PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR
SEDE ESMERALDAS

ESCUELA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

TESIS DE GRADO

EVALUACIÓN DEL USO DE JAVA Y PYTHON EN EL DESARROLLO DE ECOSISTEMAS DE INTERNET DE LAS COSAS DIRIGIDOS POR AGENTES INTELIGENTES

PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS Y COMPUTACIÓN

AUTOR (A):
CARLOS AYRTON REYNA MERINO

ASESOR (A):
PhD. PABLO PICO VALENCIA

ESMERALDAS – 2020

TRIBUNAL DE GRADUACIÓN

Trabajo de tesis aprobado luego de haber dado cumplimiento a los requisitos exigidos por el reglamento de Grado de la PUCESE previo a la obtención del título de INGENIERO DE SISTEMAS Y COMPUTACIÓN.

.....

PhD. Pablo Pico Valencia

Tutor de Tesis

.....

Mgt. Susana Patiño Rosado

Lector 1

.....

Mgt. Jaime Sayago Heredia

Lector 2

.....

Mgt. Susana Patiño Rosado

Coordinadora de la Escuela de Sistemas y Computación

AUTORÍA

Yo, CARLOS AYRTON REYNA MERINO portador de la cédula de identidad No. 0802795773 declaro que los resultados obtenidos en la investigación que presento como informe final, previo a la obtención del título de "Ingeniero de Sistemas y Computación" son absolutamente originales, auténticos y personales.

En tal virtud, declaro que el contenido, las conclusiones y los efectos legales y académicos que se desprenden del trabajo propuesto de investigación y luego de la redacción de este documento son y serán de mi sola, exclusiva responsabilidad legal y académica.

.....

Carlos Ayrton Reyna Merino

C.I. 080279577-3

AGRADECIMIENTOS

Agradezco en primer lugar a Dios por darme una familia que siempre estuvo pendiente de mí, apoyándome cada semestre y no dejándome tirar la toalla para cumplir la meta de ser un profesional.

A mi madre, Julen Merino Córdova, por motivarme y estar siempre pendiente del día a día en mis estudios y por apoyarme de forma incondicional.

A mi padre, Jorge Reyna Reyna, por haberme apoyado desde el principio y ayudándome constantemente a ser una mejor persona y profesional.

A mi hermano, Aaron Reyna Merino, que, a pesar de la diferencia de edad, siempre ha estado apoyándome y ayudándome en lo que más pueda.

A mi esposa e hijo, que son lo más importante de mi vida y mi motivación para salir adelante, apoyándome y dándome consejos a diario.

Y al resto de mi familia, que muchas veces me han brindado consejos de motivación para salir adelante y no persistir en la meta final.

DEDICATORIA

El trabajo de investigación está dedicado a mi esposa e hijo, que siempre han estado motivándome y apoyándome en toda situación; a mi padre, madre y hermano quienes siempre desde el principio me apoyaron y vivieron el día a día conmigo.

RESUMEN

La presente investigación se llevó a cabo con la finalidad de evaluar el uso del lenguaje de programación Java y Python dentro del desarrollo de ecosistemas del Internet de las Cosas (IoT) dirigidos por agentes inteligentes, en el campo de la domótica dentro de un hogar digital. Para llevar a cabo la investigación se realizó una comparación entre los dos lenguajes mencionados anteriormente, conocidos en el ámbito de los sistemas multi agentes como JADE (del inglés Java Agent Development Framework) y SPADE (del inglés Smart Python multi-Agent Development Environment).

Para el desarrollo de esta investigación se empleó la plataforma OpenHAB, cuyo software es de código abierto, diseñada para integrar el sistema de automatización de hogares. Esta plataforma es una solución establecida en Java que brinda una interfaz homogénea y amigable para el usuario, teniendo un carácter centralizado donde se basa en reglas de control de automatización para todo el sistema. Este sistema del hogar digital modelado consideró tres aspectos tales como la gestión energética (control de persianas, iluminación), la climatización (control del aire acondicionado y calefacción) y por último el aspecto de seguridad (control de alerta en puertas, persianas e iluminación).

El sistema de control del hogar digital implementado en OpenHAB fue controlado mediante agentes JADE y SPADE. Dichos agentes se organizaron en sistemas multiagentes que se ejecutaron en dos tipos de entornos tecnológicos, ordenador personal y componente de placa único como es Raspberry Pi. Se realizó una comparativa de ambos sistemas multi agentes determinando que la implementación del sistema multi agente en Java tiene menor grado de dificultad al implementar el sistema, y en cuanto a la comunicación con la plataforma OpenHAB, que realizarlo en Python ya que existen trabajos relacionados e información relevante sobre el tema, obteniendo un mejor resultado en el IDE Apache Netbeans (Java). Sin embargo, en cuanto a rendimiento del sistema, se obtuvieron resultados similares en el momento de realizar una evaluación de los procesos ejecutados por los agentes de confort, tanto en JADE como en SPADE, dando como resultados unos tiempos de ejecución parecidos por cada agente evaluado.

Palabras Clave: JADE, SPADE, domótica, OpenHAB, hogar inteligente, Internet de las Cosas, agentes software, sistemas multiagentes.

ABSTRACT

The present research was carried out with the purpose of evaluating the use of the Java and Python programming language within the development of Internet of Things ecosystems directed by intelligent agents, in the field of home automation within a home. To carry out the research, a comparison was made between the two languages mentioned above, known in the field of multi-agent systems such as JADE and SPADE.

For the implementation of this research, the OpenHab platform was used, whose software is open source, designed to integrate the home automation system. This platform is a solution established in Java that provides a homogeneous and user-friendly interface, having a centralized character where it is based on automation control rules for the entire system. This control system is based on three aspects that are energy management, such as the control of blinds, lighting, etc.; the air conditioning where the air conditioning and heating is located; and finally the security aspect such as alert systems on doors, blinds and lighting.

A comparison was made when executing the OpenHab home automation control system both in the NetBeans (JADE) development environment, and in a Raspberry Pi. A comparison of both multi-agent systems was made, determining that the implementation of the multi-agent system in Java has less difficulty than doing it in Python since there are related works and relevant information on the subject.

Keywords: JADE, SPADE, home automation, OpenHAB, smart home, Internet de Things, software agents, multiagent systems

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Presentación del tema

La presente investigación tiene como finalidad el análisis comparativo entre los marcos de trabajo *Java Agent DEvelopment Framework* (JADE) y *Smart Python Agent DEvelopment* (SPADE). Dicho análisis consiste en la evaluación de la eficiencia y las capacidades de las dos herramientas, ampliamente usadas para el desarrollo de sistemas multiagentes, para llevar a cabo el control proactivo e inteligente de un ecosistema basado en tecnologías de Internet de las Cosas (IoT por sus siglas en inglés) dirigido por agentes.

