Dossier de conception

Incrément 2
Responsable du document : Hugo BOUY
État du document : Version finale

Prose-2024-A1
PSC (Prototype Sonnette Connectée)
STMicroelectronics

Version : 2.3
8 juin 2023

Table des versions

Date	Actions	Auteur	Version	Révision
07/06/2023	Corrections finale pour AN de code du 08/06/23	Hugo BOUY	2.3	0
07/06/2023	Relecture typo et cohérence globale	Romain TROVALLET	2.2	1
05/06/2023	Relecture et correction diverses	Hugo BOUY	2.2	0
04/06/2023	Corrections des descriptions textuelles des objets de AOP et de la partie gestion du multitâche	Mathis MOULIN	2.1	2
03/06/2023	Mise à jour de la description textuelle des protocoles de communication après AN	Paul CHIRON	2.1	1
03/06/2023	Correction architecture générale après AN	Hugo BOUY	2.1	0
31/05/2023	Relecture et validation pour AN Conception du 31/05/2023	Toute l'équipe	2.0	0
30/05/2023	Correction Diagramme de séquence Starter AOP et SoftSonnette	Romain TROVALLET	1.4	0
29/05/2023	Mise en page et MAJ glossaire	Romain TROVALLET	1.3	0
29/05/2023	Ajout du diagramme de séquence détaillé du CU Ouvrir Porte	Romain TROVALLET	1.2	0
19/04/2023	Corrections MAE, diagramme de séquence et architecture candidate suite à l'AC	Tout le groupe	1.1	0
18/04/2023	Relecture et validation pour AC Conception du 19/04/2023	Tout le groupe	1.0	0
18/04/2023	Rédaction partie 2.3.1	Romain TROVALLET	0.6	0
17/04/2023	Correction partie 2.3 et ajout MAE Cameraman	Hugo BOUY	0.5	20
17/04/2023	Mise à jour MaE GUI	Bastien CASSAR	0.5	19
17/04/2023	Correction architecture candidate	Hugo BOUY	0.5	18
17/04/2023	Ajout descriptions textuelles des classes cameraman et recognitionAI	Paul JURET	0.5	17

Date	Actions	Auteur	Version	Révision
17/04/2023	Ajout de la partie 2.3.3.6 (classe Guard) et 2.3.3.7 (classe Bouncer)	Hugo BOUY	0.5	16
17/04/2023	Description textuelle des classes UISP et doorManager	Mathis MOULIN	0.5	15
16/04/2023	Ajout description textuelle de GUI	Bastien CASSAR	0.5	14
16/04/2023	Ajout classe EmployeeManager et MaE Cameraman	Paul CHIRON	0.5	13
16/04/2023	Ajout diagramme de classe connectionManager et Clock avec leur description	Laurent Letenneur	0.5	12
16/04/2023	Correction diagrammes de séquences	Hugo BOUY	0.5	11
16/04/2023	Ajout diagramme de classe UISS	Romain TROVALLET	0.5	10
15/04/2023	Ajout du diagramme de classe	Hugo BOUY	0.5	9
15/04/2033	Ajout machine à état Bouncer	Paul JURET	0.5	8
15/04/2023	Ajout du diagramme de séquence du CU quitter	Hugo BOUY	0.5	7
15/04/2023	Ajout des description des diagrammes de séquence stratégique et seConnecter	Laurent Letenneur	0.5	6
15/04/2023	Ajout MAE connectionManager	Laurent Letenneur	0.5	5
15/04/2023	Ajout des MàE de GUI	Bastien CASSAR	0.5	4
13/04/2023	Ajout diagramme de séquence du CU Regarder vidéo et Consulter Calendrier	Bastien CASSAR	0.5	3
12/04/2023	Ajouts diagrammes de séquence	Romain TROVALLET	0.5	2
11/04/2023	Modification diagramme de séquence stratégique et seConnecter	Laurent Letenneur	0.5	1
09/04/2023	Organisation 2.2, 2.3 et Conception Détaillée	Romain TROVALLET	0.5	0
08/04/2023	Rédaction 2.1	Romain TROVALLET	0.4	0
08/04/2023	Rédaction introduction	Romain TROVALLET	0.3	0
05/04/2023	Ajout diagramme de séquence CU initialiser SoftSonnette	Paul JURET	0.2	1
04/04/2023	Rédaction 2.2	Romain TROVALLET	0.2	0

Date	Actions	Auteur	Version	Révision
03/04/2023	Ajout diagramme de séquence CU Consulter Liste employés	Mathis MOULIN	0.1	8
03/04/2023	Ajout diagramme de séquence Entrer et MaE UISS	Paul CHIRON	0.1	7
03/04/2023	Ajout diagramme de séquence stratégique et seConnecter	Laurent Letenneur	0.1	6
03/04/2023	Création diagramme séquence CU Contrôler Porte À Distance	Romain TROVALLET	0.1	5
30/03/2023	Ajustement de la présentation de l'architecture candidate	Hugo BOUY	0.1	4
30/03/2023	Modification architecture candidate	Paul CHIRON	0.1	3
30/03/2023	Mise à jour de l'architecture candidate vers une v2	Hugo BOUY	0.1	2
21/03/2023	Ajustement mise en page pour correspondre au PAQL	Hugo BOUY	0.1	1
21/03/2023	Rédaction introduction	Paul CHIRON	0.1	0
18/03/2023	Création du document	Paul CHIRON	0.0	0

Table des matières

1 Introduction

1.1 Objet

Ce dossier de conception a pour objectif de rassembler toute la conception du logiciel PSC. Il permettra à l'équipe A1 de développer le logiciel ainsi qu'élaborer des tests.

Les éléments de conception présentés dans ce document ont été déterminés suite à l'étude du dossier de spécification [V2_SPEC_A1].

Ce dossier de conception suit les recommandations de la norme IEEE 29148 [IEEE-29148 :2018]. Il utilise des schémas et illustrations respectant la norme UML en version 2.5.1 [UML_2.5.1_2017]. Il respecte les exigences du Plan d'Assurance Qualité Logicielle (PAQL) défini par l'équipe A1 [PAQL_A1].

1.2 Portée

Ce document décrit les éléments de conception du Système à l'Étude (SàE). Il est destiné :

- À l'équipe de développement C et celle de développement Android afin de préciser l'implémentation des objets constituant le SàE.
- Aux testeurs, afin qu'ils puissent élaborer les tests adéquats vérifiant la philosophie de conception adaptée par l'équipe.
- Aux auditeurs de la société FORMATO lors de leurs différents suivis du projet.
- Au client pour que le cadre du projet et la direction prise par l'équipe soient claires et dans la continuité des spécifications.

1.3 Définitions, acronymes et abréviations

CdC	Cahier des charges fourni par le client
Client	Société STMicroelectronics
CU	Cas d'utilisation
Disponibilité	La disponibilité est l'aptitude d'un composant ou d'un système à être en état de marche à un instant donné
Fiabilité	La fiabilité est l'aptitude d'un composant ou d'un système à fonctionner pendant un intervalle de temps
IHM	Interface Homme Machine. Moyens permettant aux utilisateurs d'AOP d'interagir avec AOP
Maintenabilité	La maintenabilité est l'aptitude d'un composant ou d'un système à être maintenu ou remis en état de fonctionnement
N.A.	Non Applicable
OMG	(Object Management Group) Association professionelle internationale définissant entre autres des normes dans le domaine informatique
RTC	Une RTC est une horloge temps-réel, c'est un module présent sur la Board
SàÉ	Système à l'Étude. Il s'agit de l'ensemble des logiciels AOP, SoftPorte et SoftSonnette
UML	Notation graphique normalisée, définie par l'OMG et utilisée en génie logiciel

1.4 Références

Document	Référence
[ISO/IEC/IEEE 29148 :2018]	ISO/IEC/IEEE, "International Standard, Systems and software engineering — Life cycle processes — Requirements engineering", version du 30/11/2018 https ://standards.ieee.org/standard/29148-2018.html
[UML_2.5.1_2017]	OMG, "Unified Modeling Language", version 2.5.1 de décembre 2017
[PAQL_A1]	« Plan d'Assurance Qualité Logicielle » version 4.0 du 15/05/2023. Dépôt SE 2024 A1 - ST.doc, chemin d'accès : /qualite/livrable/PAQL_A1.pdf
[V2_SPEC_A1]	« Dossier de spécification » version 2.5 du 31/05/2023 Dépôt SE 2024 A1 - ST.doc, chemin d'accès : /specification/livrables/V2_SPEC_A1.pdf
[ST_Sonnette_connectée_industrielle_2023]	STMicroelectronics, « Cahier des charges initial pour le projet », version original du 09/02/2023. Dépôt SE 2024 A1 - ST.doc, chemin d'accès : /gestion_de_projet/client/ST_Sonnette_connectée_industrielle.pdf
[CR_R2_09_02_2023]	« Compte rendu de la réunion du 09/02/2023 avec les clients ». Dépôt SE 2024 A1 - ST.doc, chemin d'accès : /gestion_de_projet/réunions/CR_R2_09_02_2023.pdf

TABLE 1 – Références des documents utilisés dans ce dossier

1.5 Vue d'ensemble

Ce document de conception est structuré en plusieurs parties :

- La partie I définit l'objet et la portée du document ainsi qu'une liste des abréviations utilisées dans ce document et les références des documents cités.
- La partie II concerne la conception générale du prototype PSC. Cette partie présente l'architecture candidate et donne les grands principes de fonctionnement du projet PSC. Elle détaille ensuite chaque composante du système, en présentant pour chacune leur description structurelle. Une description comportementale est présente pour les composantes actives ayant une machine à états.
- La partie III présentera la conception détaillée. Cette partie présente les composantes du système en précisant cette fois-ci la gestion des entrées et des sorties, le multitâche ainsi que la gestion de la persistance.
- Un dictionnaire de domaine constitue la dernière partie du document.

2 Conception générale

2.1 Architecture candidate

Dans le diagramme suivant est présenté l'architecture candidate du SàE. Les différents objets sont présentes ainsi que leurs méthodes.

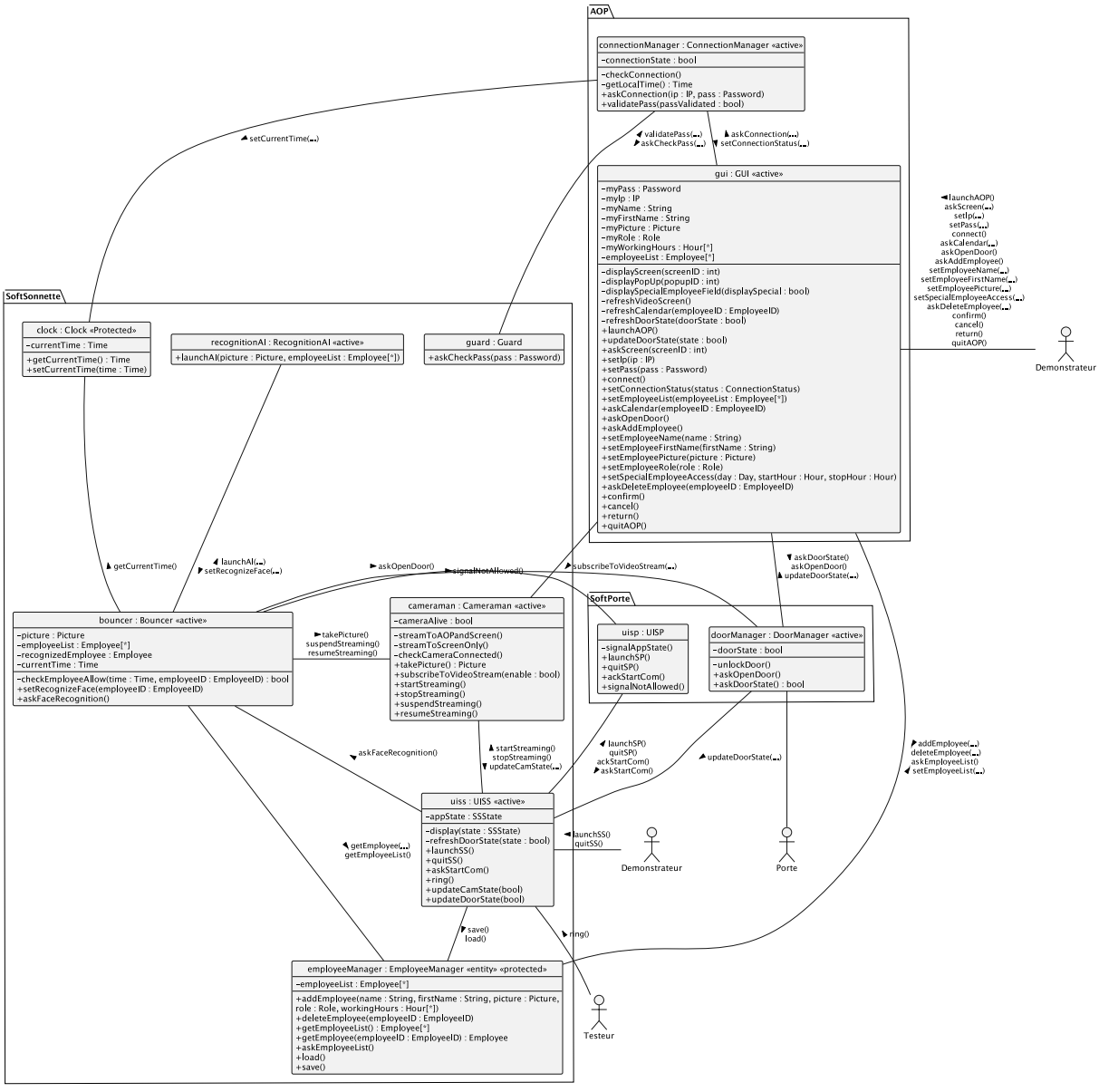

FIGURE 1 – Architecture candidate

Lors de la partie conception générale, est faite l'hypothèse d'un système matériel à ressources infinies.

On distingue dans la représentation de la figure 1 les 3 grandes entités du SàE : AOP, SoftSonnette et SoftPorte.

- AOP est l'application développée pour le téléphone et à destination du Démonstrateur. Elle contient les objets suivants :
  - Connection Manager : Permet de gérer la connexion avec SoftSonnette.
  - GUI : Interface graphique permettant à l'utilisateur d'interagir avec AOP.
- SoftSonnette est l'application exécutée sur le microprocesseur. Elle contient les objets suivants :
  - Employee Manager : Permet de gérer les données des employés.
  - Clock : Permet à SoftSonnette et AOP de synchroniser leur heure.
  - Cameraman : Permet de gérer le flux vidéo.
  - Guard : Vérifie le mot de passe pour la connexion entre SoftSonnette et AOP.
  - Bouncer : Permet de gérer les accès aux Testeurs se présentant devant la Sonnette.
  - UISS : Interface graphique permettant à l'utilisateur d'interagir avec SoftSonnette.
  - RecognitionAI : Permet d'interfacer la librairie de reconnaissance faciale propriétaire du client. La librairie n'étant pas fournie à l'équipe projet, il est de la responsabilité du client d'implémenter dans cet objet les différents appels à sa librairie pour permettre le bon fonctionnement du PSC. L'équipe de développement implémente cet objet en mode "boîte noire".
- SoftPorte est l'application exécutée sur le microcontrôleur. Elle contient les objets suivants :
  - Door Manager : Permet le contrôle de la porte simulée.
  - UISP : Interface physique permettant à l'utilisateur de visualiser l'état de SoftPorte.

2.2 Grands principes de fonctionnement

Cette partie représente le fonctionnement général de SoftSonnette, SoftPorte, AOP et leurs interactions respectives. Les diagrammes de séquences présentés sont :

- Présenter les capacités de la STM32MP15 au travers d'une application de Sonnette Connectée
- Initialiser Board
- Se connecter
- Demander à entrer
- Regarder vidéo
- Consulter calendrier
- Contrôler Porte à distance
- Consulter liste des employés
- Quitter SàE

Les diagrammes de séquence présentent l'histoire nominal des Cas d'Utilisation du dossier de spécification [V2_SPEC_A1]. Ils ne traitent donc pas les erreurs de fonctionnement.

2.2.1 Présenter les capacités de la STM32MP15 au travers d'une application de Sonnette Connectée

Ce diagramme représente le scénario nominal de l'ensemble des CUs du dossier de spécification [V2_SPEC_A1]. Il s'agit du CU Stratégique du SàE.

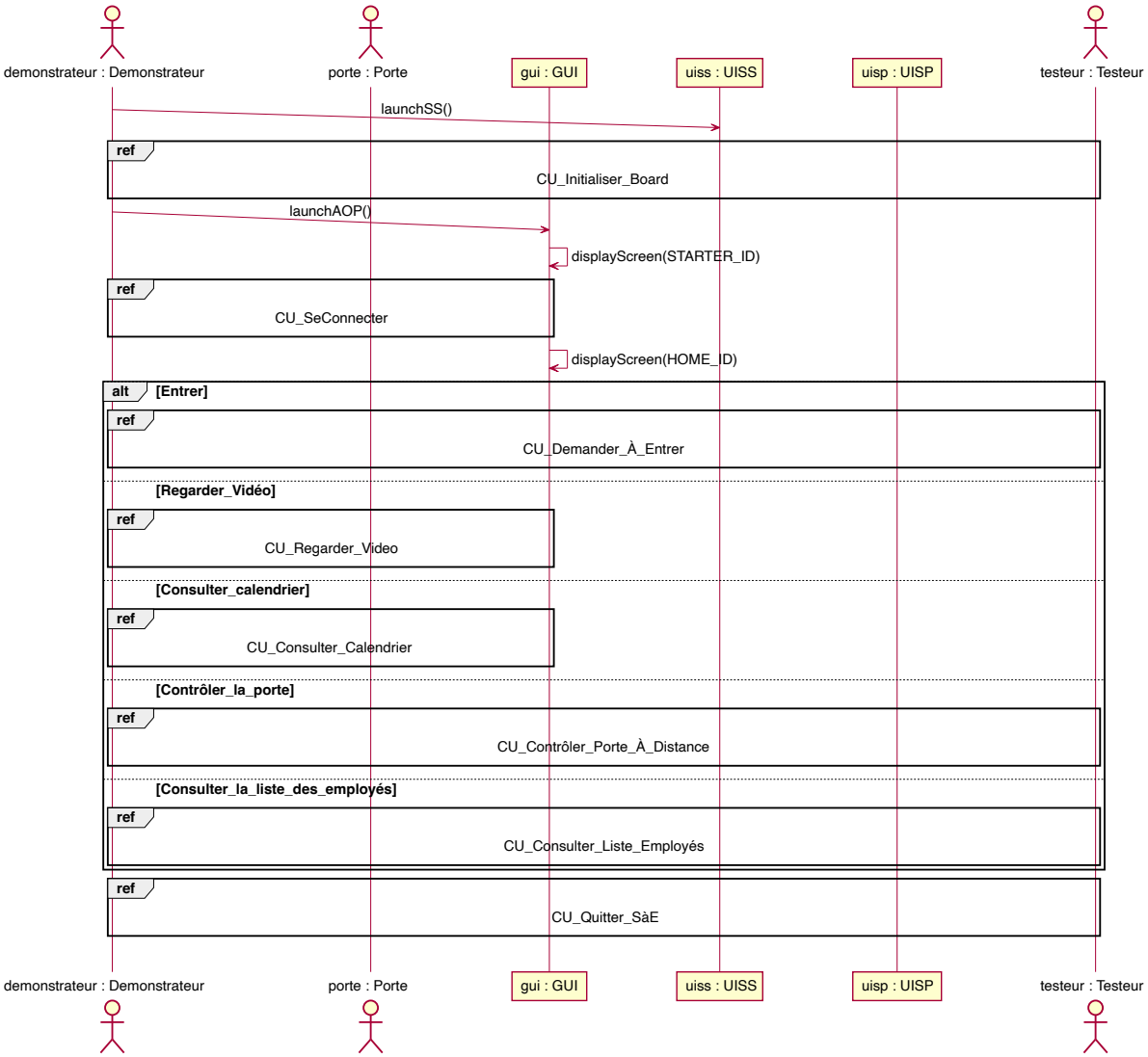

FIGURE 2 – Diagramme de séquence du scénario nominal

2.2.2 Initialiser Board

Ce diagramme représente le scénario nominal du CU "Initialiser Board" dans le dossier de spécification [V2_SPEC_A1].

