pour l'utiliser il faut installer le plug-in Pepper SDK depuis Android Studio. Ce plug-in contient l'émulateur mais également tous l'environnement, les dépendances, les packages nécessaire au développement d'une application pour le robot.

L'application est également particulière. C'est une application qui reprend l'architecture classique d'une application Android, mais qui implémente des méthodes, des fonctions particulières. Comme dans une application Android classique, on retrouve le fichier principale "MainActivity.java" contenant les méthode onCreate(Bundle savedInstanceState) et onDestroy().

Mais on retrouve en plus les méthodes "onRobotFocusGained(QiContext qiContext)", "onRobotFocusLost()" et "onRobotFocusRefused()". Ces fonctions sont liées au cycle de vie d'une application Pepper, qui est adapté sur le cycle de vie d'une application classique.[13]

L'installation du plug-in ainsi que la création d'une application sont détaillées dans le guide d'installation en annexe. On peut aussi retrouver les tutoriaux que j'ai suivi sur le site de SoftBank Robotics.[12] Le site contient globalement de très nombreuses ressources, tutoriaux, documentations facile d'accès et extrêmement utiles pour le développement.

Néanmoins il est important de noter un chose : l'émulateur de Pepper n'offre pas toutes les possibilités d'un vrai robot. En effet, les capacités de perception (suivi du regard, déplacement, engagement de la conversation, utilisation de la caméra) ne sont pas disponibles. Cela contraint donc certaines des fonctionnalités qui devaient être développées, comme le fait que le robot devait se déplacer de table en table lors des repas pour encourager spontanément les résidents. A la place, l'émulateur encouragera un utilisateur si ce dernier parle à Pepper. De même pour la fonction détente où c'est l'utilisateur qui doit engager la conversation.

Pour les fonctionnalités de reconnaissance et de synthèse vocale du système STT/TTS, j'ai utilisé le package android.speech, car il est natif à Android Studio.

## 1.2 Éléments d'implémentation, choix techniques

### 1.2.1 Applications de discussions

La première étape a été le développement des applications de discussions.

Conformément aux spécifications, le choix de la fonction (détente, nursing...) doit se faire au démarrage d'application.

C'est la fonction onRobotFocusGained(QiContext qiContext) qui est automatiquement lancée au démarrage de l'application.

Au début de cette fonction le robot va annoncer les quatre fonctions de discussions disponibles. Puis il va se mettre à l'écoute d'un des noms des fonctions.

Les objets utilisés (Say, PhraseSet, Listen...) sont des objets propres au plug-in Pepper. Ils font partie des APIs décrites précédemment.

Enumération des fonctions et écoute de la fonction à exécuter :

```

1 // Create a new say action.
2 Say say = SayBuilder.with(qiContext)
3     // Set the text to say.
4     .withText("Detente, soins, repas, endormissement")
5     // Build the say action.
6     .build();
7 // Execute the action: list the four functions
8 say.run();
9
10 // Build phraseSet corresponding to the names of the functions
11 PhraseSet detente=PhraseSetBuilder.with(qiContext)

```



```

12 .withTexts( "Detente", "Fonction détente", "Détente", "Fonction detente")
13 .build();
14 PhraseSet nursing=PhraseSetBuilder.with(qiContext)
15 .withTexts( "Soins", "Fonction soins").build();
16 PhraseSet repas=PhraseSetBuilder.with(qiContext)
17 .withTexts( "Repas", "Fonction repas").build();
18 PhraseSet endormissement=PhraseSetBuilder.with(qiContext)
19 .withTexts( "Endormissement", "Fonction endormissement").build();
20
21 // Listen to the phraseSet
22 Listen listen=ListenBuilder.with(qiContext)
23 .withPhraseSets(detente, nursing, repas, endormissement).build();
24 ListenResult listenResult= listen.run();
25
26 // If there is a match
27 PhraseSet matchedPhraseSet = listenResult.getMatchedPhraseSet();

```

Lorsque l'utilisateur prononce le nom d'une des fonctions, le chatbot est lancé avec le bon fichier contenant les bonnes règles utilisateur.

Par exemple, pour la fonction détente :

```

1 // Launch of the chatbot corresponding to the matched phraseSet
2 if(PhraseSetUtil.equals(matchedPhraseSet, detente)){
3
4     // Create a topic.
5     Topic topic = TopicBuilder.with(qiContext)
6         // Set the .topic file
7         .withResource(R.raw.detente)
8         .build();
9
10    // Create a new QiChatbot.
11    QiChatbot qiChatbot = QiChatbotBuilder.with(qiContext)
12        .withTopic(topic)
13        .build();
14
15    // Create a new Chat action.
16    chat = ChatBuilder.with(qiContext)
17        .withChatbot(qiChatbot)
18        .build();
19
20    // Add an on started listener to the Chat action.
21    chat.addOnStartedListener(()->Log.i(TAG, "Discussion détente."));
22
23    // Run the Chat action asynchronously.
24    Future<Void> chatFuture = chat.async().run();
25    chat.addOnListeningChangedListener(listening -> {
26    });
27
28    chat.addOnHeardListener(heardPhrase -> {
29    });
30
31    // Add a lambda to the action execution.

```

```

32     chatFuture.thenConsume(future -> {
33         if (future.hasError()) {
34             Log.e(TAG, "error.", future.getError());
35         }
36     });
37 }

```

On retrouve le même bloc if pour les quatre fonctions.

Seul le fichier .topic utilisé change (ci-dessus "R.raw.detente"). Les fichiers .topic contiennent les règles utilisateurs utilisées par le chatbot.

La syntaxe que ces règles doivent suivre est documentée précisément sur le site de SoftBank.<sup>[17]</sup>

Lorsqu'on lance l'application à ce stade, on peut discuter avec Pepper grâce au chatbot via la fenêtre de discussion de l'émulateur.



Figure 5.2 – Applications de discussions : émulateur et fenêtre de discussion

### 1.2.2 Système speech to text, text to speech

J'ai ensuite développé les fonctionnalités speech to text, text to speech pour pouvoir parler avec l'émulateur à l'oral et non plus à l'écrit via la fenêtre de discussion.

L'élément le plus important est l'utilisation d'un objet du type `SpeechRecognizer`, issue du package `android.speech`. Cet objet assure la reconnaissance vocal, c'est à dire la partie speech to text.

L'utilisation de cet objet, qui se fait dans la méthode `onCreate()` de l'application, implique l'implémentation de plusieurs méthodes, qui permettent de définir son comportement.

Les méthodes que j'ai implémentées sont `onBeginingOfSpeech()` et `onResults()`. La méthode `onBeginingOfSpeech()` est appelée lorsque le `speechRecognizer` a commencé à écouter et qu'il entend une voix. La méthode modifie l'interface graphique pour indiquer à l'utilisateur que l'application est bien en train de l'écouter.

La méthode `onResults()` est appelée quand le `speechRecognizer` a arrêté d'écouter et qu'il a transformé la parole de l'utilisateur en texte. Cette méthode récupère le texte et devait

normalement le passer au chatbot, avant de récupérer la réponse du chatbot sous forme de texte et de la transformer en vocal. C'est la partie text to speech. Malheureusement on verra juste après que ça n'a pas été possible.

Le speechRecognizer est piloté via un bouton que j'ai intégré sur l'interface graphique de la tablette. Quand on appuie sur le bouton, on lui ordonne de se mettre en écoute. Quand on lâche le bouton, on lui ordonne d'arrêter d'écouter.

```

1 // Button used to trigger the start/stop of speech recognition
2 imageView.setOnClickListener(new View.OnClickListener() {
3     @Override
4     public boolean onTouch(View view, MotionEvent motionEvent) {
5         // if the user release the button
6         if(motionEvent.getAction()==MotionEvent.ACTION_UP){
7             // stop speech recognition: onResults() is executed
8             speechRecognizer.stopListening();
9             //UI update: the app isn't listening anymore
10            imageView.setImageResource(R.drawable.ic_baseline_mic_24);
11
12        }
13        //If the user press the button
14        if(motionEvent.getAction()==MotionEvent.ACTION_DOWN){
15            //UI update: the app is listening
16            imageView.setImageResource(R.drawable.ic_baseline_mic_24);
17            //start speech recognition: onBeginningOfSpeech() is ready
18            speechRecognizer.startListening(speechIntent);
19        }
20
21        return false;
22    }
23 });

```

### 1.3 IHM

Voici l'interface utilisateur pour la tablette de Pepper. Cette interface est simple. En haut, la zone de texte indique ce que l'utilisateur a dit. En bas, la zone de texte était sensée indiquer ce que le robot a répondu. Au milieu on retrouve le bouton qui permet de piloter le speechRecognizer.