Los agentes inteligentes son entidades autónomas que de manera agrupada forman un sistema multi agente. Gracias a las características como autonomía, proactividad, inteligencia y colaboración, inherente a los sistemas multiagentes, en los últimos años se ha extendido su aplicación al terreno del IoT. En este sentido, en la actualidad, este tipo de entidades se ha aprovechado en ecosistemas de IoT para modelar, programar y simular sistemas que optimizan los recursos de los objetos de IoT, así como la ejecución de tareas inteligentes que los objetos de IoT no pueden ejecutar por sí solos. Así, un nuevo paradigma denominado Internet de los Agentes (IoA, por sus siglas en inglés) ha surgido en los últimos años.

Planteamiento del problema

El número de dispositivos conectados a Internet que está en manos del consumidor se ha disparado, y cada vez son más los dispositivos, objetos y cosas que están conectados a la red gracias a pequeños sensores y actuadores ubicuos que se encargan de permitir esa conexión y de enviar y recibir datos. Sin embargo, muchos de los objetos de IoT que son comercializados como objetos inteligentes no poseen dicha capacidad de manera inherente. Es para solventar esta limitación de inteligencia que los sistemas multiagentes se han empleado para poder integrar un componente de "inteligencia" a estos dispositivos conectados de manera que puedan ejecutar acciones de manera autónoma (i.e., ejecutar una acción en función de cambios que acontezcan en el entorno), inteligente (i.e., ejecutar una tarea considerando el ahorro energético de los objetos de IoT cuidando proveer confort para los usuarios) y colaborativa (i.e., compartir los datos capturados del entorno

para llevar a cabo acciones de manera más precisa). De esta manera, la nueva generación de objetos de IoT pueden desempeñarse por sí solos y ser menos dependientes del ser humano, un valor añadido a los consumidores de IoT y a los desarrolladores de este tipo de sistemas.

En los últimos años los sistemas basados en agentes han recibido una gran atención por parte del mundo de la industria, empresa, y la academia. El paradigma orientado, integrado al IoT también ha interesado a la comunidad científica. Sin embargo, los avances tecnológicos han introducido nuevas tecnologías para el desarrollo de agentes software para entornos Java y Python. Por ello, es importante conocer las prestaciones de los marcos de trabajo compatibles con dichas herramientas, como son JADE y SPADE, para determinar la mejor opción para el desarrollo de sistemas de IoT dirigidos por agentes que se ejecutan en ordenadores personales y sistemas embebidos.

Justificación

La necesidad de optimizar los procesos, mejorar la comunicación, destacar de la competencia e incluso repensar los modelos comerciales ha hecho que las empresas consideren la tecnología (IoT) como el principal recurso para alcanzar estos objetivos.

El IoT aumenta y fomenta la comunicación entre máquinas y debido a esta ingeniosa innovación, los dispositivos físicos permanecen en contacto unos con otros, lo que requiere una mayor eficiencia y calidad en la realización de procesos de monitorización en entornos como ciudades inteligentes, industria 4.0, hogar inteligente, cuidado de la salud, entre otros. Por ello, es importante integrar tecnologías como son los agentes y sistemas multiagentes en IoT para optimizar los procesos y ejecutar acciones de manera semejante a como lo haría un ser humano. Ello requiere también que se definan las herramientas con mejor rendimiento y que se adapten a las actuales tecnologías usadas para implementar la IA como son Java y Python.

El uso de la adecuada herramienta para el desarrollo de sistemas multiagentes para el control de IoT es importante. Ello ha motivado para aplicar dos de las principales herramientas para el desarrollo de sistemas multiagentes, JADE y SPADE, para controlar el IoT en entornos sofisticados y restringidos tecnológicamente. Ello permitirá un mejor control del IoT tanto para aplicaciones personales (i.e., monitorización de condiciones

fisiológicas, control inteligente de los artefactos del hogar) y a nivel organizacional y profesional (i.e., confort en la oficina, monitorización de procesos automatizados).

Objetivos

Objetivo General

Comparar el rendimiento de los agentes desarrollados con los marcos de trabajo JADE (*Java Agent DEvelopment Framework*) y SPADE (*Smart Python Agent Development*) operativos en entornos de Internet de las Cosas.

Objetivos Específicos

1. Crear un ecosistema del Internet de las Cosas mediante la herramienta OpenHab.
2. Desarrollar un sistema multi-agente para la gestión inteligente de los recursos de ecosistemas de Internet de las Cosas
3. Definir las principales métricas para evaluar el performance y la seguridad de los agentes JADE/SPADE.
4. Implementar un ecosistema de Internet de las Cosas dirigido por agentes en un ordenador personal y en uno mono placa**.**

CAPÍTULO 1: MARCO TEÓRICO

1.1. Antecedentes

En esta sección se presentan varios trabajos de investigación relacionados rigurosamente al tema del análisis comparativo entre JADE y SPADE. Estas dos plataformas de desarrollo de agentes son las que han dado un aporte significativo a la investigación.

Para entender mejor el comportamiento del rendimiento de los sistemas informáticos es esencial emplear métricas simples que resalten aspectos específicos de rendimientos o cuellos de botella particulares, así como también se debe diseñar varios puntos de referencia simples, portátiles y escalables. En este sentido, se recuperaron tres propuestas relevantes que son descritas a continuación.

En la investigación realizada por Abbas et al. [1], se presenta una técnica de organización dinámica transparente que tiene como objetivos agregar eficientemente múltiples plataformas JADE. Se realizó una evaluación de rendimiento entre JADE y otras técnicas proporcionadas por la misma herramienta, demostrando que la técnica propuesta mejoró el rendimiento aproximadamente del 50%. Es por ello, que se recomienda su implementación para el desarrollo de sistemas multi agentes a gran escala.

Por otro lado, Xu Baiquan [2], presenta un modelo de arquitectura de la plataforma para realizar pruebas de rendimiento basado de JADE, que es un entorno de desarrollo de múltiples agentes que soporta la gestión y el control de comunicaciones para agentes. En el artículo se describe JADE como herramienta de desarrollo de agentes y se presenta el método a seguir para diseñar una plataforma para pruebas de rendimiento basadas en JADE. Siguiendo el diseño de la plataforma, se puede afirmar que ésta tiene ventaja al ser una implementación flexible multiplataforma, donde su estructura ayuda a los desarrolladores en herramientas de prueba de rendimiento.

