GROUPE ESEO

Angers
Paris
Shanghai
Dijon

www.eseo.fr

Référence : CONC_EB1

Version : 3

Révision : 17

Équipe : Équipe B1

Responsable du document : Axel ROLLO

État du document : Fermé

Dossier de conception
ST Microelectronics

Notes : Le présent document est un document à but pédagogique. Il a été réalisé sous la direction de Jérôme Delatour, en collaboration avec des enseignants et des étudiants (Marwan Boughammoura, Sonasi Katoa, Sami Kouatli, Kriss Amzal, Romain Richard, Axel Rollo, Nivenn Lanos, Eva Beaubrun, Martin Bourseau, Jocelyn Girault et Alexis Gaonac'h) de l'option SE du groupe ESEO.
La société FORMATO et la société BT Company n'ont pas d'existence légale, il s'agit de sociétés fictives créées pour des besoins pédagogiques. Le personnage de César Bistrucla, un des consultants de FORMATO, est le pseudonyme de Jérôme Delatour.
Ce document est la propriété de Jérôme Delatour du groupe ESEO. En dehors des activités pédagogiques de l'ESEO, ce document ne peut être diffusé ou recopié sans l'autorisation écrite de son propriétaire.

Liste des évolutions et validations du document.

Date	**Action**	**Auteur**	**Version**	**Révision**
19 Octobre 2017	Création du livrable sous LateX	Axel ROLLO	0.1	1
9 & 10 Octobre 2017	Ajout des différentes parties réalisées	Axel ROLLO	1	5
16 Octobre 2017	Ajout des nouvelles parties complétées	Axel ROLLO	1.1	7
17 Octobre 2017	Ajout des MAE & de la nouvelle architecture candidate et des classes de GUI	Axel ROLLO	1.2	8

Table des matières

 © Equipe B1 - Tous droits réservés ESEO

 © Equipe B1 - Tous droits réservés ESEO

Chapitre 1

INTRODUCTION

1.1 Objet

Ce document de conception a pour objectif de rassembler tous les éléments utiles pour la conception logicielle et matérielle du projet « Ville intelligente – Qualité de l'air » réalisé par l'équipe de développement pour la PAVIC. Il présentera les éléments de conception déterminés suite à l'élaboration du dossier de spécification [SPEC_PFE_VI]. Ce dossier suit les recommandations de la norme IEEE 1016_2009 [IEEE- 1016_2009]. Les schéma et figures présentés dans ce document suivent la norme UML 2.4.1 [UML_2.4_2011].

1.2 Portée

Ce document décrit l'ensemble des éléments de conception se rapportant au Système à l'Etude (SaE).

Ce dossier de conception est destiné :

— à l'équipe de développement C et Android, pour permettre la réalisation des logiciels du projet « Ville intelligente – Qualité de l'air ».

— à l'équipe de développement matériel électronique.

— aux auditeurs étant donné que ce projet fera l'objet d'une évaluation finale.

— Aux testeurs, pour qu'ils puissent établir les tests à réaliser ainsi que récupérer les résultats de ces derniers.

1.3 Définitions, acronymes et abréviations

Les abréviations utilisées dans le présent document sont répertoriées et expliquées dans le tableau présenté ci-dessous. Les termes utiles pour interpréter correctement ce dossier de conception, sont définis dans le dictionnaire du domaine présent dans le dossier de spécification [SPEC_PFE_VI] et au chapitre 4 du présent document.

Acronymes, abréviations	**Définitions**
alt	Abréviation utilisée dans les diagrammes de séquences pour représenter les alternatives possibles.

BAL	Abréviation utilisée pour le terme "Boite aux Lettres"
CU	Cas d'Utilisation.
DS	Diagramme de Séquence.
Fiabilité	La fiabilité est l'aptitude d'un composant ou d'un système à fonctionner pendant un intervalle de temps.
GPIO (General Purpose Input/Output)	Désigne des ports d'entrée/sortie.
IHM (Interface Homma Machine)	Moyen permettant à l'administrateur et aux utilisateur d'interagir avec la valise.
LED (Light-Emitting Diode)	Une diode électroluminescente.
MC	Abréviation utilisée dans les diagrammes de séquence pour préciser que l'acteur appartient à MasterChef.
MM	Abréviation utilisée dans les diagrammes de séquence pour préciser que l'acteur appartient à MasterMind.
MP	Abréviation utilisée dans les diagrammes de séquence pour préciser que l'acteur appartient à MasterPower.
N.A	Non applicable.
OMG (Object Management Group)	Association à but non lucratif dont l'objectif est de standardiser et promouvoir le modèle objet.
PAQL (Plan d'Assurance Qualité Logiciel)	Dossier permettant d'assurer une qualité au projet de faciliter ce dernier.
PDU (Power Distribution Unit)	Ensemble power et carte STM32.
ProSE (Projet Système Embarqué)	Projet de l'option Système Embarqué par groupe de 10/11 étudiants.
ref	Abréviation utilisée dans les diagrammes de séquence pour indiquer une référence à un autre diagramme de séquence.
SaE (Système à l'Etude)	Il s'agit de l'ensemble des composants MaterChef, MasterMind et MasterPower.
SYTE	Nom du projet.
UI (User Interface)	Interface utilisateur.
UML (Unified Modeling Language)	Notation graphique normalisée, définie par l'OMG et utilisée en génie logiciel.
USB (Universal Serial Bus)	Norme décrivant un bus informatique en transmission série qui sert à connecter des périphériques informatiques à un système informatique.

 © Equipe B1 - Tous droits réservés ESEO

Wi-Fi (Wireless Fidelity)	Protocole de communication sans fil régis par les normes du groupe IEEE-1016_2009.

TABLE 1.1 – Tableau des acronymes et abréviations

1.4 Références

Voici un tableau récapitulatif des documents utilisés pour le dossier de conception ainsi que les liens permettant d'accéder aux fichiers.

Référence	**Nom, auteur**
[IEEE-1016_2009]	IEEE, std 1016-2009 « Recommended Practice for Software Requirements Specifications », https ://standards.ieee.org/findstds/standard/1016-2009.html, 2009.
[SPEC_PFE_VI]	Dossier de spécification du projet « Ville intelligente – Qualité de l'air », référentiel documentaire projet PAVIC 2017/2018.
[UML_2.4_2011]	OMG, « Unified Modeling Language », version 2.4.1, http ://www.omg.org/spec/UML/, 2011.

TABLE 1.2 – Tableau des références

1.5 Vue d'ensemble

Ce document de conception est structuré en plusieurs parties :

— une première partie, qui concerne la conception générale du prototype. Cette partie présente l'architecture candidate et donne les grands principes de fonctionnement du projet « Ville Intelligente – Qualité de l'air ».

— une seconde partie présentera la conception détaillée. Cette partie présente les composantes du système en précisant cette fois-ci la gestion des entrées et des sorties, la gestion multitâche ainsi que la gestion de la persistance.

— un dictionnaire du domaine constitue la dernière partie du document.

 © Equipe B1 - Tous droits réservés ESEO

Chapitre 2

Conception générale

2.1 Architecture candidate

Le diagramme d'objets suivant présente l'architecture candidate du système à l'étude. On y retrouve toutes les classes de la conception générale avec leurs méthodes.