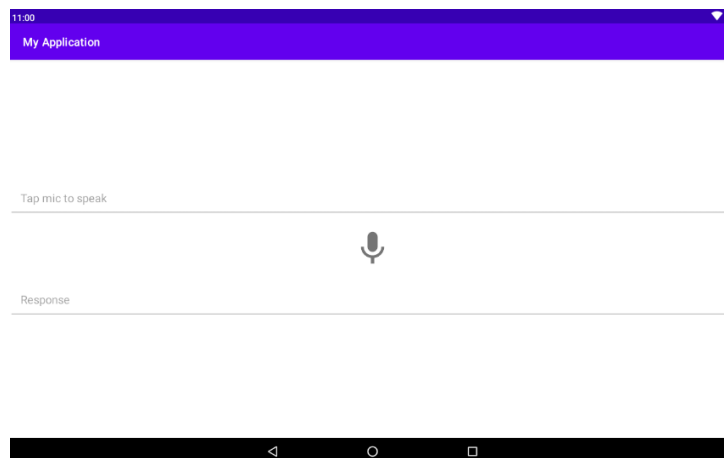


Figure 5.3 – Interface graphique sur la tablette tactile de l'émulateur

## 1.4 Analyse des résultats, évaluation, qualité

Malheureusement les résultats de cette application n'ont pas été bons. La partie speech to text, text to speech ne fonctionnait pas quand je lançais l'application sur l'émulateur.

Après des recherches, j'ai appris que les émulateurs d'Android Studio (tous les émulateurs et pas seulement celui de Pepper) ne sont pas équipés de la capacité de reconnaissance vocale. Le SpeechRecognizer n'a donc aucune chance de fonctionner. Mais le chatbot et les applications de discussion continuaient de marcher correctement, via la fenêtre de discussion.

J'ai donc essayé de lancer l'application sur mon téléphone portable. Sur mon téléphone, je savais que je n'aurais accès ni à l'aperçu 3D du robot, ni à la fenêtre de discussion, mais uniquement à l'interface graphique. Néanmoins je me suis dit que le chatbot pourrait peut être quand même fonctionner en arrière plan et que je pourrais l'utiliser via l'interface graphique.

Mais là encore j'ai rencontré un problème. Les fonctions propres au plug-in de Pepper (`onRobot-FocusGained()` notamment) ne s'exécutent pas lorsque l'application est lancée ailleurs que sur l'émulateur ou sur un vrai robot.

En revanche, la partie speech to text fonctionnait très bien.

Pour résumer, soit l'application est lancée sur l'émulateur et on peut utiliser le chatbot via la fenêtre de discussion mais la partie speech to text, text to speech ne fonctionne pas.

Soit on lance l'application sur un téléphone. On ne peut pas utiliser le chatbot, mais on peut utiliser la partie speech to text, text to speech (mais ça n'a pas d'intérêt car ce n'est pas relié au chatbot).

Après concertation avec M.Conte, nous avons décidé que je développerai un chatbot dans un autre langage, en reprenant les scénarios de discussions des spécifications, et avec des fonctionnalités speech to text, text to speech. Mais développer dans un autre langage implique l'abandon de l'émulateur.

## 2 Développement du chatbot en Python

### 2.1 Outils et librairie utilisés

J'ai choisi le langage Python car il est simple d'utilisation, il permet d'installer facilement des packages supplémentaires, dont des packages spéciaux pour développer un chatbot.

C'est le cas du package `chatterbot`<sup>[4]</sup> que j'ai utilisé. Chatterbot est très intéressant car il permet de développer des chatbots combinant règles utilisateur et machine learning. Donc il est possible de suivre les scénarios de discussions des spécifications, tout en enrichissant les possibilités avec de l'apprentissage artificielle sur une base de donnée de discussions de base fournie par le package.

Les autres packages principaux sont `speech_recognition` pour le speech to text, `pyttsx3` pour le text to speech, `pyaudio` pour pouvoir capter le son du micro et `tkinter` pour l'interface graphique.

`Pyttsx3` permet de faire du text to speech "hors ligne" (sans avoir besoin d'internet). Il utilise simplement le logiciel de reconnaissance vocal intégré au système d'exploitation (dans le cas de windows, SAPI5).

Si le système d'exploitation n'a pas de logiciel de reconnaissance vocal intégré, il est possible de remplacer `Pyttsx3` par la combinaison de packages `gTTS` et `playsound`. `gTTS` est le package de Google pour faire du text to speech "en ligne", et `playsound` permet de jouer les fichiers .mp3 générés par `gTTS`.

Mais j'ai expérimenté playsound et il provoque parfois des bugs lors de la lecture des fichiers .mp3.

C'est pour ça que j'ai utilisé pyttsx3. C'est aussi parce que pyttsx3 permet de régler la vitesse de diction, ou bien de changer la voix du synthétiseur si plusieurs sont disponibles.

En terme de version, j'ai utilisé Python 3.10. Le package chatterbot n'est pas compatible avec les versions inférieures à 3.7.

Mais le package pyaudio n'est de base pas compatible avec les versions de Python supérieures à 3.6.

La procédure d'installation, détaillée dans le guide en annexe, a donc été un peu particulière.

## 2.2 Éléments d'implémentation, choix techniques

J'ai implémenté le chatbot en premier. Comme dit précédemment, chatterbot repose sur un système d'apprentissage artificiel. Pour "enregistrer" des règles utilisateur, il suffit de lancer l'apprentissage du chatbot sur ces règles.

J'ai donc rédigé quatre fichiers : "chatbotTrainingRepas.py", "chatbotTrainingDetente.py", "chatbotTrainingEndormissement.py", "chatbotTrainingNursing.py".

Chaque fichier contient tous les scénarios de discussion de la fonction correspondante, en suivant la syntaxe des règles utilisateur de chatterbot.

J'ai ensuite écrit un fichier "chatbotTraining.py". Dans ce fichier j'ai développé la fonction training(chatbot), qui lance l'apprentissage du chatbot sur les quatre fichiers ci-dessus et sur la base de données de conversations fournie par le package.

Il est donc très simple de modifier l'apprentissage, de rajouter des règles dans de nouveaux fichiers ou d'enrichir les scénarios déjà présents.

L'apprentissage génère automatiquement un fichier "db.sqlite". C'est un fichier de base de données qui contient tout ce que le chatbot a retenu de son apprentissage.

J'ai développé un système de "test unitaire" pour évaluer la qualité de l'apprentissage. Ces tests consistent à vérifier que la réponse du robot suit bien les scénarios de discussion en fonction de ce que dit l'utilisateur.

Les tests sont séparés selon les quatre fonctions de discussions, pour pouvoir évaluer séparément la qualité de l'apprentissage des quatre méthodes d'apprentissage.

L'étape suivante a consisté à développer l'interface graphique du programme. J'ai utilisé tkinter car c'est intégré nativement à Python et c'est simple d'utilisation.

L'objectif de l'interface est fonctionnel et pas esthétique. Elle doit retranscrire à l'écrit ce que l'utilisateur a dit au chatbot et ce que le chatbot a répondu pour éviter les incompréhensions. Elle doit aussi intégrer un bouton sur lequel appuyer quand l'utilisateur souhaite parler au chatbot.

La fonction principale (speak()) est la fonction qui est exécutée quand l'utilisateur appuie sur le bouton.

Cette fonction va en premier lieu enregistrer ce que dit l'utilisateur pendant trois secondes, grâce à speech\_recognition.

Imposer un délai de trois secondes pour parler n'est peut-être pas le mieux. J'ai fait une version du programme dans laquelle il n'y a pas de délais et où l'enregistrement s'arrête quand l'utilisateur se tait. Mais à cause de bruit ambiant et de la qualité du micro, le programme a de grandes difficultés à détecter quand l'utilisateur se tait.

C'est pour ça que j'ai gardé la version avec un délai de trois secondes.

La fonction va ensuite convertir cet enregistrement en texte, puis passer ce texte au chatbot.

Puis elle récupère la réponse du chatbot et utilise pyttsx3 pour la convertir en format audio et la jouer.

