Dossier de conception de SmartHouseSim

Incrément 1

B2_2024
SmartHouseSim
Davidson Consulting
UKIO

Version : 0.0
Révision : 2
État du document : en attente de validation
Responsable du document : Valentin HUBERT
16 avril 2023

AVERTISSEMENT :

Le présent document est à but pédagogique. Il a été réalisé par les étudiants Théo FAUCHER, Eliot COULON, Valentin HUBERT, Antoine LARGEAUD, Quentin SOUTELO et Léo LAMANAC de l'équipe B2_2024 de l'option Systèmes Embarqués (SE) de l'ESEO sous la direction de Fréderic JOUAULT et de Jérôme DELATOUR. Ce document est la propriété de ses auteurs. En dehors des activités pédagogiques de l'ESEO, ce document ne peut être diffusé ou recopié sans l'autorisation écrite de l'un de ses propriétaires.

Le modèle de ce document a été mis à disposition par Edouard GAUTIER étudiant de l'ESEO selon le plan rédigé par Jérôme DELATOUR.

La société UKIO n'a pas d'existence légale, il s'agit d'une société fictive créée pour des besoins pédagogiques.

 ©2023 Droits réservés, UKIO

Table des versions

Date	Description	Auteur	Version	Révision
18/03/2023	Création du document	Théo FAUCHER	0.0	0
18/03/2023	Ajout de l'introduction	Quentin SOUTELO	0.0	1
28/03/2023	Correction de l'introduction	Léo LAMANAC	0.0	2

©2023 Droits réservés, UKIO

Table des matières

 ©2023 Droits réservés, UKIO

 ©2023 Droits réservés, UKIO

1 Introduction

1.1 Objet

Ce dossier de conception a pour objectif de définir la structuration et le comportement du projet SmartHouseSim. Il permettra à UKIO de développer le projet ainsi qu'élaborer les tests.

Les éléments de conception présentés dans ce document ont été déterminés suite à l'étude du dossier de spécification [SPEC_B2].

Ce dossier de conception suit les recommandations de la norme IEEE 830 [IEEE-830_1998]. Il utilise des schémas et illustrations respectant la norme UML en version 2.5 [UML_2.5.1_2017]. Il respecte les exigences du Plan d'Assurance Qualité Logicielle (PAQL) défini par l'équipe B2 [PAQL_B2].

1.2 Portée

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

- à l'équipe de développement C et celle de développement Android afin de préciser l'implémentation des objets constituants le SaE.
- aux testeurs, afin qu'ils puissent élaborer les tests adéquats vérifiant la philosophie de conception adoptée 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 clairs et dans la continuité des spécifications.

 ©2023 Droits réservés, UKIO

1.3 Définitions, acronymes et abréviations

Les abréviations utilisées dans le présent document sont répertoriées et expliquées dans le tableau ci-dessous. Les termes utiles pour interpréter correctement ce document sont définis dans le dictionnaire du domaine section 3.3.

Terme	Description
Client	Entreprise Davidson Consulting
CU	Cas d'Utilisation
IEEE	(Institute of Electrical and Electronics Engineers) Association professionnelle internationale définissant entre autres des, normes dans le domaine informatique et électronique.
N.A	Non Applicable
OMG	(Object Management Group) Association professionnelle internationale définissant entre autres, des normes dans le domaine informatique.
pp.	Abréviation utilisée pour référencer une ou plusieurs page(s).
RDPS	Référentiel Documentaire Projet SmartHouseSim
SaE	(Système à l'étude) Il s'agit de l'ensemble des composants SmartHouseSoft, SmartHouseApp et SmartHouseModel.
UML	(Unified Modeling Language) Notation graphique normalisée, ddéfinie par l'OMG et utilisée en génie logiciel.

1.4 Références

Voici un tableau récapitulatif des documents utilisés pour la réalisation de ce dossier de spécification, ainsi que des liens pour accéder aux éventuelles sources.

Nom du document	Référence
[CDC_Davidson_prose2023_v1.2]	Société Davidson Consulting, « Plateforme de simulation d'effacement énergétique d'un foyer », version 1.2, disponible sur le RDPS.
[CR_09_02]	Société UKIO, "Compte-rendu de la réunion du 09/02/2023", disponible sur le RDPS.
[SPEC_B2]	Société UKIO, "Dossier de spécification du projet SmartHouseSoft", Projet SmartHouseSoft, disponible sur le RDPS.
[PAQL_B2]	Société UKIO, "Plan d'Assurance Qualité Logicielle", Projet SmartHouseSoft, disponible sur le RDPS.

Suite à la page suivante...

 ©2023 Droits réservés, UKIO

. . . suite de la page précédente.

Nom du document	**Référence**
[ISO/IEC/IEEE 29148 :2018]	ISO/IEC/IEEE, "International Standard, Systems and software engineering - Life cycle processes - Requirements engineering", 2018, https://standards.ieee.org/standard/29148-2018.html.
[UML_2.5.1_2017]	OMG, "Unified Modeling Language", version 2.5.1, 2017.

 ©2023 Droits réservés, UKIO

1.5 Vue d'ensemble

Ce document de conception est structuré en plusieurs parties :

- une première partie, qui concerne la conception générale du prototype SmartHouseSoft. Cette partie présente l'architecture candidate et donne les grands principes de fonctionnement du projet SmartHouseSoft. 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.
- 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, le multitâche ainsi que la gestion de la persistance.
- un dictionnaire de domaine constitue la dernière partie du document.

 ©2023 Droits réservés, UKIO

1.6 Versions

Le projet SmartHouseSim est réalisé en 2 incréments. Nous avons détaillé le système pour les versions 1 et 2, nous devons alors définir ce que fera la version 1 et ce que fera la version 2.

- Lors de la version 1, SmartHouseApp et SmartHouseModel doivent pouvoir interagir avec SmartHouseSoft. L'ensemble des fonctionnalités présentes pour l'incrément 1 doivent être assurées.
- Lors de la version 2, toutes les fonctionnalités présentes lors de l'incrément 1 doivent être assurées. De plus, les fonctionnalités présentes pour l'incrément 2 doivent être assurées.