El uso de arquitecturas multi agentes es muy importante también en entornos de sistemas de distribución de energía donde se han implementado varios dispositivos inteligentes. En [3], se emplea un marco de trabajo distinto a JADE, esto es un marco

de desarrollo denominado SPADE cuyo objetivo es soportar procesos de restauración posterior de fallas de equipos en sistemas de distribución de energía usando sistemas multiagentes. Los resultados indicaron que mediante PADE se puede simular el proceso de restauración de manera exitosa.

En lo relacionado a la comunicación entre agentes, la investigación realizada por Ribeiro, Rocha y Barata [4], realiza un análisis comparativo de rendimiento basado en el RTT (Tiempo de ida y vuelta) de los mensajes, de los patrones de codificación. En la investigación la mayoría de las pruebas realizadas se enfocaron en la evaluación del RTT en el momento que se agregan pares de agentes comunicativos a la plataforma, centrándose en la eficiencia mostrada por cada método, ya que el agente capta mensajes continuamente. La plataforma puede ser evaluada desde algunos puntos vista, es por eso que los sistemas basados en múltiples agentes normalmente dependen de intercambio de mensajes para poder implementar su comportamiento colectivo. Algunos investigadores evaluaron el uso de JADE dentro de este contexto, dando como resultado que esta plataforma lleve a cabo la comunicación local y que el intercambio de mensajes sea más rápido. Esto implica entonces un valor de RTT bajo.

1.2. Bases conceptuales

A través de una revisión de la literatura, se plantearon también las bases teóricas-conceptuales que fundamentan la investigación. En términos generales, las bases conceptuales contemplan la descripción detallada del paradigma relacionado con el IoT y los sistemas multiagentes basados en JADE y SPADE.

1.2.1. Internet de las Cosas

El Internet de las Cosas (*Internet of Things*, IoT por sus siglas en inglés), evoluciona constantemente con el paso de los años, acorde a como las tecnologías de las Telecomunicaciones lo hacen. Cada vez son más los objetos inteligentes que nos rodean en nuestra vida cotidiana o laboral.

El IoT es un modelo tecnológico el cual ha sido pensado como una red de tipo global, la cual incluye máquinas y dispositivos que tienen la facultad de comunicarse el uno con el otro [5]. Una definición más formal, determina que el Internet de las Cosas es una gran red de abundantes objetos físicos que poseen una conexión a Internet y que obtienen información alrededor del medio ambiente donde se encuentran para comunicarse entre sí mediante el Internet, produciendo así una gran cantidad de datos útiles para utilizar servicios dependientes [6].

Cuando se menciona al Internet de las Cosas, se lo reconoce como uno de los más importantes y relevantes paradigmas dentro de una tecnología futura, así como en el presente. Dicho paradigma emergente está llamando el interés de varios campos como lo son la industria, comercio, salud, hogar, entre otros. Todos ellos enfocados en aprovechar las prestaciones del IoT en distintos términos para aumentar sus ingresos o para convertirse en líderes en sus mercados.

1.2.1.1. Arquitectura

En torno al Internet de las Cosas se han propuesto varias arquitecturas. No obstante, una de las arquitecturas genéricas más populares constituye un modelo basado en tres capas: percepción, red y aplicación. Dicha arquitectura es ilustrada en la Figura 1.

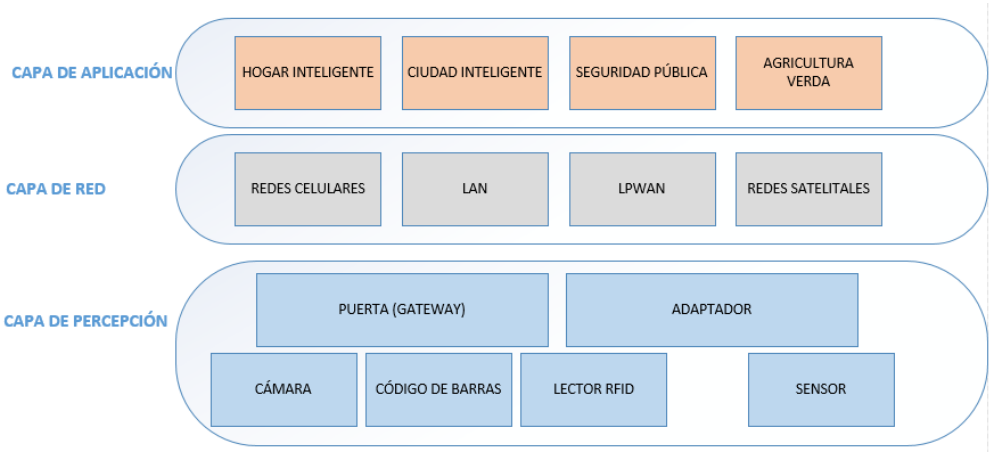

Figura 1 Arquitectura genérica de IoT[8].

En primer lugar, la capa de percepción es aquella capa que interactúa con los dispositivos y componentes físicos por medio de todo tipo de dispositivos inteligentes

como sensores, actuadores, RFID, etc. Su característica principal es calcular, recopilar y procesar la información de estos elementos por medio de los dispositivos inteligentes implementados, los cuales son los encargados de enviar la información previamente procesada a la capa superior mediante interfaces de capas [7].

Por otro lado, la capa de **red o de transmisión** es la encargada de tomar la información procesada por la capa de percepción para establecer las rutas de transmisión de los datos y de información al IoT *hub*, dispositivos y aplicaciones por medio de redes integradas. Esta es la capa con mayor grado de importancia ya que dentro de ella están integradas algunas tecnologías de comunicación inalámbrica (i.e.,, Bluetooth Low-Energy (BLE), WiFi, Near-Field Communication, Zigbee, Z-Wave, LoRa / LoRaWAN [8]), así como algunos dispositivos (i.e., hub, switch, computación en la nube) [7].

Finalmente, la capa de **aplicación** es aquella que tiene como objetivo la captación de los datos transmitidos desde la capa de red para conceder servicios u operaciones necesarias, y se definen los protocolos que utilizan las aplicaciones para intercambiar datos, gestores de base de datos y protocolos de transferencia de archivos. Estos servicios son concedidos a los usuarios que por lo general no interactúan directamente con el nivel de aplicación [7].

1.2.1.2. Aspectos importantes

Existen una serie de requisitos a tener en cuenta dentro de la elaboración de una arquitectura de IoT. Entre los más relevantes figuran las siguientes características según el estándar de calidad ISO/IEC 25000:

- Usabilidad que incluye características como capacidad de aprendizaje, capacidad de ser usado, protección contra errores de usuario, facilidad de aprendizaje, etc.
- Fiabilidad que es la madurez, tolerancia a defectos, donde el software-sistema debe conservar su capacidad-funcionalidad durante un largo periodo de tiempo.
- Seguridad que es no más que la protección de información y datos para personas no autorizadas.