Tout ceci est accompagné de mises à jour de l'interface graphique pour tenir l'utilisateur au courant de l'avancement du processus.

```

1 def speak():
2     global heard, listening, response
3
4     # prevent user to click on the button while the function is running
5     button['state'] = DISABLED
6
7     #Records using the mic:
8     with sr.Microphone() as source:
9         #UI update
10        listening.config(text="A l'ecoute")
11        listening.update()
12        print("Recognizing...")
13        # listen to the user until he is silent
14        audio_data = r.listen(source)
15        # conversion from speech to text
16        heardText = r.recognize_google(audio_data, language="fr-FR")
17
18        #UI update
19        listening.config(text=" ")
20        listening.update()
21        heard.config(text=heardText)
22        heard.update()
23
24        #Transmission of input to the chatbot and collection of response
25        botResponse=str(chatbot.get_response(heardText))
26
27        #Check if the chatbot answered correctly
28        if (botResponse.__len__() < 2):
29            botResponse = "Je n'ai pas compris"
30
31        # UI update
32        response.config(text=botResponse)
33        response.update()

```

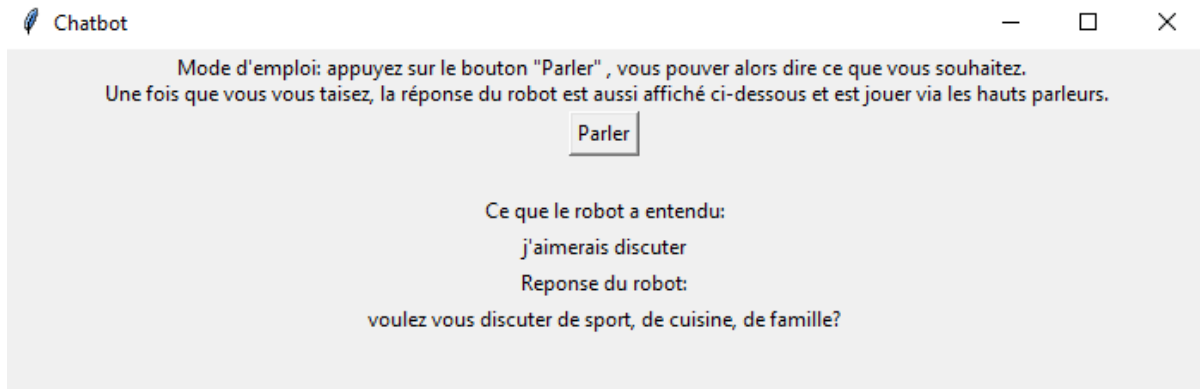
Cette fonction ainsi que la déclaration du chatbot, se trouve dans le fichier "chatbot.py". C'est ce fichier qu'il faut exécuter pour lancer le programme. Comme détaillé dans le guide d'utilisation en annexe, l'apprentissage n'a pas besoin d'être fait à chaque lancement du programme.

## 2.3 IHM

Voici l'IHM de l'application. Elle est très simple.

## 2.4 Analyse des résultats, évaluation, qualité

L'application marche correctement. Le fractionnement des fichiers pour les fonctions d'apprentissage permet une très grande modularité et la possibilité d'enrichir cet apprentissage facilement.



**Figure 5.4** – *Interface de l'application Python*

L'interface graphique est simple mais fonctionnelle. Néanmoins il y a un bug. Lorsque la réponse du chatbot est longue (poème ou fable), elle sort de la fenêtre car elle est écrite sur une seule ligne.

A ce jour je n'ai pas trouvé le moyen de faire des retours à la ligne.

L'environnement de travail est assez capricieux. Si les bonnes versions de Python et des packages ne sont pas utilisées, l'application ne fonctionnera sans doute pas.

# 6

## Bilan et conclusion

### 1 Bilan du semestre 9

Durant ce semestre, j'ai découvert le sujet. J'ai appréhendé les enjeux, rencontré les différents acteurs avec lesquels nous avons posé les bases des spécifications. Je me suis documenté sur les technologies et j'ai commencé à modéliser une solution pour le développement.

Je ne pensais pas que la location d'un robot Pepper serait aussi difficile. Cela a représenté un imprévu. La recherche d'une potentielle alternative doit permettre de résoudre cet imprévu.

### 2 Bilan du semestre 10

L'échec pour l'obtention d'un robot a grandement affecté la réalisation du projet.

Néanmoins, la gestion de cet imprévu a été très intéressante car je rencontrerai forcément d'autres imprévus dans ma carrière professionnelle.

Trouver et mettre en oeuvre une solution pour palier à cet imprévu, tout en se restant le plus proche possible des objectifs de départ est quelque chose qui me sera très utile.

### 3 Bilan sur la qualité

La qualité de l'application développée sur Android Studio n'est probablement pas optimale car elle est instable.

Cela est dû en partie par l'instabilité de l'émulateur, mais également par le fait qu'elle combine sans succès les fonctions de discussions et le système STT, TTS.

Mais les fonctions de discussions sont pertinentes et pourront être reprises si le projet est perpétué sur le robot Pepper.

L'application de chatbot en Python est de bonne qualité. Elle peut être utilisée, améliorée facilement.

### 4 Bilan auto-critique

Au S9, j'aurais sûrement pu être plus précis dans mon état de l'art et dans la modélisation. Je voulais rester général car on ne savait pas encore vraiment quel robot on allait utiliser, mais j'aurais pu quand même rentrer plus dans les détails.



Au S10, je suis content de la manière dont j'ai travaillé pour développer dans un temps réduit les deux applications, qui pourront peut être être utiles pour la suite du projet.

# Annexes

# A

## Planification, gestion de projet

### 1 Evolution du projet

Voici la planification initiale du projet au S9 sous forme d'un diagramme de Gantt.

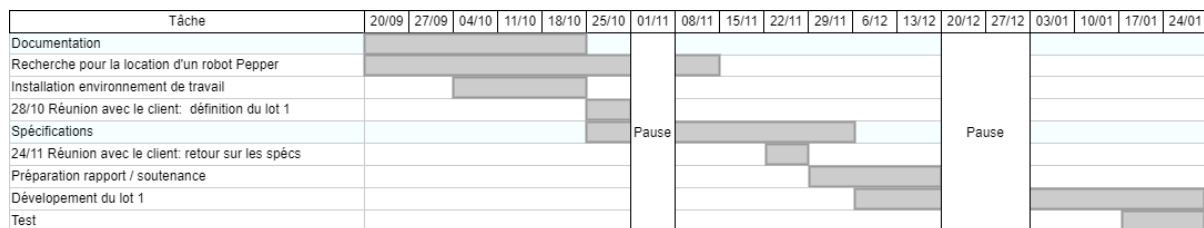


Figure A.1 – Diagramme de Gantt initial S9

Voici comment s'est réellement déroulé le projet au S9.

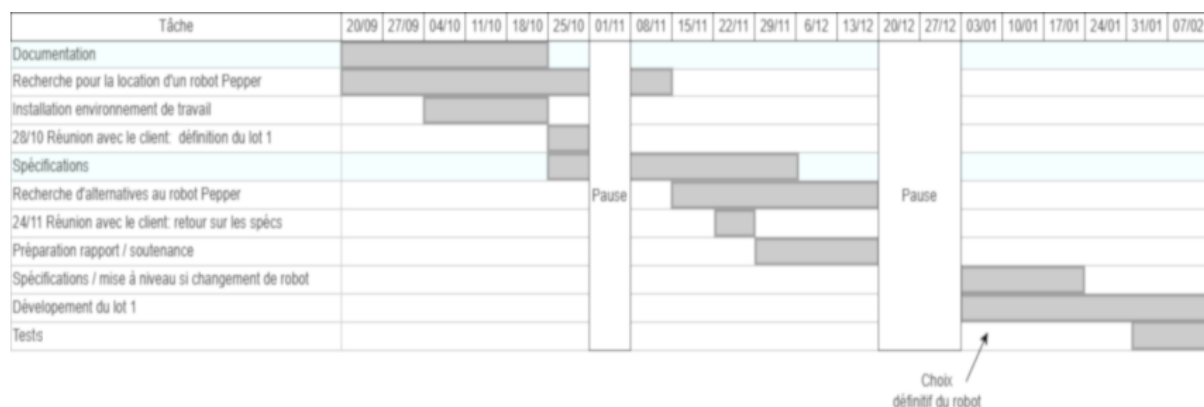


Figure A.2 – Diagramme de Gantt final S9

Les difficultés rencontrées pour louer le robot Pepper ont décalées le planning prévisionnel. Le choix définitif du robot et les changements qui pourraient en découler risquent à nouveau de décaler le planning dans le futur.

Voici comment s'est déroulé le projet au S10.

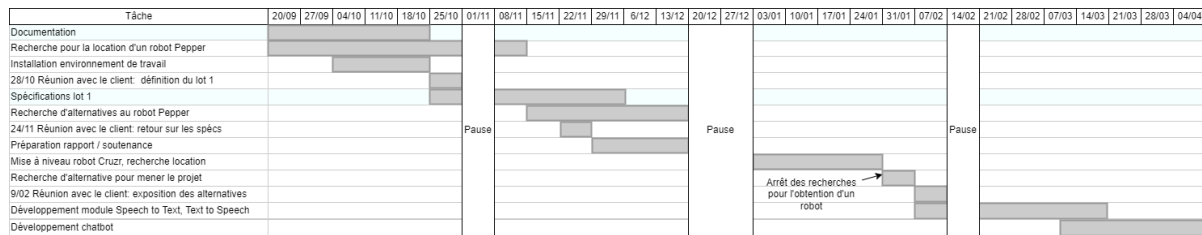


Figure A.3 – Diagramme de Gantt final S10

## 2 Description des tâches

### Tâche 1 : Documentation

- Date de début : 20/09/2021
- Date de fin : 25/10/2021
- Durée : 35 jours
- Description : découverte du sujet, compréhension des enjeux et acteurs, prise d'informations sur Pepper.

### Tâche 2 : Recherche pour la location d'un robot Pepper

- Date de début : 20/09/2021
- Date de fin : 15/11/2021
- Durée : 49 jours
- Description : contacts avec Softbank Robotics et d'autres entreprises pour la location d'un robot Pepper.