 ©2023 Droits réservés, UKIO

2 Conception générale

2.1 Architecture candidate

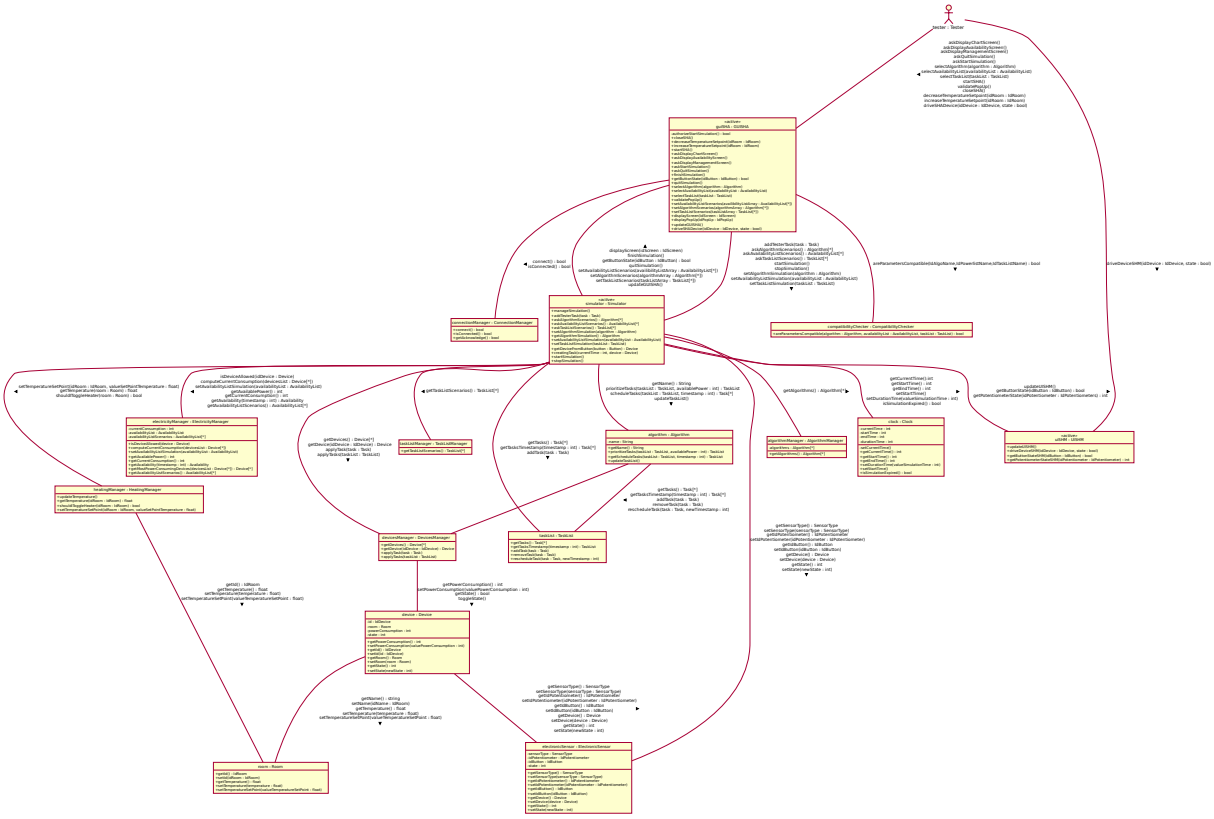

FIGURE 1 – Architecture candidate

Le diagramme de la figure 1 représente l'architecture candidate du système.

2.2 Diagrammes de séquence

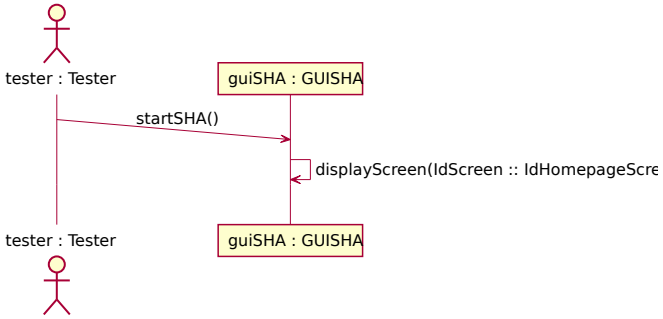

FIGURE 2 – Diagramme de séquence *Démarrer SmartHouseApp*

Le diagramme de la figure 2 représente le diagramme de séquence *Démarrer SmartHouseApp*.

 ©2023 Droits réservés, UKIO

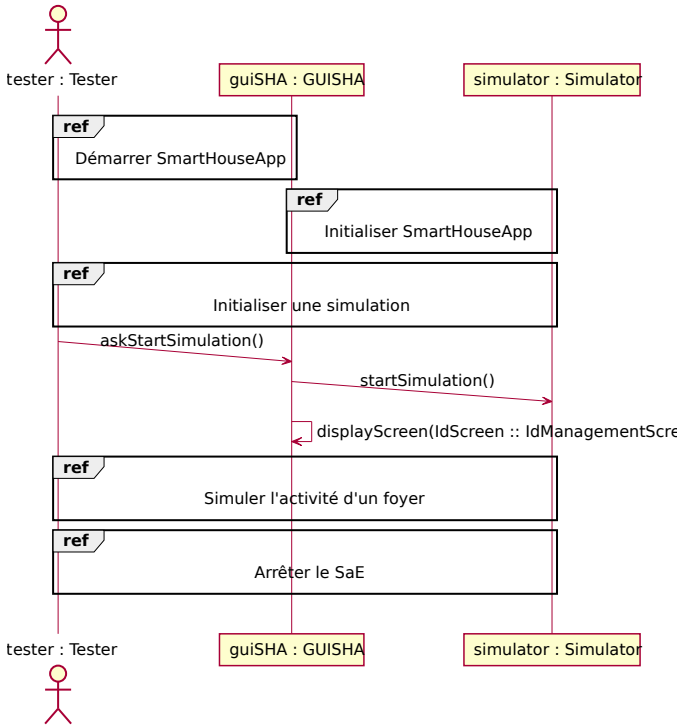

FIGURE 3 – Diagramme de séquence *Assister l'évaluation d'algorithmes d'effacement énergétique d'un foyer [scénario nominal]*

Le diagramme de la figure 3 représente le diagramme de séquence *Assister l'évaluation d'algorithmes d'effacement énergétique d'un foyer [scénario nominal]*.

FIGURE 4 – Diagramme de séquence *Initialiser SmartHouseApp*

Le diagramme de la figure 4 représente le diagramme de séquence *Initialiser SmartHouseApp*.

 ©2023 Droits réservés, UKIO

FIGURE 5 – Diagramme de séquence *Initialiser une simulation*

Le diagramme de la figure 5 représente le diagramme de séquence *Initialiser une simulation*.

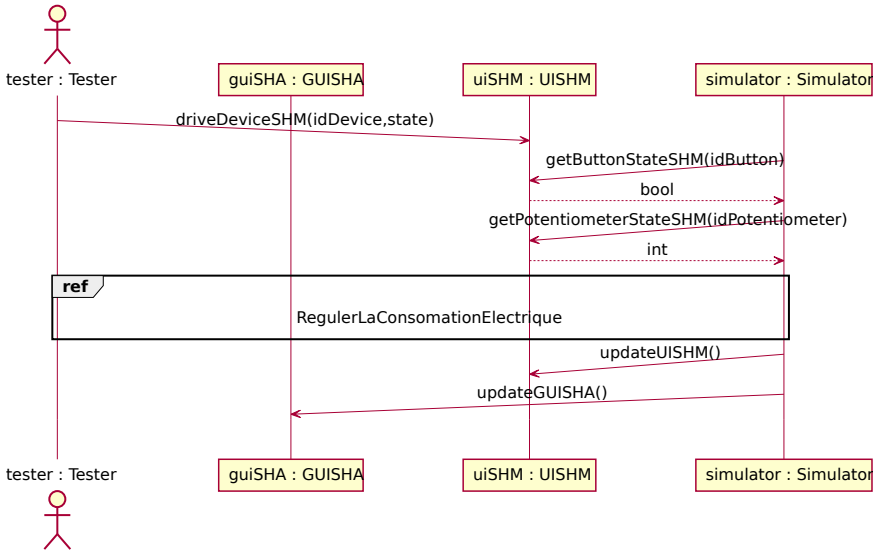

FIGURE 6 – Diagramme de séquence *Simuler l'activité d'un foyer [depuis SmartHouseModel]*

Le diagramme de la figure 6 représente le diagramme de séquence *Simuler l'activité d'un foyer [depuis SmartHouseModel]*.

