



# Le Projet Miroir Connecté

2022

CARDONA Alexandre

GRANDON Luca

LAPORTE Maxence

HUPPERTS Adrien

Brevet Technicien Supérieur Systèmes Numériques option Informatique et Réseaux |  
Électronique et Communication



## Sommaire :

<b>Vue d'ensemble</b>	<b>5</b>
a - Objectifs	6
b - Analyse de l'existant	6
c - Expression du besoin	6
d - Ressources Humaines et répartition des tâches	7
e - Coût matériel	8
Gantt et planification	10
<b>Mode d'emploi :</b>	<b>11</b>
Reconnaissance vocale :	11
Commande en lien avec la radio	11
Commande en lien avec le miroir :	12
Commande en lien avec l'IHM :	12
Commande en lien avec la musique via USB :	13
Mode d'emploi de l'interface administrateur django :	14
Création de profil	14
<b>CONTRAT Étudiant 1 EC HUPPERTS Adrien</b>	<b>17</b>
I- Présentation du contrat	17
II- Logiciel utilisé	17
III- Ressources et matériaux utilisés	18
A- Capteurs	18
a- Température : CTN et LM35	18
b- Humidité : HIH 4030	19
c- Présence : HC SR04	20
B- Résistances	21
C- PCF8591 (Convertisseur Analogique Numérique)	22
D- TSR 3-24150 (Traco Power)	23
IV- Carte électronique	24
V- Raspberry PI 4	27
VI- Code	28
VII- Diagramme d'utilisation	30
VIII- Gantt	30
IX- Problèmes rencontrés	31
X- Nomenclature	31
XI- Conclusion	32
<b>CONTRAT Étudiant 2 IR LAPORTE Maxence</b>	<b>33</b>
I - Objectifs :	33
II - Ressources logicielles, matérielles et diagramme de Gantt:	33
Diagramme de Gantt	34
III - Choix Technologiques :	35
IV - Diagramme des cas d'utilisation :	36

V - Interface Graphique :	37
VI - Diagramme de Classe :	39
VII - Diagramme d'objet :	40
VIII - Organisation de Thread, diagramme de séquence et Explication de code :	41
a - Organisation des threads :	41
b - Diagramme de séquence de la classe ApplicationIHM :	42
Explication du code pour la classe ApplicationIHM :	43
c - Diagramme de séquence de la classe widgetMeteo :	44
Explication du code pour la classe widgetMeteo :	45
d - Diagramme de séquence de la classe widgetRadio :	46
Explication du code pour la classe widgetRadio :	47
e - Diagramme de séquence de la classe controleSon :	48
Explication du code pour la classe controleSon :	49
f - Diagramme de séquence de la classe modeAveugle :	50
Explication du code pour la classe modeAveugle :	51
g - Diagramme de séquence de la classe user:	52
Explication du code pour la classe user:	53
h- Diagramme de séquence de la classe widgetHorloge:	54
Explication du code pour la classe widgetHorloge:	55
i - Diagramme de séquence de la classe widgetMusiqueUSB:	56
Explication du code pour la classe widgetMusiqueUSB:	57
IX - Conclusion personnel et remerciement	57
<b>CONTRAT Étudiant 3 IR GRANDON Luca</b>	<b>59</b>
I- Présentation du contrat	59
II- Ressources, matériel utilisées et choix technologiques	60
a- Ressources logiciels	60
b- Ressources Matériels	60
c- Choix technologiques	61
III- Présentation des diagrammes	62
a- Diagrammes des cas d'utilisation	62
b- Diagramme des classes	63
c- Diagramme de séquence de l'interface administrateur (Django)	64
d- Diagramme de séquence de la reconnaissance vocale	66
e- Diagramme de base de données	67
i - Base de données sous raspberry	68
IV - Interface administrateur	69
a - Idée de base	69
b - Interface administrateur actuelle	70
i - Page d'accueil	70
ii - Page de création de profil	
iii - Page de modification de profil	71

iiii - Page de suppression de profil	72
c - Exemple de code d'une page Django	72
V - Reconnaissance vocale	73
a - Explication	73
b - Programme	74
VI - Problèmes rencontrés	76
a - Django et Interface administrateur	76
b - Reconnaissance vocale	76
VII - Remerciements	77
VIII - Conclusion et évolutions	77
<b>CONTRAT Étudiant 4 IR CARDONA Alexandre</b>	<b>78</b>
I - Présentation du contrat	78
II - Ressources, matériel utilisées et choix technologiques	79
a - Ressources et matériels utilisés	79
b - Choix technologiques	79
III- Présentation des diagrammes et du code	81
a - Diagramme des cas d'utilisations	81
b - Diagramme de classe de la reconnaissance faciale	82
c - Diagrammes de la classe RenamePlusEncoding	84
d - Diagramme de séquence de la reconnaissance faciale	85
e - Diagramme de séquence de renamePlusEncoding	86
f - Diagramme de Base De Données	87
IV/ Problèmes rencontrés	88
V - Remerciement	90
VI - Conclusion et évolution	90
<b>Conclusion Générale &amp; amélioration possible :</b>	<b>91</b>
<b>Annexes :</b>	<b>93</b>
Partie étudiant 1 EC (HUPPERT Adrien) :	93
Programme Capteur	93
Partie étudiant 2 IR (LAPORTE Maxence):	95
ApplicationIHM	95
widgetMeteo :	97
widgetRadio :	100
controleSon :	102
modeAveugle :	103
user :	103
widgetHorloge :	110
widgetMusiqueUSB :	112
Ui_MainWindow :	114
Partie étudiant 3 IR (GRANDON Luca):	126
Programme de reconnaissance vocale :	127

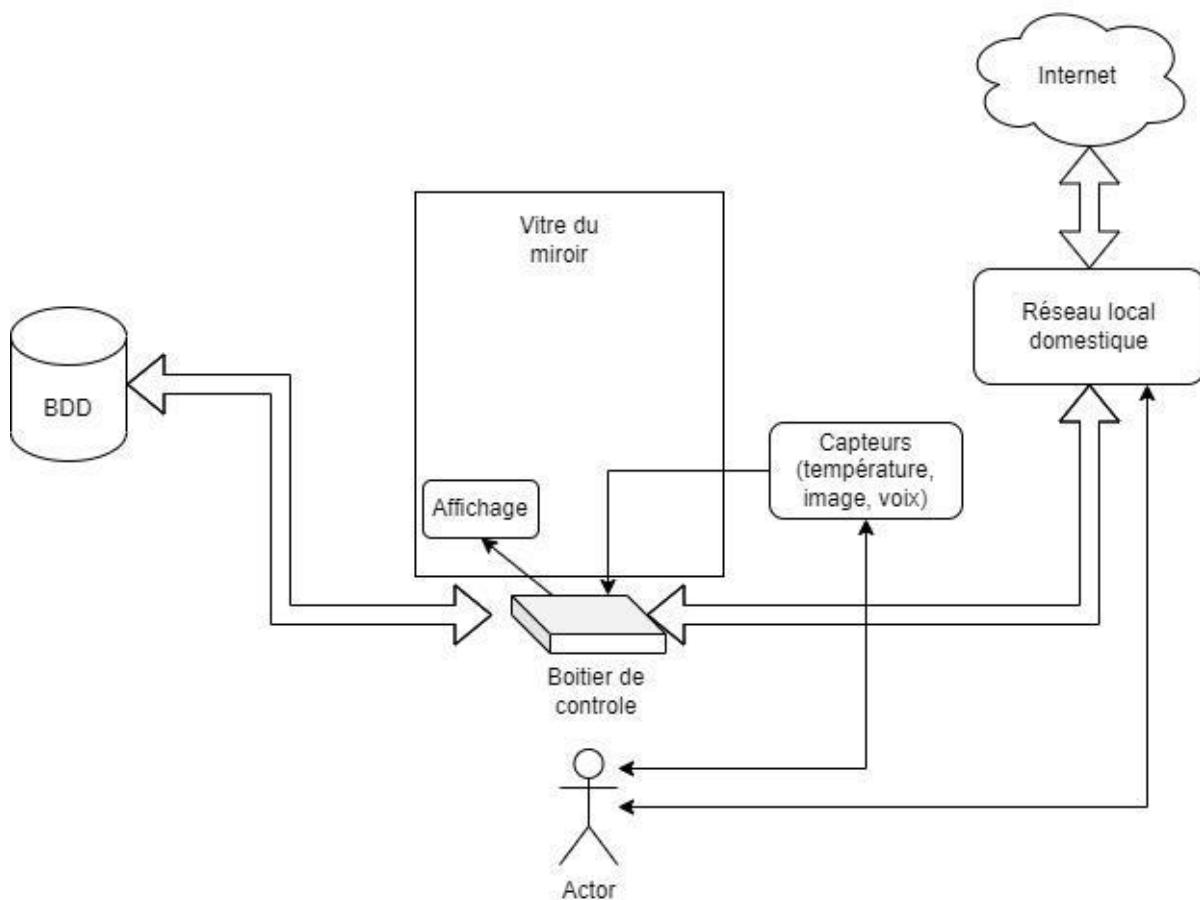
Code django / interface administrateur	132
Models.py :	132
views.py :	133
urls.py :	134
settings.py :	135
admin.py :	137
apps.py :	137
Partie étudiant 4 IR (CARDONA Alexandre):	138
Programme renamePlusEncoding :	139
Code recoF :	141

## Vue d'ensemble

Placé dans une salle de bain ou bien dans une pièce commune de la maison, Masha, le Miroir Connecté offre de nombreux services à l'utilisateur tels que jouer de la musique, mettre une station de radio, afficher la météo, la température et l'humidité de la pièce.

Grâce à sa reconnaissance faciale, Masha est capable de charger le profil, stockée dans une base de données en local sur la carte Raspberry Pi 4, de l'utilisateur afin de lui proposer de lancer directement sa station de radio préférée.

L'utilisateur est également capable de contrôler Masha à l'aide de sa voix. Par reconnaissance vocale, lancer une musique, un flux en direct de radio, allumer la lumière, et, pour les personnes malvoyantes, afin connaître toutes sortes d'informations affichées sur l'écran tels que le nom de la radio en cours de lecture, la météo, le profil chargé et plus encore.



## a - Objectifs

1. Faciliter l'accès à des informations et services.
2. Améliorer l'ergonomie au quotidien.
3. Permettre aux personnes malvoyantes d'utiliser Masha.

## b - Analyse de l'existant

Sur le marché, il y a déjà des miroirs connectés qui existent mais peu d'entre eux possède des fonctionnalités tel que le profil personnalisable ou encore de contrôle vocale. Souvent, les miroirs avec des options évoluées valent plusieurs centaines d'euros.

Notre projet permettra alors de réduire les coûts tout en ajoutant des services que nous pensons être essentiels. En plus d'être utile, Masha permet aux personnes malvoyantes de l'utiliser grâce à nos reconnaissance vocale, faciale mais aussi par des retours vocales là où les miroirs du commerce proposent une utilisation tactile. Grâce au système de capteur ultrason, Masha se met en mode veille, lorsque personne n'est devant, afin d'économiser de l'énergie.

## c - Expression du besoin

Le système développé répond aux besoins suivants :

- Au démarrage, la raspberry lance le programme et l'interface administrateur.
- Par défaut, les programmes sont en veille et ne se lancent uniquement lorsqu'un utilisateur est devant le miroir.
- Lorsqu'un utilisateur est devant Masha, la reconnaissance faciale se lance et envoie l'information de l'utilisateur. S'il est reconnu Masha affiche son profil, s'il ne l'est pas, Masha affichera le profil par défaut.
- Le système affiche des informations récupérées soit via Internet soit via des capteurs locaux.
- Le système permet d'écouter une radio spécifique, afficher la météo extérieur et intérieur, ainsi que la date et l'heure.
- Dans un souci d'intégration des différentes populations, Masha est adapté aux malvoyants de par ses multiples commandes vocales, et réponses sonores (date, heure, météo, ...).
- Pour de futures améliorations, le système stocke des statistiques sur les dates et durées d'utilisation de chaque profil.
- Via un site web, il est possible de créer des profils, de les modifier et de les supprimer afin de modifier la base de données.

## d - Ressources Humaines et répartition des tâches

Pour notre projet, nous sommes 3 étudiants d'option Informatique et Réseau, et 1 étudiant d'option Electronique et Communication dont les tâches ont été répartie comme suit :

-Étudiant IR 1 Maxence LAPORTE :

Création d'une Interface Homme-Machine qui propose, à l'aide de widgets, des services à l'utilisateur.

-Étudiant IR 2 Luca GRANDON :

Création d'une interface administrateur gérant des profils liés à une base de données ainsi que la programmation d'une reconnaissance vocale.

-Étudiant IR 3 Alexandre CARDONA:

Mise en place de la reconnaissance faciale, communication avec la base de données et collecte de statistique.

-Étudiant EC Adrien HUPPERTS:

Création de carte électronique pour gérer la température, l'humidité, la présence, la luminosité et la puissance grâce à différents composants.



## e - Coût matériel

Référence Produit	Nom produit	Prix unitaire	Quantité	Prix sans TVA	TVA	Prix avec TVA
	« FARNELL »					
3390473	Ruban de LED, 300 mm, 12 LED, Blanc froid, 12 V, 800 mW	14,63 €	2	29,26 €	0,2	35,11 €
139901	Microphone à pince	4.21	1	4.21	0.2	5,052€
	« LDLC-PRO »					
951550	Dexlan carte son USB-A	10.42	1	10.42	0.2	12.504€
JW_LCD_IPS_13IN	Johnwill moniteur 13,3 » LED PORTABLE	124,96	1	124,96	0,2	149,95 €
980000513	LOGITECH Z120	16,62	1	16,62	0,2	19,94 €
	« LEROY MERLIN »					
83379229	Dimexact filtre sans tain 100*152cm	25.01	1	25.01	0.2,	19€
83572072	Plaque Plexiglas Pmma Transparent Ep. 3 Mm L.70 X 100 Cm	15.2	1	15.2	0.2	31,27€
	« Gotronic »					

5441	Micro à électret M300	0,67	1	0,67	0,2	0,8
35861	Nappe 1m pour caméra pour Raspberry Pi	4,58	1	4,58	0,2	5,5
35897	Module caméra 5MP CAMERA-JT	11,58	1	11,58	0,2	13,9
11938	Radiateurs pour Raspberry Pi 4B COOLKIT6	2,92	1	2,92	0,2	3,5
	"INMAC WSTORE"					
7139916	Lindy High Speed HDMI Cable – câble HDMI avec Ethernet – 50cm	21,56	1	21,56	0,2	25,87

				Prix sans TVA totale	TVA	Prix avec TVA totale
		TOTAL EN €		266.99€	20%	320,38€

## Gantt et planification

	Plan...	Nom de tâche	Préde...	Durée	Début	Fin	Prog...	Priorité	Ressources	Travail	Coût	
1	→	Miroir connecté		31,5 jours?	05/01/2022	17/06/2022	80%	500		804,33 h	0,00 €	
2	✓	✉ Preparation début de projet/1er Revue		2,75 jours	05/01/2022	19/01/2022	100%	500		28,33 h	0,00 €	
3	✓	→ Attribution des rôles / contrats		0,25 jour	05/01/2022	05/01/2022	100%	500	GroupeMiroir	2 h	0,00 €	
4	✓	→ Lecture des documents, préparation règles(V.1) et mise en place des réunions hebdomadaires	3	0,25 jour	11/01/2022	11/01/2022	100%	500	GroupeMiroir	2 h	0,00 €	
5	✓	→ Réunion1 (Croquis/Rechercher/Gantt/Sujet)	4	0,04 jour	12/01/2022	12/01/2022	100%	500	GroupeMiroir	0,33 h	0,00 €	
6	✓	→ Création des croquis	4	0,25 jour	12/01/2022	12/01/2022	100%	500	Alexandre Cardona	2 h	0,00 €	
7	✓	→ recherche des matériaux	4	0,37 jour	14/01/2022	14/01/2022	100%	500	Alexandre Cardona	3 h	0,00 €	
8	✓	→ Diagramme UML	5	0,5 jour	14/01/2022	14/01/2022	100%	500	Maxence Laporte	4 h	0,00 €	
9	✓	→ Création du Gantt	4	0,75 jour	12/01/2022	12/01/2022	100%	500	Luca Grandon	6 h	0,00 €	
10	✓	→ Préparer bon de commande (V.1)	7	0,25 jour	14/01/2022	18/01/2022	100%	500	Alexandre Cardona	2 h	0,00 €	
11	✓	→ Réunion2(Problèmes et Solutions / préparation 1er revue)	10	0,37 jour	18/01/2022	18/01/2022	100%	500	GroupeMiroir	3 h	0,00 €	
12	✓	→ Diagramme Exigence (Découpage par contrat)	11	0,5 jour	19/01/2022	19/01/2022	100%	500	Luca Grandon	4 h	0,00 €	
13	→	✉ Soutenance et dossier		25,75 jours?	21/01/2022	14/06/2022	18%	500		22 h	0,00 €	
14	✓	→ Revue n°1	11	0,25 jour	21/01/2022	21/01/2022	100%	500	GroupeMiroir	2 h	0,00 €	
15	✓	→ Revue n°2	14	0,25 jour	04/04/2022	04/04/2022	100%	500	GroupeMiroir	2 h	0,00 €	
16	✓	→ Revue n°3	15	0,75 jour	25/05/2022	25/05/2022	0%	500	GroupeMiroir	6 h	0,00 €	
17	✓	→ Rendu du dossier	16	0,5 jour	27/05/2022	27/05/2022	0%	500	GroupeMiroir	4 h	0,00 €	
18	✓	→ Revue Finale	17	1 jour?	14/06/2022	14/06/2022	0%	500	GroupeMiroir	8 h	0,00 €	
19	→	✉ Contrat 2.4		29,75 jours	18/01/2022	17/06/2022	81%	500		754 h	0,00 €	
20	→	✉ Contrat 2.4.1 (Hupperts Adrien)		22 jours	18/01/2022	27/05/2022	77%	500		176 h	0,00 €	
21	→	✉ Détection de présence, gestion de lampe et capteur		22 jours	18/01/2022	27/05/2022	77%	500		176 h	0,00 €	
22	✓	→ Recherche bande led usb	5	0,5 jour	18/01/2022	18/01/2022	100%	500	Adrien Hupperts	4 h	0,00 €	
23	✓	→ Schémas d'alimentation (idée)	22	0,75 jour	19/01/2022	19/01/2022	100%	500	Adrien Hupperts	6 h	0,00 €	
24	✓	→ Carte électronique	23	3 jours	21/01/2022	02/02/2022	100%	500	Adrien Hupperts	24 h	0,00 €	
25	✓	→ Installer et gérer les composants (capteurs)	24	2,25 jours	02/02/2022	01/03/2022	100%	500	Adrien Hupperts	16 h	0,00 €	
26	✓	→ Code	25	2,5 jours	02/03/2022	09/03/2022	100%	500	Adrien Hupperts	20 h	0,00 €	
27	✓	→ Carte alimentation	26	5,5 jours	11/03/2022	04/04/2022	100%	500	Adrien Hupperts	44 h	0,00 €	
28	✓	→ Carte électronique 2	27	1,25 jours	06/04/2022	26/04/2022	100%	500	Adrien Hupperts	10 h	0,00 €	
29	✓	→ Gérer l'alimentation	28	2,5 jours	27/04/2022	10/05/2022	50%	500	Adrien Hupperts	20 h	0,00 €	
30	✓	→ Mise en veille + capteurs	29	3,75 jours	11/05/2022	27/05/2022	0%	500	Adrien Hupperts	30 h	0,00 €	
31	→	✉ Contrat 2.4.2 (Laporte Maxence)		20,75 jours	18/01/2022	24/05/2022	94%	500		166 h	0,00 €	
32	→	✉ Crée l'IHM (Interface homme-machine)		20,75 jours	18/01/2022	24/05/2022	94%	500		166 h	0,00 €	
33	✓	→ Recherches liées au film sans teint	5	0,5 jour	18/01/2022	18/01/2022	100%	500	Maxence Laporte	4 h	0,00 €	
34	✓	→ Apprendre QT Python	33	1,75 jours	19/01/2022	25/01/2022	100%	500	Maxence Laporte	14 h	0,00 €	
35	✓	→ Récupérer le site météo (Widget)	34	0,75 jour	26/01/2022	26/01/2022	100%	500	Maxence Laporte	6 h	0,00 €	
36	✓	→ Afficher la date / l'heure / la météo (Widget)	35	2,25 jours	28/01/2022	22/02/2022	100%	500	Maxence Laporte	18 h	0,00 €	
37	✓	→ Récupérer le(s) site(s) de radio (Widget)	36	0,75 jour	23/02/2022	23/02/2022	100%	500	Maxence Laporte	6 h	0,00 €	
38	✓	→ Récupérer des musiques via USB	37	1,75 jours	01/03/2022	04/03/2022	100%	500	Maxence Laporte	14 h	0,00 €	
39	✓	→ Réparation musiques	38	0,5 jour	08/03/2022	08/03/2022	100%	500	Maxence Laporte	4 h	0,00 €	
40	✓	→ Faire les threads (QtPython)	39	0,75 jour	09/03/2022	09/03/2022	100%	500	Maxence Laporte	6 h	0,00 €	
41	✓	→ Fusion IHM/recof	40	0,5 jour	11/03/2022	11/03/2022	100%	500	Maxence Laporte	4 h	0,00 €	
42	✓	→ Fusion IHM/Recof 2	41	0,5 jour	15/03/2022	15/03/2022	100%	500	Maxence Laporte	4 h	0,00 €	
43	✓	→ Widget Mode Aveugle	42	5,75 jours	16/03/2022	26/04/2022	100%	500	Maxence Laporte	46 h	0,00 €	
44	✓	→ Exporter le projet sous Linux (portabilité raspberry)	43	1,25 jours	27/04/2022	03/05/2022	100%	500	Maxence Laporte	10 h	0,00 €	
45	✓	→ Fusion totalité programme (Recoff/IhmRecov)	44	0,75 jour	04/05/2022	04/05/2022	100%	500	Maxence Laporte	6 h	0,00 €	
46	✓	→ Refaire l'IHM (supprimer les boutons)	45	0,5 jour	10/05/2022	10/05/2022	100%	500	Maxence Laporte	4 h	0,00 €	
47	✓	→ Charger un profil (BDD et Interface Admin Contrat IR 2)	46	0,75 jour	11/05/2022	11/05/2022	100%	500	Maxence Laporte	6 h	0,00 €	
48	✓	→ Appliquer les commandes vocales (Widget / Lumière)	47	0,5 jour	17/05/2022	17/05/2022	100%	500	Maxence Laporte	4 h	0,00 €	
49	✓	→ Récupération des informations capteurs ambiant (Contrat EC)	48	0,75 jour	18/05/2022	18/05/2022	0%	500	Maxence Laporte	6 h	0,00 €	
50	✓	→ Faire en sorte que l'exécutable s'exécute au lancement de la raspberry	49	0,5 jour	24/05/2022	24/05/2022	0%	500	Maxence Laporte	4 h	0,00 €	
51	✓	→ Contrat 2.4.3 (Grandon Luca)		19,5 jours?	18/01/2022	17/05/2022	100%	500		172 h	0,00 €	
52	✓	→ Administration		19,5 jours?	18/01/2022	17/05/2022	100%	500		86 h	0,00 €	
53	✓	→ Documentation Django/Python		3,75 jours	18/01/2022	28/01/2022	100%	500	Luca Grandon	28 h	0,00 €	
54	✓	→ Crée la table "profil" dans la BDD	53	0,5 jour	01/02/2022	01/02/2022	100%	500	Luca Grandon	4 h	0,00 €	
55	✓	→ Remplir la table profil (défaut)	54	2 jours	02/02/2022	23/02/2022	100%	500	Luca Grandon	16 h	0,00 €	
56	✓	→ Permettre de pouvoir créer des profils dans la bdd via l'interface	55	1,25 jours	01/03/2022	02/03/2022	100%	500	Luca Grandon	10 h	0,00 €	
57	✓	→ Permettre de pouvoir modifier et supprimer des profils de la bdd	56	1 jour	04/03/2022	06/03/2022	100%	500	Luca Grandon	8 h	0,00 €	
58	✓	→ Migration raspberry	57	0,75 jour	04/05/2022	04/05/2022	100%	500	Luca Grandon	6 h	0,00 €	
59	✓	→ Régler le problèmes BDD / Image	58	1,75 jours	10/05/2022	17/05/2022	100%	500	Luca Grandon	14 h	0,00 €	
60	✓	→ Gérer les commandes vocales		11,25 jours?	09/03/2022	17/05/2022	100%	500		86 h	0,00 €	
61	✓	→ Renseignement SpeechRecognition et PyAudio	59	2,5 jours	09/03/2022	16/03/2022	100%	500	Luca Grandon	20 h	0,00 €	
62	✓	→ Codier la reconnaissance vocal (Python)	60	6,5 jours	22/03/2022	04/05/2022	100%	500	Luca Grandon	52 h	0,00 €	
63	✓	→ Fusion IHM/Recov	62	0,5 jour	10/05/2022	10/05/2022	100%	500	Luca Grandon	4 h	0,00 €	
64	✓	→ Résoudre tout les soucis	63	1,25 jours	11/05/2022	17/05/2022	100%	500	Luca Grandon	10 h	0,00 €	
65	→	✉ Contrat 2.4.4 (Cardona Alexandre)		19,5 jours?	18/01/2022	17/05/2022	98%	500		148 h	0,00 €	
66	→	✉ Reconnaissance faciale par caméra		19,5 jours?	18/01/2022	17/05/2022	98%	500		148 h	0,00 €	
67	✓	→ Installation Linux / ajout bibliothèque OpenCV		1,25 jour	18/01/2022	19/01/2022	100%	500	Alexandre Cardona	10 h	0,00 €	
68	✓	→ Configuration OpenCV	67	1 jour	21/01/2022	25/01/2022	100%	500	Alexandre Cardona	8 h	0,00 €	
69	✓	→ Documentation OpenCV (apprentissage)	68	0,75 jour	26/01/2022	26/01/2022	100%	500	Alexandre Cardona	6 h	0,00 €	
70	✓	→ "Programmer" le système de reconnaissance faciale	69	2,25 jours	28/01/2022	22/02/2022	100%	500	Alexandre Cardona	18 h	0,00 €	
71	✓	→ Test du fonctionnement du matériels	70	0,75 jour	23/02/2022	23/02/2022	100%	500	Alexandre Cardona	6 h	0,00 €	
72	✓	→ Réparation code	71	0,5 jour	01/03/2022	01/03/2022	100%	500	Alexandre Cardona	4 h	0,00 €	
73	✓	→ Lier la reconnaissance faciale et la BDD	72	0,75 jour	02/03/2022	02/03/2022	100%	500	Alexandre Cardona	6 h	0,00 €	
74	✓	→ Faire en sorte que les données et statistiques de la RF se stocke dans la BDD	73	0,5 jour	04/03/2022	04/03/2022	100%	500	Alexandre Cardona	4 h	0,00 €	
75	✓	→ Création des classes	74	0,5 jour	08/03/2022	08/03/2022	100%	500	Alexandre Cardona	4 h	0,00 €	
76	✓	→ Fusion IHM / recof	75	0,75 jour	09/03/2022	09/03/2022	100%	500	Alexandre Cardona	6 h	0,00 €	
77	✓	→ Upgrade du programme	76	2,5 jours	16/03/2022	23/03/2022	100%	500	Alexandre Cardona	20 h	0,00 €	
78	✓	→ Fusion IHM/recof 2	77	0,5 jour	25/03/2022	25/03/2022	100%	500	Alexandre Cardona	4 h	0,00 €	
79	✓	→ Modification pour migration global vers raspberry	78	2,75 jours	30/03/2022	26/04/2022	100%	500	Alexandre Cardona	22 h	0,00 €	
80	✓	→ Protocole de test et test	79	0,75 jour	27/04/2022	27/04/2022	100%	500	Alexandre Cardona	6 h	0,00 €	
81	✓	→ Résoudre les problèmes	80	1,25 jours	03/05/2022	04/05/2022	100%	500	Alexandre Cardona	10 h	0,00 €	
82	✓	→ Mise en lien encodage image via Django (BDD/Python)	81	1,75 jours	10/05/2022	17/05/2022	75%	500	Alexandre Cardona	14 h	0,00 €	
83	→	✉ Contrat 2.4.5		11,5 jours?	11/05/2022	17/06/2022	3%	500		92 h	0,00 €	
84	→	✉ Prototype		7,75 jours?	31/05/2022	17/06/2022	0%	500		62 h	0,00 €	
85	✓	→ Construire le prototype		74;30,6...	7,75 jours	31/05/2022	17/06/2022	0%	500	GroupeMiroir	62 h	0,00 €
86	→	✉ Dossier		3,75 jours	11/05/2022	27/05/2022	10%	500		30 h	0,00 €	
87	✓	→ Écrire le dossier (Documentation, rapport de réunion, mise en oeuvre)		3,75 jours	11/05/2022	27/05/2022	10%	500	GroupeMiroir	30 h	0,00 €	

## Mode d'emploi :

Reconnaissance vocale :

### Commande en lien avec la radio

Pour lancer la radio préférée de l'utilisateur, il suffit de dire « mets la radio » au miroir.

Afin de stopper la radio, il suffit de donner la phrase « arrête la radio ».

Afin de savoir quelle radio est en cours on peut simplement donner une phrase contenant « Radio en cours » comme par exemple, « quelle est la radio en cours ? ».

Ensuite, afin de pouvoir changer la radio il suffit simplement de dire le mot clé « mets » avec le nom de la radio comme par exemple, « Mets NRJ ».

Les autres commandes pour pouvoir changer la radio y ressemblent, ne changeant que le nom de la radio.

Voici donc les différentes radios disponibles et donc leurs commandes vocales

Nom de la radio	Nom de la commande vocale
NRJ	« Mets NRJ »
France info	« Mets France info »
France musique	« Mets France musique »
Chérie FM	« Mets CherieFM »
Rire & Chanson	« Mets Rire et Chanson »
RTL	« Mets RTL »
RTL2	« Mets RTL2 »
Fun Radio France	« Mets Fun Radio France »
Europe1	« Mets Europe1 »
Virgin radio	« Mets Virgin radio »
RFM	« Mets RFM »
RMC	« Mets RMC »



BFM Business	« Mets BFM Business »
Skyrock	« Mets Skyrock »
Radio Classique	« Mets Radio Classique »
France Inter	« Mets France Inter »
France Culture	« Mets France Culture »
FIP Nationale	« Mets FIP Nationale »
Mouv'	« Mets Mouv' »
OuïFM	« Mets NRJ »
Jazz Radio	« Mets Jazz Radio »
M Radio	« Mets M Radio »
France Bleu	« Mets France bleu»

#### Commande en lien avec le miroir :

Afin d'allumer la bande led sur le miroir, il suffit de donner la commande vocale « Allume la lumière ».

Pour éteindre la bande led sur le miroir, il suffit de donner la commande vocale « Éteint la lumière ».

#### Commande en lien avec l'IHM :

Pour obtenir les informations météorologiques à l'heure actuelle, il suffit de demander au miroir « Quelle va être la météo aujourd'hui ? » par exemple car votre phrase doit contenir le mot clé, « Météo ».

Afin d'obtenir les informations permettant d'avoir accès aux résultats des capteurs et donc d'obtenir l'information telle que la température ou l'humidité de la pièce, il suffit de faire des phrases comprenant le mot « Température » comme « Quelle est la température dans la pièce ? » ou encore avec le mot « humidité » tel que « Quelle est le taux d'humidité dans la pièce ? ».

Afin d'obtenir tout d'abord l'heure à haute voix, il suffit de faire une phrase comportant le mot clé « HEURE » comme par exemple : « Quel heure est-il ? ».

Pour connaître la date du jour vous pouvez dire une phrase comportant le mot « Date » comme, « Quelle est la date d'aujourd'hui ? »

Afin de savoir quel profil est chargé actuellement sur votre miroir vous pouvez lui demander « Quel est le profil chargé ? », sachant que la phrase doit contenir les mots clés « Profil » et « chargé »

Pour pouvoir enclencher le minuteur il suffit de donner la commande vocale « Mets un minuteur ».

Afin de le stopper il suffit de donner la commande vocale « arrêter le minuteur ».

Si on veut annuler un minuteur avant de l'enclencher, la commande « Annuler le minuteur » est faite pour remettre à 0 celui-ci comme si il n'avait jamais existé.

Pour créer un minuteur, la commande « Créer un minuteur » est celle à utiliser, créant un minuteur prêt à être utilisé.

#### Commande en lien avec la musique via USB :

Afin de lancer la musique présente sur la clé USB dans l'ordre, il suffit de demander « mets de la musique » au miroir.

Si vous voulez que le miroir arrête de jouer de la musique vous pouvez simplement lui dire « Arrête la musique ».

Dans le cas où vous appréciez une musique mais que vous voulez le titre de celle-ci vous pouvez demander « Quelle est la musique en cours » car votre phrase devra contenir les mots clés « Musique », « en » et « cours ».

Pour pouvoir passer à la musique suivante, la commande vocale est « Musique suivante » et si vous voulez écouter la musique précédente il s'agira de la commande vocale « Musique précédente ».

Mode d'emploi de l'interface administrateur django :

Création de profil

Afin de créer un profil dans la base de données il suffit de remplir le formulaire sur la page "(ip de la machine):8000/profil/add/" donnant accès à la page suivante.

## Bienvenue sur la section création d'utilisateur !

**(N'oubliez pas de modifier les paramètres par défauts!)**

Pseudo :

Nom :

Prenom :

Ville :

RadioPreferee :  ▾

DateNaissance :

ImageUtilisateur :  Aucun fichier sélectionné.

[Retour à tous les profils](#) :

Il vous faudra modifier les paramètres par défaut selon vos informations, je vais prendre mon exemple en créant un profil.

## Bienvenue sur la section création d'utilisateur !

(N'oubliez pas de modifier les paramètres par défauts!)

Pseudo :

Nom :

Prénom :

Ville :

RadioPreferee :  ▾

DateNaissance :

ImageUtilisateur :  20220527\_150750.jpg

[Retour à tous les profils](#) :

Après il suffit d'appuyer sur "envoyer" pour sauvegarder le profil ce qui envoie sur un récapitulatif

### Profil détaillé : Arca

- Prénom : Luca
- Nom : Grandon
- Ville : Aix-en-Provence
- Date de Naissance : 21 février 2002
- Radio préférée : VirginRadio
- Image utilisateur : profil/20220527\_150750\_PWW72wi.jpg

[Retour à tous les profils](#) :

si jamais on doit modifier le profil ensuite il suffit de retourner sur la page d'accueil de l'outil d'administration et à côté du profil "Arca", j'appuie sur "modifier".

- [Arca](#) - [\[modifier\]](#) - [\[supprimer\]](#)

Après avoir appuyer sur modifier, la page suivante s'ouvre, permettant de changer à nouveau les informations, une fois terminé appuyez sur envoyer, cela modifera le profil.

## Mise à jour de l'utilisateur :

(La date de naissance est en JJ/MM/YYYY, n'oubliez pas les "/" entre vos nombres)

Pseudo :

Nom :

Prenom :

Ville :

RadioPreferee :  ▾

DateNaissance :

ImageUtilisateur : Actuellement: [profil/20220527\\_150750\\_o0EyLLh.jpg](#)

Modification:  Aucun fichier sélectionné.

[Retour à tous les profils :](#)

Ensuite, si le profil créé doit être supprimé, il suffit d'appuyer sur le bouton "supprimer" à la droite du nom du profil.

- [Arca](#) - [\[modifier\]](#) - [\[supprimer\]](#)

La page suivante apparaît vous demandant si vous voulez bien supprimer le profil.

## Supprimer le groupe

Etes-vous sûr de vouloir supprimer le groupe : Arca / Luca ?

[Retour à tous les profils :](#)

Si vous appuyez sur "supprimer" le profil sera supprimé et donc ne sera plus visible dans la liste des profils.

## Utilisateurs :

- [Default](#) - [\[modifier\]](#) - [\[supprimer\]](#)

## CONTRAT Étudiant 1 EC HUPPERTS Adrien

### I- Présentation du contrat

Mon contrat est destiné à la création d'une carte électronique, et à la gestion de capteurs de températures, de présence, et d'humidité.

Gestion de la détection de présence: Par l'intermédiaire d'un capteur, le système détecte la présence d'un utilisateur et se met en veille ou se réveille en fonction de cela dans un souci d'économie d'énergie.

Détection de la température et de l'humidité intérieur: Par un ou des capteurs à déterminer, une détection de la température et de l'humidité dans la pièce du miroir sont mesurées à 0.1° C prêt.

Gestion de la lampe du miroir en fonction de la luminosité: En fonction de la luminosité ambiante (à détecter par un capteur), la lumière du miroir est allumée ou éteinte si un utilisateur est présent. Si aucun utilisateur n'est présent, la lumière reste éteinte.

### II- Logiciel utilisé

- Google Drive

Pour la communication entre l'équipe et les échanges de documents.

- Proteus

Pour réaliser la carte électronique et des schémas

- Python

Pour réaliser le programme pour gérer les composants

- MindView

Pour gérer l'emploi du temps

- Draw.io

Pour créer différents diagrammes

- Google Chrome

Pour différentes recherches

### III- Ressources et matériaux utilisés

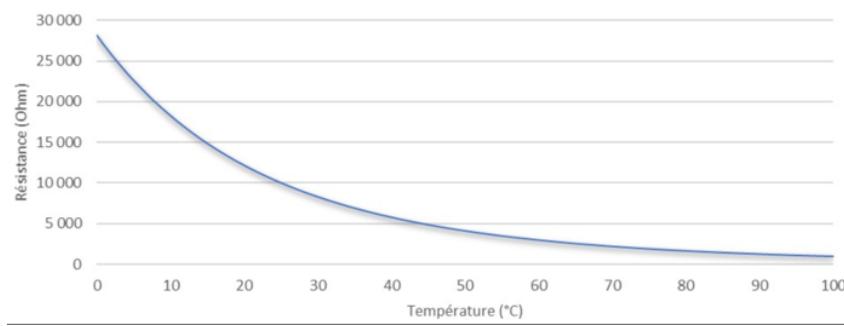
#### A- Capteurs

##### a- Température : CTN et LM35

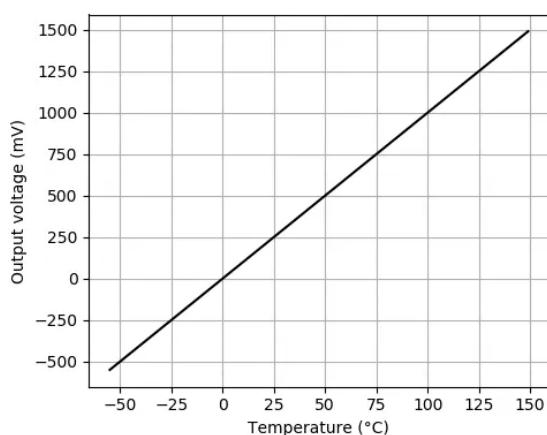
La CTN (10k) (coefficients de température négatif) est une thermistance, c'est-à-dire un capteur de température analogique. Sa résistance varie en fonction de la température: elle diminue de façon uniforme lorsque la température augmente, et inversement.

La valeur  $10\text{k}\Omega$  correspond à la valeur de la CTN à  $25^\circ\text{C}$ .

J'utilise ce composant sur une petite plage de température et grâce à la résistance  $10\text{k}\Omega$  que nous lui associons nous pouvons considérer que sa variation sera linéaire.



Le LM35 est un capteur de température centigrade, c'est-à- dire divisé en cent degrés, de précision. Il fournit un écart de température de  $-55^\circ\text{C}$  à  $+155^\circ\text{C}$ .