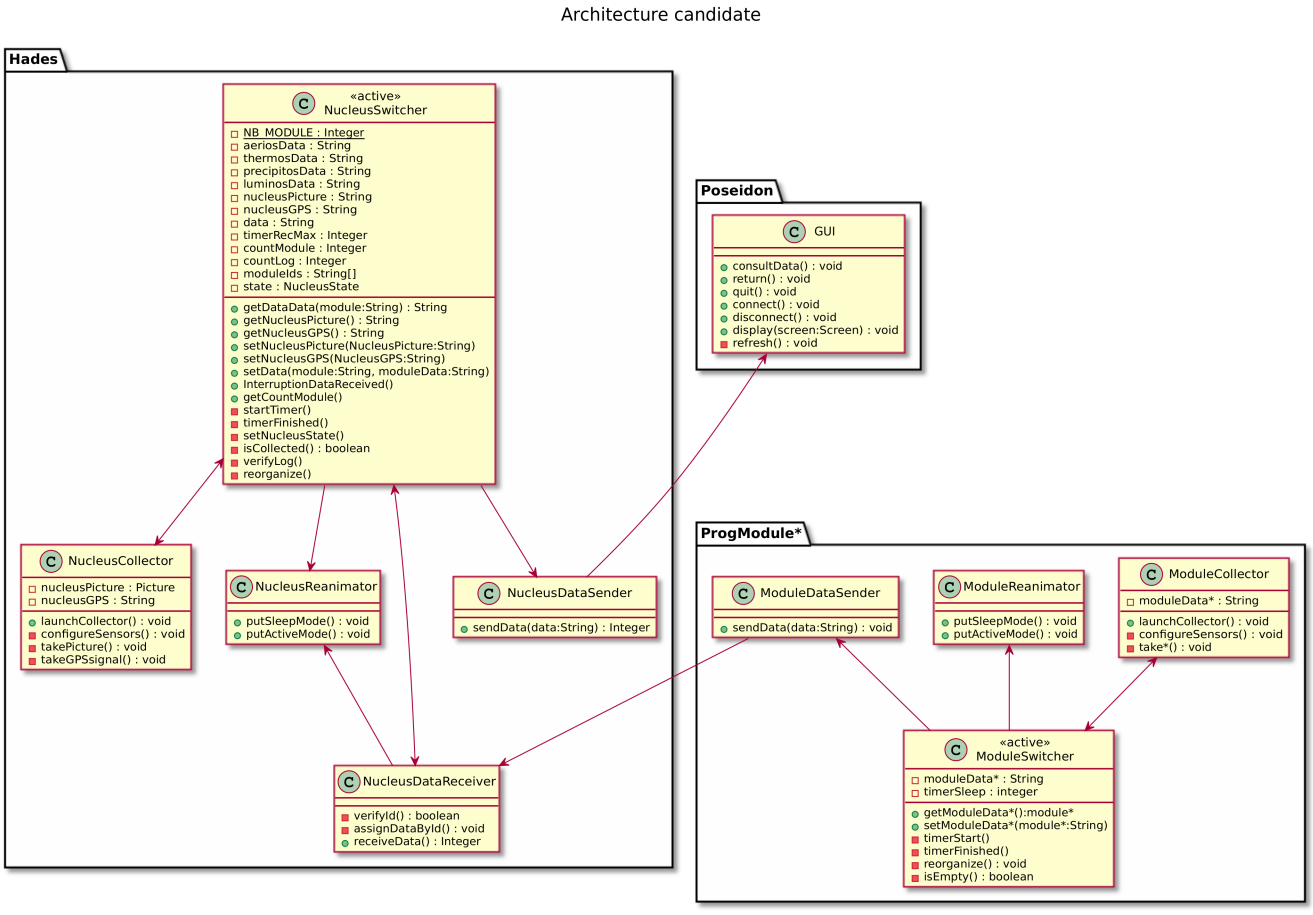

FIGURE 2.1 – Architecture candidate

Le diagramme ci-dessus représente l'architecture logique qui sera adoptée pour la réalisation du prototype de la station. Il s'agit de la conception générale ; l'hypothèse d'un système matériel à ressource infinie est pour l'instant posée.

2.1.1 Poseidon

Dans ce diagramme on retrouve « GUI » qui permet de contrôler les affichages de l'application « Poseidon » ainsi que les interactions utilisateurs. Son principal rôle est de collecter et de mettre à jour les champs de l'écran « Screen_Data ».

2.1.2 Nucleus

Le composant « NucleusDataRecever » a la responsabilité de recevoir des données sous forme de trame. Il extrait l'information de la trame de réception et envoi les données triées vers le « NucleusSwitcher ». Il peut recevoir les quatre différentes trames venant des quatre modules.

Le composant « NucleusSwitcher » a un rôle d'aiguilleur, il redirige les différentes données vers les différentes plages mémoires, afin d'organiser le contenu des données reçues. De plus il met en forme une nouvelle trame comprenant les données des quatre modules. Il l'envoi ensuite vers « NucleusDataSender ».

Le composant « NucleusCollector » a le rôle de collecter les informations des capteurs interne à Nucléus, c'est-à-dire l'image du capteur photographique et la donnée GPS. Il envoi ensuite directement les informations à « NucleusSwitcher » pour stocker les données.

Le composant « NucleusReanimator » permet de contrôler les différents modes de fonctionnement de la puce ESP8266-01 (« Sleep mode » ou « Active mode »).

Le composant « NucleusDataSender » a le rôle d'envoyer la trame mise en forme précédemment vers un Broker avec l'aide du protocole MQTT.

2.1.3 Modules

Pour le prototype réalisé, nous utilisons quatre modules avec des fonctionnalités manifestement similaires. Pour cette raison, nous avons créé un modèle pouvant être applicable à n'importe quel de nos quatre modules. Par la suite, l'ajout éventuel d'un nouveau module ne posera pas de soucis. Il faut tout de même qu'il respecte les normes de conception et de spécification du projet « Ville Intelligente – Qualité de l'air ».

Le composant « ModuleReanimator » permet de contrôler les différents modes de fonctionnement de la puce du NRF24LE1 (« Sleep mode » ou « Active mode »).

Le composant « ModuleDataSender » permet d'envoyer la trame représentant la concaténation des différents relevés des mesures des capteurs.

Le composant « ModuleSwitcher » permet d'aiguiller les différentes informations aux différentes plages mémoires disponibles sur la puce. Il a aussi un rôle de timer. Tous les temps « T_DR » il réveille la puce, lance une nouvelle acquisition de donnée puis la rendors.

Le composant « ModuleCollector » a pour rôle de récupérer tous les temps « T_DR » les différentes données des capteurs et de les envoyer au « ModuleSwitcher ».

 © Equipe B1 - Tous droits réservés ESEO

2.2 Grands principes de fonctionnement

Cette partie présentera le fonctionnement du prototype. Les diagrammes de séquence présentés sont : le diagramme de séquence principal et rafraîchir les données.

2.2.1 Principal

Ce diagramme de séquence correspond au scénario nominal du CU « Informer périodiquement les utilisateurs afin d'anticiper les risques de réactions allergiques. » du dossier de spécification. L'utilisateur effectue une séquence d'enchaînement d'actions pour observer le principe de fonctionnement du système.

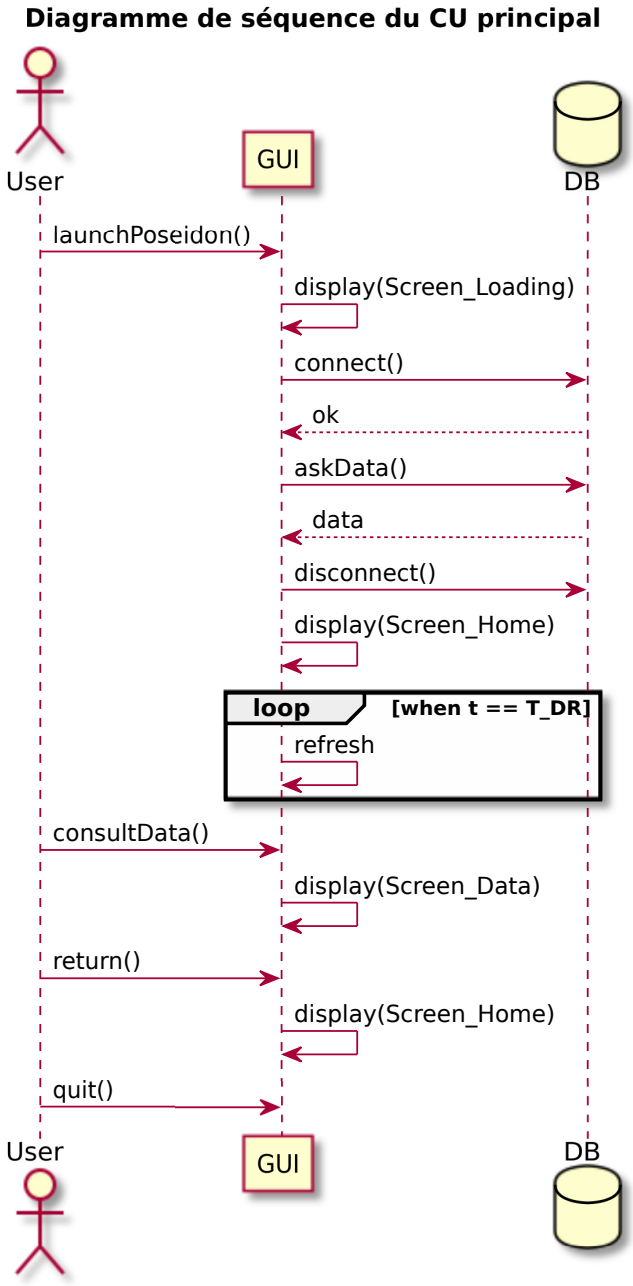

FIGURE 2.2 – Diagramme de séquence : Principal

 © Equipe B1 - Tous droits réservés ESEO

2.2.2 Rafraîchir données

Ce diagramme représente le scénario du CU « Rafraîchir les données » du dossier de spécification. Poseidon demande à se connecter à la base de donnée pour récupérer les données les plus à jour.

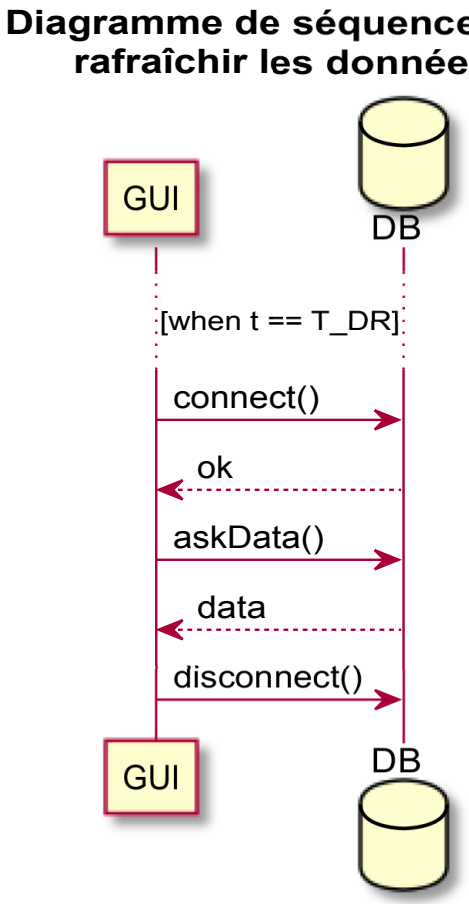

FIGURE 2.3 – Diagramme de séquence : Rafraîchir données

 © Equipe B1 - Tous droits réservés ESEO

2.3 Description des composants

2.3.1 Description des types manipulés entre composants

— aeriosData, thermosData, precipitosData, luminosData, nucleusPicture, nucleusGPS, moduleData* : Chaîne de caractère pour y stocker les données des différents capteurs à un instant t.