 ©2023 Droits réservés, UKIO

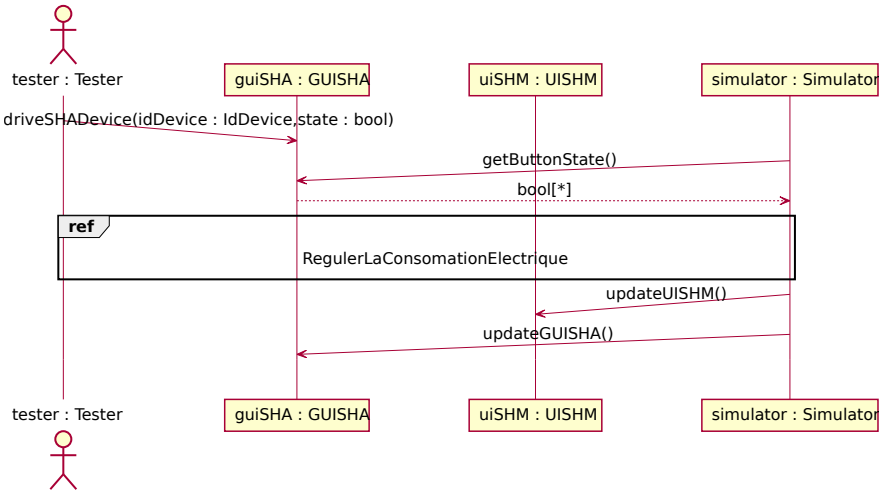

FIGURE 7 – Diagramme de séquence *Simuler l'activité d'un foyer [depuis SmartHouseApp]*

Le diagramme de la figure 7 représente le diagramme de séquence *Simuler l'activité d'un foyer [depuis SmartHouseApp]*.

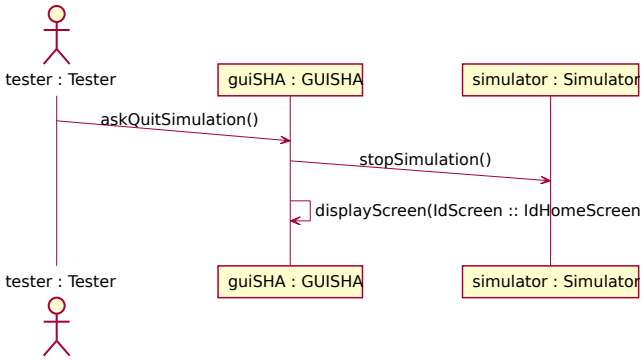

FIGURE 8 – Diagramme de séquence *Quitter une simulation [scenario nominal]*

Le diagramme de la figure 8 représente le diagramme de séquence *Quitter une simulation [scenario nominal]*.

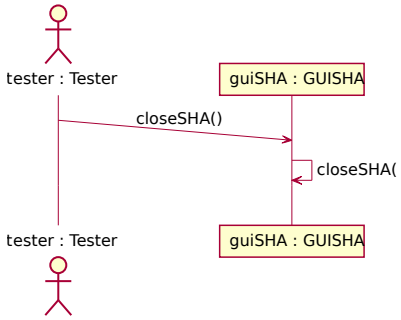

FIGURE 9 – Diagramme de séquence *Arrêter le SaE [scénario nominal]*

Le diagramme de la figure 9 représente le diagramme de séquence *Arrêter le SaE [scénario nominal]*.

 ©2023 Droits réservés, UKIO

2.3 Types de données

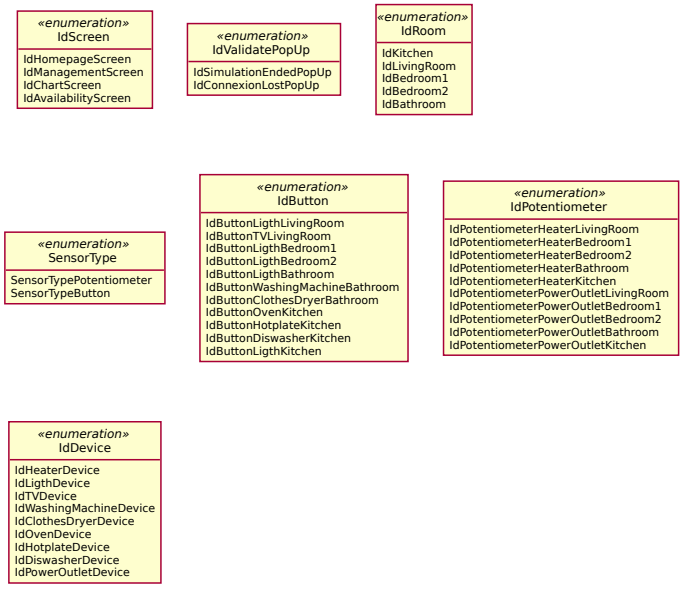

FIGURE 10 – Diagramme des types de données

Le diagramme de la figure 10 représente les types de données utilisés.
L'énumération IdScreen possède les littéraux suivants :

- **IdHomepageScreen** — Identifiant de l'écran d'accueil, ces identifiants sont utilisées pour savoir quel écran afficher
- **IdManagementScreen** — Identidiant de l'écran Management
- **IdChartScreen** — Identidiant de l'écran Graphique
- **IdAvailabilityScreen** — Identidiant de l'écran Disponibilité

L'énumération IdValidatePopUp possède les littéraux suivants :

- **IdSimulationEndedPopUp** — Identifiant du pop-up indiquant la fin de la simulation
- **IdConnexionLostPopUp** — Identifiant du pop-up indiquant que la connexion est perdue

L'énumération IdRoom possède les littéraux suivants :

- **IdKitchen** — Identifiant de la cuisine
- **IdLivingRoom** — Identifiant du salon
- **IdBedroom1** — Identifiant de la chambre 1
- **IdBedroom2** — Identifiant de la chambre 2
- **IdBathroom** — Identifiant de la salle de bain

L'énumération SensorType possède les littéraux suivants :

- **SensorTypePotentiometer** — Type de capteur potentiomètre
- **SensorTypeButton** — Type de capteur bouton

L'énumération IdButton possède les littéraux suivants :

- **IdButtonLigthLivingRoom** — Identifiant du bouton ON/OFF de la lumière du salon
- **IdButtonTVLivingRoom** — Identifiant du bouton ON/OFF de la télévision du salon
- **IdButtonLigthBedroom1** — Identifiant du bouton ON/OFF de la lumière de la chambre 1

 ©2023 Droits réservés, UKIO

- **IdButtonLigthBedroom2** — Identifiant du bouton ON/OFF de la lumière de la chambre 2
- **IdButtonLigthBathroom** — Identifiant du bouton ON/OFF de la lumière de la salle de bain
- **IdButtonWashingMachineBathroom** — Identifiant du bouton ON/OFF de la machine à laver de la salle de bain
- **IdButtonClothesDryerBathroom** — Identifiant du bouton ON/OFF de la sécheuse de la salle de bain
- **IdButtonOvenKitchen** — Identifiant du bouton ON/OFF du four de la cuisine
- **IdButtonHotplateKitchen** — Identifiant du bouton ON/OFF de la plaque de cuisson de la cuisine
- **IdButtonDiswasherKitchen** — Identifiant du bouton ON/OFF du lave-vaisselle de la cuisine
- **IdButtonLigthKitchen** — Identifiant du bouton ON/OFF de la lumière de la cuisine

L'énumération IdPotentiometer possède les littéraux suivants :