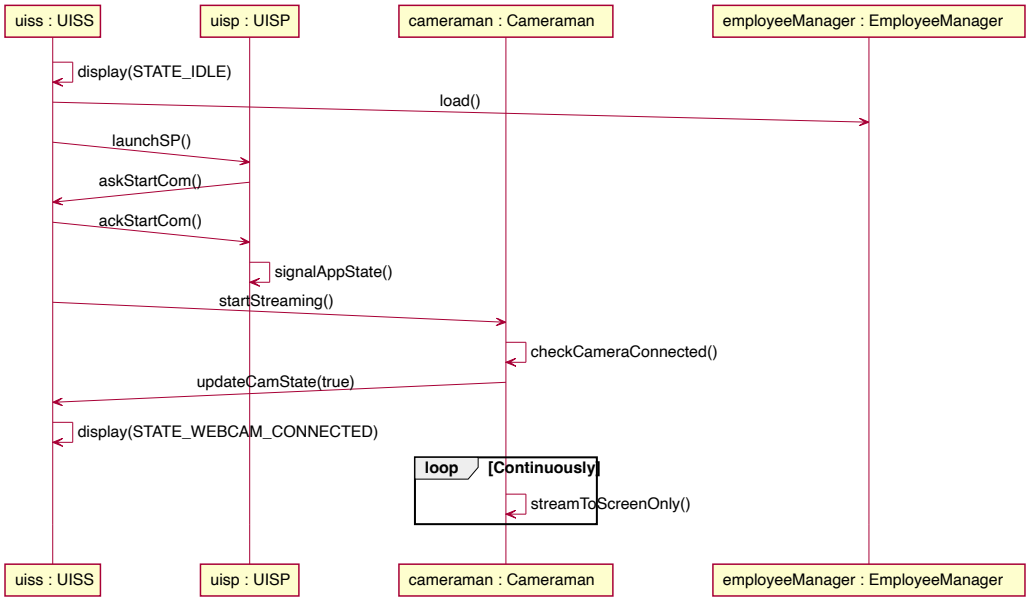

FIGURE 3 – Diagramme de séquence de l'initialisation de Board

2.2.3 Se connecter

Ce diagramme représente le scénario nominal du CU "Se connecter" dans le dossier de spécification [V2_SPEC_A1].

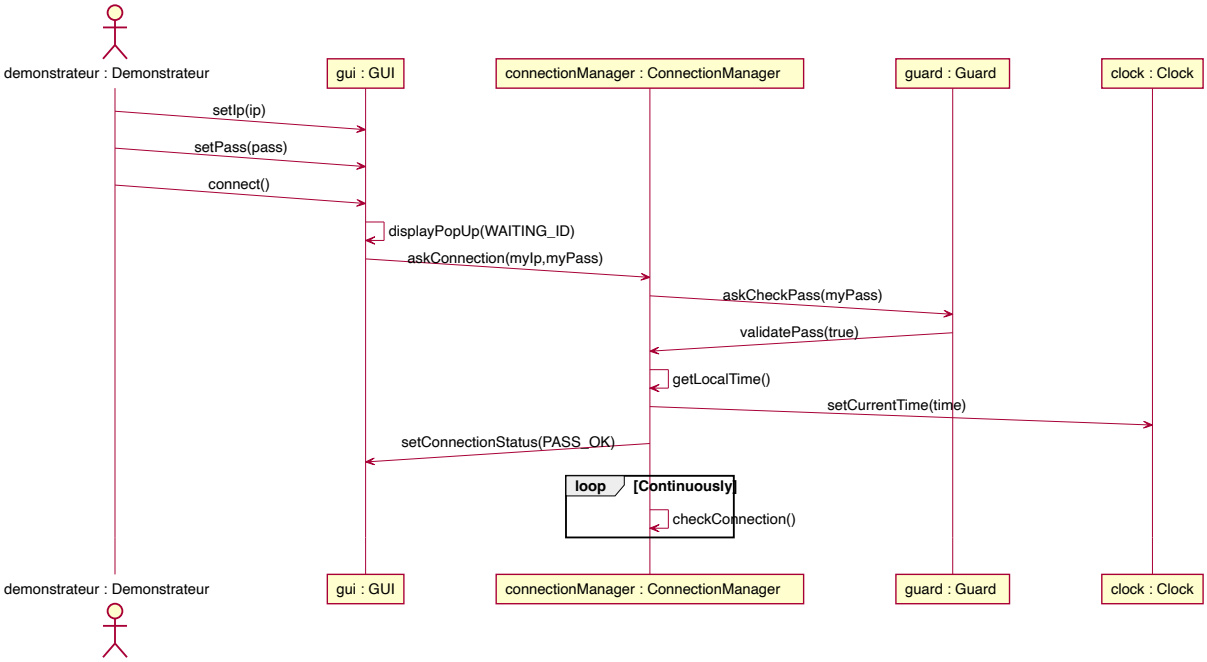

FIGURE 4 – Diagramme de séquence de la connexion entre AOP et SoftSonnette

2.2.4 Demander à entrer

Ce diagramme représente le scénario nominal du CU "Demander à entrer" dans le dossier de spécification [V2_SPEC_A1].

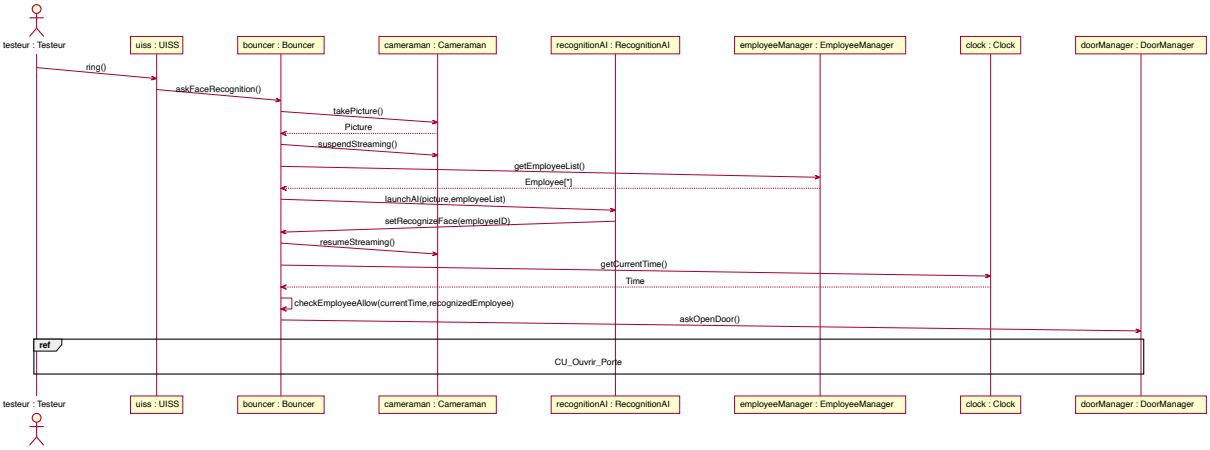

FIGURE 5 – Diagramme de séquence du CU "Demander à entrer"

2.2.5 Ouvrir Porte

Ce diagramme représente le scénario nominal du CU "Ouvrir Porte" dans le dossier de spécification [V2_SPEC_A1].

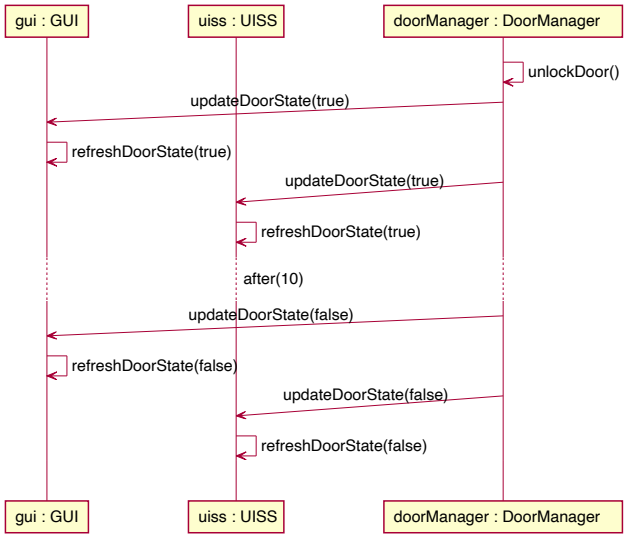

FIGURE 6 – Diagramme de séquence du CU "Ouvrir Porte"

2.2.6 Regarder vidéo

Ce diagramme représente le scénario nominal du CU "Regarder vidéo" dans le dossier de spécification [V2_SPEC_A1].

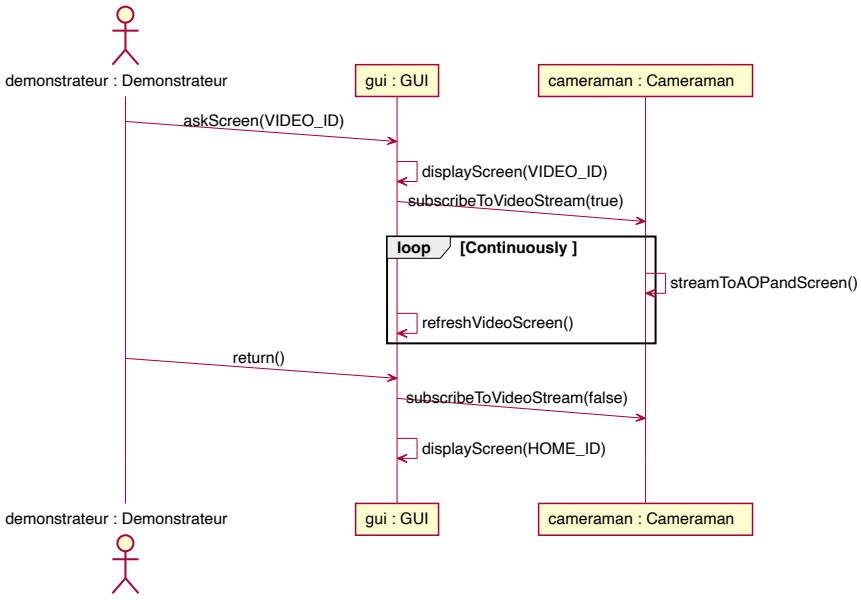

FIGURE 7 – Diagramme de séquence du CU "Regarder vidéo"

2.2.7 Consulter calendrier

Ce diagramme représente le scénario nominal du CU "Consulter calendrier" dans le dossier de spécification [V2_SPEC_A1].

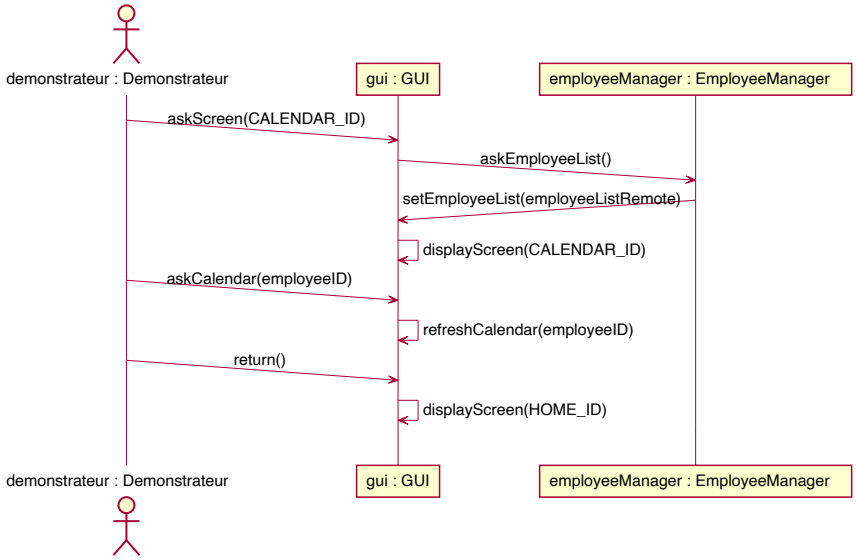

FIGURE 8 – Diagramme de séquence du CU "Consulter calendrier"

2.2.8 Contrôler Porte à distance

Ce diagramme représente le scénario nominal du CU "Contrôle Porte à distance" dans le dossier de spécification [V2_SPEC_A1].

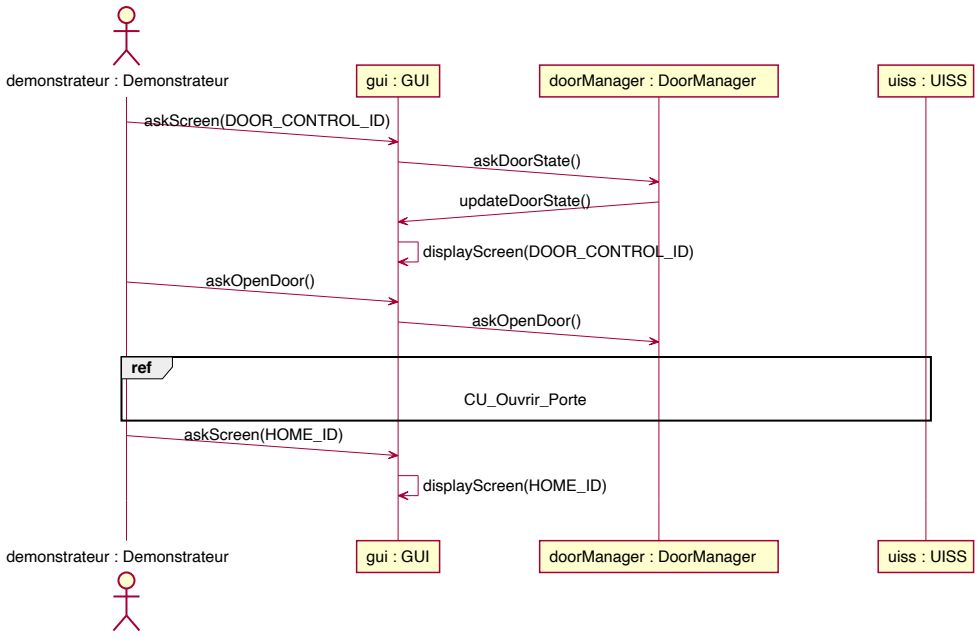

FIGURE 9 – Diagramme de séquence du CU "Contrôle Porte à distance"

2.2.9 Consulter liste employés

Ce diagramme représente le scénario nominal du CU "Consulter liste employés" dans le dossier de spécification [V2_SPEC_A1].

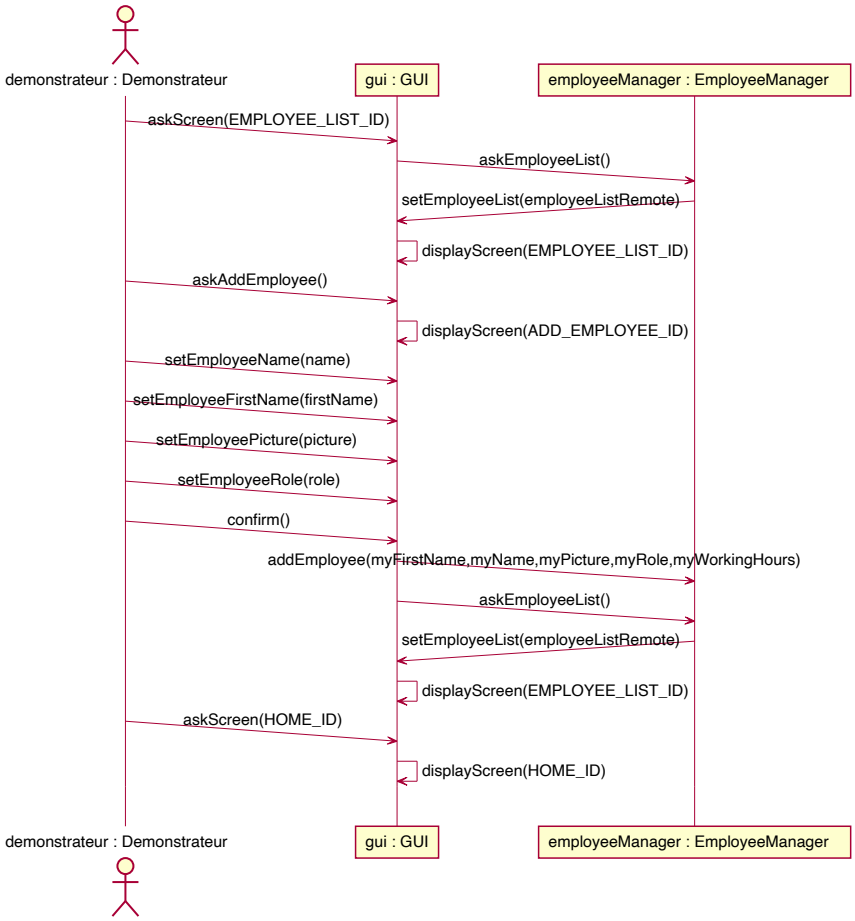

FIGURE 10 – Diagramme de séquence du CU "Consulter liste employés"

2.2.10 Quitter SàE

Ce diagramme représente le scénario nominal du CU "Quitter SàE" dans le dossier de spécification [V2_SPEC_A1].

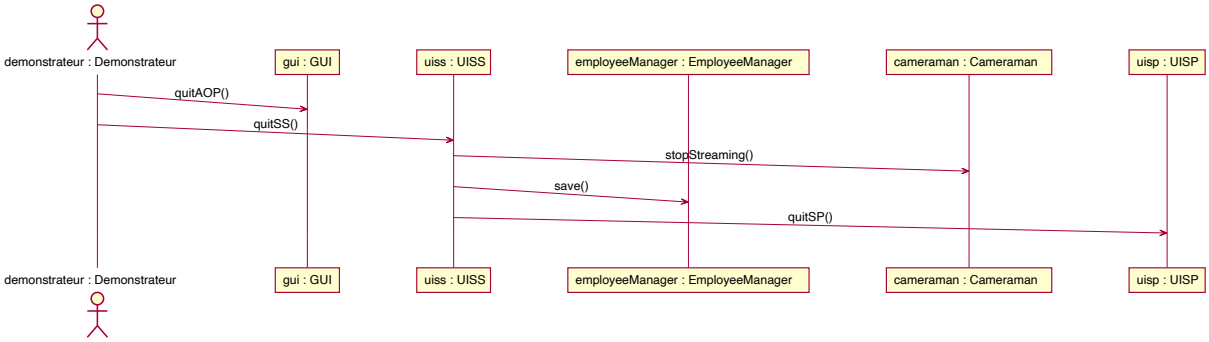

FIGURE 11 – Diagramme de séquence du CU "Quitter SàE"

2.3 Description des composants

2.3.1 Description des types manipulés entre composants

- ConnectionStatus : Énumération représentant l'état de la connexion entre AOP et SoftSonnette. ConnectionStatus peut prendre pour valeur : PASS_KO, CONNECT_KO et ALL_OK.
- Day : Énumération représentant un jour de la semaine. Day peut prendre pour valeur : MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY et SUNDAY.
- *Employee* : Structure contenant les informations de l'employé. Employee contient : firstName , name, role (Role), id (EmployeeID), workingHours [ ] [ ] et picture (chemin local vers l'image).
  - *firstName* est une chaîne de caractère comprenant entre 2 et 12 caractères alphanumériques encodés en UTF-8.
  - *name* suit les mêmes règles que firstName à la différence qu'il peut être comprit entre 1 et 12 caractères alphanumériques.
  - workingHours [ ] [ ] est une matrice de 7 lignes (jours de la semaine) et 2 colonnes (horaires début et fin de journée).
- *EmployeeID* : Un entier qui représente le numéro d'un employé allant jusqu'à MAX_EMPLOYE qui vaut 10. La constante "UNKNOWN" vaut -1, et signifie que l'employé n'est pas reconnu.
- *Hour* : Structure contenant un horaire de début ou de fin de journée, enregistrée au format 24 h avec une précision de 30 minutes. Hour contient l'heure et la minute de l'horaire.
- *Ip* : Chaîne de caractères de type IPV4 au format X.X.X.X où X est un entier compris entre 0 et 255. Correspond à l'adresse de la Board.
- *Password* : Chaîne de caractères. Les caractéristiques sont définies dans le dictionnaire des domaines.
- *Picture* : Image du visage d'un employé qui sera utilisée pour la reconnaissance faciale.
- PopUpID : Énumération représentant les différentes pop-up affichables sur AOP. PopUpID peut prendre pour valeur : WAITING_ID, ERROR_PASS_ID, ERROR_CONNECT_ID, DELETE_ID et VIDEO_NOT_AVAILABLE.
- *Role* : Énumération représentant les différents types d'employés pour délimiter leurs horaires. Role peut prendre pour valeur : E_MORNING, E_DAY, E_EVENING, E_SECURITY et E_SPECIAL.
- ScreenID : Énumération représentant les différents écrans affichables sur AOP. ScreenID peut prendre pour valeur : STARTER_ID, HOME_ID, ADD_EMPLOYEE_ID, ADD_SPECIAL_EMPLOYEE_ID, EMPLOYEE_LIST_ID, DOOR_CONTROL_ID, CALENDAR_ID et VIDEO_ID.
- Time : respecte la norme ISO 8601 : YYYY-MM-DD-HH : MM : SS. Time se met à jour lors de la connexion entre AOP et SoftSonnette.
- SSState : Énumération des affichages de SoftSonnette. SSState peut prendre pour valeur : STATE_IDLE, STATE_WEBCAM_CONNECTED, STATE_WEBCAM_NOT_CONNECTED, STATE_ERROR_COM. À noter que STATE_IDLE correspond à l'affichage vide de l'écran de SoftSonnette (sans affichage vidéo et sans label d'erreur).