### Tâche 3 : Installation de l'environnement de travail

- Date de début : 04/10/2021
- Date de fin : 25/10/2021
- Durée : 21 jours
- Description : installation d'Android studio et des outils fournis par Softbank Robotics. Prise en main des principes de base des API.

### Tâche 4 : Première réunion avec le client

- Date de début : 28/10/2021
- Date de fin : 28/10/2021
- Durée : 1 jour
- Description : visioconférence avec le Living Lab et des employés des Ehpad. Présentation du robot Pepper et définition des besoins pour le lot 1.

### Tâche 5 : Spécifications

- Date de début : 25/10/2021
- Date de fin : 06/12/2021
- Durée : 35 jours
- Description : rédaction du cahier des spécifications, en se basant sur la réunion du 28/10 et le cahier des charges d'Alfred.

### Tâche 6 : Recherche d'alternative à Pepper

- Date de début : 15/11/2021
- Date de fin : 19/12/2021
- Durée : 35 jours
- Description : recherche de robots ayant à peu près les mêmes capacités que Pepper et qui pourraient répondre aux besoins du projet.

**Tâche 7 : Deuxième réunion avec le client**

- Date de début : 24/11/2021
- Date de fin : 24/11/2021
- Durée : 1 jour
- Description : visioconférence avec le Living Lab et des employés des Ehpad. Discussions, retours et modifications des spécifications.

**Tâche 8 : Préparation rapport, soutenance**

- Date de début : 29/11/2021
- Date de fin : 13/11/2021
- Durée : 14 jours
- Description : Finalisation de la rédaction du rapport et préparation de la soutenance.

**Tâche 9 : Mise à niveau robot Cruz**

- Date de début : 03/01/2022
- Date de fin : 24/01/2022
- Durée : 21 jours
- Description : Mise à niveau, recherche de d'informations sur le robot Cruz et le Robot Operating System.

**Tâche 10 : Recherche d'alternative pour le projet**

- Date de début : 31/01/2022
- Date de fin : 07/02/2022
- Durée : 2 jours
- Description : Recherche de solutions alternative pour continuer le projet.

**Tâche 11 : Troisième réunion avec le client**

- Date de début : 09/02/2022
- Date de fin : 09/02/2022
- Durée : 1 jour
- Description : Présentation des solutions trouvées au client.

**Tâche 12 : Développement sur l'émulateur de Pepper**

- Date de début : 21/02/2022
- Date de fin : 14/03/2022
- Durée : 21 jours
- Description : Développement des applications de discussion avec l'émulateur de Pepper et les fonctionnalités Speech to Text, Text to Speech.

**Tâche 13 : Développement du chatbot sur Python**

- Date de début : 07/02/2022
- Date de fin : 28/03/2022
- Durée : 35 jours
- Description : Développement du chatbot en python avec fonctionnalités Speech to Text, Text to Speech.

# B

# Cahier de Spécifications

## 1 spécifications Fonctionnelles

### 1.1 Discussion

Le premier bloc du projet est axé sur la discussion entre le robot et les résidents. La discussion est sans doute le moyen le plus efficace de maintenir un lien social. C'est aussi une capacité facilement modulable et avec laquelle il est possible de faire énormément de choses, sans la nécessité d'un développement très complexe.

#### 1.1.1 Fonction nursing

L'objectif de la fonction nursing est de distraire le résident et d'accompagner le soignant lors de l'administration des soins, vers 9h. Pepper se poste à une distance assez grande du résident pour ne pas gêner le travail du soignant.

Le soignant peut demander plusieurs choses à Pepper :

- « Pepper peux-tu nous lire un poème ? »
- « Pepper peux-tu nous raconter une fable ? »
- « Alfred joue nous un sketch »
- →Pepper récite un poème, une fable ou un sketch
- « Pepper peux-tu nous donner les nouvelles du jour ? »
- →Pepper affiche les nouvelles du jour sur sa tablette tactile.

#### 1.1.2 Fonction détente

L'objectif est d'occuper le résident pendant la journée. Pepper peut aborder un résident au hasard. Il peut aussi être interpellé par un résident, auquel cas Pepper s'approchera pour discuter avec ce résident.

Pepper peut dire :

- « Voulez-vous vous détendre ? »
- Si la réponse est positive :
- « Respirez...Souffler... » x3
- « Voulez-vous discuter ? »
- Si la réponse est positive :

- « De la famille ? Du sport ? De cuisine ? »
- Si la réponse est famille :
- « Avez-vous des enfants ? Voulez-vous m'en parlez ? »
- Si la réponse est sport :
- « Suivez-vous la berrichonne ? »
- « Aimez-vous le vélo ? »
- « Aimez-vous la chasse ? »
- Si la réponse est négative :
- « Je vous laisse, à plus tard ! Nous sommes le [date du jour], il est [heure]. »
- Pepper va voir un autre résident.

Pepper est interpellé :

Un résident dit « J'aimerais me détendre »

Pepper répond : « Respirez...Souffler... » x3

Ou bien le résident dit « J'aimerais discuter »

Pepper répond : « De la famille ? Du sport ? De cuisine ? » et la discussion se poursuit comme précédemment.

### 1.1.3 Fonction repas

L'objectif est de stimuler les résidents lors des repas pour les encourager à manger. Pour cette fonction il y a deux niveaux de développement. Dans le premier niveau Pepper s'approche d'une table et dit une des phrases, choisit au hasard. Il y a des phrases simples pour les résidents plus dépendants :

- « C'est bon, c'est bon »
- « Mangez lentement, prenez votre temps. »
- « Il faut manger pour prendre des forces. »
- « J'adore manger. »
- « Je suis très gourmand. »
- Et des phrases plus élaborées pour les résidents plus autonomes :
- « Sentez-vous l'odeur ? »
- « Avez-vous mangé des légumes aujourd'hui ? »

On peut décider à l'avance, dans le code, d'utiliser seulement les phrases simples, seulement les phrases élaborées, ou toutes les phrases en fonction des situations. Dans le deuxième niveau Pepper s'approche d'un résident et doit déterminer grâce à sa caméra si l'assiette du résident est pleine ou vide et adapter ses répliques en conséquence. Si l'assiette est pleine :

- « Voyez-vous toutes les couleurs dans votre assiette ? »
- « Bon appétit »
- Si l'assiette est vide :
- « Avez-vous apprécié toutes les saveurs ? »
- « Vous avez bien mangé ! »

Pepper passe de table en table.

### 1.1.4 Fonction endormissement

Cette fonction est activée par les soignants pour accompagner, rassurer les résidents qui n'arrivent pas à dormir. Pepper peut dire :

- « Dormez bien. »
- « Faites de beaux rêves. »
- « Patientez le sommeil va venir. »
- « Essayez de vous reposer »
- « Ne pensez plus à rien. »

— « Je vais rester un peu avec vous »

Il y a également des phrases pour dire au résident qu'il n'est pas seul :

— « Vous n'êtes pas seul »

— « Il y a du personnel »

— « Le veilleur de nuit est là »

Pepper reste 20 minutes maximum avec le résident.

## 2 Spécifications non fonctionnelles

### 2.1 Modes de fonctionnement

Il y a un bouton sous la tablette tactile qui permet d'allumer et d'éteindre le robot, ainsi qu'un bouton sur la tablette pour allumer cette dernière.

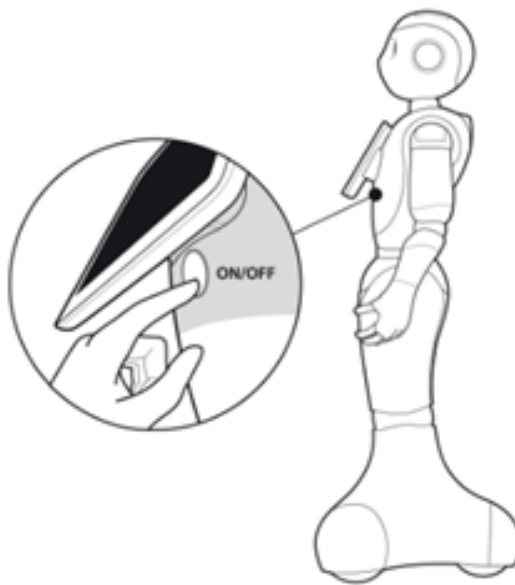


Figure B.1 – Bouton On/Off

Il y a également un bouton d'arrêt d'urgence dans le dos de Pepper.

Pepper a une autonomie d'environ 12h et est rechargeable sur une prise secteur classique.