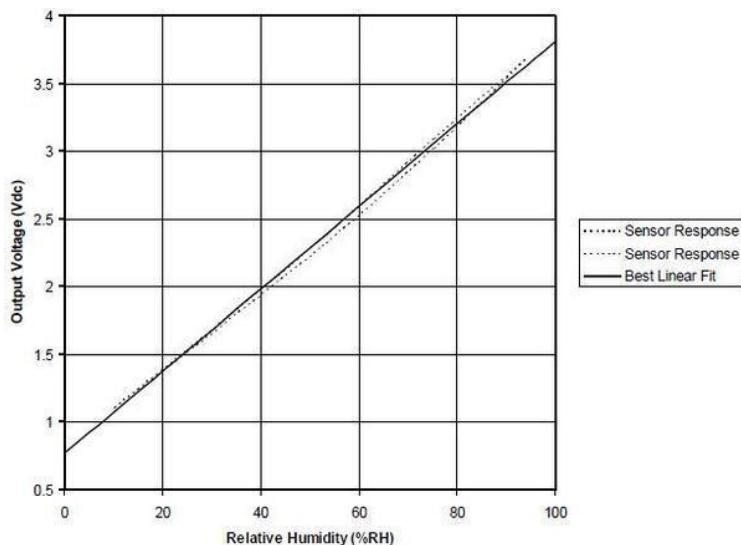
## b- Humidité : HIH 4030

Le capteur HIH-4030 est un instrument de mesure du taux d'humidité analogique.

La mesure de l'humidité ou hygrométrie est mesurée en pourcentage. Dans une pièce, le taux d'humidité se situe normalement entre 40 et 60 %.

La mesure de l'humidité : elle se fait à l'aide d'un condensateur dont la valeur de la capacité C est modifiée selon le niveau d'humidité.

A la sortie du capteur on va avoir une tension proportionnelle à l'humidité.



### c- Présence : HC SR04

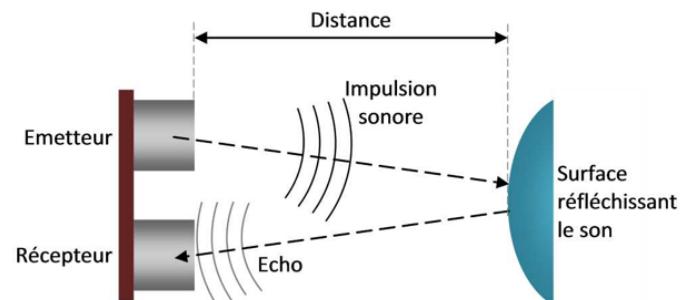
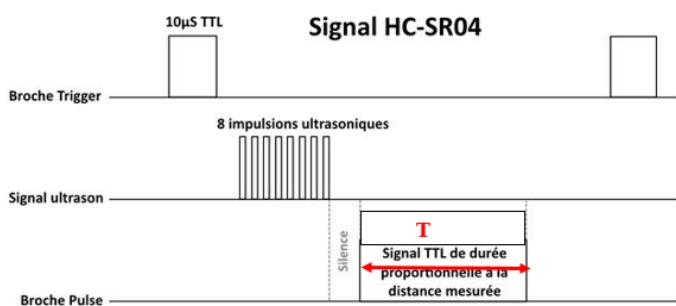
Le capteur HC-SR04 est numérique et utilise les ultrasons pour déterminer la distance d'un objet. Il offre une excellente plage de détection sans contact, avec des mesures de haute précision et stables.

Le principe de fonctionnement du capteur est entièrement basé sur la vitesse du son.

Voilà comment se déroule une prise de mesure :

- On envoie une impulsion HIGH de  $10\mu s$  sur la broche TRIGGER du capteur.
- Le capteur envoie alors une salve de 8 impulsions ultrasoniques à 40KHz
- Les ultrasons se propagent dans l'air jusqu'à toucher un obstacle et retourne dans l'autre sens vers le capteur.
- Le capteur détecte l'écho et clôture la prise de mesure.

#### **Capteur HC-SR04**



## B- Résistances

La résistance est un dipôle qui joue un rôle de protection sur les autres dipôles d'un circuit qui ne résistent pas à un surplus d'intensité. Plus la valeur d'une résistance est élevée, plus l'intensité du courant est faible.

Dans ce circuit électrique il y a 6 résistances :

**2x**     $1\text{k}\Omega$

**2x**     $1,2\text{k}\Omega$

$5\text{k}\Omega$

$10\text{k}\Omega$

**$1,2\text{k}\Omega$**



**$10\text{k}\Omega$**



**$5\text{k}\Omega$**



**$1\text{k}\Omega$**



## C- PCF8591 (Convertisseur Analogique Numérique)

Le PCF8591 est un convertisseur CAN à 8 bits, ce qui signifie que la tension peut prendre 255 valeurs différentes échelonnées entre 0 et 3,3 V.

Un convertisseur analogique numérique (CAN) est un dispositif électronique permettant la conversion d'un signal analogique en un signal numérique

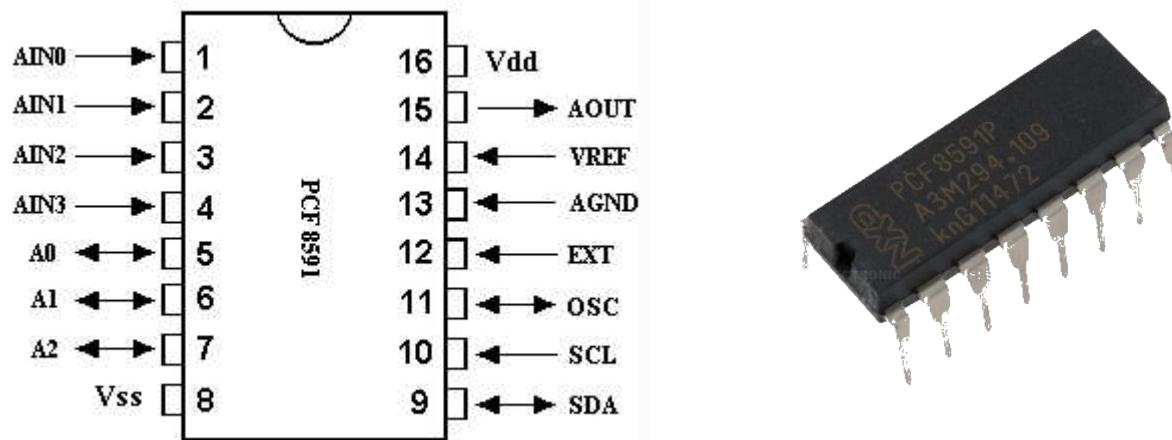
Signal analogique : signal continu en temps et en amplitude.

Signal numérique : signal échantillonné et quantifié, discret en temps et en amplitude.

- Conversion à approximation successive.
- Conversion sur 8 bits
- Communique sur bus I2C

Le PCF8591 supporte 4 entrées et une sortie analogique, toutes d'une résolution de 8 bits. Les broches AIN0 à AIN3 correspondent aux entrées analogiques et la broche AOUT à la sortie analogique.

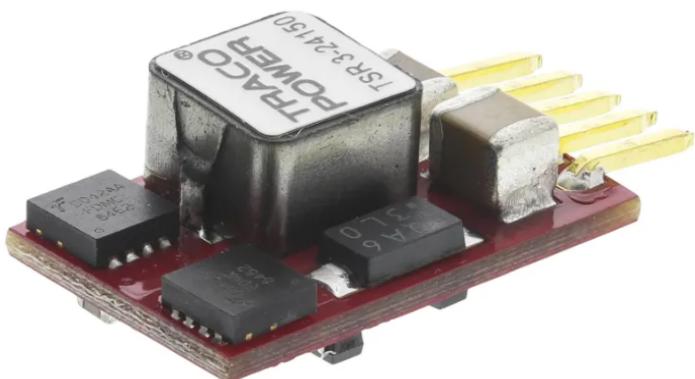
Les broches VREF et AGND correspondent aux tensions de références pour les conversions dans un sens ou dans l'autre.



## D-TSR 3-24150 (Traco Power)

Les séries TSR 3 comportent une entrée marche / arrêt à distance, une protection contre les courts-circuits et une protection contre la surchauffe. Ces modèles sont idéaux pour les points de charge et pour des applications très fiables. Convient pour des sorties de circuit positives et négatives.

Pinout		
Pin	positive	negative
1	Remote On/Off	
2	+Vin (Vcc)	
3	GND	-Vout
4	+Vout	GND
5	Trim	

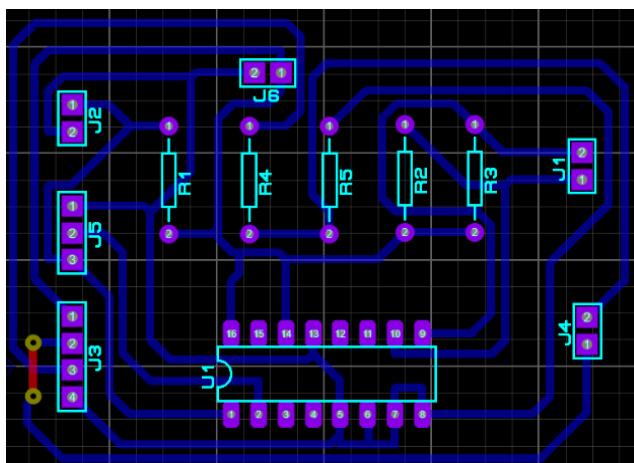
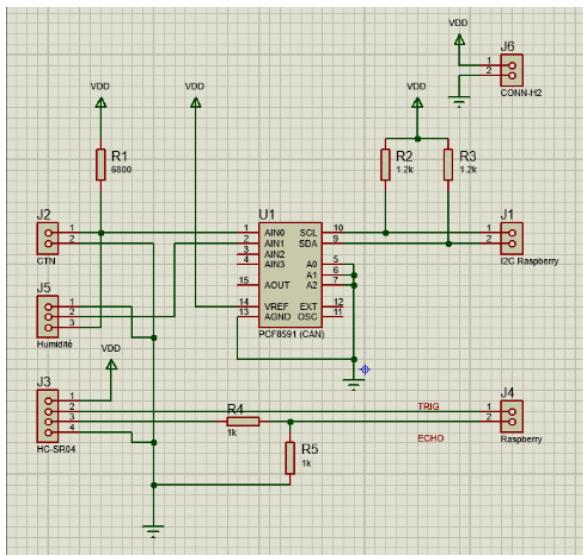


## IV- Carte électronique

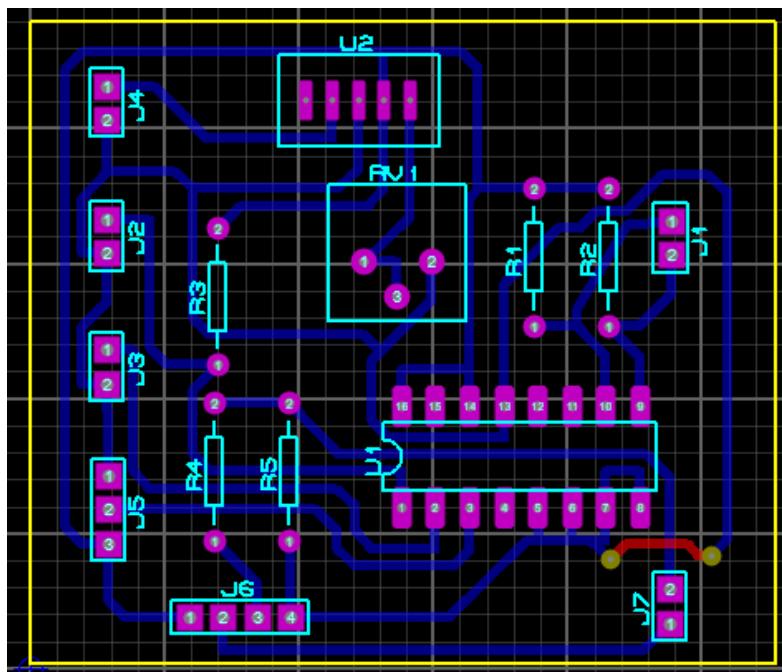
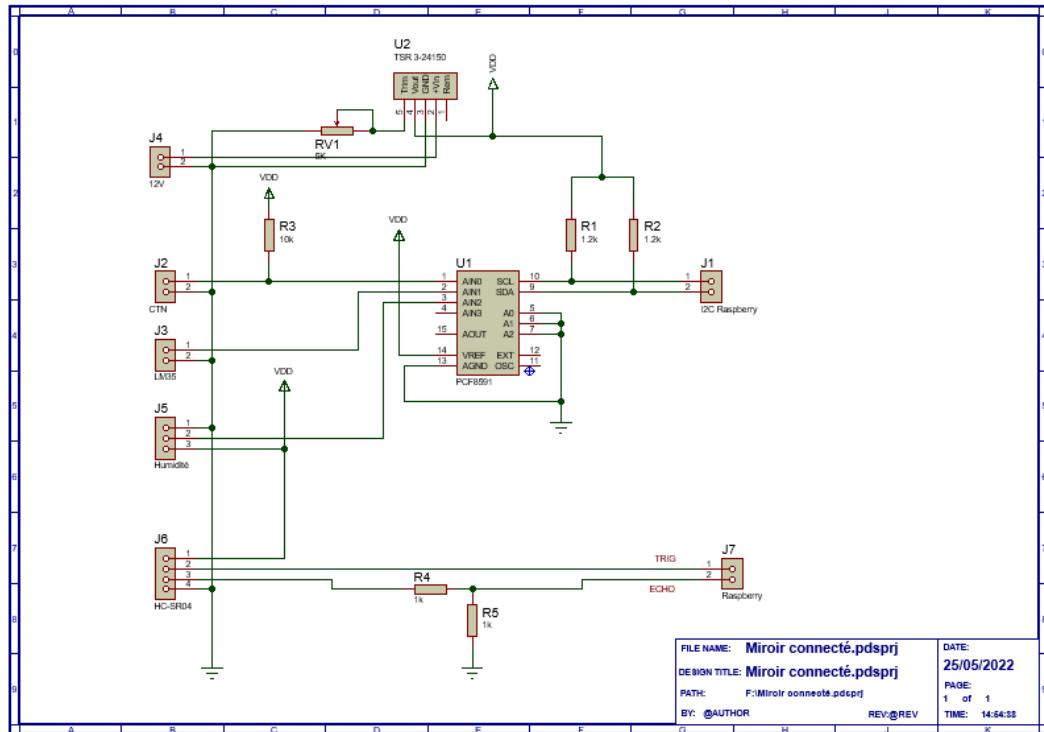
La carte à été dessinée sur le logiciel Proteus. Elle fait 5,4 cm sur 4,8 cm. Elle contient les différents composants présentés précédemment. Nous avons préféré minimiser un maximum la taille de la carte pour pouvoir économiser un maximum la place dans le miroir (tout comme le choix des composants, leurs tailles ont été l'une des caractéristiques clé lors des choix d'achats).

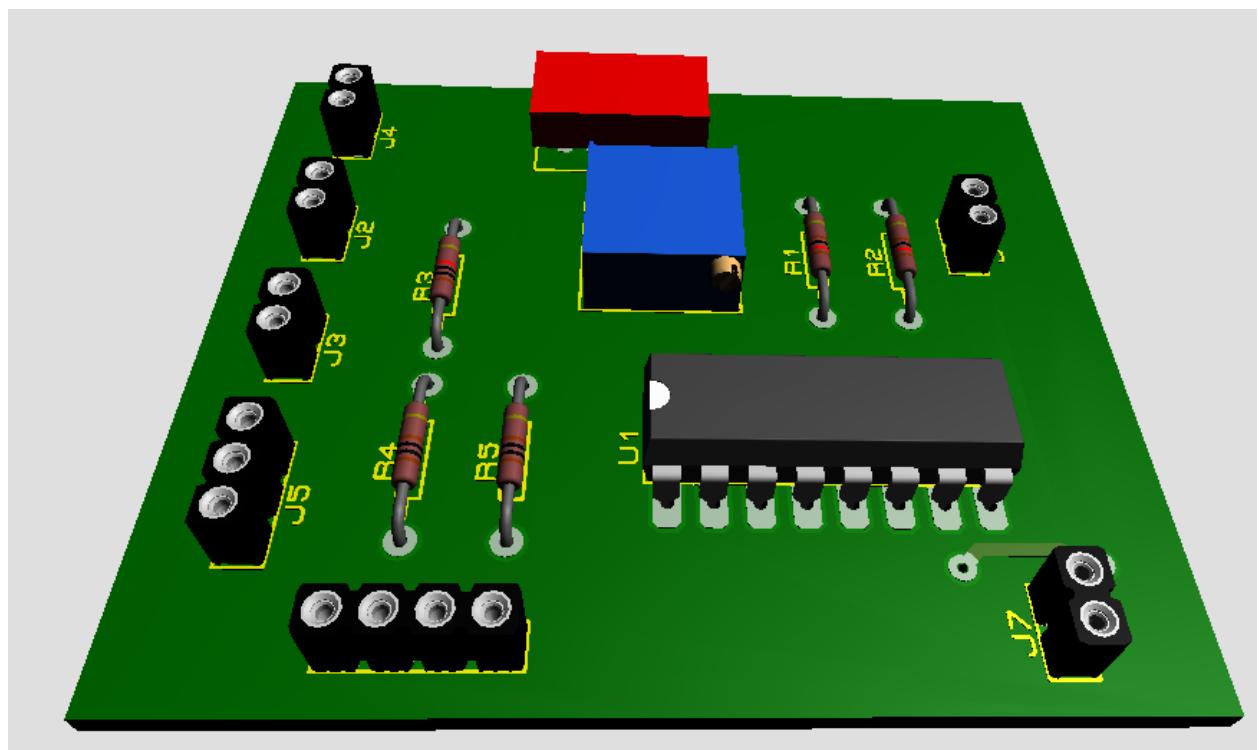
Au total deux cartes ont été créées, la première contient seulement des résistances, un convertisseur analogique numérique et des connectiques pour brancher les différents capteurs . Elle ne contient pas de composants concernant la puissance.

J2,J5,J3 correspondent aux composants pour la température, l'humidité et la présence, J1 et J2 correspondent eux à la raspberry.



La deuxième carte prend en compte des composants de puissances, tel que le Traco Power .

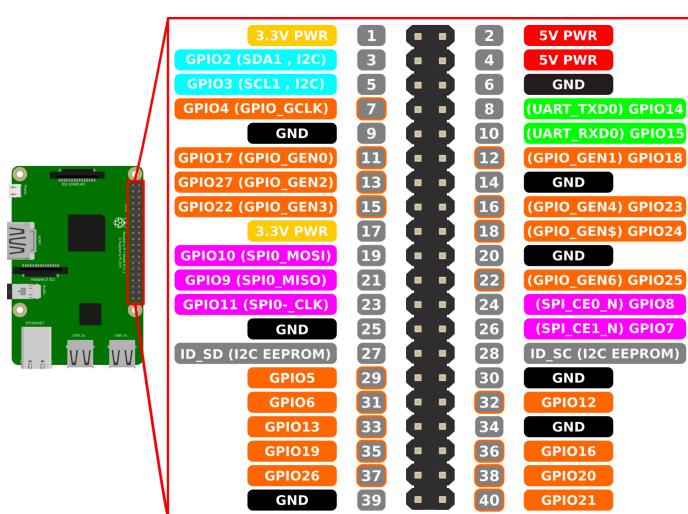
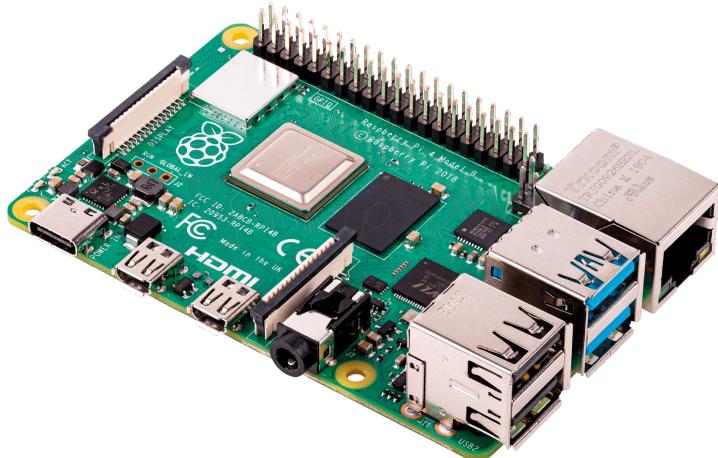




## V- Raspberry Pi 4

Grâce à son poids réduit, son faible coût, et ses sorties GPIO, couplées avec la possibilité d'ajout de modules, la Raspberry Pi est actuellement le meilleur choix possible comme ordinateur embarqué. C'est elle qui nous permettra de traiter les données numériques des différents composants.

Nous avons préféré utiliser une Raspberry pour sa taille et pour sa puissance, mais aussi pour pouvoir commander directement en Python.



## VI- Code

```
import smbus
import time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)
bus = smbus.SMBus(1)

def Temperature():
    adresse=0x48
    entree=0x40
    bus.write_byte(adresse, entree)
    bus.read_byte(adresse)
    temp= bus.read_byte(adresse)
    print("Température: %1.2f °C" %(-17.9*(temp*5/255)+77.5))
    return Temperature
def Humidite():
    adresse=0x48
    entree=0x42
    bus.write_byte(adresse, entree)
    bus.read_byte(adresse)
    bus.read_byte(adresse)
    V= bus.read_byte(adresse)
    print("Humidite: %1.2f pourcentage" %(32.2*(V*5/255)-24.5))
    time.sleep(2)
    return Humidite

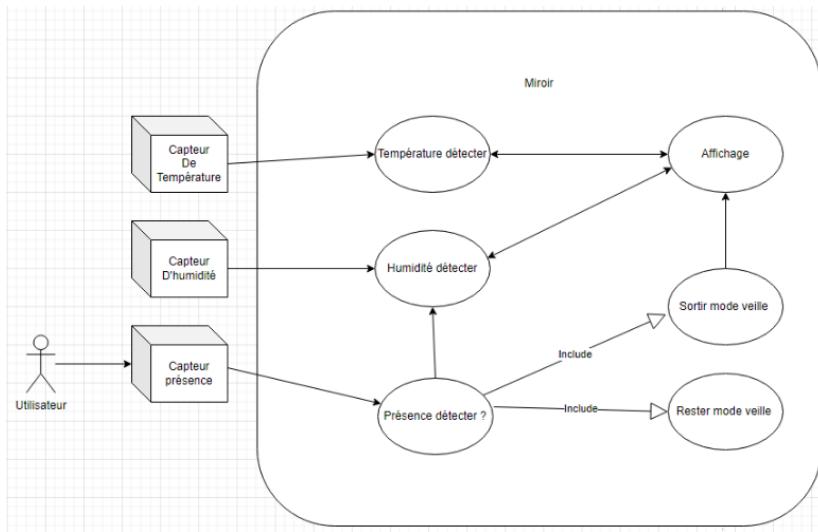
def Distance():
    GPIO_TRIGGER = 16
    GPIO_ECHO = 18

    GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
    GPIO.setup(GPIO_ECHO, GPIO.IN)
    GPIO.output(GPIO_TRIGGER, True)
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)

    StartTime = time.time()
    StopTime = time.time()
```

```
while GPIO.input(GPIO_ECHO) == 0:  
    StartTime = time.time()  
  
while GPIO.input(GPIO_ECHO) == 1:  
    StopTime = time.time()  
  
T = StopTime - StartTime  
dist= round((T*34300) / 2,0)  
print ("Mesure Distance = %.1f cm" % dist)  
time.sleep(1)  
  
return Distance  
  
def destroy():  
    adresse=0x48  
    bus.write_byte(adresse, 0x40)  
    bus.read_byte_data(adresse,0x41)  
  
while (1):  
    Humidite()  
    time.sleep(2)  
    Temperature()  
    time.sleep(2)  
    #destroy()  
    # dist = Distance()
```

## VII- Diagramme d'utilisation



## VIII- Gantt

	!	Plan...	Nom de tâche	Préde...	Durée	Début	Fin	Progr...	Priorité	Ressources	Travail	
1		➡	Miroir connecté		31,5 jours?	05/01/2022	17/06/2022	80%	500		804,33 h	
19		➡	☐ Contrat 2.4		29,75 jours	18/01/2022	17/06/2022	82%	500		754 h	
20		➡	☐ Contrat 2.4.1 (Hupperts Adrien)		22 jours	18/01/2022	27/05/2022	77%	500		176 h	
21		➡	☐ Détection de présence, gestion de lampe et capteur		22 jours	18/01/2022	27/05/2022	77%	500		176 h	
22	✓	➡	Recherche bande led usb		0,5 jour	18/01/2022	18/01/2022	100%	500	Adrien Hupperts	4 h	
23	✓	➡	Schémas d'alimentation (idée)		22	0,75 jour	19/01/2022	19/01/2022	100%	500	Adrien Hupperts	6 h
24	✓	➡	Carte électronique		23	3 jours	21/01/2022	02/02/2022	100%	500	Adrien Hupperts	24 h
25	✓	➡	Installer et gérer les composants (capteurs)		24	2,25 jours	02/02/2022	01/03/2022	100%	500	Adrien Hupperts	18 h
26	✓	➡	Code		25	2,5 jours	02/03/2022	09/03/2022	100%	500	Adrien Hupperts	20 h
27	✓	➡	Carte alimentation		26	5,5 jours	11/03/2022	04/04/2022	100%	500	Adrien Hupperts	44 h
28	✓	➡	Carte électronique 2		27	1,25 jours	06/04/2022	26/04/2022	100%	500	Adrien Hupperts	10 h
29	📅	➡	Gérer l'alimentation		28	2,5 jours	27/04/2022	10/05/2022	50%	500	Adrien Hupperts	20 h
30	📅	➡	Mise en veille + capteurs		29	3,75 jours	11/05/2022	27/05/2022	0%	500	Adrien Hupperts	30 h

## IX- Problèmes rencontrés

Lors de la réalisation de ce projet, plusieurs problèmes ont été rencontrés. Des solutions ont été trouvées. Les voici :

### - Valeurs de température et humidité erronée.

Problème : Un certain temps après le lancement du programme, les valeurs se mélangeaient pour finalement donner des valeurs erronées.

Solution : Changement de capteur

Refonte de la carte

Changement dans le code

Changement des câbles par des câbles blindés

### - Détecteur de présence (infrarouge)

Problème : Le détecteur infrarouge ne se fait pas seulement par la présence, mais aussi par la lumière.

Solution : Remplacement de capteur par un capteur ultrason.

## X- Nomenclature

Références	Prix
CTN 10k	0.73€
LM35	2.60€
HIH-4030	22.39€
HC-SR04	3.90€
PCF8591	5.36€
TSR 3-24150	31.87€

## XI- Conclusion

Ce projet m'a permis d'apprendre à réaliser une prestation en équipe. Malgré des hauts et des bas au niveau technique, le projet m'a paru bénéfique dans le domaine scolaire et humain.

J'aurais aimé consacrer plus de temps scolaire à ce projet afin de l'améliorer et de le perfectionner. Quelques parties de mon contrat, comme la lumière, n'ont pas pu être bien travaillées.

Je remercie mes professeurs, Monsieur Dubois pour l'aide apportée lors de la réalisation de la carte et Monsieur Duchiron pour les conseils qu'il m'a donné lors de mes difficultés en ce qui concerne le programme ainsi que Madame Boyer pour certaine précision qu'elle m'a donné pour les capteurs. Je remercie également mes camarades Alexandre Cardona, Luca Grandon et Maxence Laporte pour m'avoir soutenu et poussé à avancer dans ce projet et pour la bonne ambiance du groupe.

## CONTRAT Étudiant 2 IR LAPORTE Maxence

### I - Objectifs :

Mon contrat porte, en premier lieu, sur la création d'une Interface Homme-Machine (IHM) qui permet d'afficher des widgets qui offrent des services à l'utilisateur.

En second, mon contrat doit créer plusieurs widgets qui offrent des services à l'utilisateur en utilisant des informations depuis internet et depuis le stockage local mais aussi depuis les capteurs (lien avec Contrat Étudiant 1 EC).

Enfin, mon contrat doit pouvoir charger un profil, selon si l'utilisateur est reconnu ou non, depuis les informations stockées sur une base de données (lien avec Contrat Étudiant 4 IR). Mon contrat doit aussi permettre à l'utilisateur d'interagir avec le miroir afin d'accéder à des réponses audios via des commandes vocales (lien avec Contrat Étudiant 3 IR) mais également le contrôler pour pouvoir écouter de la musique et bien plus.

Mon contrat étant assez fourni, il contient à lui seul 9 fichiers de type ".py", 10 fichiers de type ".mp3" ainsi que 19 fichiers de type ".png". Il comporte en tout 9 classes.

### II - Ressources logicielles, matérielles et diagramme de Gantt:

Afin de pouvoir réaliser mon contrat, j'ai eu recours aux logiciels suivants :

- Pycharm

J'ai utilisé Pycharm afin de réaliser tous les fichiers de type ".py" (voir Choix Technologique).

- Qt Designer

J'ai utilisé Qt Designer afin de réaliser l'interface graphique (voir Choix Technologique).

- MindView

J'ai utilisé MindView afin de réaliser mon diagramme de Gantt afin de prévoir la répartition de mon temps sur les différentes parties de mon contrat.

- Google Chrome

J'ai utilisé Google Chrome afin d'accéder à de nombreuses documentations

- Paint

J'ai utilisé Paint afin de créer des schémas pour mon Interface Graphique

J'ai aussi eu recours aux matériels suivant :

- Ordinateur fixe windows

J'ai utilisé cet ordinateur, situé dans la salle de cours, pour faire les premiers tests, mon diagramme de Gantt et commencer mon contrat mais également pour des raisons d'ergonomie.

- Ordinateur portable personnel windows

J'ai utilisé cet ordinateur pour des raisons de praticité afin de pouvoir travailler sans avoir accès à la salle de cours.

- Raspberry pi 3 Linux

J'ai utilisé cette carte afin de faire des tests via une base de données fictives pour charger le profil de l'utilisateur.

- Raspberry pi 4 Linux

J'ai utilisé cette carte afin de rendre une version définitive de Masha, le miroir connecté

### Diagramme de Gantt

	I	Plan...	Nom de tâche	Préde...	Durée	Début	Fin	Progr...	Priorité	Ressources	Travail	Coût	
1		➡	Miroir connecté		31,5 jours?	05/01/2022	17/06/2022	80%	500		804,33 h	0,00 €	
2	✓	➡	▫ Préparation début de projet/1er Revue		2,75 jours	05/01/2022	19/01/2022	100%	500		28,33 h	0,00 €	
8	✓	➡	Diagramme UML		0,5 jour	14/01/2022	14/01/2022	100%	500	Maxence Laporte	4 h	0,00 €	
19		➡	▫ Contrat 2.4		29,75 jours	18/01/2022	17/06/2022	82%	500		754 h	0,00 €	
31		➡	▫ Contrat 2.4.2 (Laporte Maxence)		20,75 jours	18/01/2022	24/05/2022	94%	500		166 h	0,00 €	
32		➡	▫ Créeer l'IHM (Interface homme-machine)		20,75 jours	18/01/2022	24/05/2022	94%	500		166 h	0,00 €	
33	✓	➡	Recherches liées au film sans teint		0,5 jour	18/01/2022	18/01/2022	100%	500	Maxence Laporte	4 h	0,00 €	
34	✓	➡	Apprendre QT Python		33	1,75 jours	19/01/2022	25/01/2022	100%	500	Maxence Laporte	14 h	0,00 €
35	✓	➡	Récupérer le site météo (Widget)		34	0,75 jour	26/01/2022	26/01/2022	100%	500	Maxence Laporte	6 h	0,00 €
36	✓	➡	Afficher la date / l'heure / la météo (Widget)		35	2,25 jours	28/01/2022	22/02/2022	100%	500	Maxence Laporte	18 h	0,00 €
37	✓	➡	Récupérer le/les site/s de radio (Widget)		36	0,75 jour	23/02/2022	23/02/2022	100%	500	Maxence Laporte	6 h	0,00 €
38	✓	➡	Récupérer des musiques via USB		37	1,75 jours	01/03/2022	04/03/2022	100%	500	Maxence Laporte	14 h	0,00 €
39	✓	➡	Réparation musiques		38	0,5 jour	08/03/2022	08/03/2022	100%	500	Maxence Laporte	4 h	0,00 €
40	✓	➡	Faire les threads (QtPython)		39	0,75 jour	09/03/2022	09/03/2022	100%	500	Maxence Laporte	6 h	0,00 €
41	✓	➡	Fusion IHM/recof		40	0,5 jour	11/03/2022	11/03/2022	100%	500	Maxence Laporte	4 h	0,00 €
42	✓	➡	Fusion IHM/Recof 2		41	0,5 jour	15/03/2022	15/03/2022	100%	500	Maxence Laporte	4 h	0,00 €
43	✓	➡	Widget Mode Aveugle		42	5,75 jours	16/03/2022	26/04/2022	100%	500	Maxence Laporte	46 h	0,00 €
44	✓	➡	Exporter le projet sous Linux (portabilité raspberry)		43	1,25 jours	27/04/2022	03/05/2022	100%	500	Maxence Laporte	10 h	0,00 €
45	✓	➡	Fusion totalité programme (Recof/Ihm/Recof)		44	0,75 jour	04/05/2022	04/05/2022	100%	500	Maxence Laporte	6 h	0,00 €
46	✓	➡	Refaire IHM (supprimer les boutons)		45	0,5 jour	10/05/2022	10/05/2022	100%	500	Maxence Laporte	4 h	0,00 €
47	✓	➡	Charger un profil (BDD et Interface Admin Contrat IR 2)		46	0,75 jour	11/05/2022	11/05/2022	100%	500	Maxence Laporte	6 h	0,00 €
48	✓	➡	Appliquer les commandes vocales (Widget / Lumière)		47	0,5 jour	17/05/2022	17/05/2022	100%	500	Maxence Laporte	4 h	0,00 €
49		➡	Récupération des informations capteurs ambient (Contrat EC)		48	0,75 jour	18/05/2022	18/05/2022	0%	500	Maxence Laporte	6 h	0,00 €
50		➡	Faire en sorte que l'exécutable s'exécute au lancement de la raspberry		49	0,5 jour	24/05/2022	24/05/2022	0%	500	Maxence Laporte	4 h	0,00 €

### III - Choix Technologiques :

Afin de réaliser mon contrat, j'ai dû choisir parmi plusieurs langages de programmation. Pour n'en citer que deux, j'avais le choix entre le C++ et le Python. Pour trancher entre tous ces langages, j'ai comparé leurs avantages et leurs inconvénients.

Le langage de programmation que nous avons finalement choisi est Python car ce langage est facile à apprendre, comprendre et à coder.

Notre projet va être porté vers une carte Raspberry pi 4 sous Linux. Le python nous permet de réaliser un portage d'un OS à l'autre sans problèmes (voir Ressources logicielles et matérielles).

Enfin, c'est un langage très documenté sur internet, ce qui m'a permis de régler plus rapidement quelques erreurs que j'ai eues.

Pour conclure sur ce choix, grâce au langage Python, nous avons été plus productif que si nous avions choisi un autre langage de programmation.

J'ai dû également choisir entre Qt et tkinter pour créer l'Interface Homme-Machine.

Qt possède son propre logiciel graphique qui permet de créer l'Interface en voyant ce que l'on fait, ce qui est plus pratique et rapide. Lorsque l'on enregistre l'Interface, elle est de type ".ui", il suffit alors de convertir ce fichier en ".py" grâce à Pycharm.

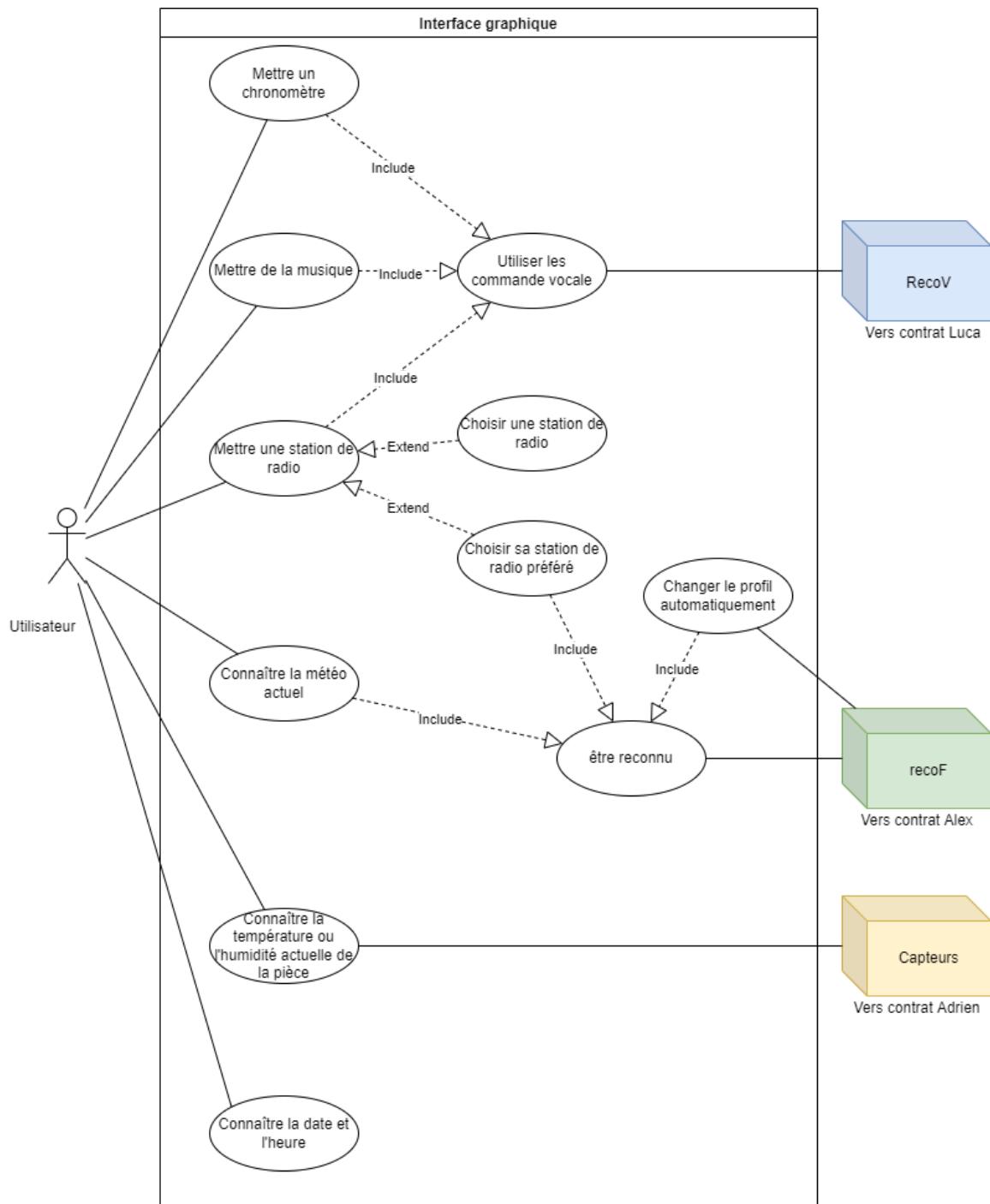
Qt prend également en charge tout type d'OS. Il est aussi beaucoup documenté sur internet car il est souvent utilisé avec Python.

Contrairement à Qt, Tkinter est moins fiable et on doit écrire un code pour placer les labels à l'aveugle.