2.3.2 Diagramme de classe

Est représenté ci-dessous le diagramme de classe de PSC, composé de 11 classes avec pour chacune leurs attributs et leurs méthodes décrites dans la suite du document.

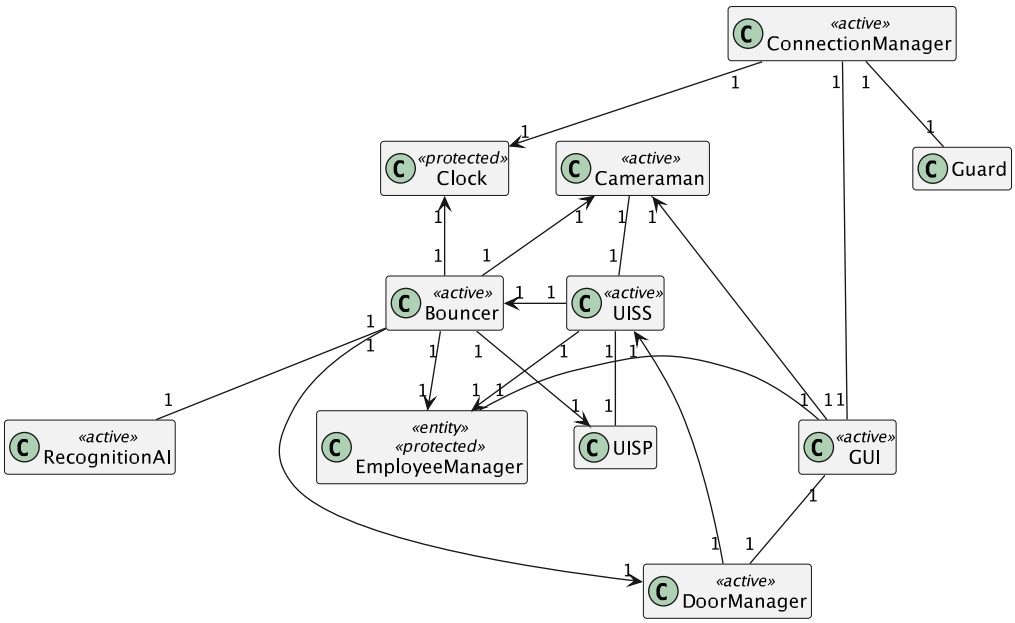

FIGURE 12 – Diagramme de classes du SàE

2.3.3 Description des classes

2.3.3.1 [Object] ConnectionManager

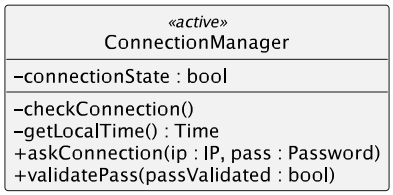

FIGURE 13 – Diagramme de classe représentant ConnectionManager

2.3.3.1.1 Philosophie de conception

La classe ConnectionManager permet de transiter les informations de connexion. Elle permet aussi d'assurer que les applications sont connectées et d'agir en conséquence. Cette classe interagit avec Guard, Clock, et GUI.

2.3.3.1.2 Description structurelle

2.3.3.1.2.1 Attributs

- connectionState : bool : Exprime l'état de la connexion actuelle.

2.3.3.1.2.2 Services offerts

- checkConnection() : void : Tâche de fond qui vérifie si la connexion est toujours opérationnelle. Dans le cas d'une perte de connexion cette fonction informe les objets intéressés.
- getLocalTime() : Time : Récupère l'heure et la date du téléphone
- askConnection(ip : IP, pass : Password) : void : Envoie les informations de connexion vers Guard.
- validatePass(passValidated : bool) : void : Reçoit l'information de connexion qui valide ou non la connexion.

2.3.3.1.3 Description comportementale

La MAE suivante permet d'expliquer le fonctionnement de l'objet actif ConnectionManager :

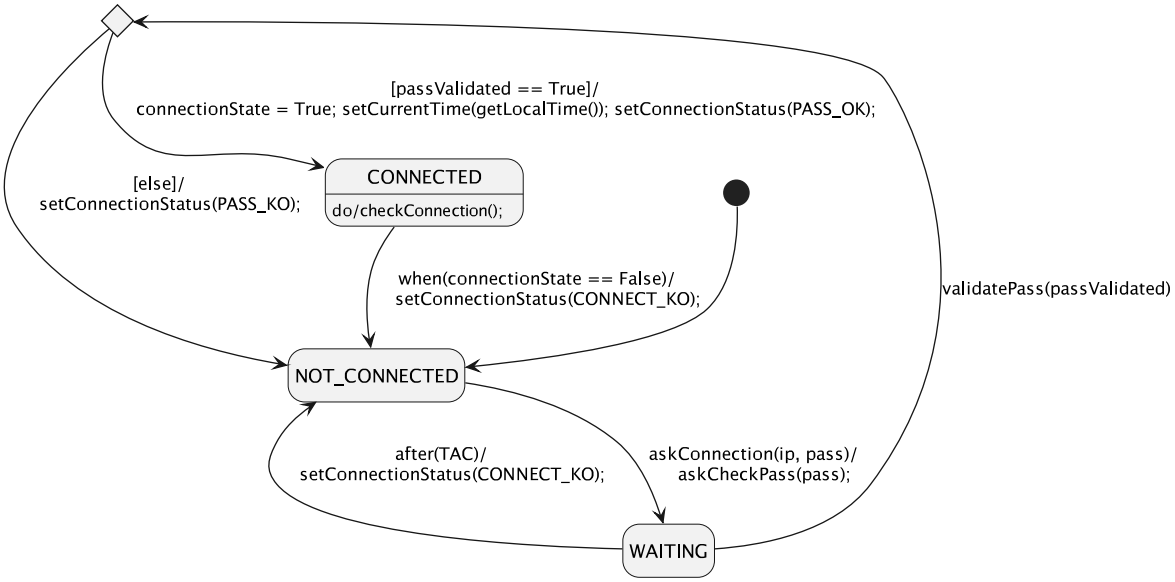

FIGURE 14 – Machine à état représentant ConnectionManager

2.3.3.2 [IHM] GUI

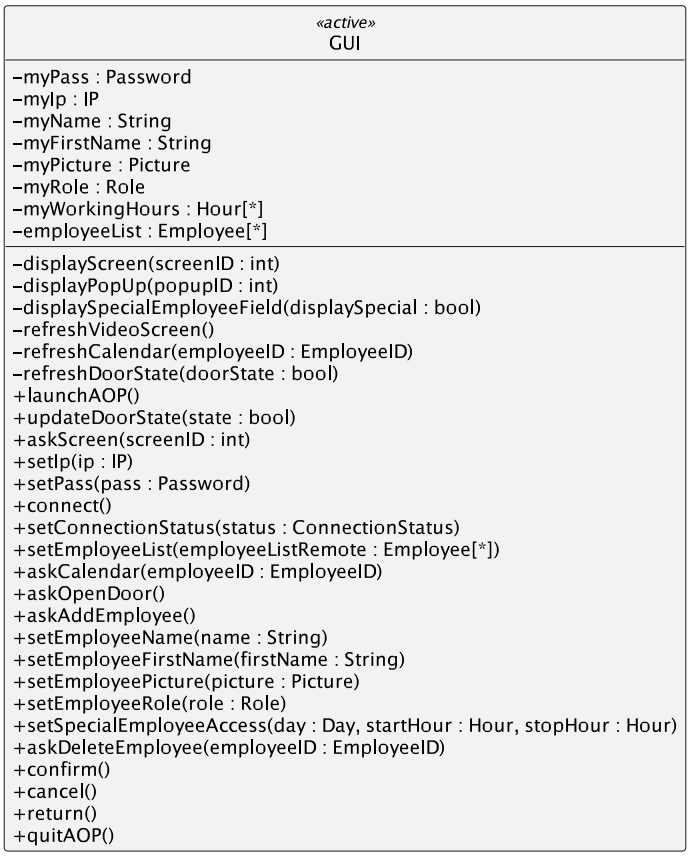

FIGURE 15 – Diagramme de classe représentant GUI

2.3.3.2.1 Philosophie de conception

La classe GUI permet de gérer les interfaces utilisateurs entre le Démonstrateur et AOP en affichant les différentes vues de AOP. Cette classe interagit avec ConnectionManager, DoorManager, EmployeeManager et Cameraman.

2.3.3.2.2 Description structurelle

2.3.3.2.2.1 Attributs

- myPass : Password : Contient le mot de passe entré lors de la connexion.
- myIP : IP : Contient l'adresse IP entrée lors de la connexion.
- myName : String : Contient le nom entré lors de l'ajout d'un employé.
- myFirstName : String : Contient le prénom entré lors de l'ajout d'un employé.
- myPicture : Picture : Contient la photo d'un employé à ajouter.
- myRole : Role : Contient le rôle d'un employé lors de son ajout.
- myWorkingHours : Hour[] : Contient les horaires de l'employé pour les 7 jours de la semaine.
- employeeList : Employee[] : Contient la liste des employés envoyés depuis la Board.

2.3.3.2.2.2 Services offerts

- displayScreen(screenID : ScreenID) : void : Affiche l'écran entré en paramètre de la fonction.
- displayPopUp(popUpID : PopUpID) : void : Affiche la PopUp entrée en paramètre de la fonction.
- displaySpecialEmployeeField(displaySpecial : bool) : void : Affiche les options afin d'ajouter les horaires d'un employé spécial.
- refreshVideoScreen() : Tâche de fond en charge de lire le flux vidéo entrant et de l'afficher sur l'écran permettant de regarder la vidéo.
- refreshCalendar(employeeID : EmployeeID) : void : Met à jour l'affichage du calendrier en fonction de l'ID de l'employé à afficher.
- refreshDoorState(doorState : bool) : Met à jour l'affichage de l'état de la Porte sur l'écran de contrôle à distance de la Porte en fonction du paramètre de type booléen.
- launchAOP() : void : Lance AOP.
- updateDoorState(doorState : bool) : void : Évènement pour demander à AOP de mettre à jour l'état de la Porte sur l'écran de contrôle à distance de la Porte.
- askScreen(screenID : ScreenID) : void : Démonstrateur demande à afficher l'écran entré en paramètre de la fonction.
- setIP(ip : IP) : void : Modifie l'attribut myIP avec le paramètre de la fonction.
- setPass(pass : Password) : void : Modifie l'attribut myPass avec le paramètre de la fonction.
- connect() : void : Demande la connexion.
- setConnectionStatus(status : ConnectionStatus) : void : Vérifie si la connexion est établie via le paramètre de la fonction status de type ConnectionStatus.
- askCalendar(employeeID : EmployeeID) : void : Demande d'afficher le calendrier correspondant au paramètre de la fonction de type EmployeeID.
- setEmployeeList(employeeListRemote : Employee[]) : void : Retour asynchrone permettant de mettre à jour la copie de la liste des employés de AOP.
- askOpenDoor() : void : Demande l'ouverture de la Porte.
- askAddEmployee() : void : Demande à ajouter un employé.
- setEmployeeName(name : String) : void : Modifie l'attribut myName avec le paramètre de la fonction.
- setEmployeeFirstName(firstName : String) : void : Modifie l'attribut myFirstName avec le paramètre de la fonction.
- setEmployeePicture(picture : Picture) : void : Modifie l'attribut myPicture avec le paramètre de la fonction.
- setEmployeeRole(role : Role) : void : Modifie l'attribut myRole avec le paramètre de la fonction.
- setSpecialEmployeeAccess(day : Day, startHour : Hour, stopHour : Hour) : void : Entre les horaires de l'employé spécial. Avec comme paramètres day, startHour et stopHour qui prend en compte le jour, le début et de la fin de la journée de l'employé. Ces paramètres permettent la mise à jour de myWorkingHours.
- askDeleteEmployee(employeeID : EmployeeID) : void : Demande la suppression d'un employé. L'employé est déterminé par le paramètre de la fonction.
- confirm() : void : Confirme les modifications faites.
- cancel() : void : Annule les modifications effectuées.
- return() : void : Retourne à l'écran précédent.
- quitAOP() : void : Quitte AOP.

2.3.3.2.3 Description comportementale

Les MAE suivantes permettent d'expliquer le fonctionnement de l'objet actif GUI :

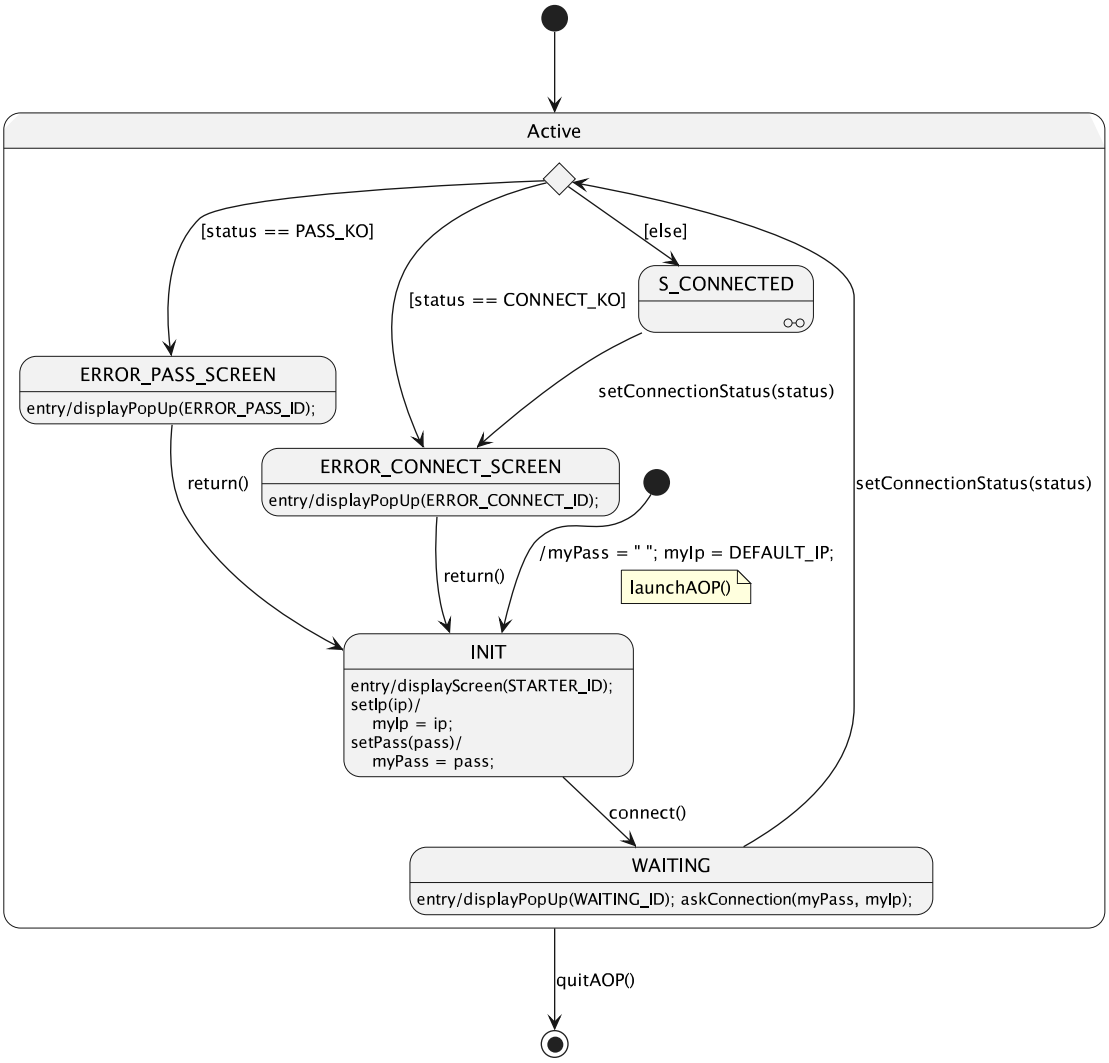

FIGURE 16 – Machine à état représentant GUI

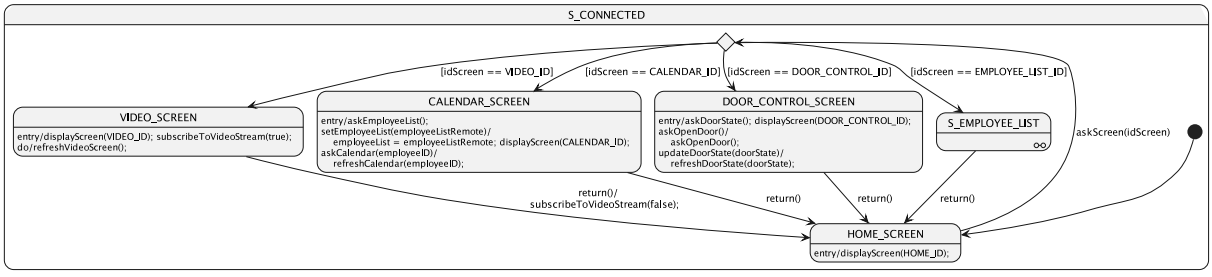

FIGURE 17 – Machine à état représentant GUI

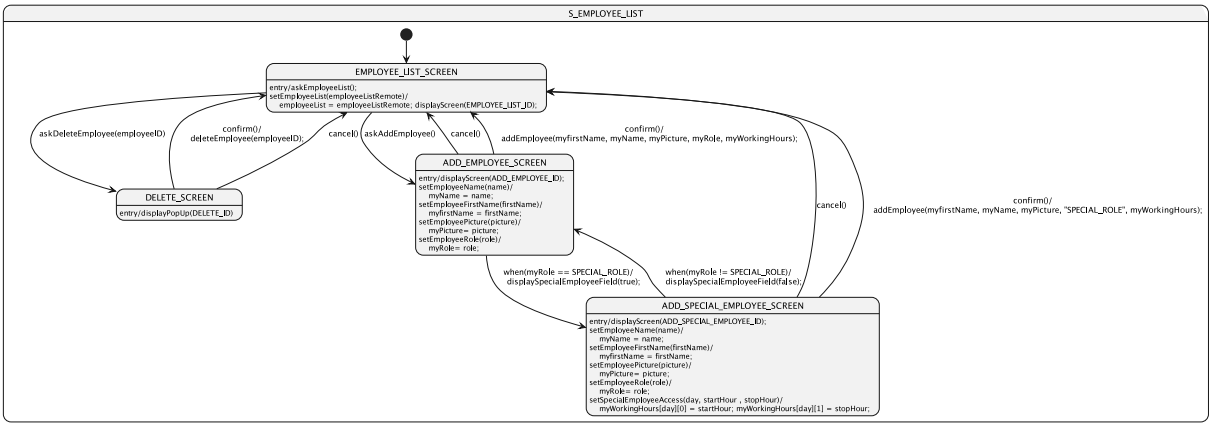

FIGURE 18 – Machine à état représentant employé liste

2.3.3.3 [Object] EmployeeManager

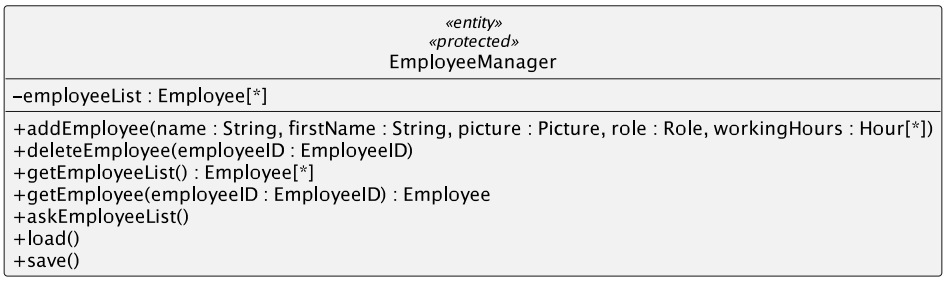

FIGURE 19 – Diagramme de classe représentant EmployeeManager

2.3.3.3.1 Philosophie de conception

La classe EmployeeManager répertorie la liste des employés enregistrés dans l'application. Cet objet gère l'ajout et la suppression d'employés et interagit avec GUI, Bouncer.

2.3.3.3.2 Description structurelle

2.3.3.3.2.1 Attributs

- employeeList : Employee[] : Contient la liste des employés. Il s'agit d'une copie des données persistantes chargée en mémoire.

