REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
**Département d'informatique**

**N° d'ordre : RTIC19/M2/2021**

## Mémoire

Présenté pour obtenir le diplôme de master académique en

# Informatique

Parcours : **Réseaux et Technologies de l'Information et de la Communication(RTIC)**

---

# Une approche auto-guérison pour le suivi médicale a distance dans l'internet des objets

---

**Par :**

**GSOURI BAHIDJA HABIBA**

Soutenu le 01/07/2021 devant le jury composé de :

| | | |
|---|---|---|
| Nom Prénom | Grade | Président |
| Sahli Sihem | Grade | Rapporteur |
| Nom Prénom | Grade | Examinateur |

Année universitaire 2020-2021

# *Remerciement*

*Je tiens tout d'abord à remercier Dieu tout puissant et miséricordieux, de m'avoir donné la volonté et la force pour terminer ce projet de fin d'études.*

*Je tiens aussi à remercier mon encadreur* ***Mm. SAHLI SIHEM*** *pour ses précieux conseils, encouragements, sa disponibilité, et son aide tout au long de la réalisation de ce travail.*

*Mes remerciements aux membres de jury pour l'honneur qu'ils nous ont fait en acceptant d'examiner ce travail.*

*Mes remerciements à* ***mes très chers parents*** *et* ***mon marie*** *de m'avoir toujours donné le courage et d'être l'une des raisons qui me pousse à me battre face à chaque obstacle. Mes remerciements à mes enseignants qui ont contribué à notre formation et au personnel de notre département.*

*Mes remerciements à tous ceux qui m'ont aidé, de près ou de loin, à réaliser ce travail.*

# *Dédicaces*

*Je dédie ce travail à :*

*Mes très* ***chers parents*** *qui m'ont soutenu pendant mon parcours d'études.*

*Mon Marie* ***Hasni Taqi Eddine*** *et mon bébé* ***Mohamed Iyad.***

*Mes sœurs* ***Manel, Mouna.***

*Mon frère* ***Seif Eddine.***

*Ma famille et mes amies pour leurs soutiens et encouragements.*

*Mes enseignants, particulièrement mon encadreur* ***Mm. Sahli Sihem.***

*Tout combattant de Cancer qui garde toujours la foi en Dieu. Et à toute personne qui ne cesse de se battre pour réaliser ses buts.*

*Gsouri Bahidja Habiba.*

# Résume

Ces dernières années, l'Internet des objets s'est largement incrusté dans le domaine de la santé, plus particulièrement dans la surveillance médicale à distance. La sûreté de fonctionnement de ce système est une nécessité. Notre objectif dans ce mémoire est de contribuer dans ce domaine et spécialement dans la tolérance aux pannes avec une approche d'exécution auto-guérison dans l'internet des objets dans le domaine médicale. L'approche est basée sur des agents qui sont capables de prendre de manière autonome des décisions pendant l'exécution à partir de leurs connaissances. Finalement, nous validons cette approche par une évaluation expérimentale en utilisant une étude de cas.

**Mots clés :** Internet des objets (IdO), auto-guérison**,** surveillance médicale, système multi-agents.

# Abstract

In recent years, the Internet of Things has become increasingly entrenched in healthcare, particularly in remote medical monitoring. The operational reliability of this system is a necessity.

Our aim in this dissertation is to contribute in this area and especially in fault tolerance with a self-healing execution approach in the Internet of Things in the medical field.

The approach is based on agents who are able to autonomously make decisions during execution based on their knowledge.

Finally, we validate this approach by an experimental evaluation using a case study.

**Keywords:**

Internet of Things (IoT), self-healing, medical surveillance, multi-agent system.

# ملخص

في السنوات الأخيرة ، أصبحت إنترنت الأشياء مترسخة بشكل متزايد في الرعاية الصحية ، لا سيما في المراقبة الطبية عن بعد الموثوقية التشغيلية لهذا النظام ضرورة .

هدفنا في هذه الرسالة هو المساهمة في هذا المجال وخاصة في التسامح مع الخطأ من خلال نهج تنفيذ الشفاء الذاتي في إنترن

الأشياء في المجال الطبي

يعتمد النهج على الوكلاء القادرين على اتخاذ القرارات بشكل مستقل أثناء التنفيذ بناءً على معرفتهم.

أخيرًا ، نتحقق من صحة هذا النهج من خلال تقييم تجريبي باستخدام دراسة حالة .

**الكلمات المفتاحية:**

الشفاء الذاتي ، المراقبة الطبية ، النظام متعدد العوامل (IoT) إنترنت الأشياء.

# List des figures

# List des figures

# Liste des tableaux

# Sommaire



## CHAPITRE I : Internet des Objets



## Chapitre II : La Tolérance aux pannes dans L'internet des objets (IdO)

## CHAPITRE III : Conceptions



## Chapitre IV : L'Implémentation De La Proposition

# Introduction générale

L'Internet des objets (IdO) est une nouvelle technologie qui offre au système informatique, un système non centré utilisateur seulement, mais prendre en compte un objet et leur gestion. Cette nouvelle discipline ouvre plusieurs axes de recherche telle que l'architecture d'IdO, la modélisation des objets. Le développement croissant dans le domaine de l'informatique a encouragé l'intégration d'une variété de dispositifs sophistiqués dans les maisons. Ces dispositifs communiquent entre eux pour aider les utilisateurs dans des situations particulières et selon leurs besoins comme dans la sécurité, le confort, et même la santé. Les dispositifs forment un environnement de connexion d'objets, cet environnement est mis en œuvre par internet des objets (IdO). Ces systèmes sont de plus en plus utilisés surtout dans le cas des personnes à autonomie réduite. Dans le domaine médical, l'IdO peut faciliter la collection des informations en temps réel en utilisant des capteurs, pour identifier l'état dans lequel se trouve le patient surveillé telles que sa position, ses signes vitaux, et toutes autres informations pertinentes. Ces systèmes ont une forte contrainte de sûreté car une faute propagée dans le système engendre des conséquences catastrophiques sur la vie humaine.

Notre objectif est l'exploitation de la corrélation entre les signes physiologiques pour créer des alternatives de réponse et tolérer l'existence des fautes. Cette tolérance assure par conséquence une disponibilité continue des fonctionnalités requises par le système. Nous allonsproposer un cadre pour l'exécutiontolérante aux pannes d'un système médical dans l'internet des objets et qui est basé sur dessystèmes multi agents.

Notre mémoire est organisé en 4 chapitres :

- **Chapitre 1:** L'internet des Objets (définition, composants, domaine d'application...)
- **Chapitre 2:** La Tolérance aux pannes des systèmes dans l'internet des objets (IdO) (définition, La Sûreté de fonctionnement, types de pannes….)
- **Chapitre 3:** Conception d'un Système (objectif et problématique, architecture, diagramme de class …)
- **Chapitre 4:** Implémentation

# Chapitre I

# L’internet Des Objets (IdO)

## Introduction

Ces quelques dernières années, l'IdO est devenu l'une des technologies les plus importantes du 21ème siècle. Maintenant que nous pouvons connecter des objets du quotidien (appareils électroménagers, voitures, thermostats, interphones bébés) à Internet par l'intermédiaire de terminaux intégrés, des communications sont possibles en toute transparence entre les personnes, les processus et les objets. L'objectif de ce chapitre est de présenter un aperçu général sur cette nouvelle ère.

## 1. Définition

Il existe plusieurs définitions, nous citons quelque unes :

L'Internet des Objets est « *un réseau de réseaux qui permet, via des systèmes d'identification électronique normalisés et unifiés, et des dispositifs mobiles sans fil, d'identifier directement et sans ambiguïté des entités numériques et des objets physiques et ainsi de pouvoir récupérer, stocker, transférer et traiter, sans discontinuité entre les mondes physiques et virtuels, les données s'y rattachant* » [2].

D'après l'Union Internationale des Télécommunications « *L'Internet des Objets (IdO) est une infrastructure mondiale pour la société de l'information, qui permet de disposer de services évolués en interconnectant des objets (physiques ou virtuels) grâce aux technologies de l'information et de la communication interopérables existantes ou en évolution* » [3].

## 2. Les Composants De L'IdO

L'IdO qui permet la connexion de nos appareils intelligents et des objets au réseau pour fonctionner efficacement et à distance. Ce point répond à la question : quelles sont les principales composantes de l'Internet des objets ?

Le tableau suivant présente les composants principaux de l'IdO: **[3]**

| **Composants IoT** | **La description** |
|---|---|
| **Objets physiques** | Un objet connecté est un objet physique équipé de capteurs ou d'une puce qui lui permettent de transcender son usage initial pour proposer de nouveaux services. Il s'agit d'un matériel électronique capable de communiquer avec un ordinateur, un smartphone ou une tablette via un réseau sans fil (Wi-Fi, Bluetooth, réseaux de téléphonie mobile, réseau radio à longue portée de type Sigfox ou LoRa, etc.), qui le relie à Internet ou à un réseau local. |
| **Capteurs** | Ils sont installés sur les es objets connectés, ils sont plus ou moins intelligents, selon qu'ils intègrent ou non eux-mêmes des algorithmes d'analyse de données, et qu'ils soient pour certains auto-adaptatifs. Les capteurs connus sont : Capteurs de température et thermostats, Capteurs de pression, Humidité / niveau d'humidité, Détecteurs d'intensité lumineuse, Capteurs d'humidité, Détection de proximité, Étiquettes RFID…. |
| **Gens** | Exemple : Les humains peuvent contrôler l'environnement via des applications mobiles |
| **Prestations de service** | Exemple : Services Cloud - peuvent être utilisés pour:<br>• Traiter les Big Data et les transformer en informations précieuses |

| | |
|---|---|
| | • Construire et exécuter des applications innovantes<br>• Optimiser les processus métier en intégrant les données de l'appareil. |
| **Plateformes** | Elle est considérée comme un type d'intergiciel utilisé pour connecter les composants IoT (objets, personnes, services, etc.) à l'environnement l'IoT. Elle fournit de nombreuses fonctions : • Accès aux appareils • Assurer une installation / un comportement correct de l'appareil • Analyse des données • Connexion interopérable avec le réseau local, le cloud ou d'autres périphériques. |
| **Réseaux** | Les composants IdO sont liés entre eux par des réseaux, utilisant diverses technologies, normes et protocoles sans fil et filaire. |

Tableau 1.1 : Les composants de base d'un système IdO

## 3. Architecture de l'IdO

• L'architecture d'une solution IoT varie d'un système à l'autre en se basant sur le type de la solution à mettre en place.

• L'architecture la plus élémentaire est une architecture à trois couches:

– **La couche perception** possède des capteurs et actionneurs qui détectent et recueillent des informations sur l'environnement.

– **La couche réseau** est responsable de la connexion, du transport et du traitement des données issues capteurs et actionneur

– **La couche application** est chargée de fournir à l'utilisateur des services spécifiques et applications intelligentes.

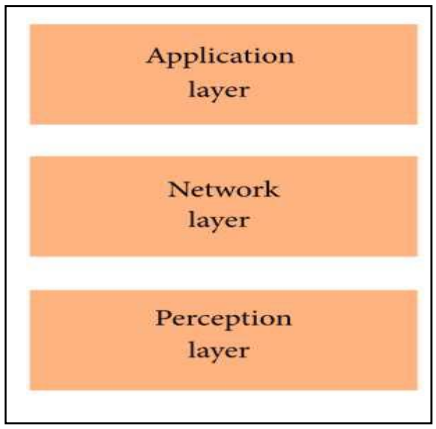


Figure 1.1 : l'architecture de l'IdO [4]

## 4. les technologies utilisées dans le fonctionnement de l'IdO

Il existe plusieurs technologies utilisées dans le fonctionnement de l'IdO, nous mettons l'accent seulement sur quelques-unes qui sont, selon Han et Zhanghang, les technologies clés de l'IdO. Ces technologies sont les suivantes : RFID, WSN et M2M, et sont définies ci-dessous.

- **RFID** (Radio Frequency Identification) : le terme RFID englobe toutes les technologies qui utilisent les ondes radio pour identifier automatiquement des objets ou des personnes. C'est une technologie qui permet de mémoriser et de récupérer des informations à distance grâce à une étiquette qui émet des ondes radio [18]. Il s'agit d'une méthode utilisée pour transférer les données des étiquettes à des objets, ou pour identifier les objets à distance. L'étiquette contient des informations stockées électroniquement pouvant être lues à distance.

- **WSN** (Wireless Sensor Network) : c'est un ensemble de nœuds qui communiquent sans fil et qui sont organisés en un réseau coopératif. Chaque nœud possède une capacité de traitement et peut contenir différents types de mémoires, un émetteur-récepteur RF et une source d'alimentation, comme il peut aussi tenir compte des divers capteurs et des actionneurs [19]. Comme son nom l'indique, le WSN constitue alors un réseau de capteurs sans fil qui peut être une technologie nécessaire au fonctionnement de l'IdO.
- **M2M** (Machine to Machine) : c'est « l'association des technologies de l'information et de la communication avec des objets intelligents dans le but de donner à ces derniers les moyens d'interagir sans intervention humaine avec le système d'information d'une organisation ou d'une entreprise » [20].

## 5. domaines d'application d'IdO

Plusieurs domaines d'application sont touchés par l'IdO. Dans leur article, Gubbi *et al*. [21] ont classé les applications en quatre domaines : 1) le domaine personnel, 2) le domaine du transport, 3) l'environnement et 4) l'infrastructure et les services publics. Comme le schéma ci-dessous le montre, on trouve alors l'IdO dans notre vie personnelle quotidienne et également dans les services publics offerts par le gouvernement.