J'ai donc choisi Qt afin de gagner du temps sur le développement de mon contrat

## IV - Diagramme des cas d'utilisation :

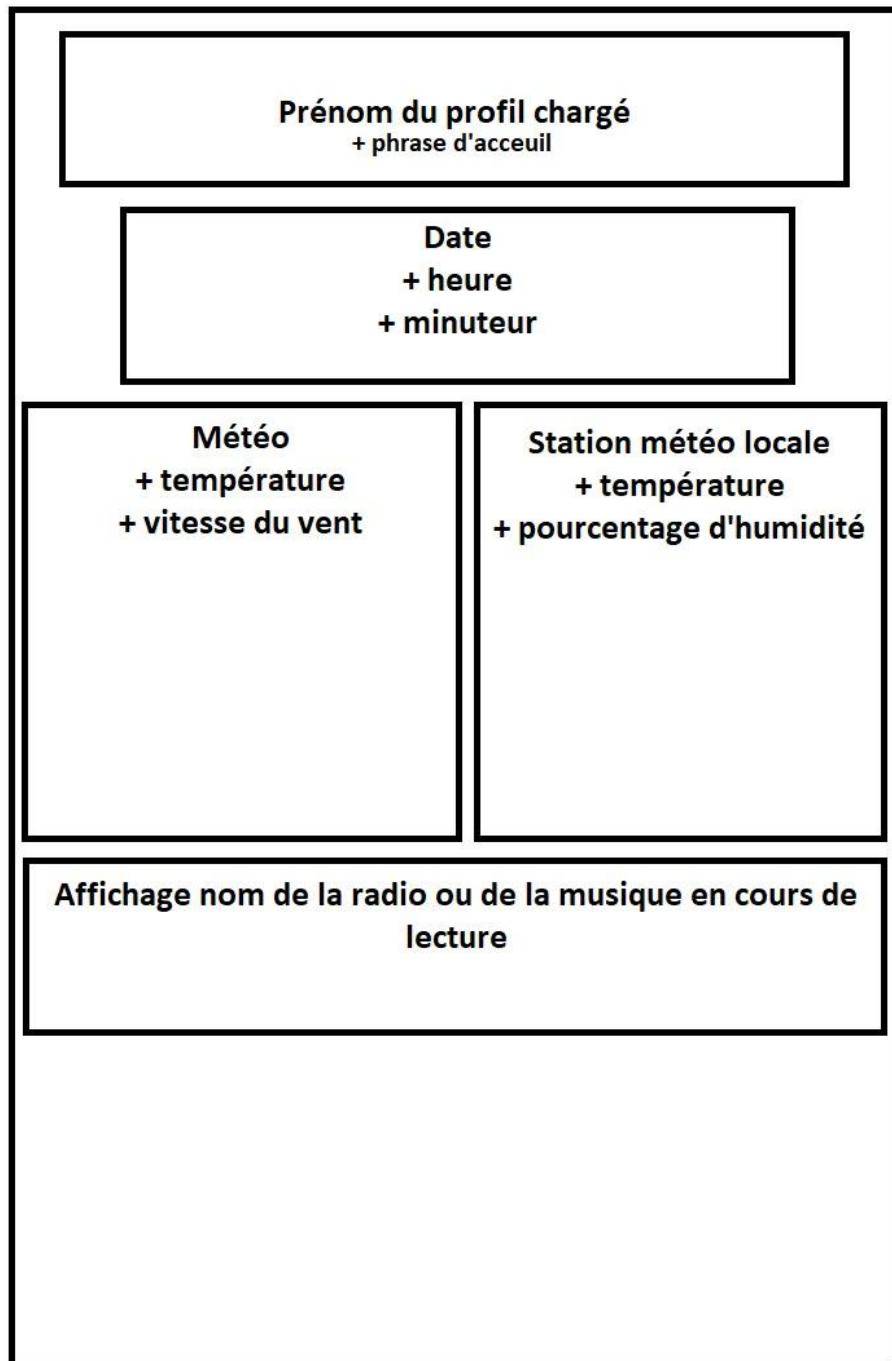
Afin de créer mon Interface Homme-Machine, je me suis mis dans la peau de l'utilisateur en exprimant mes besoins vis à vis d'un miroir connecté. Puis, j'ai créé un diagramme des cas d'utilisation qui retranscrit mes besoins.



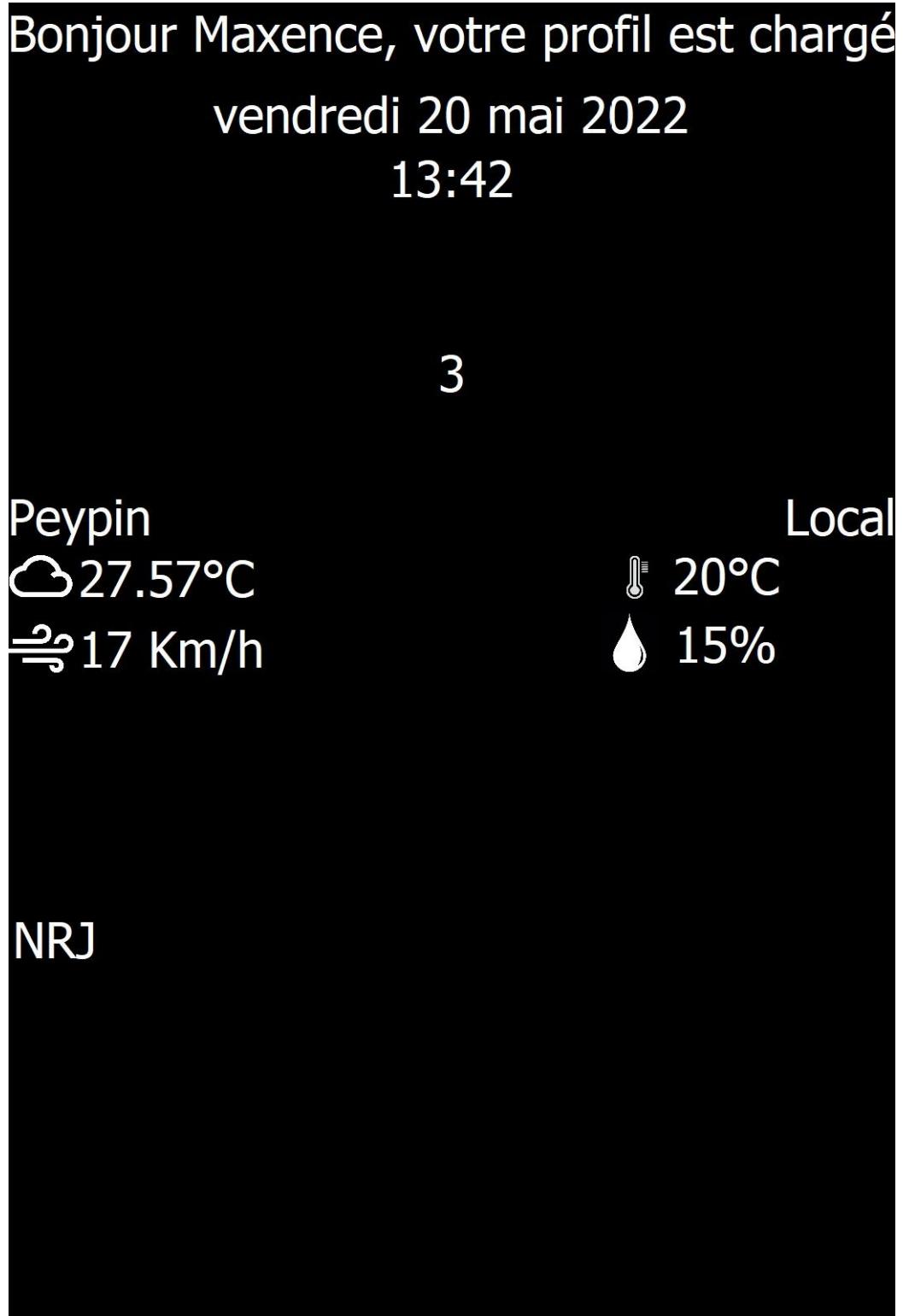
## V - Interface Graphique :

Pour créer mon Interface Graphique, j'ai regardé mon diagramme des cas d'utilisations (Voir diagramme des cas d'utilisation) afin de déterminer les éléments qui doivent être affichés à l'utilisateur.

Suite à ça, j'ai créé un schéma de ce à quoi devrait ressembler mon Interface.

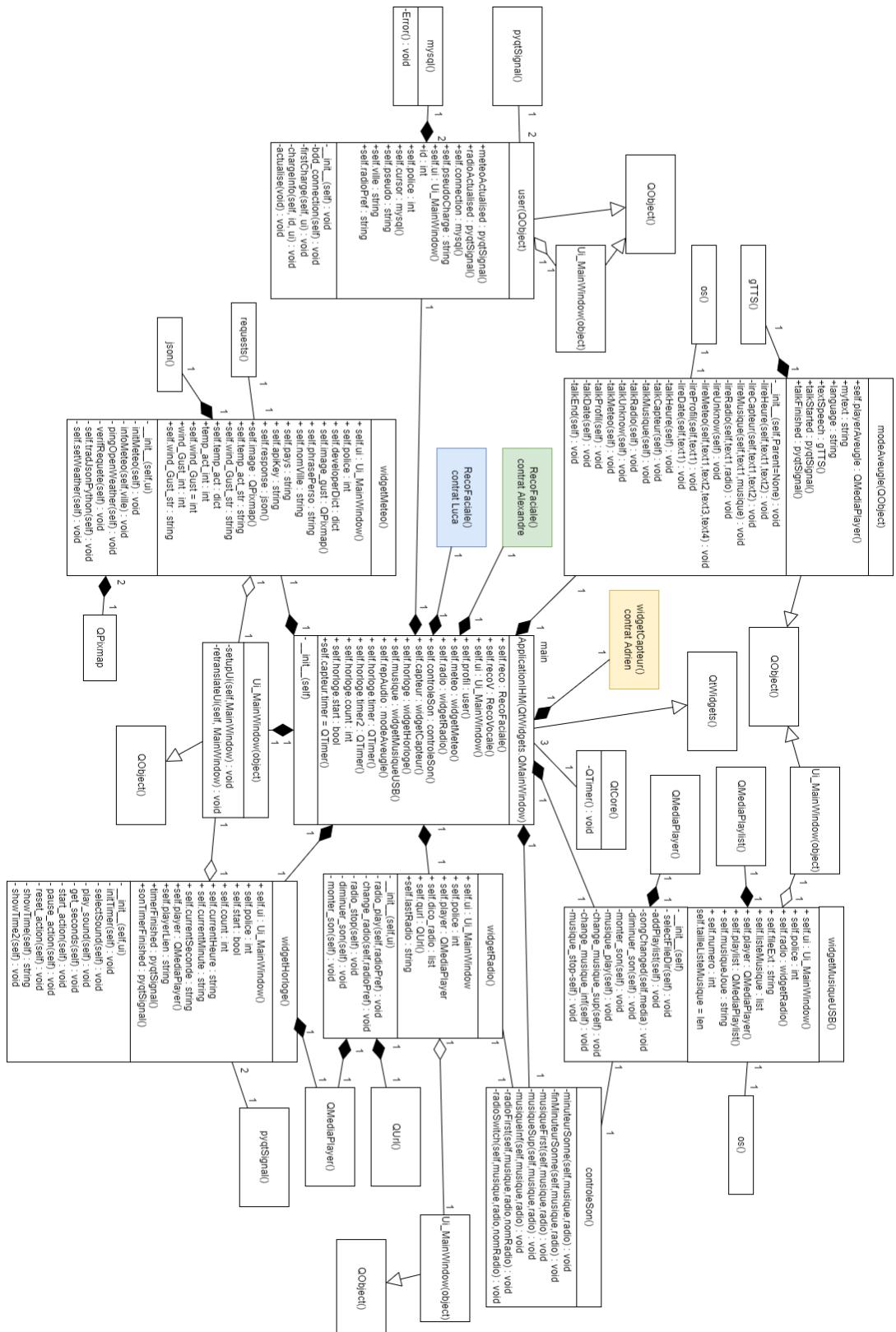


Ensuite, à l'aide de Qt Designer j'ai pu créer l'interface grâce à de nombreux Labels. En sachant que l'effet Miroir de Masha est réalisé par un film sans teint, j'ai mis le fond de l'interface en noir afin de faire ressortir les informations.



## VI - Diagramme de Classe :

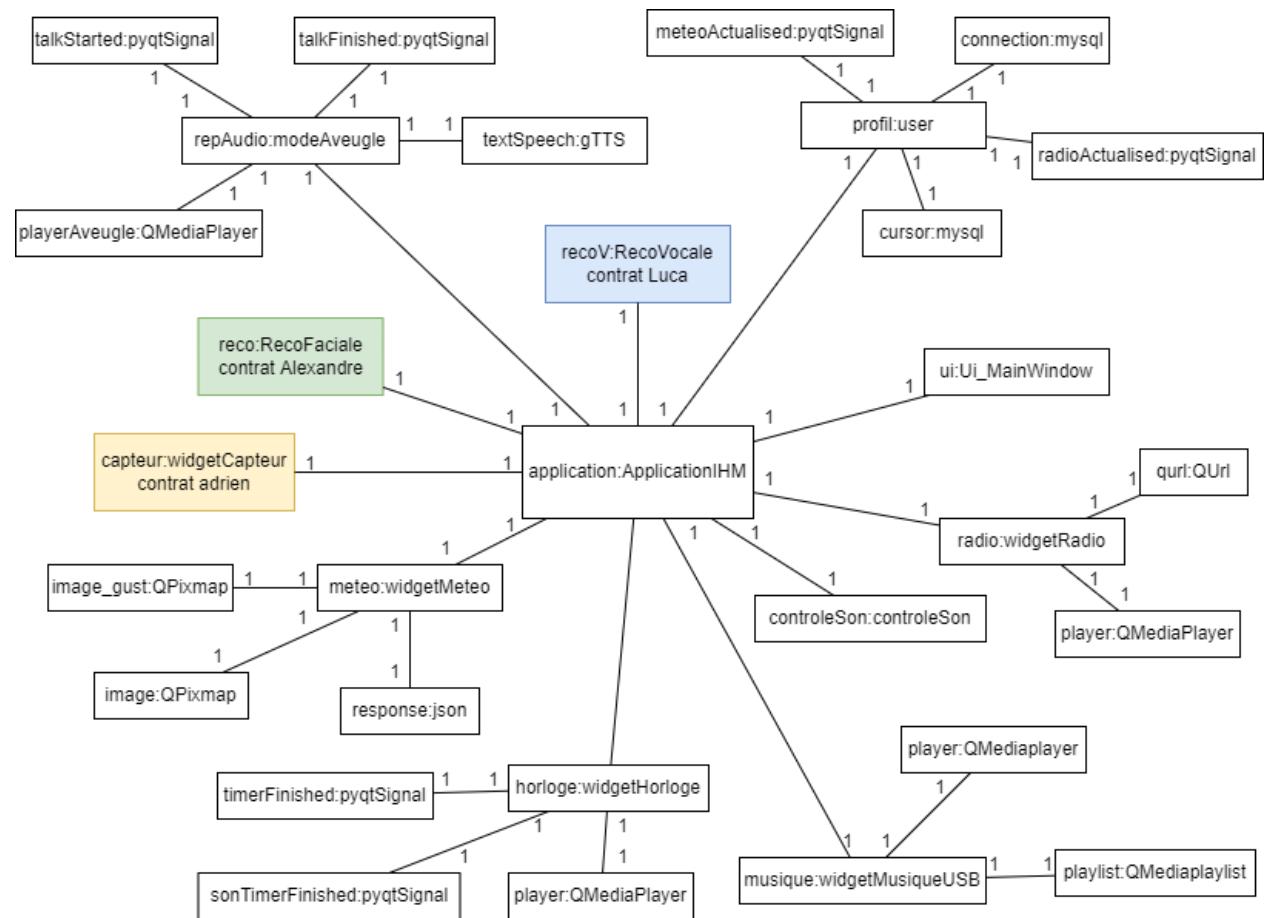
Après avoir réaliser l'ensemble de mes classes, j'ai créé un diagramme de Classe



Pour récapituler, tout part de la classe ApplicationIHM, qui hérite de QtWidgets, qui instancie toutes les autres classes par un lien de composition. La classe Ui\_MainWindow est liée à toutes les classes qui ont besoin d'afficher des informations par un lien d'agrégation.

## VII - Diagramme d'objet :

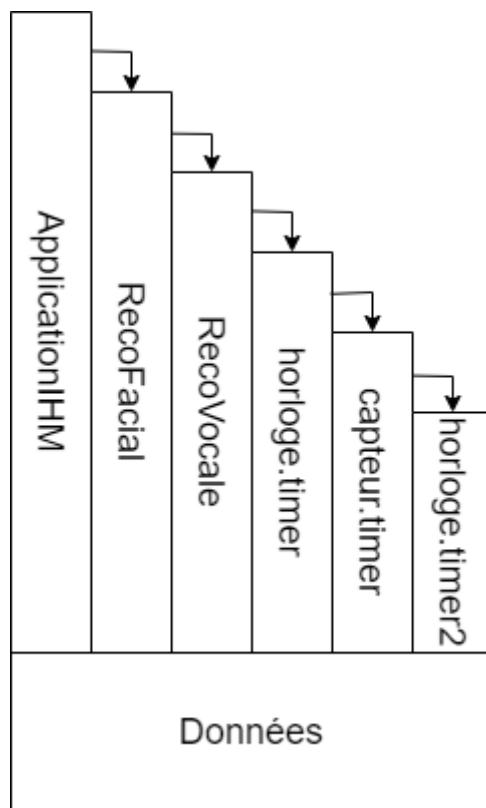
à l'aide du diagramme de classe (voir Diagramme de Classe), j'ai créé un diagramme d'objet.



## VIII - Organisation de Thread, diagramme de séquence et Explication de code :

### a - Organisation des threads :

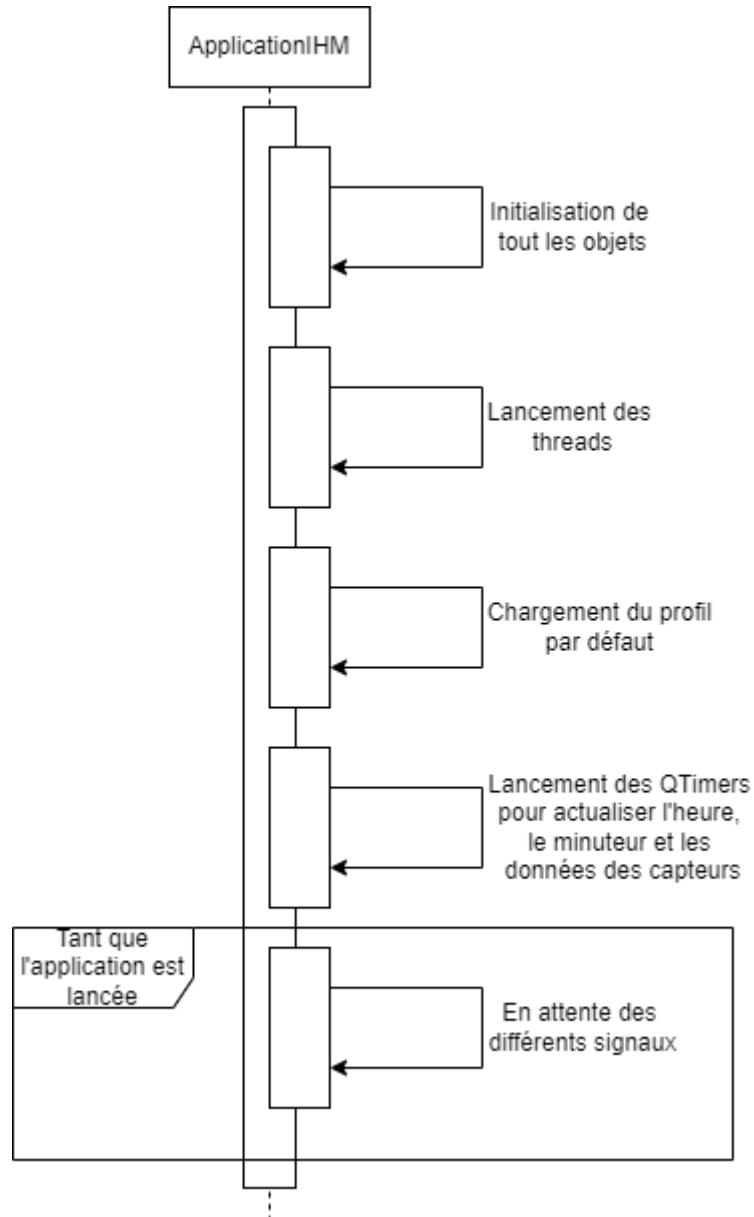
Sur le diagramme suivant, nous pouvons voir le lancement successif des threads :



Nous pouvons voir que lorsque l'on lance le programme, la classe ApplicationIHM se lance en premier. Elle lance à son tour, successivement, le Thread de la classe RecoFacial, puis celui de la classe RecoVocale, puis on lance le thread horloge.timer, capteur.timer et enfin horloge.timer2 qui sont de type QTimer.

b - Diagramme de séquence de la classe ApplicationIHM :

Pour commencer, j'ai réalisé un diagramme de séquence pour ma classe ApplicationIHM, qui est un peu la classe qui lie tous les programmes ensemble.



### Explication du code pour la classe ApplicationIHM :

Pour cette classe, je vais expliquer comment je lance mes QTimers. Ils servent à actualiser les éléments après une durée prédéfinie.

Voici une capture d'écran des lignes qui permettent de faire cette actualisation.

```
self.horloge.timer = QTimer()
self.horloge.timer.timeout.connect(self.horloge.initTimer)
self.horloge.timer.start(1000)
self.capteur.timer = QTimer()
self.capteur.timer.timeout.connect(self.capteur.receiveValue)
self.capteur.timer.start(10000)
self.horloge.timer2 = QTimer()
self.horloge.timer2.timeout.connect(self.horloge.showtime2)
self.horloge.timer2.start(1000)
```

La première ligne permet d'instancier un objet "timer" de type QTimer pour l'objet horloge qui est lui-même instancier plus haut dans l'algorithme.

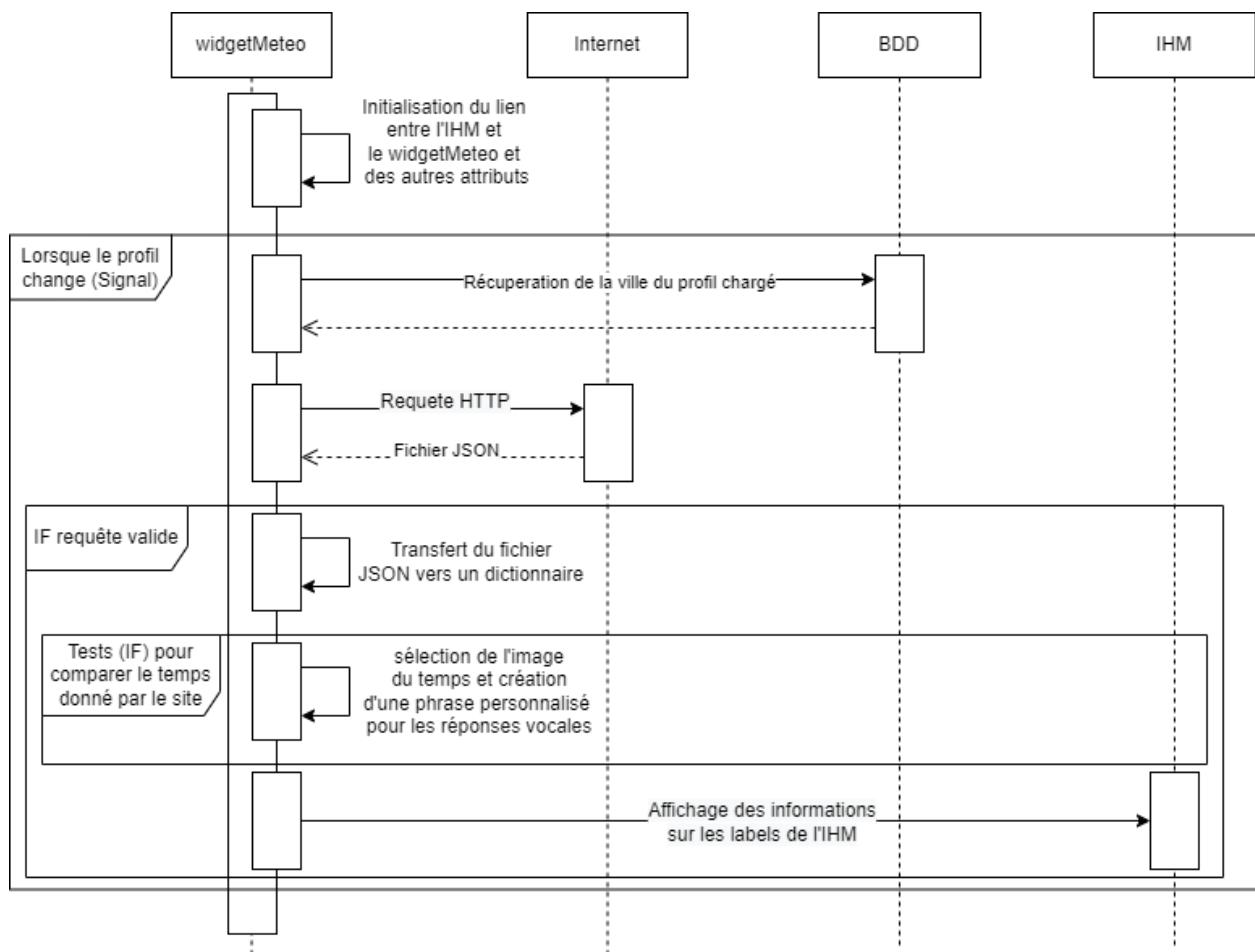
Ensuite, la deuxième ligne indique à l'objet "timer" qu'il doit activer la méthode "initTimer" de la classe "widgetHorloge" à chaque fois que le minuteur atteint la fin du décompte. Cette méthode me permet d'afficher l'heure sur l'Interface Graphique.

Enfin, la troisième ligne permet de fixer la valeur du minuteur afin d'appeler la méthode à un intervalle régulier en milliseconde.

Je répète alors ces trois même ligne afin de créer d'autres minuteurs pour d'autres méthodes tel que "receiveValue", une méthode de la classe "widgetCapteur" qui permet à Adrien, l'étudiant EC, de récupérer la valeur des capteurs toutes les 10 secondes. J'utilise également un minuteur pour la méthode "showtime2" de la classe "widgetHorloge" qui me permet d'actualiser la valeur d'un minuteur programmé sur l'IHM par l'utilisateur si ce dernier en à demandé un.

### c - Diagramme de séquence de la classe widgetMeteo :

Ensuite, j'ai réalisé un diagramme de séquence pour expliquer ma classe widgetMeteo, elle me permet d'afficher la météo lorsque l'utilisateur est reconnu. L'essentiel de cette classe est dû à la clef API qui est donnée par le site météo. Elle permet



### Explication du code pour la classe widgetMeteo :

Ici, je vais expliquer comment j'ai créé ma requête HTTP pour récupérer les informations du site météo pour pouvoir, ensuite, les afficher à l'utilisateur.

Voici une capture d'écran des lignes qui permettent de faire cette requête.

```
def infoMeteo(self, ville):
    self.nomVille = ville
    self.pingOpenWeather()

def pingOpenWeather(self):
    """Requête HTTP"""
    self.pays = 'fr'
    self.apiKey = '39f8c6363789395683394a6fe29ba54f'
    self.response = requests.get(f'https://api.openweathermap.org/data/2.5/weather?q={self.nomVille},{self.pays}&units=metric&appid={self.apiKey}')
```

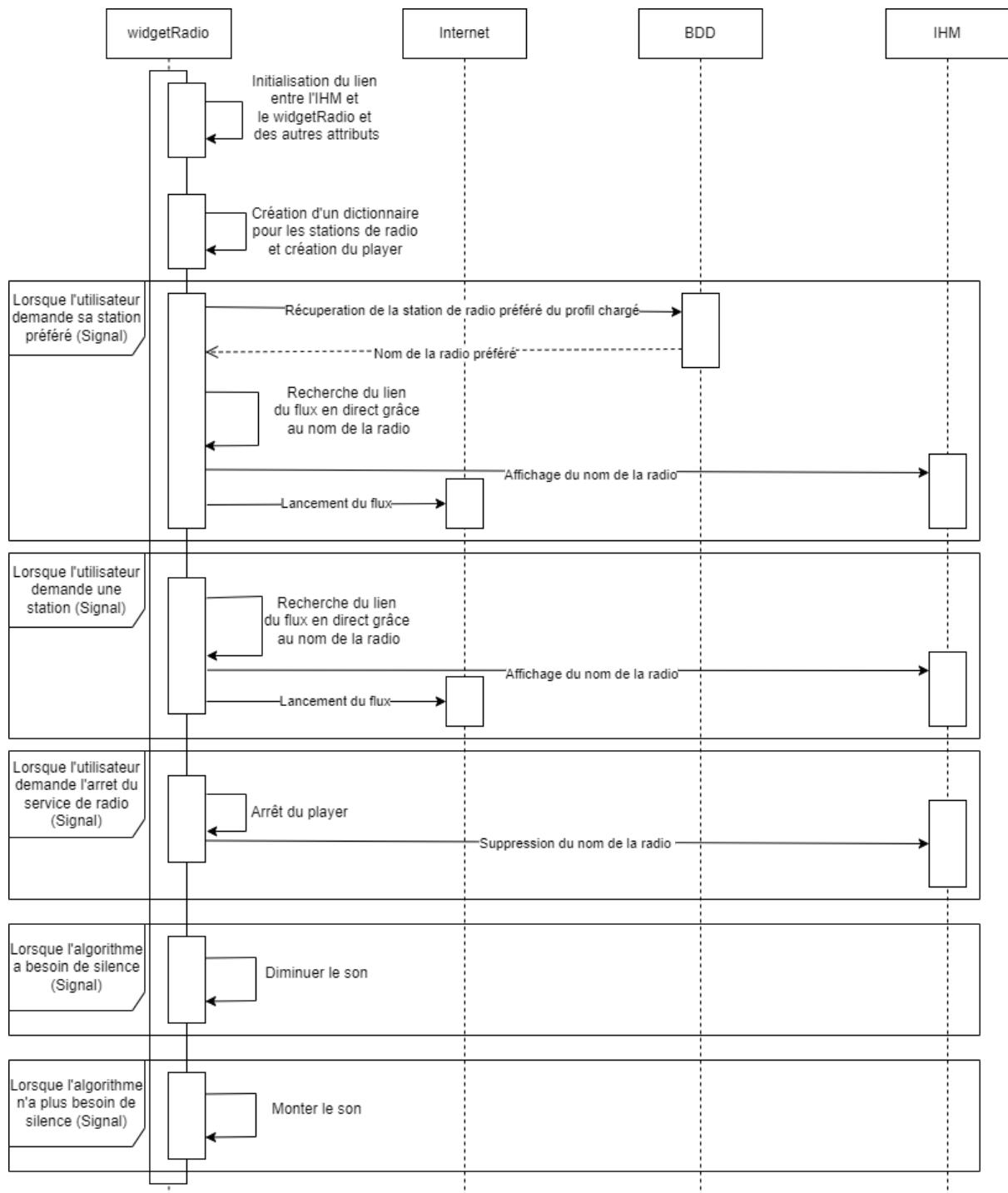
La première méthode me permet de récupérer le nom de la ville lié à l'utilisateur reconnu via la base de données. Elle met le nom de la ville dans l'attribut "nomVille".

Ensuite, dans la deuxième méthode, j'ai rempli l'attribut "pays" par "fr" ce qui correspond à la France et l'attribut "apiKey" qui est la clef API donnée par le site météo lors de l'inscription.

Enfin, grâce à la classe requests, j'utilise la méthode "get" pour envoyer au site une requête avec les paramètres stockés dans les attributs "nomVille", "pays" et "apiKey". J'envoi également le paramètre "&unit=metric" afin que le site me donne des valeurs de type metric.

#### d - Diagramme de séquence de la classe widgetRadio :

J'ai également réalisé un diagramme de séquence pour expliquer ma classe widgetRadio, elle permet à l'utilisateur de demander sa station de radio préférée mais aussi une station de radio qu'il veut écouter. Cette classe est composée de beaucoup de signaux, ils permettent de faire le lien entre mon contrat et celui de Luca (Etudiant 3 IR) mais ils servent aussi à baisser le son lorsque l'alarme du minuteur sonne puis remonte automatiquement lorsque le son du minuteur est terminé.



### Explication du code pour la classe widgetRadio :

Pour cette classe, je vais expliquer comment fonctionne le lancement de la station de radio préféré lorsque l'utilisateur la demande.

Voici une capture d'écran des lignes qui permettent de faire cette action.

```
def radio_play(self, radioPref):
    self.ui.val_radio_musique.setText("<p align=\"center\">" + f"<span style=\" font-size:{self.police}pt; \">" + "<font color = \"#ffffff\">" + radioPref)
    self.lastRadio = radioPref
    self.qurl = QUrl(self.dico_radio[radioPref])
    self.player.setMedia(QMediaContent(self.qurl))
    self.player.play()
```

La première ligne de ma méthode me permet d'écrire le nom de la radio préféré qui est récupéré dans la BDD (Voir diagramme de séquence de la classe User)

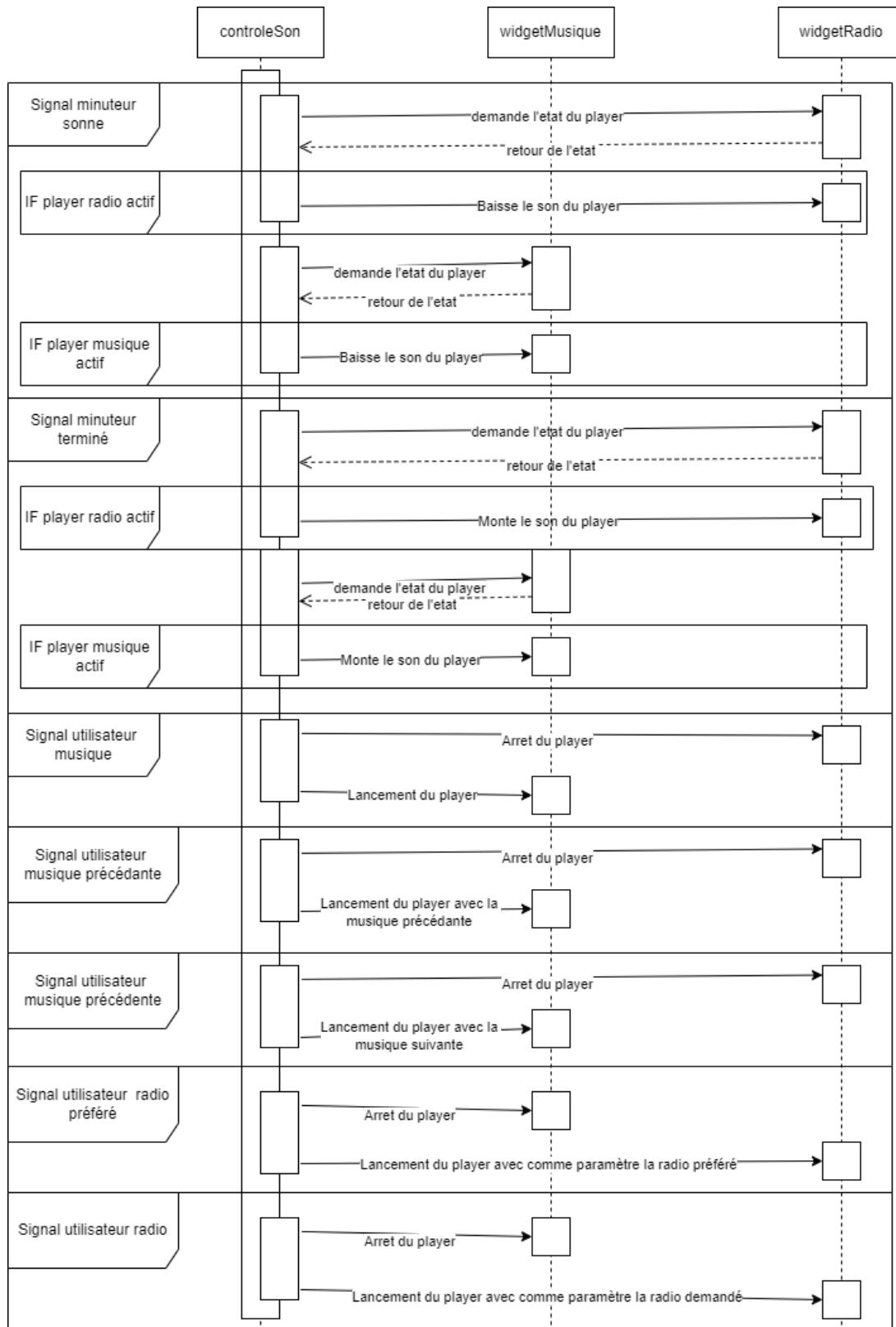
Ensuite, la deuxième ligne de la méthode permet de garder en mémoire la dernière station radio écoutée.

La troisième ligne permet de récupérer l'url de la radio préférée depuis le dictionnaire de la classe.

Enfin, les deux dernières lignes permettent, successivement, de mettre l'url dans le player et de lancer le flux en direct en activant le player.

### e - Diagramme de séquence de la classe controleSon :

J'ai réalisé un diagramme de séquence pour expliquer ma classe contoleSon, elle permet au miroir de, comme son nom l'indique, de contrôler le son et l'ordre des players afin qu'ils ne se chevauchent pas.



### Explication du code pour la classe controleSon :

Pour cette classe, je vais expliquer comment fonctionne l'anti-chevauchement.

Voici quelques captures d'écran des lignes qui permettent de faire cette action.

```
def musiqueFirst(self,musique,radio):
    radio.radio_stop()
    musique.musique_play()
```

La première ligne permet d'arrêter le player de la radio et la seconde permet de lancer la méthode qui permet à son tour de lancer le player de la musique.

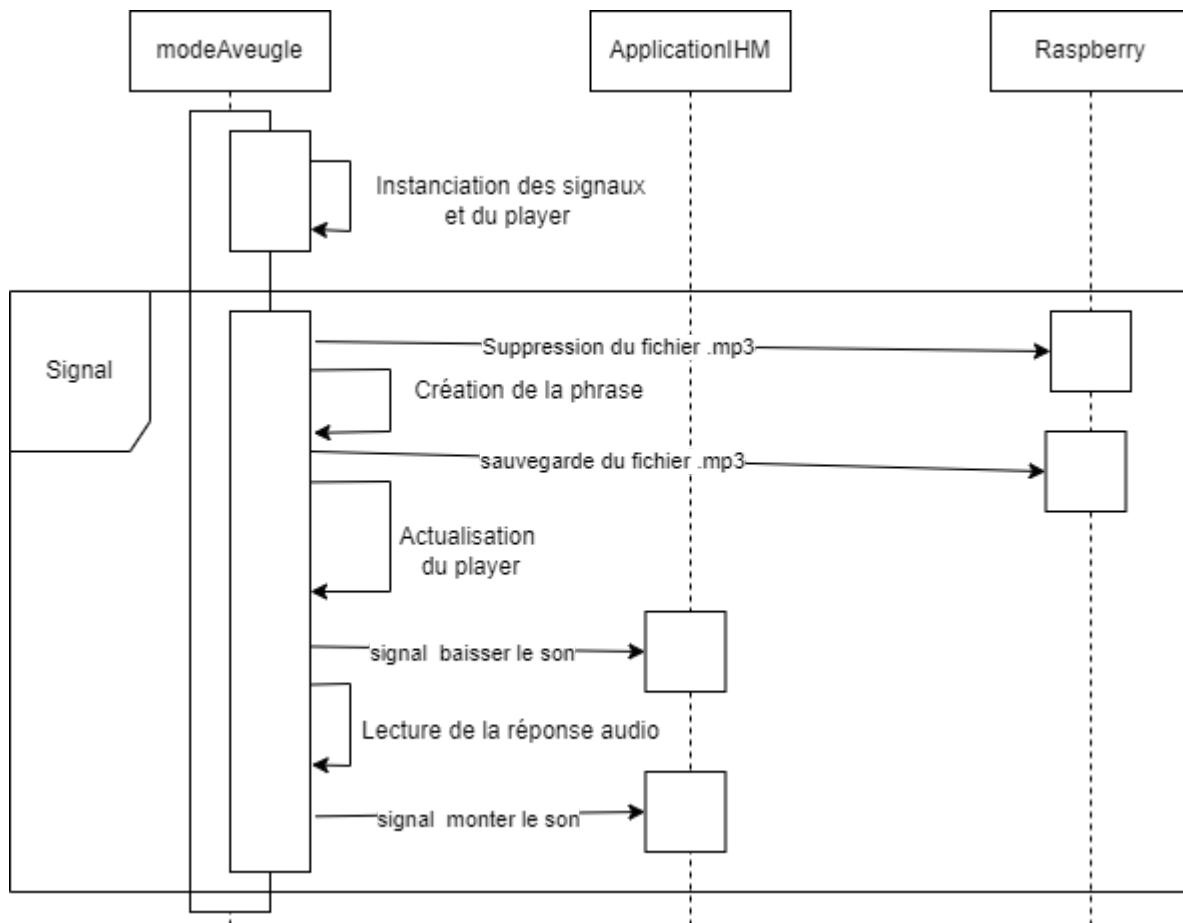
```
def radioFirst(self,musique,radio,radioPref):
    musique.musique_stop()
    radio.radio_play(radioPref)
```

La première ligne permet d'arrêter le player de la musique et la seconde permet de lancer la méthode, avec comme paramètre la radio préféré de l'utilisateur, qui permet à son tour de lancer le player de la radio.