## 3 Risques

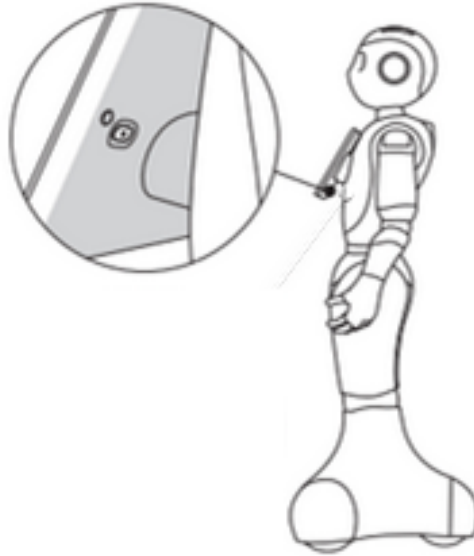
Le projet a initialement été prévu avec le robot Pepper, mais en vue des difficultés rencontrées pour en louer un, il a été envisagé d'en utiliser un autre. Si un autre robot est utilisé, l'environnement de travail (Android Studio/Java) ainsi que la structure générale du système risquent de devoir change, entraînant la nécessité d'une mise à niveau en termes de compétences vers le nouvel environnement.

### 3.1 Capacités de Pepper

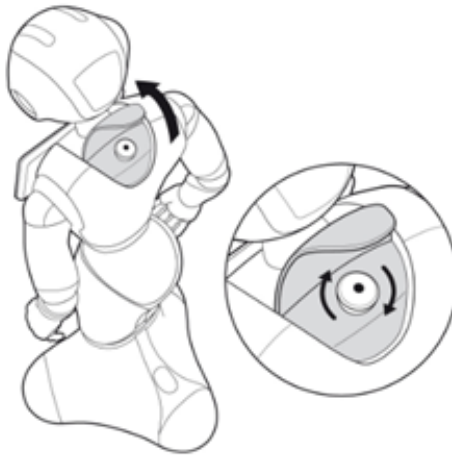
Les spécifications ci-dessus exploitent une partie des capacités de Pepper :

- Synthétiseur vocal → parler selon les répliques enregistrées
- Reconnaissance vocal → écouter selon les répliques enregistrées, choisir la fonction
- Distinction des visages (caméra) → repérer quelqu'un, suivre du regard





**Figure B.2** – *Bouton On/Off de la tablette*



**Figure B.3** – *Bouton d'arrêt d'urgence*

- Chatbot → élaboration des discussions (les outils décrits en 3.1 sont propres à l'API « Conversation » de Pepper)
- Tablette tactile → affichage des infos, (choix de la fonction)
- Roues → déplacement
- Capteurs (laser, caméra) → analyser l'environnement, éviter les obstacles, déplacement autonome, cartographier l'environnement
- « Autonomous Abilities » → simulation de mouvements naturels (clignement des yeux, légers mouvements des bras)
- On peut aussi souligner l'apparence humanoïde de Pepper qui rend les interactions plus agréables.

Si un autre robot est utilisé, il faudra adapter les spécifications en fonction de ses capacités. Les capacités de Pepper qui ne sont pas exploitées dans les spécifications ci-dessus :

- Nombreuses articulations (bras, tête) → faire des animations
- Caméra → prendre une photo
- Capteur tactile (mains, tête et jambes) → réaction si touché
- Perception des émotions → réagir en conséquence
- Connaissances → retenir des informations

- Tablette tactile → jeux
- Reconnaissance vocal → ordre de mouvement

# C

# Guide d'installation et d'utilisation de l'application pour l'émulateur de Pepper

## 1 Introduction

Ce document a pour objectif de donner un maximum d'informations pour l'installation et l'utilisation de l'application Android Studio destinée à l'émulateur de Pepper. Il apporte des détails à ce qui est expliqué dans le rapport.

Le code source principale se situe dans les fichiers :

1	—	MainActivity.java	(fonctions principales)
2	—	activity_main.xml	(interface utilisateur)
3	—	alertcustom.xml	(interface utilisateur)
4	—	repas.top	(règles utilisateur pour le chatbot)
5	—	détente.top	(règles utilisateur pour le chatbot)
6	—	endormissement.top	(règles utilisateur pour le chatbot)
7	—	nursing.top	(règles utilisateur pour le chatbot)

Les fichiers build.gradle et settings.gradle sont aussi importants. Ils sont générés automatiquement lorsqu'on crée une Robot Application, mais j'ai eu besoin de les modifier.

L'ensemble des fichiers ci-dessous ne sont pas une fin en soi et font partie d'une Robot Application (application android), comme on va le voir juste après.

L'objectif de cette application était de combiner les fonctions de discussion du cahier des spécifications sur l'émulateur de Pepper, en rajoutant un module Speech to Text/ Text to Speech. Car de base, la conversation avec l'émulateur de Pepper se fait à l'écrit.

Malheureusement l'émulateur de Pepper (comme tous les émulateurs disponibles avec Android Studio) n'est pas compatible avec la reconnaissance vocale.

## 2 Installation

### 2.1 OS, IDE

J'ai développé cette application sous Windows 10.

J'ai utilisé l'IDE Android Studio, version Arctic Fox, téléchargé depuis le site officiel.

### 2.2 Plug-in Pepper SDK

Pour avoir accès aux fonctionnalités de développement spécifiques de Pepper et à l'émulateur, il est nécessaire d'installer le plug-in Pepper SDK dans Android Studio.

Pour cela suivez le tutoriel disponible sur le site de Softbank robotics : <https://developer.softbankrobotics.com/pepper-qisdk/getting-started/installing-pepper-sdk-plug>

Remarques :

- Vous ne pouvez pas utiliser l'émulateur de Pepper depuis une machine virtuelle, car la virtualisation évoquée dans le tutoriel n'est pas fonctionnelle sur une machine virtuelle.
- Gardez en tête le tout dernier paragraphe du tutoriel (« Modifying the AVD graphical acceleration type »). En effet si l'émulateur ne démarre pas correctement, ou s'il ne démarre pas du tout, ou si l'écran reste noir, essayer de modifier le paramètre indiqué. Cela a fonctionné dans mon cas.
- De manière générale, j'ai remarqué que l'émulateur n'était pas parfaitement stable, des bugs peuvent arriver (application qui refuse de s'exécuter correctement, latences...). Redémarrer l'émulateur, réinstaller l'application peuvent résoudre certains de ces bugs.

### 2.3 Robot Application

Android studio permet de développer des applications pour Android. Une Robot application est une variante spéciale pour Pepper.

Après avoir installé le plug-in, vous pouvez créer une Robot Application : <https://developer.softbankrobotics.com/pepper-qisdk/getting-started/creating-robot-applicationcreating-robot-app>

L'arborescence de l'application avec tous les dossiers et fichiers de base sont créés automatiquement. Il faut maintenant remplacer le contenu de MainActivity.java et activity\_main.xml par le contenu de mes fichiers.

Il faut aussi importer mon fichier alertcustom.xml dans le dossier res/layout (dossier contenant activity\_main.xml) et les 4 fichiers .top dans le dossier res/raw.

#### 2.3.1 Correction des dépendances

La création d'une Robot Application génère automatiquement des fichiers dans le dossier « Gradle Scripts ». Ces fichiers gèrent les dépendances de l'application.

Dans mon cas deux de ces fichiers ne contenaient pas les bonnes informations. Cela provoquait une erreur au lancement de l'application : « Cannot resolve symbol RobotActivity ». (Voir <https://github.com/aldebaran/qisdk-tutorials/issues/12> ).

Pour corriger cette erreur il faut :

Dans le fichier « build.gradle » il faut ajouter deux dépendances :

```
1 dependencies {  
2     implementation 'com.aldebaran:qisdk:1.7.5 '  
3     implementation 'com.aldebaran:qisdk-design:1.7.5 '  
4 }
```

Dans le fichier settings.gradle il faut ajouter un dépôt :

```
1 repositories {  
2     maven {  
3         url 'https://qisdsk.softbankrobotics.com/sdk/maven/'  
4     }  
5 }
```

### 3 Utilisation

Comme dit précédemment, l'émulateur ne peut pas faire de reconnaissance vocale. La partie du code concernant la reconnaissance vocale n'est pas exécutée par l'émulateur.

D'un autre côté, un smartphone possède la capacité de reconnaissance vocale, mais ne peut pas utiliser les méthodes et fonctions spécifiques à Pepper. La partie du code concernant le lancement du chatbot n'est pas exécutée (fonction onRobotFocusGained() notamment).

L'application a donc un comportement complètement différent dépendamment de si elle est lancée sur l'émulateur ou sur un smartphone.

Si elle est lancée sur un smartphone, les fonctionnalités de Speech to Text / Text to Speech vont fonctionner. Mais nous n'aurons pas accès au chatbot. On peut donc rester appuyé sur le micro au milieu de l'écran puis parler. Quand on lâche le micro, le texte entendu s'affiche et est répété par le synthétiseur vocal. L'intérêt est donc nul puisque le chatbot est inaccessible.

Si elle est lancée sur l'émulateur, nous aurons accès au chatbot à l'écrit via la fenêtre de discussion de l'émulateur, mais les fonctionnalités STT TTS ne fonctionneront pas. On peut quand même discuter avec le chatbot et suivre les scénarios de discussion, mais uniquement à l'écrit.