- **IdPotentiometerHeaterLivingRoom** — Identifiant du potentiomètre de la température du chauffage du salon
- **IdPotentiometerHeaterBedroom1** — Identifiant du potentiomètre de la température du chauffage de la chambre 1
- **IdPotentiometerHeaterBedroom2** — Identifiant du potentiomètre de la température du chauffage de la chambre 2
- **IdPotentiometerHeaterBathroom** — Identifiant du potentiomètre de la température du chauffage de la salle de bain
- **IdPotentiometerHeaterKitchen** — Identifiant du potentiomètre de la température du chauffage de la cuisine
- **IdPotentiometerPowerOutletLivingRoom** — Identifiant du potentiomètre de la puissance consommée de la prise du salon
- **IdPotentiometerPowerOutletBedroom1** — Identifiant du potentiomètre de la puissance consommée de la prise de la chambre 1
- **IdPotentiometerPowerOutletBedroom2** — Identifiant du potentiomètre de la puissance consommée de la prise de la chambre 2
- **IdPotentiometerPowerOutletBathroom** — Identifiant du potentiomètre de la puissance consommée de la prise de la salle de bain
- **IdPotentiometerPowerOutletKitchen** — Identifiant du potentiomètre de la puissance consommée de la prise de la cuisine

L'énumération IdDevice possède les littéraux suivants :

- **IdHeaterDevice** — Identifiant du chauffage
- **IdLigthDevice** — Identifiant de la lumière
- **IdTVDevice** — Identifiant de la télévision
- **IdWashingMachineDevice** — Identifiant de la machine à laver
- **IdClothesDryerDevice** — Identifiant de la sèche-linge
- **IdOvenDevice** — Identifiant du four
- **IdHotplateDevice** — Identifiant de la plaque de cuisson
- **IdDiswasherDevice** — Identifiant du lave-vaisselle
- **IdPowerOutletDevice** — Identifiant de la prise

 ©2023 Droits réservés, UKIO

2.4 Classes

2.4.1 Vue générale

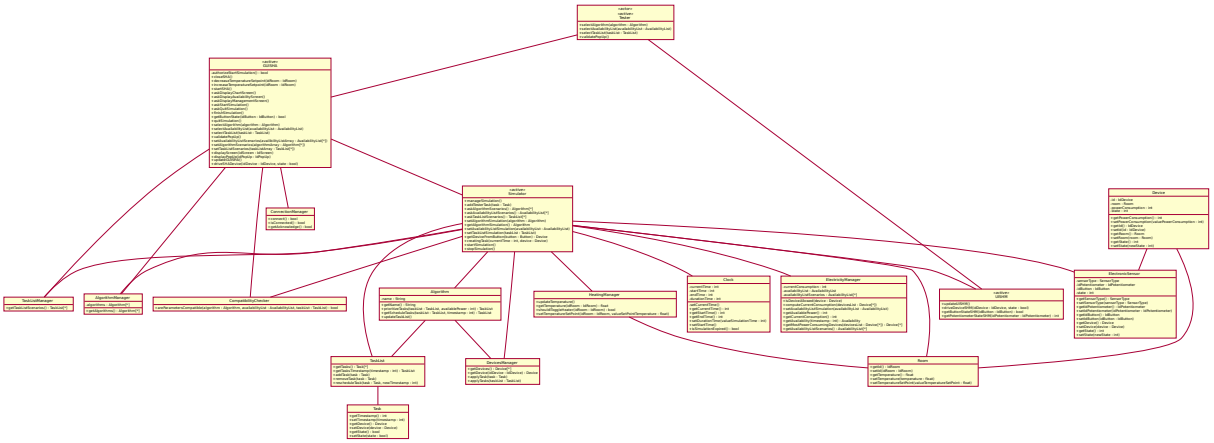

FIGURE 11 – Diagramme de classes

Le diagramme de la figure 11 représente les classes du système.

2.4.2 La classe Algorithm

Le diagramme de la figure 12 représente la classe Algorithm.

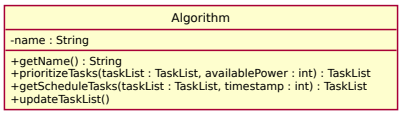

FIGURE 12 – Diagramme de la classe Algorithm

2.4.2.1 Attributs

- **name : String** — Nom de l'algorithme

2.4.2.2 Services offerts

- **getName() : String** — Renvoie le nom de l'algorithme
- **prioritizeTasks(taskList : TaskList, availablePower : int) : TaskList** — Renvoie la liste de tâches triées à effectuer par ordre de priorité
- **getScheduleTasks(taskList : TaskList, timestamp : int) : TaskList** — Renvoie la liste de tâches à faire pour le step courant
- **updateTaskList()** — Met à jour la liste de tâches dans TaskList (à voir pour rajouter des params de fct) TODO : SERT A RIEN ?

2.4.3 La classe AlgorithmManager

Le diagramme de la figure 13 représente la classe AlgorithmManager.

 ©2023 Droits réservés, UKIO

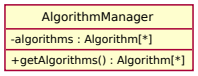

FIGURE 13 – Diagramme de la classe AlgorithmManager

2.4.3.1 Attributs

- **algorithms : Algorithm[*]** — Liste des algorithmes disponibles

2.4.3.2 Services offerts

- **getAlgorithms() : Algorithm[*]** — Renvoie la liste de tous les algorithmes disponibles

2.4.4 La classe Clock

Le diagramme de la figure 14 représente la classe Clock.

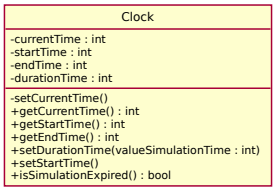

FIGURE 14 – Diagramme de la classe Clock

2.4.4.1 Attributs

- **currentTime : int** — Temps courant de la simulation
- **startTime : int** — Temps au début de la simulation
- **endTime : int** — Temps à la fin de la simulation
- **durationTime : int** — Durée de la simulation

2.4.4.2 Services offerts

- **setCurrentTime()** — Définit le temps courant de la simulation
- **getCurrentTime() : int** — Renvoie le temps courant de la simulation
- **getStartTime() : int** — Renvoie le temps de début de la simulation
- **getEndTime() : int** — Renvoie le temps de fin de la simulation
- **setDurationTime(valueSimulationTime : int)** — Définit la durée de la simulation
- **setStartTime()** — Définit le temps de début de la simulation
- **isSimulationExpired() : bool** — Verifie si on dépassé la dur

2.4.5 La classe CompatibilityChecker

Le diagramme de la figure 15 représente la classe CompatibilityChecker.

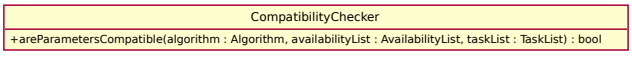

FIGURE 15 – Diagramme de la classe CompatibilityChecker

 ©2023 Droits réservés, UKIO

2.4.5.1 Attributs

N.A.

2.4.5.2 Services offerts

- **areParametersCompatible(algorithm : Algorithm, availabilityList : AvailabilityList, taskList : TaskList) : bool** — Vérifie la compatibilité des paramètres de simulation, c'est à dire qu'ils sont définis sur une même période de temps. Renvoie true s'ils sont compatibles, false sinon.

2.4.6 La classe ConnectionManager

Le diagramme de la figure 16 représente la classe ConnectionManager.

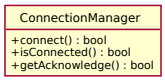

FIGURE 16 – Diagramme de la classe ConnectionManager

2.4.6.1 Attributs

N.A.