- La calidad de servicio, la cual constituye uno de los requisitos más importantes dentro de una arquitectura. Dentro de la calidad es de gran importancia definir un modelo que permita ejecutar una evaluación detallada con una secuencia específica, que por consecuente estructure puntos a evaluar [9].

1.2.1.3. Seguridad de IoT

La seguridad es uno de los principales inconvenientes del IoT, ya que, si por alguna u otra razón los datos de una persona se ven comprometidos alguna vez, esto puede conllevar a la desconfianza del usuario en IoT. Es por ello, que la seguridad, debe ser prioritaria al momento de diseñar una arquitectura para el IoT. Dentro de la seguridad de IoT hay que destacar requisitos claves como son la autenticación y la confidencialidad, donde son importantes para poder garantizar una comunicación segura ya que IoT transfiere constantemente datos personales susceptibles, entre cosas y usuarios para alcanzar objetivos[10].

Dentro del estándar de calidad ISO/IEC 25000, el cual busca la ejecución de un marco de trabajo común para la evaluación de calidad del producto de software, está incluida la característica de seguridad que no es más que la facultad de proteger información y datos de tal forma que sean incapaces de ser leídos o modificados por otras personas o sistemas no autorizados, dentro de la cual la autenticidad, la confidencialidad y la responsabilidad son sus principales características.

1.2.1.4. Protocolos de comunicación

Las tecnologías de comunicación IoT tienen la capacidad de comunicar múltiples objetos para ofrecer servicios inteligentes determinados. Comúnmente, los nodos de IoT deben realizarse con un bajo consumo de energía en concurrencia de vínculos de comunicación ruidosos. A continuación, se describen 3 tipos de protocolos:

- **Protocolo de aplicación restringido (CoAP).** Este protocolo es implementado en IoT. Es un protocolo de transferencia web que se apoya en la transferencia de estado representación, más conocida como REST a través de las funcionalidades HTTP. REST permite establecer una conexión en caché

que se fundamenta en una arquitectura cliente-servidor. CoAP tiene como meta brindar ciertas habilidades de computación y comunicación a dispositivos pequeños para realizar las interacciones RESTful [11].

- **Transporte de telemetría de la cola de mensajes (MQTT).** Dicho protocolo de mensajería tiene como prioridad dar conexión a dispositivos y redes integradas con aplicaciones y middleware. Esta conexión que se establece mediante un mecanismo de enrutamiento lo cual habilita a MQTT a ser un protocolo de conexión adecuado para IoT. El protocolo también implementa un patrón que permite una flexibilidad de transición y simplicidad de implementación [11].
- **Protocolo extensible de mensajería y presencia (XMPP).** XMPP es catalogado como un estándar de mensajería instantánea que se implementa en cuanto se quieren realizar, por ejemplo, llamadas de voz y video, llamadas entre varios usuarios y telepresencia. Este protocolo es de carácter descentralizado, seguro y libre de *spam*. Esto permite a los usuarios interactuar entre sí a través del envío de mensajes instantáneos mediante Internet y sin importar que sistema operativo tenga el usuario [11]. Implementando este protocolo, las aplicaciones de mensajería instantánea son capaces de tener un control de acceso, controlar la medición de la privacidad, proporcionar un cifrado de extremo a extremo y brindar una compatibilidad con otros tipos de protocolos.

1.2.1.5. Estándares de comunicación

Las tecnologías de comunicación del IoT tienen la capacidad de conectar objetos heterogéneos para brindar una serie de servicios inteligentes. Para brindar estos servicios, los dispositivos de IoT no solo requieren capturar datos, sino que los deben comunicar o transmitir a otros dispositivos o a la nube. Los estándares de comunicación más conocido son Bluetooth, WiFi, IEEE 802.15.4, Z-Wave, LoRa / LoRaWAN y LTE-Advanced.

- **Bluetooth Low-Energy (BLE)**. Se utiliza en dispositivos con ciertas limitaciones de energía, esto es, en sensores o controles inalámbricos. La implementación en estos dispositivos requiere de un bajo consumo de energía

ya que la cantidad de transmisión de datos es pequeña y la comunicación se realiza con poca frecuencia [12].

- **WiFi**. Es un conjunto de estándares de comunicación de red local de área inalámbrica (WLAN), proporcionando velocidades de datos y rangos de comunicación de distintas medidas [13]. Generalmente es usado por dispositivos móviles y sensores, así como en escenarios densos como estadios deportivos y aeropuertos con el objetivo de mejorar la calidad del servicio.
- **IEEE 802.15.4.** Delimita una capa física y un control de acceso al medio para redes inalámbricas de baja potencia que direccionan a conexiones seguras y escalables [11]. Generalmente es usado por sensores, en líneas telefónicas, módems por cable y líneas de transmisión de energía eléctrica. También está presente en aplicación como la automatización del hogar y aplicaciones de seguridad ya que estas no manejan protocolos muy pesados.
- **Near Field Communication (NFC)**. Es un tecnología emergente sin contacto que está basada en el estándar RFID, ISO/IEC 14443 y se conecta dentro de un reducido alcance para habilitar el cambio de datos entre dispositivos en una distancia corta [14].
- **LORA**. Tiene como significado comunicación de largo alcance. Este protocolo es bastante utilizado en sistemas inalámbricos con el objetivo esencial de centrarse en IoT, así como en la comunicación bidireccional. LORA se caracteriza por tener un largo alcance en su comunicación siendo probado con una línea de visión de hasta 10 km. Una de sus ventajas es que consume muy poca potencia, y su rango de frecuencia es libre, es decir, el costo de comunicación es menor. Este protocolo proporciona tres clases para la comunicación donde la primera clase es para comunicación bidireccional con una transmisión de enlace ascendente, la segunda clase consta de las mismas características de la primera con la diferencia que proporciona una transmisión de enlace de carácter descendente, y la tercera se caracteriza por tener ventanas abiertas continuamente excepto durante la transmisión [15].

1.2.1.6. Aplicaciones

Al hacer referencia a las aplicaciones del IoT se está involucrando un espectro muy amplio. Aunque como se analiza en Borgia [16], existen un sinnúmero de aplicaciones en las que el IoT puede coadyuvar, a continuación se describe cómo aplicar esta tecnología en 7 campos y casos particulares.