### 4 Astuces

Dans une application Android classique le lien entre MainActivity.java et activity\_main.xml est automatique.

Ce n'est pas le cas pour une Robot Application. Il faut donc, dans le fichier MainActivity.java, au début de la fonction onCreate, rajouter l'instruction :

```
1 setContentView(R.layout.activity_main);
```

Sinon l'interface que vous réalisez dans activity\_main ne s'affichera jamais.

Contrairement à ce qui est indiqué sur le site de softbank (<https://developer.softbankrobotics.com/pepper-qisdsk/tools/layout-editor>), la tablette de Pepper s'appelle « 10.1" WXGA (Tablet) (1280 x800, mdpi) » (voir figure C.1).

Sur l'émulateur, l'application est assez instable. Parfois il faut la relancer plusieurs fois.

Entre chaque lancement, il faut arrêter manuellement l'application depuis la tablette de l'émulateur (depuis le gestionnaire d'applications, comme sur un téléphone normal, figure C.2).

La réinstaller peut aussi régler le problème.

### 5 Liens utiles

Voici une liste des liens qui m'ont été utiles :

<https://github.com/aldebaran/qisdsk-tutorials/issues/12> : Erreurs des dépendances

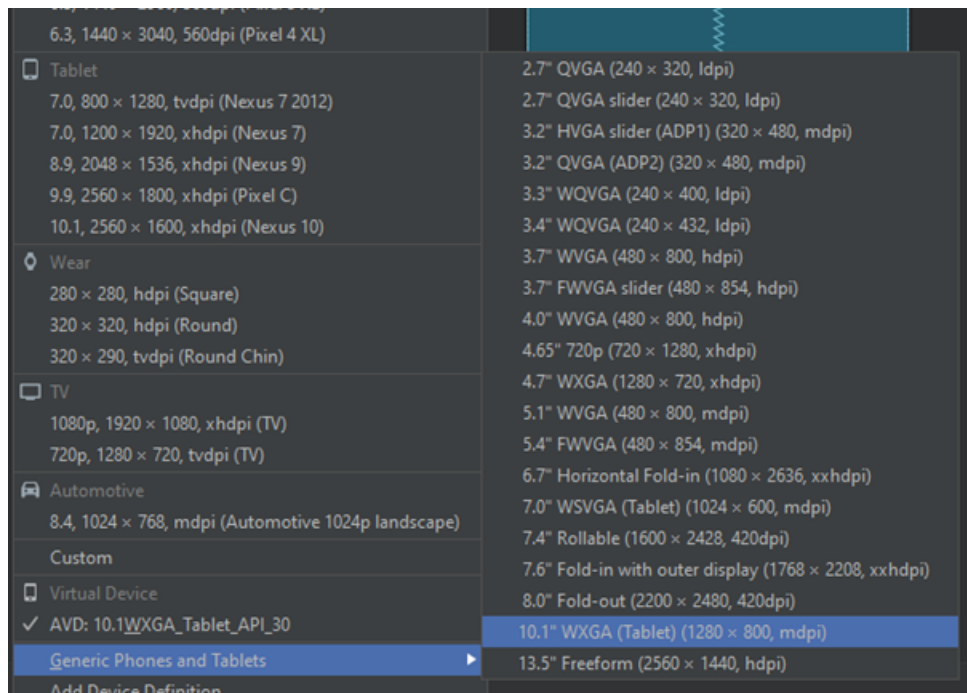


Figure C.1 – Nom de la tablette de Pepper dans Android Studio

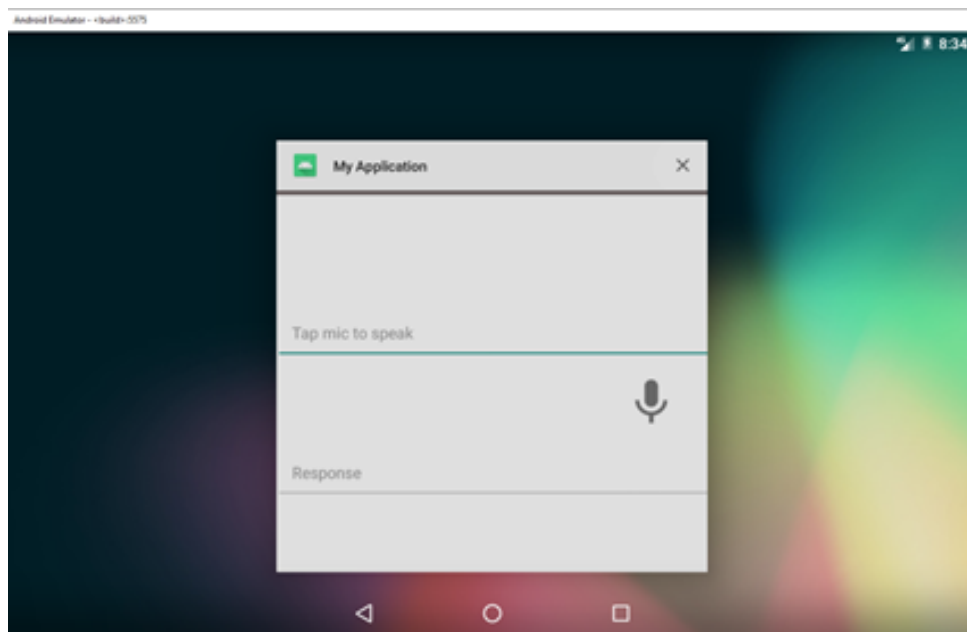


Figure C.2 – Fermer l'application depuis le gestionnaire des tâche de la tablette

<https://stackoverflow.com/questions/30583532/activity-main-xml-and-activity-main-xmlv14> :  
affichage de l'interface

<https://stackoverflow.com/questions/31986689/bind-to-recognition-service-failed> : Emulateur  
non compatible avec la reconnaissance vocale

# D

# Guide d'installation et d'utilisation Chatbot Python

## 1 Introduction

Ce guide a pour but de donner un maximum d'informations relatives à l'installation et l'utilisation du chatbot Python couplé aux fonctionnalités Speech to Text / Text to Speech.

L'environnement dans lequel j'ai travaillé et la manière dont j'ai installé tous les outils et packages ne sont sans doute pas le seul moyen possible pour faire fonctionner l'application.

J'essaie simplement de décrire au mieux le moyen que j'ai employé. Ainsi ce que je décris ci-dessous ne s'applique sûrement pas à tous les IDE et les versions de Python.

L'ensemble du code source est répartie dans six fichiers :

```
1
2 - chatbot.py
3
4 - chatbotTraining.py
5
6 - chatbotTrainingDetente.py
7
8 - chatbotTrainingRepas.py
9
10 - chatbotTrainingEndormissement.py
11
12 - chatbotTrainingNursing.py
```

Le fichier principal permettant de lancer le programme étant chatbot.py

## 2 Installation

### 2.1 OS, IDE et version de Python

J'ai développé ce programme sous Windows 10.

J'ai utilisé PyCharm pour développer ce programme et le faire fonctionner.

J'ai utilisé Python en version 3.10, que j'ai téléchargé depuis le site officiel : <https://www.python.org/downloads/>

Je conseille vivement d'utiliser cette même version de Python. En effet cela permet d'assurer la compatibilité avec tous les packages qu'il faut installer.

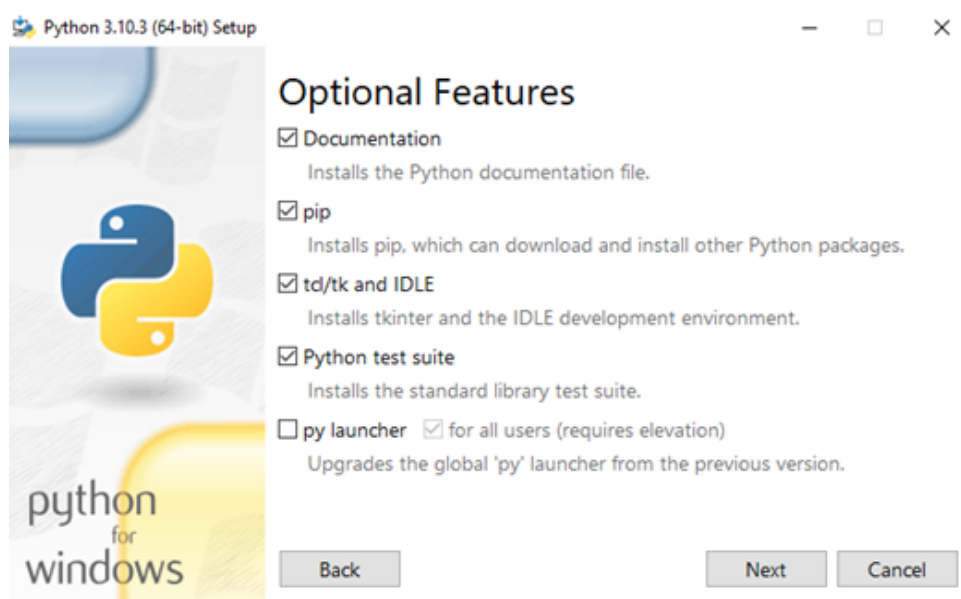
Je déconseille en revanche d'utiliser la version 3.10 de Python disponible sur le Microsoft Store car elle provoque des bugs.