2.3.3.3.2.2 Services offerts

- addEmployee(name : String, firstName : String, picture : Picture, role : String, workingHours : Hour[]) : void : Ajoute un employé à la liste à partir de son nom, prénom, sa photo, son rôle et les horaires de travail qui lui sont liés.
- deleteEmployee(employeeID : EmployeeID) : void : Supprime l'employé lié à l'identifiant employeeID passé en paramètre.
- getEmployeeList() : Employee[] : Renvoie la liste des employés à SoftSonnette.
- getEmployee(employeeID : EmployeeID) : Employee : Retourne l'objet employé lié à l'identifiant employeeID passé en paramètre.
- askEmployeeList() : Employee[] : Renvoie la liste des employés à AOP.
- load() : void : Charge les données persistantes des informations employés en mémoire vive.
- save() : void : Sauvegarde les données persistantes en mémoire non volatile.

2.3.3.4 [Object] Clock

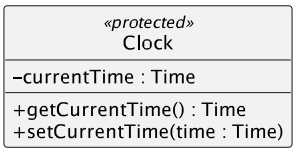

FIGURE 20 – Diagramme de classe représentant Clock

2.3.3.4.1 Philosophie de conception

La classe Clock est en charge du temps. Elle synchronise les heures entre AOP et SoftSonnette. Elle est aussi utilisée pour savoir si un employé peut entrer en fonction de ses horaires attribués.

2.3.3.4.2 Description structurelle

2.3.3.4.2.1 Attributs

- currentTime : Time : Contient le temps courant.

2.3.3.4.2.2 Services offerts

- getCurrentTime() : Time : Permet de donner l'heure courante pour la comparaison des heures dans Bouncer.
- setCurrentTime(time : Time) : void : Permet de modifier l'heure courante de la Board.

2.3.3.5 [Object] Cameraman

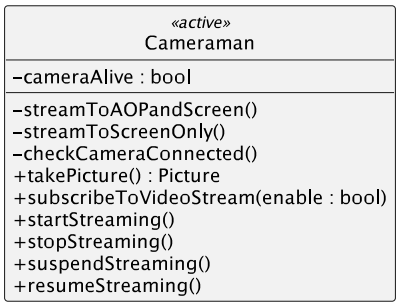

FIGURE 21 – Diagramme de classe de Cameraman

2.3.3.5.1 Philosophie de conception

La classe Cameraman interface E_Caméra. Il permet de prendre une image du Testeur (pour qu'il se fasse reconnaître) et de capturer le flux vidéo retransmis sur l'écran de SoftSonnette et sur l'AOP.

2.3.3.5.2 Description structurelle

2.3.3.5.2.1 Attributs

- cameraAlive : bool : Indique si la caméra est connectée ou non.

2.3.3.5.2.2 Services offerts

- streamToAOPandScreen() : void : Tâche de fond permettant de streamer la vidéo sur AOP et sur l'écran de SoftSonnette.
- streamToScreenOnly() : void : Tâche de fond permettant de streamer la vidéo sur l'écran de SoftSonnette.
- checkCameraConnected() : void : Vérifie la présence de la caméra.
- takePicture() : Picture : Fonction passive qui prend une photo de ce que voit Cameraman. Cette photo est utilisée pour reconnaître le visage d'un Testeur.
- subscribeToVideoStream(enable : bool) : void : Fonction appelée par AOP pour demander à Cameraman de lui Streamer la vidéo.
- startStreaming() : void : Permet de démarrer le streaming.
- stopStreaming() : void : Stop définitivement le streaming - Sortie de la MàE.
- suspendStreaming() : void : Permet de mettre en pause le streaming.
- resumeStreaming() : void : Permet reprendre le streaming.

2.3.3.5.3 Description comportementale

La MAE suivante permet d'expliquer le fonctionnement de l'objet actif Cameraman :

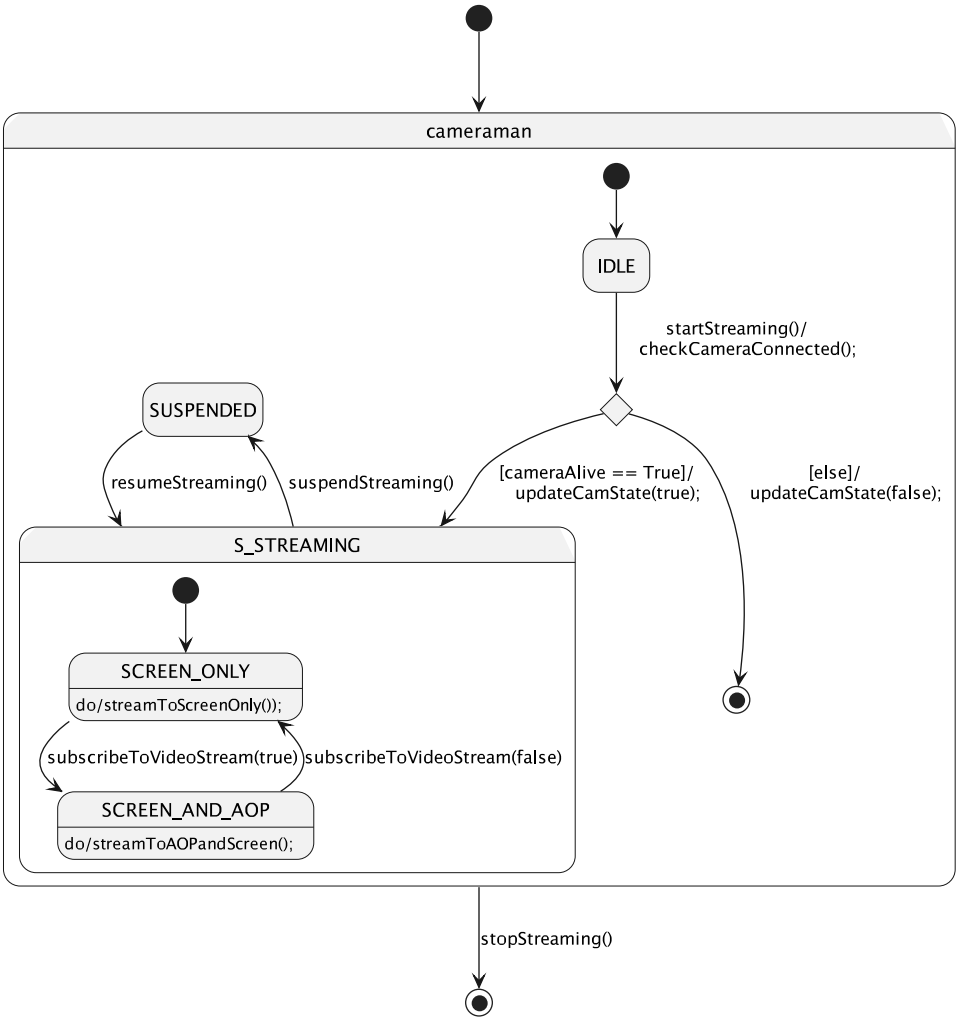

FIGURE 22 – Machine à États de Cameraman

2.3.3.6 [Object] Guard

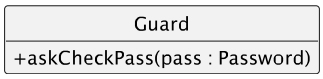

FIGURE 23 – Diagramme de classe de Guard

2.3.3.6.1 Philosophie de conception

La classe Guard est le gardien du mot de passe permettant à AOP d'interagir avec SoftSonnette. Il vérifie que le mot de passe reçu est celui codé en dur dans l'application puis autorise la connexion si ce dernier est correct.

2.3.3.6.2 Description structurelle

2.3.3.6.2.1 Attributs

N.A.

2.3.3.6.2.2 Services offerts

- askCheckPass(pass : Password) : void : Permet de demander à Guard de vérifier le mot de passe passé en paramètre.

2.3.3.7 [Object] Bouncer

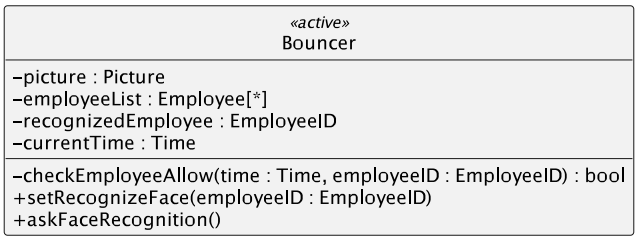

FIGURE 24 – Diagramme de classe représentant Bouncer

2.3.3.7.1 Philosophie de conception

La classe Bouncer permet au SàE d'autoriser ou non un Testeur à entrer. En interaction avec Cameraman elle déclenche la prise d'une photo du Testeur se présentant à la sonnette. Elle interagit également avec EmployeeManager pour récupérer la liste des employés. Elle déclenche la reconnaissance faciale avec la classe RecognitionAI à qui elle envoie toutes les données nécessaires. Elle vérifie qu'un employé reconnu ait l'autorisation d'entrer sur l'horaire courant qu'elle récupère auprès de la classe Clock. Elle demande enfin l'ouverture de la Porte à DoorManager ou signale à UISP que le Testeur n'est pas reconnu ou non autorisé.

2.3.3.7.2 Description structurelle

2.3.3.7.2.1 Attributs

- picture : Picture : Photo du Testeur à reconnaître.
- employeeList : Employee[] : Liste courante des employés.
- recognizedEmployee : EmployeeID : ID de l'employé reconnu par RecognitionAI.
- currentTime : Time : Temps utilisé pour vérifier que l'employé est autorisé à entrer.

2.3.3.7.2.2 Services offerts

- checkEmployeeAllow(time : Time, employeeID : EmployeeID) : bool : À partir du temps courant et de l'ID de l'employé, elle vérifie si l'employé est autorisé à entrer à l'horaire courant.
- setRecognizeFace(employeeID : EmployeeID) : void : Appelée par RecognitionAI pour retourne l'employé reconnu par l'IA, ou UNKNOWN s'il n'y a pas de correspondance.
- askFaceRecognition() : void : Trigger permettant de déclencher le processus reconnaissance facial et d'autorisation d'entrée.

2.3.3.7.3 Description comportementale

La MAE suivante permet d'expliquer le fonctionnement de l'objet actif Bouncer :

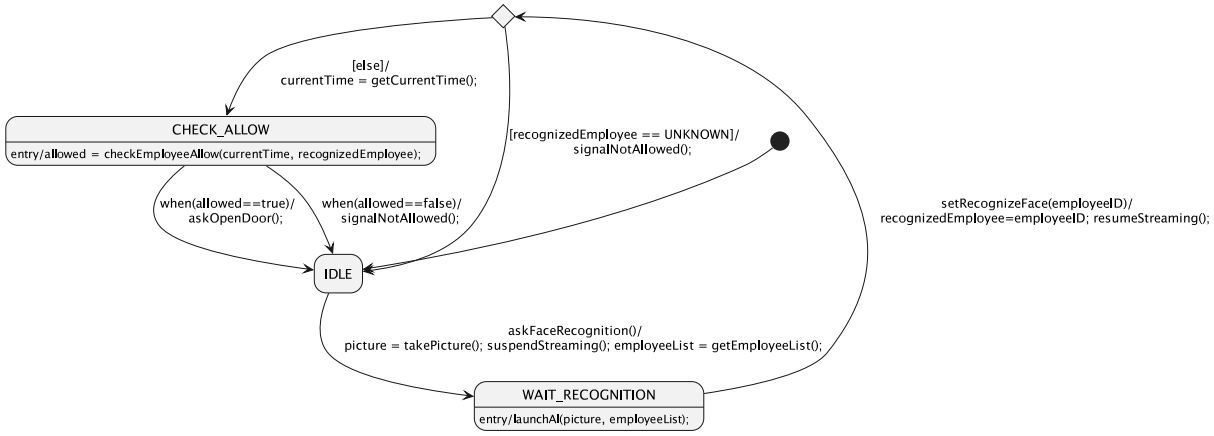

FIGURE 25 – Machine à États de Bouncer

2.3.3.8 [IHM] UISS

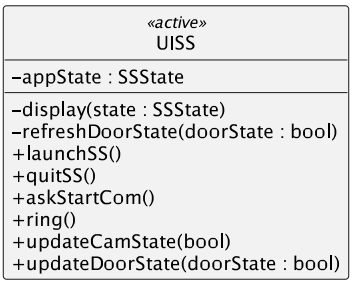

FIGURE 26 – Diagramme de classe d'UISS

2.3.3.8.1 Philosophie de conception

La classe UISS s'occupe de la gestion de l'interface utilisateur entre le Testeur , le Démonstrateur et SoftSonnette. Elle permet l'affichage de la vidéo, l'état de la Porte mais aussi au Testeur de sonner et au Démonstrateur de quitter l'application.

2.3.3.8.2 Description structurelle

2.3.3.8.2.1 Attributs

- appState : SSState : Variable de mise a jour de l'écran qui prend trois valeurs possibles.

2.3.3.8.2.2 Services offerts

- display(state : SSState) : void : Affiche l'écran de SoftSonnette avec la variante transmise par le paramètre state.
- refreshDoorState(doorState : bool) : Met à jour l'affichage de l'état de la Porte sur l'écran de SoftSonnette en fonction du paramètre de type booléen passé.
- launchSS() : void : Démarre l'application SoftSonnette.
- quitSS() : void : Ferme l'application SoftSonnette.
- askStartCom() : void : Évènement reçu de la part de SoftPorte qui demande l'établissement de la communication.
- ring() : void : Testeur sonne.
- updateCamState(bool) : void : Évènement en provenance de Cameraman indiquant à UISS l'état de la caméra en fonction du paramètre de type booléen passé. UISS met ensuite à jour l'écran avec un appel à la fonction display().
- updateDoorState(doorState : bool) : void : Évènement en provenance de DoorManager indiquant à UISS l'état de la Porte en fonction du paramètre de type booléen passé. UISS met ensuite à jour le label indiquant l'état de la Porte sur l'écran avec un appel à la fonction refreshDoorState().

2.3.3.8.3 Description comportementale

La MAE suivante permet d'expliquer le fonctionnement de l'objet actif UISS.

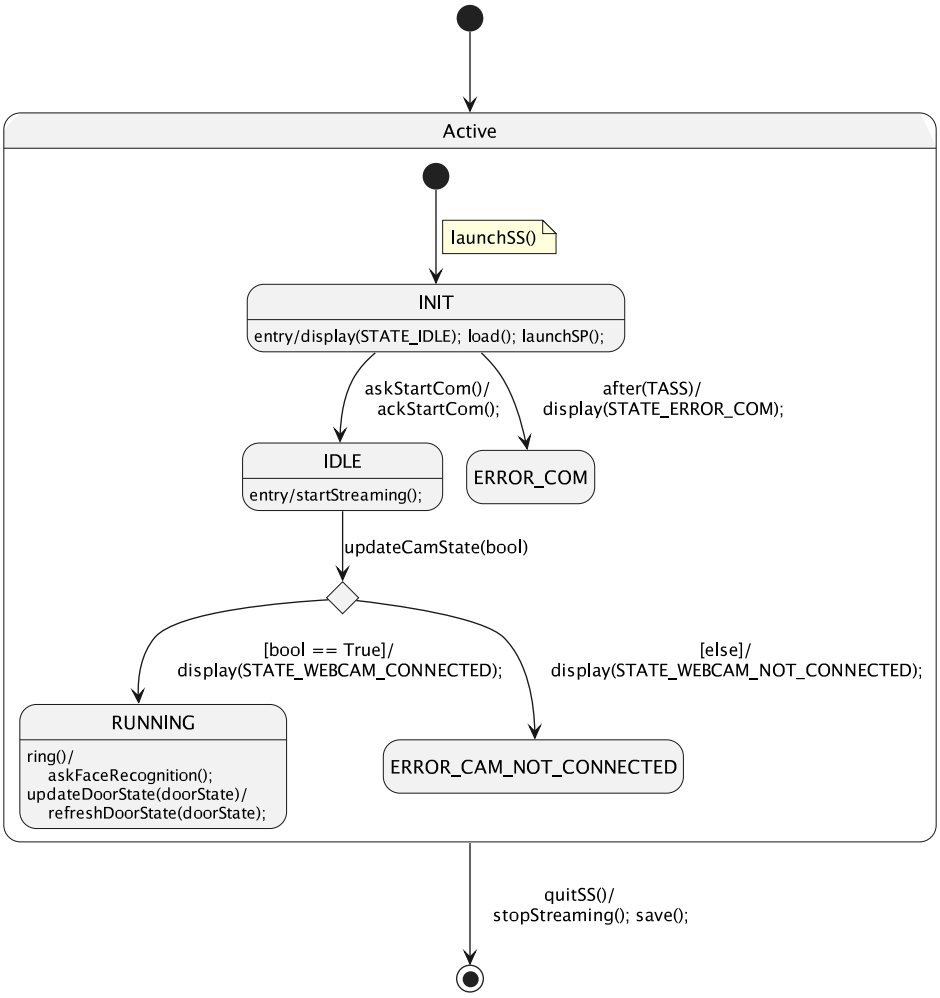

FIGURE 27 – Machine à États d'UISS

2.3.3.9 [Object] DoorManager

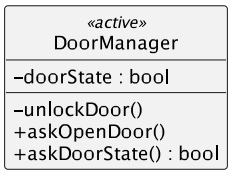

FIGURE 28 – Diagramme de classe représentant DoorManager

2.3.3.9.1 Philosophie de conception

La classe DoorManager est en charge du contrôle de la Porte simulée. Elle permet de commander son déverrouillage, et de fournir l'état de cette dernière.

2.3.3.9.2 Description structurelle

2.3.3.9.2.1 Attributs

- doorState : bool : Indique l'état de la Porte (ouvert/fermé).

2.3.3.9.2.2 Services offerts

- unlockDoor() : void : Déverrouille la Porte pendant *TOP*.
- askOpenDoor() : void : Envoie la demande d'ouverture de la Porte vers UISP.
- askDoorState() : void : Demande l'état de la porte à UISP (retour asynchrone).

2.3.3.9.3 Description comportementale

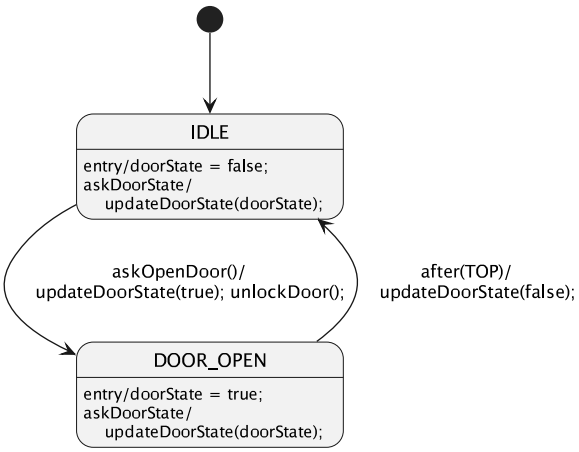

FIGURE 29 – Machine à état représentant DoorManager

2.3.3.10 [IHM] UISP

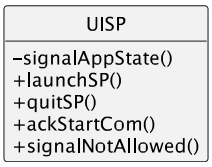

FIGURE 30 – Diagramme de classe d'UISP

2.3.3.10.1 Philosophie de conception

La classe UISP permet de mettre à jour les différents éléments physiques de la Board. Elle met en place la communication entre SoftPorte et SoftSonnette, signale son bon démarrage en agissant sur la LED LD5 verte et signale lorsqu'un Testeur n'est pas autorisé à entrer avec la LED LD6 rouge.

2.3.3.10.2 Description structurelle

2.3.3.10.2.1 Attributs

N.A.

2.3.3.10.2.2 Services offerts

- signalAppState() : void : Signale le lancement de l'application en allumant la LED verte.
- launchSP() : void : Lance SoftPorte.
- quitSP() : void : Quitte SoftPorte.
- signalNotAllowed() : void : Signale l'interdiction d'entrer en allumant la LED rouge.
- ackStartCom() : void : Confirme l'initialisation de la communication entre SoftPorte et SoftSonnette. Cette fonction appel ensuite signalAppState() si la communication est établie avec succès.

2.3.3.11 [Object] RecognitionAI

FIGURE 31 – Diagramme de classe de RecognitionAI

2.3.3.11.1 Philosophie de conception

La classe RecognitionAI interface la librairie client de reconnaissance faciale. En lui fournissant la photo d'un Testeur, celle-ci détermine si le Testeur en question est autorisé à entrer. Cet objet est appelé par Bouncer lorsqu'une demande d'ouverture est faite. Une fois la reconnaissance faciale terminée, RecognitionAI envoie à Bouncer le résultat.

2.3.3.11.2 Description structurelle

2.3.3.11.2.1 Attributs

N.A.