2.4.6.2 Services offerts

- **connect() : bool** — Tente de se connecter au serveur. Renvoie true si la connexion est établie, false sinon.
- **isConnected() : bool** — Vérifie l'état courant de la connexion. Renvoie true si la connexion est en cours, false sinon.
- **getAcknowledge() : bool** — Renvoie true si la requête a été traitée, false sinon.

2.4.7 La classe ElectronicSensor

Le diagramme de la figure 17 représente la classe ElectronicSensor.

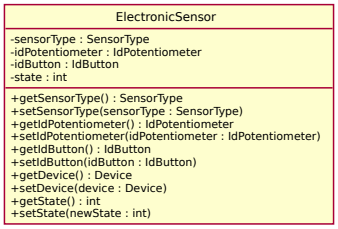

FIGURE 17 – Diagramme de la classe ElectronicSensor

2.4.7.1 Attributs

- **sensorType : SensorType** — Type de capteur
- **idPotentiometer : IdPotentiometer** — Identifiant du potentiomètre du capteur
- **idButton : IdButton** — Identifiant du bouton du capteur
- **state : int** — Etat du capteur

 ©2023 Droits réservés, UKIO

2.4.7.2 Services offerts

- **getSensorType() : SensorType** — Renvoie le type de capteur
- **setSensorType(sensorType : SensorType)** — Définit le type de capteur
- **getIdPotentiometer() : IdPotentiometer** — Renvoie l'identifiant du potentiomètre du capteur
- **setIdPotentiometer(idPotentiometer : IdPotentiometer)** — Définit l'identifiant du potentiomètre du capteur
- **getIdButton() : IdButton** — Renvoie l'identifiant du bouton du capteur
- **setIdButton(idButton : IdButton)** — Définit l'identifiant du bouton du capteur
- **getDevice() : Device** — Renvoie le device auquel est relié le capteur
- **setDevice(device : Device)** — Définit le device auquel est relié le capteur
- **getState() : int** — Renvoie la valeur du device. Dans le cas d'un interrupteur, renvoie 0 si l'interrupteur est éteint, 1 si il est allumé. Dans le cas d'un potentiomètre, renvoie la valeur du potentiomètre.
- **setState(newState : int)** — Change l'état du device. Dans le cas d'un interrupteur, newState doit être 0 ou 1. Dans le cas d'un potentiomètre, newState doit être compris entre 0 et X.

2.4.8 La classe Device

Le diagramme de la figure 18 représente la classe Device.

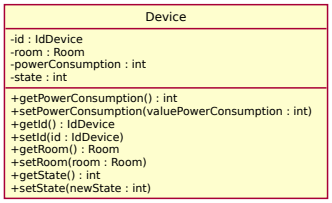

FIGURE 18 – Diagramme de la classe Device

2.4.8.1 Attributs

- **id : IdDevice** — Identifiant du device
- **room : Room** — Pièce dans laquelle se trouve le device
- **powerConsumption : int** — Consommation du device en Watt
- **state : int** — Etat du device

2.4.8.2 Services offerts

- **getPowerConsumption() : int** — Appelle getState() et renvoie 0 ou bien la consommation du device en Watt.
- **setPowerConsumption(valuePowerConsumption : int)** — Fixe la consommation de la prise électrique en Watt.s
- **getId() : IdDevice** — Renvoie l'identifiant du device.
- **setId(id : IdDevice)** — Définit l'identifiant du device.

 ©2023 Droits réservés, UKIO

- **getRoom() : Room** — Renvoie la pièce dans laquelle se trouve le device.
- **setRoom(room : Room)** — Définit la pièce dans laquelle se trouve le device.
- **getState() : int** — Renvoie l'état du device. 0 si le device est éteint, 1 si il est allumé.
- **setState(newState : int)** — Change l'état du device. newState doit être 0 ou 1.

2.4.9 La classe ElectricityManager

Le diagramme de la figure 19 représente la classe ElectricityManager.

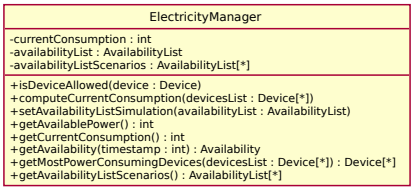

FIGURE 19 – Diagramme de la classe ElectricityManager

2.4.9.1 Attributs

- **currentConsumption : int** — Consommation actuelle totale des devices
- **availabilityList : AvailabilityList** — Liste des disponibilités électriques
- **availabilityListScenarios : AvailabilityList[*]** — Liste des disponibilités électriques possibles pour une simulation

2.4.9.2 Services offerts

- **isDeviceAllowed(device : Device)** — Renvoie si oui ou non le device peut être allumé. Doit appeler getAvailablePower().
- **computeCurrentConsumption(devicesList : Device[*])** — Calcule la consommation totale des devices. Renvoie la consommation totale.
- **setAvailabilityListSimulation(availabilityList : AvailabilityList)** — Définit la liste de disponibilité électrique à utiliser pour la simulation.
- **getAvailablePower() : int** — Renvoie la puissance électrique dipsonible en fonction de celle consommée.
- **getCurrentConsumption() : int** — Renvoie la consommation actuelle totale des devices. Il faut appeller computeCurrentConsumption(devicesList : Device[*]) avant pour avoir la valeur courante.
- **getAvailability(timestamp : int) : Availability** — Renvoie la disponibilité électrique à un instant donné.
- **getMostPowerConsumingDevices(devicesList : Device[*]) : Device[*]** — Renvoie la liste triées des devices les plus consommateurs d'énergie
- **getAvailabilityListScenarios() : AvailabilityList[*]** — Renvoie les listes des disponibilités électriques possibles pour une simulation.

 ©2023 Droits réservés, UKIO

2.4.10 La classe GUISHA

Le diagramme de la figure 20 représente la classe GUISHA.

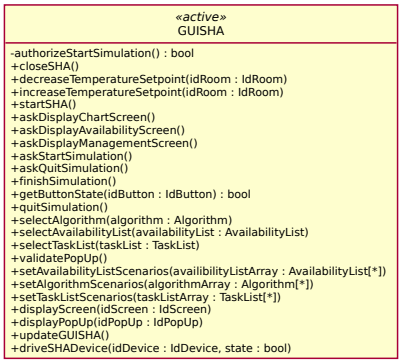

FIGURE 20 – Diagramme de la classe GUISHA

2.4.10.1 Attributs

N.A.