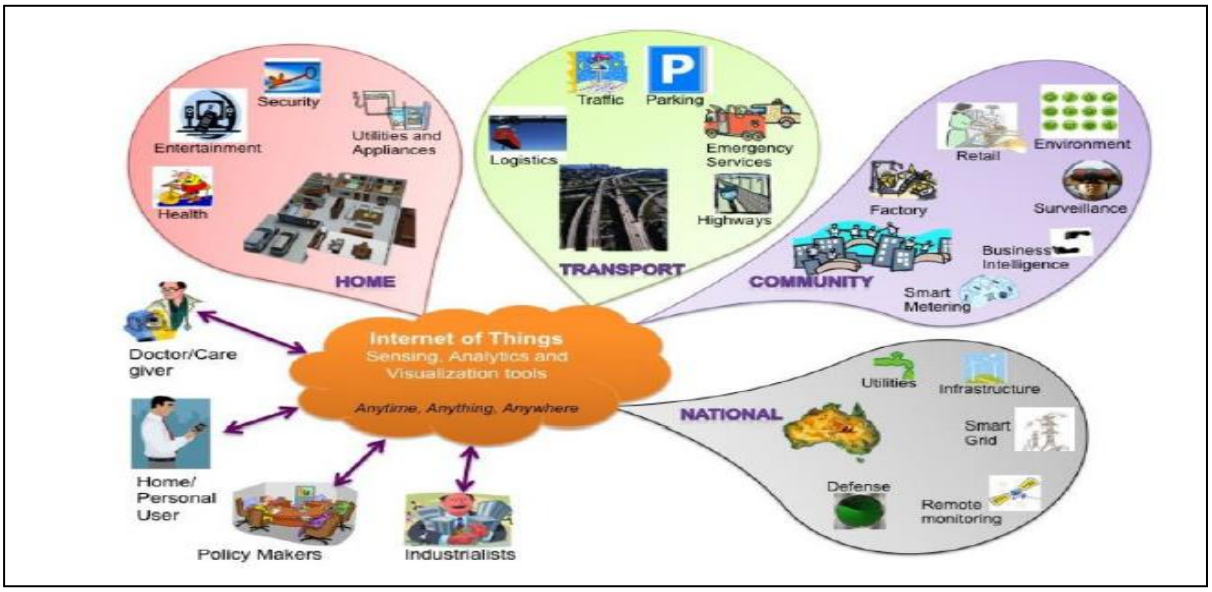


Figure 1.2 : Les domaines d'Internet des objets [21]

## 6. Sécurité de l'IdO

- **La protection de la technologie** : concerne la sécurité des données, des communications et des infrastructures réseaux et leurs fonctionnalités.
- **la protection des personnes :** concerne la protection de la vie privée des usagers pour éviter des litiges causés éventuellement par l'IdO.
- **La protection des systèmes interconnectés** : hébergeant les objets de l'IdO, concerne la protection des objets eux-mêmes livrés à ces systèmes et les processus qu'ils contrôleront **[8].**

## 7. La Motivation

L'apport de l'Internet des objets peut ainsi permettre de : **[2]**

• **Connectivité** omniprésente : diversité des solutions de connectivité sans fil, possibilité de connecter ―tout‖.

• **Disponibilité et adoption généralisée d'IP** (Internet Protocol).

• **Miniaturisation** et **coût faible** des composants électroniques.

• Progrès dans le **domaine Cloud Computing** : disponibilité des services qui permettent de bénéficier de capacités de calcul avec les objets physiques.

• Progrès dans le domaine **Big Data** : une multitude d'algorithmes sont disponibles pour collecter et analyser les données.

• Croissance du **marché de masse :** la vision du monde connecté a atteint une maturité et l'engagement est irréversible

## 8. Les avantages et les inconvénients d'IdO

### 8.1. Les avantage :

Les avantages des IdO sont nombreux, nous pouvons citer les avantages suivants :

- Améliorer les services traditionnels généraux comme le transport et les parkings.
- La surveillance et maintenance des lieux publics.
- Suivi le taux de la validité des instructions pour le travail.
- Réduire le temps perdu dans les transactions administratives dans la ville.
- Economiser du temps.
- Renforcer la sécurité routière.
- L'organisation et l'amélioration de la qualité d'Airlines.
- Economiser la consommation de l'énergie dans la ville.
- L'éclairage intelligent.

### 8.2. Les inconvénients :

- La protection des données, la vie privée et la sécurité sont souvent les principales inquiétudes des sceptiques de l'IdO. Pour calmer ces inquiétudes, il serait utile de donner aux clients les informations relatives au lieu de stockage et à la nature des données qui les concernent.
- L'installation des objets connectés est couteuse.

## 9. Les limites et les défis d'IdO

Donc, les principaux défis que rencontre l'IdO sont [11] :

- **La détection d'un environnement complexe :** des façons novatrices de saisir et de diffuser de l'information - du monde physique au nuage.
- **Connectivité :** Une variété de normes de connectivité câblées et sans fil sont requises pour permettre différents besoins d'application.
- **La puissance est critique :** de nombreuses applications IdO doivent fonctionner pendant des années sur des batteries et réduire la consommation globale d'énergie.
- **La sécurité est vitale :** protéger la confidentialité des utilisateurs et l'IP des fabricants; Détection et blocage d'activités malveillantes.
- **IoT est complexe :** le développement d'applications IdO doit être facile pour tous les développeurs, pas seulement pour les experts.
- **Cloud est important :** les applications IdO requièrent des solutions de bout en bout, y compris des services en nuage.

Donc, plusieurs obstacles pourraient ralentir la progression de l'IdO, notamment le déploiement du protocole IPv6, l'alimentation des capteurs et la définition de normes [13].

## Conclusion

Dans ce chapitre on a présenté une étude détaillée sur l'internet des objets, sa définition, son architecture et les technologies. Puis les principales composantes d'IdO, sécurité d'IdO, motivation, et ses défis. Parmi les défis majeurs de l'IdO c'est la tolérance aux fautes qui est l'une des techniques assurant la sûreté de fonctionnement, seront présentées dans le chapitre suivant.

# Chapitre II

# La Tolérance aux pannes dans L'internet des objets (l'IdO)

## Introduction

L'Un des problèmes de l'IdO est sa sûreté de fonctionnement qui est une forte contrainte spécialement si elle concerne un système médical.

Nous présentons dans ce chapitre les concepts de base de la tolérance aux pannes, nous donnons tout d'abord quelques définitions, puis les types de panne. Puis, nous présentons l'approche **auto-guérison** dans l'internet des objets

# I.1 Définitions

Cette section présente un ensemble de notion de base liée à la sûreté de fonctionnement.

### I.1.1 La Sûreté De Fonctionnement

La tolérance aux pannes s'inscrit dans le contexte plus large de la sûreté de fonctionnement. La sureté de fonctionnement (dependability) d'un système informatique est le degré de confiance relatif aux services soumis aux utilisateurs.

### I.1.2 Les Attributs De La Sureté De Fonctionnement

La tolérance aux pannes proprement dite n'est en réalité qu'une partie du concept de sûreté de fonctionnement, La sûreté de fonctionnement est composée de différents aspects, ou attributs :

- **La disponibilité :** la capacité à rendre un service a tout instant.
- **La fiabilité:** la continuité du service.
- **La sécurité -innocuité :** l'absence de conséquences catastrophiques sur l'utilisateur ou son environnement.
- **L'intégrité :** l'absence de données corrompues dans le système
- **La confidentialité** : l'absence de divulgation de données non-autorisées.
- **La maintenabilité** : l'aptitude du système à être réparé ou amélioré.

### I.1.3 Les Moyens D'Assurer La Sureté De Fonctionnement

Les principales approches pour assurer la sureté de fonctionnement sont :

- **La prévention des fautes:** qui s'attache aux moyens permettant d'éviter l'occurrence de fautes dans le système. Ce sont généralement les approches de vérification des modèles conceptuels (modularisation, utilisation de langage fortement typé, preuve formelle,. . .).
- **L'élimination des fautes:** qui se focalise sur les techniques permettant de réduire la présence de fautes ou leurs impacts. Cela est réalisé par des méthodes statiques de preuve de la validité du système (simulation, preuves analytiques, tests,. . .).
- **La prévision des fautes:** qui prédit l'occurrence des fautes (temps, nombre, impact) et leurs conséquences. Ceci est réalisé généralement par des méthodes d'injection de fautes afin de valider le système relativement à ces fautes.
- **La tolérance aux pannes:** qui essaye de fonctionner en dépit des fautes. Le degré de tolérance aux pannes se mesure par la capacité du système à continuer a délivrer son service en présence des fautes.

## I.2 Terminologie de base

Pour mieux comprendre la notion de la tolérance aux pannes, nous introduisons la terminologie suivante :

• Une **faute** est toute faiblesse (défaut) inhérente à un système qui mène à une erreur.

• Une **erreur** est un état système incorrect ou non défini que peut mener à un échec.

• Un **échec** (ou panne) est une déviation du bon fonctionnement du système.

• La **tolérance aux pannes** est la capacité d'un système fonctionnel de continuer à exécuter son fonctionnement escompté en présence de fautes ou d'erreurs.

• La **détection des pannes** consiste à la détection de la fonctionnalité défectueuse dans un système par l'auto-diagnostic ou le diagnostic coopératif.

• Le **recouvrement des pannes** est la récupération du fonctionnement correct, après la détection de la faute, en réparant ou en remplaçant le composant responsable de la panne.

## I.3 Types des pannes

Les pannes qui peuvent survenir durant une exécution repartie peuvent être classées en quatre catégories **[7]** , la Figure 2.1 montre la relation entre les types de pannes :

- ✓ **Les pannes franches (crash, fail-stop) :** que l'on appelle aussi arrêts défaillance, c'est le cas le plus simple. On considère qu'un processus peut-être dans deux états,

soit il fonctionne et donne le résultat correct, soit il ne fait plus rien. Dans le second cas, le processus est considère comme définitivement défaillant.

- ✓ **Les pannes par omission (transient, omission failures) :** Dans ce cas, on considère que le système peut perdre des messages. Ce modèle peut servir a représenter des défaillances du réseau plutôt que du processus.
- ✓ **Les pannes de temporisation (timing, performance failures) :** Ce sont les comportements anormaux par rapport à un temps, comme par exemple l'expiration d'un délai de garde.
- ✓ **Les pannes arbitraires, ou byzantines (**malicious**, byzantine failures)** : Cette classe représente toutes les autres pannes : le processus peut alors faire "n'importe quoi", y compris avoir un comportement malveillant.

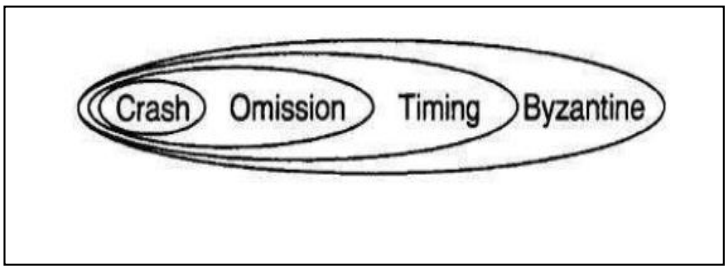


Figure 2.1 : les classes des pannes

## I.4 Systèmes Auto-guérison

Jusqu'à présent, nous avons présenté les mécanismes de contrôle d'exécution et de tolérance aux pannes. Cependant, l'objectif de cette thèse est de fournir une approche d'exécution plus intelligente pour les services internet des objets. Cette raison, une partie de cette thèse améliore les services internet des objets avec des capacités d'auto-guérison

Les quatre principaux aspects de l'informatique autonome sont l'auto configuration, l'auto optimisation, l'auto-guérison et l'autoprotection. IBM a défini ces aspects comme suit: **[5]**

• **Auto-configuration:** il s'agit de l'installation, de la configuration et de l'intégration automatiques des systèmes.

• **Auto-optimisation:** il s'agit de l'amélioration automatique des performances et de l'efficacité des systèmes; par exemple, en réglant les paramètres du système.

• **Auto-guérison:** il s'agit de la détection et de la récupération automatiques des problèmes et des pannes.

• **Autoprotection:** il s'agit de la détection et de la récupération automatiques des attaques.

Plus tard, dans une enquête sur la recherche sur les systèmes d'auto-guérison, Ghosh et ses co-auteurs [6 ] ont donné la définition suivante:

«**L'auto-guérison** peut être définie comme la propriété qui permet à un système de percevoir qu'il ne fonctionne pas correctement et, sans (ou avec) intervention humaine, de faire les ajustements nécessaires pour se rétablir.»

De plus, Ghosh et ses co-auteurs [6] ont introduit le modèle d'auto-guérison illustré à la

Figure 2.2 pour souligner l'importance de comprendre la normale vs.

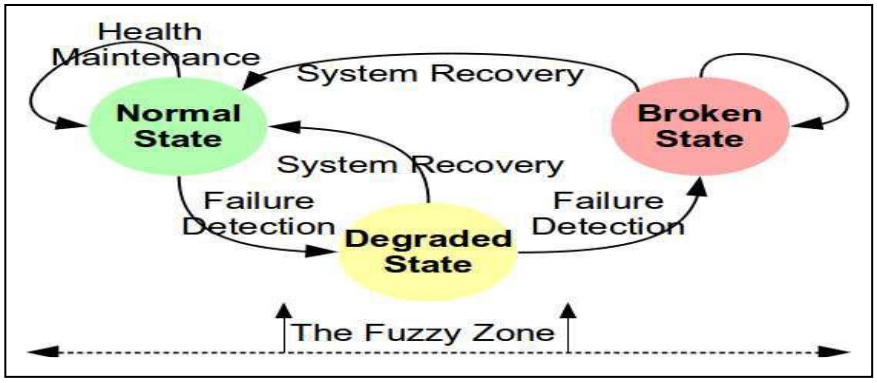


Figure 2.2 : États d'auto-guérison (tirés de [6])

Comportement anormal lors de la conception et de l'étude de systèmes complexes. La figure 2.2 illustre les trois états possibles d'un système d'auto-guérison et les transitions entre eux. L'état

«Dégradé» fait référence à une zone floue où il n'y a pas de distinction claire entre un état «normal» et un état «cassé». Ces zones floues modélisent le fait que les systèmes distribués à grande échelle sont composés de composants modulaires; donc; si une

petite partie du système tombe en panne, le reste devrait pouvoir continuer ses opérations sans interruption, tandis que les mécanismes de tolérance aux pannes tentent de résoudre les problèmes détectés et de revenir à l'état «normal».

La transition de maintien de la santé fait référence à la vérification et à la maintenance constante de la fonctionnalité normale du système. Une méthode courante pour maintenir l'intégrité du système consiste à assurer la redondance des composants du système. Le système peut passer d'un état «normal» à un état «dégradé» en détectant une panne, mais il peut revenir à un état «normal» en réparant la panne.

De la même manière, le système peut passer d'un état «dégradé» à un état «cassé» en détectant une panne, et passer à un état «normal» en réparant la panne. Le système peut également rester dans un état «cassé» si la panne n'est pas réparée.