— data : Chaîne de caractère qui correspond à l'assemblage des différentes données de capteurs.

— timerRecMax : Entier non signé qui correspond à un temps d'échantillonnage maximal.

— timerSleep : Entier non signé qui correspond au temps pour lequel le module doit se mettre à l'état "Sleep Mode".

2.3.2 Description des unités manipulés pour chaque type

— ppm : La "partie par million", est la faction valant un millionième. On l'utilise pour exprimer une faction massique (cf. mg/m^3, 1 mg/m^3 = 0.001 ppm).

— mg/m^3 : Le "milligramme par mètre cube" est une unité de mesure généralement utilisé pour mesurer la concentration de polluant dans l'air.

— °C : Le "degré celcius" est unité pour exprimer une différence de température.

— % : Le pourcentage est un nombre représenté comme un nombre relatif de l'ordre de 10^{-2}. Il est utilisé pour représenter une ordre de grande allant de 0 à 100.

— Pa : Le pascal est une unité de mesure de pression.

— Lx : Le lux est une unité de mesure de l'éclairement lumineux.

— m/s : Le "mètre par seconde", mesure une vitesse de propagation.

— m : Le mètre mesure une distance.

 © Equipe B1 - Tous droits réservés ESEO

Chapitre 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/sorties et les IHMs.

— élaborer les protocoles de communication.

— définir le parallélisme et l'initialisation.

3.1 Architecture physique

Cette partie présente l'architecture physique et logiciel détaillé du projet "Ville intelligente - Qualité de l'air". Elle reprend la conception générale auquel nous ajoutons certaines des éléments pour que l'architecture soit portée sur des contraintes réelles.

L'architecture physique est séparée en plusieurs parties, qui comprend, la partie Nucelus et la partie des Modules. Nous présentons également une autre partie qui présente la gestion des entrées/sorties et la communication.

Voici un rappel de l'architecture matériel. L'architecture physique du projet repose sur la configuration de ce schéma :

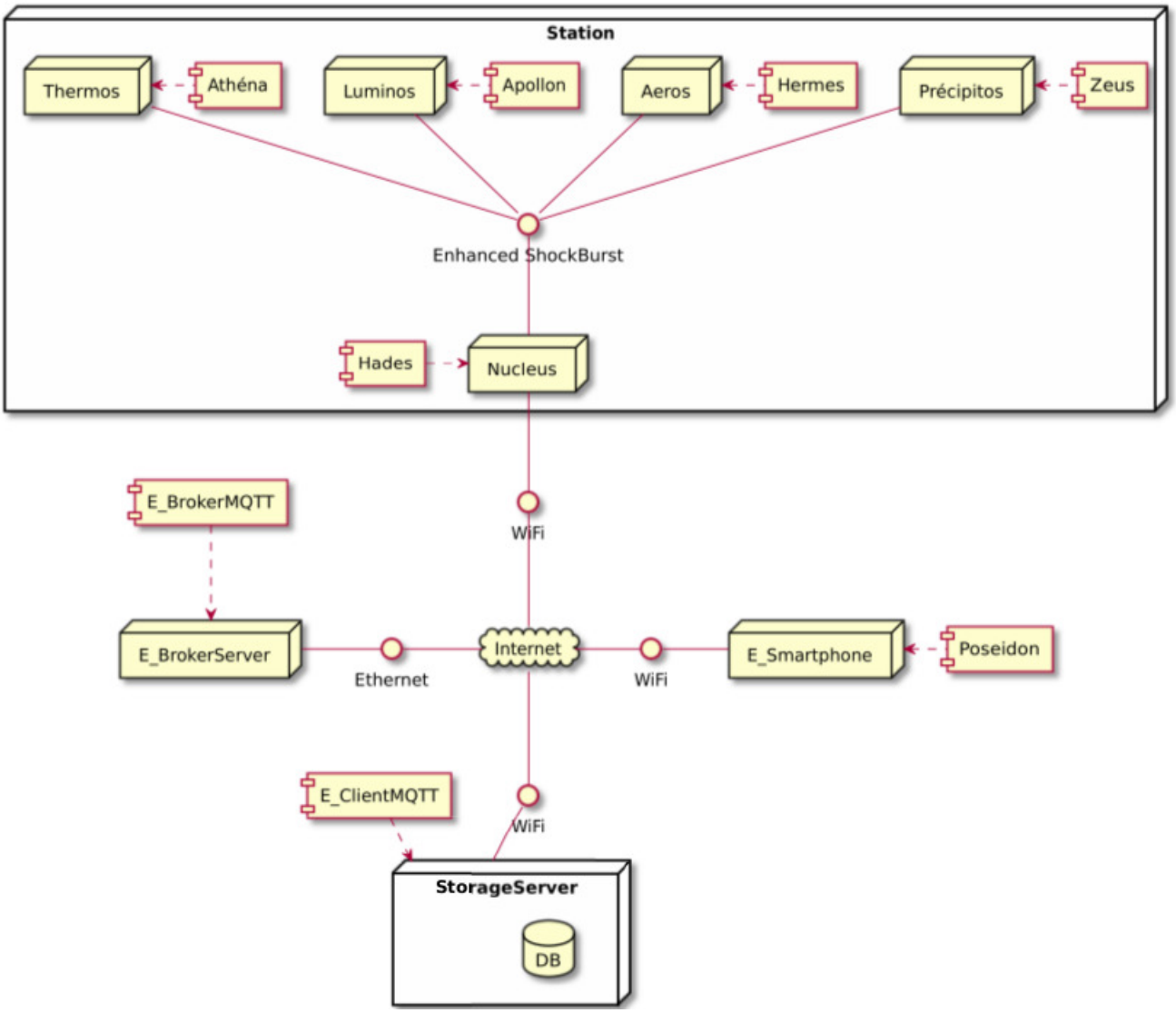

FIGURE 3.1 – Schéma de l'architecture matérielle simplifiée

 © Equipe B1 - Tous droits réservés ESEO

3.1.1 Schéma Nucleus

Le diagramme ci-dessous représente l'architecture logiciel Hades de Nucleus dans son intégralité.