2.4.10.2 Services offerts

- **authorizeStartSimulation() : bool** — Autorise ou non le lancement de la simulation en fonction des paramètres sélectionnés. Renvoie true si la simulation peut être lancée, false sinon.
- **closeSHA()** — Ferme l'application SmartHouseApp
- **decreaseTemperatureSetpoint(idRoom : IdRoom)** — Décremente d'un pas la consigne de la pièce passée en paramètre
- **increaseTemperatureSetpoint(idRoom : IdRoom)** — Incrémente d'un pas la consigne de la pièce passée en paramètre
- **startSHA()** — Initialise l'application SmartHouseApp
- **askDisplayChartScreen()** — Evènement déclenché lors du clic sur le bouton "Graphique" en haut de l'écran
- **askDisplayAvailabilityScreen()** — Evènement déclenché lors du clic sur le bouton "Disponibilité" en haut de l'écran
- **askDisplayManagementScreen()** — Evènement déclenché lors du clic sur le bouton "Management" en haut de l'écran
- **askStartSimulation()** — Testeur demande à démarrer la simulation
- **askQuitSimulation()** — Testeur demande à quitter la simulation
- **finishSimulation()** — Termine la simulation
- **getButtonState(idButton : IdButton) : bool** — Renvoie l'état du bouton de l'appareil passé en paramètre
- **quitSimulation()** — Quitte la simulation et retour à l'écran d'accueil
- **selectAlgorithm(algorithm : Algorithm)** — Sélectionne un algorithme parmi la liste d'algorithmes
- **selectAvailabilityList(availabilityList : AvailabilityList)** — Sélectionne une availabilityList parmi toutes les availabilityList

 ©2023 Droits réservés, UKIO

- **selectTaskList(taskList : TaskList)** — Sélectionne une liste de tâches parmi la liste de liste de tâches
- **validatePopUp()** — Ferme la pop-up affichée à l'écran
- **setAvailabilityListScenarios(availibilityListArray : AvailabilityList[*])** — Définit la sélection de toutes les liste de disponibilité électrique qu'un utilisateur peut choisir pour une simulation.
- **setAlgorithmScenarios(algorithmArray : Algorithm[*])** — Définit la sélection de toutes les listes d'algorithme qu'un utilisateur peut choisir pour une simulation.
- **setTaskListScenarios(taskListArray : TaskList[*])** — Définit la sélection de toutes les listes de tâches qu'un utilisateur peut choisir pour une simulation.
- **displayScreen(idScreen : IdScreen)** — affiche l'écran donné par l'id.
- **displayPopUp(idPopUp : IdPopUp)** — affiche la popup donnée par l'id.
- **updateGUISHA()** — Met à jour les informations affiché par GUISHA.
- **driveSHADevice(idDevice : IdDevice, state : bool)** — Demande à allumer ou éteindre le device passé en paramètre.

2.4.10.3 Description comportementale

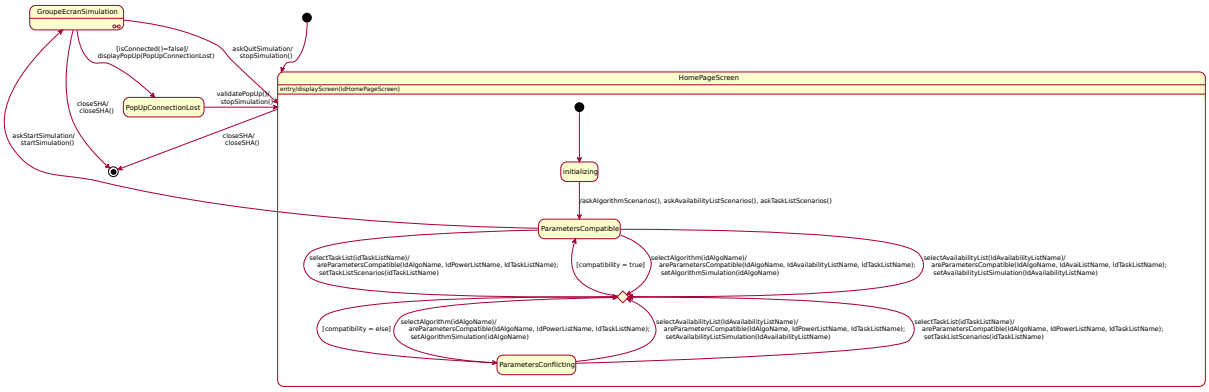

FIGURE 21 – Machine à états de *GUISHA*

Le diagramme de la figure 21 représente la machine à états de *GUISHA*.

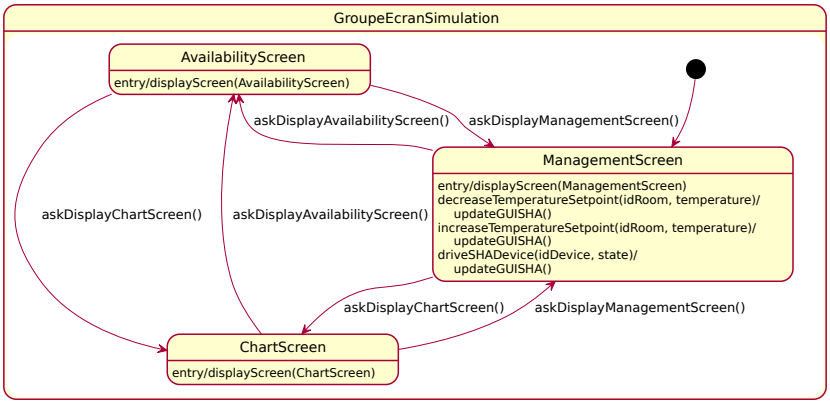

FIGURE 22 – Sous-état *GroupeEcranSimulation* de *GUISHA*

 ©2023 Droits réservés, UKIO

Le diagramme de la figure 22 représente le sous-état *GroupeEcranSimulation* de la machine à états de *GUISHA*.

2.4.11 La classe Room

Le diagramme de la figure 23 représente la classe Room.

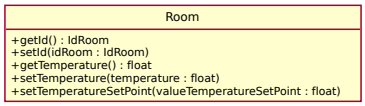

FIGURE 23 – Diagramme de la classe Room

2.4.11.1 Attributs

N.A.

2.4.11.2 Services offerts

- **getId() : IdRoom** — Renvoie le nom de la pièce.
- **setId(idRoom : IdRoom)** — Définit le nom de la pièce.
- **getTemperature() : float** — Renvoie la température de la pièce.
- **setTemperature(temperature : float)** — Définit la température de la pièce.
- **setTemperatureSetPoint(valueTemperatureSetPoint : float)** — Ajoute à un radiateur passé en paramètre une consigne de température

2.4.12 La classe HeatingManager

Le diagramme de la figure 24 représente la classe HeatingManager.

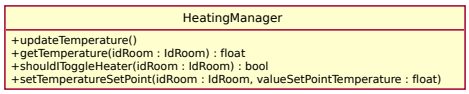

FIGURE 24 – Diagramme de la classe HeatingManager

2.4.12.1 Attributs

N.A.

2.4.12.2 Services offerts

- **updateTemperature()** — Met à jour la température de tous les radiateurs
- **getTemperature(idRoom : IdRoom) : float** — Renvoie la température de la pièce passée en paramètre
- **shouldIToggleHeater(idRoom : IdRoom) : bool** — Vérifie si le radiater passé en paramètre doit être arreté
- **setTemperatureSetPoint(idRoom : IdRoom, valueSetPointTemperature : float)** — Ajoute à un radiateur passé en paramètre une consigne de température

 ©2023 Droits réservés, UKIO

2.4.13 La classe DevicesManager

Le diagramme de la figure 25 représente la classe DevicesManager.

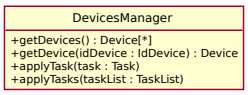

FIGURE 25 – Diagramme de la classe DevicesManager

2.4.13.1 Attributs

N.A.

2.4.13.2 Services offerts

- **getDevices() : Device[*]** — Renvoie la liste des devices
- **getDevice(idDevice : IdDevice) : Device** — Renvoie le device via l'id passé en paramètre
- **applyTask(task : Task)** — Applique la tâche passée en paramètre
- **applyTasks(taskList : TaskList)** — Applique les tâches passées en paramètres

2.4.14 La classe Simulator

Le diagramme de la figure 26 représente la classe Simulator.