Plus loin dans cette thèse, nous utilisons le modèle illustré à la figure 2.2 comme base pour décrire les propriétés d'auto-guérison des services IdO; c'est-à-dire que nous dé finissons les états normal, dégradé et cassé pour les services IdO.

## II. Système multi Agent

Le domaine des Systèmes Multi-Agents (SMA) est un axe de recherche très actif. Cette discipline est à la connexion de plusieurs domaines en particulier de l'intelligence artificielle, des systèmes informatique distribués et du génie logiciel.

### II.1 Définition

D'après Ferber « *On appelle système multi-agent (ou SMA), un système composé des éléments suivants :*

- *Un environnement E, c'est à dire un espace disposant généralement d'une métrique.*
- *Un ensemble d'objets O. Ces objets sont situés, c'est à dire que, pour tout objet, il est possible à un moment donné, d'associer une position dans E. Ces objets sont passifs, c'est à dire qu'ils peuvent être perçus, crées, détruits et modifiés par les agents.*
- *Un ensemble A d'agents, qui représentent les entités actives du système.*
- *Un ensemble de relations R qui unissent les objets et les agents entre eux.*
- *Un ensemble d'opérations Op permettant aux agents A de percevoir, produire, consommer, transformer et manipuler les objets de O.*

- *Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers.* » [12]

## II.2 Caractéristiques des systèmes multi agents

Les systèmes multi-agent sont caractérisés par : [13]

- Tout agent à des informations ou des capacités de résolution de problèmes limitées, ainsi tout agent à un point de vue partiel ;
- Pas de contrôle global du système multi-agents ;
- Les données sont distribuées ;
- Le calcul est asynchrone.

## II.3 Définition d'un agent

Il en existe plusieurs définitions du concept d'agent :

- D'après Ferber *« Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement, qui, dans un univers multi-agents, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents. »* [12]
- D'après Russell *« Un agent est une entité qui perçoit son environnement et agisse sur celui-ci. »* [14]

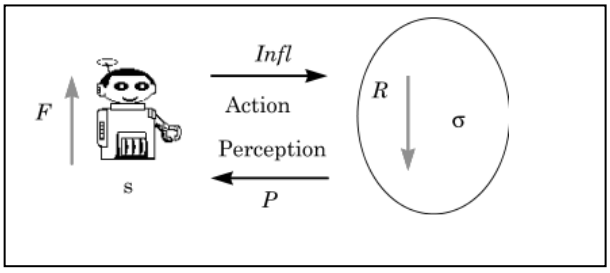


Figure 2.3 : Un agent en interaction avec son environnement dans un système multi-agent [15]

## II.5. Caractéristiques d'agents

Un agent est caractérisé par plusieurs propriétés qui varient d'un agent à un autre, nous citons à titre d'exemples les suivantes : [13]

- **Situé** : L'agent doit être capable de percevoir et agir sur son environnement.
- **Autonome** : L'agent doit être capable d'agir sans l'intervention d'un tiers, et contrôler ses propres actions ainsi que son état interne.
- **Proactif** : L'agent est capable de prendre l'initiative.
- **Social** : L'agent doit avoir la capacité d'interagir avec d'autres agents, pour accomplir ses propres tâches ou ceux des autres.
- **Réactif** : L'agent doit être capable de répondre dans le temps requis lors d'une perception de son environnement.

## II.6. Classement des agents

Les agents peuvent être classes en deux catégories principales selon leur comportement et leur granularité. Cette notion de granularité est bien sur très subjective, elle exprime la complexité de « raisonnement » d'un agent afin de séparer les agents dits "intelligents" et des agents moins "intelligents". On parle d'agents cognitifs et d'agents réactifs.

### II.6.1. Cognitifs :

Ils peuvent anticiper, prévoir le futur, mémoriser des choses ... ils réfléchissent.

Il est intelligent par lui-même c'est-à-dire qu'il effectue un certain raisonnement pour choisir ses actions. [11]

### II.6.2. Réactifs :

Ils réagissent directement à l'environnement perçu, par pulsion (ex : les fourmis). Un tel agent se contente simplement d'acquérir des perceptions et de réagir à celles-ci en appliquant certaines règles prédéfinies. [11]

| Systèmes d'agent cognitifs | Systèmes d'agent réactifs |
|---|---|
| Représentation explicite de l'environnement | Pas de représentation explicite |
| Peut tenir compte de son passé | Pas de mémoire de son historique |
| Agents complexes | Fonctionnement stimulus/action |
| Petit nombre d'agents | Grand nombre d'agent |

Tableau 2.1 : Définir la différence entre l'agent cognitif et l'agent réactif [10]

### II.6.3. Hybride

Combinaison des deux, Chaque agent hybride est caractérisé par la notion de couches et chaque couche représente soit les agents cognitifs, soit les agents réactifs. Un agent hybride est composé de plusieurs couches arrangées selon une hiérarchie. [11]

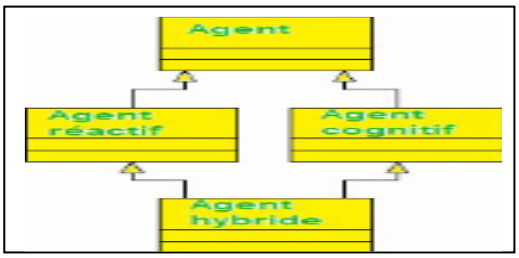


Figure 2.4 : La structure des agents hybrides

## II.7. Le rôle des agents

- Le rôle est une représentation abstraite d'une fonction ou d'un service propose par un agent.
- Un rôle peut être attribue dynamiquement a un agent.

- Chaque méthodologie peut appréhender le rôle de différentes façons. Certaines proposent d'associer potentiellement plusieurs rôles à un agent. D'autres spécifient qu'un rôle est au contraire tenu par plusieurs agents. [9]

## II.8. Les systèmes multi-agents dans l'IdO

Dans [22], les auteurs ont présenté une architecture qui illustre l'intégration des SMA dans l'architecture IdO (figure.5). Nous allons utiliser cette architecture comme référence pour l'architecture que nous allons proposer dans chapitre III .

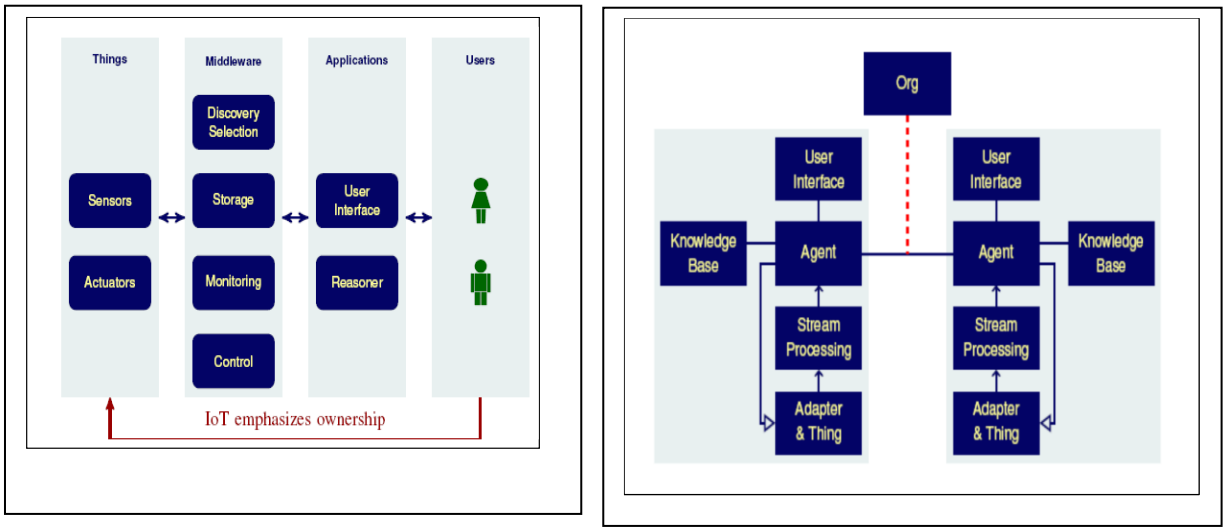


Figure 2.5 : vers une architecture multi-agents de l'IdO

Comme le montre la figure 2.5, la décentralisation nécessite que les agents interagissent les uns avec les autres. C'est-à-dire que les protocoles auxquels nous nous référons doivent être mis en œuvre via des communications sans lien de dépendance, en d'autres termes, via la messagerie entre les agents.

## III. Travaux connexes

La tolérance aux pannes d'un système IdO est l'aptitude de ce système à accomplir sa fonction malgré la présence ou l'occurrence des fautes, qu'il s'agisse de dégradations physiques du matériel, de défauts logiciels, d'attaques malveillantes, d'erreurs d'interaction homme-machine. Après une vision d'ensemble de la tolérance aux pannes des systèmes IdO et la capacité des systèmes multi-agents d'automatiser ce processus, cette section permet de synthétiser quelques travaux dans ce domaine :

Dans [5], l'auteur présente une approche d'exécution auto-corrective (self-healing) de services composites, basée sur des agents capables de prendre, de manière autonome, des décisions pendant l'exécution des services, à partir de leurs connaissances. Angaritta

défini, de manière formelle, en utilisant des réseaux de Pétri colorés, les services composites, leur processus d'exécution, et leurs mécanismes de tolérance aux pannes. L'approche offre plusieurs mécanismes de reprise sur panne alternatifs : la récupération en arrière avec compensation ; la récupération en avant avec réexécution et/ou remplacement de service ; et le point de contrôle (check- pointing), à partir duquel il est possible de reprendre l'exécution du service ultérieurement. Les services sont contrôlés par des agents, i.e. des composants dont le rôle est de s'assurer que l'exécution des services est tolérante aux pannes.

Dans [23], les auteurs ont proposé, une architecture générique, multicouche et basée sur des agents pour les systèmes IoT, appelée IoTAA (Internet of Things Agents Architecture). IoTAA est composé de quatre couches ; chaque couche maintient et planifie des fonctionnalités spéciales, telles que la connectivité et la communication entre les objets, la coordination locale, la coordination globale, les fonctionnalités dépendantes du domaine. D'autre part, le diagnostic des systèmes IoT est un problème important qui vise à détecter les anomalies du comportement souhaité du système, à identifier les causes de défaillance et à localiser les composants de défaillance. En conséquence, le travail vise à montrer l'efficacité de l'IoTAA pour modéliser le problème de diagnostic de l'IoT. Les quatre couches sont développées et des traitements supplémentaires sont proposés. L'architecture de diagnostic proposée basée sur l'IoTAA est illustrée par un exemple de surveillance des soins de santé.

## Conclusion

La tolérance aux fautes est un problème important pour l'IdO. Selon la déclaration de l'IETF [**8**] des concepts de l'IdO, le but de la tolérance aux fautes dans l'Internet des objets est de « maintenir robuste et confiance un réseau dynamique a basée système multi agent », et le chapitre suivant détail la notre contribution dans ce domaine.

# Chapitre III

# Conception

## Introduction

Ce chapitre est consacré à la conception et la réalisation de service d'internet des objet pour le suivi médical a distance avec l'approche auto-guérison a base système multi Agents.

## 1. Problématique et objectif

Pour répondre au besoin de l'utilisateur qui cherche à obtenir le bon service au bon moment et au bon endroit, on pense qu'il existe plusieurs défis à la mise en œuvre réussie de l'Internet of Things (IoT).

Mais le problème est que si le système tombe en panne, que va-t-il se passer ? Et que faisons-nous ?

**L'objectif** de ce projet est de continuer le fonctionnement du système en cas de panne au moyen d'une approche **d'auto-guérison**

## 2. L'utilisation d'agent dans l'approche auto-guérison et IdO

Dans cette approche, les agents de service sont des agents basés sur des connaissances. Pour effectuer la détection, le diagnostic et la récupération a d'une manière centralisé.

### 2.1 Motivations d'utilisations d'agents

Plusieurs raisons peuvent expliquer l'engouement soudain pour cette technologie.

- De nombreux problèmes sont par nature distribués.
- La vitesse des transmissions, la sécurité des communications et la standardisation des protocoles rendent plus facile leur mise en œuvre.
- Des langages de haut niveau (Java, C++, …) permettent l'écriture rapide de système d'agents.
- En plus, la migration des agents mobiles à des sites distants permet la confrontation de plusieurs experts, permettant ainsi de bénéficier et d'appendre de leurs savoir-faire.

### 2.2 Les avantage des systèmes multi agents

- **La modularité :** la modularité du système est due essentiellement à l'autonomie des agents. Comme les interactions inter agents se basent essentiellement sur

l'échange de messages, le couplage entre eux est faible. Ainsi d'autres concepts de base des systèmes multi agents contribuent à cette propriété.

- L**a réutilisation :** la réutilisabilité est due à la modularité du système.
- **La facilité de maintenance :** cet avantage est vérifié grâce à la modularité du système.
- **La fiabilité :** la fiabilité est la capacité d'un système de continuer à fonctionner malgré l'apparition des pannes ou des erreurs, cet avantage est vérifié grâce à l'autonomie des agents.
- **L'efficacité :** L'efficacité des logiciels est mesurée en fonction des ressources utilisées au cours du processus de la résolution du problème. Comme les systèmes multi agents sont des systèmes distribués avec des ressources distribuées, ces dernières sont utilisées de façon rationnelle.
- **L'adaptation de réalité :** grâce aux caractéristiques des agents, ce paradigme permet de modéliser des phénomènes réels.
- **Les modes d'interaction sophistiqués :** ce paradigme supporte des modes d'interaction sophistiqués par rapport au paradigme d'objet, comme la coopération, la coordination et la négociation.
- **L'intelligence.**

## 2.3 Caractéristiques des systèmes multi agents

Les systèmes multi-agent sont caractérisés par :

- Tout agent a des informations ou des capacités de résolution de problèmes limitées, ainsi tout agent a un point de vue partiel ;
- Pas de contrôle global du système multi-agents ;
- Les données sont distribuées ;
- Le calcul est asynchrone.

## 2.4 Caractéristiques d'agents

Un agent est caractérisé par plusieurs propriétés qui varient d'un agent à un autre, nous citons à titre d'exemples les suivantes :

**Situé** : L'agent doit être capable de percevoir et agir sur son environnement.

**Autonome** : L'agent doit être capable d'agir sans l'intervention d'un tiers, et contrôler ses propres actions ainsi que son état interne.