Grâce à ses deux méthodes, l'utilisateur ne sera pas dérangé par le mélange d'une musique et d'une radio, ce qui rendrait l'écoute impossible.

f - Diagramme de séquence de la classe modeAveugle :

J'ai réalisé un diagramme de séquence pour expliquer ma classe modeAveugle, elle permet au miroir de donner une réponse audio à l'utilisateur à la suite d'une demande par commande vocale.



### Explication du code pour la classe modeAveugle :

Pour cette classe, je vais expliquer comment fonctionne l'actualisation ou bien le rafraîchissement du player afin que l'utilisateur ait une réponse correcte et actuelle.

Voici une capture d'écran des lignes qui permettent de faire cette action.

```
def talkHeure(self):
    self.playerAveugle.setMedia(QMediaContent(QUrl.fromLocalFile("/home/pi/Documents/Raspberry/refresh.mp3")))
    self.playerAveugle.setMedia(QMediaContent(QUrl.fromLocalFile("/home/pi/Documents/Raspberry/heure.mp3")))
```

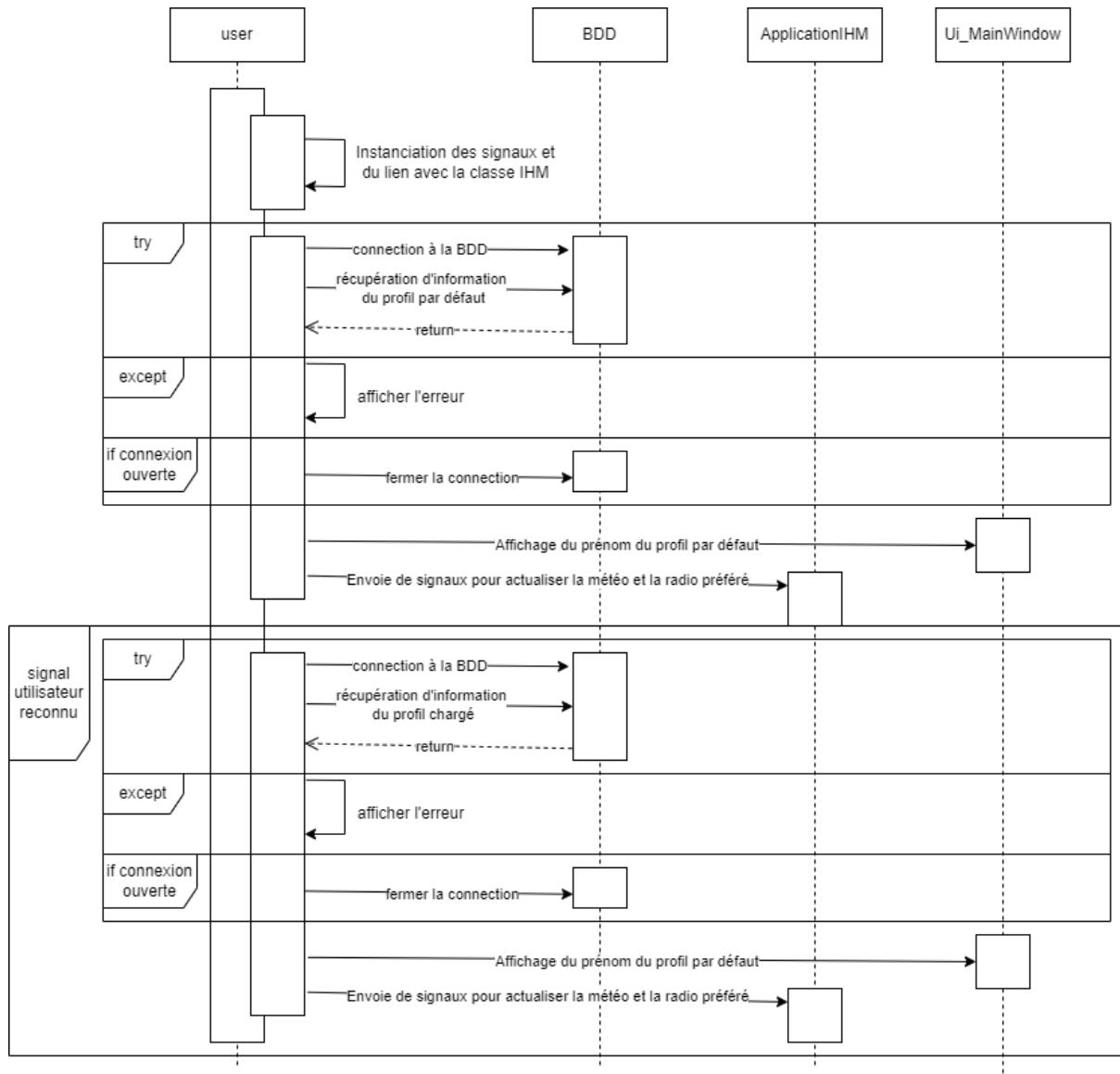
La première ligne de cette méthode permet d'actualiser le player en sélectionnant un autre fichier de type .mp3.

La seconde permet alors l'écoute du bon fichier .mp3, ici c'est le fichier heure.mp3.

Sans ce rafraîchissement, si l'utilisateur demande 2 fois l'heure avec une intervalle de 5 min, Masha répondrait deux fois la même chose. Par exemple, pour la première réponse, elle donnerait cette réponse "Il est 16 heures 50" et lorsque l'utilisateur le lui redemande avec 5 minutes d'intervalles, au lieu de dire "Il est 16 heures 55", elle répètera "Il est 16 heures 50".

### g - Diagramme de séquence de la classe user:

J'ai réalisé un diagramme de séquence pour expliquer ma classe user, elle permet au miroir de charger les différentes informations du profil chargé.



### Explication du code pour la classe user:

Ici, nous allons voir comment je récupère les différentes informations du profil chargé  
Voici une capture d'écran des lignes qui permettent de récupérer ces informations.

```
self.connection = mysql.connector.connect(host='172.20.10.4', database='projetMiroir', user='projet', password='miroir!')
```

Comme l'indique la capture d'écran, je récupère les informations de l'utilisateur reconnu dans une base de données. J'utilise donc le langage sql.

Cette ligne me permet de me connecter à la base de données. On peut voir que la base de données à pour nom "projetMiroir".

Pour récupérer les informations j'utilise les requêtes sql suivantes :

```
SELECT ville FROM dummy WHERE id_image = {id};  
SELECT prenom FROM dummy WHERE id_image = {id};  
SELECT radioP FROM dummy WHERE id_image = {id};
```

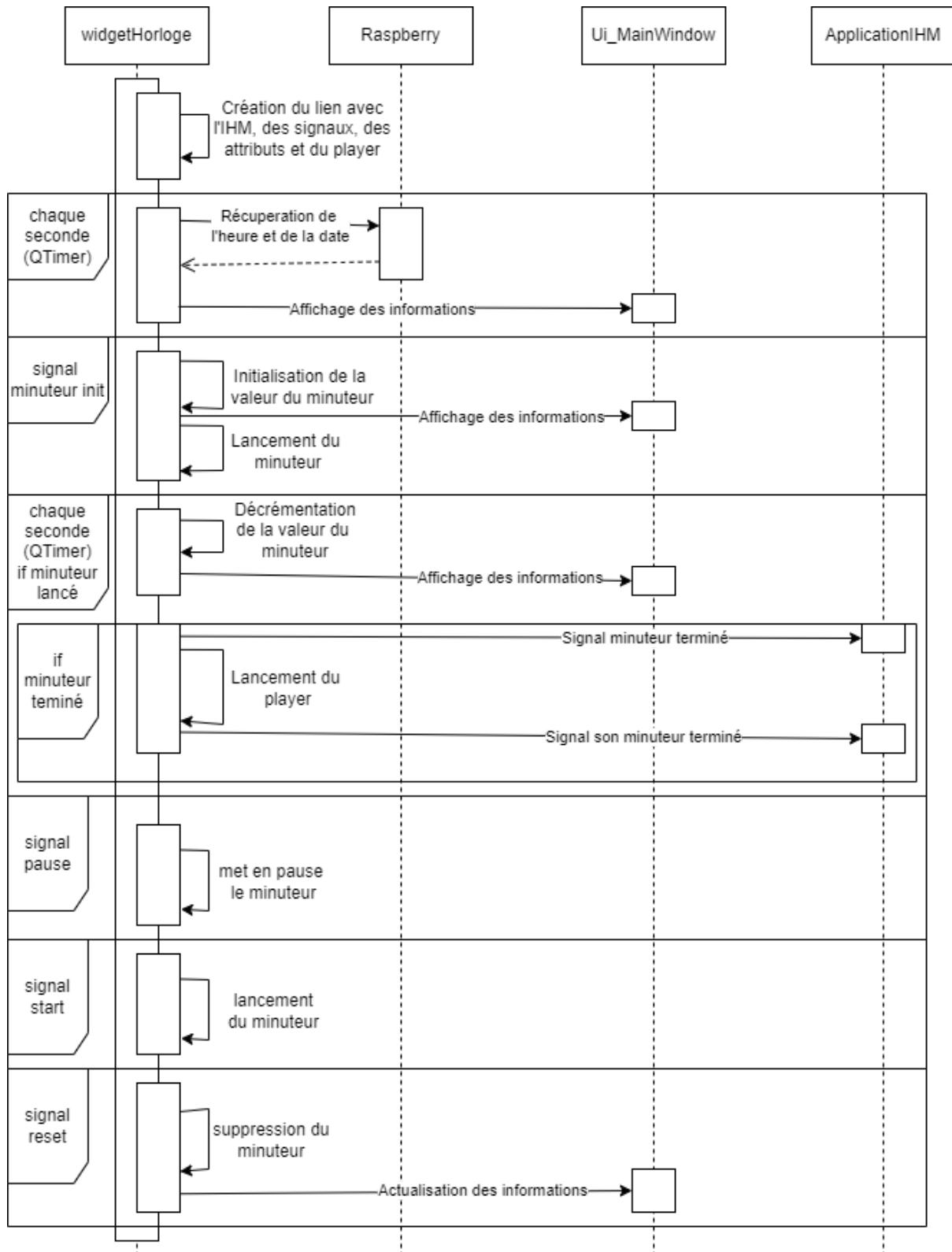
La première requête permet de récupérer le nom de la ville renseigné par l'utilisateur lors de son inscription via l'interface administrateur (voir Contrat étudiant 3 IR), cette information se trouve dans la table dummy et on tri les informations par rapport à l' "id\_image" qui correspond à l'id envoyé par la reconnaissance faciale (voir Contrat étudiant 4 IR).

Pour la deuxième requête permet de récupérer le prénom renseigné lors de l'inscription par l'utilisateur.

Enfin, la dernière requête permet de récupérer sa station de radio préférée.

### h- Diagramme de séquence de la classe widgetHorloge:

J'ai réalisé un diagramme de séquence pour expliquer ma classe widgetHorloge, elle permet au miroir d'afficher la date, l'heure sur l'interface Homme-Machine. Elle permet également à l'utilisateur de demander un minuteur.



### Explication du code pour la classe widgetHorloge:

Ici, nous allons voir comment je récupère les différentes informations tel que l'heure, les minutes mais aussi la date.

```
def showTime(self):
    current_time = QTime.currentTime()
    self.currentHeure = current_time.toString('hh')
    self.currentMinute = current_time.toString('mm')
    self.currentSeconde = current_time.toString('ss')
    now = QDate.currentDate()
    self.currentDate = now.toString(Qt.DefaultLocaleLongDate)
    return current_time.toString('hh:mm')
```

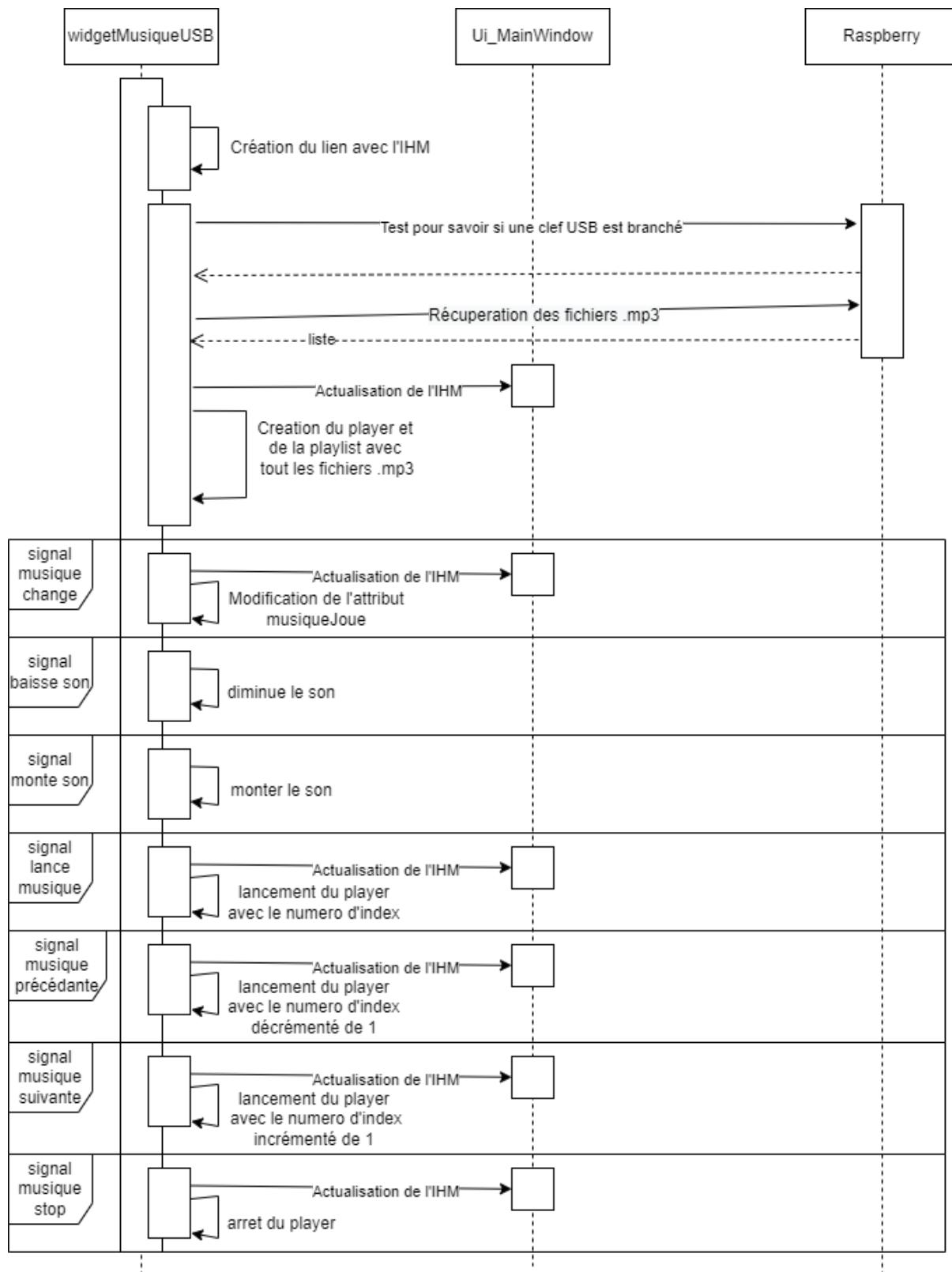
Dans cette méthode, "current\_time" est l'attribut dans lequel l'heure au format "heure:minute:seconde" est stockée. Suite à ça, je peux séparer les heures, minutes et secondes dans les attributs suivants: currentHeure, currentMinute et currentSeconde.

Dans un second temps, dans l'attribut "now", c'est la date qui est stockée. Ensuite, j'utilise l'attribut "currentDate" afin de stocker la date en version longue tel que "jour:numéro Du Jour:mois:année".

Enfin, je retourne l'heure actuelle sous format heure:minute afin de l'afficher sur l'IHM.

### i - Diagramme de séquence de la classe widgetMusiqueUSB:

J'ai réalisé un diagramme de séquence pour expliquer ma classe widgetMusiqueUSB, elle permet au miroir de proposer à l'utilisateur de jouer de la musique via une clef usb ou bien depuis la mémoire interne de la raspberry.



### Explication du code pour la classe widgetMusiqueUSB:

Ici, nous allons voir comment je crée la playlist. Grâce à cette dernière, les musiques vont pouvoir se lancer les une après les autres sans action de la part de l'utilisateur.

```
def addPlaylist(self):
    self.numero = 0
    self.tailleListeMusique = len(self.listeMusique)
    while self.numero != self.tailleListeMusique:
        self.playlist.addMedia(QMediaContent(QUrl.fromLocalFile(self.fileDir + "/" + self.listeMusique[self.numero])))
        self.numero = self.numero + 1
    self.numero = 1
```

Cette méthode permet de créer une playlist avec l'ensemble des fichiers de type .mp3 présent dans la carte ou bien dans la clef usb.

Je récupère la taille de la liste dans laquelle se trouve tous les fichiers .mp3 afin de pouvoir faire une boucle "while" qui entrera, dans la playlist, chaque lien vers le fichier .mp3 tant qu'il restera des fichiers non inclue dans la playlist.

## IX - Conclusion personnel et remerciement

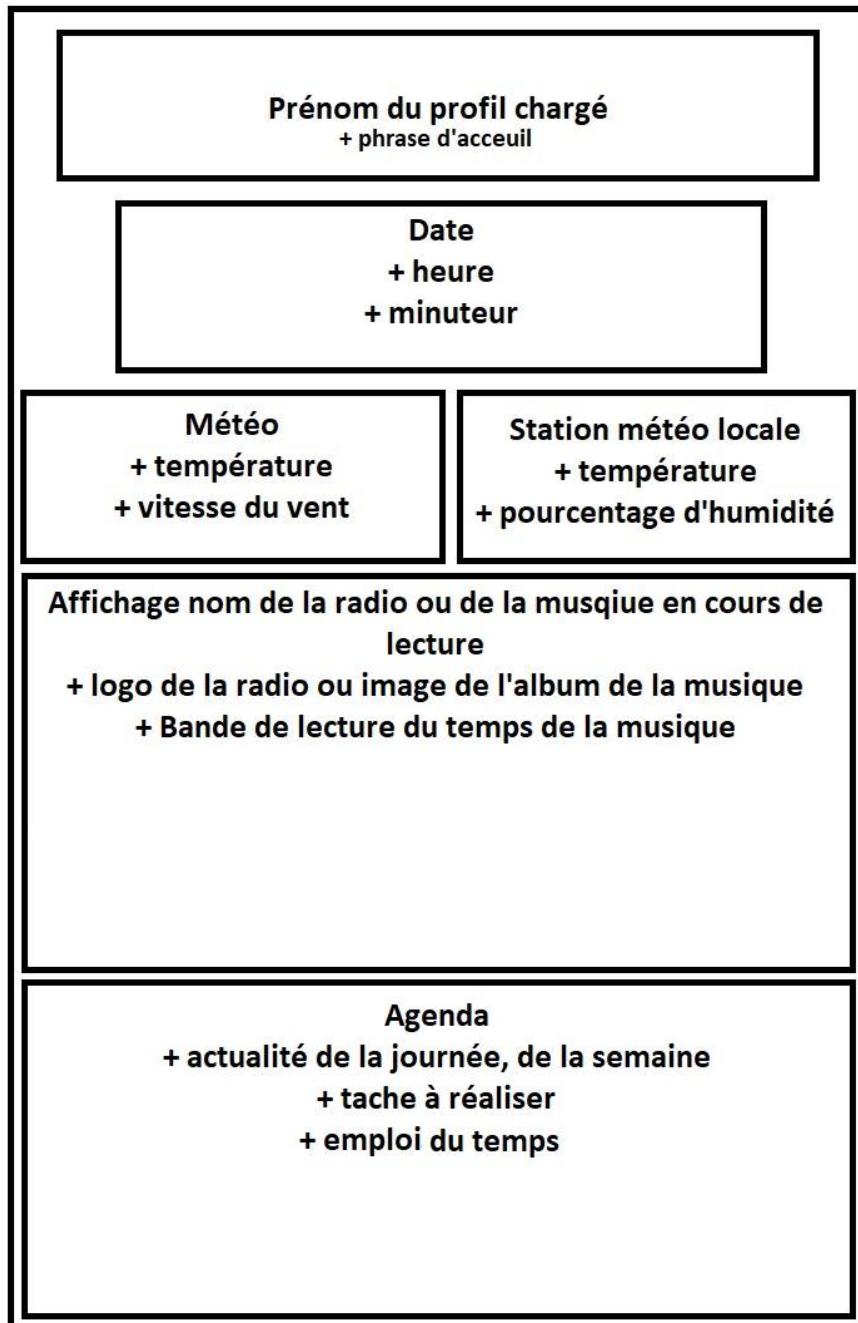
Je tiens tout d'abord à remercier Monsieur Duchiron qui s'est toujours montré à l'écoute et très disponible tout au long de la réalisation de mon projet. Je le remercie pour son aide, ses conseils et ses cours. Enfin, je n'oublie pas de remercier sincèrement Luca Grandon, Alexandre Cardona, et Adrien Hupperts qui ont fait un bout de chemin dans ce projet avec moi.

Pour conclure, la plupart des fonctionnalités du Miroir Connecté sont opérationnels tels que visualiser l'heure, la date, la radio et la musique jouée, le chargement de profil (lien contrat étudiant 4 IR), les commandes vocales (lien contrat étudiant 3 IR), le minuteur (avec une valeur fixe), les réponses vocales avec des valeurs fiables, la météo et enfin, l'affichage général est bien organisé par rapport aux widgets présents.

Certaines fonctionnalités ne sont pas totalement opérationnelles telles que nous l'imaginons. Avec le temps qu'il nous reste, je vais, avec l'aide de Luca (étudiant 3 IR), faire en sorte que l'utilisateur puisse choisir la valeur du minuteur grâce aux commandes vocales. J'intégrerai les programmes d'Adrien (étudiant 1 EC) au reste du programme afin d'avoir les valeurs des capteurs et de pouvoir contrôler la lumière du miroir grâce aux commandes vocales. La sélection des clef USB ne marche que sur windows actuellement, je vais donc l'adapter afin que cela marche sur la raspberry. Enfin, je travaillerai sur l'interface graphique afin de la rendre encore plus compacte et organisée.

Avec plus de temps, j'aurai aimé faire un widget agenda qui serait lié à un compte google afin d'offrir encore plus de fonctionnalités à l'utilisateur. J'aurais approfondi le widget radio et musique afin d'ajouter une barre de lecture et le logo de la radio jouée et pour la musique ajouté l'image de l'album joué.

J'ai réalisé un schéma qui montrerait les changement que ça apporterait au miroir



Ce projet m'a appris à être à l'écoute de mon équipe. Nous avons favorisé le travail d'équipe afin de travailler tous ensemble en prenant en compte les idées des autres. Cette expérience m'a été très bénéfique en m'apportant de nouvelles connaissances et compétences.

## CONTRAT Étudiant 3 IR GRANDON Luca

### I- Présentation du contrat

Mon contrat s'occupe de la création d'une interface administrateur, permettant de créer un profil, le modifier ou le supprimer dans la base de données du miroir, permettant dès lors de charger automatiquement un profil en fonction de la personne détectée via l'intermédiaire du contrat 2.3.4.

Mon contrat s'occupe également du système de reconnaissance vocale, permettant à l'utilisateur, via l'intermédiaire d'une phrase de demander l'heure, la date, changer la radio ou encore d'allumer ou d'éteindre la lumière.

	!	Plan...	Nom de tâche	Préde...	Durée	Début	Fin	Progr...	Priorité	Ressources	Trava...
1		➡	Miroir connecté		31,5 jours?	05/01/2022	17/06/2022	80%	500		804,33 h
2	✓	➡	□ Préparation début de projet/1er Revue		2,75 jours	05/01/2022	19/01/2022	100%	500		28,33 h
9	✓	➡	Création du Gantt	4	0,75 jour	12/01/2022	12/01/2022	100%	500	Luca Grandon	6 h
12	✓	➡	Diagramme Exigence (Découpé par contrat)	11	0,5 jour	19/01/2022	19/01/2022	100%	500	Luca Grandon	4 h
19		➡	□ Contrat 2.4		29,75 jours	18/01/2022	17/06/2022	82%	500		754 h
51	✓	➡	□ Contrat 2.4.3 (Grandon Luca)		19,5 jours	18/01/2022	17/05/2022	100%	500		172 h
52	✓	➡	□ Administration		19,5 jours	18/01/2022	17/05/2022	100%	500		86 h
53	✓	➡	Documentation Django/Python		3,5 jours	18/01/2022	28/01/2022	100%	500	Luca Grandon	28 h
54	✓	➡	Créer la table "profil" dans la BDD	53	0,5 jour	01/02/2022	01/02/2022	100%	500	Luca Grandon	4 h
55	✓	➡	Remplir la table profil (défaut)	54	2 jours	02/02/2022	23/02/2022	100%	500	Luca Grandon	16 h
56	✓	➡	Permettre de pouvoir créer des profils dans la bdd via l'interface	55	1,25 jours	01/03/2022	02/03/2022	100%	500	Luca Grandon	10 h
57	✓	➡	Permettre de pouvoir modifier et supprimer des profils de la bdd	56	1 jour	04/03/2022	08/03/2022	100%	500	Luca Grandon	8 h
58	✓	➡	Migration raspberry	57	0,75 jour	04/05/2022	04/05/2022	100%	500	Luca Grandon	6 h
59	✓	➡	Régler le problèmes BDD / Image	58	1,75 jours	10/05/2022	17/05/2022	100%	500	Luca Grandon	14 h
60	✓	➡	□ Gérer les commandes vocales		11,25 jours	09/03/2022	17/05/2022	100%	500		86 h
61	✓	➡	Renseignement SpeechRecognition et PyAudio	57	2,5 jours	09/03/2022	16/03/2022	100%	500	Luca Grandon	20 h
62	✓	➡	Coder la reconnaissance vocal (Python)	61	6,5 jours	22/03/2022	04/05/2022	100%	500	Luca Grandon	52 h
63	✓	➡	Fusion IHM/RecoV	62	0,5 jour	10/05/2022	10/05/2022	100%	500	Luca Grandon	4 h
64	✓	➡	Résoudre tout les soucis	63	1,25 jours	11/05/2022	17/05/2022	100%	500	Luca Grandon	10 h

## II- Ressources, matériel utilisées et choix technologiques

### a- Ressources logiciels

Afin de réaliser à bien mon contrat, j'ai dû utiliser divers logiciels tel que :

- PyCharm

Un IDE Python, permettant de réaliser les divers programmes du projet.

- MindView

Un logiciel utilisé pour créer, modifier et mettre à jour le diagramme de Gantt qui exprime les tâches à faire au cours du projet à faire ou terminé ainsi que l'avancé de celui-ci.

- Draw.io

Logiciel permettant de concevoir les diagrammes UML et SysML du projet (classe, exigence, etc..)

- MariaDB

Un système de gestion de base de données qui nous permet de vérifier le bon fonctionnement de la base de données.

- Mozilla Firefox

Navigateur web, m'ayant permis de trouver des documentations et des libraires à utiliser voir des frameworks comme "Django" (voir choix technologique).

- OpenClassroom

Un site web pour apprendre à programmer de manière simple et expliqué, m'ayant aidé à programmer l'outil d'administration via Django.

### b- Ressources Matériels

Les ressources matériels utilisées au cours de mon contrat sont :

-Ordinateur Windows fixe

Afin d'effectuer divers tests, les programmations ainsi que de créer les différents diagrammes et mettre à jour le diagramme de Gantt.

-Raspberry Pi 3 sous Raspbian

Pour effectuer les premiers tests d'interaction avec linux ainsi qu'avec la base de données "Dummy".

-Raspberry Pi 4 sous Raspbian

Elle est utilisée pour la version finale du miroir connecté et effectue les portages des différents systèmes.

## c- Choix technologiques

Pour réaliser mon contrat ainsi que le projet, divers choix technologiques

Mon choix principal portait sur le Python, ayant comparé les avantages et les inconvénients de celui-ci avec le JavaScript quant à la conception de l'interface administrateur.

Le langage de programmation que nous avons finalement choisi avec le groupe est le Python car ce langage est facile à apprendre, à comprendre et à coder. Également ce n'est pas un langage compilé, il est interprété ce qui facilite grandement le portages d'un système à l'autre.

Du fait que notre projet est porté sur une carte Raspberry pi 4 sous Linux. Le choix du python était donc nécessaire, puisque nous codons sous Windows puis intégrons sous Linux soit un autre OS.

Enfin, c'est un langage très documenté sur internet, ce qui m'a permis de régler plus rapidement quelques erreurs que j'ai eues avec l'aide de divers forum (comme StackOverflow).

Pour conclure sur ce choix, grâce au langage Python, nous avons été plus productif que si nous avions choisi un autre langage de programmation.

Suite à ce choix, j'ai fait diverses recherches sur la manière de créer une interface administrateur jusqu'à trouver le framework "Django", permettant de créer un site en localhost pouvant être relié à une base de données.

Celui-ci est simple d'apprentissage au départ quand à la confection de page web ou de création de base de données.

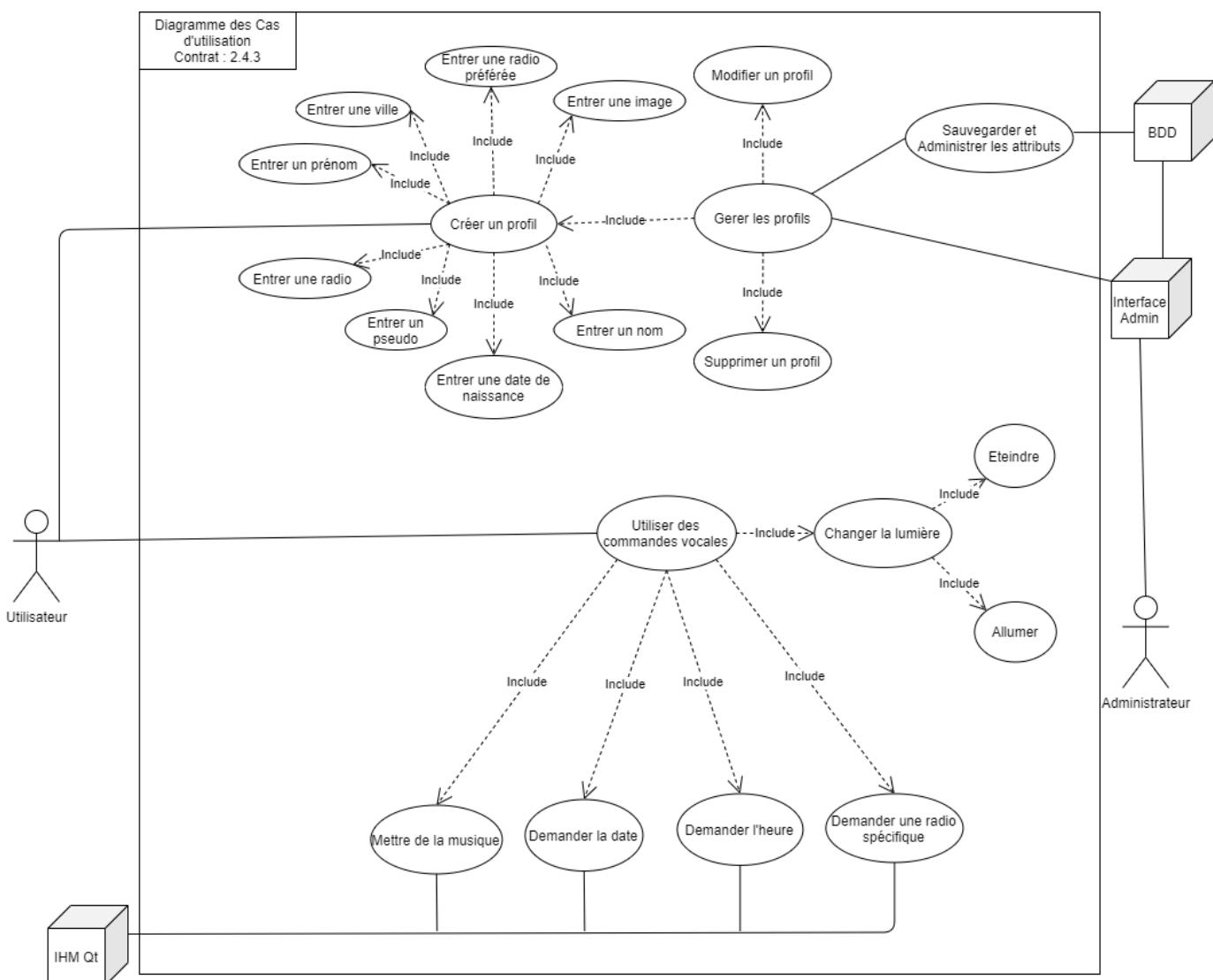
Grâce à sa communauté active et aux différentes aides aussi bien d'architecture logiciel via le MVC (Model-View-Controller) ou au programme informatique ORM (Object relational mapping) qui permettent une de plus grande facilité quant à la conception de l'interface.

Suite à cela j'ai aussi eu à programmer la reconnaissance vocale via Python, j'ai donc décidé d'utiliser la librairie "SpeechRecognition" ainsi que la librairie "pyAudio".

PyAudio permet de créer des ports audio afin de pouvoir enregistrer ou écouter un son provenant d'un microphone, ce qui permettra via Speech Recognition et son AI Google de vérifier la phrase, si elle est dans la bonne langue (dans notre cas Français Fr-FR) et si elle à un sens.

### III- Présentation des diagrammes

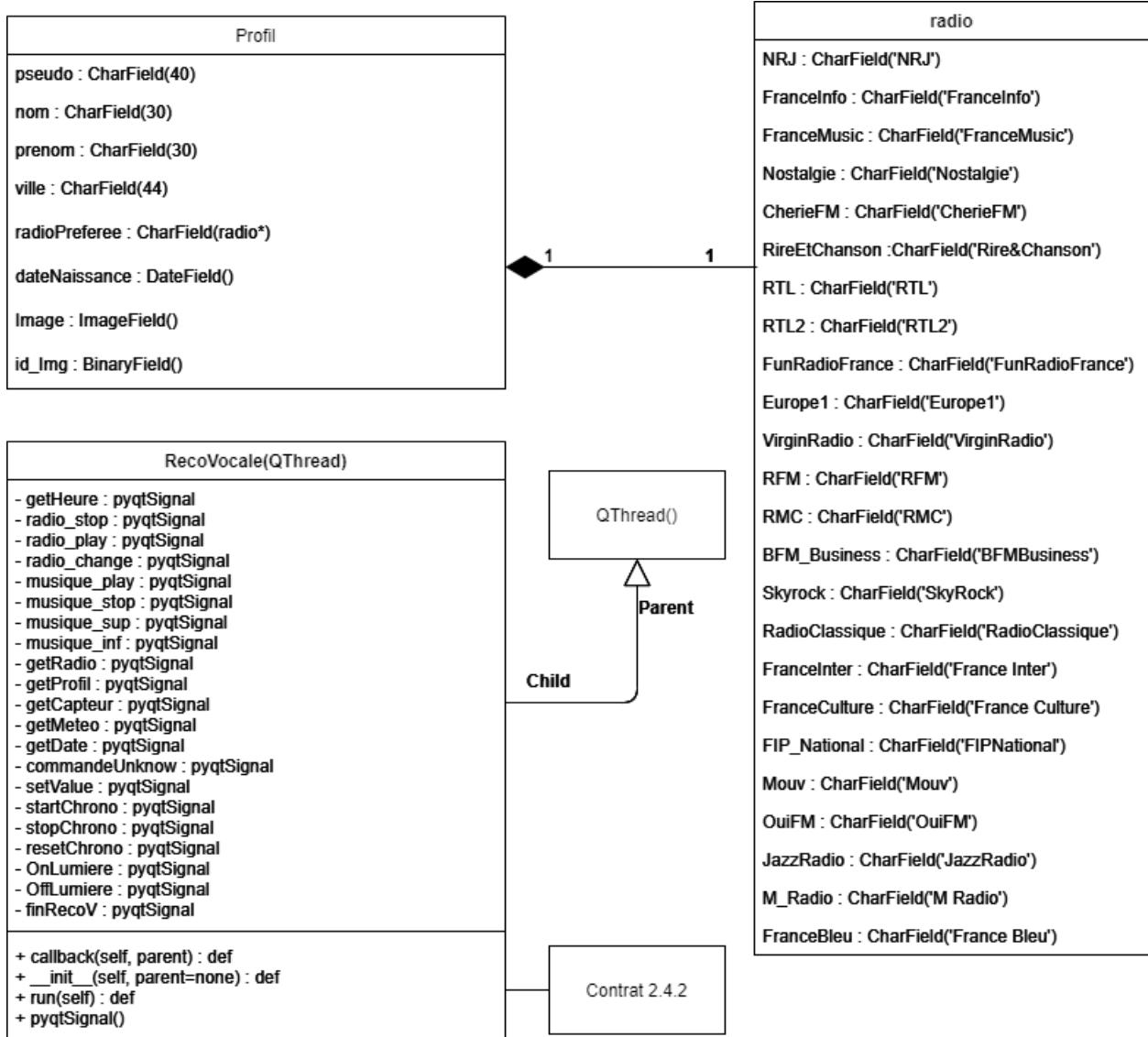
## a- Diagrammes des cas d'utilisation



Dans ce diagramme des cas d'utilisation on peut voir 2 acteurs et 3 systèmes extérieurs. L'utilisateur peut utiliser des commandes vocales tel qu'allumer la lumière ou l'éteindre, le reste des commandes tel que pour demander la date ou l'heure envoie des signaux vers l'IHM afin de répondre aux demandes.

L'utilisateur peut aussi au départ créer un profil en entrant plusieurs paramètres (obligatoires) et ensuite avoir accès à la modification ou la suppression dudit profil qui modifiera la base de données liée à l'interface admin.

## b- Diagramme des classes



A l'intérieur de ce diagramme de classe on peut voir les différentes classes créées et utilisées au cours de ce projet.