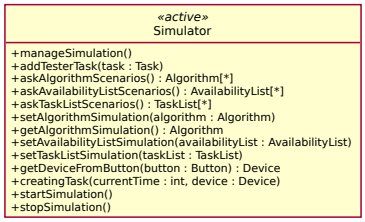

FIGURE 26 – Diagramme de la classe Simulator

2.4.14.1 Attributs

N.A.

2.4.14.2 Services offerts

- **manageSimulation()** — TODO ? Change l'état d'un device lors d'une simulation.
- **addTesterTask(task : Task)** — Ajoute à la liste des tâches à effectuer une tâche de l'utilisateur
- **askAlgorithmScenarios() : Algorithm[*]** — Demande la liste des algorithmes implémentés
- **askAvailabilityListScenarios() : AvailabilityList[*]** — Demande la liste des listes de puissance disponible implémentées
- **askTaskListScenarios() : TaskList[*]** — Demande la liste des listes de tâches implémentées

 ©2023 Droits réservés, UKIO

- **setAlgorithmSimulation(algorithm : Algorithm)** — Fixe l'algorithme à utiliser pour la simulation
- **getAlgorithmSimulation() : Algorithm** — Récupère l'algorithme à utiliser pour la simulation
- **setAvailabilityListSimulation(availabilityList : AvailabilityList)** — Fixe la liste de puissance disponible à utiliser pour la simulation
- **setTaskListSimulation(taskList : TaskList)** — Fixe la liste de tâches à utiliser pour la simulation
- **getDeviceFromButton(button : Button) : Device** — Renvoie un device en fonction du bouton passé en paramètre. Permet de faire le lien entre un device et un bouton
- **creatingTask(currentTime : int, device : Device)** — TODO
- **startSimulation()** — Démarre la simulation
- **stopSimulation()** — Arrête la simulation

2.4.14.3 Description comportementale

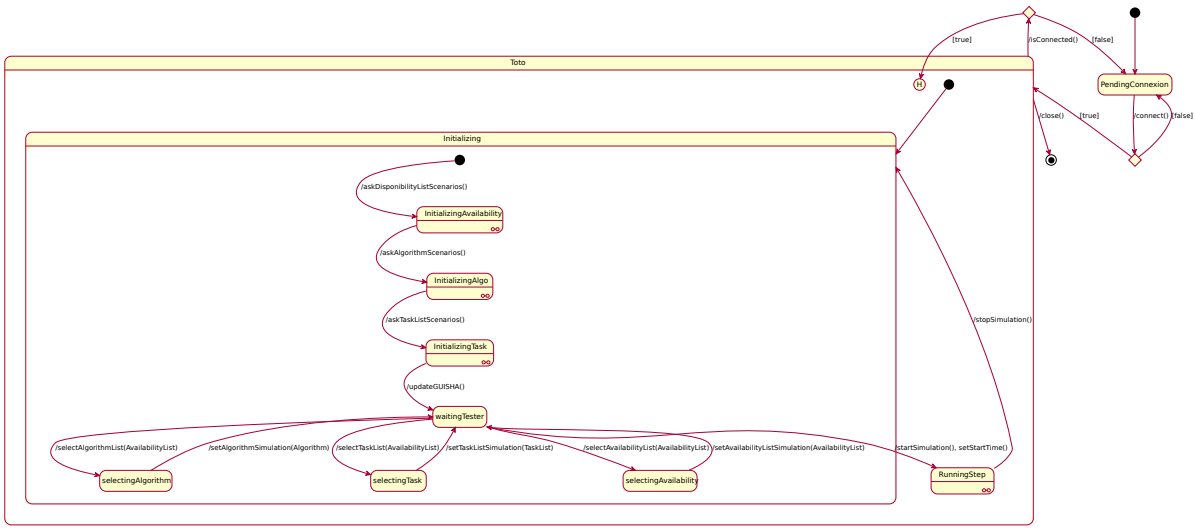

FIGURE 27 – Machine à états de *Simulator*

Le diagramme de la figure 27 représente la machine à états de *Simulator*.

 ©2023 Droits réservés, UKIO

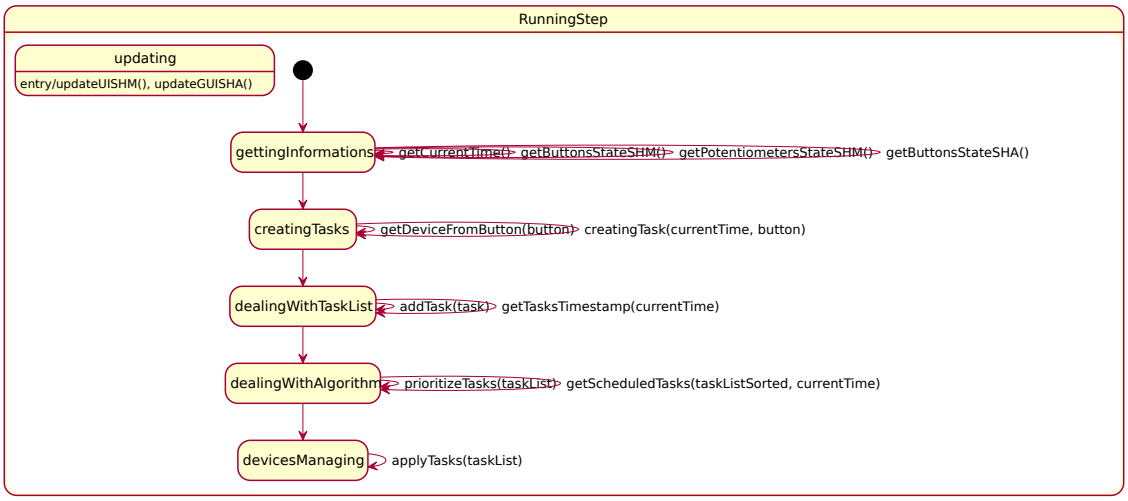

FIGURE 28 – Sous-état *Toto.RunningStep* de *Simulator*

Le diagramme de la figure 28 représente le sous-état *Toto.RunningStep* de la machine à états de *Simulator*.

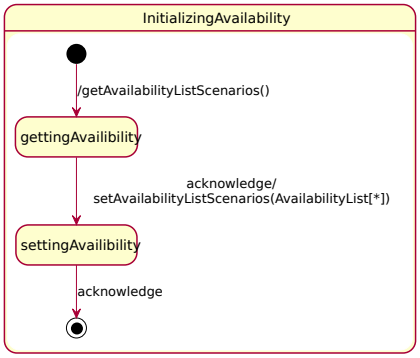

FIGURE 29 – Sous-état *Toto.Initializing.InitializingAvailability* de *Simulator*

Le diagramme de la figure 29 représente le sous-état *Toto.Initializing.InitializingAvailability* de la machine à états de *Simulator*.

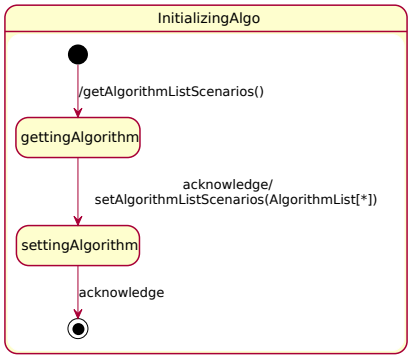

FIGURE 30 – Sous-état *Toto.Initializing.InitializingAlgo* de *Simulator*

Le diagramme de la figure 30 représente le sous-état *Toto.Initializing.InitializingAlgo* de la machine à états de *Simulator*.