**Proactif** : L'agent est capable de prendre l'initiative.

**Social** : L'agent doit avoir la capacité d'interagir avec d'autres agents, pour accomplir ses propres tâches ou ceux des autres.

**Réactif** : L'agent doit être capable de répondre dans le temps requis lors d'une perception de son environnement.

## 3. Scénario

,

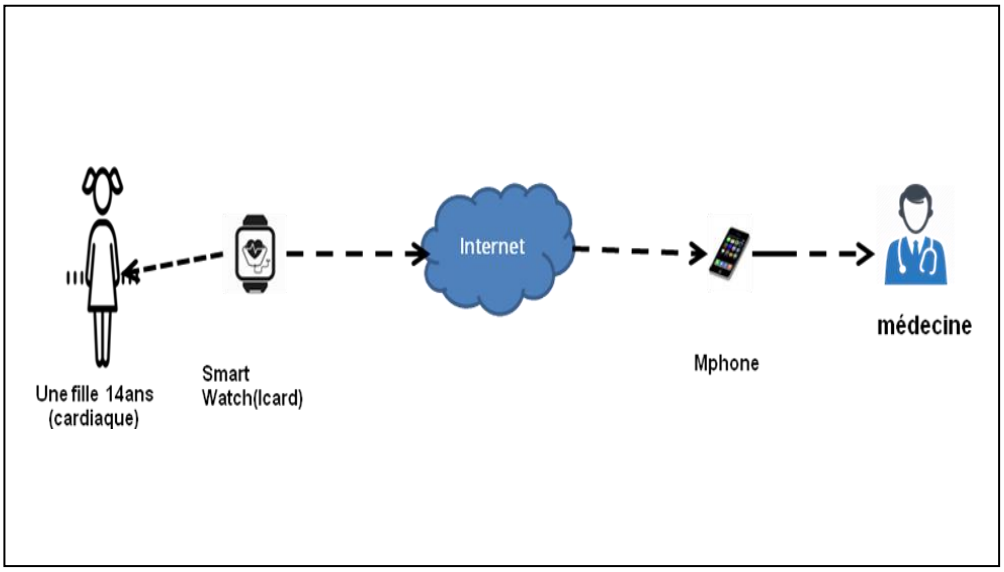


Figure 3.1 : Schéma de Scenario

Dans la figure 1.3 **:** Jouri une fille cardiaque âgé de 14 ans il s'est fait implanter une montre-bracelet intelligente sans fil (**icard**) .Il mesure la fréquence cardiaque et respiratoire et lui dit quand il est sur le point d'atteindre un état critique.

Hier, Jouri a fait de sport dans l'école avec ses camarades désobéissant aux prescriptions de son médecin, Quelques minutes plus tard, la fréquence cardiaque et respiratoire commence à augmenter et il dépasse peut-être le seuil normal en quelques instants. L'**icard** reconnaît l'état et envoie un message à son **cPhone**, un téléphone intelligent équipé de communicateurs **IoT** intégrés qui lui permettent de communiquer avec n'importe quelle interface réseau sans fil (avec la famille et la médecine ou l'hôpital).

Le cPhone met en place une l'alarme et une voix agréable commence à lui parler, Lui rappelle d'arrêter son activité.

## 4. Architecture de système à base d'Agents

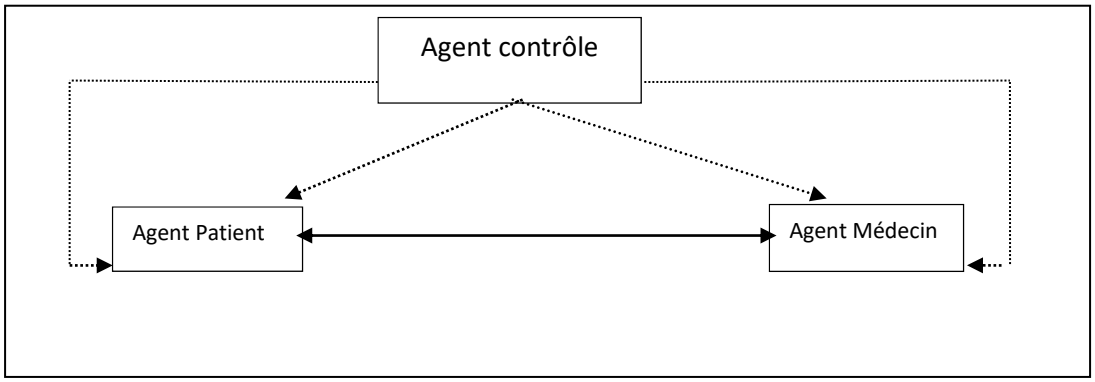


Figure 3.2 : Architecture de Système

Dans cette figure (3.2) a été explique le système est compose trois agent (agent patient, agent médecin, agent contrôle)

- **Agent contrôle :** pour contrôler si existe un en panne et utilise le recouvrement pour la tolérance aux pannes.
- **Agent patient :** c'est le malade (cardiaque) utilise une montre connectée pour mesurer la fréquence du cœur.
- **Agent médecin :** c'est lui que surveille l'état du patient.

## 5. Architecture d'agent

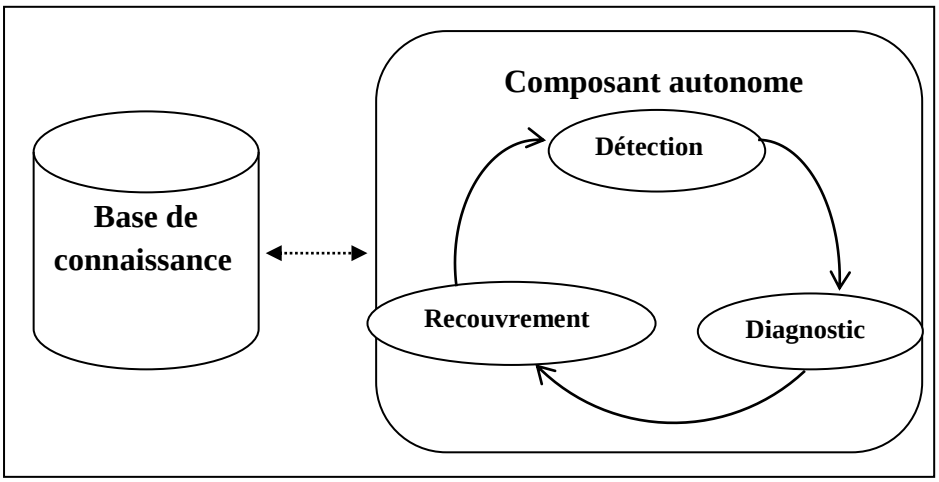


Figure 3.3 : Architecture d'agent

Dans la figure 3.3 qui représente l'architecteur d'agents qui compose :

• **Composant autonome:** il détecte les dégradations sur le comportement de système, sélectionne une action appropriée et l'applique. Contient :

**Détection :** pour découvrir un dysfonctionnement, une interruption.

**Diagnostique** : le diagnostic de système désigne toute méthode permettant de déterminer si un machine est défaillante ou non et de déterminer l'origine de la panne a partir des informations relevées par observation, contrôles et tests.

**Recouvrement :** elle consiste à réparer la panne pour permettre au systeme de poursuivre son exécution, réessayer l'invocation de service ou trouver un service remplacant sont des techniques utilisées pour fournir une récupération.

• **Base de connaissances:** une base de connaissances regroupe des connaissances spécifiques à un domaine spécialisé donné, sous forme exploitable par un ordinateur, et dans notre système contient des informations sur l'agent de service lui-même, son service correspondant et le contexte d'exécution. Il contient également un ensemble de règles pour la transition entre les états d'auto-guérison et un ensemble de règles pour déduire les actions à entreprendre.

## 6. Modalisation AUML

AUML « Agent UML » est un projet relativement jeune proposé en 1999 et les premiers travaux de spécification ont débuté en 2003. AUML vise à proposer un ensemble de notations mieux adaptées au paradigme multi-agent, en étendant le langage UML.

AUML adopte trois niveaux de représentations :

**Niveau1 :** représentez le protocole global (diagramme de séquence, Package, Template).

**Niveau2 :** une représentation des interactions entre les agents (diagramme de séquence, activités, collaboration, état)

**Niveau3 :** une représentation interne aux agents (diagramme d'activité, et d'état de transition) [9]

## 6.1. Diagramme de classe

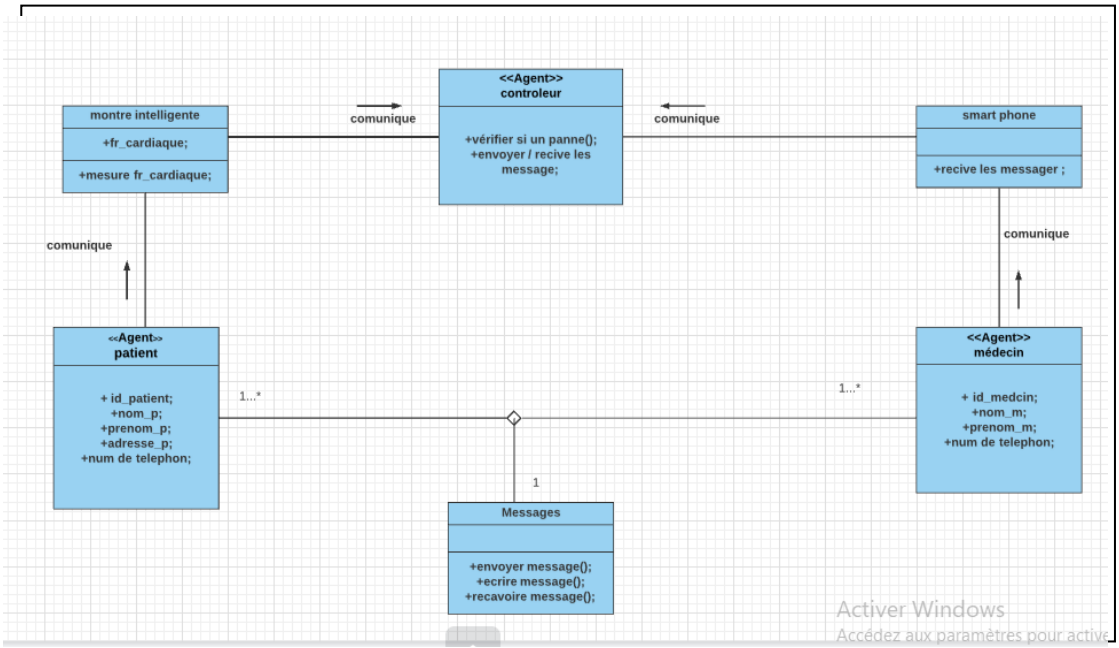


Figure 3.4 diagramme de classe

## 6.2 Diagramme de cas d'utilisation

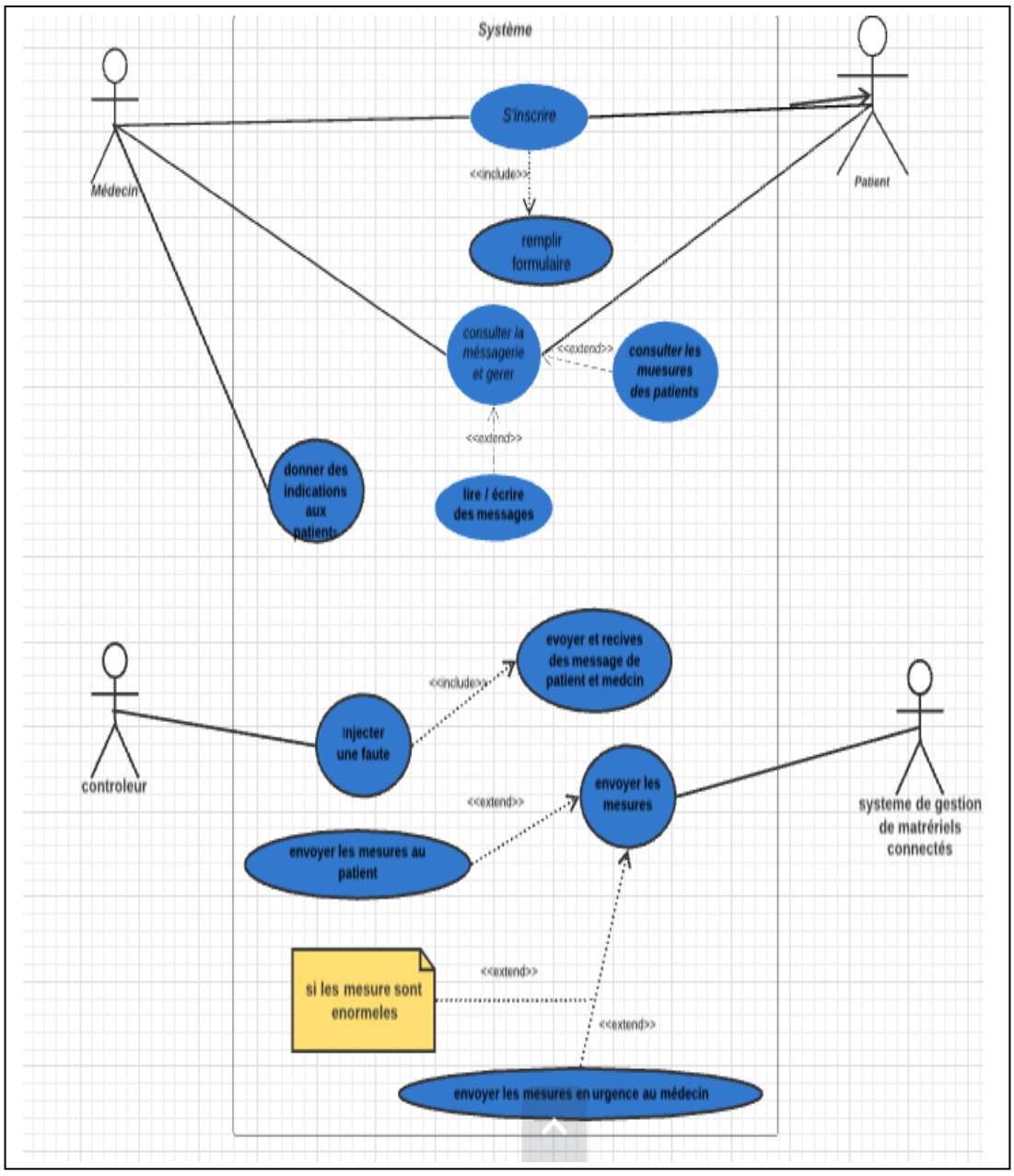


Figure 3.5 : diagramme cas d'utilisation

### 6.3.1 Digramme de séquence : « l'authentification »

L'authentification pour un système informatique est un processus permettant au système de s'assurer de la légitimité de la demande d'accès faite par une entité afin d'autoriser l'accès de cette entité à des ressources du système, cette procédure permet donc de valider la légitimité de l'accès de l'entité, ensuite le système attribue à cette entité les données d'identité pour cette session.