2.3.3.11.2.2 Services offerts

- launch(picture : Picture, employeeList : Employee[]) : void : Lance le processus de reconnaissance faciale.

3 Conception détaillée

La conception détaillée définit l'architecture logique du système, c'est à dire :

- Répartir le système sur l'architecture matérielle.
- Gérer les entrées et sorties, ainsi que les IHM.
- Élaborer les protocoles de communication.
- Définir les parallélismes et l'initialisation.
- Définir le démarrage, l'arrêt et la destruction du SàE.

3.1 Architecture détaillée

Les diagrammes ci-dessous représentent le SàE dans son intégralité. En plus des classes vues précédemment, ce diagramme présente la gestion de la communication ainsi que les objets frontières nécessaires au bon fonctionnement du logiciel.

Ces diagrammes, dans un souci de lisibilité, ne détaillent pas les méthodes ainsi que les attributs des classes. Ces derniers sont détaillés dans les diagrammes de classe qui suivent.

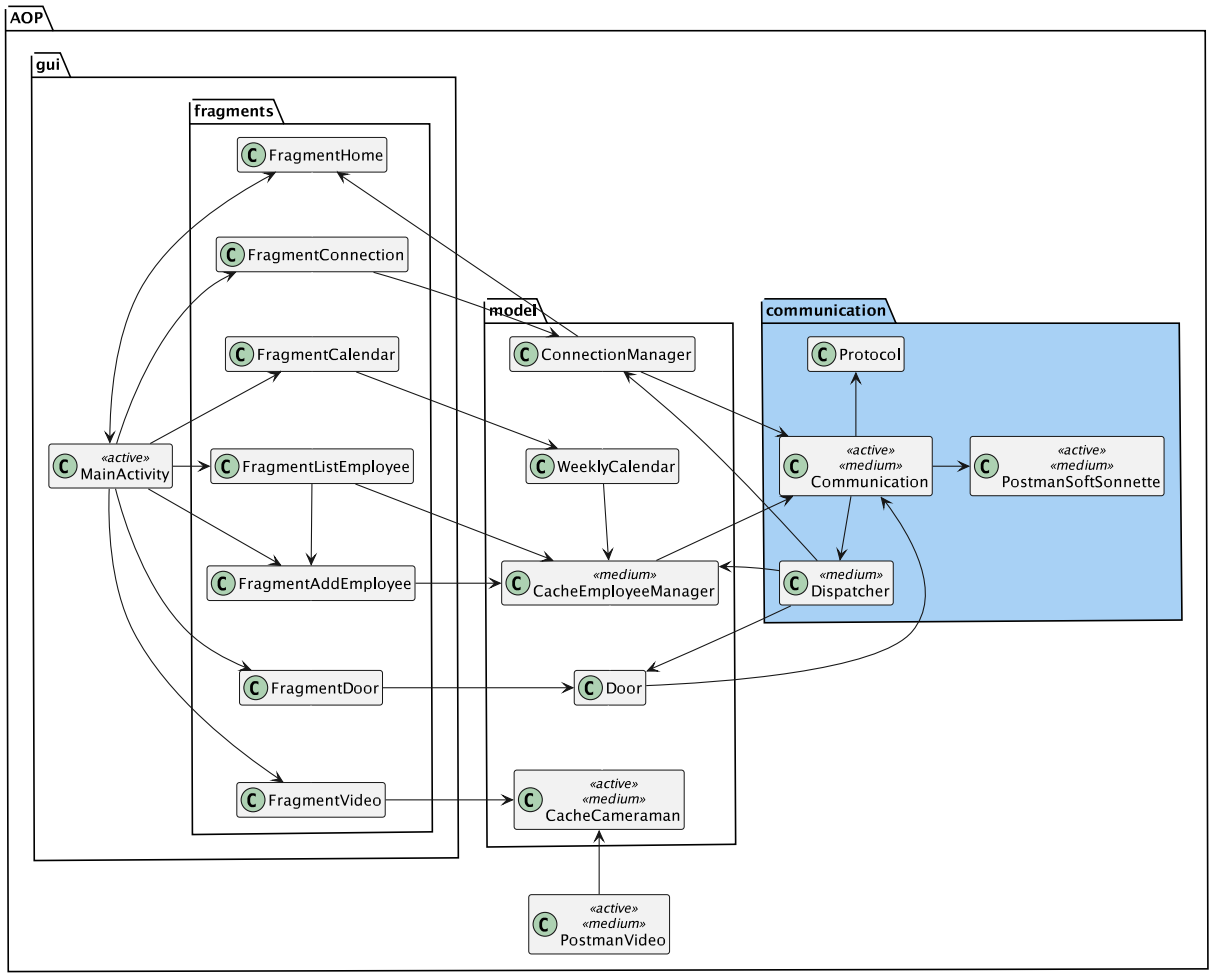

FIGURE 32 – Architecture candidate détaillée de AOP représentée par un diagramme de communication

Il est a noté que le classe GUI de la conception générale a été divisée en plusieurs Fragment représentant les différents écrans. Ces fragments sont regroupés dans le package gui.

AOP adopte une stratégie de communication en utilisant une classe Communication, Protocol, Postman et Dispatcher. A l'inverse, SoftSonnette et SoftPorte adoptent une stratégie avec des Proxys, Postamn, Protcol et Dispatcher. Des détails sur ces deux stratégies sont proposés dans la partie 3.3. "Protocole de communication".

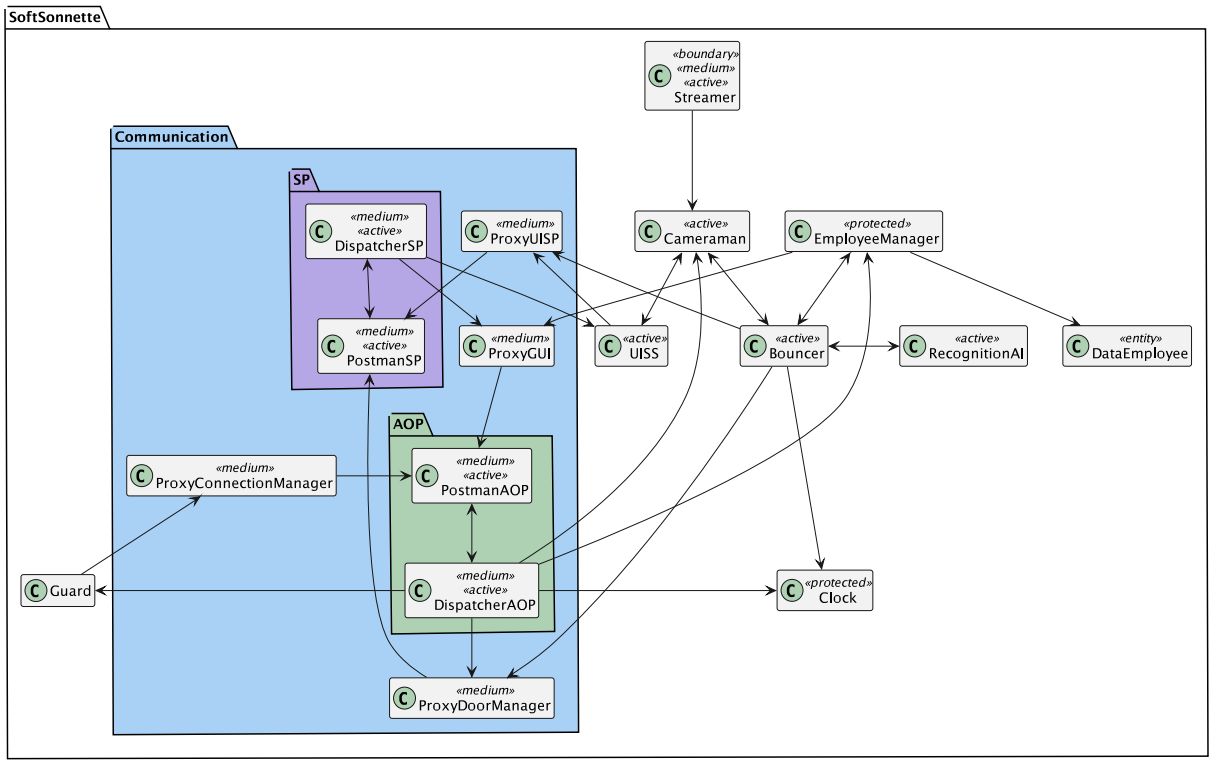

FIGURE 33 – Architecture candidate détaillée de SoftSonnette représentée par un diagramme de communication

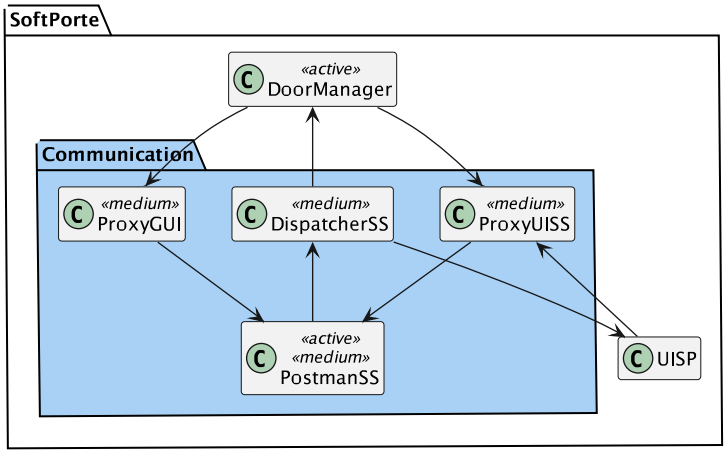

FIGURE 34 – Architecture candidate détaillée de SoftPorte représentée par un diagramme de communication

Les objets Starter ne sont pas représentés dans les diagrammes de communication et sont détaillés dans la suite du document.

3.2 Description des classes

3.2.1 Description des classes de SoftSonnette

3.2.1.1 [Object] Starter

FIGURE 35 – Diagramme de classe de Starter

3.2.1.1.1 Philosophie de Conception

La classe Starter a pour rôle d'instancier tous les objets actifs utilisés par SoftSonnette et de démarrer leur machine à état respective. Les objets instanciés sont les suivants :

- Cameraman
- UISS
- PostmanAOP
- DispatcherAOP
- Bouncer

La classe Starter permet aussi de stopper les machines à état et détruire tous les objets instanciés lors de l'arrêt de SoftSonnette.

3.2.1.1.2 Description structurelle

3.2.1.1.2.1 Attributs

N.A.

3.2.1.1.2.2 Services offerts

- main() : int : La méthode initiale appelée au lancement de l'application SoftSonnette. Elle instancie et démarre les objets indiqués ci-dessus.
- cleanup() : void : Détruit les objets après l'arrêt de l'application et libère la mémoire.

3.2.1.1.3 Diagramme de séquence du démarrage et de l'arrêt de SoftSonnette

Le diagramme suivant représente la séquence d'initialisation de SoftSonnette.

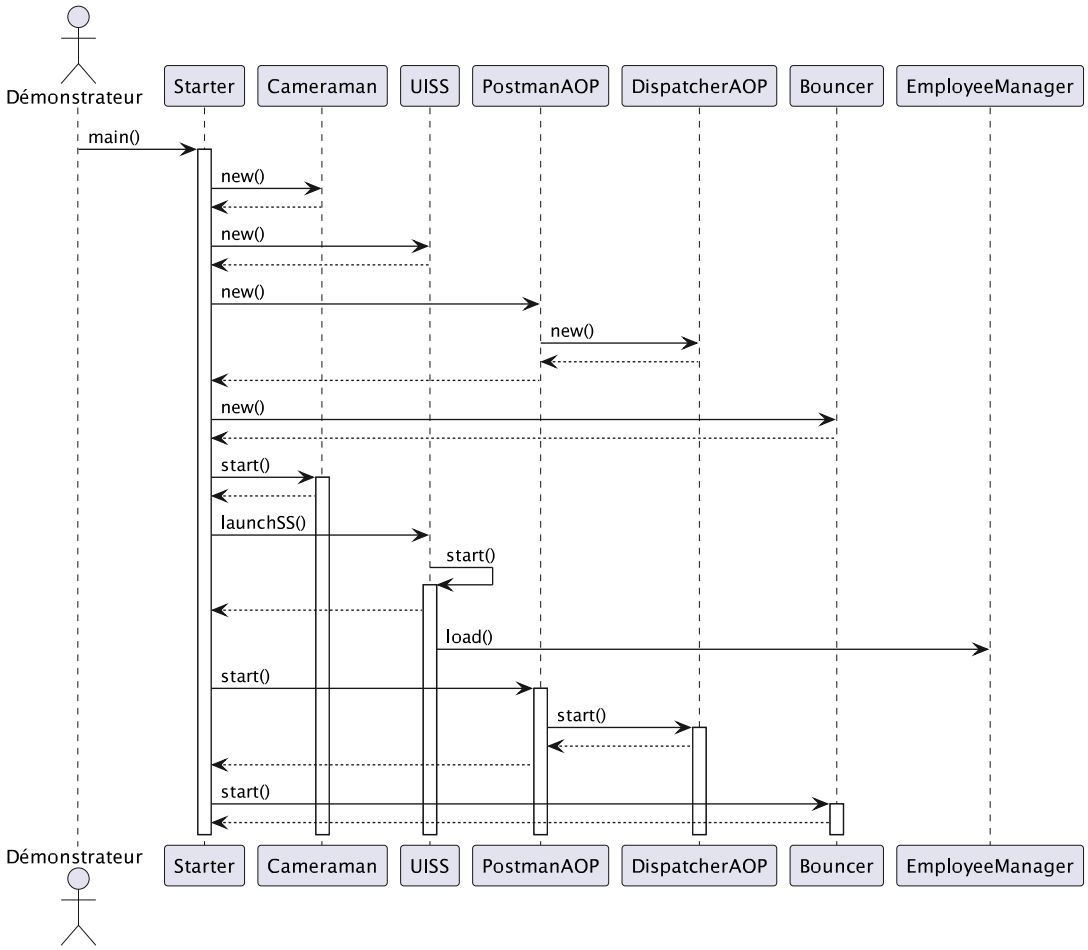

FIGURE 36 – Diagramme de séquence de l'initialisation de SoftSonnette

Le diagramme suivant représente la séquence d'arrêt de SoftSonnette.

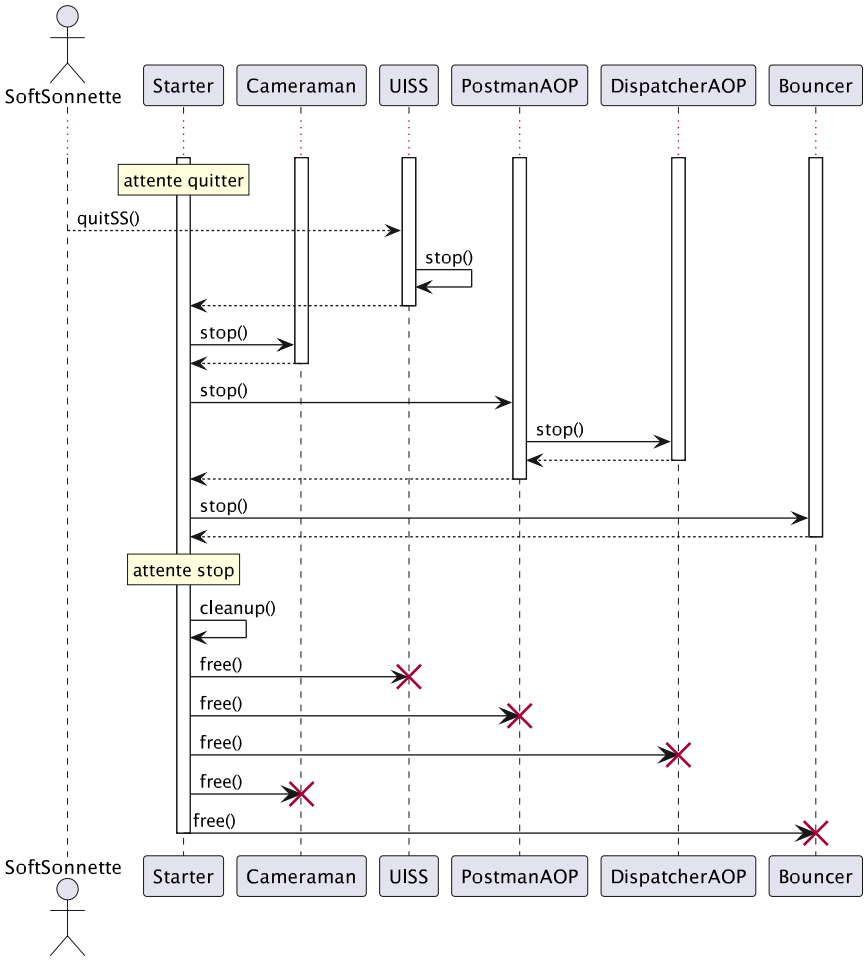

FIGURE 37 – Diagramme de séquence de l'arrêt de SoftSonnette

3.2.1.2 [Medium] ProxyGUI

FIGURE 38 – Diagramme de classe de ProxyGUI

3.2.1.2.1 Philosophie de Conception

La classe ProxyGUI dans SoftSonnette simule le comportement de la classe GUI et gère l'encodage de la trame pour la communication vers AOP. Cette classe sert également d'intermédiaire pour la communication entre SoftPorte et AOP en ré-encodant les messages en provenance de SoftPorte et à destination de AOP. Elle gère la communication d'une seule méthode entre DoorManager et GUI, à travers DispatcherSP et PostmanAOP.

3.2.1.2.2 Description structurelle

3.2.1.2.2.1 Attributs

N.A.

3.2.1.2.2.2 Services offerts

- updateDoorState(doorState : bool) : void : Met à jour l'affichage de l'état de la Porte sur AOP selon l'état state.

3.2.1.3 [Medium] ProxyConnectionManager

FIGURE 39 – Diagramme de classe de ProxyConnectionManager

3.2.1.3.1 Philosophie de Conception

La classe ProxyConnectionManager simule le comportement de la classe ConnectionManager, et permet la communication entre Guard et ConnectionManager en passant par l'objet PostmanAOP.

3.2.1.3.2 Description structurelle

3.2.1.3.2.1 Attributs

N.A.

3.2.1.3.2.2 Services offerts

- validatePass(passValidated : bool) : void : Méthode permettant d'envoyer le résultat de la vérification de la connexion à AOP en passant par PostmanAOP, en l'encodant.

3.2.1.4 [Medium][Boundary] Streamer

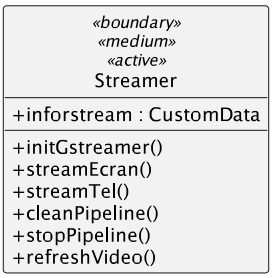

FIGURE 40 – Diagramme de classe de Streamer

3.2.1.4.1 Philosophie de Conception

La classe Streamer est un objet frontière actif. Il permet l'envoie du flux vidéo depuis SoftSonnette vers AOP. Streamer est basé sur la librairie Linux open-source Gstreamer.

3.2.1.4.2 Description structurelle

3.2.1.4.2.1 Attributs

- inforStream : CustomData : Variable stockant les données du flux vidéo flux

CustomData est une structure permettant de créer le flux video. CustomData contient les informations sur le bus, le pipeline, l'état, l'état de la boucle de rafraîchissement et la valeur du flux video.

3.2.1.4.2.2 Services offerts

- initGstreamer() : void : Méthode permettant d'initialiser les instances pour utiliser Gstreamer.
- streamEcran() : void : Méthode permettant d'afficher le flux video sur l'écran de SoftSonnette.
- streamTel() : void : Méthode permettant d'afficher le flux video sur l'écran de SoftSonnette et d'AOP.
- cleanPipeline() : void : Méthode permettant de nettoyer le pipeline du flux video.
- stopPipeline() : void : Méthode permettant d'arrêter le flux vidéo.
- refreshVideo : void : Méthode permettant de gérer les différents messages d'interruption de Gstreamer.

3.2.1.5 [Medium] ProxyUISP

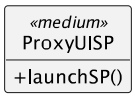

FIGURE 41 – Diagramme de classe de ProxyUISP

3.2.1.5.1 Philosophie de Conception

La classe ProxyUISP simule le comportement de la classe UISP, et permet la communication entre Bouncer et UISP en passant par l'objet PostmanSP.

3.2.1.5.2 Description structurelle

3.2.1.5.2.1 Attributs

N.A.

3.2.1.5.2.2 Services offerts

- launchSP() : void : Méthode utilisée pour démarrer SoftPorte.

3.2.1.6 [Medium] ProxyDoorManager

FIGURE 42 – Diagramme de classe de ProxyDoorManager

3.2.1.6.1 Philosophie de Conception

La classe ProxyDoorManager simule le comportement de la classe DoorManager. Elle permet ainsi la communication entre Bouncer et DoorManager en passant par l'objet PostmanSP.