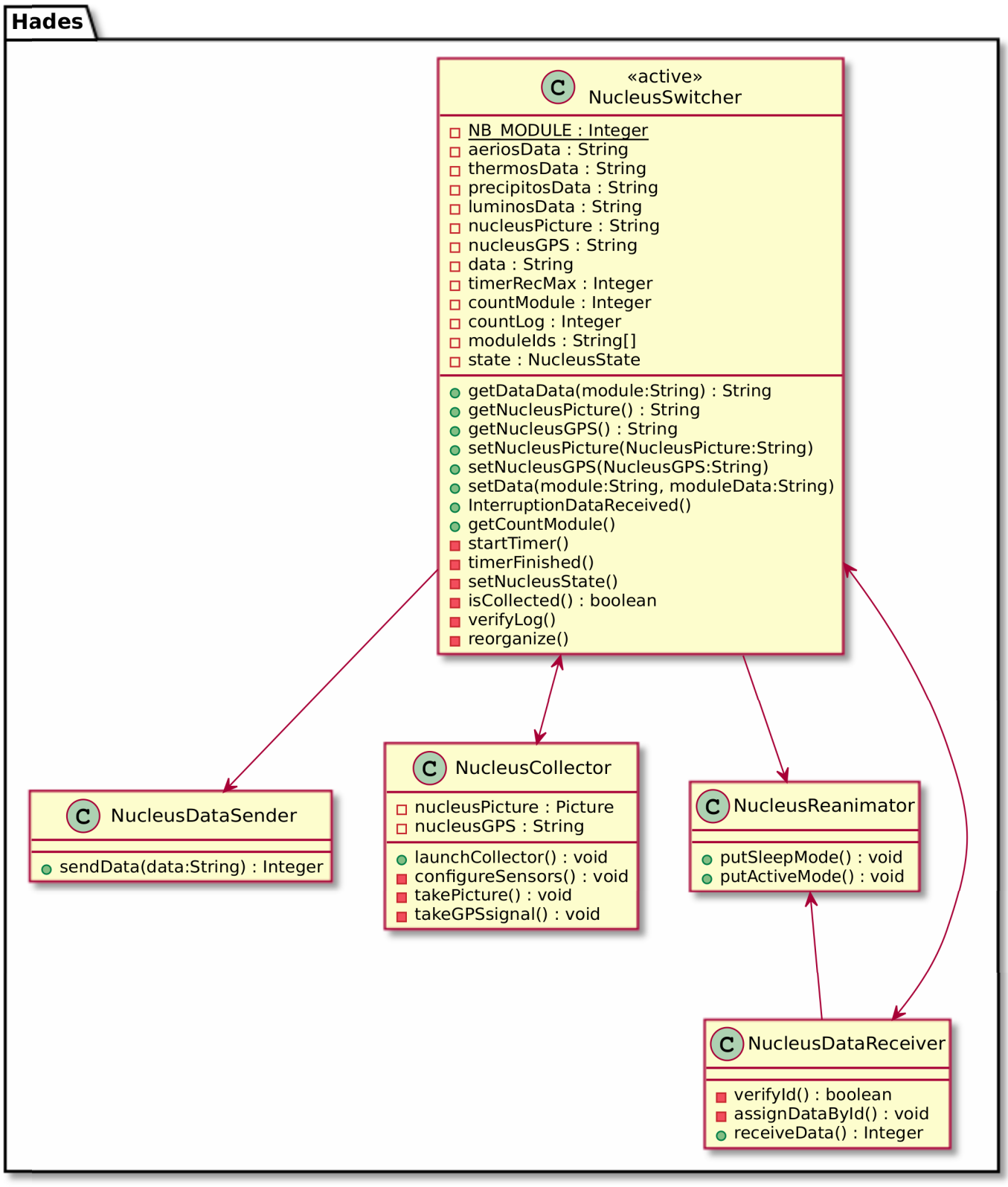

FIGURE 3.2 – Schéma de MasterChef

3.2 Description des classes

3.2.1 Description des classes de Nucleus

3.2.1.1 La Classe NucleusSwitcher «Singleton»

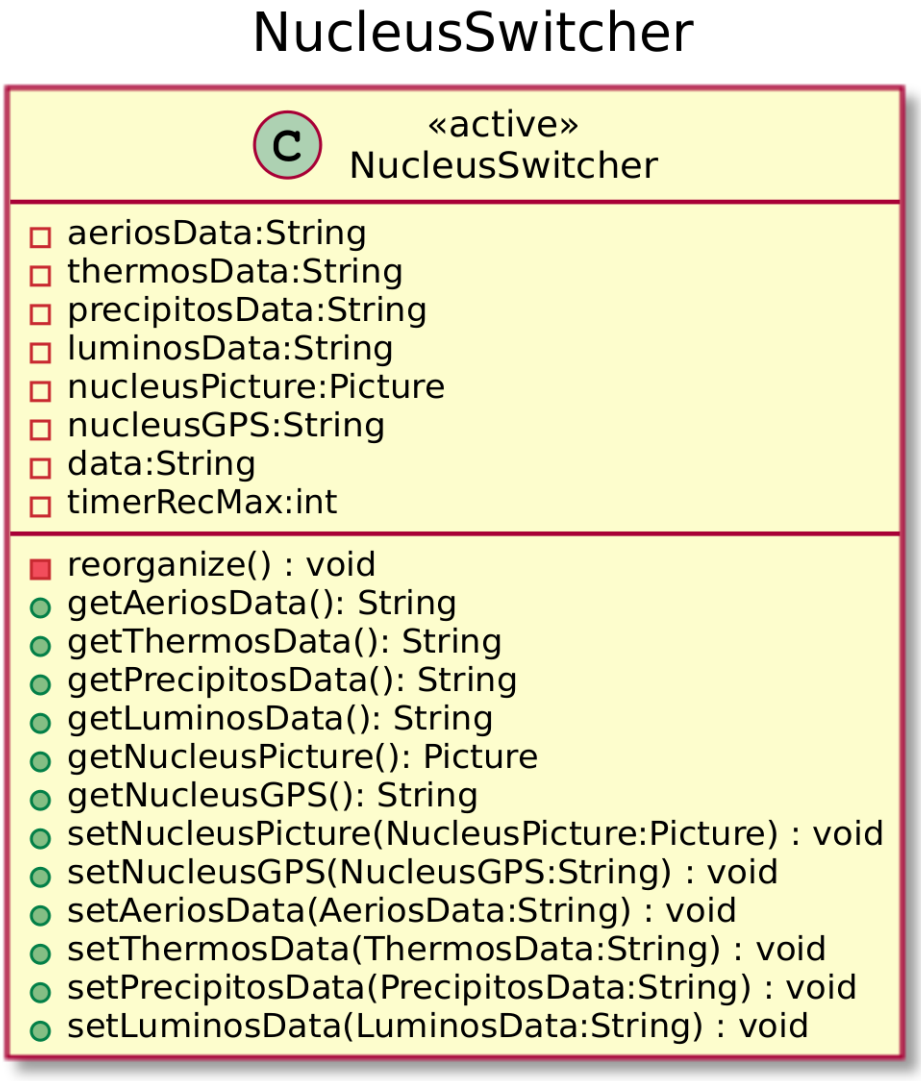

FIGURE 3.3 – Description de la classe NucleusSwitcher.

3.2.1.1.1 Philosophie de conception

La classe NucleusSwitcher est la classe du Nucleus responsable de la machine à états de la Nucleus. NucleusSwitcher gère les différentes actions effectués par la Nucleus.

Cette classe se distingue sous la forme d'un Singleton.

Responsabilités :

— Initialiser les classes de la Nucleus.

— Gèrer la machine à états de la Nucleus en appelant les méthodes des autres classes.

Collaborateurs :

— NucleusDataReceiver

— NucleusCollector

 © Equipe B1 - Tous droits réservés ESEO

— NucleusDataSender
— NucleusReanimator

3.2.1.1.2 Description structurelle

3.2.1.1.2.1 Attributs

— aerosData : String : les informations provenant de aeros.
— thermosData : String : les informations provenant de thermos.
— precipitosData : String : : les informations provenant de precipitos.
— luminosData : String : les informations provenant de luminos.
— nucleusPicture : VOIR : la photo prise par NucleusCollector.
— nucleusGPS : String : la localisation GPS prise pas NucleusCollector.
— TIMER_REC_MAX : int : le temps maximum que la Nucleus attend entre la première reception et la reception suivante avant de se remettre en sommeil.
— dataSend : String : l'ensemble des informations des différents modules une fois mises en forme.

3.2.1.1.2.2 Services offerts

— reorganize() : Met en forme l'ensemble des informations reçus pour pouvoir les envoyer ensuite.
— Accesseurs et mutateurs des attributs module*Data, nucleusPicture et nucleusGPS.

3.2.1.1.3 Gestion du multitâche

Il n'y a pas de gestion de multitâche sur l'ESP8266, les instructions sont séquentielles.

3.2.1.1.4 Gestion des entrées/sorties

NucleusSwitcher possède deux entrées physiques pour effectuer des relevés sur la caméra et sur le module GPS. La caméra envoie une donnée sous forme de chaîne de caractère via une liaison série SPI. Le GPS envoi lui-aussi une donnée sous forme de chaîne de caractère en utilisant une liaison série UART.

3.2.1.1.5 Gestion comportementale

La réception est échantillonnée toutes les 15 minutes, le logiciel surveille et bascule le microprocesseur du "Sleep Mode" au "Active Mode" lorsqu'une émission de donnée en provenance d'un module est détecté.

Ce qu'il faut retenir c'est que, pendant une durée de 15 minutes, la Nucleus reçoit les données de tous les modules. Le cas échéant, la Nucléus comprendra que le module absent est déconnecté ou hors service. A la fin de ce temps donné, le logiciel Hades met en forme une trame pour la transmettre sur le serveur MQTT.