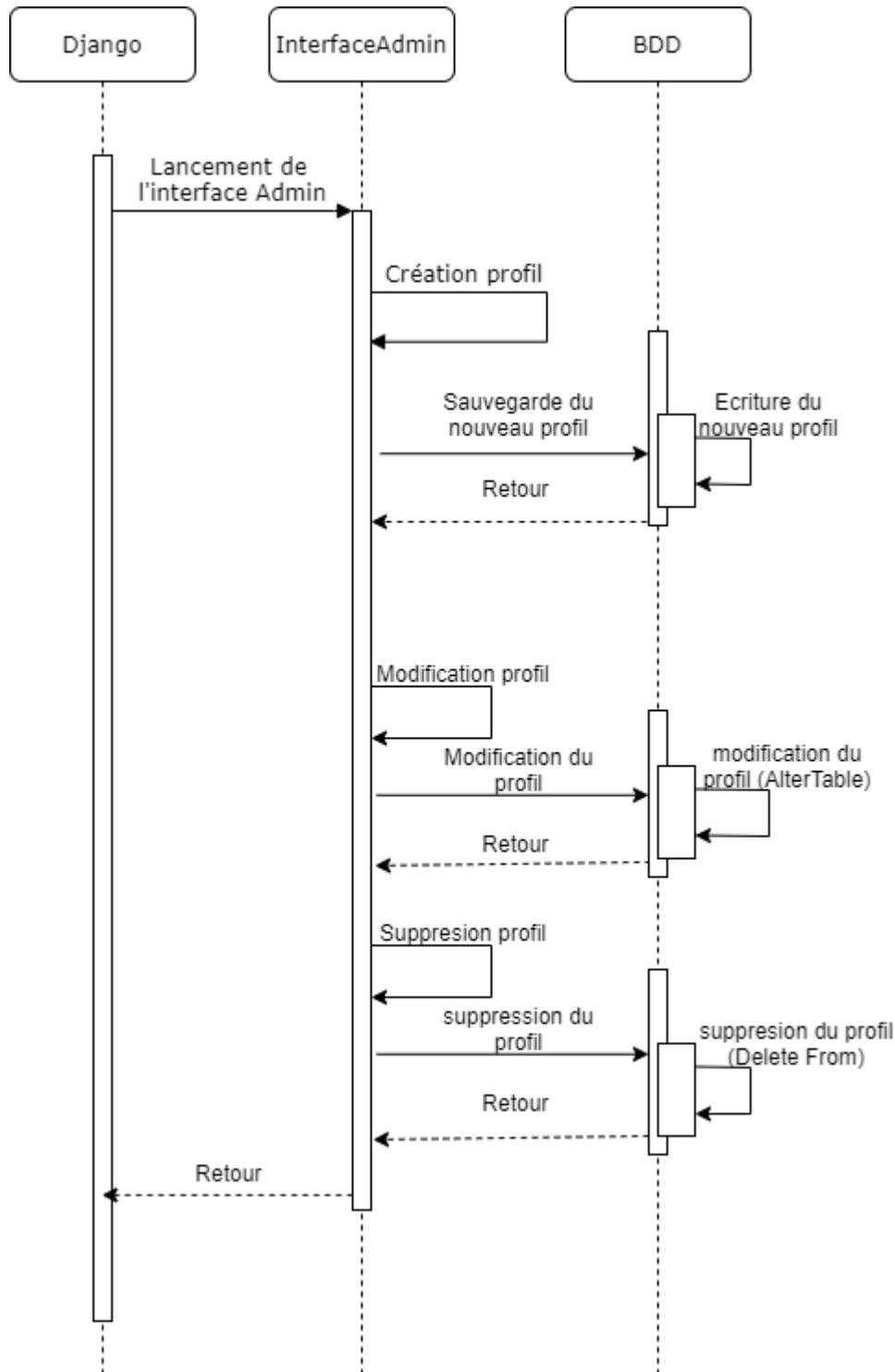
Pour la partie Django il faudra se focaliser sur les classes "Profil" et "Radio" liées par une agrégation.

La classe "Profil" est composée des différents paramètres qu'un profil peut prendre comme un nom, prénom, un pseudo ou une date de naissance, ceux-ci pouvant ensuite remplir la base de données lorsqu'un profil est créé sur l'interface administrateur.

La classe "Radio" elle, est visible comme un menu déroulant dans l'interface administrateur, servant à choisir, dans la classe Profil, la "Radio Préférée" via la liste montrée dans la classe radio.

La partie "reconnaissance vocale" elle, est composée de la classe **RecoVocale(QThread)**, utilisant un héritage de la classe "QThread" et étant lié au contrat 2.4.2 soit, l'Interface Homme Machine.

c- Diagramme de séquence de l'interface administrateur (Django)



Dans le diagramme de séquence, précisant l'utilisation de Django.

Tout d'abord, on commence par lancer l'interface administrateur via une commande dans le CMD de Python, permettant de lancer le serveur et donc, les différentes pages du site de l'interface administrateur.

Suite à cela, si nous avons déjà un profil il existe 3 boutons, "ajouter un profil", "modifier" ou "supprimer le profil", autrement nous n'avons accès qu'au fait de créer un profil.

Pour créer un profil il suffit d'appuyer sur le bouton "ajouter un profil" ce qui ouvrira l'accès au "formulaire" du profil, demandant les variables vu dans le diagramme des classes.

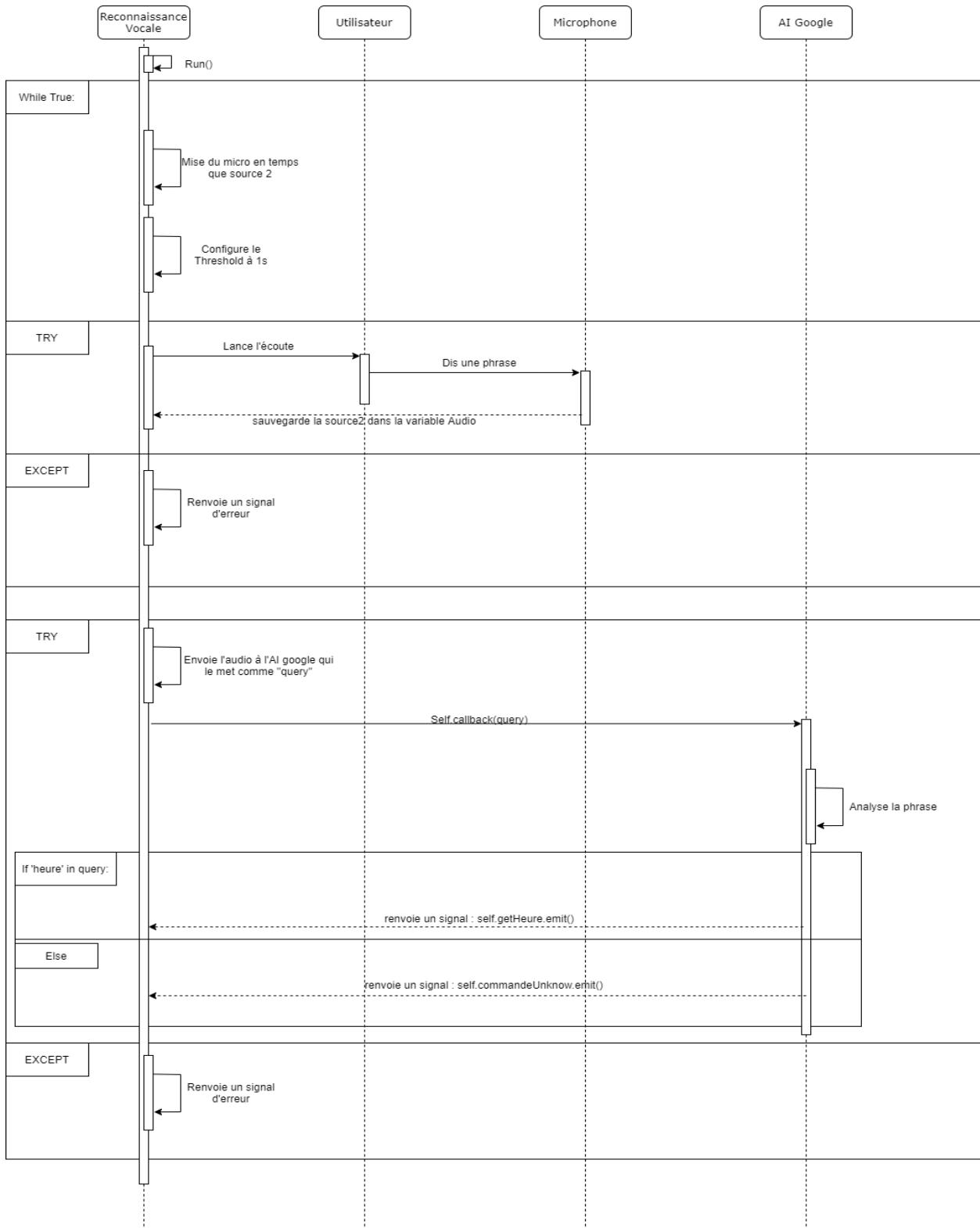
Un fois celui-ci rempli en envoyer, il est sauvegardé dans la base de données comme un nouveau profil et écrit à l'intérieur de celle-ci les paramètres reçus depuis l'interface administrateur.

Une fois le profil existant, il est visible depuis l'interface admin via son pseudo, permettant de vérifier les paramètres, si certains doivent changer comme par exemple votre pseudo, ou votre radio préféré il suffira d'appuyer sur modifier, ouvrant une page où vous pourrez modifier chaque paramètre.

Une fois terminé et que vous aurez cliqué sur "envoyer" les modifications apportées seront aussi apportées dans la base de données et elle modifiera donc le profil correspondant.

La dernière possibilité est de supprimer un profil, permettant après avoir vérifié si vous êtes sûr de vouloir supprimer le profil, de le supprimer du site mais aussi de la base de données, supprimant la ligne du profil correspondant.

#### d- Diagramme de séquence de la reconnaissance vocale



Le diagramme de séquence de la reconnaissance vocale contient 4 grands éléments, tout d'abord le programme de reconnaissance, l'utilisateur, le micro et enfin l'AI Google.

Au début on laisse le signal avant de rentrer dans un "While:True" permettant de laisser

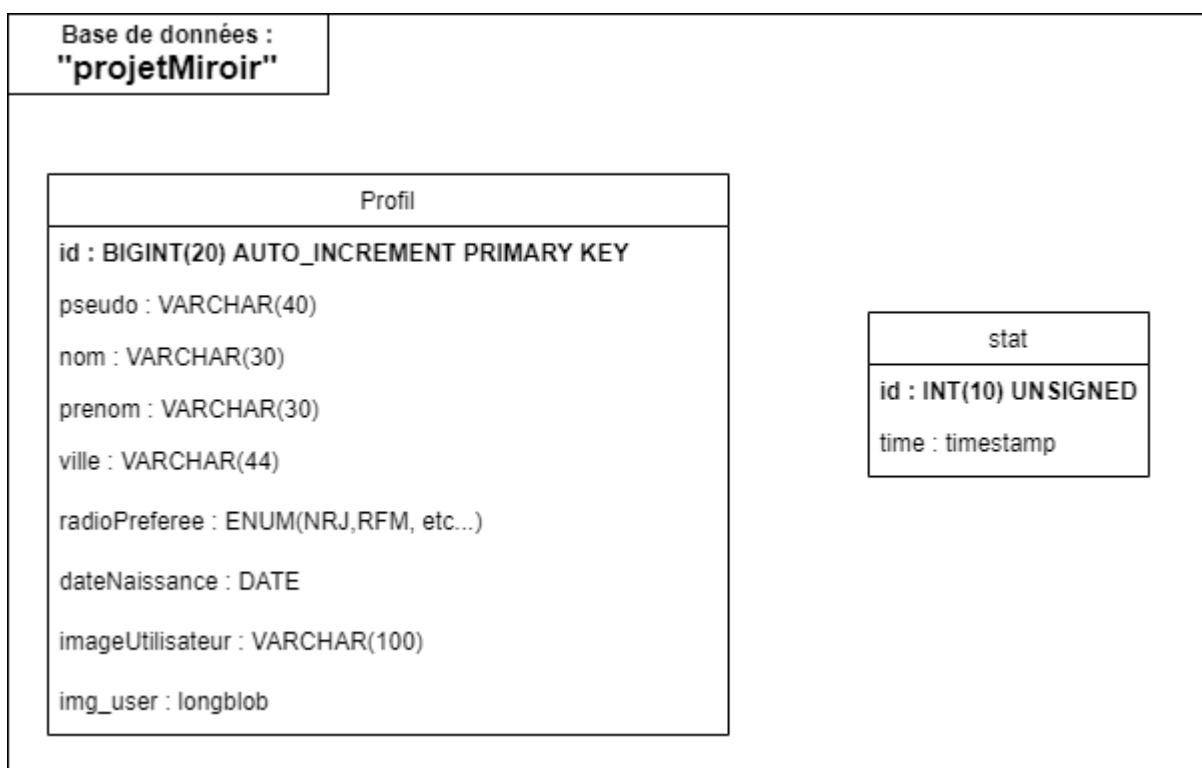
tourner le programme en boucle, celui-ci met le micro en source et configure le threshold à 1s, avant d'entrer dans un "TRY".

Dans le "TRY" on lance l'écoute, et on attend un retour.

Lorsque l'écoute est lancée, l'utilisateur peut parler dans le micro et donner sa commande vocale, ici, demander l'heure, celui-ci parle donc dans le micro qui renvoie la phrase dans la variable Audio.

Si celui-ci ne reconnaît pas, il renvoie une erreur.

#### e- Diagramme de base de données



La base de données du projet est composée de deux grandes tables que nous utiliserons, la table Profil, composée des éléments présents dans la classe "Profil", mais aussi de la table "stats" permettant de récupérer les infos temporelles du temps passé devant son miroir lorsque votre profil est chargé.

(Sachant que la table "stats" n'est que prototype, celle-ci ne garde actuellement que l'heure du login et de la déconnexion).

### i - Base de données sous raspberry

```
pi@raspberrypi:~/Documents/django/django-web-app/merchex $ sudo mysql -u projet -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 43
Server version: 10.5.15-MariaDB-0+deb11u1 Raspbian 11

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use projetMiroir;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [projetMiroir]> show tables;
+-----+
| Tables_in_projetMiroir |
+-----+
| auth_group
  auth_group_permissions
  auth_permission
  auth_user
  auth_user_groups
  auth_user_user_permissions
  django_admin_log
  django_content_type
  django_migrations
  django_session
  listings_profil
  stats
+-----+
12 rows in set (0.001 sec)
```

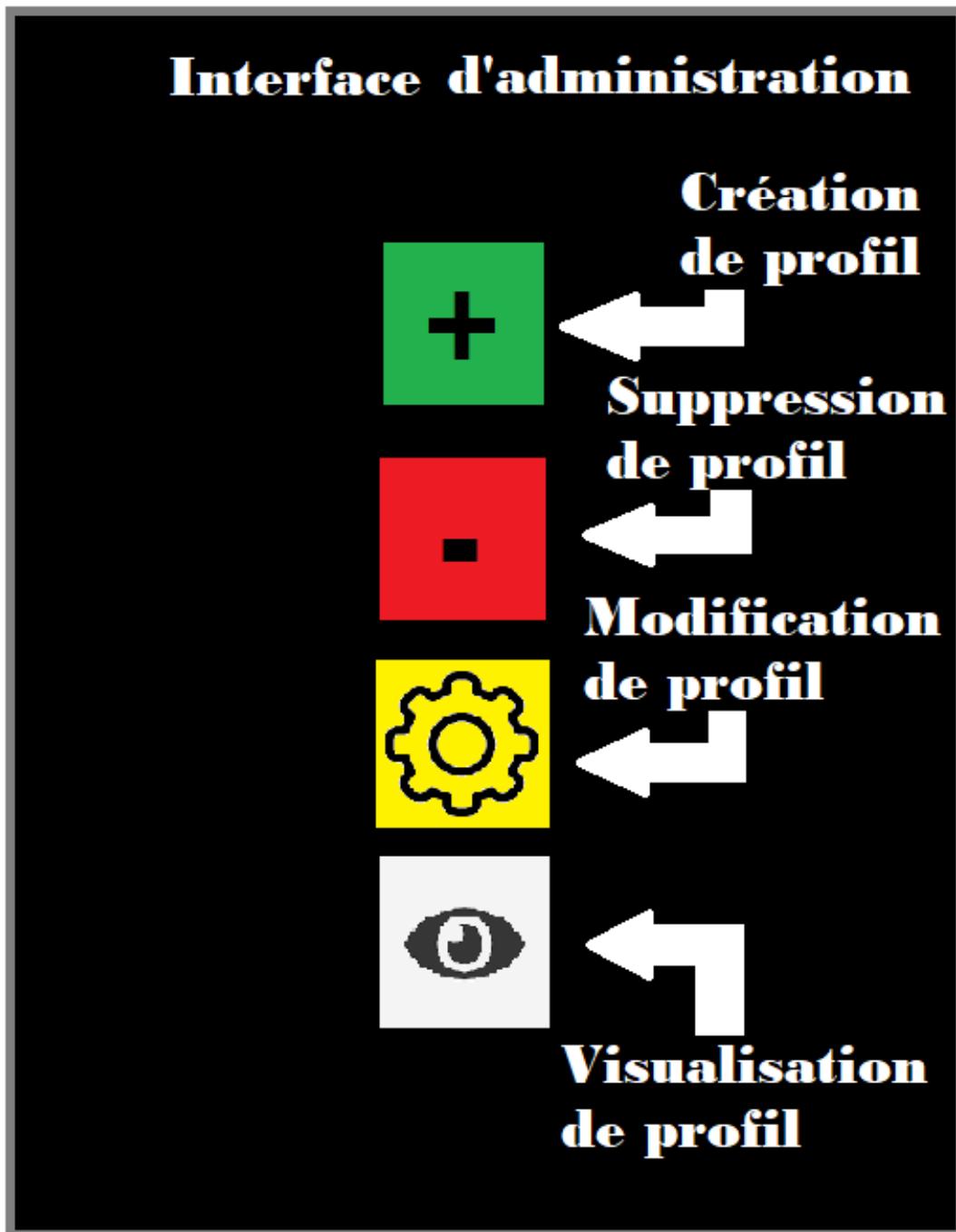
La base de données visible sous raspberry nous montre d'autres tables présentes qui sont en fait les tables liées au framework "Django".

Celles-ci permettent en réalité d'utiliser les commandes SQL pour assigner les données transmises via l'interface administrateur dans la bonne table de base de données.

Suivant alors les sessions, les migrations (modifications de la classe profil ou radio par exemple afin de modifier la base de données) ou encore montrer la liste des "SuperUser" que nous n'utilisons pas dans ce projet.

## IV - Interface administrateur

### a - Idée de base



Cette image représente mon idée basique d'interface administrateur via laquelle nous aurions pu créer des profils, les modifier ou les supprimer.

## b - Interface administrateur actuelle

Mais les résultats finaux sont plus légers au vu de mes connaissances en CSS, préférant alors me concentrer sur la reconnaissance vocale plutôt que sur l'esthétique de l'interface administrateur.

Cette interface se devait de répondre à des demandes simples comme créer un profil, le modifier et pouvoir le supprimer tout en le sauvegardant sur une base de données.

J'ai donc opté pour un habillage assez simple lorsque les fonctionnalités étaient belles et bien implémentées, un habillage noir avec une écriture blanche.

L'interface faite en django est elle même dénotée sur la localhost soit l'adresse local de la machine sous le port 8000, soit l'adresse "127.0.0.1:8000".

J'ai ensuite découpé l'interface d'administration en 4 grandes parties, l'une permettant de créer les profils (voir partie 'ii - page de création de profil'), une afin des les modifieurs ('iii - page de modifications de profil') et une 'iiii - page de suppression de profil').

La page d'accueil à aussi était faite, permettant de voir les profils existant (et voir leurs paramètres en cliquant dessus, mais aussi en ayant accès au 3 autres pages assez facilement).

Cette interface est liée à la base de données du projet, contenant alors la table profil qui est équivalente à ceux rentrer dans l'interface admin.

### i - Page d'accueil



ii - Page de création de profil

127.0.0.1:8000/profil/add/

## Bienvenue sur la section création d'utilisateur !

(N'oubliez pas de modifier les paramètres par défauts!)

Pseudo :

Nom :

Prenom :

Ville :

RadioPreferee :

DateNaissance :

Img User :  Aucun fichier sélectionné.

[Retour à tous les profils](#)

iii - Page de modification de profil

127.0.0.1:8000/profil/3/update/

## Mise à jour de l'utilisateur :

(La date de naissance est en JJ/MM/YYYY, n'oubliez pas les "/" entre vos nombres)

Pseudo :

Nom :

Prenom :

Ville :

RadioPreferee :

DateNaissance :

Img User : Actuellement: **True**  
Modification:  Aucun fichier sélectionné.

[Retour à tous les profils](#)

ffff - Page de suppression de profil



c - Exemple de code d'une page Django

```
def profil_add(request):
    if request.method == 'POST':
        form = ProfilForm(request.POST)
        if form.is_valid():
            profil = form.save(commit=False)
            profil.save()
            return redirect('profil-detail', profil.id)
    else:
        form = ProfilForm()

    return render(request, 'listings/profil_add.html', {'form': form})
```

Ci-dessus est visible un exemple de page, celle ci étant celle de création de profil. Si la page reçoit une méthode 'post', via l'envoie d'un profil, celle-ci crée une requête "form" qui, si elle est valide, sauvegarde le profil dans la base de données ainsi que dans le site d'administration relié avant de renvoyer sur les détails du profil créé. Autrement, celle-ci ne fait rien et renvoie simplement sur la page de création de profil en réinitialisant le "form" soit, un profil.

```

{% extends 'listings/base.html' %}

{% block content %}

<h1>Bienvenue sur la section création d'utilisateur !</h1>
<h4>(N'oubliez pas de modifier les paramètres par défauts!)</h4>

    <form action="" method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <input type="submit" value="Envoyer">
    </form>

<a href="{% url 'profil-list' %}">Retour à tous les profils : </a>

{% endblock %}

```

L'html de la page est assez simple, nous avons tout d'abord un grand titre en "header 1" ou "<h1>" ainsi qu'une phrase informative en "header 4".

Cette page est donc codée de sorte à remplir un form via "{{form.as\_p}}" présent avant le type d'entrer "submit" ou envoyer vers la fonction "profil\_add(request)" vu au dessus.

En bas de la page se trouve un lien permettant de retourner à la page contenant tous les profils, si l'on s'est trompé de page par exemple.

Quand à l'habillage du site, il est très simple, ne tenant actuellement que sur 2 lignes de codes servant simplement à changer la couleur de fond de page en noir et les mots en blanc.

```

/* { background: black }
* { color: white }

```

## V - Reconnaissance vocale

### a - Explication

La reconnaissance vocale sert à modifier des paramètres, demander certaines choses tels que l'heure, la date ou encore à changer la musique en cours ou la radio. Lorsqu'une commande vocale est donnée, elle renvoie un signal vers l'IHM de l'étudiant Maxence (vers contrat IR 1) avec qui, nous avons numérisé nos idées et ajouté les

signaux.

Une semaine à été réquisitionné pour travailler en binôme avec mon camarade Maxence pour rendre fonctionnel la reconnaissance vocale et l'envoie de signaux afin de faire marcher l'IHM via commande vocale.

La reconnaissance vocale, faite avec Maxence, sert donc à relier des phrases contenant des mots clés afin de permettre une utilisation plus fluide du miroir sans avoir de bouton mais aussi afin de ne pas avoir à poser ses doigts sur le miroir.

## b - Programme

La classe "RecoVocale" possède plusieurs signaux permettant de renvoyer des signaux précis en fonction des demandes de l'utilisateur, ce signal servira ensuite à modifier certains paramètres de l'IHM ou alors afin de recevoir des réponses.

Cette classe utilisé aussi plusieurs fonctions comme "run" étant la fonction écoutant le micro et le notifiant comme la source, permettant ensuite de récupérer la phrase de l'utilisateur et de la reconnaître avant de la placer dans la variable "Query" avant d'appeler la fonction "callback" contenant la variable Query.

```
def run(self):
    while True:
        r = sr.Recognizer()
        print("test1")
        with sr.Microphone() as source2:
            #r.adjust_for_ambient_noise(source2)
            print("listening")
            r.pause_threshold = 1
            try:
                print("dans le try")
                audio = r.listen(source2)# si snowboy alors timeout=5
            except Exception as e:
                print(e)
            print("écoute réussie")
        try:
            print("Reco en cours...")
            query = r.recognize_google(audio, language='fr-FR')
            print("Phrase reconnue :" + query)
            self.callback(query)

        except Exception as e:
            print(e)
        print("test2")
```

Une fois la fonction callback après, on essaye de reconnaître les demandes via des mots clés en regardant s'ils se trouvent, ou non, à l'intérieur de la variable "Query" soit la phrase reconnue.

Si elle est reconnue par une commande via l'intermédiaire des mots clés elle renvoie le signal correspondant (exemple avec une demande de l'heure).

```
def callback(self, query):

    try:
        if 'heure' in query:      #Quel heure est'il ?
            print('1')
            self.getHeure.emit()
```

Dans le contrat, 2.4.2, le signal "self.getHeure.emit()" servira à donner l'heure à son utilisateur via une voix synthétique lisant l'heure actuelle.

Le retour de message visible dans le terminal durant la demande d'une commande vocale.

```
listening
dans le try
écoute réussie
Reco en cours...
Phrase reconnu :mets NRJ
5
```

Sachant que le code de la commande 'NRJ' renvoie bien 5 en plus du signal, celle-ci renvoie bien la bonne chose en plus du signal lançant la radio NRJ.

```
elif "mets NRJ" in query: #mets NRJ
    print('5')
    self.nomRadio = "NRJ"
    self.radio_play.emit()
```

## VI - Problèmes rencontrés

### a - Django et Interface administrateur

Au cours de ce projet, plusieurs problèmes sont apparu, tout d'abord sur l'interface administrateur Django, étant principalement des problèmes de liaisons de base de données au départ, ne sauvegardant pas les profils fait, qui fût résolu via l'utilisation de PhPMyAdmin sur Windows puis via MariaDB en donnant les bons liens de base de données avec login et mot de passe afin de bien sauvegarder les données.

Ensuite d'autres problèmes ont vu le jour, l'implémentation des pages web via des erreurs bêtes, oubliant certaines parties de programme autant Python que Html pour l'affichage des profils, de leurs créations ou de leurs modification réglés simplement en relisant et en utilisant les aides trouvées sur le web ainsi que par les conseils de M.Duchiron.

Les liens entre les tables de bases de données ont été un problème qui, au départ était assez grave, Django n'acceptant que sa base de données en local et pas de clés étrangères, il à fallu relier les tables de "Profil" et de "RecoFacial" afin de régler le problème.

Un problème actuellement présent est l'ajout d'image, qui sous windows renvoie une erreur et qui sous Linux fonctionne inégalement, nous cherchons encore actuellement une solution afin de corriger cette erreur dans les plus bref délai.

### b - Reconnaissance vocale

La reconnaissance vocale n'a posé qu'un seul problème venant de la recherche de mot clé dans la variable "query" demandant alors de modifier toute la fonction "callback" afin de la rendre fonctionnelle, celle ci, n'acceptant pas les opérateurs logiques tel que les "AND", "OR", "XOR".

Corrigez simplement en modifiant chaque phrase à donner ou mots clés par une seule et même phrase pour celles-ci.

Il y a aussi le problème de la radio durant les commandes vocales pouvant rendre plus difficile l'écoute, la solution serait d'augmenter le "threshold", afin de n'écouter qu'au dessus d'un certains nombres de décibels.

## VII - Remerciements

Je tiens à remercier sincèrement Monsieur F.Duchiron, et Monsieur P.Dubois qui, en tant que professeur encadrant les projets, se sont montrés toujours à l'écoute et très disponibles tout au long de la réalisation de ce projet.

Ainsi je les remercie pour leurs aides et tout le temps qu'ils ont bien voulu me consacrer afin de répondre à mes questions et m'aider lors de problèmes.

Enfin, je n'oublie pas de remercier sincèrement Maxence Laporte, Alexandre Cardona, et Adrien Hupperts qui ont fait un bout de chemin dans ce projet avec moi et sans eux, je ne serais pas arrivé jusqu'ici sans leurs aides et la bonne ambiance du groupe.

## VIII - Conclusion et évolutions

Pour conclure, la plupart des fonctionnalités du Miroir Connecté sont opérationnelles telles que créer des profils, les modifier, supprimer mais aussi les commandes vocales.

Certaines fonctionnalités ne sont pas totalement opérationnelles telles que nous l'imaginons. Avec le temps qu'il nous reste, je vais, avec l'aide de Maxence (étudiant 2 IR), faire en sorte que l'utilisateur puisse choisir la valeur du minuteur grâce aux commandes vocales ou encore faire un "SnowBoy" permettant via une commande vocale tel que "Ok Google" ou "Dit Siri" de pouvoir activer la reconnaissance vocale.

Mais aussi travailler si le temps nous le permet sur le CSS et enfin régler les derniers soucis de l'interface administrateur mais aussi de travailler la mise en réseau de l'interface administrateur sur le réseau local afin d'avoir accès à la création de profil.

Ce projet m'a beaucoup appris, autant dans le travail d'équipe que dans l'écoute des autres et l'analyse des programmes de mes camarades afin de travailler en adéquation avec leurs programmes et leurs idées.

Les réalisations que j'ai pu faire, en apprenant de nouvelles choses me seront bénéfiques dans le futur au vu des connaissances acquises et des compétences humaines développées au cours de ce projet.

## CONTRAT Étudiant 4 IR CARDONA Alexandre

### I - Présentation du contrat

Mon contrat porte sur la création d'une reconnaissance faciale par caméra sur la Raspberry Pi 4 qui permet de charger automatiquement un profil en fonction de la personne détectée. Le programme renvoie l'Id du profil détecté pour pouvoir charger le profil correspondant dans le but de donner les informations demandées.

Mon système de reconnaissance faciale est capable d'insérer dans la base de données des images des utilisateurs en binaire. Pour les utiliser dans le programme de la reconnaissance.

Également, mon système est capable de stocker des statistiques d'utilisation des différents profils dans la base de données.

Voici comment je me suis organisé pour mon contrat dans le temps :

	Plan...	Nom de tâche	Prédé...	Durée	Début	Fin	Progr...	Priorité	Ressources	Travail
1	→	Miroir connecté		31,5 jours?	05/01/2022	17/06/2022	80%	500		804,33 h
2	✓ →	□ Préparation début de projet/1er Revue		2,75 jours	05/01/2022	19/01/2022	100%	500		28,33 h
3	✓ →	Création des croquis	4	0,25 jour	12/01/2022	12/01/2022	100%	500	Alexandre Cardona	2 h
4	✓ →	recherche des matériaux	4	0,37 jour	14/01/2022	14/01/2022	100%	500	Alexandre Cardona	3 h
5	✓ →	Préparer bon de commande (V.1)	7	0,25 jour	14/01/2022	18/01/2022	100%	500	Alexandre Cardona	2 h
6	→	□ Contrat 2.4		29,75 jours	18/01/2022	17/06/2022	82%	500		754 h
65	✓ →	□ Contrat 2.4.4 (Cardona Alexandre)		19,5 jours	18/01/2022	17/05/2022	100%	500		148 h
66	✓ →	□ Reconnaissance faciale par caméra		19,5 jours	18/01/2022	17/05/2022	100%	500		148 h
67	✓ →	Installation Linux / Ajout bibliothèque OpenCV		1,25 jours	18/01/2022	19/01/2022	100%	500	Alexandre Cardona	10 h
68	✓ →	Configuration OpenCV	67	1 jour	21/01/2022	25/01/2022	100%	500	Alexandre Cardona	8 h
69	✓ →	Documentation OpenCV (apprentissage)	68	0,75 jour	26/01/2022	26/01/2022	100%	500	Alexandre Cardona	6 h
70	✓ →	"Programmer" le système de reconnaissance faciale	69	2,25 jours	28/01/2022	22/02/2022	100%	500	Alexandre Cardona	18 h
71	✓ →	Test du fonctionnement du matériels	70	0,75 jour	23/02/2022	23/02/2022	100%	500	Alexandre Cardona	6 h
72	✓ →	Réparation code	71	0,5 jour	01/03/2022	01/03/2022	100%	500	Alexandre Cardona	4 h
73	✓ →	Lier la reconnaissance faciale et la BDD	72	0,75 jour	02/03/2022	02/03/2022	100%	500	Alexandre Cardona	6 h
74	✓ →	Faire en sorte que les données et statistiques de la RF se stocke dans la BDD	73	0,5 jour	04/03/2022	04/03/2022	100%	500	Alexandre Cardona	4 h
75	✓ →	Création des classes	74	0,5 jour	08/03/2022	08/03/2022	100%	500	Alexandre Cardona	4 h
76	✓ →	Fusion IHM / recof	75	0,75 jour	09/03/2022	09/03/2022	100%	500	Alexandre Cardona	6 h
77	✓ →	Upgrade du programme	76	2,5 jours	16/03/2022	23/03/2022	100%	500	Alexandre Cardona	20 h
78	✓ →	Fusion IHM/recof 2	77	0,5 jour	25/03/2022	25/03/2022	100%	500	Alexandre Cardona	4 h
79	✓ →	Modification pour migration global vers raspberry	78	2,75 jours	30/03/2022	26/04/2022	100%	500	Alexandre Cardona	22 h
80	✓ →	Protocole de test et test	79	0,75 jour	27/04/2022	27/04/2022	100%	500	Alexandre Cardona	6 h
81	✓ →	Résoudre les problèmes	80	1,25 jours	03/05/2022	04/05/2022	100%	500	Alexandre Cardona	10 h
82	✓ →	Mise en lien encodage image via Django (BDD/Python)	81	1,75 jours	10/05/2022	17/05/2022	100%	500	Alexandre Cardona	14 h

## II - Ressources, matériel utilisées et choix technologiques

### a - Ressources et matériels utilisés

Pour mener à bien mon contrat j'ai utilisé différents logiciels et matériels que je vais détailler ici :

-Pycharm sera l'interface de développement choisie pour la réalisation de mes différents programmes.

-MindView est le logiciel utilisé pour créer, modifier et mettre à jour mon diagramme de Gantt pour connaître mon avancée dans mon contrat.

-La documentation nécessaire sera trouvée sur internet via des forums (notamment StackOverFlow) ou via les documentations officielles Python et des différentes librairies.

-J'ai également utilisé Fusion 360 pour les modélisations 3D du miroir pour la création du prototype.

J'ai également utilisé le matériel suivant :

-Ordinateur Windows fixe pour effectuer les tests, les programmations, créer les différents diagrammes, mettre à jour le diagramme de Gantt.

-Raspberry Pi 3 sous Raspbian pour effectuer les premiers tests d'interaction avec la Base De Données.

-Raspberry Pi 4 sous Raspbian également qui, elle, est utilisée pour la version finale du miroir connecté et effectuer les portages des différents systèmes.

### b - Choix technologiques

Pour la réalisation de mon contrat, j'ai dû choisir parmi plusieurs langages de programmation. Le choix principal portait entre le C++ et le Python. Pour trancher entre tous ces langages, j'ai comparé leurs avantages et leurs inconvénients.

Le langage de programmation que nous avons finalement choisi est le Python car ce langage est facile à apprendre, comprendre et à coder. Également ce n'est pas un langage compilé, il est interprété ce qui facilite grandement le portages d'un système à l'autre, travaillant sur des postes sous Windows avec des logiciels que l'on connaît et efficace pour ce genre de tâche il est donc nécessaire de porter nos programmes sous linux. Le choix du Python est donc la meilleure option

Notre projet va être porté vers une carte Raspberry pi 4 sous Linux. Le choix du python était donc nécessaire, puisque nous codons sous Windows puis intégrons sous Linux.

Enfin, c'est un langage très documenté sur internet, ce qui m'a permis de régler plus rapidement quelques erreurs que j'ai eues. Pour conclure sur ce choix, grâce au langage Python, nous avons été plus productif que si nous avions choisi un autre langage de programmation.

Explication de l'encodage utilisé : Pour la reconnaissance faciale il existe deux types d'encodages principaux: le HOG (histogram of oriented gradients) et le CNN (Convolutional Neural Network).

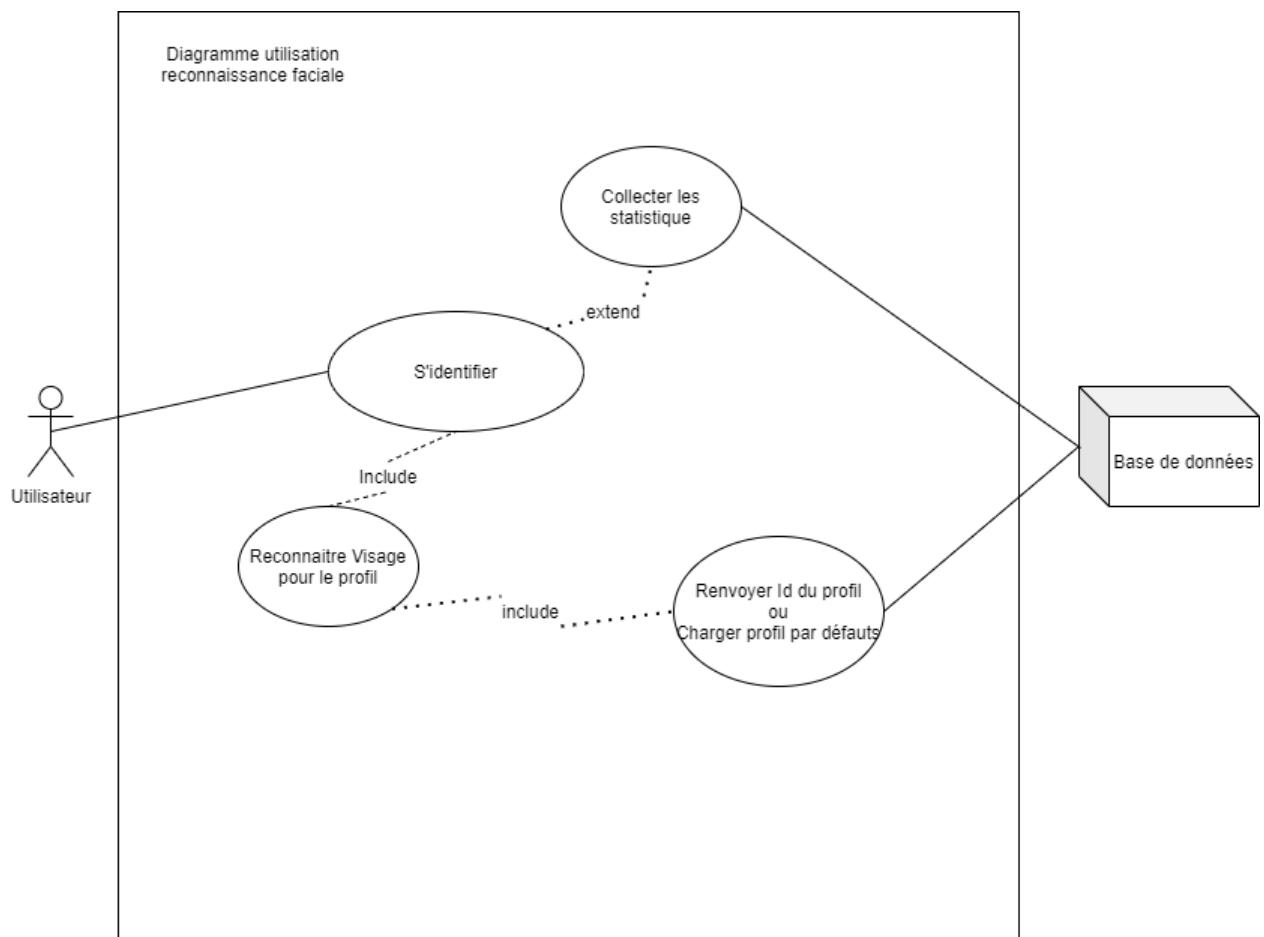
-Le HOG fonctionne sur un principe de points de clés (des points de reconnaissance comme la position et forme de certains éléments du visage) qui sont déjà définis par l'algorithme, le traitement de l'image envoyé est donc une analyse entre les points clés prédefinis et ceux de l'image. L'avantage de ce protocole est sa rapidité d'exécution et le fait qu'une fois que l'on a le fichier avec les points clé de l'image encodée il suffit de les comparer avec les images captées par la caméra.

-Le CNN lui fonctionne sur un principe de réseaux de neurones convolutionnel, ce qui signifie que sa reconnaissance va se baser sur un découpage de l'image en différentes parties, chaque parties étant traitées avec des points clé que l'algorithme définit lui-même. Ce traitement est fait jusqu'à obtention d'un taux de reconnaissance élevé (l'algorithme est sûr de son résultat). Une fois le taux obtenu on lui donne à analyser une nouvelle image de la même personne et il va re traiter cette image de la même manière que la précédente mais cette fois ci en se basant sur les deux images pour faire un croisement des points de reconnaissances, et ainsi de suite, on entraîne la reconnaissance jusqu'à ce qu'elle soit capable d'identifier la personne peu importe la photo. Ce protocole a pour avantages d'être extrêmement sûr et fiable mais est très lourd en termes de calcul, de traitement et de mémoire occupée.

Dans un souci d'optimisation de la charge de calcul et du taux de sécurité de la reconnaissance qui n'a pas à être extrêmement élevé le protocole HOG a été retenue pour le traitement de la reconnaissance faciale.