- **Infraestructura inteligente.** La implementación de objetos inteligentes dentro de una infraestructura conlleva a mejorar la flexibilidad, confiabilidad y eficiencia en la operación de infraestructura, así como abaratar costos y mejorar en la seguridad. Dentro de este grupo se incluye el control de equipos domésticos, así como por ejemplo un hogar inteligente, o una iluminación inteligente, etc. [17]. En este campo, las tecnologías de IoT tienen múltiples beneficios, como, por ejemplo, hacer que las ciudades sean más eficientes de manera que los servicios ofrecidos a los ciudadanos y visitantes ayuden a mejorar su nivel de calidad de vida.
- **Cuidado de la salud inteligente**. Dentro de este campo, el objetivo es lograr una automatización de algunas tareas básicas realizadas por el humano. Una aplicación sería la implementación de sensores en el equipo de monitoreo de salud que usan los pacientes, así como poder llevar un monitoreo y del tratamiento que debe seguir cada paciente para poder evaluar posibles riesgos de nuevos medicamentos [18].
- **Transporte inteligente**. En el área del transporte inteligente, el objetivo es conseguir que los transportes sean seguros, confiables y eficientes. Dentro de este tipo de sistemas, un gran número de vehículos se encuentran conectados entre sí a través de redes inalámbricas. Por ejemplo, dentro de esta rama nos encontramos con vehículos que pueden detectar objetos alrededor, así como administrar la velocidad sin que el humano intervenga. Cada vehículo inteligente se implementa con una determinada unidad de control electrónico para poder monitorear y tener un control de sus subsistemas. Asimismo, se implanta en los vehículos interfaces de comunicación para establecer una conexión con la red externa [7].
- **Cadenas de suministro/ logística**. Las redes de sensores ya hace algún tiempo vienen teniendo un papel importante dentro de la gestión de cadenas de suministro. Se han estado utilizando durante mucho tiempo en instalaciones

de fabricación, así como en líneas de montaje, etc. De esta manera, la implementación del IoT dentro de la logística permite aumentar la eficiencia de una cadena de suministro, así como mejorar la logística, ya que brinda información más detallada y actualizada [18].

- **Objetos perdidos**. Dentro de este campo, se menciona a una aplicación basada en la web la cual es un motor de búsqueda que pretende que los usuarios vean la última localización registrada de cualquier objeto etiquetado. La aplicación se vale de los eventos definidos por el usuario para alertar cuando la última ubicación registrada de un objeto coincida con alguna condición [19].
- **Operaciones militares.** Los campos de guerra actualmente están teniendo un fuerte vínculo con la red y con la digitalización. Por tanto, se espera que un futuro no muy lejano el IoT invada dominantemente en la guerra. Se calcula y se espera que en guerras futuras haya variedades de 'cosas' inteligentes, ya sea para detectar, comunicarse, actuar y cooperar entre sí. Dentro de estas 'cosas' podemos encontrar armamento, vehículos, robots y dispositivos portátiles los cuales actuarán como agentes para realizar acciones defensivas coordinadas [20].
- **Turismo inteligente**. Los mapas turísticos pueden implantar una serie de etiquetas que permitan a los celulares modernos, navegar y llamar automáticamente a los servicios web que recomienden de manera personalizada sobre distintos lugares como restaurantes y hoteles. [19].

1.2.1.7. Herramientas de desarrollo

Existen múltiples plataformas dentro del Internet de las Cosas, las cuales permiten el desarrollo de aplicaciones para recoger y analizar datos, por ejemplo, de un sensor. Dentro de esas herramientas se encuentra Amazon AWS, Microsoft Azure, IBM Watson y ThingWorx. Para [21], la gran mayoría de estas aplicaciones generan un descubrimiento y registro, así como transacciones seguras, así como motores que se basan en normas para brindar una respuesta reactiva a los mensajes y salidas de los sensores del Internet de las Cosas.

Existen herramientas que permiten modelar sistemas basados en el Internet de las Cosas como son los middlewares para IoT. Un middleware IoT totalmente funcional necesita integrar las tecnologías máquina a máquina (M2M) y adquisición de datos para admitir los diferentes dominios de aplicación. Lógicamente no hay ningún middleware que permita aceptar todos los requisitos necesarios para las aplicaciones IoT. Es por eso que en últimos años han ido surgiendo distintos middlewares específicos de IoT como por ejemplo basados en eventos (i.e., Hermes, EMMA, GREEN), orientados al servicio (i.e., Hydra, SOCRADES, Servilla), orientados a la base de datos (i.e., SINA, COUGAR, IrisNet), basado en agentes (i.e., Impala, Agilla, AFME) [22].

A pesar de la existencia de múltiples middlewares para implementar sistemas de IoT, se ha considerado relevante el uso de la herramienta OpenHAB para el desarrollo de esta investigación. Esto se debe a que es una tecnología especializada en la gestión del hogar digital conectado a través de tecnologías de IoT.

1.2.1.8. OpenHAB

Open Home Automation Bus (OpenHAB), es una plataforma muy utilizada y común entre los dispositivos compatibles con el IoT. Para Ramljak [23], OpenHAB es un software que tiene como capacidad la integración de distintos sistemas y tecnologías de automatización del hogar es una sola solución. El software no intenta modificar las soluciones actuales, sino que permite mejorarlas. Dentro de la arquitectura de este software, existe un diseño modular que permite al sistema extenderse en un tiempo de ejecución sin pausas.

OpenHAB tiene los conceptos de “Cosas”, “Canales”, “Elementos” y “Enlaces”. Las cosas se las define como objetos. Entre ellos pueden estar dispositivos de hardware, servicios web, que se pueden implementar físicamente a un sistema. La funcionalidad de estos objetos se trasmite por medio de varios canales. Estos canales funcionan por medio de los enlaces personalizados que nos brinda la interfaz con los objetos externos. Los elementos exhiben el funcionamiento que utilizan las aplicaciones como interfaces de usuario y están asociados a los canales mediante enlaces [21]. Cada uno de estos conceptos se relacionan como se ilustra en la Figura 2.