3.2.1.6.2 Description structurelle

3.2.1.6.2.1 Attributs

N.A.

3.2.1.6.2.2 Services offerts

- askOpenDoor() : void : Méthode utilisée pour demander l'ouverture la Porte.
- getDoorState() : bool : Méthode utilisée pour savoir si la Porte est actuellement ouverte ou fermée.

3.2.1.7 [Medium] PostmanSP

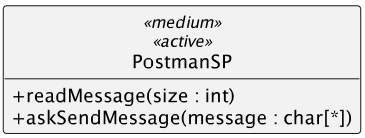

FIGURE 43 – Diagramme de classe de PostmanSP

3.2.1.7.1 Philosophie de Conception

La classe PostmanSP est utilisée pour permettre à SoftSonnette d'envoyer des messages à SoftPorte.

3.2.1.7.2 Description structurelle

3.2.1.7.2.1 Attributs

3.2.1.7.2.2 Services offerts

- readMessage(size : int) : void : Réception d'un message provenant de SoftPorte.
- askSendMessage(message : char*) : void : Envoi d'un message à SoftPorte.

3.2.1.8 [Object] ProtocolSS

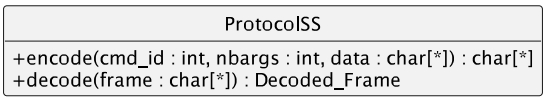

FIGURE 44 – Diagramme de classe de ProtocolSS

3.2.1.8.1 Philosophie de conception

La classe ProtocolSS gère les fonctions d'encodage et de décodage des trames lors de la communication entre AOP et SoftSonnette, côté SoftSonnette.

3.2.1.8.2 Description structurelle

3.2.1.8.2.1 Attributs

N.A.

3.2.1.8.2.2 Services offerts

- encode(cmd_id : int, nbargs : int, data : char*) : char* : Fonction haut niveau d'encodage de la trame. Construit la trame en utilisant les fonctions intermédiaires d'encodage.
- decode(frame : char[]) : Decoded_Frame : Fonction de décodage de la trame. Reconstruit la trame via une instance de la structure Decoded_Frame.

3.2.1.9 [Object] ProtocolSP

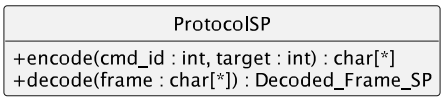

FIGURE 45 – Diagramme de classe de ProtocolSP

3.2.1.9.1 Philosophie de conception

L'objet ProtocolSP gère les fonctions d'encodage et de décodage des trames lors de la communication entre SoftPorte et SoftSonnette, côté SoftSonnette.

3.2.1.9.2 Description structurelle

3.2.1.9.2.1 Attributs

N.A.

3.2.1.9.2.2 Services offerts

- encode(cmd_id : int, target : int) : char* : Fonction haut niveau d'encodage de la trame. Construit la trame en utilisant les fonctions intermédiaires d'encodage.
- decode(frame : char[]) : Decoded_Frame_SP : Fonction de décodage de la trame. Reconstruit la trame via une instance de la structure Decoded_Frame_SP.

3.2.1.10 [Medium] PostmanAOP

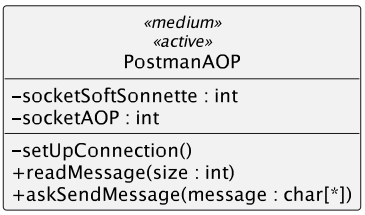

FIGURE 46 – Diagramme de classe de PostmanAOP

3.2.1.10.1 Philosophie de Conception

La classe PostmanAOP a pour objectif de gérer l'envoi et la réception de messages à travers les sockets de communication entre SoftSonnette et AOP. C'est aussi la classe qui met en place le serveur de cette communication. Elle est en interaction avec les proxys de SoftSonnette et DispatcherAOP pour l'encodage et le décodage des trames.

3.2.1.10.2 Description structurelle

3.2.1.10.2.1 Attributs

- socketSoftSonnette : int : Socket serveur qui écoute les connexions d'AOP sur le port 1234.
- socketAOP : int : Socket client utilisé par AOP pour communiquer avec SoftSonnette.

3.2.1.10.2.2 Services offerts

- readMessage(size : int) : void : Réception d'un message provenant d'AOP.
- askSendMessage(message : char[]) : void : Envoi d'un message à AOP via la boîte aux lettres.
- setUpConnection() : void : Initialise le socket serveur et attend la connexion du client.

3.2.1.10.3 Gestion du multitâche

La machine à état de PostmanAOP ainsi que l'attente de la connexion du client s'exécutent dans des threads séparés lancés à partir de la méthode start() par le thread principal. Les évènements tels que l'envoi d'un message ou l'arrêt du service sont transmis à partir d'une boîte aux lettres.

3.2.1.11 [Medium] DispatcherAOP

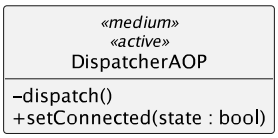

FIGURE 47 – Diagramme de classe de DispatcherAOP

3.2.1.11.1 Philosophie de Conception

L'objet DispatcherAOP a pour objectif de lire les messages reçus depuis le facteur afin de les décoder et de les renvoyer à l'objet SoftSonnette correspondant.

3.2.1.11.2 Description structurelle

3.2.1.11.2.1 Attributs

N.A.

3.2.1.11.2.2 Services offerts

- dispatch() : void : Décode le message lu sur le facteur de SoftSonnette, puis envoie le message à l'objet concerné.
- setConnected(state : bool) : void : Indique au dispatcher le statut de connexion.

3.2.1.12 [Medium] DispatcherSP

FIGURE 48 – Diagramme de classe de DispatcherSP

3.2.1.12.1 Philosophie de Conception

DispatcherSP est l'objet permettant de recevoir et décoder les messages en provenance de l'objet PostmanSP. Ces messages sont ensuite envoyés à UISS ou à ProxyGUI.

3.2.1.12.2 Description structurelle

3.2.1.12.2.1 Attributs

N.A.

3.2.1.12.2.2 Services offerts

- dispatch() : void : Décode le message lu sur le facteur de SoftPorte, puis envoie le message à l'objet UISS ou à l'objet ProxyGUI.

3.2.1.13 [Entity] DataEmployee

FIGURE 49 – Diagramme de classe de DataEmployee

3.2.1.13.1 Philosophie de Conception

L'entité DataEmployee représente les données persistantes des employés. Cette entité est interfacée uniquement par EmployeeManager.

3.2.1.13.2 Description structurelle

3.2.1.13.2.1 Attributs

N.A.

3.2.1.13.2.2 Services offerts

N.A.

3.2.2 Description des classes de AOP

3.2.2.1 [Object] StarterAOP

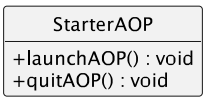

FIGURE 50 – Diagramme de classe de StarterAOP

3.2.2.1.1 Philosophie de Conception

La classe StartAOP permet d'instancier les autres objets de AOP. La classe StartAOP est reliée à MainActivity.

3.2.2.1.2 Description structurelle

3.2.2.1.2.1 Attributs

N.A.

3.2.2.1.2.2 Services offerts

- launchAOP() : void : Démarrage des instanciations.
- quitAOP() : void : Détruit les instanciations.

3.2.2.1.3 Diagramme de séquence du démarrage et de l'arrêt de AOP

Le diagramme suivant représente la séquence d'initialisation d'AOP.

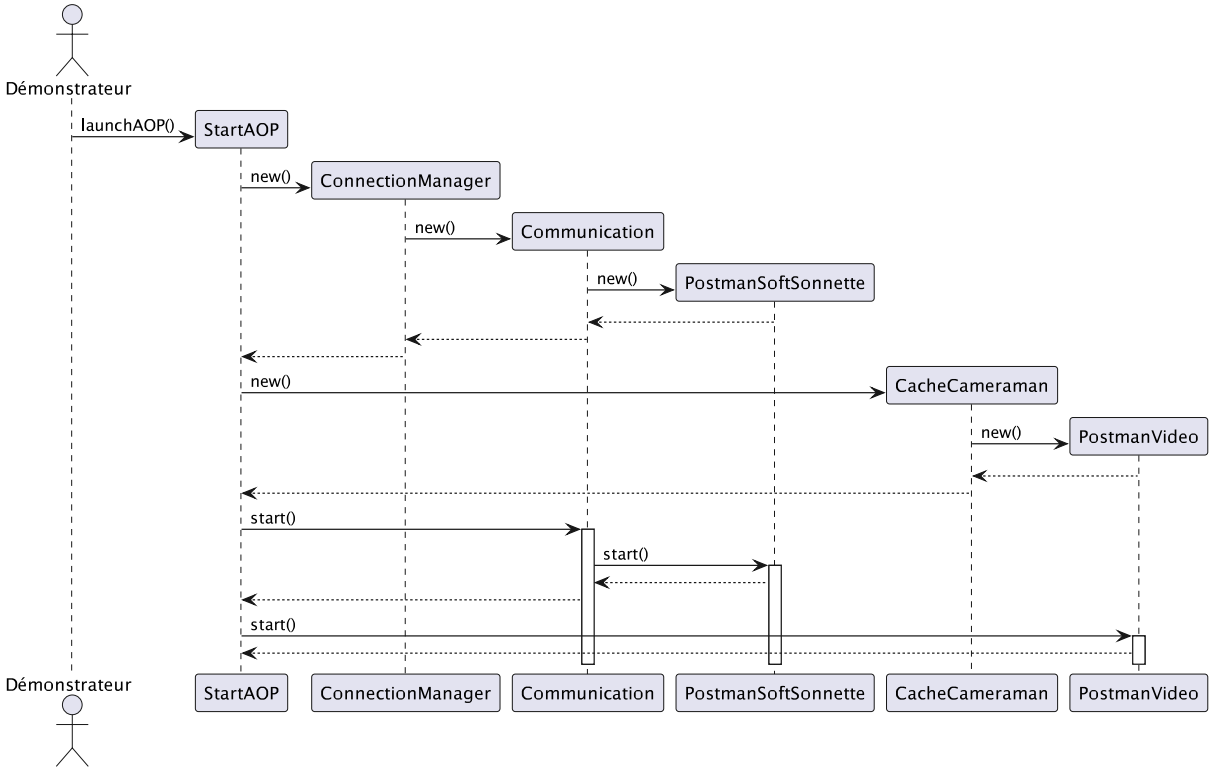

FIGURE 51 – Diagramme de séquence de l'initialisation d'AOP

Le diagramme suivant représente la séquence d'arrêt d'AOP.

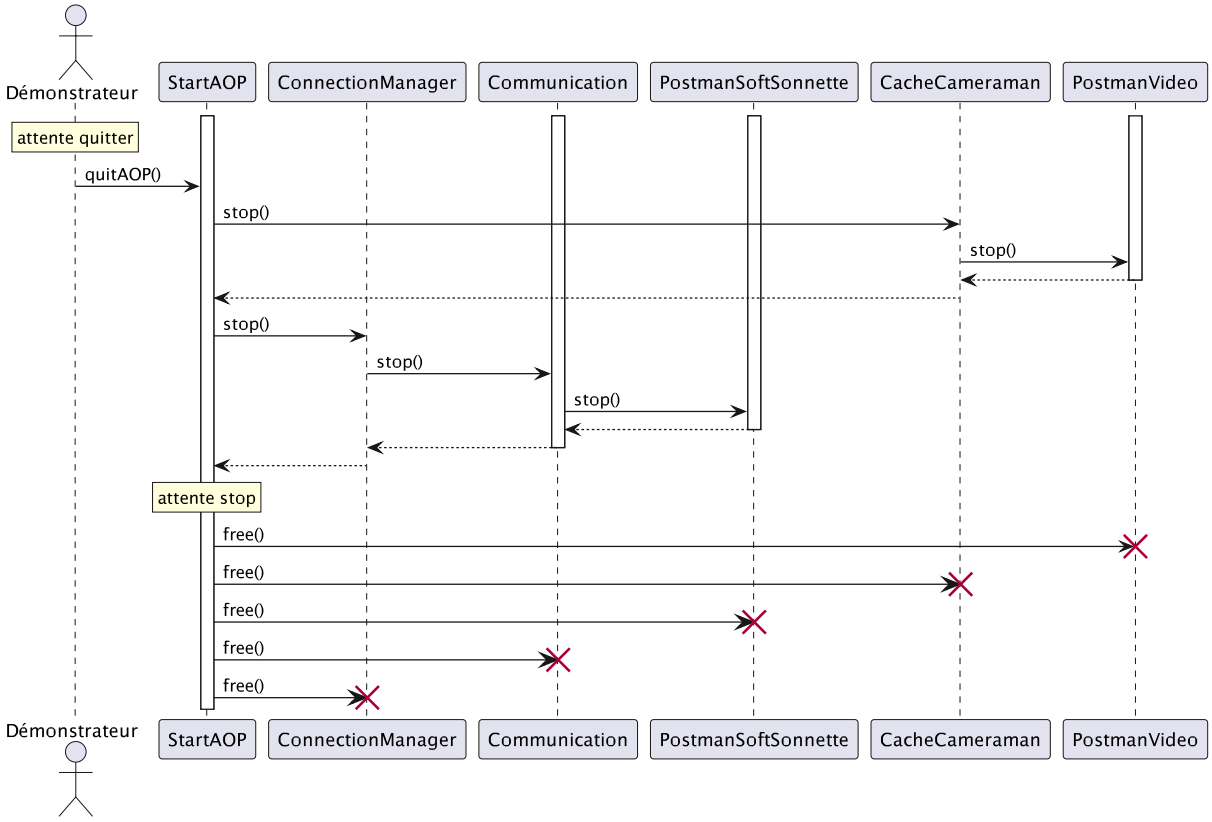

FIGURE 52 – Diagramme de séquence de l'arrêt d'AOP

3.2.2.2 [Package] GUI

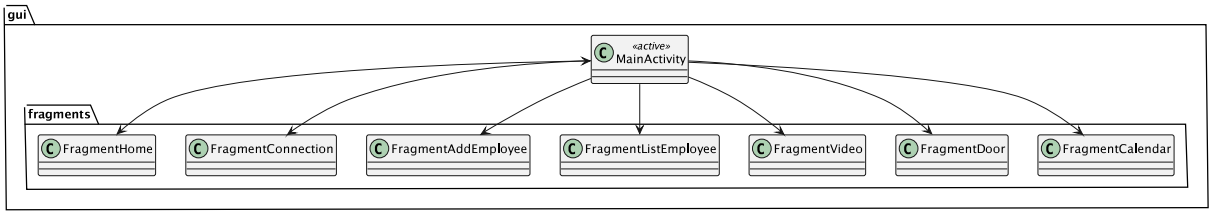

FIGURE 53 – Package de GUI

3.2.2.2.1 Description structurelle de GUI

Le package GUI correspond à l'objet GUI de la conception générale. Il se compose de plusieurs fragments, chaque fragment étant un écran de l'application AOP. GUI possède également une activité qui gère toute l'application et est en charge des fragments : FragmentCalendar, FragmentVideo, FragmentListEmployee, FragmentAddEmployee et FragmentDoor.

3.2.2.3 [Medium] Communication

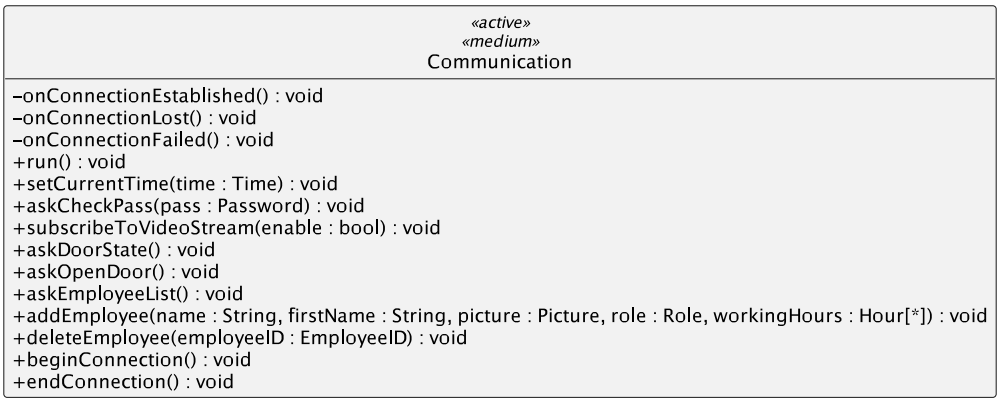

FIGURE 54 – Diagramme de classe de Communication

3.2.2.3.1 Philosophie de Conception

La classe Communication permet de transiter les informations de connexion. Elle permet aussi d'assurer que les applications sont connectées et d'agir en conséquence. Elle reçoit les demandes de l'utilisateur puis communique avec le Dispatcher et le PostmanSoftSonnette. setConnectionStatus() est défini par les méthodes onConnectionEstablished(), onConnectionLost() et onConnectionFailed().

3.2.2.3.2 Description structurelle

3.2.2.3.2.1 Attributs

N.A.

3.2.2.3.2.2 Services offerts

- onConnectionEstablished() : void : Évènement appelé lorsque la connexion entre AOP et SoftSonnette est faite.
- onConnectionLost() : void : Évènement appelé lors de la perte de la connexion entre AOP et SoftSonnette.
- onConnectionFailed() : void : Évènement appelé lors de l'échec de la connexion entre AOP et SoftSonnette.
- errorConnection() : void : Permet de signaler à l'utilisateur une erreur lors de la connexion.
- askCheckPass (pass : String) : void : Envoie le mot de passe à SoftSonnette.
- setCurrentTime(time : Time) : void : Envoie l'heure d'AOP à SoftSonnette.
- sendClock(clock : String) : void : Envoie la clock de AOP à SoftSonnette.
- askCalendar() : void : Demande à recevoir la liste des employés pour le calendrier.
- askEmployeeList() : void : Demande la liste des employés.
- askOpenDoor() : void : Demande l'ouverture de la Porte.
- askDoorState() : void : Demande l'état de la Porte.

- addEmployee(employeeID : byte, name : String, firstName : String, role : byte, workingHours : byte[][]) : void : Demande l'ajout d'un employé.
- deleteEmployee(employeeID : EmployeeID) : void : Demande la suppression d'un employé des données persistantes.
- subscribeToVideoStream (enable : bool) : void : Active ou désactive l'affichage de la vidéo.
- run() : void : Lance un thread lisant les messages envoyés par SoftSonnette.
- beginConnection() : void : Lance la connexion avec SoftSonnette.
- endConnection() : void : Termine la connexion avec SoftSonnette.

3.2.2.4 [Medium] PostmanSoftSonnette

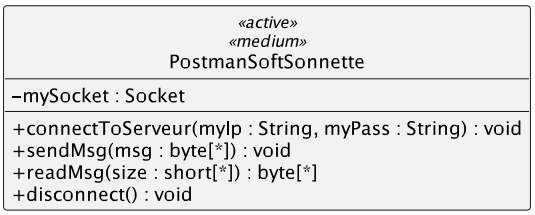

FIGURE 55 – Diagramme de classe de PostmanSoftSonnette

3.2.2.4.1 Philosophie de Conception

La classe PostmanSoftSonnette a pour objectif de gérer l'envoi et la réception de messages à travers les sockets de communication entre SoftSonnette et AOP. C'est aussi l'objet qui met en place le serveur de cette communication. Il est en interaction avec les proxys de GUI et UISS pour l'encodage et le décodage des trames.

3.2.2.4.2 Description structurelle

3.2.2.4.2.1 Attributs

N.A.

3.2.2.4.2.2 Services offerts

- mySocket() : void : Enregistre le socket utilisé pour la communication avec SoftSonnette.
- connectToServer(ip : IP, port : Port) : void : Lance la connexion grâce au socket et à l'adresse IP entrés en paramètre.
- readMsg(short) : byte[ ] : Lis dans le socket la taille du message envoyé, puis lit le reste du message grâce au paramètre de la méthode.
- sendMsg(byte : byte[ ]) : void : Envoie un message à SoftSonnette.
- disconnect() : void : Ferme le socket.

3.2.2.5 [Medium] Dispatcher

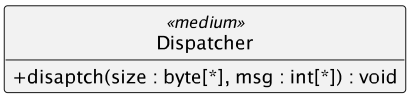

FIGURE 56 – Diagramme de classe de Dispatcher

3.2.2.5.1 Philosophie de Conception

La classe Dispatcher a pour objectif de gérer l'envoi et la réception de messages à travers les sockets de communication. C'est aussi l'objet qui met en place le serveur de cette communication.

3.2.2.5.2 Description structurelle

3.2.2.5.2.1 Attributs

N.A.