Titre **:** authentification

Résumé **:** Ce cas d'utilisation permet à l'utilisateur authentifier le systeme

Acteur : l'utilisateur. Système.

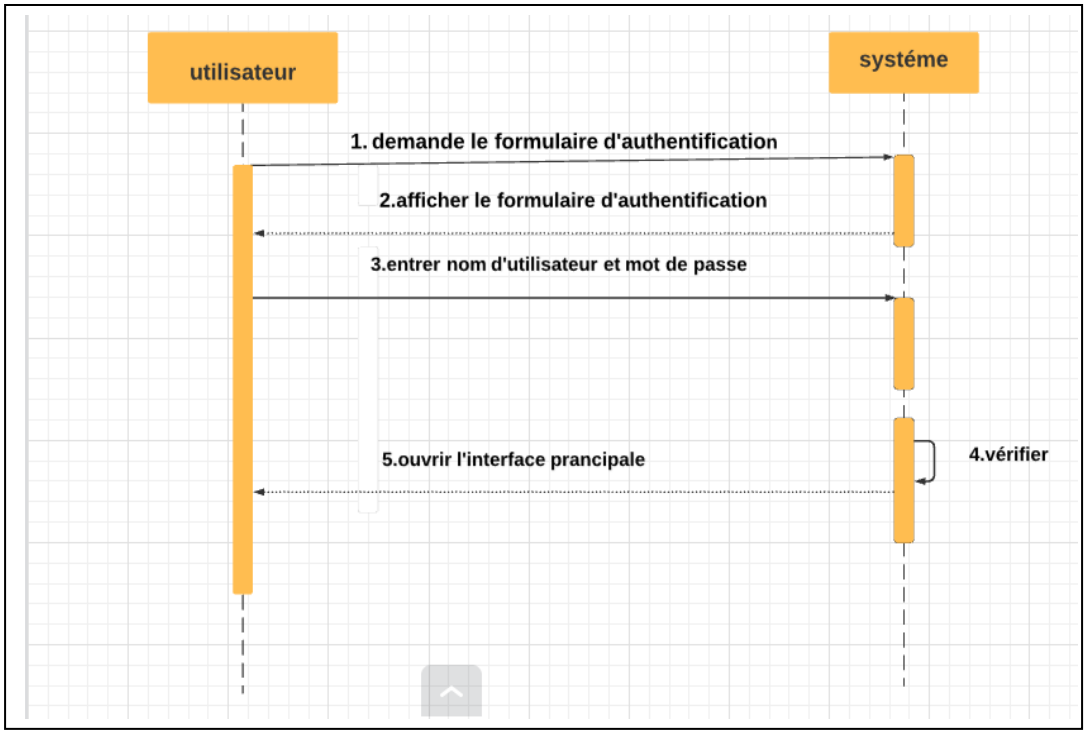


Figure 3.6 : diagramme de séquence « authentification »

### 6.3.2 Diagramme de séquence : « Ajouter ou supprime ou modifier un patient »

Le système fournie à l'agent patient une interface lui permettant de s'enregistrer dans la base de données. Lorsque agent patient choisit enregistrement, l'application affiche le formulaire dédié, patient remplis les champs et valide son action, puis l'application

sauvegarde les informations fournies par le patient pour ajouter dans la base , en plus capable supprime ou modifier le patient.

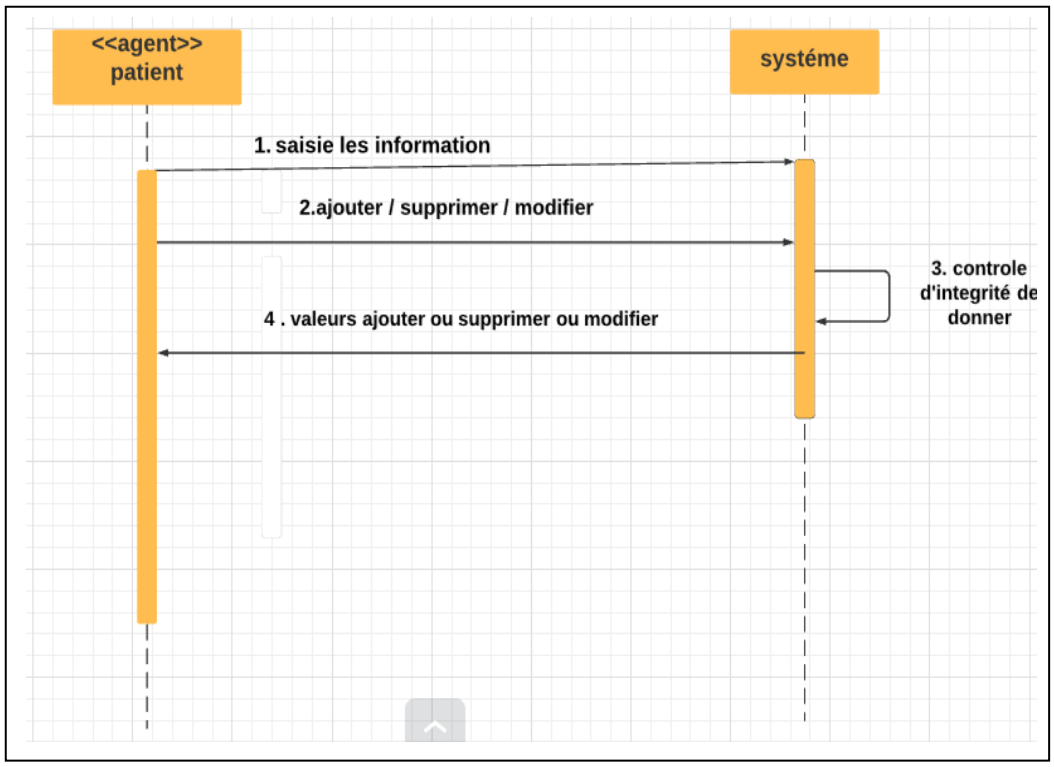


Figure 3.7 : diagramme de séquence « ajouter/supprime/modifier un patient »

### 6.3.3 Diagramme de séquence : « système globale »

**Titre :** système globale.

**Résumé :** Ce cas d'utilisation permet à l'agent contrôleur vérifier si un panne par envoyer des messages (chaque 5s) vers l'agent patient et l'agent médecin, et il faut répondre (bien reçu)
En plus changer d message entre l'agent patient et l'agent médecin pour les mesures de fréquence cardiaque (si critique ou non).

**Acteur** : agent contrôleur, agent patient , agent médecin.

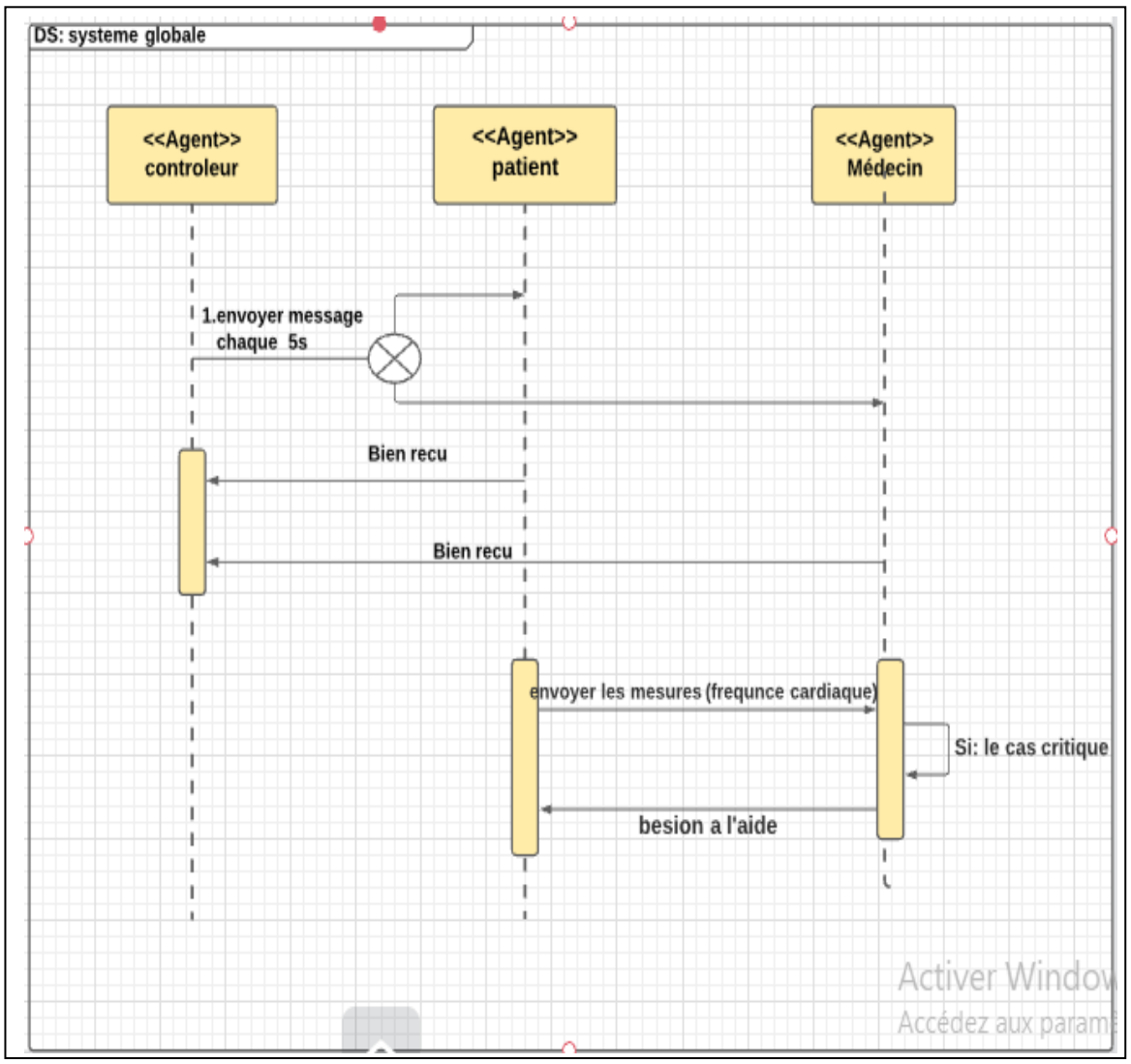


Figure 3.8 : diagramme de séquence « système globale »

## Conclusion

Dans ce chapitre nous avons vu la conception générale de notre système proposé en utilisant l'AUML comme outil de modélisation, puis nous allons faire l'implémentation de ce système, Ce qui va être présenté dans le prochain chapitre.

# Chapitre IV

# L’Implémentation De La Proposition

## Introduction

Dans ce chapitre on va présenter deux parties principales : la première partie dévoile l'environnement de développement qui contient les différents langages et technologies, et la deuxième partie présente la description de la simulation notre proposition ainsi que les interfaces permettant son déroulement.

## 1. La Représentation Des Technologies et Les Langages

Cette partie concerne la présentation des outils technologiques

### 1.1 PhpMyAdmin

phpMyAdmin (PMA) (voir figure 4.1) est une application Web de gestion pour les systèmes de gestion de base de données MySQL réalisée principalement en PHP et distribuée sous licence GNU GPL…[16]


*Figure 4.1 :* figure *qui représente* le logo de phpmyadmin [16]

### 1.2 NetBeans

Est un environnement de développement intégré (IDE) pour Java, placé en open source par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, XML et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages web). NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X et Open VMS. NetBeans est lui-même développé en Java, ce qui peut le rendre assez lent et gourmand en ressources mémoires. [17]


*Figure 4.2 : figure qui représente* le logo de NetBeans

### 1.3 Jade

Java Agent Développement Framework, ou JADE (voir figure 4.2), est un Framework logiciel pour le développement d'agent intelligent, implémenté en Java. Le système JADE prend en charge la coordination entre plusieurs agents FIPA et fournit une implémentation standard du langage de communication FIPA-ACL, ce qui facilite la communication entre agents. JADE a été initialement développée par Telecom Italie et distribué sous forme de logiciel libre.


*Figure 4.3: figure qui représente le logo de JADE*

## 2. La Description De La Simulation

Notre simulation repose sur la définition des composants du système, la création des agents et l'injection de fautes pour tester l'efficacité de notre proposition.

### 2.1 L'Authentification

La figure suivante (figure 4.4) permet aux utilisateurs de s'authentifier. L'utilisateur doit taper le nom d'utilisateur et le mot de passe adéquat. S'ils sont valides il accède à la page d'accueil de la simulation (voir figure 4.5)

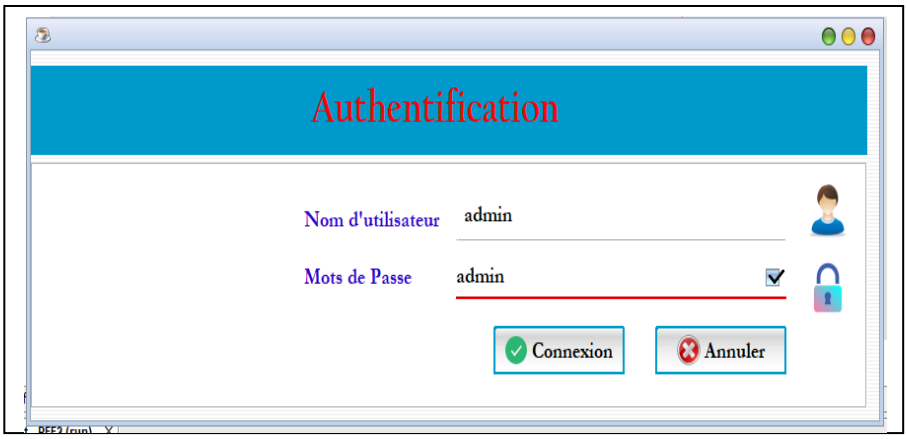


Figure 4.4 : présentation de l'Interface De l'Authentification

A la connexion deux événements se produisent :

- l'affichage de l'interface du simulateur.