J'ai utilisé cette version de Python car le module chatterbot, que j'ai utilisé pour implémenter le chatbot, n'est compatible qu'à partir de Python 3.7.

Il est tout à fait possible d'avoir plusieurs versions de Python installées simultanément. Avec PyCharm, je conseille de créer un nouveau projet et de choisir la version 3.10 de Python dans l'encadré « Base interpreter ».

Ainsi les modules que nous installerons juste après seront pris en compte uniquement par cette version de Python.

Lors de la configuration de l'installation de Python, veillez à bien cocher la case « tcl/tk and IDLE » pour que tkinter s'installe (voir ci-dessous).



**Figure D.1** – *Configurateur d'installation Python*

Si Python est déjà installé sur votre ordinateur, vous pouvez vérifier si la case est cochée en relançant le configurateur d'installation de Python (son nom est du type « python-3.10.3-amd64.exe »).

## 2.2 Packages

Un certain nombre de packages (ou modules) sont nécessaires au fonctionnement du programme. PyCharm contient un gestionnaire de module permettant de les installer simplement.

Mais il est également possible d'installer les packages depuis un terminal grâce à la commande `pip install nomDuPackage` si vous souhaitez utiliser un autre Ide.

Les packages nécessaires sont :

- SpeechRecognition (pour la partie Speech To Text)
- pyttsx3 (pour la partie Text To Speech)
- chatterbot (pour le chatbot)
- pytz (pour le chatbot)

Les minuscules et majuscules sont importantes.



Un autre package est nécessaire mais son installation est plus difficile : pyaudio.

En effet ce package n'est de base pas compatible avec les versions de python supérieures à 3.6.

Heureusement on peut trouver sur internet une version compatible avec python 3.10 à l'adresse "<https://www.lfd.uci.edu/~gohlke/pythonlibs/pyaudio>".

Il faut télécharger le fichier « PyAudio-0.2.11-cp310-cp310 win\_amd64.whl ».

On peut ensuite placer ce fichier dans le dossier que l'on souhaite, puis ouvrir un terminal dans ce dossier.

On vérifie ensuite sa version de Python grâce à la commande « python -version ». La version doit être 3.10.x.

Si c'est bien le cas, on peut alors exécuter la commande « pip install PyAudio-0.2.11-cp310-cp310 win\_amd64.whl ».

Logiquement les packages installés devraient être les suivants (commande pip freeze) :

```

1  certifi
2  charset-normalizer
3  ChatterBot
4  chatterbot-corpus
5  click
6  colorama
7  comtypes
8  engineering-notation
9  idna
10 joblib
11 mathparse
12 nltk
13 packaging
14 Pint
15 playsound
16 PyAudio
17 pymongo
18 pyparsing
19 pypiwin32
20 python-dateutil
21 pyttsx3
22 pytz
23 pywin32
24 PyYAML
25 regex
26 requests
27 six
28 SpeechRecognition
29 SQLAlchemy
30 tk-tools
31 tqdm
32 urllib3

```

Le package chatterbot installe lui-même d'autres packages, ce qui fait que son installation dure plusieurs minutes.

## 2.3 Correction de fonctions

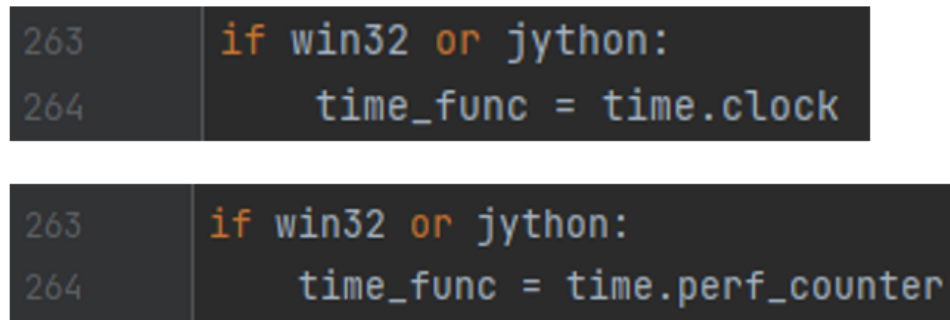
Si vous essayez de lancer chatbot.py maintenant vous risquez d'obtenir cette erreur :

"AttributeError : module 'time' has no attribute 'clock'"

Cette erreur est provoquée par une fonction dépréciée utilisée par package sqlalchemy dans le fichier compat.py.

(Voir <https://stackoverflow.com/questions/58569361/attributeerror-module-time-has-no-attribute-clock-in-python-3-8> )

Ouvrez ce fichier et remplacez `time.clock` par `time.perf_counter` comme ci-dessous.



```

263     if win32 or jython:
264         time_func = time.clock

263     if win32 or jython:
264         time_func = time.perf_counter

```

Figure D.2 – Corriger l'erreur `time.clock`

Une autre erreur est provoquée : "AttributeError : module 'collections' has not attribute 'hashable'".

Par le package yaml qui utilise une fonction avec un mauvais paramètre.

(Voir <https://github.com/ablab/spades/issues/873> )

Remplacez « `collections.Hashable` » par « `collections.abc.Hashable` » comme ci-dessous



```

124     for key_node, value_node in node.value:
125         key = self.construct_object(key_node, deep=deep)
126         if not isinstance(key, collections.Hashable):
127             raise ConstructorError("while constructing a mapping", node.start_mark,
128                                   "found unhashable key", key_node.start_mark)
129         value = self.construct_object(value_node, deep=deep)
130         mapping[key] = value

124     for key_node, value_node in node.value:
125         key = self.construct_object(key_node, deep=deep)
126         if not isinstance(key, collections.abc.Hashable):
127             raise ConstructorError("while constructing a mapping", node.start_mark,
128                                   "found unhashable key", key_node.start_mark)
129         value = self.construct_object(value_node, deep=deep)
130         mapping[key] = value

```

Figure D.3 – Corriger l'erreur `collections.hashable`

Après cela, normalement tout est installé correctement.

## 3 Utilisation

### 3.1 Apprentissage

Le module chatterbot repose sur de l'intelligence artificielle. Le chatbot a donc besoin de s'entraîner sur une base d'apprentissage pour pouvoir fonctionner.

L'apprentissage est réalisé la fonction « `training()` » dans le fichier « `chatbotTraining.py` ».

Cette fonction permet au chatbot de s'entraîner sur un ensemble d'exemple de petite discussions incluses dans le package chatterbot ("`chatterbot.corpus.french`").

Mais elle va aussi permettre au chatbot de s'entraîner sur de multiples petites discussions que j'ai implémenté dans les 4 fichiers « chatbotTrainingEndormissement.py », « chatbotTrainingNursing.py », « chatbotTrainingRepas.py » et « chatbotTrainingDetente.py ».

Ces discussions sont issues des scénarios du cahier des spécifications.

Cet entraînement va déboucher sur la création automatique d'un fichier « db.sqlite3 ». C'est grâce à ce fichier que le chatbot va déterminer sa réponse en fonction de ce qu'on lui dit.

Le chatbot n'a pas besoin de s'entraîner à chaque fois qu'on exécute « chatbot.py ». Au contraire, plusieurs entraînement consécutifs provoque des réponses incohérentes.

Il suffit donc de lancer une première fois le script « chatbotTraining.py » pour créer le fichier « db.sqlite3 ». Puis on peut exécuter le programme « chatbot.py »

Néanmoins, il est tout à fait possible de modifier l'entraînement du chatbot. Il suffit de supprimer le fichier « db.sqlite3 », de modifier comme on le souhaite les fonctions d'entraînement, avant de relancer le programme « chatbotTraining .py ». On peut facilement implémenter d'autres entraînements que les quatre que j'ai implémenté, ou bien les enrichir.

Dans les fonctions d'entraînement, chaque scénario se présente sous la forme suivante :

```

1 list.append([
2     "j'aimerais discuter",
3     "voulez-vous discuter de sport, de cuisine, de famille ?",
4     "de cuisine",
5     "quel est votre plat préféré ?",
6     "j'adore les lasagnes",
7     "c'est vrai que c'est bon !",
8 ])

```

La première ligne ("j'aimerais discuter") correspond à ce qu'on dit au chatbot. La deuxième ligne ("voulez-vous discuter de sport, de cuisine, de famille?") correspond à la réponse du chatbot. Et ainsi de suite.

(Voir <https://chatterbot.readthedocs.io/en/latest/training.html#training-via-list-data> ).

Il est essentiel d'écrire les phrases sans majuscule au début des phrases, car la fonction de Speech to Text génère des textes sans majuscule.