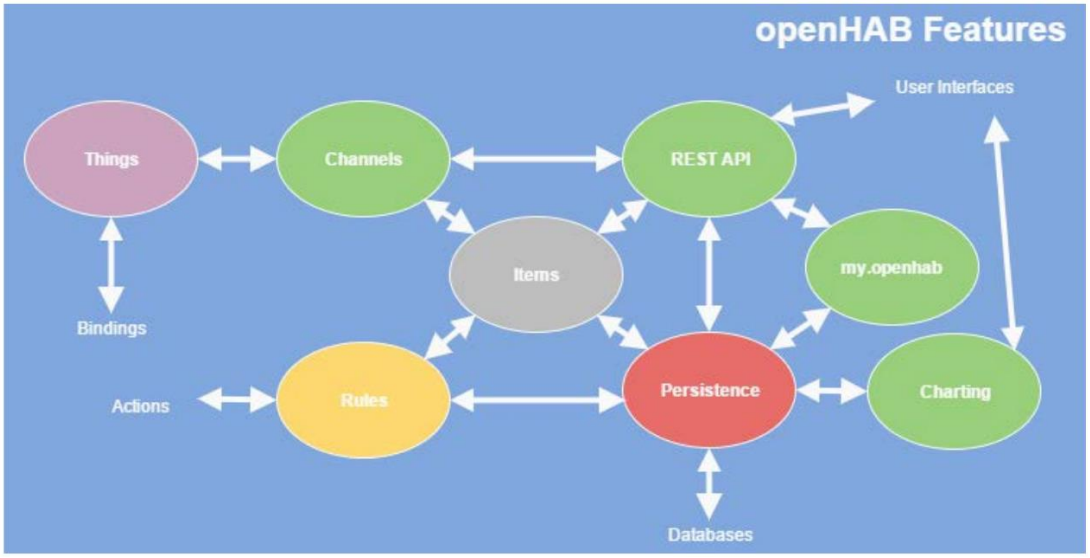

Figura 2 Esquema de funcionamiento de los distintos bloques de OpenHAB[21]

OpenHAB es una extensión del proyecto *EclipseSmartHome*, basado en Java y fue construido específicamente para integrar nuevos dispositivos, protocolos y servicios de una manera más cómoda. Los componentes de este software mantienen comunicación mediante un flujo de eventos [24]. Algunos de los componentes más destacados son, la biblioteca de complementos, componente para la gestión de servicios REST y el servicio HTTP.

La biblioteca de complementos da a conocer al componente más utilizado de la arquitectura, que son los enlaces. Los enlaces mantienen contacto con el núcleo de OpenHAB mediante un bus de mensajes que se basa en eventos. El servicio HTTP es el encargado de la configuración y administración principal de OpenHAB. Mientras que el servicio REST, es mayoritariamente el encargado de implementar la interfaz API JSON, la cual, genera puntos HTPP para que otras aplicaciones de terceras personas tengan comunicación con OpenHAB. Ejemplo, comunicación entre aplicaciones móviles de Android que tengan implementado OpenHAB y aplicaciones iOS [23].

Se implementa también en marcos que contienen extensiones para describir el contenido de una interfaz de una forma declarativa (i.e., widgets, iconos, gráficos) [25] .

1.2.2. Tecnología orientada a agentes

Se denomina agente inteligente a aquella entidad que tiene la capacidad de ser autónoma, con el objetivo de lograr un propósito, mediante acciones en su entorno y percepción del mismo. Pueden ser objetos de carácter físico como por ejemplo un robot equipado con distintos sensores y actuadores o pueden ser entidades virtuales como agentes de software [26].

La evaluación de un agente se realiza mediante la función de la optimización de su robustez, autonomía, eficiencia, solución y capacidad de aprendizaje y mejora. Esto habilita a los agentes para que muestren un comportamiento proactivo, reactivo y social [27]. No obstante, los agentes pueden incorporar otras características como autonomía, capacidad de liderazgo, soporte a la toma de decisiones, heterogeneidad, estructural, capacidad social, movilidad, etc.; dependiendo de las necesidades o del rol que deban desempeñar en determinados entornos.

1.2.2.1. Sistema multi agente

Los sistemas multi agentes (*Multiagent Systems*, MAS por sus siglas en inglés) es la composición de varios agentes autónomos interactuantes con un objetivo en común. En general, estos sistemas son descentralizados y asíncronos, pero hay situaciones en que pueden ser centralizados o híbridos [26].

Los agentes que actúan dentro de estos sistemas pueden ser de tipo homogéneos o heterogéneos. Por ello, deben tener la destreza de cooperar, coordinar y negociar sus actividades. Estos sistemas representan una subclase de sistemas de procesamiento distribuidos, paralelos o concurrentes, todo desde un punto de vista estructural [28].

En la actualidad, los sistemas multi agentes han sido capaces de generar una gran expectativa, así como ser cada vez más populares en áreas destacadas como son las matemáticas, tecnología, redes, inteligencia artificial, economía, entre otras. También son de mucha relevancia en el área de robótica, ciencias sociales, en sistemas de apoyo a la toma de decisiones, en el campo de minería de datos, y de a poco se introducen también en el IoT.

1.2.2.2. Características

Los agentes tienen la capacidad de reaccionar a su entorno sin tener noción del pasado o del estado futuro del entorno. Su razonamiento se centra en el conocimiento explícito, teniendo la facultad de efectuar mejoras en su elemento de ejecución al buscar retroalimentación acerca de sus acciones e interacciones con su entorno [29].

Los sistemas multi agentes tienen las siguientes características:

- **Autónomo**. Los agentes mantienen un cierto control en sus acciones y en su estado interno para intentar influir en los resultados sin ningún tipo de intervención humana o de dispositivos externos.
- **Social.** Los agentes son capaces de mantener una comunicación con personas, dispositivos externos u otros agentes para alcanzar un objetivo mediante la coordinación de acciones.
- **Reactivo.** Los agentes tienen la característica de adaptarse y reaccionar adecuadamente a los distintos cambios que pueda tener en su entorno.
- **Proactivo.** Los agentes muestran conductas dirigidas a objetivos para así tomar la iniciativa para conseguir sus propósitos [29].

Dentro de las características se pueden discutir distintas categorizaciones que van surgiendo a medida que se van considerando como característica las cuales son:

- **Capacidad de liderazgo.** Un agente puede ser el líder de otros agentes, es decir, el agente líder define los objetivos y las acciones a realizar para los otros agentes basándose en un objetivo general o global. Los otros agentes entonces, se comunican y comparten el conocimiento para localizar la posición del líder. El líder no solamente puede ser un solo agente sino también puede ser un grupo de agentes los cuales pueden moverse de un lado a otro. Por tanto, en ocasiones, los agentes tienen la necesidad de rastrear la posición del líder que incurren en gastos generales y de retardo adiciones como es comunicación y procesamiento. Cuando hay que tomar decisiones en el caso de tener varios líderes, se necesita la colaboración de los otros agentes seguidores para mantener comunicación entre ellos y así poder efectuar decisiones [30].