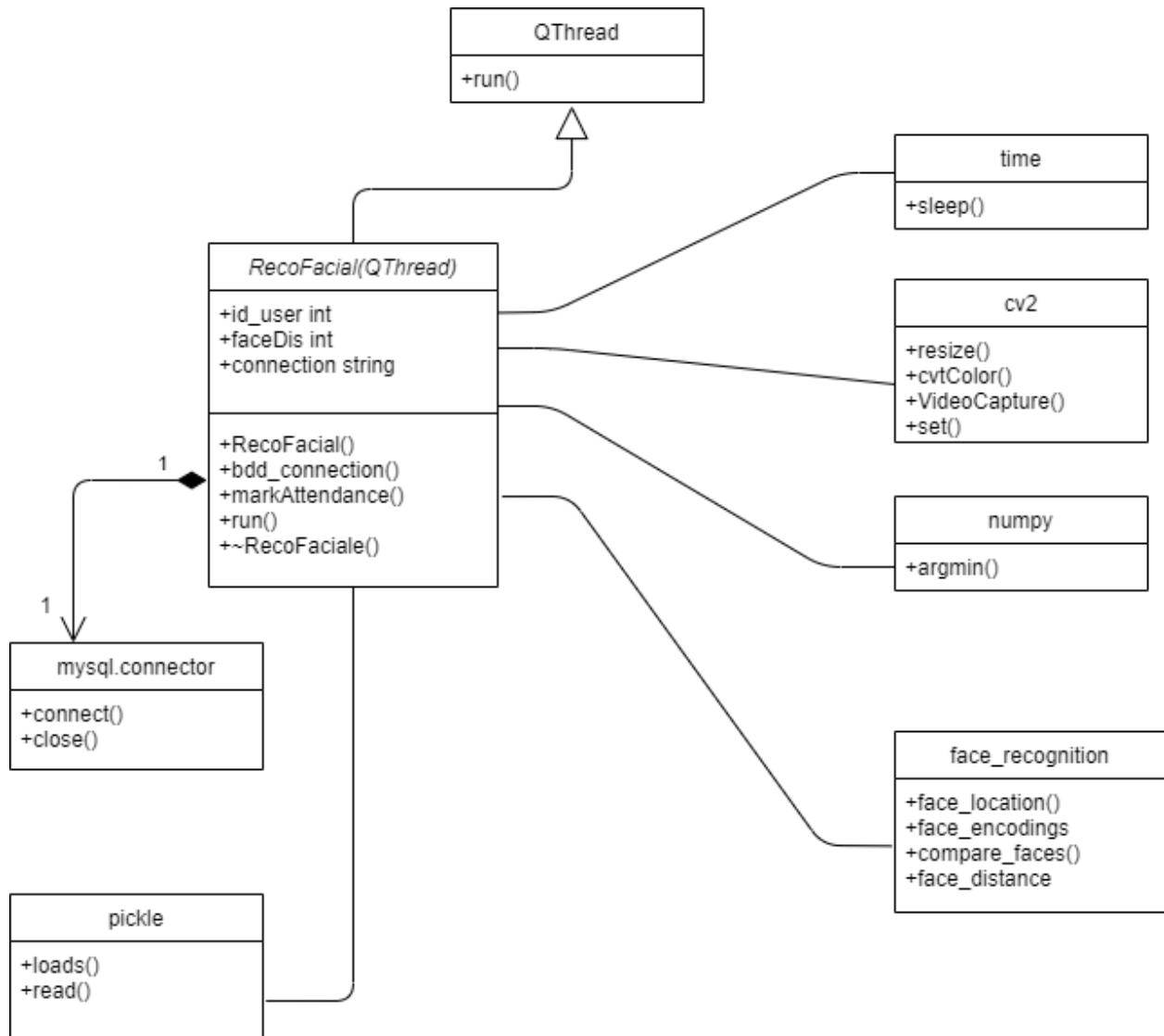
### III- Présentation des diagrammes et du code

#### a - Diagramme des cas d'utilisations



Mon diagramme des cas d'utilisations est simple, l'utilisateur veut s'identifier pour cela il faut que son visage soit reconnu par la reconnaissance faciale. Ceci inclut de renvoyer l'id correspondant au profil ou si l'utilisateur n'est pas reconnu de renvoyer l'id du profil par défaut.

## b - Diagramme de classe de la reconnaissance faciale



Dans ce diagramme de classe nous pouvons voir les différents attributs et méthodes qui composent ma classe, ainsi que toutes les librairies que j'ai dû utiliser (pour les détails du code référez vous à l'annexe). Ici j'explique les grandes lignes des fonctions et certaines méthodes.

Dans ma classe “`RecoFacial(QThred)`” on retrouve :

Le constructeur et destructeur de la classe.

Nous avons également “`bdd_connection()`” qui nous sert à nous connecter à la base de données avec les bons arguments (d'adresse ip, de nom de database, de nom d'utilisateur et de mot de passe). Qui facilite et allège le code.

Nous possédons également une fonction "mark Attendance()" qui sert à enregistrer les statistiques dans la base de données, statistiques qui consiste en une date, heure et un id correspondant.

Enfin, j'ai créé une fonction "run()" qui contient le code même de la reconnaissance faciale, qui, pour faire simple va récupérer les images de webcam, les encoder (voir paragraphe explicatif sur les encodages partie II - b Choix technologiques) , puis nous comparons ces encodages à ceux que l'on a qui réfère aux images de la base de données suite à cette comparaison un coefficient de ressemblance est émis et en fonction de ce coefficient le programme renvoie l'id correspondant à la personne reconnue. Si personne n'est reconnu on renvoie l'id du profil par défaut.

Je vais détailler maintenant quelques fonctions importantes des librairies que j'ai utilisé :

Dans la bibliothèque OpenCV :

-cvtColor() : cette fonction permet de convertir d'un mode d'encodage de couleur à un autre par exemple passer du BGR au RGB.

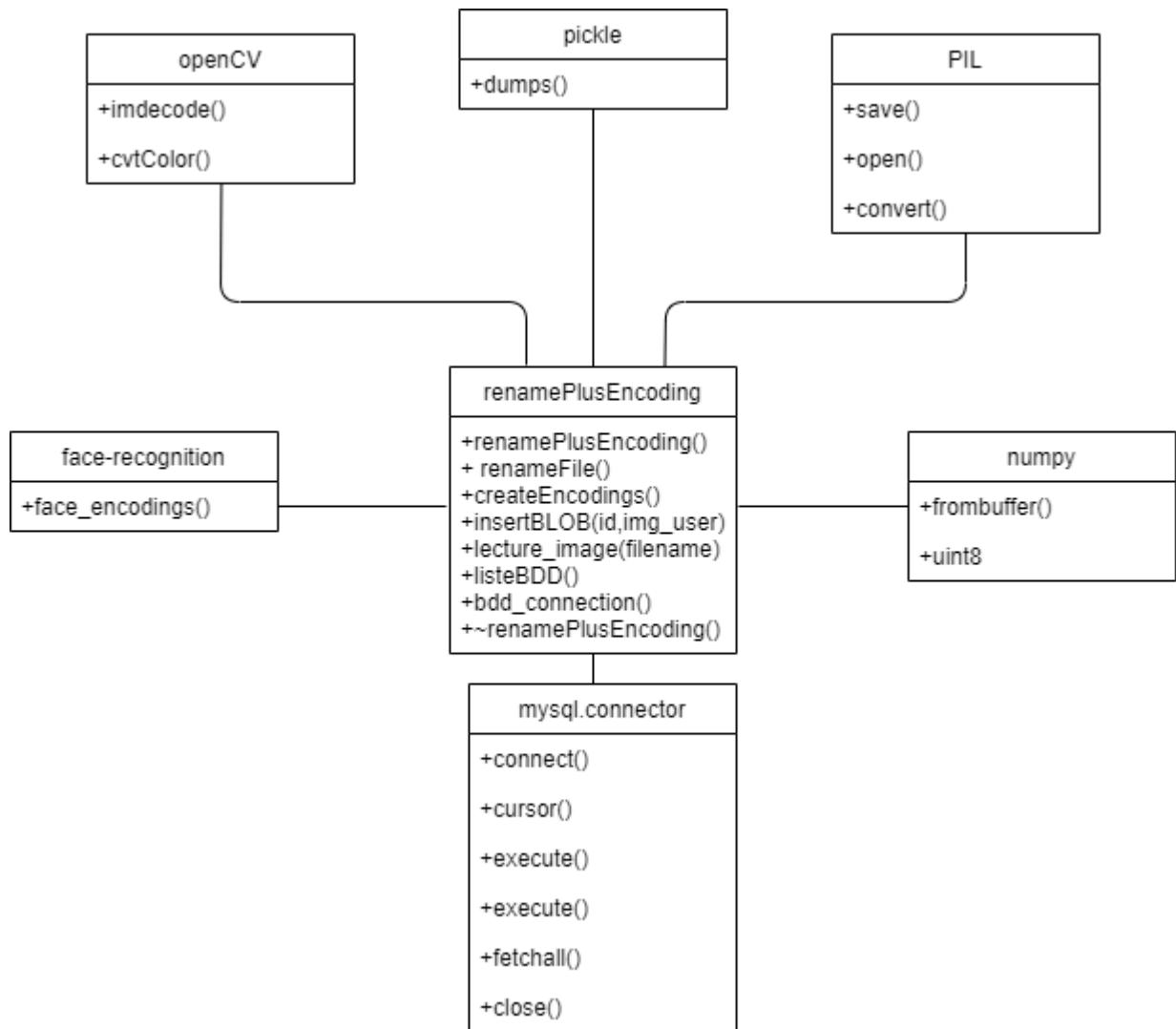
-set() : permet de fixer la résolution et/ou le nombre d'images par secondes capturées par la caméra.

Dans la bibliothèque face\_recognition :

-face\_location : permet de repérer le visage dans l'image capturée.

-compare\_faces(): permet de comparer les encodages des images pour trouver les similitudes.

### c - Diagrammes de la classe RenamePlusEncoding



Dans ce diagramme de classe nous pouvons voir les différents attributs et méthodes qui composent ma classe, ainsi que toutes les librairies que j'ai dû utiliser (pour les détails du code référez vous à l'annexe). Ici j'explique les grandes lignes des fonctions et certaines méthodes.

Dans ma classe "renamePlusEncoding" on retrouve le constructeur et destructeur de la classe. Ainsi que les méthodes suivantes :

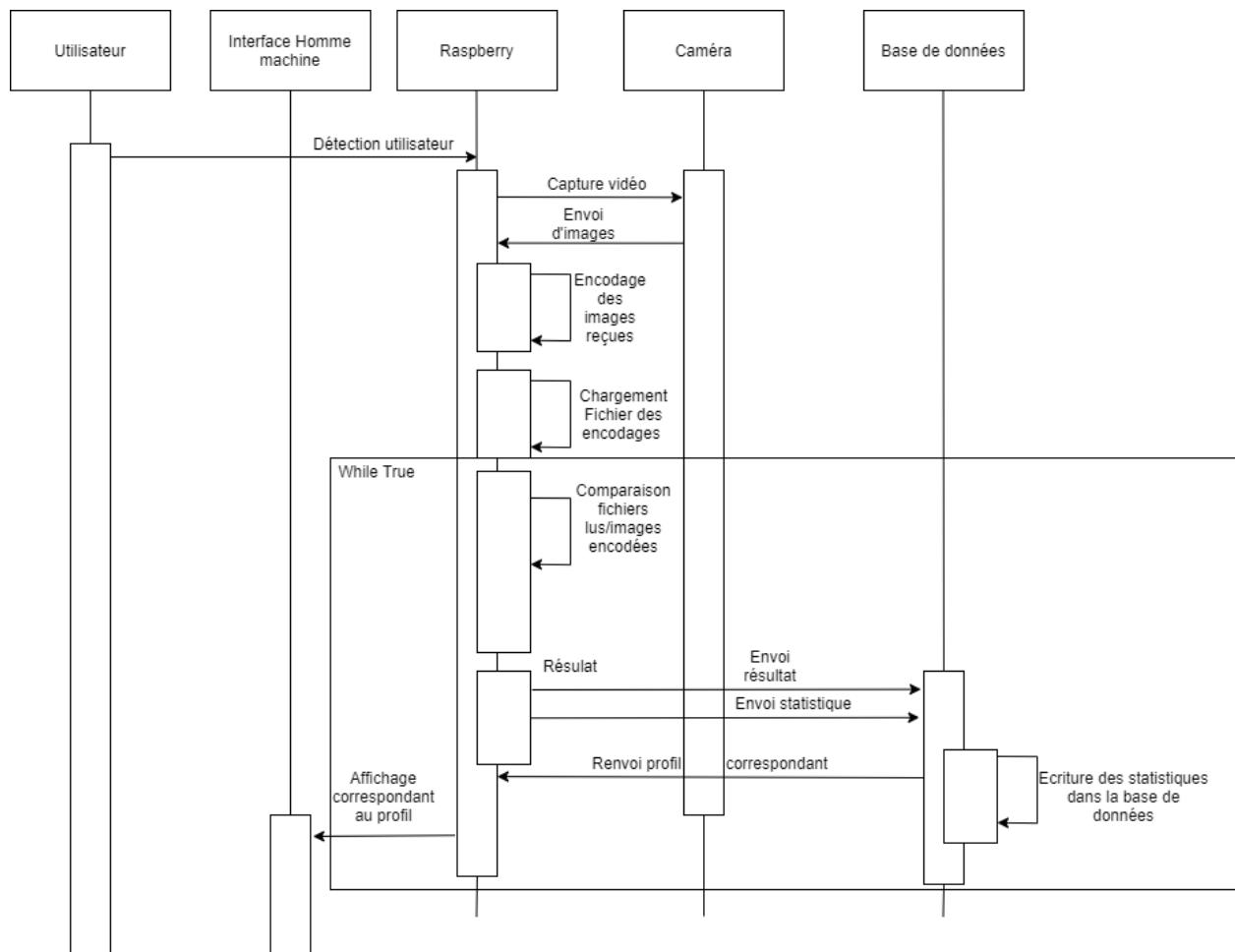
-`renameFile()`: Il s'agit de la méthode la plus importante de ce code, en effet elle appelle toutes les autres méthodes présentes pour pouvoir modifier une image et l'insérer dans la base de données (le détail du fonctionnement se trouve sous le diagramme de séquence de ce programme).

-createEncoding(): Cette fonction comme son nom l'indique va créer le fichier des images de la base de données encodées pour pouvoir les utiliser par la reconnaissance faciale. Pour générer ce fichier j'utilise la bibliothèque pickle qui permet de créer des fichiers en .pickle réutilisable en lecture pour le programme.

-insertBLOB(id, img\_user) : Cette fonction permet d'insérer une image dans la base de données à l'id indiqué, pour insérer une image dans une base de données il faut la convertir en binaire puis l'insérer dans un type "binary" ce que permet de faire la méthode lecture\_image(filename). Je récupère donc cette conversion et l'insère dans la BDD avec une requête SQL.

-listeBDD(): cette fonction permet de récupérer les éléments [0] et [1] pour récupérer l'id et l'encodage de l'image qui y correspond.

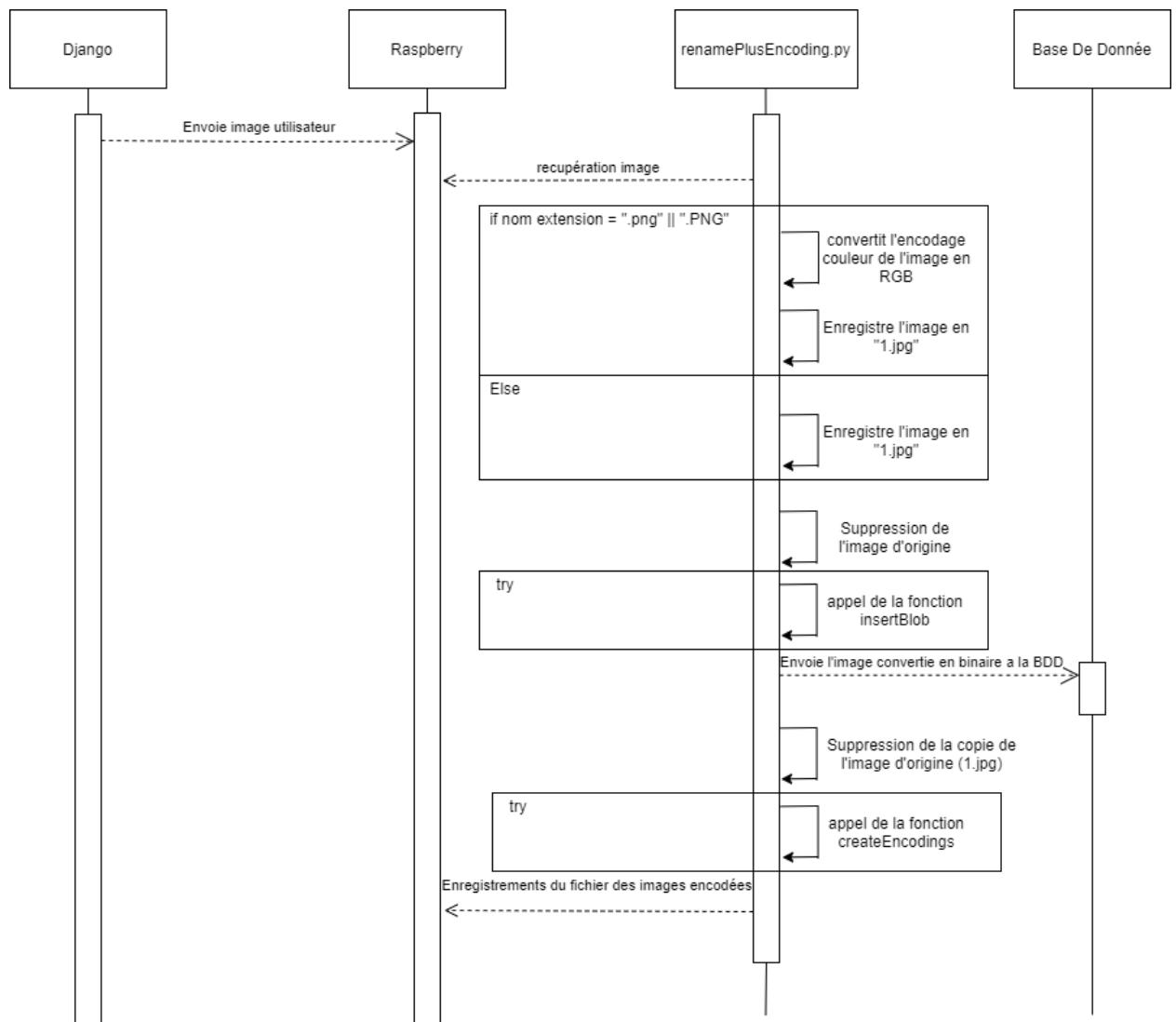
#### d - Diagramme de séquence de la reconnaissance faciale



Ce diagramme décrit le fonctionnement de la reconnaissance faciale. L'utilisateur arrive devant le miroir et est détecté par les capteurs ultrason (mis en place par l'étudiant du contrat Électronique et Communication), puis la capture vidéo se lance. Les images récupérées sont traitées et encodées (voir paragraphe explicatif sur les encodages dans la partie II - b choix technologique). Une fois encodés, le programme

charge le fichier des images de la Base De Données encodé généré par le programme renamePlusEncoding (qui sera détaillé dans la prochaine sous partie) puis il va comparer les points clés des images de la base de donnée et ceux des images traiter avec la caméra. On envoie le résultat a la base de donnée, soit l'id correspondant à la personne détectée soit l'id du profil par défaut, le programme envoie également les statistiques. La base de donnée envoi le profil correspondant à l'id envoyé puis l'interface homme machine (voir contrat étudiant IR 1) affiche les informations du profil.

#### e - Diagramme de séquence de renamePlusEncoding

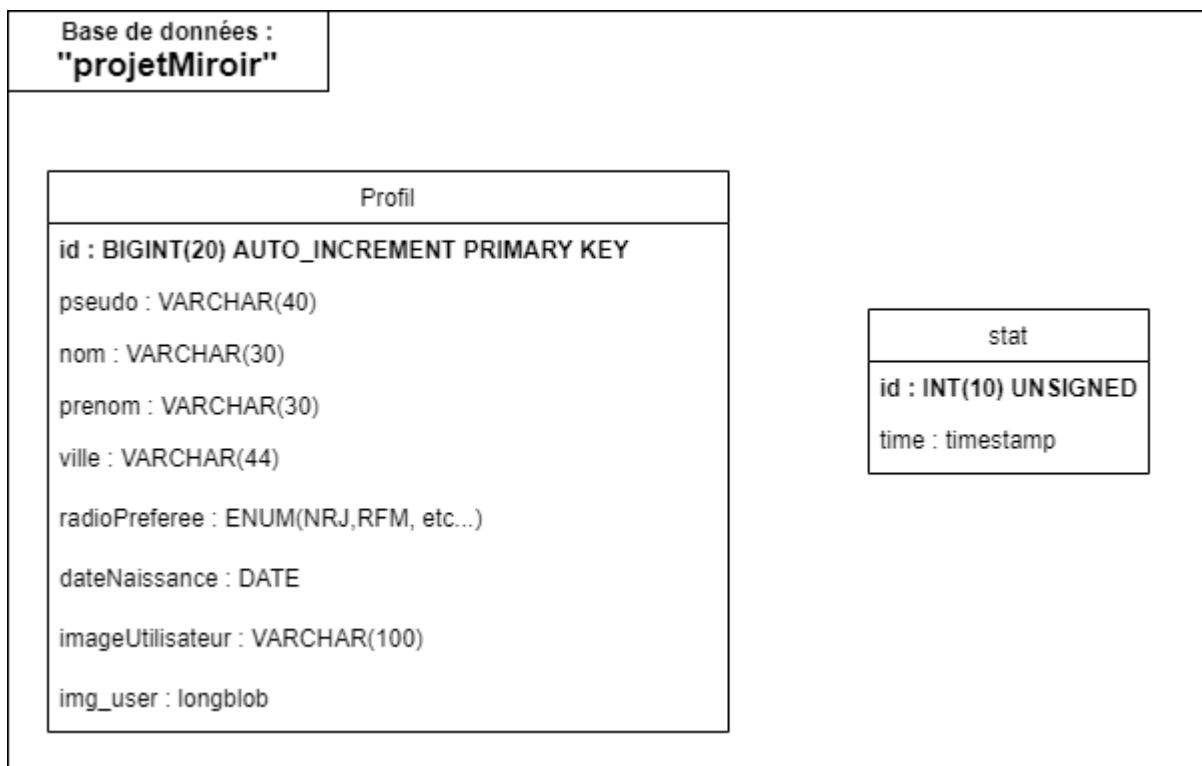


Le fonctionnement du programme renamePlusEncoding est expliqué par le diagramme de séquence et est détaillé ci après :

L'utilisateur entre l'image pour la reconnaissance faciale lors de création de son profil (voir contrat de l'étudiant IR 2) le programme va chercher l'extension du fichier si l'extension correspond à .PNG ou .png il va venir convertir l'encodage couleur de BGR à RGB pour pouvoir l'enregistrer en .jpg on enregistre une copie sous le nom "1.jpg" (il est nécessaire pour l'encodage que les images soient au format jpg pour l'encodage couleur). Si l'image est déjà en .jpg on enregistre juste une copie de l'image au nom "1.jpg". La copie et le changement de nom de l'image est nécessaire car pour utiliser la méthode insertBlob() il faut donner en argument le chemin et le nom de l'image que l'on souhaite insérer il est donc nécessaire que celui-ci soit fixe.

Une fois la copie faite, le programme supprime l'image d'origine, il appelle la fonction insetrBLOB(id, img\_user) (détailé dans la partie III - c), qui va venir insérer l'image dans la base de données au format binaire. Dans un souci de conflit, le programme va supprimer la copie de l'image. Une fois fait il appelle la méthode createEncoding (détailé partie III - c), le fichier des encodages est donc enregistré à l'issu de cette méthode.

#### f - Diagramme de Base De Données



La base de données du projet possède deux tables que nous utiliserons, la table Profil (liée au contrat 2.4.3), composée des éléments d'un profil comme un nom, prénom ou une radio préférée mais aussi une table "stats" permettant de récupérer les infos temporelles du temps passé devant son miroir lorsque votre profil est chargé. (Sachant que la table "stats" n'est qu'un simple prototype, celle-ci ne garde actuellement que l'heure du login et de la déconnexion).

## IV/ Problèmes rencontrés

Au cours de ce projet j'ai rencontré de multiples problèmes notamment au moment de l'intégration de mon code dans la carte raspberry Pi 4. En effet j'ai été confronté à un problème de vitesse de traitement des images et de faux résultat de la reconnaissance faciale. j'ai supposé que le problème pouvait venir du taux de reconnaissance qui était trop élevé et qui par conséquent donnait de faux résultat j'ai donc mené une batterie de test avec différentes valeur de taux 0.5 correspond a une corrélation de 50% entre les différents encodage et 0.90 correspondant à 10% de corrélation :

MatchIndex = 0,50							MatchIndex = 0,60							
Combinaison	Réussi ?	Latence	Combinaison	Réussi ?	Latence	Combinaison	Réussi ?	Latence	Combinaison	Réussi ?	Latence	Combinaison	Réussi ?	Latence
Défaut ► Alex	Oui	6,4	Défaut ► Luca	Oui	5,7	Défaut ► Maxence	Oui	7,8	Défaut ► Alex	Oui	7,2	Défaut ► Luca	Oui	4,1
Défaut ► Alex	Oui	7,3	Défaut ► Luca	Oui	8,9	Défaut ► Maxence	Oui	4	Défaut ► Alex	Oui	5	Défaut ► Luca	Oui	8
Défaut ► Alex	Oui	8,6	Défaut ► Luca	Oui	7,9	Défaut ► Maxence	Oui	3,5	Défaut ► Alex	Oui	3	Défaut ► Luca	Oui	6,5
Défaut ► Alex	Oui	5,3	Défaut ► Luca	Oui	4,1	Défaut ► Maxence	Oui	7,1	Défaut ► Alex	Oui	6,3	Défaut ► Luca	Oui	6,7
Défaut ► Alex	Oui	5,5	Défaut ► Luca	Oui	7	Défaut ► Maxence	Oui	7,8	Défaut ► Alex	Oui	4,9	Défaut ► Luca	Oui	5,2
Défaut ► Alex	Oui	7,2	Défaut ► Luca	Oui	4,7	Défaut ► Maxence	Oui	3,3	Défaut ► Alex	Oui	5,4	Défaut ► Luca	Oui	5,1
Défaut ► Alex	Oui	4,7	Défaut ► Luca	Oui	8,8	Défaut ► Maxence	Oui	4,8	Défaut ► Alex	Oui	4	Défaut ► Luca	Oui	3,4
Défaut ► Alex	Oui	8,5	Défaut ► Luca	Oui	6,8	Défaut ► Maxence	Oui	6,8	Défaut ► Alex	Oui	5,2	Défaut ► Luca	Oui	7,5
Défaut ► Alex	Oui	5,4	Défaut ► Luca	Oui	8,7	Défaut ► Maxence	Oui	4	Défaut ► Alex	Oui	3,9	Défaut ► Luca	Oui	7,6
Défaut ► Alex	Oui	4,2	Défaut ► Luca	Oui	5,6	Défaut ► Maxence	Oui	7,5	Défaut ► Alex	Oui	5,4	Défaut ► Luca	Oui	4,6
Alex ► Luca	Oui	8,9	Luca ► Alex	Oui	8,8	Maxence ► Alex	Oui	7	Alex ► Luca	Oui	3	Luca ► Alex	Oui	5,1
Alex ► Luca	Oui	8,3	Luca ► Alex	Oui	6,5	Maxence ► Alex	Oui	6,7	Alex ► Luca	Oui	6,7	Luca ► Alex	Oui	6,9
Alex ► Luca	Oui	4,1	Luca ► Alex	Oui	8,1	Maxence ► Alex	Oui	4,9	Alex ► Luca	Oui	6,7	Luca ► Alex	Oui	4
Alex ► Luca	Oui	8,4	Luca ► Alex	Oui	6,3	Maxence ► Alex	Oui	6,1	Alex ► Luca	Oui	6,9	Luca ► Alex	Oui	7,1
Alex ► Luca	Oui	8,7	Luca ► Alex	Oui	7,3	Maxence ► Alex	Oui	3,3	Alex ► Luca	Oui	3,9	Luca ► Alex	Oui	4,1
Alex ► Luca	Oui	6,8	Luca ► Alex	Oui	8,5	Maxence ► Alex	Oui	7,1	Alex ► Luca	Oui	5,8	Luca ► Alex	Oui	4,3
Alex ► Luca	Oui	4,3	Luca ► Alex	Oui	5	Maxence ► Alex	Oui	3,3	Alex ► Luca	Oui	4,1	Luca ► Alex	Oui	5,3
Alex ► Luca	Oui	4,2	Luca ► Alex	Oui	5,3	Maxence ► Alex	Oui	3,2	Alex ► Luca	Oui	4,1	Luca ► Alex	Oui	5,6
Alex ► Luca	Oui	7,2	Luca ► Alex	Oui	8,4	Maxence ► Alex	Oui	4,8	Alex ► Luca	Oui	6,2	Luca ► Alex	Oui	5,3
Alex ► Luca	Oui	6,7	Luca ► Alex	Oui	4,3	Maxence ► Alex	Oui	6,5	Alex ► Luca	Oui	4,5	Luca ► Alex	Oui	4,7
Alex ► Maxence	Oui	5,6	Luca ► Maxence	Oui	8,9	Maxence ► Luca	Oui	5,1	Alex ► Maxence	Oui	4,1	Luca ► Maxence	Oui	4,8
Alex ► Maxence	Oui	6	Luca ► Maxence	Oui	9	Maxence ► Luca	Oui	4,7	Alex ► Maxence	Oui	7,5	Luca ► Maxence	Oui	5,2
Alex ► Maxence	Oui	5,1	Luca ► Maxence	Oui	6,4	Maxence ► Luca	Oui	3,5	Alex ► Maxence	Oui	4,1	Luca ► Maxence	Oui	4,6
Alex ► Maxence	Oui	6,4	Luca ► Maxence	Oui	4,7	Maxence ► Luca	Oui	7,2	Alex ► Maxence	Oui	5,4	Luca ► Maxence	Oui	8
Alex ► Maxence	Oui	6,5	Luca ► Maxence	Oui	7,2	Maxence ► Luca	Oui	3	Alex ► Maxence	Oui	3,4	Luca ► Maxence	Oui	3,6
Alex ► Maxence	Oui	6,5	Luca ► Maxence	Oui	4,3	Maxence ► Luca	Oui	6,9	Alex ► Maxence	Oui	5,8	Luca ► Maxence	Oui	4,7
Alex ► Maxence	Oui	4,4	Luca ► Maxence	Oui	8,8	Maxence ► Luca	Oui	4,6	Alex ► Maxence	Oui	7	Luca ► Maxence	Oui	5,4
Alex ► Maxence	Oui	4,8	Luca ► Maxence	Oui	6,8	Maxence ► Luca	Oui	3,3	Alex ► Maxence	Oui	4,8	Luca ► Maxence	Oui	3,8
Alex ► Maxence	Oui	5,7	Luca ► Maxence	Oui	8,4	Maxence ► Luca	Oui	3,8	Alex ► Maxence	Oui	3,4	Luca ► Maxence	Oui	4,5
Alex ► Maxence	Oui	7,9	Luca ► Maxence	Oui	4,5	Maxence ► Luca	Oui	6,9	Alex ► Maxence	Oui	3,4	Luca ► Maxence	Oui	3,2

MatchIndex = 0,70							MatchIndex = 0,80							
Combinaison	Réussi ?	Latence	Combinaison	Réussi ?	Latence	Combinaison	Réussi ?	Latence	Combinaison	Réussi ?	Latence	Combinaison	Réussi ?	Latence
Défaut ► Alex	Oui	5,6	Défaut ► Luca	Oui	5,9	Défaut ► Maxence	Oui	6	Défaut ► Alex	Oui	8,4	Défaut ► Luca	Oui	4,4
Défaut ► Alex	Oui	6	Défaut ► Luca	Oui	8,4	Défaut ► Maxence	Oui	8,2	Défaut ► Alex	Oui	6,5	Défaut ► Luca	Oui	2,6
Défaut ► Alex	Oui	5,4	Défaut ► Luca	Non	6,4	Défaut ► Maxence	Oui	6,9	Défaut ► Alex	Oui	5,8	Défaut ► Luca	Oui	4,7
Défaut ► Alex	Oui	5,3	Défaut ► Luca	Oui	8,9	Défaut ► Maxence	Oui	4,6	Défaut ► Alex	Oui	7,2	Défaut ► Luca	Oui	2,6
Défaut ► Alex	Oui	5,3	Défaut ► Luca	Oui	8,8	Défaut ► Maxence	Oui	8,5	Défaut ► Alex	Oui	7,1	Défaut ► Luca	Oui	4,7
Défaut ► Alex	Oui	5,4	Défaut ► Luca	Oui	4,2	Défaut ► Maxence	Oui	7,4	Défaut ► Alex	Oui	7,7	Défaut ► Luca	Oui	2,6
Défaut ► Alex	Oui	5,3	Défaut ► Luca	Oui	5,3	Défaut ► Maxence	Oui	8,3	Défaut ► Alex	Oui	5,8	Défaut ► Luca	Oui	3,2
Défaut ► Alex	Oui	4,6	Défaut ► Luca	Oui	5,4	Défaut ► Maxence	Oui	8,1	Défaut ► Alex	Oui	6,5	Défaut ► Luca	Oui	4,9
Défaut ► Alex	Oui	4,8	Défaut ► Luca	Oui	6,4	Défaut ► Maxence	Non	5,4	Défaut ► Alex	Oui	7,4	Défaut ► Luca	Oui	3,1
Défaut ► Alex	Oui	5,1	Défaut ► Luca	Oui	6,4	Défaut ► Maxence	Oui	6,1	Défaut ► Alex	Oui	6,5	Défaut ► Luca	Oui	4,6
Alex ► Luca	Oui	6	Luca ► Alex	Oui	5,9	Maxence ► Alex	Oui	4,4	Alex ► Luca	Oui	8	Luca ► Alex	Oui	2,9
Alex ► Luca	Oui	4,4	Luca ► Alex	Oui	6,6	Maxence ► Alex	Oui	8,8	Alex ► Luca	Oui	5,6	Luca ► Alex	Oui	2,9
Alex ► Luca	Oui	5,7	Luca ► Alex	Oui	5,5	Maxence ► Alex	Oui	7,9	Alex ► Luca	Oui	5,6	Luca ► Alex	Oui	2,1
Alex ► Luca	Oui	6	Luca ► Alex	Oui	8,9	Maxence ► Alex	Oui	6,3	Alex ► Luca	Oui	7,6	Luca ► Alex	Oui	2,1
Alex ► Luca	Oui	5,5	Luca ► Alex	Oui	4	Maxence ► Alex	Oui	4,1	Alex ► Luca	Oui	7,3	Luca ► Alex	Oui	2,4
Alex ► Luca	Oui	5,9	Luca ► Alex	Oui	6	Maxence ► Alex	Oui	5,5	Alex ► Luca	Oui	8,3	Luca ► Alex	Oui	4,9
Alex ► Luca	Oui	6,5	Luca ► Alex	Oui	5,5	Maxence ► Alex	Oui	6,1	Alex ► Luca	Oui	8	Luca ► Alex	Oui	3,7
Alex ► Luca	Oui	6	Luca ► Alex	Oui	8,9	Maxence ► Alex	Oui	5,9	Alex ► Luca	Oui	6	Luca ► Alex	Oui	3,1
Alex ► Luca	Oui	3,6	Luca ► Alex	Oui	8,2	Maxence ► Alex	Non	8	Alex ► Luca	Oui	5,7	Luca ► Alex	Oui	4,9
Alex ► Luca	Oui	5,6	Luca ► Alex	Oui	6,5	Maxence ► Alex	Oui	7	Alex ► Luca	Oui	7,2	Luca ► Alex	Oui	2,1
Alex ► Maxence	Oui	5,2	Luca ► Maxence	Oui	7,6	Maxence ► Luca	Oui	7,4	Alex ► Maxence	Oui	8,8	Luca ► Maxence	Oui	9
Alex ► Maxence	Oui	6,4	Luca ► Maxence	Oui	7,5	Maxence ► Luca	Oui	6,7	Alex ► Maxence	Oui	6,5	Luca ► Maxence	Oui	5,7
Alex ► Maxence	Non	6,5	Luca ► Maxence	Oui	8,3	Maxence ► Luca	Oui	5,8	Alex ► Maxence	Non	6,4	Luca ► Maxence	Oui	2,8
Alex ► Maxence	Non	3	Luca ► Maxence	Oui	6,1	Maxence ► Luca	Oui	4,3	Alex ► Maxence	Oui	5,4	Luca ► Maxence	Oui	2,9
Alex ► Maxence	Oui	5,6	Luca ► Maxence	Oui	7,6	Maxence ► Luca	Oui	8,9	Alex ► Maxence	Oui	8	Luca ► Maxence	Oui	4,1
Alex ► Maxence	Oui	6,3	Luca ► Maxence	Oui	8,7	Maxence ► Luca	Oui	6,9	Alex ► Maxence	Oui	8,6	Luca ► Maxence	Oui	3,9
Alex ► Maxence	Oui	3,7	Luca ► Maxence	Oui	4,9	Maxence ► Luca	Oui	7,8	Alex ► Maxence	Oui	5,9	Luca ► Maxence	Oui	3,7
Alex ► Maxence	Oui	6	Luca ► Maxence	Oui	8,7	Maxence ► Luca	Oui	7,7	Alex ► Maxence	Oui	7,3	Luca ► Maxence	Oui	4,5
Alex ► Maxence	Oui	5,9	Luca ► Maxence	Oui	7,6	Maxence ► Luca	Oui	4,2	Alex ► Maxence	Oui	5,9	Luca ► Maxence	Oui	3,4
Alex ► Maxence	Oui	4,9	Luca ► Maxence	Oui	8,1	Maxence ► Luca	Oui	6,5	Alex ► Maxence	Oui	5,6	Luca ► Maxence	Oui	6,4

MatchIndex = 0,90								
Combinaison	Réussi ?	Latence	Combinaison	Réussi ?	Latence	Combinaison	Réussi ?	Latence
Défaut ► Alex	Oui	8,8	Défaut ► Luca	Oui	8,5	Défaut ► Maxence	Oui	4,5
Défaut ► Alex	Oui	4,9	Défaut ► Luca	Oui	7,8	Défaut ► Maxence	Oui	7,3
Défaut ► Alex	Oui	4,7	Défaut ► Luca	Oui	7	Défaut ► Maxence	Oui	6,5
Défaut ► Alex	Oui	8,8	Défaut ► Luca	Oui	5,9	Défaut ► Maxence	Oui	5,2
Défaut ► Alex	Oui	8,3	Défaut ► Luca	Oui	6,1	Défaut ► Maxence	Oui	5,9
Défaut ► Alex	Oui	7,6	Défaut ► Luca	Oui	7,9	Défaut ► Maxence	Oui	7,2
Défaut ► Alex	Oui	7,7	Défaut ► Luca	Oui	5,2	Défaut ► Maxence	Oui	6,7
Défaut ► Alex	Oui	7,2	Défaut ► Luca	Oui	8,4	Défaut ► Maxence	Oui	5,7
Défaut ► Alex	Oui	8,9	Défaut ► Luca	Oui	6,3	Défaut ► Maxence	Oui	5,5
Défaut ► Alex	Oui	4,9	Défaut ► Luca	Oui	7,1	Défaut ► Maxence	Oui	7
Alex ► Luca	Oui	7,3	Luca ► Alex	Oui	6	Maxence ► Alex	Oui	8,7
Alex ► Luca	Oui	7,4	Luca ► Alex	Oui	9	Maxence ► Alex	Oui	8
Alex ► Luca	Oui	4,7	Luca ► Alex	Oui	6,3	Maxence ► Alex	Oui	8,2
Alex ► Luca	Oui	4,6	Luca ► Alex	Oui	7,2	Maxence ► Alex	Non	8,3
Alex ► Luca	Oui	7,6	Luca ► Alex	Oui	6,9	Maxence ► Alex	Non	6,2
Alex ► Luca	Oui	8,6	Luca ► Alex	Oui	7,4	Maxence ► Alex	Oui	7,2
Alex ► Luca	Oui	8,5	Luca ► Alex	Oui	5,9	Maxence ► Alex	Non	7,2
Alex ► Luca	Oui	7,7	Luca ► Alex	Oui	6,7	Maxence ► Alex	Oui	8,6
Alex ► Luca	Oui	7,2	Luca ► Alex	Oui	7,5	Maxence ► Alex	Oui	7,2
Alex ► Luca	Oui	8,9	Luca ► Alex	Oui	7,8	Maxence ► Alex	Oui	5
Alex ► Maxence	Oui	5,1	Luca ► Maxence	Oui	8,2	Maxence ► Luca	Oui	5,5
Alex ► Maxence	Oui	8	Luca ► Maxence	Oui	4,8	Maxence ► Luca	Oui	6,7
Alex ► Maxence	Non	8,8	Luca ► Maxence	Oui	7,1	Maxence ► Luca	Oui	5,3
Alex ► Maxence	Oui	8,5	Luca ► Maxence	Oui	7,2	Maxence ► Luca	Oui	8,1
Alex ► Maxence	Non	8,7	Luca ► Maxence	Oui	5,3	Maxence ► Luca	Oui	6,7
Alex ► Maxence	Non	4,8	Luca ► Maxence	Oui	5,5	Maxence ► Luca	Oui	5,2
Alex ► Maxence	Oui	8,8	Luca ► Maxence	Oui	5,7	Maxence ► Luca	Oui	7,7
Alex ► Maxence	Oui	6,8	Luca ► Maxence	Oui	5,5	Maxence ► Luca	Oui	5,8
Alex ► Maxence	Oui	7,7	Luca ► Maxence	Oui	6,8	Maxence ► Luca	Oui	7,6
Alex ► Maxence	Oui	6,9	Luca ► Maxence	Oui	8,6	Maxence ► Luca	Oui	8

Après ces tests je me suis rendu compte que le problème ne venait pas de la.