- La création de l'agent globale qui lance l'interface du simulateur, et crée 3 agents locaux patient, médecin, contrôleur chaque agent dans un conteneur défèrent comme montre les figure 4.5, 4.6.

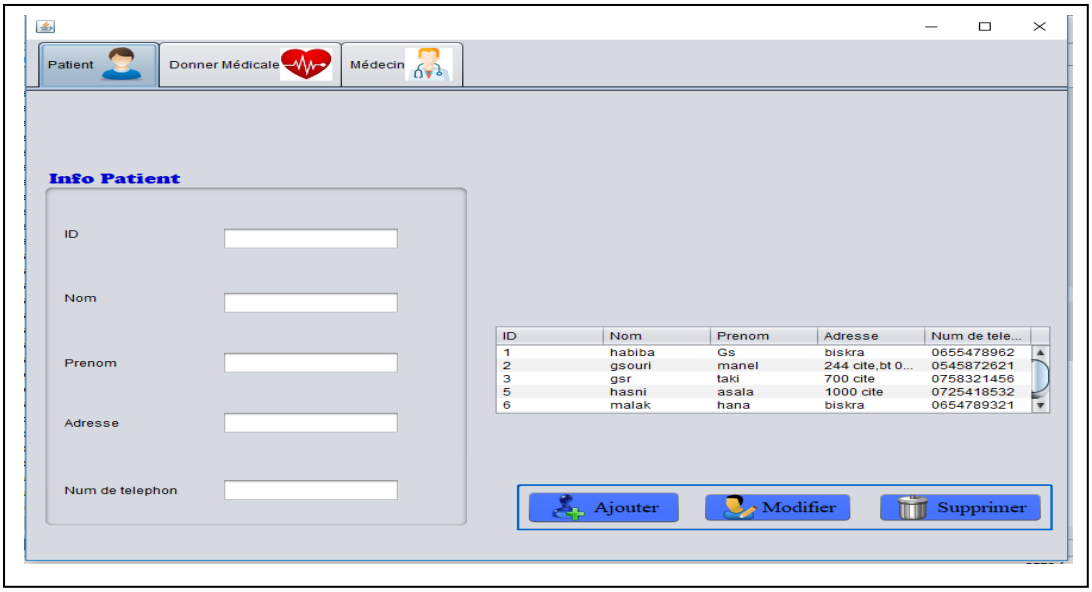


Figure 4.5 : présentation d'interface principale

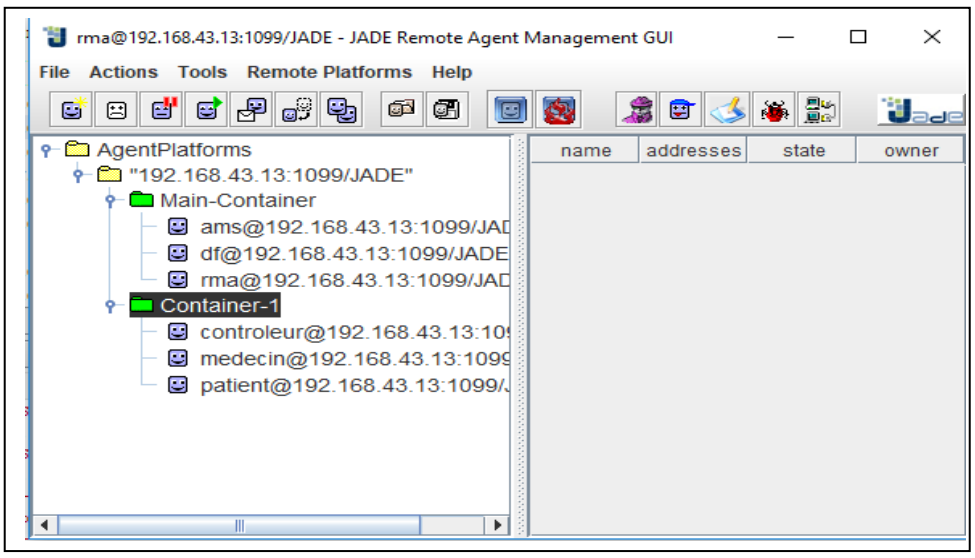


Figure 4. 6: figure qui représente l'interface jade avec des agents crées

Les tables créé dans la base de donné sont : user, patient, médecin, donner (voir le figure 4.7)

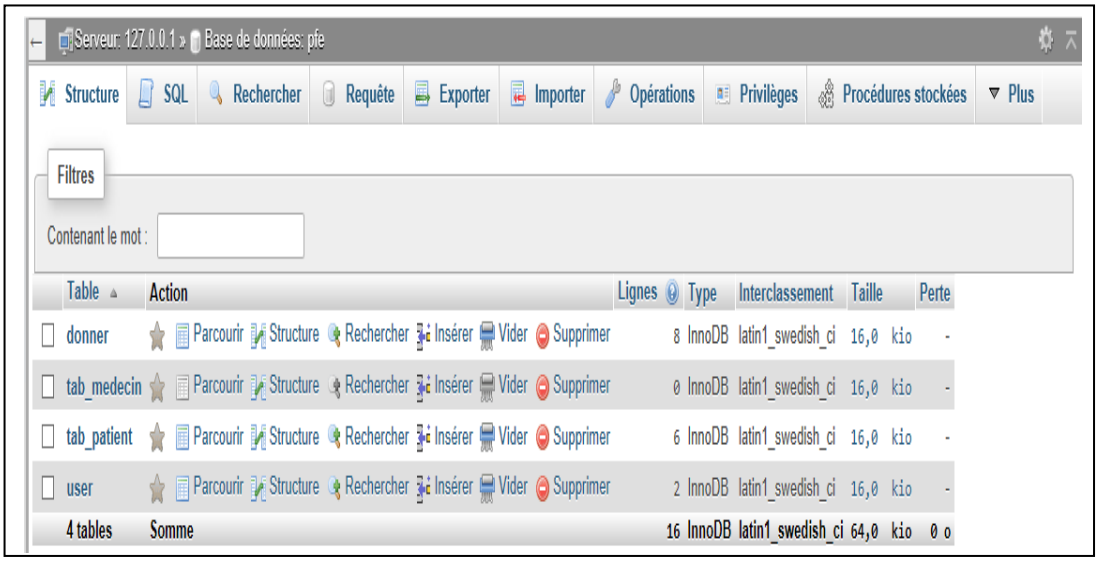


Figure 4.7 : figure qui représente la base de données

## 2.2 Patient

Dans la figure 4.8 l'onglet « patient » est ouvert car il est le premier onglet, les champs qui sont affichés doivent être rempli pour ajouter un patient.

En plus capable modifier ou supprimer patient.

Et la figure 4.9 présenter table tab_patient qui contient les patients.

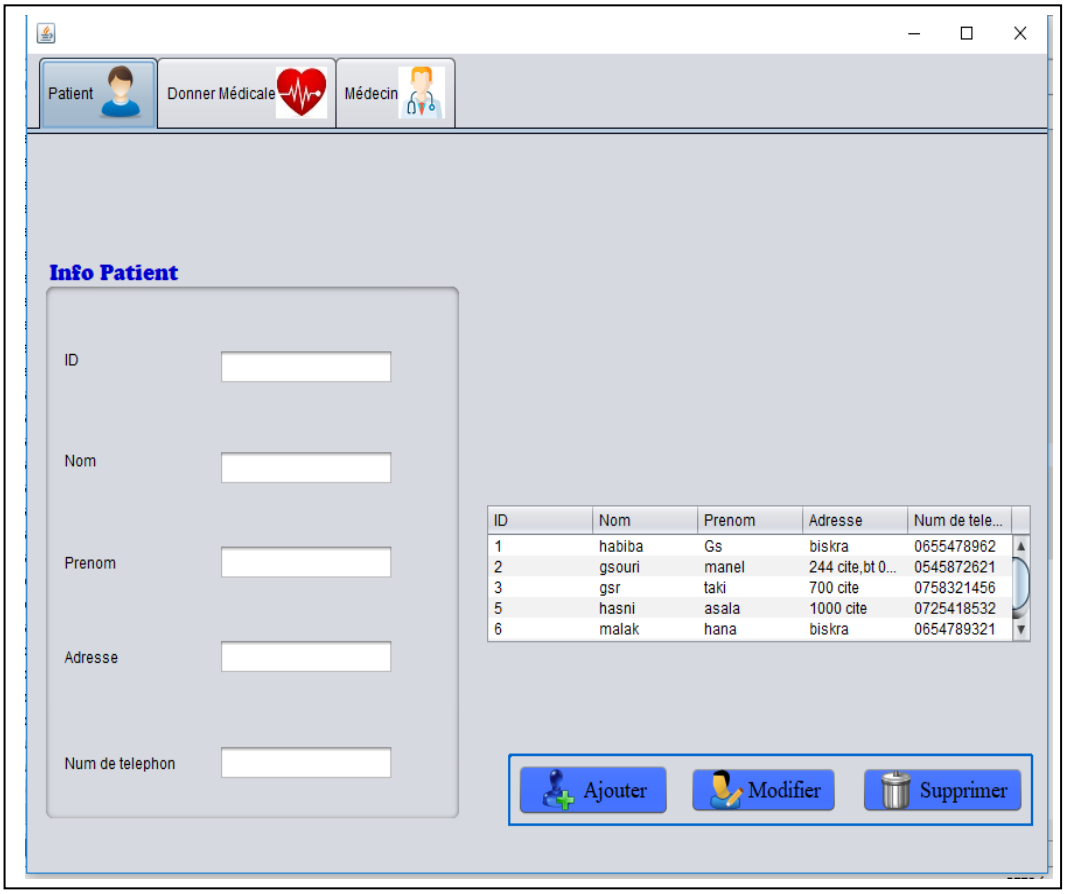


Figure 4.8 : présentation d'interface des patients

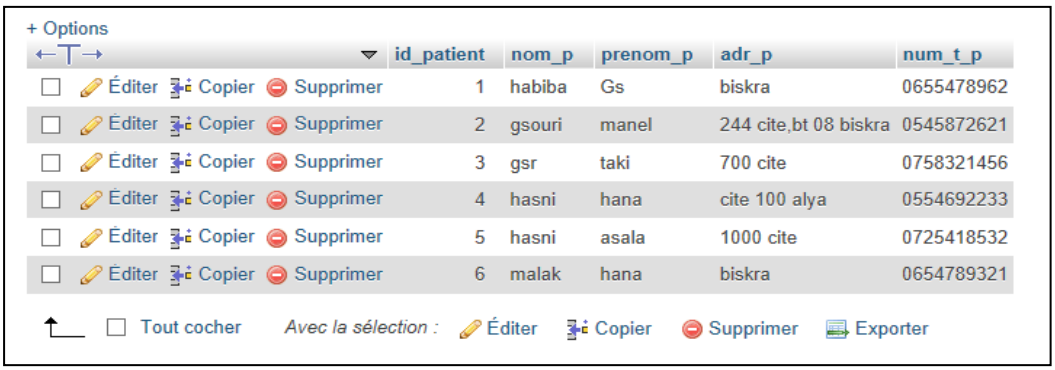


Figure 4.9 : représente table des patients (tab_patient)

## 2.2 Médecin

Dans la figure 4.10 l'onglet « médecin » pour ajouter médecin.

Et la figure 4.11 présenter table tab_medecin qui contient les médecins.

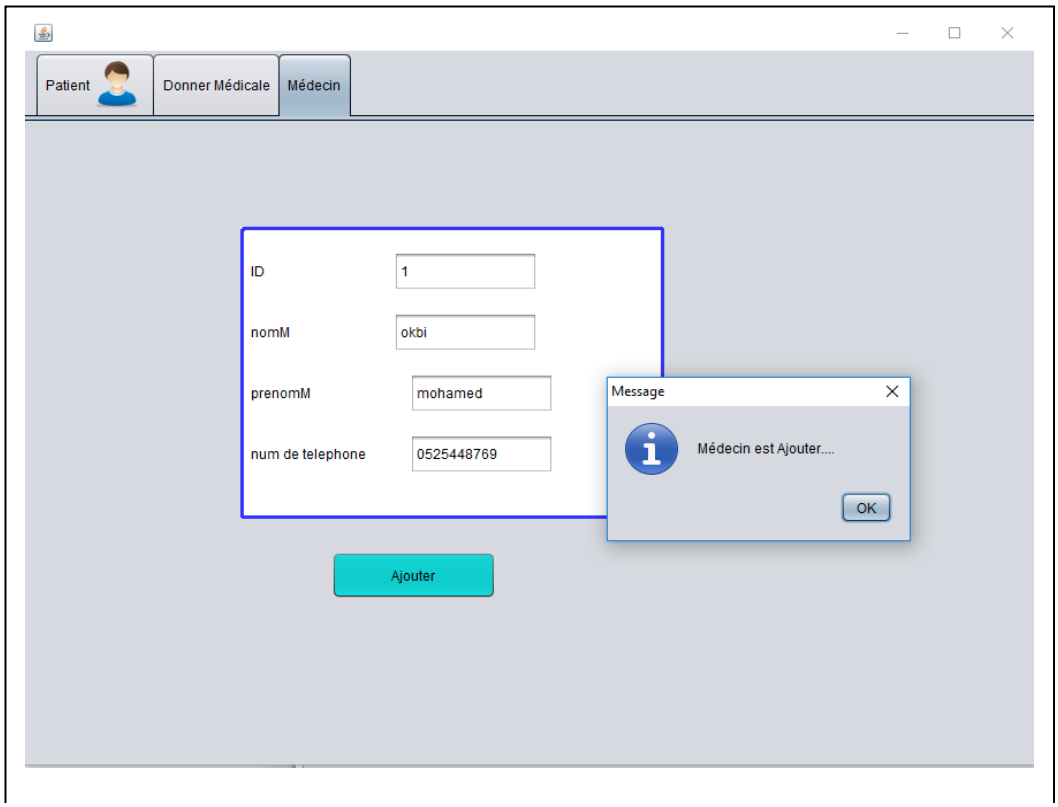


Figure 4.10 : présentation d'interface de médecin


Figure 4.11 : table de médecins (tab_medecin)

## 2.3 Page donné médicale

Dans la figure 4.12 l'onglet « **donner médicale** » pour saisies les fréquences cardiaque et envoyer (**bouton Envoyer**), et un bouton pour déclencher le panne (**Déclencher panne**).