 © Equipe B1 - Tous droits réservés ESEO

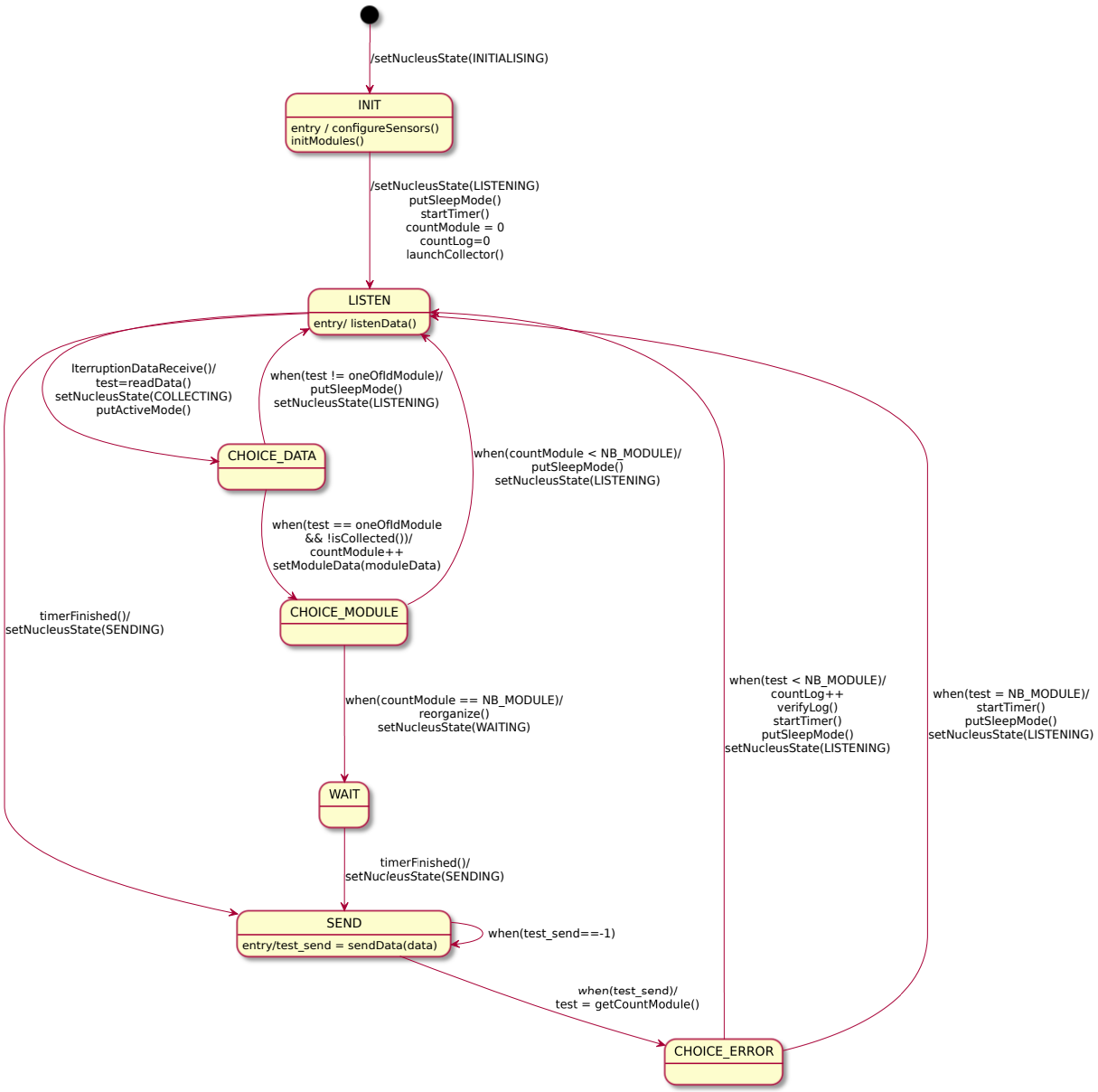

FIGURE 3.4 – Description de la machine à état du switcher de Nucleus.

3.2.1.2 La Classe NucleusCollector «Singleton»

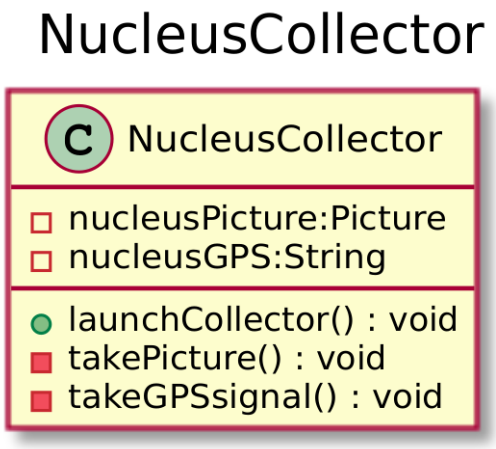

FIGURE 3.5 – Description de la classe NucleusCollector.

3.2.1.2.1 Philosophie de conception

La classe NucleusCollector est la classe du Nucleus responsable de la prise de la photo et de la localisation GPS de la Nucleus.

Cette classe se distingue sous la forme d'un Singleton.

Responsabilités :

— Prendre une photo.

— Localiser la Nucleus.

Collaborateur :

— NucleusSwitcher.

3.2.1.2.2 Description structurelle

3.2.1.2.2.1 Attributs

— nucleusPicture : String : Buffer contenant les données de la photo prise par la caméra.

— nucleusGPS : String : Chaîne de caractère correspondant à la localisation GPS.

3.2.1.2.2.2 Services offerts

— launchCollector() : Demande de lancer la collecte d'information.

— takePicture() : Capture la photo.

— takeGPS() : Prend la localisation GPS.

— Accesseurs et mutateurs des attributs.

 © Equipe B1 - Tous droits réservés ESEO

3.2.1.3 La Classe NucleusReanimator «Singleton»

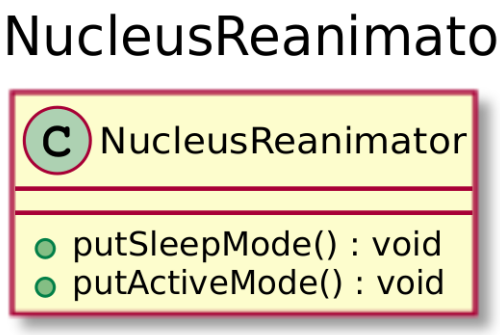

FIGURE 3.6 – Description de la classe NucleusReanimator.

3.2.1.3.1 Philosophie de conception

La classe NucleusReanimator est la classe de la Nucleus responsable de la mise en sommeil du système et de son reveil.

Cette classe se distingue sous la forme d'un Singleton.

Responsabilités :

— Endormir la Nucleus.

— Reveiller la Nucleus.

Collaborateurs :

— NucleusSwitcher.

— NucleusDataReceiver.

3.2.1.3.2 Description structurelle

3.2.1.3.2.1 Services offerts

— putSleepMode() : Mets la Nucleus en mode sleep.

— putActiveMode() : Mets la Nucleus en mode actif.

 © Equipe B1 - Tous droits réservés ESEO

3.2.1.4 La Classe NucleusDataReceiver «Singleton»

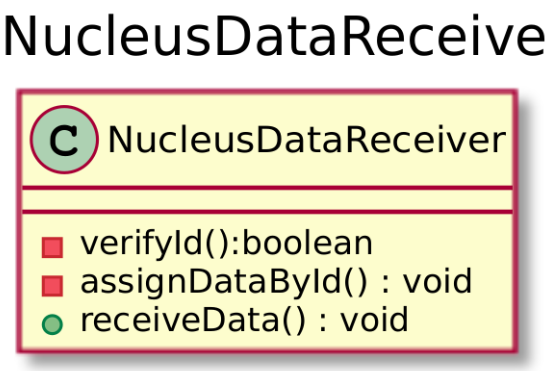

FIGURE 3.7 – Description de la classe NucleusDataReceiver.

3.2.1.4.1 Philosophie de conception

La classe NucleusDataSender est la classe du Nucleus responsable de la réception des données provenant des nœuds de capteurs.

Cette classe se distingue sous la forme d'un Singleton.

Responsabilités :

- Recevoir les données envoyées par les noeuds de capteurs.
- Vérifier si le message reçu est correct.
- Trouver de quel interlocuteur provient le message.
- Enregistrer les données reçus au bon endroit en mémoire.
- Réveiller le Nucleus si on est en mode sleep.

Collaborateurs :

- NucleusSwitcher.
- NucleusReanimator.

3.2.1.4.2 Description structurelle

3.2.1.4.2.1 Services offerts

- receiveData() : String : Récupère les données envoyées par le nœud de capteurs.
- verifyId(data : String) : Boolean : Vérifie le message reçu pour savoir s'il est valide et trouve à qui appartient le message.
- assignDataById(data : String) : Enregistre les données reçues au bon endroit en mémoire.

 © Equipe B1 - Tous droits réservés ESEO

3.2.1.5 La Classe NucleusDataSender «Singleton»

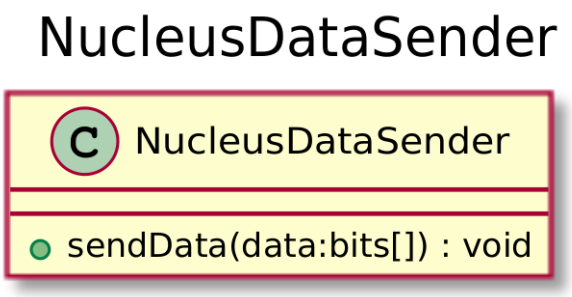

FIGURE 3.8 – Description de la classe NucleusDataSender.

3.2.1.5.1 Philosophie de conception

La classe NucleusDataSender est la classe du Nucleus responsable de la communication entre le Nucleus et le Broker MQTT. Son rôle est d'envoyer les données récoltées grâce au protocole MQTT.

Cette classe se distingue sous la forme d'un Singleton.

Responsabilité :

— Envoyer les données au Broker MQTT.

Collaborateur :

— NucleusSwitcher.

3.2.1.5.2 Description structurelle

3.2.1.5.2.1 Services offerts

— sendData(data : String) : Envoie l'ensemble des données relevées au broker MQTT.

 © Equipe B1 - Tous droits réservés ESEO

3.2.2 Schéma Modules

Le diagramme ci-dessous représente l'architecture logiciel générique des logiciels "Athena", "Apollon", "Hermes" et "Zeus" des différents modules.