La caméra de la raspberry prend 30 images par secondes avec une résolution de 2560 par 3240 pixels ce qui est extrêmement lourd puisque le programme de reconnaissance faciale traite chaque images prise par la caméra et la raspberry n'a pas les ressources nécessaire pour un tel traitement. J'ai donc baissé la résolution et les images traités par secondes avec ces lignes de code là :

```
cap.set(cv2.CAP_PROP_FPS, 5)
cap.set(3, 640)
cap.set(4, 480)
# ici on paramètre la camera à 5 images prises/secondes et avec une résolution de 640px par 480px
```

Ici on ajuste les dimensions à 640 par 480 pixels et 5 images secondes et cette modification a réglé tous les problèmes de latence.

Un autre problème rencontré est que la carte Raspberry Pi 4 chauffait beaucoup en effet sa température d'utilisation optimale est d'environ 40° or elle montait à plus de 56° nous avons donc installé des dissipateur thermique pour le processeur et les autres puces en surface de la carte. Nous avons également ajouté un ventilateur de 40mm pour assurer un refroidissement constant, la température atteinte est de 35° et de manière stable.

## V - Remerciement

Je tiens à remercier sincèrement Monsieur F.Duchiron, et Monsieur P.Dubois qui, en tant que professeur encadrant les projets, se sont montrés toujours à l'écoute et très disponibles afin de répondre à mes questions et pour m'aider à résoudres des problèmes tout au long de la réalisation de ce projet.

Enfin, je n'oublie pas de remercier mes camarades et membres de mon groupe de projet, Maxence Laporte, Luca Grandon et Adrien Hupperts qui ont fait un excellent travail au cours de ce projet avec moi et sans qui, je ne me serais pas dépassé afin de réaliser à bien ce projet.

## VI - Conclusion et évolution

Ce projet a été très bénéfique pour ma personne que ce soit du point de vue compétences ainsi que sur le plan de l'humain. En effet travailler en groupe m'a beaucoup appris.

Pour la réalisation de mon contrat je suis parvenu à faire tout ce qui m'était demandé ainsi qu'à l'optimiser.

Comme futur amélioration de mon contrat je pourrais ajouter une partie pour traiter les statistiques émises, ou encore passer sur un encodage et traitement des images sur CNN pour avoir une reconnaissance plus fiable et robuste. Ceci entraînerait une amélioration nécessaire de la carte utilisée pour qu'elle soit plus puissante pour pouvoir effectuer cet encodage

## Conclusion Générale & amélioration possible :

Notre groupe tient tout d'abord à remercier nos professeurs Monsieur Duchiron, Monsieur Dubois, qui se sont toujours montrés à l'écoute et très disponibles tout au long de la réalisation de notre projet.

Nous les remercions aussi pour leurs aides, leurs conseils et leurs cours nous ayant permis d'évoluer et de faire avancer grandement le projet.

Pour conclure ce projet, la plupart des fonctionnalités du Miroir Connecté sont opérationnels tels que visualiser l'heure, la date, la radio et la musique jouée, le chargement de profil via la reconnaissance facial (lien contrat étudiant 4 IR), les commandes vocales (lien contrat étudiant 3 IR), le minuteur (avec une valeur fixe), les réponses vocales avec des valeurs fiables, la météo et enfin, l'affichage général est bien organisé par rapport aux widgets présents mais aussi sont opérationnelles telles que créer des profils, les modifier, supprimer de l'interface administrateur.

Certaines fonctionnalités ne sont pas totalement opérationnelles telles que nous l'imaginons. Avec le temps qu'il nous reste, nous allons peaufiner le projet, tout d'abord les commandes vocales afin que l'utilisateur puisse choisir la valeur du minuteur. Ensuite nous intégrerons les programmes des capteurs afin d'avoir accès aux valeurs des capteurs et de pouvoir contrôler la lumière du miroir grâce aux commandes vocales. Mais aussi travailler si le temps nous le permet sur l'esthétique général du projet, autant en hardware qu'en software.

Ce projet nous a beaucoup appris, à avoir un point de vue professionnel sur notre avenir, ainsi qu'une autonomie et une gestion du travail d'équipe. Cela nous a aussi permis d'apprendre à imaginer, préparer et concevoir un produit de A à Z via sa création autant matériel que logiciel. Ses nouvelles compétences que nous avons pu obtenir au fur et à mesure du projet seront toujours, se montrer bénéfiques dans nos projets futurs. Comme futur amélioration du projet, il serait possible d'ajouter une partie pour traiter les statistiques émises, ou encore passer sur un encodage et traitement des images sur CNN pour avoir une reconnaissance plus fiable et robuste. La sélection de clefs USB afin d'obtenir des musiques sauvegardé sur celle-ci afin de pouvoir les jouer ou encore de pouvoir connecter un agenda relié à un compte Google dans l'IHM afin d'avoir accès à son emploi du temps. La gestion de la luminosité des bandes leds permettant l'éclairage externe du miroir aurait pu être un plus à ajouter afin de faire varier la luminosité de la pièce en fonction



de la luminosité de la pièce ou encore une mise en veille de l'appareil lorsqu'il n'est pas utilisé par une personne, permettant de consommer moins d'énergie.

Ajouter une « Snowboy » à la reconnaissance vocale pourrait permettre une plus grande facilité et autonomie à cette partie, qui pourraut éviter beaucoup de problèmes liés aux bruits ambients.

## Annexes :

### Partie étudiant 1 EC (HUPPERT Adrien) :

Programme Capteur

```
import smbus
import time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)
bus = smbus.SMBus (1)

def Temperature():
    adresse=0x48
    entree=0x40
    bus.write_byte(adresse, entree)
    bus.read_byte(adresse)
    temp= bus.read_byte(adresse)
    print("Température: %1.2f °C" %(-17.9*(temp*5/255)+77.5))
    return Temperature

def Humidite():
    adresse=0x48
    entree=0x42
    bus.write_byte(adresse, entree)
    bus.read_byte(adresse)
    bus.read_byte(adresse)
    V= bus.read_byte(adresse)
    print("Humidite: %1.2f pourcentage" %(32.2*(V*5/255)-24.5))
    time.sleep(2)
    return Humidite

def Distance():
    GPIO_TRIGGER = 16
    GPIO_ECHO = 18

    GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
    GPIO.setup(GPIO_ECHO, GPIO.IN)
    GPIO.output(GPIO_TRIGGER, True)
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)

    StartTime = time.time()
    StopTime = time.time()
```

```
while GPIO.input(GPIO_ECHO) == 0:  
    StartTime = time.time()  
  
while GPIO.input(GPIO_ECHO) == 1:  
    StopTime = time.time()  
  
T = StopTime - StartTime  
dist= round((T*34300) / 2,0)  
print ("Mesure Distance = %.1f cm" % dist)  
time.sleep(1)  
  
return Distance  
  
def destroy():  
    adresse=0x48  
    bus.write_byte(adresse, 0x40)  
    bus.read_byte_data(adresse,0x41)  
  
while (1):  
    Humidite()  
    time.sleep(2)  
    Temperature()  
    time.sleep(2)  
    #destroy()  
    # dist = Distance()
```

## Partie étudiant 2 IR (LAPORTE Maxence):

## ApplicationIHM

```
from PyQt5 import QtWidgets
from IHM import Ui_MainWindow
from PyQt5.QtCore import QTimer
from widgetMeteo import widgetMeteo
from widgetRadio import widgetRadio
from widgetHorloge import widgetHorloge
from widgetCapteur import widgetCapteur
from controleSon import controleSon
from widgetMusiqueUSB import widgetMusiqueUSB
from modeAveugle import modeAveugle
from RecoFacial import RecoFacial
from recoV import RecoVocale
from user import user
import sys

class ApplicationIHM(QtWidgets.QMainWindow):
    def __init__(self):
        super(ApplicationIHM, self).__init__()

        """v Instantiation des l'objets v"""
        self.reco = RecoFacial()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self) # Mise en place de l'IHM
        self.profil = user()
        self.meteo = widgetMeteo(self.ui)
        self.radio = widgetRadio(self.ui)
        self.controleSon = controleSon()
        self.capteur = widgetCapteur(self.ui)
        self.horloge = widgetHorloge(self.ui)
        self.musique = widgetMusiqueUSB(self.ui)
        self.repAudio = modeAveugle()
        self.recoV = RecoVocale()

        """v Lancement des threads v"""
        self.reco.start() # Lancement du Thread pour la reconnaissance
Faciale
        self.recoV.start() # Lancement du Thread pour la reconnaissance
Vocale

        """v Premier profil chargé v"""
        self.profil.firstCharge(self.ui) # Lancement du profil par
défaut au lancement du miroir
```

```

    """v Timer v"""
    self.horloge.timer = QTimer()
    self.horloge.timer.timeout.connect(self.horloge.initTimer)
    self.horloge.timer.start(1000)
    self.capteur.timer = QTimer()
    self.capteur.timer.timeout.connect(self.capteur.receiveValue)
    self.capteur.timer.start(10000)
    self.horloge.timer2 = QTimer()
    self.horloge.timer2.timeout.connect(self.horloge.showtime2)
    self.horloge.timer2.start(1000)

    """v Signaux v"""
    self.profil.meteoActualised.connect(lambda:
self.meteo.infoMeteo(self.profil.ville))
        self.reco.userChanged.connect(lambda :
self.profil.chargeInfo(self.reco.id_pic, self.ui))
        self.recoV.radio_change.connect(lambda :
self.controleSon.radioSwitch(self.musique,self.radio,
self.recoV.nomRadio))
            self.recoV.radio_stop.connect(self.radio.radio_stop)
            self.recoV.radio_play.connect(lambda:
self.controleSon.radioFirst(self.musique,self.radio,
self.profil.radioPref))
                self.horloge.timerFinished.connect(lambda:
self.controleSon.minuteurSonne(self.musique, self.radio))
                self.horloge.sonTimerFinished.connect(lambda:
self.controleSon.finMinuteurSonne(self.musique, self.radio))
                    self.repAudio.talkStarted.connect(lambda:
self.controleSon.minuteurSonne(self.musique, self.radio))
                    self.repAudio.talkFinished.connect(lambda:
self.controleSon.finMinuteurSonne(self.musique, self.radio))
                        self.recoV.getHeure.connect(lambda :
self.repAudio.lireHeure(self.horloge.currentHeure,self.horloge.currentMi
nute))
                            self.recoV.getDate.connect(lambda:
self.repAudio.lireDate(self.horloge.currentDate))
                            self.recoV.getMusique.connect(lambda:
self.repAudio.lireMusique(self.musique.musiqueJoue,self.musique))
                                self.recoV.getRadio.connect(lambda:
self.repAudio.lireRadio(self.radio.lastRadio, self.radio))
                                self.recoV.commandeUnknow.connect(lambda:
self.repAudio.lireUnknow())
                                    self.recoV.getMeteo.connect(lambda:
self.repAudio.lireMeteo(self.meteo.nomVille, self.meteo.phrasePerso,

```

```

        self.meteo.temp_act_str, self.meteo.wind_Gust_str))
            self.recoV.getProfil.connect(lambda:
self.repAudio.lireProfil(self.profil.pseudoCharge))
            self.recoV.getCapteur.connect(lambda:
self.repAudio.lireCapteur(self.capteur.temp_loc, self.capteur.humi_loc))
            self.recoV.OnLumiere.connect(lambda: self.capteur.turnOnLight)
            self.recoV.OffLumiere.connect(lambda: self.capteur.turnOffLight)
            self.recoV.setValue.connect(self.horloge.getSeconds)
            self.recoV.startChrono.connect(self.horloge.start_action)
            self.recoV.stopChrono.connect(self.horloge.pause_action)
            self.recoV.resetChrono.connect(self.horloge.reset_action)
            self.recoV.musique_play.connect(lambda:
self.controleSon.musiqueFirst(self.musique, self.radio))
            self.recoV.musique_stop.connect(self.musique.musique_stop)
            self.recoV.musique_sup.connect(lambda:
self.controleSon.musiqueSup(self.musique, self.radio))
            self.recoV.musique_inf.connect(lambda:
self.controleSon.musiqueInf(self.musique, self.radio))

self.musique.playlist.currentMediaChanged.connect(self.musique.songChanged)

```

```

def main():
    app = QtWidgets.QApplication(sys.argv)
    application = ApplicationIHM()
    application.show()
    sys.exit(app.exec_())

if __name__ == "__main__":
    main()

```

```

widgetMeteo :

import requests
import json
from PyQt5.QtGui import QPixmap
from PyQt5.QtCore import Qt

class widgetMeteo():
    def __init__(self, ui):
        self.ui = ui
        """
        v Police v"""
        self.police = 36
        """
        Requête HTTP"""

```

```

#self.pingOpenWeather(self, ville, init)
self.developerDict = {}
self.image_gust = QPixmap()
self.image_gust.load('Image_Meteo/Wind.png')
self.phrasePerso = None
self.temp_act_str = None
self.wind_Gust_str = None

def initMeteo(self):
    self.pingOpenWeather()

def infoMeteo(self, ville):
    self.nomVille = ville
    self.pingOpenWeather()

def pingOpenWeather(self):
    """Requête HTTP"""
    self.pays = 'fr'
    self.apiKey = '39f8c6363789395683394a6fe29ba54f'
    self.response =
    requests.get(f'https://api.openweathermap.org/data/2.5/weather?q={self.nomVille},{self.pays}&units=metric&appid={self.apiKey}') # Récupère le
    json du site météo
    # print('Ping site météo')
    self.verifRequete()

def verifRequete(self):
    """Vérifie que la requête HTTP marche"""
    if self.response.status_code == 200:
        #print('Success!')
        self.tradJsonPython()
    elif self.response.status_code == 404:
        print('Not Found.')

def tradJsonPython(self):
    """Traduit le fichier JSON que le site renvoie en DICT pour que
    Python puisse le traiter"""
    self.developerDict = dict(json.loads(self.response.text))
    #print('Traduction du JSON en dictionnaire')
    self.calculMeteo()

def calculMeteo(self):
    """Récupère la température actuelle et calcul la vitesse du vent
    en Km/h"""
    self.temp_act = self.developerDict["main"]["temp"]

```

```

        self.ui.val_act.setText("<p align=\"left\">" + f"<span style=\""
font-size:{self.police}pt;\">" + "<font color = \"#ffffff\>" +
str(self.temp_act) + "°C")
        temp_act_int = int(self.temp_act)
        self.temp_act_str = str(temp_act_int)
        self.wind_Gust = round(self.developerDict["wind"]["gust"]*3.214)
        self.ui.val_gust.setText("<p align=\"left\">" + f"<span style=\""
font-size:{self.police}pt;\">" + "<font color = \"#ffffff\>" +
str(self.wind_Gust) + " Km/h")
        wind_Gust_int = int(self.wind_Gust)
        self.wind_Gust_str = str(wind_Gust_int)
        self.ui.val_lieu.setText("<p align=\"left\">" + f"<span style=\""
font-size:{self.police}pt;\">" + "<font color = \"#ffffff\>" +
str(self.nomVille))
        #print('Calcule de la température et de la vitesse du vent')
        self.setWeather()

def setWeather(self):
    """Affiche le logo de la météo en fonction du temps"""
    self.image = QPixmap()
    # print('Selectionne l\'image')

    if self.developerDict["weather"][0]["main"] == 'Thunderstorm':
        self.image.load('Image_Meteo/Thunderstorm.png')
        self.phrasePerso = 'Le temps est Orageux'
    elif self.developerDict["weather"][0]["main"] == 'Drizzle':
        self.image.load('Image_Meteo/Drizzle.png')
        self.phrasePerso = 'Le temps est bruineux'
    elif self.developerDict["weather"][0]["main"] == 'Rain':
        self.image.load('Image_Meteo/Rain.png')
        self.phrasePerso = 'Le temps est pluvieux'
    elif self.developerDict["weather"][0]["main"] == 'Snow':
        self.image.load('Image_Meteo/Snow.png')
        self.phrasePerso = 'Le temps est neigeux'
    elif self.developerDict["weather"][0]["main"] == 'Clouds':
        self.image.load('Image_Meteo/Clouds.png')
        self.phrasePerso = 'Le temps est nuageux'
    elif self.developerDict["weather"][0]["main"] == 'Clear':
        self.image.load('Image_Meteo/Clear.png')
        self.phrasePerso = 'Le temps est dégagé'
    elif self.developerDict["weather"][0]["main"] == 'Mist':
        self.image.load('Image_Meteo/Mist.png')
        self.phrasePerso = 'Le temps est brumeux'
    elif self.developerDict["weather"][0]["main"] == 'Smoke':
        self.image.load('Image_Meteo/Smoke.png')

```

```

        self.phrasePerso = 'Le temps est brumeux'
    elif self.developerDict["weather"][0]["main"] == 'Haze':
        self.image.load('Image_Meteo/Haze.png')
        self.phrasePerso = 'Le temps est brumeux'
    elif self.developerDict["weather"][0]["main"] == 'Dust':
        self.image.load('Image_Meteo/Dust.png')
        self.phrasePerso = 'Le temps est poussiéreux'
    elif self.developerDict["weather"][0]["main"] == 'Fog':
        self.image.load('Image_Meteo/Fog.png')
        self.phrasePerso = 'Il y a du brouillard'
    elif self.developerDict["weather"][0]["main"] == 'Sand':
        self.image.load('Image_Meteo/Sand.png')
        self.phrasePerso = 'Tempête de sable'
    elif self.developerDict["weather"][0]["main"] == 'Ash':
        self.image.load('Image_Meteo/Ash.png')
        self.phrasePerso = 'Tempête de cendre'
    elif self.developerDict["weather"][0]["main"] == 'Squall':
        self.image.load('Image_Meteo/Squall.png')
        self.phrasePerso = 'Il a des bourrasques de vent'
    elif self.developerDict["weather"][0]["main"] == 'Tornado':
        self.image.load('Image_Meteo/Tornado.png')
        self.phrasePerso = 'Une tornade a été détectée'
        self.ui.val_image.setPixmap(self.image.scaled(80, 80,
Qt.KeepAspectRatio))
        self.ui.val_image_gust.setPixmap(self.image_gust.scaled(80, 80,
Qt.KeepAspectRatio))

```

widgetRadio :

```

from PyQt5.QtCore import QUrl
from PyQt5.QtMultimedia import QMediaPlayer, QMediaContent

class widgetRadio():
    def __init__(self,ui):
        self.ui=ui
        """v Police v"""
        self.police = 36
        self.lastRadio = None
        self.player = QMediaPlayer()
        self.ui.text_radio_musique.setText("<p align=\"center\">" +
f"<span style=\" font-size:{self.police}pt;\">" + "<font color = "
"\#ffffff\">" + "Radio en cours : ")
        self.dico_radio = {"NRJ":"
"http://scdn.nrjaudio.fm/adwz2/fr/30001/mp3_128.mp3?origine=fluxradios",

```

"France Info":  
 "http://icecast.radiofrance.fr/franceinfo-midfi.mp3",  
 "France Musique":  
 "http://direct.francemusique.fr/live/francemusique-midfi.mp3",  
 "Nostalgie":  
 "http://scdn.nrjaudio.fm/adwz2/fr/30601/mp3\_128.mp3?origine=fluxradios",  
 "Chérie FM":  
 "http://scdn.nrjaudio.fm/adwz2/fr/30201/mp3\_128.mp3?origine=fluxradios",  
 "Rire & Chansons":  
 "http://scdn.nrjaudio.fm/adwz2/fr/30401/mp3\_128.mp3?origine=fluxradios",  
 "RTL":  
 "http://icecast.rtl.fr/rtl-1-44-128?listen=webCwsBCggNCQgLDQUGBAcGBg",  
 "RTL2":  
 "http://icecast.rtl2.fr/rtl2-1-44-128?listen=webCwsBCggNCQgLDQUGBAcGBg",  
 "FunRadioFrance":  
 "http://icecast.funradio.fr/fun-1-44-128?listen=webCwsBCggNCQgLDQUGBAcGBg",  
 "Europe1":  
 "http://stream.europe1.fr/europe1.mp3",  
 "Virgin Radio":  
 "http://stream.virginradio.fr/virgin.mp3",  
 "RFM":  
 "https://ais-live.cloud-services.paris:8443/rfm.mp3",  
 "RMC": "http://chai5she.cdn.dvmr.fr/rmcinfo",  
 "BFM Business":  
 "http://chai5she.cdn.dvmr.fr/bfmbusiness",  
 "Skyrock":  
 "http://icecast.skyrock.net/s/natio\_mp3\_128k",  
 "RadioClassique":  
 "http://radioclassique.ice.infomaniak.ch/radioclassique-high.mp3",  
 # "FranceInfo":  
 "http://icecast.radiofrance.fr/franceinfo-hifi.aac",  
 "France Inter":  
 "http://icecast.radiofrance.fr/franceinter-hifi.aac",  
 "France Culture":  
 "http://icecast.radiofrance.fr/franceculture-hifi.aac",  
 "FIP Nationale":  
 "http://icecast.radiofrance.fr/fip-hifi.aac",  
 "Mouv'":  
 "http://direct.mouv.fr/live/mouv-midfi.mp3",  
 "Oui FM":  
 "http://target-ad-2.cdn.dvmr.fr/ouifm-high.mp3",  
 "Jazz Radio":  
 "http://jazzradio.ice.infomaniak.ch/jazzradio-high.mp3",  
 "M Radio":

```

"http://mfm.ice.infomaniak.ch/mfm-128.mp3",
        "France Bleu":
"http://direct.francebleu.fr/live/fb1071-midfi.mp3",
    }

#
def radio_play(self, radioPref):
    self.ui.val_radio_musique.setText("<p align=\"center\>" +
f"<span style=\" font-size:{self.police}pt;\>" + "<font color =
\"#ffffff\>" + radioPref)
    self.lastRadio = radioPref
    self.qurl = QUrl(self.dico_radio[radioPref])
    self.player.setMedia(QMediaContent(self.qurl))
    self.player.play()

def change_radio(self, nomRadio):
    self.ui.val_radio_musique.setText("<p align=\"center\>" +
f"<span style=\" font-size:{self.police}pt;\>" + "<font color =
\"#ffffff\>" + nomRadio)
    self.lastRadio = nomRadio
    self.qurl = QUrl(self.dico_radio[nomRadio])
    self.player.setMedia(QMediaContent(self.qurl))
    self.player.play()

def radio_stop(self):
    self.player.stop()
    self.ui.val_radio_musique.setText("<p align=\"center\>" +
f"<span style=\" font-size:{self.police}pt;\>" + "<font color =
\"#ffffff\>" + "")"

def diminuer_son(self):
    self.player.setVolume(30)

def monter_son(self):
    self.player.setVolume(100)

```

controleSon :

```

class controleSon():

def minuteurSonne(self,musique,radio):
    if radio.player.state():
        radio.diminuer_son()

```



```

    elif musique.player.state():
        musique.diminuer_son()

def finMinuteurSonne(self,musique,radio):
    if radio.player.state():
        radio.monter_son()
    elif musique.player.state():
        musique.monter_son()

def musiqueFirst(self,musique,radio):
    radio.radio_stop()
    musique.musique_play()

def musiqueSup(self,musique,radio):
    radio.radio_stop()
    musique.change_musique_sup()

def musiqueInf(self,musique,radio):
    radio.radio_stop()
    musique.change_musique_inf()

def radioFirst(self,musique,radio,radioPref):
    musique.musique_stop()
    radio.radio_play(radioPref)

def radioSwitch(self,musique,radio,nomRadio):
    musique.musique_stop()
    radio.change_radio(nomRadio)

```

modeAveugle :

```

from gtts import gTTS
import os
from PyQt5.QtCore import QUrl
from PyQt5.QtMultimedia import QMediaPlayer, QMediaContent
from PyQt5.QtCore import pyqtSignal, QObject

"""changer le chemin par '/home/pi/Documents/Raspberry/son.mp3'"""

class modeAveugle(QObject):
    talkStarted = pyqtSignal()
    talkFinished = pyqtSignal()
    def __init__(self,parent=None):
        QObject.__init__(self,parent)

```

```

    self.playerAveugle = QMediaPlayer()
    self.playerAveugle.stateChanged.connect(self.talkEnd)

def lireHeure(self,text1,text2):
    print("heuuuuure")
    os.remove("/home/pi/Documents/Raspberry/heure.mp3")
    mytext = 'Il est ' + text1 + ' Heure' + text2
    language = 'fr'
    textSpeech = gTTS(text=mytext, lang=language, slow=False)
    textSpeech.save("/home/pi/Documents/Raspberry/heure.mp3")
    self.talkHeure()

def talkHeure(self):

    self.playerAveugle.setMedia(QMediaContent(QUrl.fromLocalFile("/home/pi/D
ocuments/Raspberry/refresh.mp3")))

    self.playerAveugle.setMedia(QMediaContent(QUrl.fromLocalFile("/home/pi/D
ocuments/Raspberry/heure.mp3")))
    self.talkStarted.emit()
    self.playerAveugle.play()
    self.talkEnd()

def lireCapteur(self,text1,text2):
    print("heuuuuure")
    os.remove("/home/pi/Documents/Raspberry/capteur.mp3")
    mytext = 'Dans la pièce, il fait ' + str(text1) + ' degrès
celsius, et l\'humidité est de ' + str(text2) + ' pourcent'
    language = 'fr'
    textSpeech = gTTS(text=mytext, lang=language, slow=False)
    textSpeech.save("/home/pi/Documents/Raspberry/capteur.mp3")
    self.talkCapteur()

def talkCapteur(self):

    self.playerAveugle.setMedia(QMediaContent(QUrl.fromLocalFile("/home/pi/D
ocuments/Raspberry/refresh.mp3")))

    self.playerAveugle.setMedia(QMediaContent(QUrl.fromLocalFile("/home/pi/D
ocuments/Raspberry/capteur.mp3")))
    self.talkStarted.emit()
    self.playerAveugle.play()
    self.talkEnd()

def lireMusique(self,text1,musique):

```

```

os.remove("/home/pi/Documents/Raspberry/musique.mp3")
if musique.player.state():
    mytext = 'La musique joué est ' + text1
else:
    mytext = 'Il n\'y as aucune musique en cours de lecture'
language = 'fr'
textSpeech = gTTS(text=mytext, lang=language, slow=False)
textSpeech.save("/home/pi/Documents/Raspberry/musique.mp3")
self.talkMusique()

def talkMusique(self):

self.playerAveugle.setMedia(QMediaContent(QUrl.fromLocalFile("/home/pi/D
ocuments/Raspberry/refresh.mp3")))

self.playerAveugle.setMedia(QMediaContent(QUrl.fromLocalFile("/home/pi/D
ocuments/Raspberry/musique.mp3")))
    self.talkStarted.emit()
    self.playerAveugle.play()
    self.talkEnd()

def lireRadio(self,text1,radio):
os.remove("/home/pi/Documents/Raspberry/radio.mp3")
if radio.player.state():
    mytext = 'La radio en cours de lecture est ' + text1
else:
    mytext = 'Il n\'y as aucune radio en cours de lecture'
language = 'fr'
textSpeech = gTTS(text=mytext, lang=language, slow=False)
textSpeech.save("/home/pi/Documents/Raspberry/radio.mp3")
self.talkRadio()

def talkRadio(self):

self.playerAveugle.setMedia(QMediaContent(QUrl.fromLocalFile("/home/pi/D
ocuments/Raspberry/refresh.mp3")))

self.playerAveugle.setMedia(QMediaContent(QUrl.fromLocalFile("/home/pi/D
ocuments/Raspberry/radio.mp3")))
    self.talkStarted.emit()
    self.playerAveugle.play()
    self.talkEnd()

def lireUnknow(self):
os.remove("/home/pi/Documents/Raspberry/Unknow.mp3")

```

```

mytext = 'Commande non reconnue'
language = 'fr'
textSpeech = gTTS(text=mytext, lang=language, slow=False)
textSpeech.save("/home/pi/Documents/Raspberry/Unknow.mp3")
self.talkUnknow()

def talkUnknow(self):

    self.playerAveugle.setMedia(QMediaContent(QUrl.fromLocalFile("/home/pi/Documents/Raspberry/refresh.mp3")))

    self.playerAveugle.setMedia(QMediaContent(QUrl.fromLocalFile("/home/pi/Documents/Raspberry/Unknow.mp3")))
    self.talkStarted.emit()
    self.playerAveugle.play()
    self.talkEnd()

    def lireMeteo(self,text1,text2, text3, text4): #text1 = ville |
text2 = phrase perso temps | text3 = température | text4 = vitesse du
vent
        os.remove("/home/pi/Documents/Raspberry/meteo.mp3")
        mytext = 'Actuellement à ' + text1 + ',' + text2 + ', la
température est de ' + text3 + ' degrès celsius, la vitesse du vent est
de ' + text4 + ' kilomètre par heure'
        language = 'fr'
        textSpeech = gTTS(text=mytext, lang=language, slow=False)
        textSpeech.save("/home/pi/Documents/Raspberry/meteo.mp3")
        self.talkMeteo()

    def talkMeteo(self):

        self.playerAveugle.setMedia(QMediaContent(QUrl.fromLocalFile("/home/pi/Documents/Raspberry/refresh.mp3")))

        self.playerAveugle.setMedia(QMediaContent(QUrl.fromLocalFile("/home/pi/Documents/Raspberry/meteo.mp3")))
        self.talkStarted.emit()
        self.playerAveugle.play()
        self.talkEnd()

    def lireProfil(self,text1): #text1 = pseudo
        os.remove("/home/pi/Documents/Raspberry/profil.mp3")
        mytext = 'Le profil de ' + text1 + ' est actuellement chargé'
        language = 'fr'
        textSpeech = gTTS(text=mytext, lang=language, slow=False)

```

```

textSpeech.save("/home/pi/Documents/Raspberry/profil.mp3")
self.talkProfil()

def talkProfil(self):

    self.playerAveugle.setMedia(QMediaContent(QUrl.fromLocalFile("/home/pi/D
ocuments/Raspberry/refresh.mp3")))

    self.playerAveugle.setMedia(QMediaContent(QUrl.fromLocalFile("/home/pi/D
ocuments/Raspberry/profil.mp3")))
    self.talkStarted.emit()
    self.playerAveugle.play()
    self.talkEnd()

def lireDate(self, text1): #text1 = date
    os.remove("/home/pi/Documents/Raspberry/date.mp3")
    mytext = 'Nous sommes le ' + text1
    language = 'fr'
    textSpeech = gTTS(text=mytext, lang=language, slow=False)
    textSpeech.save("/home/pi/Documents/Raspberry/date.mp3")
    self.talkDate()

def talkDate(self):

    self.playerAveugle.setMedia(QMediaContent(QUrl.fromLocalFile("/home/pi/D
ocuments/Raspberry/refresh.mp3")))

    self.playerAveugle.setMedia(QMediaContent(QUrl.fromLocalFile("/home/pi/D
ocuments/Raspberry/date.mp3")))
    self.talkStarted.emit()
    self.playerAveugle.play()
    self.talkEnd()

def talkEnd(self):
    if self.playerAveugle.state() == 0:
        self.talkFinished.emit()

```

user :

```

import mysql.connector
from mysql.connector import Error
from PyQt5.QtCore import pyqtSignal, QObject

class user(QObject):

```

```

meteoActualised = pyqtSignal()
radioActualised = pyqtSignal()
def __init__(self):
    super().__init__()
    self.connection = None
    self.pseudoCharge = None

    def bdd_connection(self):
        self.connection = mysql.connector.connect(host='172.20.10.4',
database='projetMiroir', user='projet', password='miroir')

    def firstCharge(self, ui):
        self.ui = ui
        id = 1
        self.police = 30
        try:
            self.bdd_connection()
            self.cursor = self.connection.cursor(buffered=True)
            self.cursor.execute(f"SELECT pseudo FROM dummy WHERE id = {id};")
            pseudo = self.cursor.fetchone()
            self.pseudo = ''.join(pseudo)
            self.cursor.execute(f"SELECT ville FROM dummy WHERE id = {id};")
            ville = self.cursor.fetchone()
            self.ville = ''.join(ville)
            self.cursor.execute(f"SELECT radioP FROM dummy WHERE id = {id};")
            radioPref = self.cursor.fetchone()
            self.radioPref = ''.join(radioPref)
            #self.cursor.execute(f"SELECT radioAct FROM dummy WHERE id = {id};")
            #radioActiveLancement = self.cursor.fetchone()
            #self.radioActiveLancement = int(''.join(map(str, radioActiveLancement)))
            #self.cursor.execute(f"SELECT modeAveugle FROM dummy WHERE id = {id};")
            #modeAveugle = self.cursor.fetchone()
            #self.modeAveugle = int(''.join(map(str, modeAveugle)))
            self.connection.commit()

        except Error as error:
            print("Failed to insert statistic {}".format(error))

        finally:

```

```

        if self.connection.is_connected():
            self.cursor.close()
            self.connection.close()
        self.ui.helloUser.setText("<p align=\"center\>" + f"<span
style=\" font-size:{self.police}pt;\">" + "<font color = \"#ffffff\>" +
"Bonjour " + self.pseudo + ", votre profil est chargé")
        self.actualise()

def chargeInfo(self, id, ui):
    self.ui = ui
    self.police = 36
    try:
        self.bdd_connection()
        self.cursor = self.connection.cursor(buffered=True)
        self.cursor.execute(f"SELECT prenom FROM dummy WHERE
id_image = {id};")
        pseudo = self.cursor.fetchone()
        pseudoReconnu = ''.join(pseudo)
        self.cursor.execute(f"SELECT ville FROM dummy WHERE id_image
= {id};")
        ville = self.cursor.fetchone()
        self.ville = ''.join(ville)
        self.cursor.execute(f"SELECT radioP FROM dummy WHERE
id_image = {id};")
        radioPref = self.cursor.fetchone()
        self.radioPref = ''.join(radioPref)
        #self.cursor.execute(f"SELECT radioAct FROM dummy WHERE
id_image = {id};")
        #radioActiveLancement = self.cursor.fetchone()
        #self.radioActiveLancement = int(''.join(map(str,
radioActiveLancement)))
        #self.cursor.execute(f"SELECT modeAveugle FROM dummy WHERE
id_image = {id};")
        #modeAveugle = self.cursor.fetchone()
        #self.modeAveugle = int(''.join(map(str, modeAveugle)))
        self.connection.commit()

    except Error as error:
        print("Charge info :: ".format(error))

    finally:
        if self.connection.is_connected():
            self.cursor.close()
            self.connection.close()
        self.ui.helloUser.setText("<p align=\"center\>" + f"<span

```

```

style=" font-size:{self.police}pt;" + "<font color = \"#ffffff\>" +
"Bonjour " + pseudoReconnu + ", votre profil est chargé")
self.actualise()
self.pseudoCharge = pseudoReconnu

def actualise(self):
    self.meteoActualised.emit()
    self.radioActualised.emit()

widgetHorloge :

from PyQt5.QtCore import QTime, QDate, Qt, QUrl
from widgetRadio import widgetRadio
from widgetMusiqueUSB import widgetMusiqueUSB
from PyQt5.QtCore import pyqtSignal, QObject
from PyQt5.QtMultimedia import QMediaPlayer, QMediaContent

class widgetHorloge(QObject):
    timerFinished = pyqtSignal()
    sonTimerFinished = pyqtSignal()
    def __init__(self, ui):
        super().__init__()
        self.ui = ui
        #self.musique = widgetMusiqueUSB(self.ui)
        #self.radio = widgetRadio(self.ui)
        """v Police v"""
        self.police = 36
        self.ui.timer_value.setText("<p align=\"center\>" + f"<span"
style=" font-size:{self.police}pt;" + "<font color = \"#ffffff\>" +
"")
        self.player = QMediaPlayer()
        self.playerLien = 'Son_Miroir/TNIC.mp3'
        self.count = 0
        self.start = False
        self.initTimer()

    def initTimer():
        """Affiche la date et l'heure actuel"""
        self.ui.val_heure_act.setText("<p align=\"center\>" + f"<span"
style=" font-size:{self.police}pt;" + "<font color = \"#ffffff\>" +
self.showTime())
        self.ui.val_jour_act.setText("<p align=\"center\>" + f"<span"
style=" font-size:{self.police}pt;" + "<font color = \"#ffffff\>" +
self.currentDate)

```

```
def play_sound(self):
    print('Joue du son')

self.player.setMedia(QMediaContent(QUrl.fromLocalFile(self.playerLien)))
    self.player.play()

def get_seconds(self):
    self.start = False
    second = 3 # A definir avec la commande vocale
    self.count = second
    self.ui.timer_value.setText("<p align=\"center\">" + f"<span"
style=" font-size:{self.police}pt;" + ">" + "<font color = \"#ffffff\">" +
str(second))

def start_action(self):

    self.start = True
    if self.count == 0:
        self.start = False

def pause_action(self):

    self.start = False

def reset_action(self):

    self.start = False
    self.count = 0
    self.ui.timer_value.setText("<p align=\"center\">" + f"<span"
style=" font-size:{self.police}pt;" + ">" + "<font color = \"#ffffff\">" +
" ")

def showTime(self):
    current_time = QTime.currentTime()
    self.currentHeure = current_time.toString('hh')
    self.currentMinute = current_time.toString('mm')
    self.currentSeconde = current_time.toString('ss')
    now = QDate.currentDate()
    self.currentDate = now.toString(Qt.DefaultLocaleLongDate)
    return current_time.toString('hh:mm')

def showtime2(self):
    if self.start:
```

```

        self.count -= 1
        self.ui.timer_value.setText("<p align=\"center\>" + f"<span
style=\" font-size:{self.police}pt;\>" + "<font color = \"#ffffff\>" +
str(self.count))
        if self.count == 0:
            self.start = False
            self.timerFinished.emit()      #Baisser le son
            self.play_sound()
            self.sonTimerFinished.emit()
            self.ui.timer_value.setText("<p align=\"center\>" +
f"<span style=\" font-size:{self.police}pt;\>" + "<font color =
\"#ffffff\>" + "")
```

widgetMusiqueUSB :

```

from PyQt5.QtCore import QUrl
from PyQt5.QtMultimedia import QMediaPlayer, QMediaPlaylist,
QMediaContent
from widgetRadio import widgetRadio
import os
from mysql.connector import Error

class widgetMusiqueUSB():
    def __init__(self, ui):
        self.ui=ui
        """v Police v"""
        self.police = 36
        """v numero v"""
        #self.radio = widgetRadio(self.ui)
        self.selectFileDir()
        self.fileExt = r".mp3"
        self.listeMusique = [_ for _ in os.listdir(self.fileDir) if
_.endswith(self.fileExt)]
        self.ui.text_radio_musique.setText("<p align=\"center\>" +
f"<span style=\" font-size:{self.police}pt;\>" + "<font color =
\"#ffffff\>" + "")
```

```

        self.ui.val_radio_musique.setText("<p align=\"center\>" +
f"<span style=\" font-size:{self.police}pt;\>" + "<font color =
\"#ffffff\>" + "")
```

```

        self.player = QMediaPlayer()
        self.playlist = QMediaPlaylist()
        self.musiqueJoue = None
        self.addPlaylist()
```

```

def selectFileDir(self):
    self.fileDir = "Musique"
    for lettre in "ABCDEFGHIJKLMNPQRSTUVWXYZ":
        if os.access(lettre + ":\\", os.R_OK):
            if os.access(lettre + ":\\" + "test.txt", os.R_OK):
                self.fileDir = (lettre + ":\\")
    #print(self.fileDir)

def addPlaylist(self):
    self.numero = 0
    self.tailleListeMusique = len(self.listeMusique)
    while self.numero != self.tailleListeMusique:

        self.playlist.addMedia(QMediaContent(QUrl.fromLocalFile(self.fileDir +
        "/" + self.listeMusique[self.numero])))
        self.numero = self.numero + 1
    self.numero = 1

    def songChanged(self, media):
        self.ui.val_radio_musique.setText("<p align=\"center\">" +
f"<span style=\" font-size:{self.police}pt;\">" + "<font color =
\"#ffffff\">" + media.canonicalUrl().fileName())
        self.musiqueJoue = media.canonicalUrl().fileName()

    def diminuer_son(self):
        self.player.setVolume(10)

    def monter_son(self):
        self.player.setVolume(100)

    def musique_play(self):
        self.ui.text_radio_musique.setText("<p align=\"center\">" +
f"<span style=\" font-size:{self.police}pt;\">" + "<font color =
\"#ffffff\">" + "Musique en cours : ")
        self.playlist.setCurrentIndex(self.numero)
        self.player.setPlaylist(self.playlist)
        self.player.play()

    def change_musique_sup(self):
        self.ui.text_radio_musique.setText("<p align=\"center\">" +
f"<span style=\" font-size:{self.police}pt;\">" + "<font color =
\"#ffffff\">" + "Musique en cours : ")
        self.tailleListeMusique = len(self.listeMusique)
        self.numero = self.numero + 1
        if self.numero == self.tailleListeMusique:

```

```

        self.numero = 0
self.playlist.setCurrentIndex(self.numero)
self.player.play()

def change_musique_inf(self):
    self.ui.text_radio_musique.setText("<p align=\"center\">" +
f"<span style=\" font-size:{self.police}pt;\">" + "<font color =
\"#ffffff\">" + "Musique en cours : ")
    self.tailleListeMusique = len(self.listeMusique)
    self.numero = self.numero - 1
    if self.numero < 0:
        self.numero = self.tailleListeMusique - 1
    self.playlist.setCurrentIndex(self.numero)
    self.player.play()

def musique_stop(self):
    #print("stop musique")
    self.player.stop()
    self.ui.val_radio_musique.setText("<p align=\"center\">" +
f"<span style=\" font-size:{self.police}pt;\">" + "<font color =
\"#ffffff\">" + "")
```

### Ui\_MainWindow :

```

# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'New_IHM_Miroir.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5
# is
# run again. Do not edit this file unless you know what you are doing.
```

```
from PyQt5 import QtCore, QtGui, QtWidgets
```

```
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
```

```

MainWindow.resize(1080, 1920)
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base,
brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window,
brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base,
brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window,
brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base,
brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window,
brush)
MainWindow.setPalette(palette)
self.centralwidget = QtWidgets.QWidget(MainWindow)
self.centralwidget.setObjectName("centralwidget")
self.gridLayoutWidget_2 = QtWidgets.QWidget(self.centralwidget)
self.gridLayoutWidget_2.setGeometry(QtCore.QRect(-1, 99, 1081,
231))
    self.gridLayoutWidget_2.setObjectName("gridLayoutWidget_2")
    self.gridLayout_5 =
QtWidgets.QGridLayout(self.gridLayoutWidget_2)
    self.gridLayout_5.setContentsMargins(0, 0, 0, 0)
    self.gridLayout_5.setObjectName("gridLayout_5")
    self.val_heure_act = QtWidgets.QLabel(self.gridLayoutWidget_2)
    self.val_heure_act.setObjectName("val_heure_act")
    self.gridLayout_5.addWidget(self.val_heure_act, 2, 0, 1, 1)
    spacerItem = QtWidgets.QSpacerItem(20, 40,
QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)
    self.gridLayout_5.addItem(spacerItem, 3, 0, 1, 1)
    self.val_jour_act = QtWidgets.QLabel(self.gridLayoutWidget_2)
    self.val_jour_act.setObjectName("val_jour_act")

```

```

        self.gridLayout_5.addWidget(self.val_jour_act, 1, 0, 1, 1)
        self.gridLayoutWidget_4 = QtWidgets.QWidget(self.centralwidget)
        self.gridLayoutWidget_4.setGeometry(QtCore.QRect(0, 960, 1081,
361))
        self.gridLayoutWidget_4.setObjectName("gridLayoutWidget_4")
        self.gridLayout_8 =
QtWidgets.QGridLayout(self.gridLayoutWidget_4)
        self.gridLayout_8.setContentsMargins(0, 0, 0, 0)
        self.gridLayout_8.setObjectName("gridLayout_8")
        spacerItem1 = QtWidgets.QSpacerItem(40, 20,
QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
        self.gridLayout_8.addItem(spacerItem1, 0, 1, 1, 1)
        self.horizontalLayout_7 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_7.setObjectName("horizontalLayout_7")
        self.gridLayout_8.addLayout(self.horizontalLayout_7, 0, 2, 1, 1)
        self.horizontalLayout_6 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_6.setObjectName("horizontalLayout_6")
        self.gridLayout_8.addWidget(self.horizontalLayout_6, 0, 3, 1, 1)
        self.text_radio_musique =
QtWidgets.QLabel(self.gridLayoutWidget_4)
        palette = QtGui.QPalette()
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text,
brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255, 128))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active,
QtGui.QPalette.PlaceholderText, brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text,
brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255, 128))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Inactive,
QtGui.QPalette.PlaceholderText, brush)
        brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text,
brush)
        brush = QtGui.QBrush(QtGui.QColor(0, 0, 0, 128))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled,
QtGui.QPalette.PlaceholderText, brush)
        self.text_radio_musique.setPalette(palette)

```

```

        self.text_radio_musique.setObjectName("text_radio_musique")
        self.horizontalLayout_6.addWidget(self.text_radio_musique)
        self.val_radio_musique =
QtWidgets.QLabel(self.gridLayoutWidget_4)
        self.val_radio_musique.setObjectName("val_radio_musique")
        self.horizontalLayout_6.addWidget(self.val_radio_musique)
        self.gridLayout_8.addLayout(self.horizontalLayout_6, 0, 0, 1, 1)
        self.gridLayoutWidget_6 = QtWidgets.QWidget(self.centralwidget)
        self.gridLayoutWidget_6.setGeometry(QtCore.QRect(0, 330, 1081,
241))
        self.gridLayoutWidget_6.setObjectName("gridLayoutWidget_6")
        self.gridLayout = QtWidgets.QGridLayout(self.gridLayoutWidget_6)
        self.gridLayout.setContentsMargins(0, 0, 0, 0)
        self.gridLayout.setObjectName("gridLayout")
        spacerItem2 = QtWidgets.QSpacerItem(40, 20,
QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
        self.gridLayout.addItem(spacerItem2, 0, 2, 1, 1)
        self.timer_value = QtWidgets.QLabel(self.gridLayoutWidget_6)
        self.timer_value.setObjectName("timer_value")
        self.gridLayout.addWidget(self.timer_value, 0, 1, 1, 1)
        spacerItem3 = QtWidgets.QSpacerItem(40, 20,
QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
        self.gridLayout.addItem(spacerItem3, 0, 0, 1, 1)
        self.horizontalLayoutWidget =
QtWidgets.QWidget(self.centralwidget)
        self.horizontalLayoutWidget.setGeometry(QtCore.QRect(0, 660,
1081, 301))

self.horizontalLayoutWidget.setObjectName("horizontalLayoutWidget")
        self.horizontalLayout =
QtWidgets.QHBoxLayout(self.horizontalLayoutWidget)
        self.horizontalLayout.setContentsMargins(0, 0, 0, 0)
        self.horizontalLayout.setObjectName("horizontalLayout")
        self.gridLayout_4 = QtWidgets.QGridLayout()
        self.gridLayout_4.setObjectName("gridLayout_4")
        self.val_act = QtWidgets.QLabel(self.horizontalLayoutWidget)
        self.val_act.setObjectName("val_act")
        self.gridLayout_4.addWidget(self.val_act, 1, 1, 1, 1)
        self.val_image = QtWidgets.QLabel(self.horizontalLayoutWidget)
        palette = QtGui.QPalette()
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text,
brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255, 128))

```

```

        brush.setStyle(QtCore.Qt.NoBrush)
        palette.setBrush(QtGui.QPalette.Active,
QtGui.QPalette.PlaceholderText, brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text,
brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255, 128))
        brush.setStyle(QtCore.Qt.NoBrush)
        palette.setBrush(QtGui.QPalette.Inactive,
QtGui.QPalette.PlaceholderText, brush)
        brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text,
brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255, 128))
        brush.setStyle(QtCore.Qt.NoBrush)
        palette.setBrush(QtGui.QPalette.Disabled,
QtGui.QPalette.PlaceholderText, brush)
        self.val_image.setPalette(palette)
        self.val_image.setObjectName("val_image")
        self.gridLayout_4.addWidget(self.val_image, 1, 0, 1, 1)
        spacerItem4 = QtWidgets.QSpacerItem(20, 40,
QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)
        self.gridLayout_4.addItem(spacerItem4, 3, 1, 1, 1)
        self.val_image_gust =
QtWidgets.QLabel(self.horizontalLayoutWidget)
        palette = QtGui.QPalette()
        brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active,
QtGui.QPalette.WindowText, brush)
        brush = QtGui.QBrush(QtGui.QColor(240, 240, 240))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button,
brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light,
brush)
        brush = QtGui.QBrush(QtGui.QColor(227, 227, 227))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight,
brush)
        brush = QtGui.QBrush(QtGui.QColor(160, 160, 160))

```

```
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark,
brush)
    brush = QtGui.QBrush(QtGui.QColor(160, 160, 160))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid,
brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text,
brush)
    brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Active,
QtGui.QPalette.BrightText, brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Active,
QtGui.QPalette.ButtonText, brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base,
brush)
    brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window,
brush)
    brush = QtGui.QBrush(QtGui.QColor(105, 105, 105))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow,
brush)
    brush = QtGui.QBrush(QtGui.QColor(51, 153, 255))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Active,
QtGui.QPalette.Highlight, brush)
    brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Active,
QtGui.QPalette.HighlightedText, brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 255))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Link,
brush)
    brush = QtGui.QBrush(QtGui.QColor(255, 0, 255))
    brush.setStyle(QtCore.Qt.SolidPattern)
```

```
palette.setBrush(QtGui.QPalette.Active,
QtGui.QPalette.LinkVisited, brush)
    brush = QtGui.QBrush(QtGui.QColor(245, 245, 245))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Active,
QtGui.QPalette.AlternateBase, brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
    brush.setStyle(QtCore.Qt.NoBrush)
    palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.NoRole,
brush)
    brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Active,
QtGui.QPalette.ToolTipBase, brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Active,
QtGui.QPalette.ToolTipText, brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 128))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Active,
QtGui.QPalette.PlaceholderText, brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive,
QtGui.QPalette.WindowText, brush)
    brush = QtGui.QBrush(QtGui.QColor(240, 240, 240))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button,
brush)
    brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light,
brush)
    brush = QtGui.QBrush(QtGui.QColor(227, 227, 227))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive,
QtGui.QPalette.Midnight, brush)
    brush = QtGui.QBrush(QtGui.QColor(160, 160, 160))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark,
brush)
    brush = QtGui.QBrush(QtGui.QColor(160, 160, 160))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid,
```

```
brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text,
brush)
    brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive,
QtGui.QPalette.BrightText, brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive,
QtGui.QPalette.ButtonText, brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base,
brush)
    brush = QtGui.QBrush(QtGui.QColor(84, 84, 84))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window,
brush)
    brush = QtGui.QBrush(QtGui.QColor(105, 105, 105))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow,
brush)
    brush = QtGui.QBrush(QtGui.QColor(240, 240, 240))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive,
QtGui.QPalette.Highlight, brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive,
QtGui.QPalette.HighlightedText, brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 255))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Link,
brush)
    brush = QtGui.QBrush(QtGui.QColor(255, 0, 255))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive,
QtGui.QPalette.LinkVisited, brush)
    brush = QtGui.QBrush(QtGui.QColor(245, 245, 245))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive,
QtGui.QPalette.AlternateBase, brush)
```

```

brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.NoBrush)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.NoRole,
brush)
    brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive,
QtGui.QPalette.ToolTipBase, brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive,
QtGui.QPalette.ToolTipText, brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 128))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Inactive,
QtGui.QPalette.PlaceholderText, brush)
    brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Disabled,
QtGui.QPalette.WindowText, brush)
    brush = QtGui.QBrush(QtGui.QColor(240, 240, 240))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button,
brush)
    brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light,
brush)
    brush = QtGui.QBrush(QtGui.QColor(247, 247, 247))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Disabled,
QtGui.QPalette.Midlight, brush)
    brush = QtGui.QBrush(QtGui.QColor(160, 160, 160))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark,
brush)
    brush = QtGui.QBrush(QtGui.QColor(160, 160, 160))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid,
brush)
    brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text,
brush)
    brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))

```

```
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled,
QtGui.QPalette.BrightText, brush)
    brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Disabled,
QtGui.QPalette.ButtonText, brush)
    brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base,
brush)
    brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window,
brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow,
brush)
    brush = QtGui.QBrush(QtGui.QColor(51, 153, 255))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Disabled,
QtGui.QPalette.Highlight, brush)
    brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Disabled,
QtGui.QPalette.HighlightedText, brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 255))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Link,
brush)
    brush = QtGui.QBrush(QtGui.QColor(255, 0, 255))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Disabled,
QtGui.QPalette.LinkVisited, brush)
    brush = QtGui.QBrush(QtGui.QColor(245, 245, 245))
    brush.setStyle(QtCore.Qt.SolidPattern)
    palette.setBrush(QtGui.QPalette.Disabled,
QtGui.QPalette.AlternateBase, brush)
    brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
    brush.setStyle(QtCore.Qt.NoBrush)
    palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.NoRole,
brush)
    brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
    brush.setStyle(QtCore.Qt.SolidPattern)
```

```
palette.setBrush(QtGui.QPalette.Disabled,
QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled,
QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0, 128))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled,
QtGui.QPalette.PlaceholderText, brush)
self.val_image_gust.setPalette(palette)
self.val_image_gust.setObjectName("val_image_gust")
self.gridLayout_4.addWidget(self.val_image_gust, 2, 0, 1, 1)
self.val_gust = QtWidgets.QLabel(self.horizontalLayoutWidget)
self.val_gust.setObjectName("val_gust")
self.gridLayout_4.addWidget(self.val_gust, 2, 1, 1, 1)
spacerItem5 = QtWidgets.QSpacerItem(40, 20,
QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
self.gridLayout_4.addItem(spacerItem5, 1, 2, 1, 1)
spacerItem6 = QtWidgets.QSpacerItem(40, 20,
QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
self.gridLayout_4.addItem(spacerItem6, 2, 2, 1, 1)
self.horizontalLayout.addLayout(self.gridLayout_4)
spacerItem7 = QtWidgets.QSpacerItem(40, 20,
QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
self.horizontalLayout.addItem(spacerItem7)
self.gridLayout_7 = QtWidgets.QGridLayout()
self.gridLayout_7.setObjectName("gridLayout_7")
self.ima_temp_local =
QtWidgets.QLabel(self.horizontalLayoutWidget)
self.ima_temp_local.setObjectName("ima_temp_local")
self.gridLayout_7.addWidget(self.ima_temp_local, 1, 0, 1, 1)
self.ima_humi_local =
QtWidgets.QLabel(self.horizontalLayoutWidget)
self.ima_humi_local.setObjectName("ima_humi_local")
self.gridLayout_7.addWidget(self.ima_humi_local, 2, 0, 1, 1)
spacerItem8 = QtWidgets.QSpacerItem(40, 20,
QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
self.gridLayout_7.addItem(spacerItem8, 2, 2, 1, 1)
self.val_temp_local =
QtWidgets.QLabel(self.horizontalLayoutWidget)
self.val_temp_local.setObjectName("val_temp_local")
self.gridLayout_7.addWidget(self.val_temp_local, 1, 1, 1, 1)
self.val_humi_local =
QtWidgets.QLabel(self.horizontalLayoutWidget)
```

```

        self.val_humi_local.setObjectName("val_humi_local")
        self.gridLayout_7.addWidget(self.val_humi_local, 2, 1, 1, 1)
        spacerItem9 = QtWidgets.QSpacerItem(40, 20,
QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
        self.gridLayout_7.addItem(spacerItem9, 1, 2, 1, 1)
        spacerItem10 = QtWidgets.QSpacerItem(20, 40,
QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)
        self.gridLayout_7.addItem(spacerItem10, 3, 0, 1, 1)
        self.horizontalLayout.addLayout(self.gridLayout_7)
        self.gridLayoutWidget_3 = QtWidgets.QWidget(self.centralwidget)
        self.gridLayoutWidget_3.setGeometry(QtCore.QRect(0, 0, 1081,
101))
        self.gridLayoutWidget_3.setObjectName("gridLayoutWidget_3")
        self.gridLayout_6 =
QtWidgets.QGridLayout(self.gridLayoutWidget_3)
        self.gridLayout_6.setContentsMargins(0, 0, 0, 0)
        self.gridLayout_6.setObjectName("gridLayout_6")
        self.helloUser = QtWidgets.QLabel(self.gridLayoutWidget_3)
        self.helloUser.setObjectName("helloUser")
        self.gridLayout_6.addWidget(self.helloUser, 0, 0, 1, 1)
        spacerItem11 = QtWidgets.QSpacerItem(20, 40,
QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)
        self.gridLayout_6.addItem(spacerItem11, 1, 0, 1, 1)
        self.gridLayoutWidget_7 = QtWidgets.QWidget(self.centralwidget)
        self.gridLayoutWidget_7.setGeometry(QtCore.QRect(0, 570, 1081,
91))
        self.gridLayoutWidget_7.setObjectName("gridLayoutWidget_7")
        self.gridLayout_9 =
QtWidgets.QGridLayout(self.gridLayoutWidget_7)
        self.gridLayout_9.setContentsMargins(0, 0, 0, 0)
        self.gridLayout_9.setObjectName("gridLayout_9")
        spacerItem12 = QtWidgets.QSpacerItem(40, 20,
QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
        self.gridLayout_9.addItem(spacerItem12, 1, 1, 1, 1)
        self.ima_local = QtWidgets.QLabel(self.gridLayoutWidget_7)
        self.ima_local.setObjectName("ima_local")
        self.gridLayout_9.addWidget(self.ima_local, 1, 2, 1, 1)
        self.val_lieu = QtWidgets.QLabel(self.gridLayoutWidget_7)
        self.val_lieu.setObjectName("val_lieu")
        self.gridLayout_9.addWidget(self.val_lieu, 1, 0, 1, 1)
        spacerItem13 = QtWidgets.QSpacerItem(20, 40,
QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)
        self.gridLayout_9.addItem(spacerItem13, 0, 1, 1, 1)
        MainWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)

```

```

    self.statusbar.setObjectName("statusbar")
    MainWindow.setStatusBar(self.statusbar)

    self.retranslateUi(MainWindow)
    QtCore.QMetaObject.connectSlotsByName(MainWindow)

def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow",
"MainWindow"))
    self.val_heure_act.setText(_translate("MainWindow",
"<html><head/><body><p align=\"center\"><span style=\" font-size:36pt;
font-weight:600;\">HH:MM:SS</span></p></body></html>"))
    self.val_jour_act.setText(_translate("MainWindow",
"<html><head/><body><p align=\"center\"><span style=\" font-size:36pt;
font-weight:600;\">DDD dd MMM YYYY</span></p></body></html>"))
    self.text_radio_musique.setText(_translate("MainWindow", "Radio
en cours de lecture :"))
    self.val_radio_musique.setText(_translate("MainWindow",
"Radio"))
    self.timer_value.setText(_translate("MainWindow",
"<html><head/><body><p>Na</p></body></html>"))
    self.val_act.setText(_translate("MainWindow",
"<html><head/><body><p align=\"right\"><span style=\"
font-size:36pt;\">Temp</span></p></body></html>"))
    self.val_image.setText(_translate("MainWindow",
"<html><head/><body><p align=\"center\">logo</p></body></html>"))
    self.val_image_gust.setText(_translate("MainWindow",
"WindLogo"))
    self.val_gust.setText(_translate("MainWindow",
"<html><head/><body><p align=\"right\"><span style=\"
font-size:36pt;\">km/h</span></p></body></html>"))
    self.ima_temp_local.setText(_translate("MainWindow", "logo temp
local"))
    self.ima_humi_local.setText(_translate("MainWindow", "logo humi
local"))
    self.val_temp_local.setText(_translate("MainWindow", "10°c"))
    self.val_humi_local.setText(_translate("MainWindow", "23%"))
    self.helloUser.setText(_translate("MainWindow", "TextLabel"))
    self.ima_local.setText(_translate("MainWindow",
"<html><head/><body><p><span style=\"
font-size:36pt;\">LOCAL</span></p></body></html>"))
    self.val_lieu.setText(_translate("MainWindow",
"<html><head/><body><p align=\"center\"><span style=\"
font-size:36pt;\">Lieu</span></p></body></html>"))

```

## Partie étudiant 3 IR (GRANDON Luca):

Programme de reconnaissance vocale :

```

import speech_recognition as sr
from PyQt5.QtCore import pyqtSignal, QThread
import time

class RecoVocale(QThread):
    getHeure = pyqtSignal()
    radio_stop = pyqtSignal()
    radio_play = pyqtSignal()
    radio_change = pyqtSignal()
    musique_play = pyqtSignal()
    musique_stop = pyqtSignal()
    musique_sup = pyqtSignal()
    musique_inf = pyqtSignal()
    getRadio = pyqtSignal()
    getMeteo = pyqtSignal()
    getDate = pyqtSignal()
    getMusique = pyqtSignal()
    getProfil = pyqtSignal()
    getCapteur = pyqtSignal()
    commandeUnknow = pyqtSignal()
    setValue = pyqtSignal()
    startChrono = pyqtSignal()
    resetChrono = pyqtSignal()
    stopChrono = pyqtSignal()
    OnLumiere = pyqtSignal()
    OffLumiere = pyqtSignal()
    finRecoV = pyqtSignal()

    def __init__(self, parent=None):
        QThread.__init__(self, parent)
        self.nomRadio = None

    def callback(self, query):

        try:
            if 'heure' in query:      #Quel heure est'il ?
                print('1')
                self.getHeure.emit()
            elif 'mets la radio' in query:   #mets la radio #selon
radio pref

```

```

        print('2')
        self.radio_play.emit()
    elif 'arrête la radio' in query: #arrête la radio
        print('3')
        self.radio_stop.emit()
    elif 'radio en cours' in query: #Quel est la radio en
cours de lecture ?
        print('4')
        self.getRadio.emit()
    elif "mets NRJ" in query: #mets NRJ
        print('5')
        self.nomRadio = "NRJ"
        self.radio_play.emit()
    elif "mets France Info" in query: #mets France Info
        print('6')
        self.nomRadio = "France Info"
        self.radio_change.emit()
    elif "mets France Musique" in query: #mets France Musique
        print('7')
        self.nomRadio = "France Musique"
        self.radio_change.emit()
    elif "mets nostalgie" in query: #mets nostalgie
        print('8')
        self.nomRadio = "Nostalgie"
        self.radio_change.emit()
    elif "mets Chérie FM" in query: #mets Chérie FM
        print('9')
        self.nomRadio = "Chérie FM"
        self.radio_change.emit()
    elif "mets rire et chansons" in query: #mets rire et
chansons
        print('10')
        self.nomRadio = "Rire & Chansons"
        self.radio_change.emit()
    elif "mets RTL" in query: #mets RTL
        print('11')
        self.nomRadio = "RTL"
        self.radio_change.emit()
    elif "mets RTL2" in query: #mets RTL2
        print('12')
        self.nomRadio = "RTL2"
        self.radio_change.emit()
    elif "mets Fun Radio France" in query: #mets Fun Radio
France
        print('13')

```

```

        self.nomRadio = "FunRadioFrance"
        self.radio_change.emit()
    elif "mets Europe 1" in query: #mets Europe 1
        print('14')
        self.nomRadio = "Europe1"
        self.radio_change.emit()
    elif "mets Virgin Radio" in query: #mets Virgin Radio
        print('15')
        self.nomRadio = "Virgin Radio"
        self.radio_change.emit()
    elif "mets RFM" in query: #mets RFM
        print('16')
        self.nomRadio = "RFM"
        self.radio_change.emit()
    elif "mets RMC" in query: #mets RMC
        print('17')
        self.nomRadio = "RMC"
        self.radio_change.emit()
    elif "mets BFM Business" in query: #mets BFM Business
        print('18')
        self.nomRadio = "BFM Business"
        self.radio_change.emit()
    elif "mets Skyrock" in query: #mets Skyrock
        print('19')
        self.nomRadio = "Skyrock"
        self.radio_change.emit()
    elif "mets Radio Classique" in query: #mets Radio
        print('20')
        self.nomRadio = "RadioClassique"
        self.radio_change.emit()
    elif "mets France Inter" in query: #mets France Inter
        print('21')
        self.nomRadio = "France Inter"
        self.radio_change.emit()
    elif "mets France Culture" in query: #mets France
        print('22')
        self.nomRadio = "France Culture"
        self.radio_change.emit()
    elif "mets FIP nationale" in query: #mets FIP nationale
        print('23')
        self.nomRadio = "FIP Nationale"
        self.radio_change.emit()
    elif "mets move" in query: #mets move

```

```

        print('24')
        self.nomRadio = "Mouv'"
        self.radio_change.emit()
    elif "mets wii FM" in query:    #mets oui FM
        print('25')
        self.nomRadio = "Oui FM"
        self.radio_change.emit()
    elif "mets Jazz Radio" in query:   #mets Jazz Radio
        print('26')
        self.nomRadio = "Jazz Radio"
        self.radio_change.emit()
    elif "mets m Radio" in query: #mets m Radio
        print('27')
        self.nomRadio = "m Radio"
        self.radio_change.emit()
    elif "mets France Bleu" in query: #mets France Bleu
        print('28')
        self.nomRadio = "France Bleu"
        self.radio_change.emit()
    elif "allume la lumière" in query:   #allume la lumière
        self.OnLumiere.emit()
    elif "éteint la lumière" in query:   #éteint la lumière
        self.OffLumiere.emit()
    elif "météo" in query:   #donne moi la météo
        print('29')
        self.getMeteo.emit()
    elif "température" in query: #Quel est la température de
la salle/Quel est le pourcentage d'humidité de la salle ?
        self.getCapteur.emit()
    elif "humidité" in query: # Quel est la température de
la salle/Quel est le pourcentage d'humidité de la salle ?
        self.getCapteur.emit()
    elif "profil charger" in query:   #Quel est le profil
chargé ?
        print('30')
        self.getProfil.emit()
    elif "date" in query:   #Quel est la date d'aujourd'hui ?
        print("31")
        self.getDate.emit()
    elif "mets un minuteur" in query: #mets un minuteur
        print('32')
        self.setValue.emit()
        self.startChrono.emit()
    elif "arrête le minuteur" in query: #arrête le minuteur
        print('33')

```

```

        self.stopChrono.emit()
    elif "annule le minuteur" in query: #annule le minuteur
        print('34')
        self.resetChrono.emit()
    elif "créer un minuteur" in query: #créer un minuteur
        print('35')
        self.setValue.emit()
    elif "mets de la musique" in query: #mets de la musique
        print('36')
        self.musique_play.emit()
    elif "arrête la musique" in query: #arrête la musique
        print('37')
        self.musique_stop.emit()
    elif "musique en cours" in query: #Quel est la musique en
cours de lecture ?
        print('38')
        self.getMusique.emit()
    elif "musique suivante" in query: #musique suivante
        print('39')
        self.musique_sup.emit()
    elif "musique précédente" in query: #musique précédente
        print('40')
        self.musique_inf.emit()

except sr.UnknownValueError :
    self.commandeUnknow.emit()

def run(self):
    while True:
        r = sr.Recognizer()
        print("test1")
        with sr.Microphone() as source2:
            #r.adjust_for_ambient_noise(source2)
            print("listening")
            r.pause_threshold = 1
            try:
                print("dans le try")
                audio = r.listen(source2)# si snowboy alors timeout=5
            except Exception as e:
                print(e)
                print("écoute réussie")
            try:
                print("Reco en cours...")
                query = r.recognize_google(audio, language='fr-FR')
                print("Phrase reconnu :" + query)

```

```

        self.callback(query)

    except Exception as e:
        print(e)
    print("test2")

```

Code django / interface administrateur

Models.py :

```

from django.core.validators import MaxValueValidator, MinValueValidator
from django.db import models

class Profil(models.Model):

    class radio(models.TextChoices):
        NRJ = 'NRJ'
        FranceInfo = 'FranceInfo'
        FranceMusic = 'FranceMusic'
        Nostalgie = 'Nostalgie'
        CherieFM = 'CherieFM'
        RireEtChanson = 'Rire&Chanson'
        RTL = 'RTL'
        RTL2 = 'RTL2'
        FunRadioFrance = 'FunRadioFrance'
        Europe1 = 'Europe1'
        VirginRadio = 'VirginRadio'
        RFM = 'RFM'
        RMC = 'RMC'
        BFM_Business = 'BFMBusiness'
        Skyrock = 'SkyRock'
        RadioClassique = 'RadioClassique'
        FranceInter = 'France Inter'
        FranceCulture = 'France Culture'
        FIP_National = 'FIPNational'
        Mouv = 'Mouv'
        OuiFM = 'OuiFM'
        JazzRadio = 'JazzRadio'
        M_Radio = 'M Radio'
        FranceBleu = "France Bleu"

        pseudo = models.fields.CharField(max_length=40, default="Default")
        nom = models.fields.CharField(max_length=30, default="Default")
        prenom = models.fields.CharField(max_length=30, default="Default")

```

```

ville = models.fields.CharField(max_length=44, default="Paris")
radioPreferee = models.fields.CharField(choices=radio.choices,
max_length=20, default="NRJ")
dateNaissance = models.DateField(default="1900-01-01")
img_User = models.ImageField(upload_to='profil', default=False)
img_user = models.BinaryField(default=True)

def __str__(self):
    return f'{self.pseudo}'

```

### views.py :

```

from django.shortcuts import render, redirect
from listings.models import Profil
from listings.forms import ProfilForm
from django.http import HttpResponse

def profil_list(request):
    profil = Profil.objects.all()
    return render(request,
                  'listings/profil.html',
                  {'profil': profil})

def profil_detail(request, id):
    profil = Profil.objects.get(id=id)
    return render(request,
                  'listings/profil_detail.html',
                  {'profil': profil})

def profil_add(request):
    if request.method == 'POST':
        form = ProfilForm(request.POST)
        if form.is_valid():
            profil = form.save(commit=False)
            profil.save()
            return redirect('profil-detail', profil.id)
    else:
        form = ProfilForm()

    return render(request, 'listings/profil_add.html', {'form': form})

def profil_update(request, id):
    profil = Profil.objects.get(id=id)

    if request.method == 'POST':

```

```
form = ProfilForm(request.POST, instance=profil)

if form.is_valid():
    form.save()
    return redirect('profil-detail', profil.id)

else:
    form = ProfilForm(instance=profil)

return render(request,
              'listings/profil_update.html',
              {'form': form})

def profil_delete(request, id):
    profil = Profil.objects.get(id=id)

    if request.method == 'POST' :
        profil.delete()
        return redirect('profil-list')

    return render(request,
                  'listings/profil_delete.html',
                  {'profil': profil})
```

urls.py :

```
from django.contrib import admin
from django.urls import path
from listings import views
```



```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('profil/', views.profil_list, name='profil-list'),
    path('profil/add/', views.profil_add, name='profil-add'),
    path('profil/<int:id>/', views.profil_detail, name='profil-detail'),
    path('profil/<int:id>/update/', views.profil_update,
name='profil-up'),
    path('profil/<int:id>/delete/', views.profil_delete,
name='profil-del'),
]
```

#### settings.py :

```
from pathlib import Path

EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'

BASE_DIR = Path(__file__).resolve().parent.parent

SECRET_KEY =
'django-insecure-t7b0apyym48qg0d%c0m3uqs@mb$97x!7zf9+6q)ktz6q0h8+n='

DEBUG = True

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'listings',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
```

```

'django.middleware.common.CommonMiddleware',
'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'merchex.urls'

TEMPLATES = [
{
    'BACKEND': 'django.template.backends.django.DjangoTemplates',
    'DIRS': [],
    'APP_DIRS': True,
    'OPTIONS': {
        'context_processors': [
            'django.template.context_processors.debug',
            'django.template.context_processors.request',
            'django.contrib.auth.context_processors.auth',
            'django.contrib.messages.context_processors.messages',
        ],
    },
},
],
WSGI_APPLICATION = 'merchex.wsgi.application'

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'profilprojet',
        'USER': 'root',
        'HOST': '',
        'PORT': '',
        'OPTIONS': {
            'init_command': "SET sql_mode='STRICT_TRANS_TABLES'"
        }
    }
}

AUTH_PASSWORD_VALIDATORS = [
{
    'NAME':

```



```
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
],
{
    'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
},
{
    'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
},
{
    'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
},
]
]

LANGUAGE_CODE = 'fr-FR'
TIME_ZONE = 'Europe/Paris'
USE_I18N = True
USE_L10N = True
USE_TZ = True

STATIC_URL = '/static/'

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

admin.py :

```
from django.contrib import admin
from listings.models import Profil

class ProfAdmin(admin.ModelAdmin):
    list_display = ('pseudo', 'id', 'radioPreferee', 'dateNaissance')

admin.site.register(Profil, ProfAdmin)
```

apps.py :

```
from django.apps import AppConfig

class ListingsConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'listings'
```

## Partie étudiant 4 IR (CARDONA Alexandre):

Programme renamePlusEncoding :

```

from PIL import Image
import os
import mysql.connector
from mysql.connector import Error
import pickle
import cv2
import numpy as np
import face_recognition as fr

def bdd_connection():
    connection = mysql.connector.connect(host='172.20.0.165', database='projetMiroir',
                                          user='projet', password='miroir')
    return connection

#Insertion des données dans la table renvoie False si une erreur est détecté
def insertBLOB(id, img_user):
    print("Inserting BLOB into table")
    try:
        connection = bdd_connection()
        cursor = connection.cursor()
        sql_insert_blob_query = f""" UPDATE TABLE listings_profil(img_user) VALUES
        ({img_user}) WHERE id = ({id})"""

        empPicture = lecture_image(img_user)

        # Convert data into tuple format
        insert_blob_tuple = (id, empPicture)
        result = cursor.execute(sql_insert_blob_query, insert_blob_tuple)
        connection.commit()
        print("Image inserted successfully as a BLOB into table", result)

    except Error as error:
        print("Failed inserting BLOB data into MySQL table {}".format(error))

    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
            print("Image insérée")
    return 1

```

```

def listeBDD():
    try:
        connection = bdd_connection()
        cursor = connection.cursor(buffered=True)
        sql_select_blob_query = "SELECT id_pic, img_user FROM recoF"
        result = cursor.execute(sql_select_blob_query)
        connection.commit()
        resultat_listBDD = cursor.fetchall()
        return {element[0]:element[1] for element in resultat_listBDD}

    except Error as error:
        print("Failed to select BLOB data into MySQL table {}".format(error))

    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
            print("MySQL connection is closed")

def createEncodings():
    try:
        print("Début de l'encodage")
        encodeList = []
        list_id = []
        result_bdd = listeBDD()
        # for img in images:
        for key in result_bdd.keys():
            nparr = np.frombuffer(result_bdd[key], np.uint8)
            img_np = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
            rgb = cv2.cvtColor(img_np, cv2.COLOR_BGR2RGB)
            encode = fr.face_encodings(rgb)[0]
            encodeList.append(encode)
            list_id.append(key)
            # Ajout du fichier pickle
        data1 = {"encodings": encodeList, "list_id": list_id}
        f = open("encodings.pickle", "wb")
        f.write(pickle.dumps(data1))
        f.close()
        print("Encodage terminé")
        return 1

    except Error as error:
        print("Failed to create pickle from encodings".format(error))

```

```

    return 0

def lecture_image(filename):
    with open(filename, 'rb') as file:
        binaryData = file.read()
    return binaryData

def renameFile():
    fileExt = r".jpg"
    fileExt2 = r".png"
    fileExt3 = r".PNG"
    NomImg= [_ for _ in os.listdir() if _.endswith(fileExt)] or [_ for _ in os.listdir() if
_.endswith(fileExt2)] or [_ for _ in os.listdir() if _.endswith(fileExt3)]
    ImgNom = NomImg[0]
    print(NomImg)
    im = Image.open(NomImg[0])
    if ImgNom.endswith(fileExt2) or ImgNom.endswith(fileExt3):
        rgb_im = im.convert('RGB')
        rgb_im.save("1.jpg")

    else:
        im.save("1.jpg")
    blob = lecture_image("1.jpg")
    img = NomImg[0]
    os.remove(img)
    insertBLOB('1', "1.jpg", )
    os.remove("1.jpg")
    return blob

renameFile()
createEncodings()

```

Code recoF :

```

import time
import cv2
import numpy as np
import face_recognition as fr
import mysql.connector
from PyQt5.QtCore import pyqtSignal
from mysql.connector import Error
from PyQt5.QtCore import QThread
import pickle

class RecoFacial(QThread):
    userChanged = pyqtSignal()

```

```

def __init__(self, parent=None):
    QThread.__init__(self, parent)
    self.id_user = 0
    self.faceDis = 0
    self.connection = None

def bdd_connection(self):
    self.connection = mysql.connector.connect(host='172.20.10.4',
database='projetMiroir', user='projet', password='miroir')

def markAttendance(self):
    try:
        self.bdd_connection()
        cursor = self.connection.cursor(buffered=True)
        insert_stat = f"INSERT INTO stat(id) VALUES ({self.id_user});"
        result = cursor.execute(insert_stat)
        self.connection.commit()

    except Error as error:
        print("Failed to insert statistic {}".format(error))

    finally:
        if self.connection.is_connected():
            cursor.close()
            self.connection.close()

def run(self):
    self.data =
pickle.loads(open('C:\\\\Users\\\\maxen\\\\Documents\\\\encoding\\\\encodings.pickle',
"rb").read())
    images = []
    classNames = []
    self.encodeListKnown = self.data["encodings"]
    self.classNames = self.data["list_id"]

    cap = cv2.VideoCapture(0)

    while True:
        cap.set(cv2.CAP_PROP_EXPOSURE, 1)
        cap.set(cv2.CAP_PROP_FPS, 5)
        cap.set(3,640)
        cap.set(4,480)
        success, img = cap.read()

```

```
img = captureScreen()
imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

facesCurFrame = fr.face_locations(imgS)
encodesCurFrame = fr.face_encodings(imgS, facesCurFrame)

for encodeFace, faceLoc in zip(encodesCurFrame,
facesCurFrame):
    matches = fr.compare_faces(self.encodeListKnown,
encodeFace)
    faceDis = fr.face_distance(self.encodeListKnown,
encodeFace)
    matchIndex = np.argmin(faceDis)
    if faceDis[matchIndex] < 0.50: # profil reconnu
        self.changer = False
        if self.id_user != self.classNames[matchIndex]:
            self.changed = True
            self.id_user = self.classNames[matchIndex]
            self.id_pic = self.id_user
            if self.changed == True:
                self.userChanged.emit()
            self.markAttendance()
            time.sleep(5)
    else: # profil inconnu
        self.changed = False
        if self.id_user != self.classNames[matchIndex]:
            self.changed = True
            self.id_user = 1
            self.id_pic = self.id_user
            if self.changed == True:
                self.userChanged.emit()
            self.markAttendance()
            time.sleep(5)
```