Et la figure 4.13 présenter table **donner** qui contient les fréquences cardiaques.

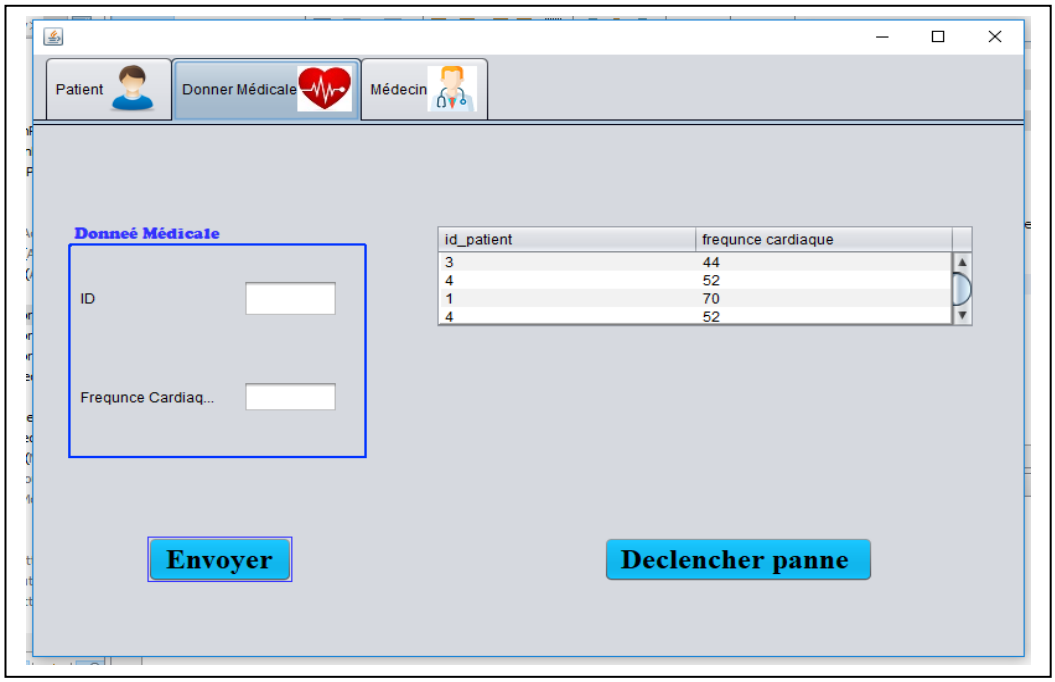


Figure 4.12 : présentation d'interface de Donner médicale

| id_patient | fr_cardiaque |
|---|---|
| 1 | 100 |
| 2 | 80 |
| 3 | 140 |
| 4 | 52 |
| 5 | 90 |
| 6 | 74 |

Figure 4.13 : représente table des fréquences cardiaques (donner)

- **Si n'est pas cas critique**

Dans la figure 4.13 dernière fréquence (id=6, fr_cardiaque= 74) donc cas normale puisque la fréquence normal entre **60 >= fr_cardiaque <=100 battement par minute.**

Sauf l'agent contrôleur contrôler les agents patient et médecine.

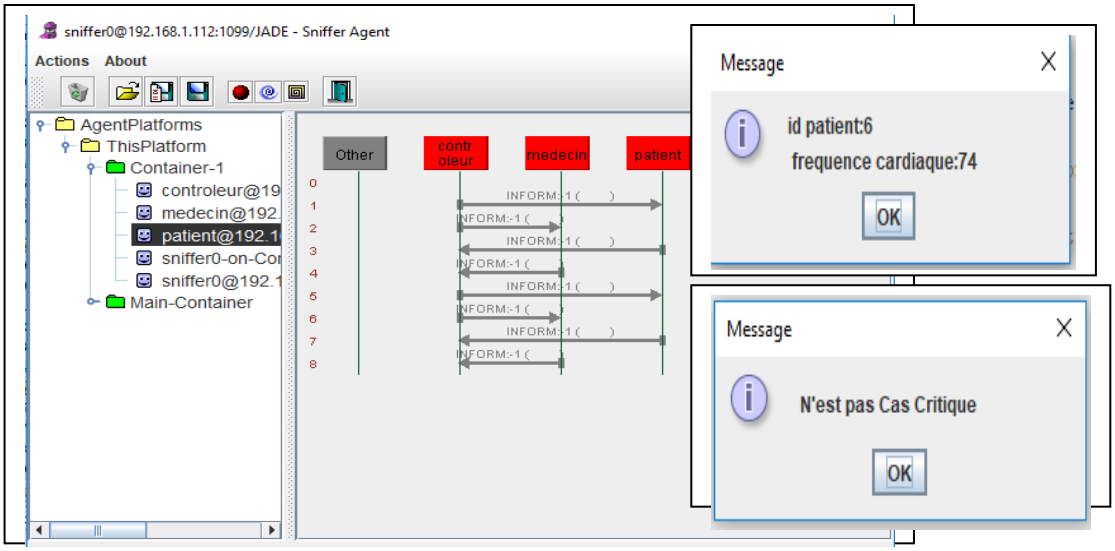


Figure 4.14 : représente le cas pas critique

- **Si un cas critique :**

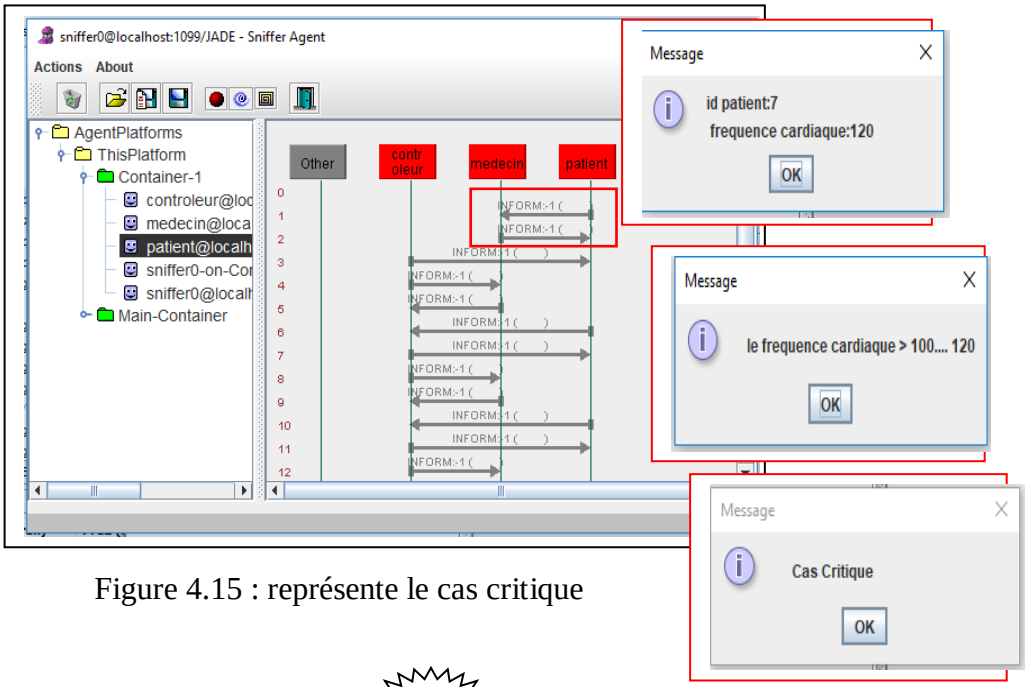


Figure 4.15 : représente le cas critique

## 2.4. Déclencher panne

Dans la figure (4.16) l'agent contrôleur contrôler les agents patient et médecin (chaque période envoyer message (**exp** : Hi..) il faut répondre (**exp** : Bien reçu..) **sinon** il ya une panne

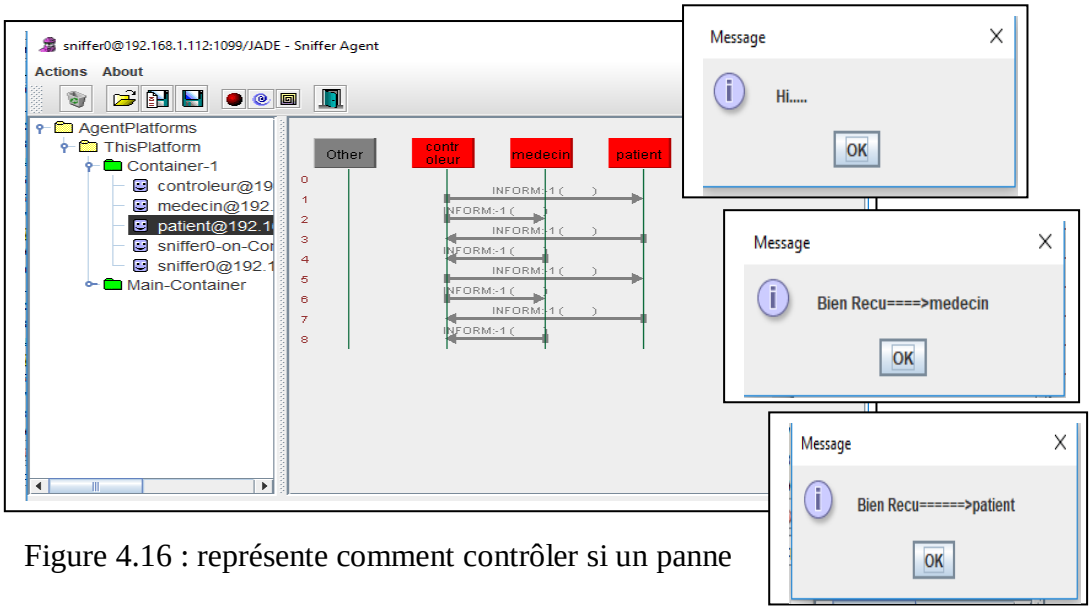


Figure 4.16 : représente comment contrôler si un panne

- Pour déclencher une panne click le bouton (Déclenche panne) comme la figure (4.17).

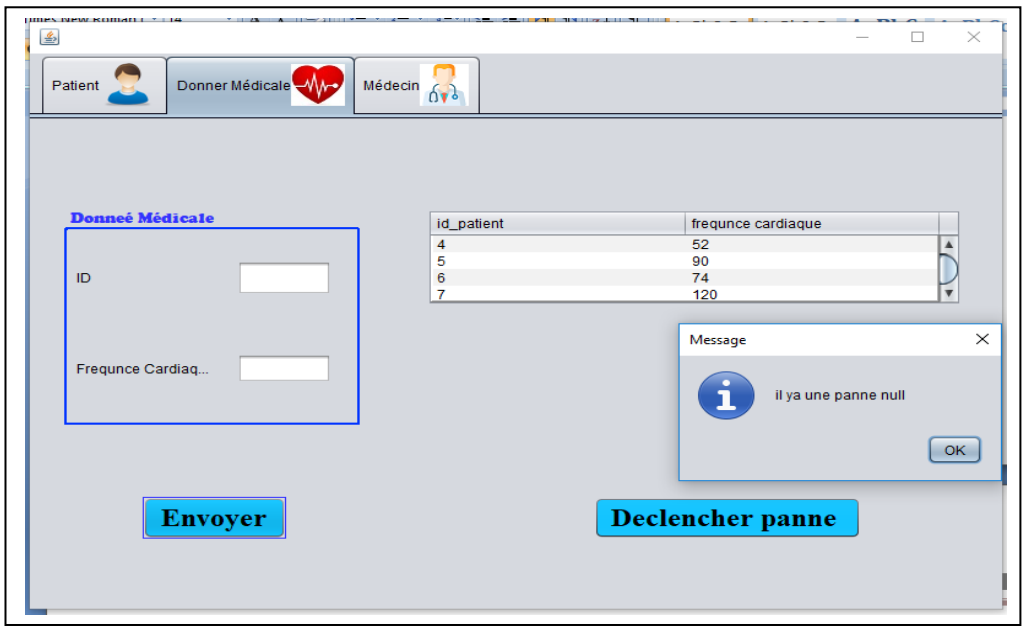


Figure 4.17 : représente le bouton déclencher panne

## 3. code source

C'est quelque classe de notre application :

### 3.1 code source de classe connecte.java

```
import java.sql.*;
public class connecter {

    Connection con;

   public connecter(){
       try{
       Class.forName("com.mysql.jdbc.Driver");
       }
       catch(ClassNotFoundException e){
                System.err.println(e);//pour afficher l'erreur
                }
       try{
           con=DriverManager.getConnection("jdbc:mysql://localhost:3306/pfe","root","");
       }catch(SQLException e){
           System.err.println(e);
       }
   }
    Connection obtenirconnexion(){return con;}


    boolean Chekuserandpass(String username, String password) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body c
    }
}
```

Figure 4.18 : représente code source de connecter avec base de donnée

## 3.2 code source de maincontraine.java

```
public class maincontainer {
    public static void main(String[] args) {
try{
jade.core.Runtime runtime=jade.core.Runtime.instance();
Properties properties=new ExtendedProperties();
properties.setProperty(Profile.GUI,"true");
ProfileImpl profileImpl=new ProfileImpl(properties);
AgentContainer mainContainer=runtime.createMainContainer(profileImpl);
mainContainer.start();
}
catch(Exception e){ e.printStackTrace(); }
}

}
```

Figure 4.19 : représente code source de classe mainecontraine

## 3.3 code source de classe PFD2 pour crée les agents

```
public class PFE2 {
 public static jade.core.Runtime runtime;
    public static Profile profile;
    public static AgentController agentController = null;
    public static ContainerController containerController;
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

try {
Runtime runtime=Runtime.instance();
ProfileImpl profileImpl=new ProfileImpl(false);
profileImpl.setParameter(ProfileImpl.MAIN_HOST,"localhost");
AgentContainer agentContainer=runtime.createAgentContainer(profileImpl);

AgentController agentController3=agentContainer.createNewAgent("controleur","pfe2.controleur", new Object[]{});
agentController3.start();
AgentController agentController=agentContainer.createNewAgent("patient","pfe2.patient", new Object[]{});
agentController.start();
//doWait(10000);
AgentController agentController1=agentContainer.createNewAgent("medecin","pfe2.medecin", new Object[]{});
agentController1.start();


} catch (ControllerException e) {
e.printStackTrace();
}
}}
```