3.2.2.5.2.2 Services offerts

- dispatch(size : byte[], msg : int[]) : void : Dispatche les évènements reçus par le serveur. Prend en premier paramètre la taille du message et en deuxième paramètre le message.

3.2.2.6 [Object] Protocol

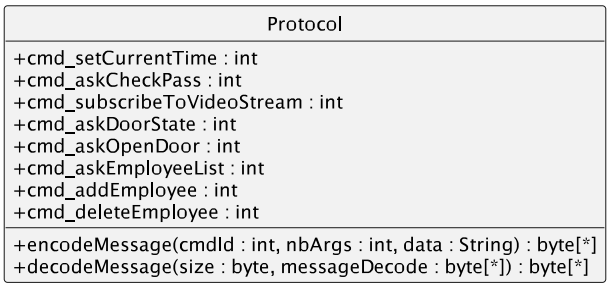

FIGURE 57 – Diagramme de classe de Protocol

3.2.2.6.1 Philosophie de Conception

La classe Protocol est utilisée pour encoder les messages à envoyer et décoder les différents messages reçus.

3.2.2.6.2 Description structurelle

3.2.2.6.2.1 Attributs

- cmd_setCurrentTime : int : ID de la commande pour l'envoi de l'heure.
- cmd_askCheckPass : int : ID de la commande pour la demande de vérification du mot de passe.
- cmd_subscribeToVideoStream : int : ID de la commande pour la demande de stream.
- cmd_askDoorState : int : ID de la commande pour la demande de l'état de la Porte.
- cmd_askOpenDoor : int : ID de la commande pour l'ouverture de la Porte.
- cmd_askEmployeeList : int : ID de la commande pour recevoir la liste des employés.
- cmd_addEmployee : int : ID de la commande pour ajouter un employé.
- cmd_deleteEmployee : int : ID de la commande pour supprimer un employé.

3.2.2.6.2.2 Services offerts

- encodeMessage(cmdId : int, nbargs : int, data : String) : byte : Encode le message pour l'envoyer à SoftSonnette.
- decodeMessage(size : byte, messageDecode : byte[]) : byte : Décode la trame reçue de SoftSonnette.

3.2.2.7 [Medium] PostmanVideo

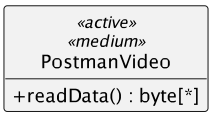

FIGURE 58 – Diagramme de classe de PostmanVideo

3.2.2.7.1 Philosophie de Conception

La classe PostmanVideo est utilisée pour réceptionner du flux UDP provenant de SoftSonnette.

3.2.2.7.2 Description structurelle

3.2.2.7.2.1 Attributs

N.A.

3.2.2.7.2.2 Services offerts

- readData() : byte[] : Lit le paquet de données reçu et retourne sa taille.

3.2.2.8 [Cache] CacheCameraman

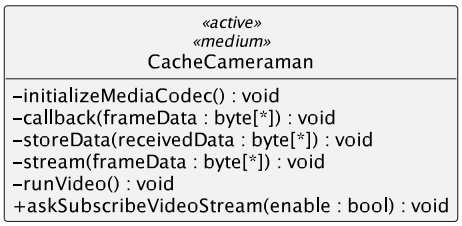

FIGURE 59 – Diagramme de classe de CacheCameraman

3.2.2.8.1 Philosophie de Conception

La classe CacheCameraman est utilisée pour le traitement des données du flux vidéo, et donc de la mise à jour de la vidéo. Le temps de synchronisation est difficile à évalué, car il dépend de la vitesse de streaming du Streamer, ainsi que de la vitesse de calcul du téléphone. La vidéo étant envoyé à 15 images par secondes, le temps de synchronisation minimum sera donc de 15 fois par secondes.

CacheCameraman utilise un mediaCodec, c'est un objet permettant de décoder le flux vidéo en lui fournissant des trames.

3.2.2.8.2 Description structurelle

3.2.2.8.2.1 Attributs

- dataFrame : byte[] : Stocke temporairement les données reçues.

3.2.2.8.2.2 Services offerts

- initializeMediaCodec() : void : Initialise le mediaCodec (avec les paramètres de la vidéo entrés en dur).
- callback(frameData : byte[]) : void : Envoie au mediaCodec les trames détectées pour qu'il les recompose.
- storeData(receivedData : byte[]) : void : Stocke les données temporairement avant de les donner au mediaCodec.
- stream(frameData : byte[]) : void : Traite la frame qui vient d'être détectée avant de l'envoyer au mediaCodec.
- runVideo() : void : Analyse les données reçues et les décompose en trames.
- askSubscribeVideoStream(enable : bool) : void : Demande le début du streaming de la vidéo.

3.2.2.9 [Object] Door

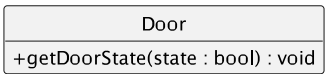

FIGURE 60 – Diagramme de classe de Door

3.2.2.9.1 Philosophie de Conception

La classe Door permet d'avoir l'état de la Porte et de pouvoir ouvrir la Porte.

3.2.2.9.2 Description structurelle

3.2.2.9.2.1 Attributs

N.A.

3.2.2.9.2.2 Services offerts

- getDoorState(state : bool) : void : Permet d'avoir l'état de la Porte.

3.2.2.10 [Object] WeeklyCalendar

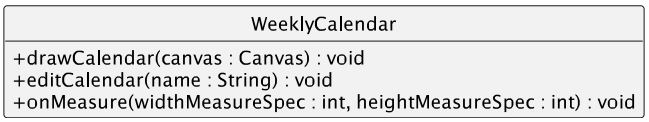

FIGURE 61 – Diagramme de classe de WeeklyCalendar

3.2.2.10.1 Philosophie de Conception

La classe WeeklyCalendar est utilisée pour créer et dessiner le calendrier.

3.2.2.10.2 Description structurelle

3.2.2.10.2.1 Attributs

N.A.

3.2.2.10.2.2 Services offerts

- drawCalendar(canvas : Canvas) : void : Méthode de dessin du calendrier.
- editCalendar(name : String) : void : Coloriage des créneaux horaires à dessiner selon l'employé.
- onMeasure(widthMeasureSpec : int, heightMeasureSpec : int) : void : Méthode de mesure permettant le dessin du calendrier.

3.2.2.11 [Cache] CacheEmployeeManager

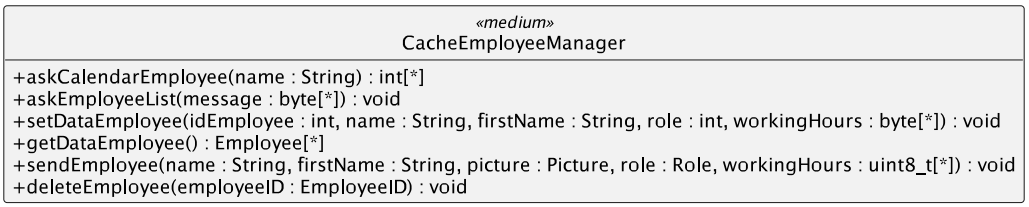

FIGURE 62 – Diagramme de classe de CacheEmployeeManager

3.2.2.11.1 Philosophie de Conception

La classe CacheEmployeeManager est utilisée pour stocker temporairement les données des employés. Il est appelé lorsque l'utilisateur souhaite consulter le calendrier ou la liste des employés. Chaque ajout d'employé est ajouté dans le cache puis envoyé au serveur. Ce cache est synchronisé à chaque appel des écrans EMPLOYEE_LIST_ID et CALENDAR_ID.

3.2.2.11.2 Description structurelle

3.2.2.11.2.1 Attributs

- listEmployees : Employee[] : Liste courante des employés.

3.2.2.11.2.2 Services offerts

- askCalendarEmployee(name : String) : int[] : Méthode permettant de préparer l'affichage du calendrier selon l'employé. Elle permet d'obtenir les heures de début et de fin de chaque jour.
- askEmployeeList(message : byte[]) : void : Méthode permettant d'obtenir la liste des employés.
- setDataEmployee(idEmployee : int, name : String, firstName : String, role : int, workingHours : byte[]) : void : Méthode permettant de modifier la liste des employés dans le cache.
- getDataEmployee() : Employee[] : Fonction qui retourne la liste cache des employés.
- sendEmployee(name : String, firstName : String, picture : Picture, role : Role, workingHours : int[]) : void : Méthode d'envoi des données de l'employé vers les données persistantes du projet.
- sendDeleteEmployee(employeeID : EmployeeID) : void : Méthode d'envoi de suppression d'un employé.

3.2.3 Description des classes de SoftPorte

3.2.3.1 [Object] Starter

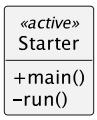

FIGURE 63 – Diagramme de classe de Starter

3.2.3.1.1 Philosophie de Conception

La classe Starter correspond au thread principal de l'application SoftPorte. C'est elle qui possède la méthode main(), méthode appelée au lancement de l'application.

3.2.3.1.2 Description structurelle

3.2.3.1.2.1 Attributs

N.A.

3.2.3.1.2.2 Services offerts

- main : Méthode d'entrée du programme
- run : Tâche de fond qui attend l'évènement indiquant la fin de l'execution

3.2.3.1.3 Diagramme de séquence du démarrage et de l'arrêt de SoftPorte

Le diagramme suivant représente la séquence d'initialisation d'SoftPorte :

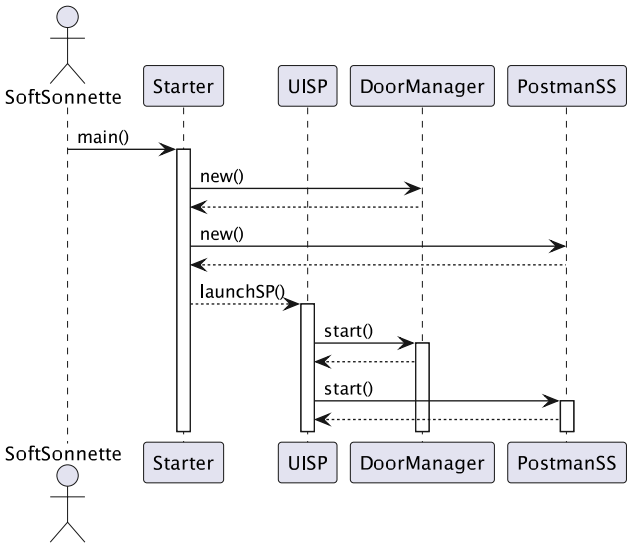

FIGURE 64 – Diagramme de séquence de l'initialisation de SoftPorte

Le diagramme suivant représente la séquence d'arrêt d'AOP.

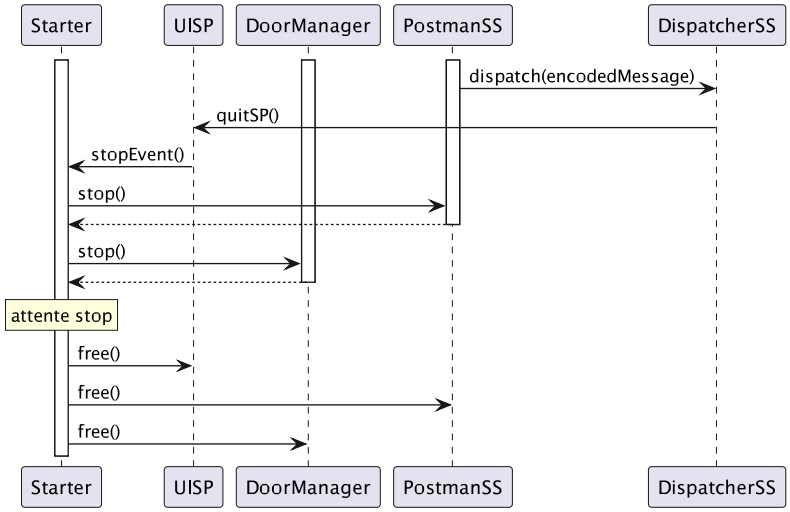

FIGURE 65 – Diagramme de séquence de l'arrêt de SoftPorte

3.2.3.2 [Medium] ProxyGUI

FIGURE 66 – Diagramme de classe de ProxyGUI

3.2.3.2.1 Philosophie de Conception

La classe ProxyGUI est utilisé pour simuler le comportement de la classe GUI. Elle encode les trames à destination de GUI avant de les envoyer au postman.

3.2.3.2.2 Description structurelle

3.2.3.2.2.1 Attributs

N.A.

3.2.3.2.2.2 Services offerts

- updateDoorState(doorState : bool) : Informe GUI du nouvel état de la porte.

3.2.3.3 [Medium] ProxyUISS

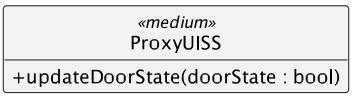

FIGURE 67 – Diagramme de classe de ProxyUISS

3.2.3.3.1 Philosophie de Conception

La classe ProxyUISS est utilisée pour simuler le comportement de la classe UISS. Elle encode les trames à destination de UISS avant de les envoyer au postman.

3.2.3.3.2 Description structurelle

3.2.3.3.2.1 Attributs

N.A.

3.2.3.3.2.2 Services offerts

- updateDoorState(doorState) : Informe UISS du nouvel état de la porte.

3.2.3.4 [Medium] PostmanSS

FIGURE 68 – Diagramme de classe de PostmanSS

3.2.3.4.1 Philosophie de Conception

La classe PostmanSS est en charge de la communication entre SoftPorte et SoftSonnette. C'est elle qui gère l'envoie et la réception de messages sur le Virtual UART.

3.2.3.4.2 Description structurelle

3.2.3.4.2.1 Attributs

N.A.

3.2.3.4.2.2 Services offerts

- read() : Tâche de fond qui lis les messages reçu et les envoie au dispatcher.
- run() : Tâche de fond qui envoie les messages en attentes à SoftSonnette.
- send() : Fonction utilisée pour mettre en attente l'envoi d'un message vers SoftSonnette.

3.2.3.5 [Medium] DispatcherSS

FIGURE 69 – Diagramme de classe de DispatcherSS

3.2.3.5.1 Philosophie de Conception

La classe DispatcherSS a pour objectif de recevoir et de décoder les différents messages reçus afin de les distribuer à DoorManager ou UISP.

3.2.3.5.2 Description structurelle

3.2.3.5.2.1 Attributs

N.A.

3.2.3.5.2.2 Services offerts

- dispatch(encodedMessage byte[*]) : Fonction utilisée pour décoder les messages reçus et les dispatcher aux objets destinataire. Cette fonction prend en argument le message brut encodé sur 2 octets

3.3 Protocole de communication

Les communications entre AOP et SoftSonnette se font via socket TCP/IP et celles entre SoftSonnette et SoftPorte via VirtualUART.

3.3.1 Protocole de communication entre SoftSonnette et AOP

3.3.1.1 Formalisation du protocole

Le protocole de communication entre SoftSonnette et AOP est défini comme une suite d'octets selon la forme suivante :

TAILLE	CMD	NB_ARGS	DONNEES

Ce protocole est dit symétrique. Cela signifie que l'encodage de la trame est là même d'un bout à l'autre de la communication. Une trame se compose ainsi :

- TAILLE : Nombre entier codé sur deux octets donnant la taille totale de la trame en octet. Cette taille comprend ces 2 octets de TAILLE.
- CMD : Valeur pouvant aller de 0x00 à 0x0B et indiquant la commande que l'on veut exécuter.
- NB_ARGS : Indique le nombre d'arguments situés dans la partie données, codé sur un octet.
- DONNEES : La partie données a une taille variable et est facultative, elle est construite de cette façon :

TAILLE_ARG	ARG

Précisons que la taille variable de la partie "données" dépend du nombre d'arguments que l'on veut transmettre. Voici la description de ses différentes parties :

- TAILLE_ARG : Entier codé sur deux octets donnant la taille totale de l'argument en octet. Contrairement à TAILLE, TAILLE_ARG ne compte pas sa propre taille.
- ARG : Chaîne de caractères contenant l'argument.

Cette dernière structure est donc à répéter selon la valeur de NB_ARGS, et donc de CMD. Par exemple, une trame ne contenant pas d'argument avec un CMD de 5 prend la forme suivante :

TAILLE	**CMD**	**NB_ARGS**
0x00 0x04	0x05	0x00

Tableau récapitulatif des différentes commandes :

ID	CMD	Commentaire
0	ERR	Trame réservée à la mention d'une erreur de réception
1	SUBSCRIBE_VIDEO	GUI demande à recevoir la vidéo de SoftSonnette
2	ASK_CHECK_PASS	ConnectionManager demande à Guard si le mot de passe entré sur l'application est correct
3	VALIDATE_PASS	Réponse de Guard si le mot de passe est valide
4	SET_TIME	ConnectionManager modifie l'heure courante sur Clock
5	ASK_OPEN_DOOR	Demande d'ouverture de la Porte depuis AOP vers DoorManager
6	ASK_DOOR_STATE	Demande si la Porte est ouverte ou fermée
7	UPDATE_DOOR_STATE	DoorManager met à jour l'état de la porte sur l'écran d'AOP
8	ADD_EMPLOYEE	AOP demande à ajouter un employé dans les données persistantes situées sur la carte
9	DELETE_EMPLOYEE	AOP demande à supprimer un employé des données persistantes situées sur la carte
10	ASK_EMPLOYEE_LIST	AOP demande la liste des employés à SoftSonnette
11	SET_EMPLOYEE_LIST	Réponse d'EmployeeManager sur la liste d'employés

Les différents arguments selon la CMD sont les suivants :

CMD	ARG1	ARG2	ARG3	ARG4	ARG5
ERR	X	X	X	X	X
SUBSCRIBE_VIDEO	Bool	X	X	X	X
ASK_CHECK_PASS	Password	X	X	X	X
VALIDATE_PASS	Bool	X	X	X	X
SET_TIME	Time	X	X	X	X
ASK_OPEN_DOOR	X	X	X	X	X
ASK_DOOR_STATE	X	X	X	X	X
UPDATE_DOOR_STATE	Bool	X	X	X	X
ADD_EMPLOYEE	Name	Surname	Picture	Role	Hours
DELETE_EMPLOYEE	ID	X	X	X	X
ASK_EMPLOYEE_LIST	X	X	X	X	X
SET_EMPLOYEE_LIST	EmployeeID	Name	Surname	Role	Hours

Les tailles des différents types sont précisées ci-dessous :

- Bool : Booléen codé sur un octet, prend les valeurs 0x00 ou 0x01.
- Password : Chaîne de caractères codé sur 4 octets pour les quatre chiffres qu'il contient.
- Time : Chaîne de caractères codé sur 7 octets : deux pour l'année, puis un pour le mois, le jour, l'heure, la minute et la seconde.
- EmployeeID : Entier codé sur 1 octet.
- Name : Chaîne de caractères de taille variable, codé entre 1 et 12 octets selon TAILLE_ARG.
- Surname : Chaîne de caractères de taille variable, codé entre 2 et 12 octets selon TAILLE_ARG.
- Picture : Chaîne d'octets de taille variable selon TAILLE_ARG.
- Rôle : Entier codé sur 1 octet.
- Hours : Chaîne de caractères, codé sur 14 octets soit 7x2 créneaux horaires.

3.3.1.2 Exemples

3.3.1.2.1 Demande de connexion entre AOP et SoftSonnette

Le diagramme de séquence ci-dessous décrit le processus de connexion entre AOP et SoftSonnette. A noter que la phase de synchronisation du temps qui survient après validatePass() n'est pas présentée par soucis de lisibilité.