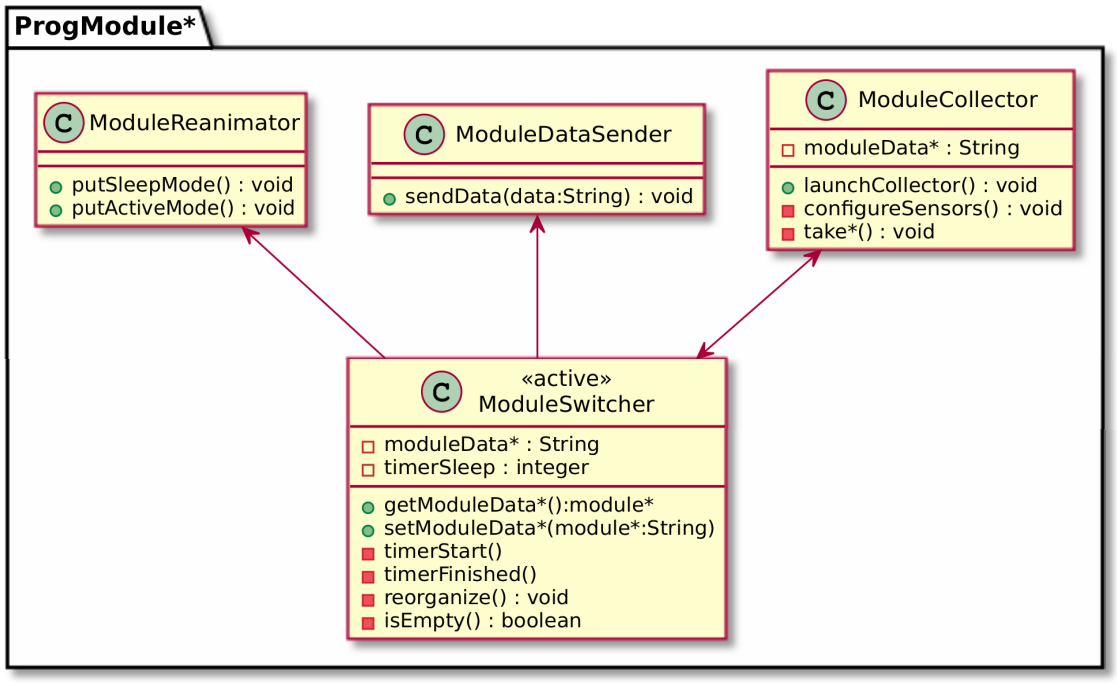

FIGURE 3.9 – Architecture logicielle générique des modules

 © Equipe B1 - Tous droits réservés ESEO

3.2.3 Description des classes des Modules

3.2.3.1 La Classe ModuleSwitcher «Singleton»

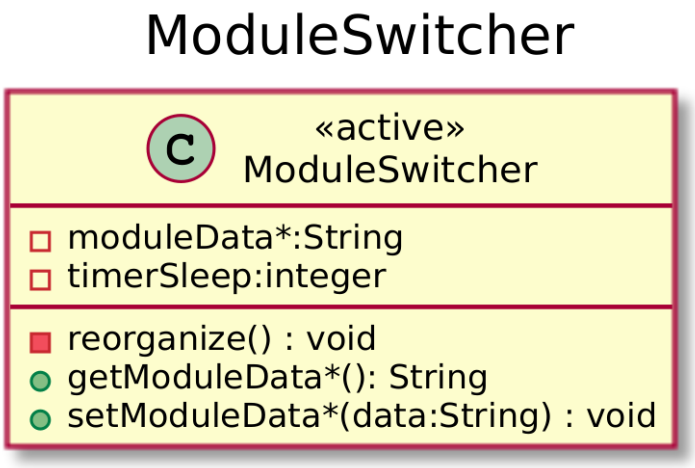

FIGURE 3.10 – Description de la classe ModuleSwitcher.

3.2.3.1.1 Philosophie de conception

La classe ModuleSwitcher est la classe du nœud de capteurs responsable de la machine à état des nœuds de capteurs. ModuleSwitcher gère les différentes actions effectués par les modules.

Cette classe se distingue sous la forme d'un Singleton. Elle se décline en 4 formes en fonctions du nœud de capteurs dans lequel elle opère : ThermosSwitcher dans Athéna, LuminosSwitcher dans Apollon, AerosSwitcher dans Hermes et PrecipitosSwitcher dans Zeus.

Cette classe se distingue sous la forme d'un Singleton.

Responsabilités :

— Initialiser les classes du module.

— Gérer la machine à état du module en appelant les méthodes des autres classes.

Collaborateurs :

— ModuleDataReceiver

— ModuleCollector

— ModuleDataSender

— ModuleReanimator

3.2.3.1.2 Description structurelle

3.2.3.1.2.1 Attributs

— dataSend : String : les informations récupérés par le module, une fois mises en forme.

— TIMER_SLEEP : int : le temps de sommeil du module.

3.2.3.1.2.2 Services offerts

— reorganize() : mets en forme l'ensemble des informations reçues pour pouvoir les envoyer ensuite.

— Accesseurs et mutateurs des attributs module*Data, nucleusPicture et nucleusGPS.

 © Equipe B1 - Tous droits réservés ESEO

3.2.3.1.3 Gestion du multitâche

Il n'y a pas de gestion de multitâche puisque les instructions sur la puce "NRF24LE1" sont exécutées séquentiellement.

3.2.3.1.4 Gestion comportementale

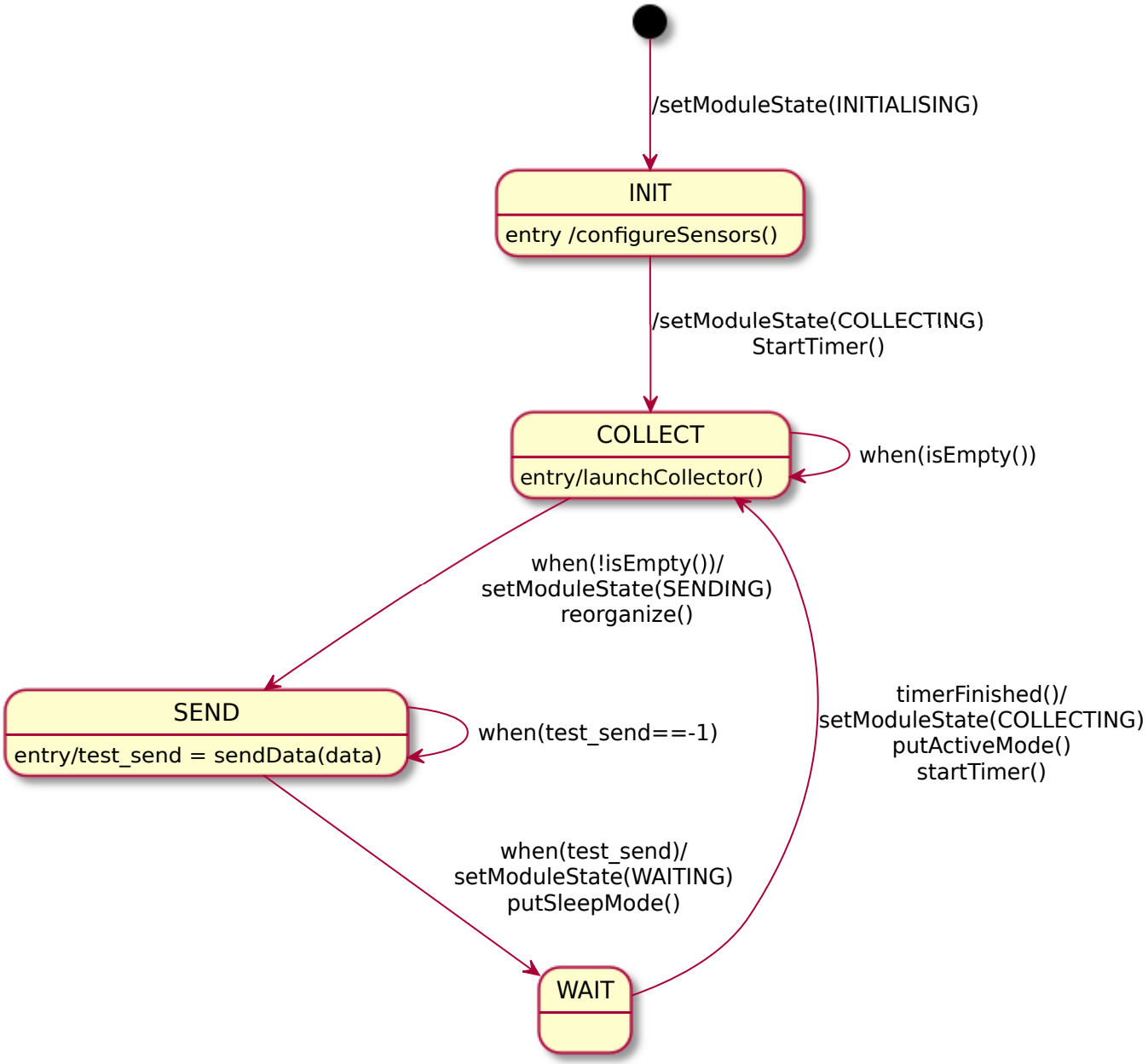

FIGURE 3.11 – Description de la machine à état d'un module.

 © Equipe B1 - Tous droits réservés ESEO

3.2.3.2 La Classe ModuleCollector «Singleton»

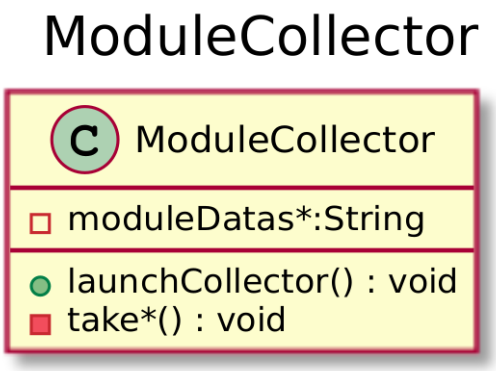

FIGURE 3.12 – Description de la classe ModuleCollector.

3.2.3.2.1 Philosophie de conception

La classe ModuleCollector est la classe du noeud de capteurs responsable de la collecte des données environnementales.

Cette classe se distingue sous la forme d'un Singleton. Elle se décline en 4 différentes formes en fonctions du noeud de capteurs dans lequel elle opère : ThermosCollector dans Athéna, LuminosCollector dans Apollon, AerosCollector dans Hermes et enfin PrecipitosCollector dans Zeus.

Responsabilités :

Pour ThermosCollector :

— Relever la température.

— Relever l'humidité.

— Relever la pression.

Pour LuminosCollector :

— Relever la luminosité.

— Relever l'UV.