Figure 4.20 : représente code source de classe PFD2

## 3.4 code source d'agent patient (partie1)

```
public class patient extends Agent {
    connecter conn = new connecter();
    Statement stm;
    ResultSet Rs;
    DefaultTableModel model = new DefaultTableModel();
    int id_patient;
    int fr_cardiaque;
    int i;
@Override
    protected void setup() {
        System.out.println("Agent patient ..... ");

        try {
            stm = conn.obtenirconnexion().createStatement();
            ResultSet Rs = stm.executeQuery(" Select * from donner ");

            if (Rs.last()) {
               System.out.println("id patient:" + Rs.getInt("id_patient"));
                System.out.println("frequence Cardiaque :" + Rs.getInt("fr_cardiaque"));
                 fr_cardiaque = Rs.getInt("fr_cardiaque");
                JOptionPane.showMessageDialog(null, "id patient:" + Rs.getInt("id_patient")
                        + "\n frequence cardiaque:" + Rs.getInt("fr_cardiaque"));

                if (Rs.getInt("fr_cardiaque") < 60) {
                    addBehaviour(new WakerBehaviour(this, 20000) {
                        @Override
                        protected void handleElapsedTimeout() {

                            ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
                            msg.addReceiver(new AID("medecin", AID.ISLOCALNAME));
```

Figure 4.21 : représente code source d'agent patient

### 3.5 code source de agent patient (partie 2)

```
                    ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
                    msg.addReceiver(new AID("medecin", AID.ISLOCALNAME));
                    msg.setContent(" la frequnce cardiaque < 60" + fr_cardiaque);
                    send(msg);
                }
            });
        }
        if (Rs.getInt("fr_cardiaque") >= 100) {
            addBehaviour(new WakerBehaviour(this, 30000) {
                @Override
                protected void handleElapsedTimeout() {

                    ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
                    msg.addReceiver(new AID("medecin", AID.ISLOCALNAME));
                    msg.setContent(" le frequence cardiaque > 100.... " + fr_cardiaque);
                    send(msg);
                }
            });
        }

    }
} catch (Exception e) {
    System.err.println(e);
}

//receicer de medecin
addBehaviour(new CyclicBehaviour() {
    @Override
    public void action() {
```

Figure 4.22 : représente code source d'agent patient

## 3.6 code source d'agent patient (partie3)

```
@Override
public void action() {
    ACLMessage  msg = new ACLMessage(ACLMessage.INFORM);
    msg = getAgent().receive();
    if (msg != null) {
        JOptionPane.showMessageDialog(null, "   "
                + msg.getContent());
    } else {
        block();
    }
}
});
addBehaviour(new CyclicBehaviour() {
    @Override
    public void action() {

        ACLMessage msgrecu = new ACLMessage(ACLMessage.INFORM);
         msgrecu= getAgent().receive();
          if(msgrecu!=null){
              JOptionPane.showMessageDialog(null," "
                      + msgrecu.getContent());
           }else block();} });
addBehaviour(new TickerBehaviour(this, 70000) {
                @Override
               protected void onTick() {

                    ACLMessage msgrecu = new ACLMessage(ACLMessage.INFORM);
if(msgrecu!=null)   {

                    msgrecu.addReceiver(new AID("controleur", AID.ISLOCALNAME));
                   //JOptionPane.showMessageDialog(null, "Bien Recu ");
```

2.patient > setup > TickerBehaviour > onTick

Figure 4.23 : représente code source d'agent patient

## 3.7 code source d'agent patient (partie 4)

```
addBehaviour(new TickerBehaviour(this, 70000) {
                @Override
               protected void onTick() {

                    ACLMessage msgrecu = new ACLMessage(ACLMessage.INFORM);
if(msgrecu!=null)   {

                    msgrecu.addReceiver(new AID("controleur", AID.ISLOCALNAME));
                   //JOptionPane.showMessageDialog(null, "Bien Recu ");
                   msgrecu.setContent("Bien Recu======>patient");
                    send(msgrecu);
                }}
            });


    }

    void send(String msg) {
        patient p = null;
        ACLMessage msgrecu = new ACLMessage(ACLMessage.INFORM);
        JOptionPane.showMessageDialog(null,"il ya une panne "
                + msgrecu.getContent());

            p.doActivate();

    }
}
```

Figure 4.24 : représente code source d'agent patient

## 3.8 code source d'agent médecin (partie 1)

```
public class medecin extends Agent {
 connecter conn = new connecter();
    Statement stm;
    ResultSet Rs;
    DefaultTableModel model = new DefaultTableModel();

    int id_patient;
    int fr_cardiaque;
    int i;
        @Override
        protected void setup() {
             System.out.println(" Agent Medecin..... ");
            try {
            stm = conn.obtenirconnexion().createStatement();
            ResultSet Rs = stm.executeQuery(" Select * from donner ");

            if (Rs.last()) {
                if (Rs.getInt("fr_cardiaque") > 55){

                    if(Rs.getInt("fr_cardiaque") <= 90) {
                         JOptionPane.showMessageDialog(null," N'est pas Cas Critique");

                    }
                    }
             if((Rs.getInt("fr_cardiaque") < 55)||(Rs.getInt("fr_cardiaque") >= 90))
             {

                     addBehaviour(new CyclicBehaviour() {
                @Override
```

Figure 4.25 : représente code source d'agent médecin

## 3.9 code source d'agent médecin (partie2)

```
            public void action() {
               // i=1;

                //recieve the msg
              ACLMessage msg =new ACLMessage(ACLMessage.INFORM);
                 msg=getAgent().receive();

                if(msg!=null){
                   JOptionPane.showMessageDialog(null," "
                           + msg.getContent());
                }else block();

            }

        });
        addBehaviour(new WakerBehaviour(this, 50000) {

             @Override
        protected void handleElapsedTimeout() {

            ACLMessage msg=new ACLMessage(ACLMessage.INFORM);
             msg.addReceiver(new AID("patient",AID.ISLOCALNAME));
            msg.setContent(" Cas Critique ");
            send(msg);
        }
        });
```

Figure 4.26 : représente code source d'agent médecin

## 3.10 code source d'agent médecin (partie3)

```
        addBehaviour(new CyclicBehaviour() {
            @Override
            public void action() {

                ACLMessage msgrecu = new ACLMessage(ACLMessage.INFORM);
                msgrecu= getAgent().receive();
                if(msgrecu!=null){
                      JOptionPane.showMessageDialog(null," "
                              + msgrecu.getContent());

                   }else block();


            } });
             addBehaviour(new TickerBehaviour(this, 80000) {
                        @Override
                       protected void onTick() {
                            // System.out.println("Agent Patient ");
                            ACLMessage msgrecu = new ACLMessage(ACLMessage.INFORM);
if(msgrecu!=null)  {

                            msgrecu.addReceiver(new AID("controleur", AID.ISLOCALNAME));
                           // JOptionPane.showMessageDialog(null, "Bien Recu ");
                           msgrecu.setContent("Bien Recu====>medecin");
                            send(msgrecu);
                       }}
                    });
}}
```

Figure 4.27 : représente code source d'agent médecin

## 3.10 code source d'agent contrôleur

```
public class controleur extends Agent {
    @Override
    protected void setup(){
         System.out.println(" Agent controleur..... ");
    addBehaviour(new TickerBehaviour(this, 60000) {
        //private int compteur=1;
      protected void onTick() {
          ACLMessage msgrecu = new ACLMessage(ACLMessage.INFORM);
                            msgrecu.addReceiver(new AID("patient", AID.ISLOCALNAME));
                            msgrecu.addReceiver(new AID("medecin", AID.ISLOCALNAME));
                            msgrecu.setContent(" Hi..... ");
                           //  JOptionPane.showMessageDialog(null , " Hi... ");
                            send(msgrecu);}
    });
         addBehaviour(new CyclicBehaviour() {
            @Override
            public void action() {

                ACLMessage msgrecu = new ACLMessage(ACLMessage.INFORM);
                msgrecu= getAgent().receive();
                if(msgrecu!=null){
                      JOptionPane.showMessageDialog(null," "
                              + msgrecu.getContent());

                   }else block();
            } });

    }}
```

Figure 4.28 : représente code source d'agent contrôleur

## Conclusion

Dans ce chapitre nous avons présenté notre simulation par la présentation de ses principales interfaces graphiques. Enfin grâce à cette simulation nous avons pu montrer l'utilité de notre proposition.

# Conclusion générale

L’expansion d’internet des objets(IdO) n’a cessé de s’accroître ces dernières années. Ceci a permis de généraliser son utilisation dans différents domaines applicatifs (médicales, environnementales, militaires, etc.). Malgré la diversité de ces applications et leurs exigences (nature du trafic, contraintes de débit, de délai,...), elles requièrent toutes un degré de tolérance aux pannes qui se varie selon la criticité de l’application afin de garantir son bon fonctionnement. Ainsi, plusieurs solutions et mécanismes ont été proposés dans la littérature pour assurer le recouvrement de pannes dans les systèmes IdO. Dans le chapitre 1, nous avons présenté une description détaillée sur la technologie d’internet des objets, son architecture, ses protocole..etc.

Nous avons vu aussi, dans le chapitre 2 le traitement des fautes dans les systèmes IdO médicaux. Nous avons présenté par la suite les différents aspects théoriques de notre proposition notamment les étapes de conception et de réalisation de notre simulateur a base le système multi agent. Afin de modéliser notre proposition, nous avons commencé par la conception de notre système en utilisant le formalisme AUML, et pour la réalisation il y a plusieurs technologies qui ont été nécessaires. Ces dernières sont présentées dans le dernier chapitre on citera donc le langage de programmation java pour la réalisation des interfaces ainsi que la programmation des agents. Ces derniers sont manipulés en exploitant la plate-forme JADE qui permet de construire des systèmes multi agents (SMA). Nous avons aussi utilisés MySQL pour l'élaboration des requêtes d'interrogation de la base de donnée, et enfin l’environnement de développement NetBeans IDE pour l'écriture du code.

A cause de manque de temps et de matériels, nous n'avons pas pu réellement mettre en œuvre ce projet.

# Bibliographie

[1] (Alok Kulkarni,2014) Alok Kulkarni, Sampada Sathe (2014). Healthcare applications of the Internet of Things: A Review. (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (5) , 2014, 6229-6232

[2] (André Pedroza,2016)André Pedroza dos Santos, Dalfrede Welkener Soares Lima, Fhelipe Silva Freitas, Geiziany Mendes da Silva CESAR . 2016. A Motivational Study Regarding IoT and Middleware for Health Systems. The Tenth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies

[3] (Zeinab Kamal elddin,2017) Zeinab Kamal Aldein Mohammed, Elmustafa Sayed Ali Ahmed.(2017). Internet of Things Applications, Challenges and Related Future Technologies. WSN 67(2) 126-148

[4] Hend Ben Hadji , Ph'D, KAIST (Corée du Sud) , Directrice au Centre d'Etudes et de Recherche des Télécommunications(CERT), Tunisie, Hend.benhadji@tunsia.gov.tn

[5] Rafael Enrique Angarita Arocha. Une approche pour les services composites transactionnels auto réparateurs. Autre [cs.OH]. Universit? Paris Dauphine - Paris IX, 2015. Anglais. <NNT: 2015PA090051>.
<tel-01281384>

[6] Debanjan Ghosh, Raj Sharman, H. Raghav Rao et
Shambhu Upad-hyaya. Systèmes d'auto-guérison - Enquête et synthèse. Decis. Support Syst.,42 (4): 2164-2185, janvier 2007.

[7] X. Besseron. Tolérance aux fautes et reconfiguration dynamique pour les applications distribuées a grande échelle. PhD thesis, Universit´e de Grenoble, France, 2010. (Cite pages21,23et30.

[8] (Zilic, 2011) Zeljko Zilic and Katarzyna Radecka. 2011. Fault Tolerant Glucose Sensor Readout and Recalibration. Wireless Health'11, October 10-13, 2011, San Diego, USA. ACM 978-1-4503-0982-0.

[9] : FERBER J., Les Systèmes Multi-Agent - Vers une Intelligence Collective, InterEditions, 1995.
[10] : René Mandiau, Emmanuelle Grisling Lestrugeon. « Systèmes Multi-agents». Techniques de l'ingénieur, traité Informatique Industrielle S7216. 2002.

[11] : F.Vernadat and P.Azéma. prototypage de système d'agents communicants.In journée système multi-agents PRC-GDR intelligence artificielle .Nancy.decembre 1992.

[12] J. Ferber, Les systèmes multi-agents. Vers une intelligence collective, Paris: InterEditions, 1995.

[13] J. Ferber, Les systèmes multi-agents. Vers une intelligence collective, Paris: InterEditions, 1995.
[14] S. J. Russell et P. Norvig, Artificial Intelligence. A Modern Approach, Prentice- Hall, Englewood Cliffs, 1995.

[15] J. Ferber, «Les systemes multi-agents : un aperçu general,» Technique Et Science Informatiques - TSI, 1997.

[16] http://dx.doi.org/10.1016/j.future.2017.04.004

[17] https://www.techno-science.net/definition/5346.html

[18] les experts Ooreka, "Système RFID : définition et fonctionnement d'un système RFID ", 22 déc. 2015; http://rfid.comprendrechoisir.com/comprendre/systeme-rfid.

[19] J. A. Stankovic, "Wireless sensor networks," *IEEE Computer Society,* vol. 41, no. 10, pp. 92-95, 2008.

[20] N. Daniel, R. Marcel, and K. Daniel, *Livre blanc Machine To Machine enjeux et perspectives*: Orange Business Services, Syntec informatique, Fing, 2006, 40 p.

[21] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems,* vol. 29, no. 7, pp. 1645-1660, 2013.

[22] Munindar P. Singh Department of Computer Science North Carolina State University Raleigh, NC 27695, USA singh@ncsu.edu & Amit K. Chopra School of Computing and Communications Lancaster University Lancaster LA1 4WA, UK amit.chopra@lancaster.ac.uk (The Internet of Things and Multiagent Systems: Decentralized Intelligence in Distributed Computing).

[23] Sofia KOUAH RELA(CS)2 Laboratory, University of Larbi Ben M'Hidi, Oum El Bouaghi, Algeria kouahs@yahoo.fr & Ilham KITOUNI MISC Laboratory, Constantine 2- Abdelhamid Mehri University- Algeria ilham.kitouni@univ constantine2.dz. (Internet of Things Agents Diagnosis Architecture: Application to Healthcare IoT System)