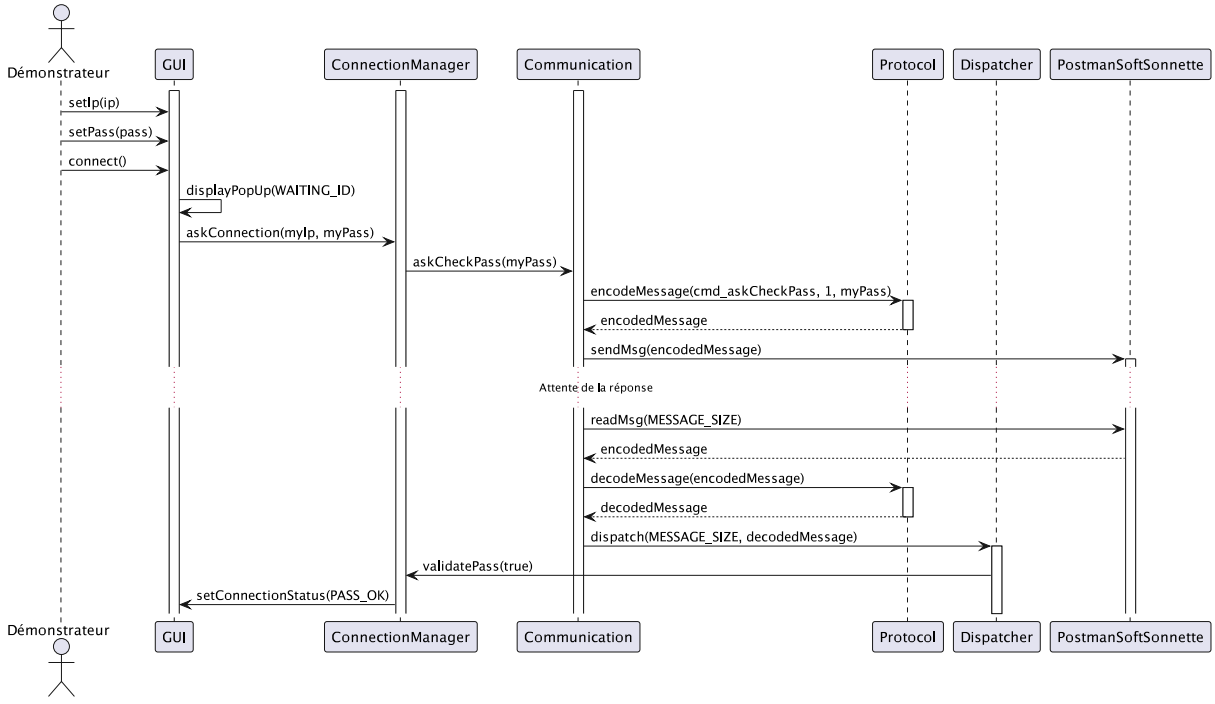

FIGURE 70 – Processus de connexion entre AOP et SoftSonnette du point de vue de AOP

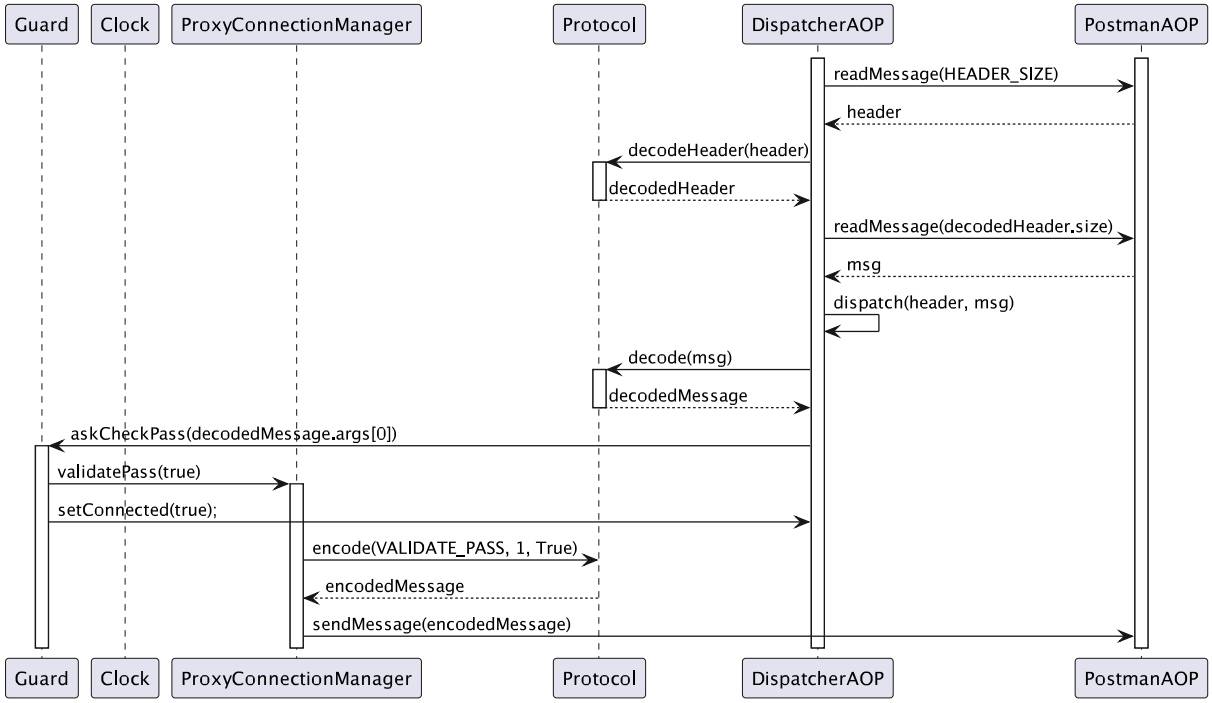

FIGURE 71 – Processus de connexion entre AOP et SoftSonnette du point de vue de SoftSonnette

Dans ce processus de connexion, une requête ASK_CHECK_PASS correspondant à l'appel de la méthode askCheckPass() est envoyée. La requête est encodée de cette manière :

TAILLE	**CMD**	**NB_ARGS**	**TAILLE_ARG**	**ARG**
0x00 0x0A	0x02	0x01	0x00 0x04	0x31 0x32 0x33 0x34

Dans cet exemple :

- TAILLE vaut 0x00 0x0A ce qui signifie que la trame a une longueur de 10 octets.
- CMD vaut 0x02 ce qui équivaut à la requête ASK_CHECK_PASS.
- NB_ARGS vaut 0x01 car il n'y a qu'un seul argument à suivre.
- TAILLE_ARG vaut 0x00 0x04 indiquant que ARG a une taille de 4 octets.
- ARG vaut 0x31 0x32 0x33 0x34 correspondant au code en ASCII de la chaîne "1234".

3.3.2 Protocole de communication entre SoftSonnette et SoftPorte

3.3.2.1 Formalisation du protocole

Le protocole de communication entre SoftSonnette et SoftPorte est défini comme une suite d'octets selon la forme suivante :

CMD	TARGET

- CMD : Valeur pouvant aller de 0x00 à 0x09 et indiquant la commande que l'on veut exécuter.
- TARGET : Indique la cible du message entre AOP et SoftSonnette, codé sur un octet. Elle peut prendre les valeurs SOFTS_ID = 0x00, AOP_ID = 0x01 et SOFTP_ID = 0x02.

Tableau récapitulatif des différentes commandes :

ID	**CMD**	**Commentaire**
0	ERR	Trame réservée à la mention d'une erreur de réception
1	ASK_OPEN_DOOR	Demande d'ouverture de la Porte depuis SoftSonnette ou AOP vers DoorManager
2	UPDATE_DOOR_STATE_TRUE	DoorManager met à jour l'état de la Porte (ouverte) sur l'écran d'AOP ou sur l'écran d'UISS
3	UPDATE_DOOR_STATE_FALSE	DoorManager met à jour l'état de la Porte (fermée) sur l'écran d'AOP ou sur l'écran d'UISS
4	ASK_DOOR_STATE	Demande si la Porte est ouverte ou fermée
5	SIGNAL_NOT_ALLOWED	Bouncer signale à UISP l'interdiction d'ouvrir la Porte
6	LAUNCH_SP	Lance l'application SoftPorte
7	QUIT_SP	Quitte l'application SoftPorte
8	ASK_START_COM	Envoie un signal à UISS
9	ACK_START_COM	Réponse de UISS à UISP

Les différents arguments selon la CMD sont les suivants :

CMD	**TARGET**
ERR	SOFTS_ID / AOP_ID / SOFTP_ID
ASK_OPEN_DOOR	SOFTP_ID
UPDATE_DOOR_STATE_TRUE	SOFTS_ID / AOP_ID
UPDATE_DOOR_STATE_FALSE	SOFTS_ID / AOP_ID
ASK_DOOR_STATE	SOFTP_ID
SIGNAL_NOT_ALLOWED	SOFTP_ID
LAUNCH_SP	SOFTP_ID
QUIT_SP	SOFTP_ID
ASK_START_COM	SOFTS_ID
ACK_START_COM	SOFTP_ID

3.3.2.2 Exemples

Le diagramme de séquence ci-dessous décrit le processus d'ouverture de la Porte.

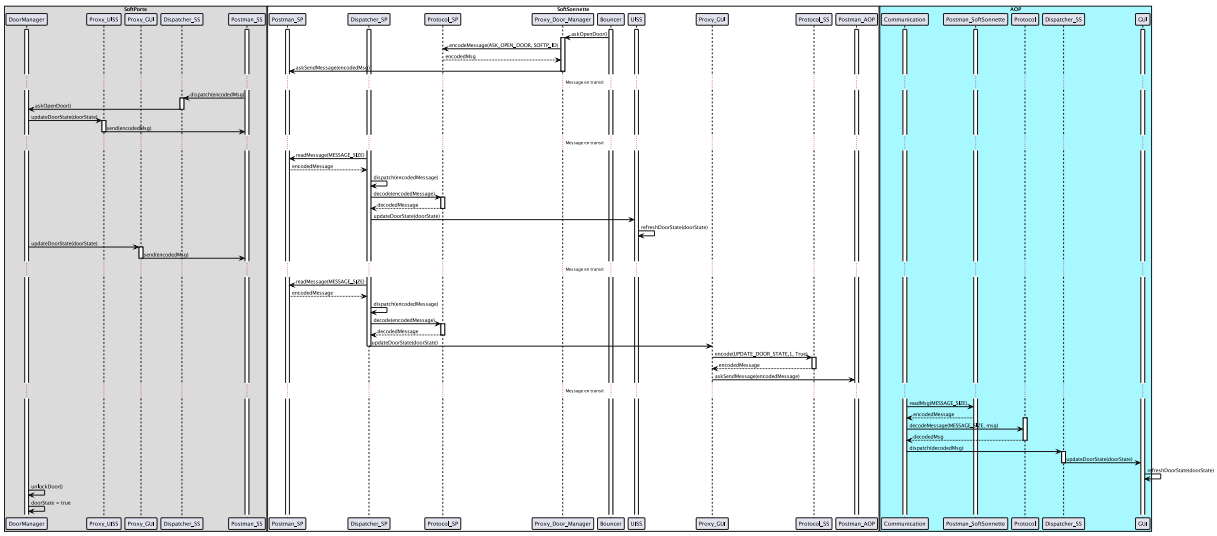

FIGURE 72 – Processus d'ouverture de la Porte

Dans cet exemple, une requête UPDATE_DOOR_STATE_TRUE correspondant à l'appel de la méthode updateDoorState(true) sur l'objet GUI est envoyée. Cette requête est encodée de telle manière :

CMD	TARGET
0x02	0x01

Dans cet exemple :

- CMD vaut 0x02 ce qui équivaut à la requête UPDATE_DOOR_STATE_TRUE.
- TARGET vaut 0x01 ce qui indique que la cible est un objet d'AOP, ici GUI.

3.4 Gestion du multitâche

3.4.1 Identification des accès concurrents dans SoftSonnette

Dans un soucis de lisibilité, le nom des classes est abrégé dans le tableau 3 répertoriant les accès concourants. Les acronymes utilisés sont répertoriés dans le tableau ci-dessous :

Tâche	**Abrégé**	**Autre**	**Abrégé**
RecognitionAI	RAI	Clock	Cl
Bouncer	Bo	EmployeeManager	EM
Cameraman	Ca	ProxyGUI	PG
UISS	UISS	ProxyConnectionManager	PCM
Starter	Sta	ProtocolSP	ProSP
PostmanAOP	PAOP	Guard	G
PostmanSP	PSP	ProxyDoorManager	PDM
DispatcherAOP	DAOP	DataEmployee	DE
DispatcherSP	DSP	ProtocolSS	PSS
Streamer	Str	ProxyUISP	PUISP

Table 2 – Tableau des classes abrégées dans SoftSonnette

	Cl	EM	PG	PCM	ProSP	G	PDM	DE	PSS	PUISP
RAI										
Bo	X	X	X				X	X		X
Ca										
UISS										X
Sta										
PAOP										
PSP										
DAOP	X	X	X	X		X	X	X	X	
DSP			X		X					
Str										

Table 3 – Tableau des accès concurrents dans SoftSonnette

Les classes DispatcherAOP et Bouncer appellent concurremment Clock et EmployeeManager. Pour éviter tout soucis ces deux classes possèdent l'attribut "protected" indiquant aux développeur de mettre en place des stratégies de verrouillage des ressources critiques permettant ainsi d'éviter les accès concourants.

3.4.2 Identification des accès concurrents dans SoftPorte

	ProxyGUI	ProxyUISS	UISP	DispatcherSS
DoorManager	X	X		
PostmanSS			X	X

Table 4 – Tableau des accès concurrents dans SoftPorte

Il n'y a pas de problème de concurrence.

3.4.3 Identification des accès concurrents dans AOP

Dans un soucis de lisibilité, le nom des classes est abrégé dans le tableau 6 répertoriant les accès concourants. Les acronymes utilisés sont répertoriés dans le tableau ci-dessous :

Tâche	**abréger**	**Autre**	**abréger**
Communication	Com	Protocol	Pr
PostmanVideo	PV	ConnectionManager	CM
PostmanSoftSonnette	PS	Dispatcher	Di
GUI	GUI	CacheEmployeeManager	CEM
CacheCameraman	CCa	WeeklyCalendar	WC
		ProxyDoorManager	PDM
		ProxyClock	PC
		ProxyGuard	PG

Table 5 – Tableau des classe abrégées dans AOP

	Pr	CM	Di	CEM	WC	PDM	PC	PG
GUI		X	X	X	X	X	X	X
Com	X	X	X	X	X	X	X	X
PV								
PS								
CCa								

Table 6 – Tableau des accès concurrents dans AOP

La tâche MainActivity, appartenant à GUI, est le cœur d'AOP. Or, MainActivity n'est pas susceptible de manipuler des ressources critiques puisqu'elle délègue ses tâches aux objets concernés, il n'y a donc pas de problèmes de concurrence ici.

La tâche Communication se trouve en haut de la hiérarchie, régissant les demandes, les appels à ses méthodes de lecture et d'écriture, et gère le socket de communication, comme son nom l'indique elle communique avec tous les objets.

Les tâches PostmanVideo, PostmanSoftSonnette et CacheCameraman n'accèdent à aucun objet "non actif", aucun problème de concurrence n'est à détecter ici.

4 Dictionnaire du domaine

A

AOP (Application Ouverture Porte)

AOP est l'application Android utilisée par le Démonstrateur comme interface utilisateur pour gérer les différentes fonctionnalités du PSC.

B

Board

Est désigné par le terme Board la STM32MP15 formant l'ensemble Microcontrôleur + Microprocesseur.

byte

Type de données sur 1 octet.

D

Données persistantes

Les données persistantes sont les données sauvegardées par le PSC d'une session à une autre. Ces données sont sauvegardées dans l'entité *Données_ Testeurs*. L'opération de chargement des données persistantes diffère en fonction du type de donnée :

- Les données textuelles sont chargées en mémoire vive lors du lancement de SoftSonnette.
- Les *photos* demeurent stockées en mémoire non volatile.

Le processus de sauvegarde des données persistantes diffère en fonction du type de donnée :

- La copie en mémoire vive des données textuelles en mémoire non volatile est assurée en fin d'exécution de SoftSonnette.
- L'ajout ou la suppression de *photos* se fait au fur et à mesure de l'exécution du PSC.

Données_Testeurs

Correspond à l'entité qui stock de manière persistantes les *informations Testeurs* sur la Board. En fonction du type de donnée, le mode de stockage varie :

- Les données textuelles sont stockées dans un fichier csv, cela comprend le *nom*, le *prénom*, le *rôle* et les *horaires*.
- Les *photos* sont stockées dans un dossier "picture".

Déverrouiller la Porte

L'action de déverrouillage de la Porte est réalisée à titre indicatif. L'ouverture de la Porte est simulée par la mise à jour du voyant indicant l'état de cette dernière sur les écrans d'AOP et de SoftSonnette.

E

employeID

Correspond à l'identifiant de l'employé dans *Données_ Testeurs*. Il s'agit d'un entier compris entre 1 et MAX_EMPLOYE.

Employé

Un employé est équivalent à un Testeur dans le cadre du prototype PSC. Il s'agit d'une personne susceptible de se présenter devant le PSC pour entrer, et dont les informations peuvent être entrées dans l'application AOP.

F

Flux vidéo

Le flux vidéo est généré par la webcam (E_Caméra), qui peut filmer avec une résolution de 1280x720p. Une résolution plus faible est préférée pour l'envoi du flux, il est défini que cette résolution est de 320x240p. Stopper l'affichage du flux vidéo correspond à arrêter son affichage sur l'Écran_SoftSonnette et son envoi à AOP (si ce dernier le requiert pour l'Écran_Video). Reprendre l'affichage du flux vidéo permet d'afficher de nouveau cette dernière sur Écran_SoftSonnette mais ne redémarre pas son envoi à AOP, ce dernier doit être redemandé par le Démonstrateur.

FR

La Fréquence de Rafraîchissement correspond à la fréquence d'affichage de chaque image lors de flux vidéo. Il est défini que FR vaut 15 Hz.

H

Horaires

Les horaires d'accès pour déclencher l'ouverture de la Porte sont sous la forme d'un bloc horaire unique pour chaque journée. Les horaires sont regroupés en créneaux de 30 min et le format d'affichage est le format 24h. Ces horaires se basent sur l'heure de la Board, qui est équipée d'une RTC. Il est convenu que sa synchronisation est effectuée lors de la connexion entre l'AOP et SoftSonnette.

I

Infomations de connexion

Les informations de connexion correspondent à l'adresse *IP* de la Board et au *mot de passe* requis pour entrer dans AOP.

Informations Testeurs

Chaque Testeur est identifié par : un *nom*, un *prénom*, un *rôle*, une *photo* et pour les employés spéciaux des *horaires* d'accès. La convention d'encodage des données textuelles est l'UTF-8.

Initialiser la communication avec SoftSonnette

Correspond au processus durant lequel SoftPorte configure la Board et met en place la communication entre les deux applications.

IP

L'adresse IP permettant la connexion entre AOP et SoftSonnette. Cette adresse est représentée sous la forme IPv4 "x.x.x.x" où x représente un entier allant de 0 à 255.

L

Linux

Fait référence à l'OS OpenSTLinux (OSTL) version 4.1 (Kernel Linux : 5.15) déployé sur le Microprocesseur.

M

MAX_EMPLOYE

Régit le nombre maximal d'employés enregistrés dans *Données_Testeurs* supporté par le PSC. MAX_EMPLOYE vaut 10.

Mot de passe

Le mot de passe permettant au Démonstrateur de se connecter à AOP. Celui-ci est composé de 4 chiffres.

N

Nom

Le nom d'un employé est une chaine de caractère comprenant entre 1 et 12 alphanumériques encodés en UTF-8.

P

Photo

Les photos sont des fichiers au format jpeg ou jpg d'un poids maximal de POIDS_MAX_PHOTO.

POIDS_MAX_PHOTO

Défini le poids maximal autorisé d'une photo en méga-octets. POIDS_MAX_PHOTO vaut 3.5 Mo.

Prénom

Le prénom d'un employé est une chaine de caractère comprenant entre 2 et 12 alphanumériques encodés en UTF-8.

R

Rôle

Le rôle d'un employé définit sa plage horaire d'accès. Il existe cinq rôles différents :

- Employé matin : 3h-13h, du lundi au vendredi.
- Employé journée : 8h-20h, du lundi au vendredi.
- Employé soir : 13h-23h, du lundi au vendredi.
- Employé sécurité : Accès illimité (24/7).
- Employé spécial : Les horaires sont définis manuellement pour chaque jour lors de l'ajout de l'employé dans l'application.

S

short

Type de données sur 2 octets.

SoftPorte

SoftPorte est l'application C déployée sur le Microcontrôleur de la Board en charge du contrôle de la Porte.

SoftSonnette

SoftSonnette est l'application C déployée sur l'OS de la Board tournant sur le Microprocesseur en charge des interactions avec le Testeur.

SàE correctement installé

Pour une installation correcte il faut les éléments suivants :

- AOP installé sur le téléphone.
- Linux installé sur le Microprocesseur.
- SoftSonnette installé sur Linux.
- SoftPorte déployé sur le Microcontrôleur.

T

TAC

Le Temps d'Attente de Connexion correspond à la durée maximale après laquelle une connexion est considérée comme non établie, entre une demande de connexion et la réponse associée. Il est défini que TAC vaut 5 secondes.

TAL

Le Temps d'Allumage LED rouge correspond à la durée pendant laquelle la LED LD6 s'allume pour informer que le Testeur est refusé. Il est défini que TAL vaut 5 secondes.

TASS

Le Temps d'Attente SoftSonnette correspond à la durée pendant laquelle SoftSonnette attend une réponse de SoftPorte lors de l'initialisation de ce dernier. Il est défini que TASS vaut 1 seconde.

TOP

Le Temps d'Ouverture Porte correspond à la durée pendant laquelle la Porte s'ouvre. Il est défini que TOP vaut 10 secondes.

5 Table des figures

GRANDE ÉCOLE D'INGÉNIEURS