 ©2023 Droits réservés, UKIO

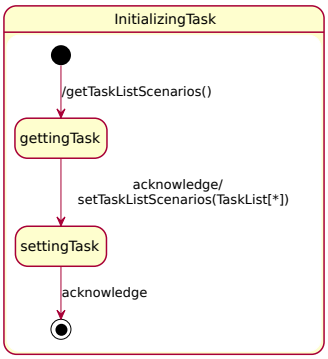

FIGURE 31 – Sous-état *Toto.Initializing.InitializingTask* de *Simulator*

Le diagramme de la figure 31 représente le sous-état *Toto.Initializing.InitializingTask* de la machine à états de *Simulator*.

2.4.15 La classe Task

Le diagramme de la figure 32 représente la classe Task.

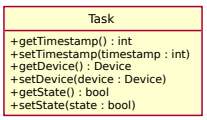

FIGURE 32 – Diagramme de la classe Task

2.4.15.1 Attributs

N.A.

2.4.15.2 Services offerts

- **getTimestamp() : int** — Récupère l'étape de simulation pour laquelle la tâche doit être effectuée
- **setTimestamp(timestamp : int)** — Fixe l'étape de simulation pour laquelle la tâche doit être effectuée
- **getDevice() : Device** — Récupère le device sur lequel la tâche doit être effectuée
- **setDevice(device : Device)** — Fixe le device sur lequel la tâche doit être effectuée
- **getState() : bool** — Récupère l'état du device sur lequel la tâche doit être effectuée
- **setState(state : bool)** — Fixe l'état du device sur lequel la tâche doit être effectuée

2.4.16 La classe TaskList

Le diagramme de la figure 33 représente la classe TaskList.

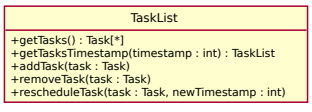

FIGURE 33 – Diagramme de la classe TaskList

 ©2023 Droits réservés, UKIO

2.4.16.1 Attributs

N.A.

2.4.16.2 Services offerts

- **getTasks() : Task[*]** — Renvoie la liste de tâches planifiées
- **getTasksTimestamp(timestamp : int) : TaskList** — Renvoie la liste de tâches planifiées pour l'étape de simulation donnée en paramètre
- **addTask(task : Task)** — Ajoute une tâche à la liste de tâches planifiées
- **removeTask(task : Task)** — Supprime une tâche de la liste de tâches planifiées
- **rescheduleTask(task : Task, newTimestamp : int)** — Modifie l'étape de simulation pour laquelle la tâche doit être effectuée

2.4.17 La classe TaskListManager

Le diagramme de la figure 34 représente la classe TaskListManager.

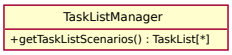

FIGURE 34 – Diagramme de la classe TaskListManager

2.4.17.1 Attributs

N.A.

2.4.17.2 Services offerts

- **getTaskListScenarios() : TaskList[*]** — Récupère la liste de tâches planifiées implémentées.

2.4.18 La classe Tester

Le diagramme de la figure 35 représente la classe Tester.

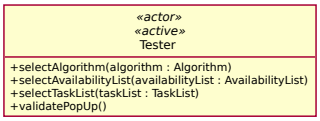

FIGURE 35 – Diagramme de la classe Tester

2.4.18.1 Attributs

N.A.

2.4.18.2 Services offerts

- **selectAlgorithm(algorithm : Algorithm)** — Permet à testeur de sélectionner l'algorithme à utiliser pour la simulation

 ©2023 Droits réservés, UKIO

- **selectAvailabilityList(availabilityList : AvailabilityList)** — Permet à testeur de sélectionner la liste de puissance disponible à utiliser pour la simulation
- **selectTaskList(taskList : TaskList)** — Permet à testeur de sélectionner la liste de tâches à utiliser pour la simulation
- **validatePopUp()** — Permet à testeur de fermer la pop-up affichée à l'écran

2.4.19 La classe UISHM

Le diagramme de la figure 36 représente la classe UISHM.

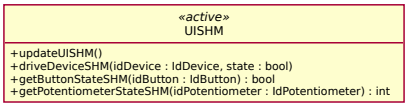

FIGURE 36 – Diagramme de la classe UISHM

2.4.19.1 Attributs

N.A.

2.4.19.2 Services offerts

- **updateUISHM()** — Met à jour l'interface utilisateur
- **driveDeviceSHM(idDevice : IdDevice, state : bool)** — Action de Testeur : Pilote l'appareil passé en paramètre avec l'état passé en paramètre
- **getButtonStateSHM(idButton : IdButton) : bool** — Permet à simulator de récupérer l'état d'un bouton ON/OFF (interrupteur) de SHM
- **getPotentiometerStateSHM(idPotentiometer : IdPotentiometer) : int** — Permet à simulator de récupérer l'état d'un potentiomètre de SHM

2.4.19.3 Description comportementale

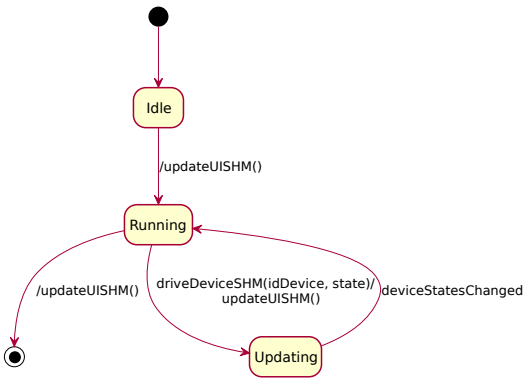

FIGURE 37 – Machine à états de *UISHM*

Le diagramme de la figure 37 représente la machine à états de *UISHM*.

 ©2023 Droits réservés, UKIO

 ©2023 Droits réservés, UKIO

4 Dictionnaire du domaine

Les abréviations utilisées dans le présent document sont répertoriées et expliquées dans le tableau ci-dessous. Les termes utiles pour interpréter correctement ce document sont définis dans le dictionnaire du domaine section 3.3.

A

Algorithme
: Code qui régule les pics de consommation du foyer.

Appareil
: Ensemble des appareils pouvant être pilotés suite à une action de Testeur modifiant ainsi son état et sa représentation physique et virtuelle sur SmartHouseModel et/ou SmartHouseApp.

E

Entreprise cliente
: Entité requérant les services du projet, susceptible d'avoir des consignes spécifiques quant à l'apparence globale du SaE et la charte graphique de ses IHMs.

O

OpenSTLinux
: OpenSTLinux est une distribution Linux inspirée de OpenEmbedded qui est un environnement de compilation utilisé pour créer des distributions Linux pour les appareils embarqués.

S

Simulation
: Processus principal du SaE, représentation de la consommation et production électrique possible d'une maison sur une durée équivalente à DureeSimulation * NBJour.

SmartHouseApp
: Application Android sur la tablette.

SmartHouseControl
: Carte électronique de développement STM32MP157C-DK2 avec la distribution Linux OpenSTLinux 5.15.

SmartHouseModel
: Maquette du foyer.

SmartHouseSim
: Nom du projet.

SmartHouseSoft
: Logiciel embarqué dans SmartHouseControl.

SmartHouseTablet
: Tablette Asus Iconia Tab 10 A3-A40 avec Android 6.0.

T

Testeur
: Utilisateur de SmartHouseSim. Celui-ci a la possibilité d'interagir et de paramétrer SmartHouseSim.

 ©2023 Droits réservés, UKIO

5 Table des figures

 ©2023 Droits réservés, UKIO