Pour AerosCollector :

— Relever le taux de dioxyde de carbone.

— Relever le taux de monoxyde de carbone.

— Relever le taux de d'ozone.

— Relever le taux de dioxyde d'azote.

— Relever le taux de d'ammoniac.

Pour PrecipitosCollector :

— Relever la précipitation.

— Relever la vitesse du vent.

— Relever la direction du vent.

— Relever les particules.

Collaborateur :

— ModuleSwitcher associé.

 © Equipe B1 - Tous droits réservés ESEO

3.2.3.2.2 Description structurelle

3.2.3.2.2.1 Attributs

Pour ThermosCollector :

- thermosTemperature : Float : la température
- thermosHumidity : Float : l'humidité.
- thermosPressure : Float : la pression.

Pour LuminosCollector :

- luminosBrightness :Float : la luminosité.
- luminosUV : Float : l'UV.

Pour AerosCollector :

- aerosCO2 : Float : le taux de dioxyde de carbone.
- aerosCO : Float : le taux de monoxyde de carbone.
- aerosO3 : Float : le taux de d'ozone.
- aerosNO2 : Float : le taux de dioxyde d'azote.
- aerosNH3 : Float : le taux de d'ammoniac.

Pour PrecipitosCollector :

- precipitosPrecipitations : Float : la précipitation.
- precipitosWindSpeed : Float : la vitesse du vent.
- precipitosWindDirection : Direction : la direction du vent.
- precipitosParticules : Float : le taux de particules fines dans l'air.

3.2.3.2.2.2 Services offerts

Global :

- Accesseurs et mutateurs des attributs.

Pour ThermosCollector :

- takeTemperature : float : Relève la température
- takeHumidity : float : Relève l'humidité.
- takePressure : float : Relève la pression.

Pour LuminosCollector :

- takeBrightness : float : Relève la luminosité.
- takesUV : float : Relève l'UV.

Pour AerosCollector :

- takeCO2 : float : Relève le taux de dioxyde de carbone.
- takeCO : float : Relève le taux de monoxyde de carbone.
- takeO3 : float : Relève le taux de d'ozone.
- takeNO2 : float : Relève le taux de dioxyde d'azote.
- takeNH3 : float : Relève le taux de d'ammoniac.

Pour PrecipitosCollector :

- takePrecipitations : float : Relève la précipitation.
- takeWindSpeed : float : Relève la vitesse du vent.
- takeWindDirection : Direction : Relève la direction du vent.
- takeParticules : float : Relève le taux de particules fines dans l'air.

 © Equipe B1 - Tous droits réservés ESEO

3.2.3.2.3 Gestion des entrées/sorties

On utilise les liaisons série "UART", "SPI" et "I2C" pour communiquer avec les différents capteurs.

 © Equipe B1 - Tous droits réservés ESEO

3.2.3.3 La Classe ModuleReanimator «Singleton»

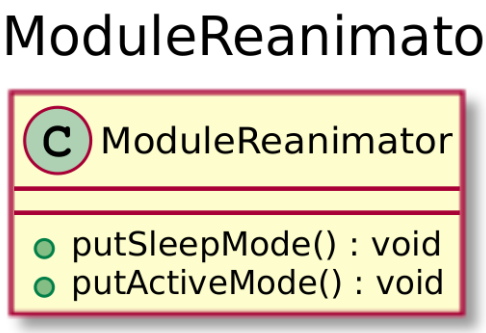

FIGURE 3.13 – Description de la classe ModuleReanimator.

3.2.3.3.1 Philosophie de conception

La classe ModuleReanimator est la classe du nœud de capteurs responsable de la mise en sommeil du système et de son réveil.

Cette classe se distingue sous la forme d'un Singleton. Elle se décline en 4 différentes formes en fonction du nœud de capteurs dans lequel elle opère : ThermosReanimator dans Athéna, LuminosReanimator dans Apollon, AerosReanimator dans Hermes et PrecipitosReanimator dans Zeus.

Responsabilités :

- Endormir la module.
- Réveiller la module.

Collaborateurs :

- ModuleSwitcher associé.
- ModuleDataReceiver associé.

3.2.3.3.2 Description structurelle

3.2.3.3.2.1 Services offerts

- putSleepMode() : Met le module en mode sleep.
- putActiveMode() : Met le module en mode actif.

 © Equipe B1 - Tous droits réservés ESEO

3.2.3.4 La Classe ModuleDataSender «Singleton»

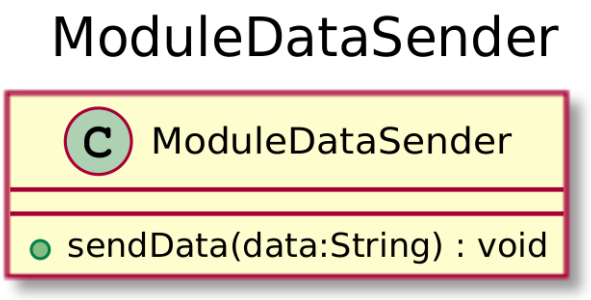

FIGURE 3.14 – Description de la classe ModuleDataSender.

3.2.3.4.1 Philosophie de conception

La classe ModuleDataSender est la classe des noeuds de capteurs responsable de la communication entre le nœud de capteur et la Nucleus. Son rôle est d'envoyer les données récoltées par ShockBurst.

Cette classe se distingue sous la forme d'un Singleton. Elle se décline en 4 formes en fonctions du nœud de capteurs dans lequel elle opère : ThermosDataSender dans Athéna, LuminosDataSender dans Apollon, AerosDataSender dans Hermes et PrecipitosDataSender dans Zeus.

Responsabilité :

— Envoyer les données à la Nucleus.

Collaborateur :

— ModuleSwitcher associé.

3.2.3.4.2 Description structurelle

3.2.3.4.2.1 Services offerts

— sendData(data : String) : Envoie l'ensemble des données relevées à la Nucleus.

 © Equipe B1 - Tous droits réservés ESEO

3.3 Décomposition de GUI

3.3.1 Schéma MVC de la classe GUI

Dans cette partie, on représente la classe GUI avec le modèle MVC. Ce dernier respecte un modèle destiné aux interfaces graphique, qui permet notamment d'obtenir une architecture divisée en trois packages différents. Le package modèle contient les données à afficher, le package vue contient toues les vues de l'application et le package contrôleur contient la logique concernant les actions de l'utilisateur.

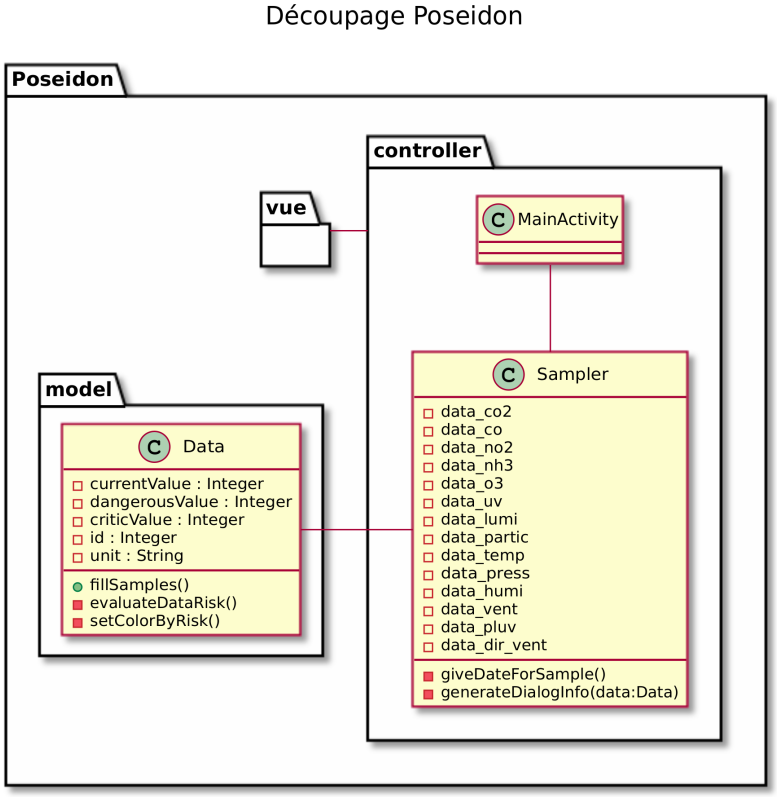

FIGURE 3.15 – Description du modèle MVC de la classe GUI

— Le package "view" représente les différents layout en "*.xml" de l'application.

— Le package "models" représente les objets métier de notre projet.

— Le package "controller" représente les activités et ses différentes classes de gestion des activités.

 © Equipe B1 - Tous droits réservés ESEO

3.3.2 Description des classes de GUI

3.3.2.1 La Classe Data

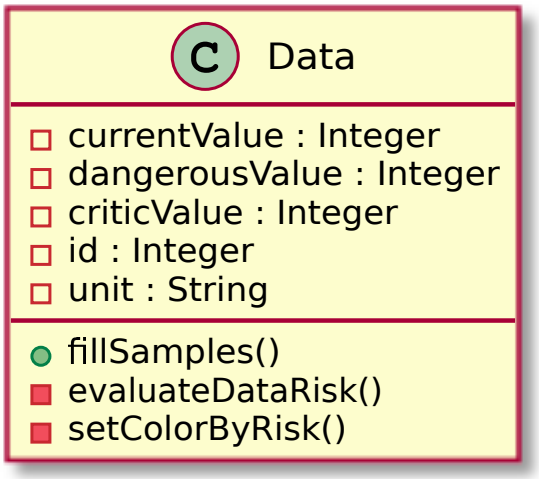

FIGURE 3.16 – Description de la classe Data.