- **Soporte a la toma de decisiones.** Dentro de esta característica, se clasifica el sistema multi-agente según la proporcionalidad de las modificaciones en la salida de la función de decisión respecto a las modificaciones de entrada. Dependiendo de la función de decisión empleada se clasifican en lineal (aquellos sistemas donde la decisión de una agente es proporcional a los parámetros encontrados del entorno) y no lineal (donde la toma decisiones que realiza el agente no es proporcional a las métricas encontradas).
- **Heterogeneidad.** Dependiendo de la heterogeneidad de los agentes se los puede clasificar en homogéneos y heterogéneos. Los sistemas multi-agente heterogéneos incorporan agentes que comparten las mismas características y funcionalidades, mientras que los heterogéneos son lo contrario, es decir, tienen diferentes características [30].
- **Estructural.** La topología se refiere a la ubicación y a la relación que hay entre los agentes. Existen dos tipos de topología para los sistemas multi-agentes; esto es dinámicas o estáticas. En los sistemas multi-agentes estáticos, la posición y las relaciones de un agente no varían durante la vida útil del agente. Cuando se refiere a los dinámicos, es lo contrario, ya que la posición y las relaciones varían en cuanto un agente se mueve, abandona al sistema multi-agente o crea comunicaciones con otros agentes [30].
- **Capacidad social**. Los agentes transmiten los datos encontrados con otros agentes de dos maneras: por activación de tiempo o por evento. Por activación de tiempo, el agente en primer lugar halla continuamente el entorno, luego recopila dos datos y mediante intervalos de tiempo predefinidos los envía a los otros agentes. Mientras que, por evento, a diferencia de por activación de tiempo, el entorno solo es hallado por el agente cuando se produce un evento en especial, y luego esos datos son enviados a otros agentes [30].
- **Movilidad**. Los agentes se clasifican en agentes estáticos o móviles dependiendo de su dinámica. Entonces, un agente estático es aquel que se mantiene siempre en una misma posición dentro de un entorno, mientras que los agentes dinámicos son aquellos que tienen movilidad en un entorno. El agente móvil puede ser establecido por otros agentes, esto significa que este agente emplea aquellos recursos de los otros agentes para posteriormente supervisarlos o detecta el entorno desde la posición de otros agentes para efectuar acciones [30].

1.2.2.3. Arquitecturas

Dentro de los sistemas multi agentes, los datos y el entorno están descentralizados, es por ello que la función de los agentes debe estar bien clara y definida para evitar futuros problemas en cuanto a la interacción de agentes. Por un lado, dependiendo de cómo se organizan los agentes tenemos las siguientes arquitecturas: centralizadas, distribuidas, jerárquicas.

Arquitectura centralizada

Esta arquitectura se caracteriza por ser un conjunto de agentes simples, los cuales tienen un carácter no comunicativo, son administrados solamente por el centro de control y son agentes homogéneos. Dentro de esta arquitectura se involucran dos tipos de agentes, los cognitivos (deliberativos) y los reactivos. Los agentes cognitivos son aquellos que manejan un mayor nivel de comunicación, así como de inteligencia, mientras que los agentes reactivos son aquellos que no cuentan con una toma de decisión de nivel superior y carecen de un mecanismo de inteligencia que se asimile al del ser humano [29].

Arquitectura distribuida

Esta arquitectura se basa en tener una sola estructura de control para gestionar un conjunto de agentes comunicativos. Los agentes desconocen el dominio de red donde están envueltos, es decir, cada agente local es consciente solamente de su parte de red. Es por ello, que intervienen los agentes individuales, los cuales pueden encontrar información de carácter global mediante la comunicación y coordinación con sus vecinos [31]. Este tipo de arquitectura brinda la capacidad de crear sistemas robustos con la inclusión de agentes que se reorganizan cuando existen perdidas de otros agentes.

Arquitectura jerárquica

La arquitectura jerárquica, como su propio nombre indica, se trata de establecer una estructura mediante un orden de autoridad entre los agentes, es decir, existen agentes que tienen autoridad sobre los actos de otros agentes [32].

Generalmente los agentes de un nivel jerárquico superior son aquellos que toman las decisiones más importantes o críticas, así como son los encargados del control de gran cantidad de datos y de la política en general.

Por otro lado, los agentes intermedios en cuanto a jerarquía, son los encargados del control de la red, así como de minimizar pérdidas, localizar errores y restaurar el servicio. Mientras que los agentes del nivel más bajo, son los que mantienen una comunicación con los sensores y dispositivos conectados a la micro red.

El modelo jerárquico brinda una gran escalabilidad mediante la determinación de los roles a los agentes para tener un control operacional en tiempo real [29]. Dependiendo de cómo los agentes están estructurados y del nivel de inteligencia que soportan, éstos se clasifican en: reactivos, deliberativos e híbridos.

Arquitectura reactiva

Este tipo de arquitecturas se caracterizan por no contar como elemento central de razonamiento un modelo simbólico y por no implementar razonamiento simbólico complejo. Los agentes reactivos se caracterizan porque operan rápida y efectivamente sin tener que procesar una representación simbólica del entorno. Dentro de esta arquitectura los agentes funcionan siguiendo un esquema estímulo-respuesta dependiendo del estado actual del entorno en que están inmersos [33]. Una ilustración de esta arquitectura está reflejada en la Figura 3.

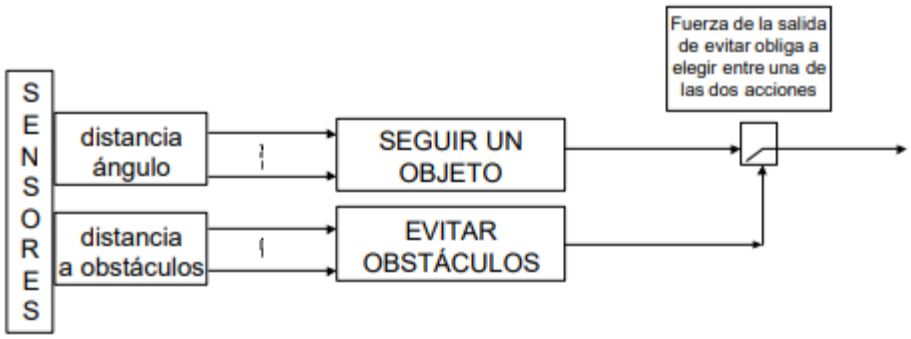

Figura 3 Ejemplo de arquitectura reactiva.[34].

Estas arquitecturas se han centrado mayormente en el desarrollo de controladores dentro del ámbito robótico, ya que se puede considerar a un robot como un agente real que actúan en un entorno cambiante.

Arquitectura deliberativa

Esta arquitectura se centra en las ciencias cognitivas y basa su idea en que las reglas sociales deben tener presencia en el proceso de toma de decisiones de un agente. El agente debe ser capaz de razonar, comunicarse y negociar sobre estas normas. Debe haber una comunicación entre los objetos mentales (creencias, deseos e intenciones) y las normas, las cuales deben verse también como un objeto mental con su propia representación. Los objetivos de un agente también deben mantener comunicación con los elementos mencionados anteriormente [35].