J'ai développé un système de test pour évaluer la qualité de l'apprentissage. Cela se trouve dans le fichier « trainingTest.py » et dans les quatre fichiers .py dans le dossier « trainingTest ». Les tests sont des tests unitaires. On vérifie que la réponse du robot suit bien les scénarios de conversation. L'objectif n'est pas d'avoir 100% de réussite car cela pourrait traduire un sur-apprentissage. Au contraire, un taux inférieur à 100% démontre que les discussions ne sont pas monotones et que le chatbot peut s'adapter.

## 3.2 Fonctionnement

Une fois que l'apprentissage a été lancé une fois on peut lancer le programme « chatbot.py »  
Lorsqu'on lance le programme l'interface utilisateur apparaît :



Figure D.4 – Interface obtenue au lancement du programme

Vous pouvez discuter avec le chatbot.

Bien sûr votre ordinateur doit être équipé d'un micro.

## 4 Liens utiles

Voici quelques liens qui m'ont été utiles :

<https://www.thepythoncode.com/article/using-speech-recognition-to-convert-speech-to-text-python> : speech to text

<https://www.thepythoncode.com/article/convert-text-to-speech-in-python> : text to speech

<https://www.lfd.uci.edu/~gohlke/pythonlibs/pyaudio> : Pyaudio pour Python 3.10

<https://chatterbot.readthedocs.io/en/stable/> : Documentation Chatterbot

# E

## Bibliographie

- [1] Science Avenir. « Le robot Nao, coach pour seniors dans une maison de retraite ». In : (2015). URL : [https://www.sciencesetavenir.fr/sante/le-robot-nao-coach-pour-seniors-dans-une-maison-de-retraite\\_28926](https://www.sciencesetavenir.fr/sante/le-robot-nao-coach-pour-seniors-dans-une-maison-de-retraite_28926).
- [2] Julie Le BOLZER. « Buddy, le robot qui facilite le maintien à domicile ». In : (2019). URL : <https://www.lesechos.fr/thema/articles/buddy-le-robot-qui-facilite-le-maintien-a-domicile-960265>.
- [3] Amaury CHEVALIER, Clara BAILLEHACHE et Aditya Pratap SINGH. « Comparison of Pepper's OS versions ». In : (2018). URL : <https://developer.softbankrobotics.com/blog/comparison-peppers-os-versions>.
- [4] Gunther COX. « About ChatterBot ». In : (). URL : <https://chatterbot.readthedocs.io/en/stable/>.
- [5] La Revue du DIGITAL. « Chez Sephora, le robot Pepper remplace la borne magasin ». In : (2018). URL : <https://www.larevuedudigital.com/chez-sephora-le-robot-pepper-remplace-la-borne-magasin/>.
- [6] Isabelle GIRARDIN. « Coronavirus : des robots humanoïdes mis à la disposition de maisons de retraite du Nord pendant le confinement ». In : (2020). URL : <https://france3-regions.francetvinfo.fr/hauts-de-france/nord-0/video-coronavirus-robots-humanoides-mis-disposition-maisons-retraite-du-nord-confinement-1809414.html>.
- [7] France INFO. « Un robot à commandes vocales pour assister et divertir les personnes âgées en Ehpad ». In : (2021). URL : [https://www.francetvinfo.fr/sante/maladie/coronavirus/confinement/un-robot-a-commandes-vocales-pour-assister-et-divertir-les-personnes-agees-en-ehpad\\_4292587.html](https://www.francetvinfo.fr/sante/maladie/coronavirus/confinement/un-robot-a-commandes-vocales-pour-assister-et-divertir-les-personnes-agees-en-ehpad_4292587.html).
- [8] Clément LESAFFRE. « Pepper, le robot symbole de la French Tech, déjà en retraite forcée ». In : (2021). URL : <https://www.europe1.fr/technologies/pepper-le-robot-symbole-de-la-french-tech-deja-en-retraite-forcee-4054936>.
- [9] NERCES. « Pepper, le robot humanoïde à 1 790 \$, se fait virer de tous ses postes ». In : (2021). URL : <https://www.clubic.com/robotique/actualite-377926-sr-pepper-le-robot-humanoide-a-1-790-se-fait-virer-de-tous-ses-postes.html>.

- [10] Justine PÉROU. « Le robot Pepper arrive dans les concessions Renault ! » In : (2017). URL : <https://pro.largus.fr/actualites/le-robot-pepper-arrive-dans-les-concessions-renault-8431390.html>.
- [11] Hélène PERRAUDEAU. « Le robot humanoïde Pepper testé dans un hôpital de la Manche ». In : (2019). URL : [https://actu.fr/normandie/saint-lo\\_50502/le-robot-humanoide-pepper-teste-dans-hopital-manche\\_23960340.html](https://actu.fr/normandie/saint-lo_50502/le-robot-humanoide-pepper-teste-dans-hopital-manche_23960340.html).
- [12] Softbank ROBOTICS. « Getting Started ». In : (). URL : <https://developer.softbankrobotics.com/pepper-qisdk/getting-started>.
- [13] Softbank ROBOTICS. « Mastering Focus Robot lifecycle ». In : (). URL : <https://developer.softbankrobotics.com/pepper-qisdk/principles/mastering-focus-robot-lifecycle>.
- [14] Softbank ROBOTICS. « Pepper (NAOqi 2.5) | Softbank Robotics Developer Center ». In : (). URL : <https://developer.softbankrobotics.com/pepper-naoqi-25>.
- [15] Softbank ROBOTICS. « Pepper QiSDK | Softbank Robotics Developer Center ». In : (). URL : <https://developer.softbankrobotics.com/pepper-qisdk>.
- [16] Softbank ROBOTICS. « Pepper QiSDK API ». In : (). URL : <https://developer.softbankrobotics.com/pepper-qisdk/api>.
- [17] Softbank ROBOTICS. « QiChat - Syntax ». In : (). URL : <https://developer.softbankrobotics.com/pepper-qisdk/api/conversation/qichat-language/qichat-syntax#first-function>.
- [18] Vikram SINGH. *Pepper Robot fallen person detection*. 2018. URL : [https://www.youtube.com/watch?v=n\\_cCs7YTf70](https://www.youtube.com/watch?v=n_cCs7YTf70).
- [19] AI Lab of SOFTBANK ROBOTICS. « [AI Lab] Pepper Robot Learning "Ball in a Cup ». In : (2016). URL : <https://www.softbankrobotics.com/emea/fr/blog/videos/ai-lab-pepper-robot-learning-ball-cup>.
- [20] Hotellerie SUISSE. « Des robots en service dans l'hôtellerie ». In : (). URL : <https://www.hotelleriesuisse.ch/fr/priorites-et-tendances/histoires/robotics>.

Romain FRAGNIER

Encadrement : Donatello CONTE



En collaboration avec Polytech

## Objectifs

- Comprendre les besoins des soignants et des résidents
- Combiner besoins et possibilités techniques
- Développement d'applications



## Mise en œuvre

1. Modélisation
2. Adaptation face aux imprévus
3. Développement



Robot Pepper

## Résultats attendus

Des applications pour le robot Pepper pour permettre aux résidents de maisons de retraite d'avoir plus d'interactions



Chatbot

# Développement d'applications pour le robot Pepper :

Romain FRAGNIER

Encadrement : Donatello CONTE

## Objectifs

- Comprendre les besoins des soignants et des résidents
- Combiner besoins et possibilités techniques
- Développement d'applications

## Mise en œuvre

1. Modélisation
2. Adaptation face aux imprévus
3. Développement



En collaboration avec Polytech

## Résultats attendus

Des applications pour le robot Pepper pour permettre aux résidents de maisons de retraite d'avoir plus d'interactions



Robot Pepper



Chatbot



# Développement d'applications pour le robot Pepper

## Résumé

Alors que certaines personnes âgées souffrent de solitude, le laboratoire de recherche Usetech'Lab, en collaboration avec des Ehpad et Polytech Tours, souhaite étudier l'impact qu'un robot Pepper pourrait avoir sur les résidents. Pepper est un robot humanoïde capable de parler, d'écouter, de bouger. Ce PRD a pour objectif de développer des applications sur ce robot, pour qu'il puisse interagir avec les résidents. La première phase du projet se concentre sur la discussion entre le robot et les résidents

## Mots-clés

Robotique, Développement, Seniors, robot Pepper, Recherche

## Abstract

While some seniors are suffering from solitude, the Usetech'Lab research laboratory, in collaboration with Ehpad and Polytech Tours, wants to study the impact of a Pepper robot on the residents. Pepper is an humanoid robot which is able to listen, to speak, to move. The purpose of this PRD is to develop applications for this robot, to give him the abilitie to interact with the residents. The first step of this project is focused on the discussion between the robot and the residents.

## Keywords

Usetech'Lab, Robotic, Development, Seniors, Pepper robot, Research, Usetech'Lab, Chatbot

Entreprise

Polytech



Tuteur entreprise

Usetech'Lab

Étudiant

Romain FRAGNIER (DI5)

Tuteur académique

Donatello CONTE