3.3.2.1.1 Philosophie de conception

Classe appartenant au package "model" qui spécifie l'objet métier "Data". "Data" est utiliser pour instancier une nouvelle donnée.

Responsabilités :

— Instancier des données.

— Évaluer le risque.

Collaborateurs :

— Sampler associé.

3.3.2.1.2 Description structurelle

3.3.2.1.2.1 Attributs

— **id : Integer** : l'identifiant d'une donnée.

— **criticValue : Integer** : La valeur critique d'une donnée.

— **dangerousValue : Integer** : La valeur dangereuse pour l'homme d'une donnée.

— **currentValue : Integer** : La valeur courante de la donnée.

— **unit : String** : Chaîne de caractère correspondant à l'unité utilisée pour la donnée.

— **name : String** : Le nom de la donnée.

3.3.2.1.2.2 Services offerts

— **fillSamples(List : Data[]) : void** : rempli les données dans les zones de texte correspondant.

— **evaluateDataRisk(data :Data) : void** : évalue le risque de la donnée passée en paramètre.

— **setColorByRisk(aTextView :TextView, priority :Integer) : void** : donne une couleur à la zone de texte en fonction de son risque.

 © Equipe B1 - Tous droits réservés ESEO

3.3.2.2 La Classe Sampler

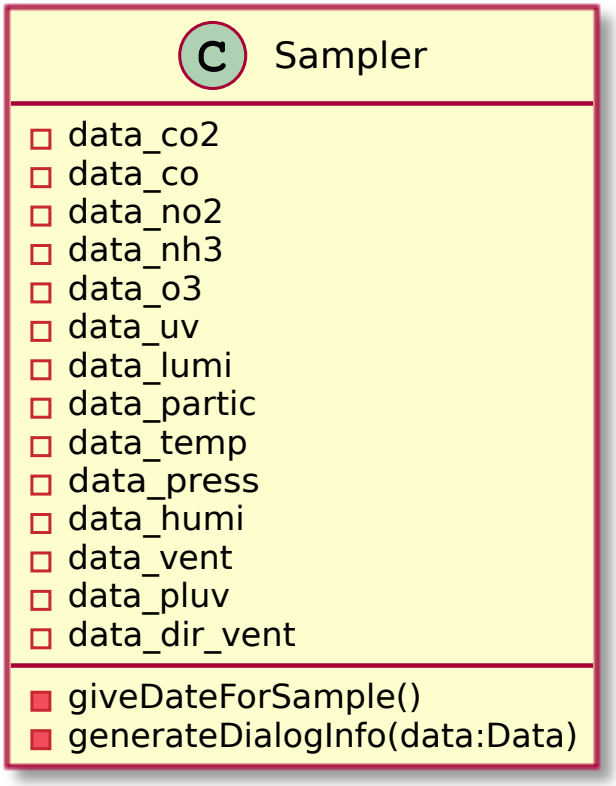

FIGURE 3.17 – Description de la classe Sampler.

3.3.2.2.1 Philosophie de conception

La classe Sampler est la classe principale du package "controller" et permet d'établir la communication avec la base de donnée afin de rafraîchir les données internes.

Elle a aussi le rôle de mettre en forme et d'informer l'utilisateur des valeurs actuelles, critiques, dangereuses et des courbes sur chacune des données.

Responsabilités :

- Rafraîchir les données.
- Établir des boites de dialogue informatives.
- Établir des courbes de données.

Collaborateurs :

- MainActivity associé.
- Data associé.

3.3.2.2.2 Description structurelle

3.3.2.2.2.1 Attributs

- **data_co2 : String** : Chaîne de caractère pour identifier le taux de co2 dans l'air.
- **data_co : String** : Chaîne de caractère pour identifier le taux de co dans l'air.
- **data_no2 : String** : Chaîne de caractère pour identifier le taux de no2 dans l'air.

 © Equipe B1 - Tous droits réservés ESEO

— **data_nh3 : String** : Chaîne de caractère pour identifier le taux de nh3 dans l'air.
— **data_o3 : String** : Chaîne de caractère pour identifier le taux de o3 dans l'air.
— **data_uv : String** : Chaîne de caractère pour identifier l'indice UV.
— **data_lumi : String** : Chaîne de caractère pour identifier la luminosité.
— **data_partic : String** : Chaîne de caractère pour identifier le taux de particules dans l'air.
— **data_temp : String** : Chaîne de caractère pour identifier la température ambiante extérieur.
— **data_press : String** : Chaîne de caractère pour identifier la pression atmosphérique.
— **data_humi :String** : Chaîne de caractère pour identifier l'humidité.
— **data_vent : String** : Chaîne de caractère pour identifier la vitesse du vent.
— **data_pluv : String** : Chaîne de caractère pour identifier le nombre de mini-mètre d'eau tombé sur une période d'environ 15 minutes.
— **data_dir_vent : String** : Chaîne de caractère pour identifier la direction du vent.

3.3.2.2.2.2 Services offerts

— **giveDateForSample() : void** : Donne la date et l'heure précise à la seconde du dernier relevé.
— **generateDialogInfo() : void** : Génère des boites de dialogue pour donner des informations à l'utilisateur.

 © Equipe B1 - Tous droits réservés ESEO

3.3.2.3 La Classe MainActivity

FIGURE 3.18 – Description de la classe MainActivity.

3.3.2.3.1 Philosophie de conception

Responsabilités :

— Gérer les intentions utilisateurs.

Collaborateurs :

— Sampler associé.

 © Equipe B1 - Tous droits réservés ESEO

3.4 Protocole de communication

Les échanges entre Poseidon et la base de données se feront par l'intermédiaire d'un réseau internet considéré comme un service web nommé "Phpmyadmin".

A chaque ouverture de l'application, "Poseidon" vient chercher les derniers relevés présents sur la base de données. Ensuite il effectue un cycle de renouvellement des données toutes les 15 minutes en utilisant le même processus.

3.4.1 Formalisation du protocole

La réponse JSON envoyée à l'application est encodée d'une manière spécifique. Ci-dessous sont définis les l'encodage des composants de façon formelle.

3.4.1.1 Envoyer les résultats d'un scénario

ProxyTestCampaign de MasterChef demande à TestCampaign de MasterMind de mettre à jour les résultats d'un scenario.

```
{
  "type": 4,
  "scenario": "scenario_2017_02_05_14_32_13",
  "results": [{
    "script": "SCRIPT_EXEMPLE3",
    "results": [{
      "name": "PING",
      "result": "PASS"
    }, {
      "name": "CPU",
      "result": "FAIL"
    }, {
      "name": "ETH",
      "result": "FAIL"
    }],
    "log": "Lorem ipsum dolor sit amet, consectetur adipiscing
       elit, sed do eiusmod tempor incididunt ut labore et dolore
        magna aliqua."
  }, {
    "script": "SCRIPT_EXEMPLE4",
    "results": [{
      "name": "TRY",
      "result": "PASS"
    }, {
      "name": "CATCH",
      "result": "PASS"
    }, {
      "name": "SWITCH",
      "result": "PASS"
```

 © Equipe B1 - Tous droits réservés ESEO

```
28      }],
29      "log": ""
30    }]
31  }
```

 © Equipe B1 - Tous droits réservés ESEO

Chapitre 4

Dictionnaire du Domaine

Ce dictionnaire du domaine s'ajoute au dictionnaire du domaine de la spécification, qui défini davantage de termes.

Activité : Élément Android représentant un écran. Une activité contient tout les éléments de l'écran comme un bouton, un message, etc. Elle représente donc ce que l'utilisateur peut faire durant son exécution.

Adresse IP : C'est un numéro d'identification propre à chaque appareil connecté à un réseau informatique. On utilise le protocole IPv4.

Adresse MAC : Adresse unique constituée de 6 octets variant de 0 à 255 bits pour toutes les cartes électronique. Cette adresse est fixée par le constructeur afin de d'identifier de façon unique une carte sur un réseau local. C'est avec cette adresse que MasterPower va identifier les DUT branchés.

Connexion serial : Transmission de données dans laquelle les éléments d'information se succèdent.

I2C : (Inter-Integrated Circuit) C'est un bus informatique de transmission de données.

JSON : (JavaScript Object Notation) Est un format de données textuelles, il permet de représenter de l'information de façon structurée.

Packages : Correspond au regroupement de plusieurs classes pour organiser et établir une hiérarchie dans développement C et Android.

Port : Numéro correspondant à la couche de transport du modèle OSI, la notion de port logiciel permet, sur un ordinateur donné, de distinguer différents interlocuteurs.

SPI : Une liaison SPI (pour Serial Peripheral Interface) est un bus de données série synchrone. Les circuits communiquent selon un schéma maître-esclaves, où le maître contrôle la communication.

Thread : Fil d'exécution dans un programme.

Timeout : Définition d'un temps pour lequel une exception est levée.

UART : (Universal Asynchronous Receiver Transmitter) C'est un émetteur-récepteur asynchrone universel.

UIThread : Fil d'exécution qui donne la priorité pour l'affichage graphique d'une application.

 © Equipe B1 - Tous droits réservés ESEO

Table des figures

Liste des tableaux