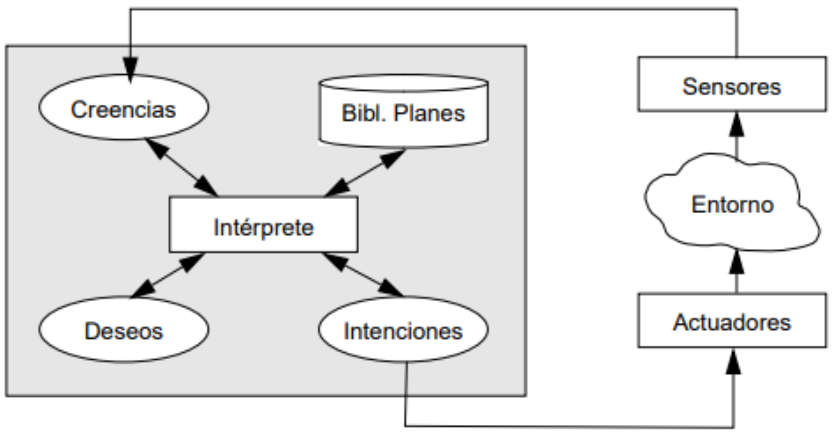

Figura 4 Arquitectura de un agente deliberativo[36].

Dentro de la Figura 4, las creencias son el conocimiento que un agente tiene sobre el mismo y su entorno, mientras que los deseos son los objetivos del agente a largo plazo. Las intenciones representan el estado deliberativo del agente; es decir, lo que el agente ha elegido hacer.

Arquitectura híbrida

Para superar limitaciones presentadas por otras arquitecturas, se utilizan este tipo de arquitectura, bastante popular en los últimos años, con la idea fundamental de estructurar las funcionalidades de un agente en dos o más capas organizadas de forma jerárquica, las cuales interactúan entre sí para conseguir un comportamiento coherente del agente en su conjunto, así como se ilustra en la Figura 3. Dentro de las ventajas que ofrece esta arquitectura tenemos es la compatibilidad con la modularización de un agente, las distintas funcionalidades están visiblemente separadas y unidas por interfaces bien definidas. Todo esto hace que el diseño de los agentes sea más compacto y robusto [37].

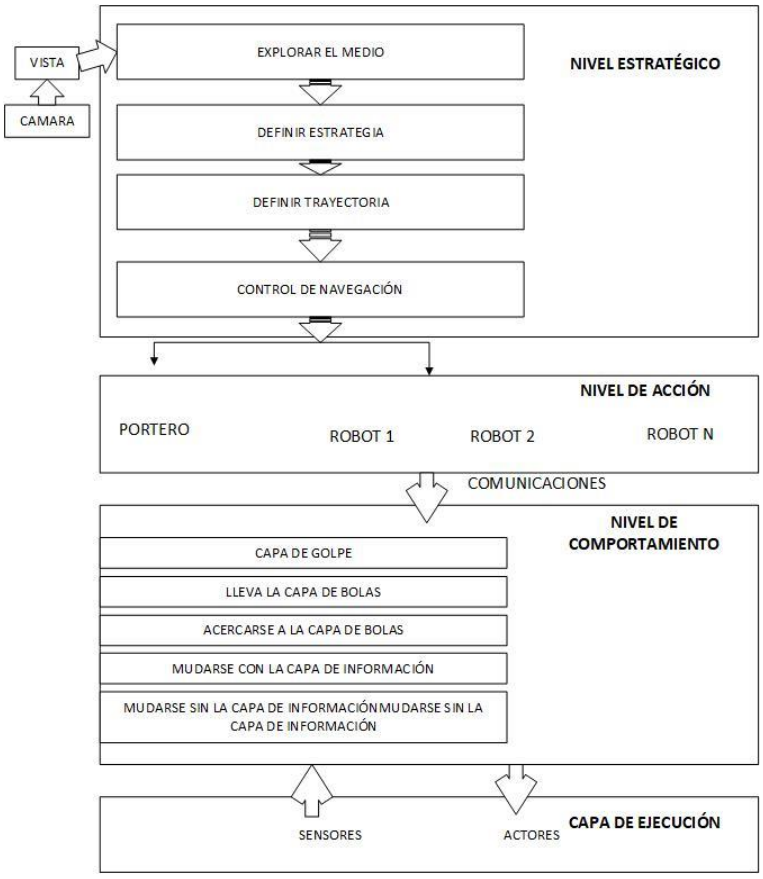

Figura 5 Arquitectura híbrida de sistemas multi agentes[38].

1.2.2.4. Marcos de trabajo para el desarrollo de agentes

Durante los últimos años se han realizado una gran de cantidad de trabajos donde se describen arquitecturas de diseño y modelos y comunicación entre los agentes. Asimismo, desde finales de los años 90, se han venido desarrollando numerosas plataformas y simuladores de entornos basados en agentes destinados a modelar comportamientos en escenarios específicos.

A continuación, se enlistan algunas de los principales marcos de trabajos que existen en la actualidad para desarrollar agentes y sistemas multiagentes. Entre los más relevantes figuran los siguientes: Voltrron, Zeus, RIAPS, Rapide, Darwin, ADML, Gaia, MaSE, Tropos, AAII (Australian Artificial Intelligence Institute), etc. A pesar de la existencia de una amplia gama de herramientas, se ha considerado emplear la herramienta JADE, orientada a Java, así como su versión SPADE, específica para Python.

1.2.2.5. JADE

Java Agent DEvelopment Framework (JADE), es un marco que se centra en un middleware que tiene como capacidad facilitar el desarrollo de aplicaciones multiagente distribuidas, el cual se basa en una arquitectura de comunicación punto a punto [39].

JADE como su propio nombre indica, se lleva a cabo solamente en el lenguaje JAVA, y tiene como requisito mínimo la versión 1.2 de Java, es decir, tiene la capacidad de poder acoplarse para ser empleado en dispositivos de bajos recursos como por ejemplo los celulares. Es un software gratuito, de código abierto [40].

La arquitectura JADE consta de varios componentes como se ilustra en la Figura 6. La gestión de la tabla contenedores, que es el registro de las referencias de objetos y el transporte de las direcciones de todos los nodos contenedores que componen la plataforma. El *Agent Management System* (AMS), es el agente que supervisa toda la plataforma, siendo el punto de contacto de todos los agentes que necesitan interactuar para poder acceder a las páginas blancas de la plataforma y así también poder

gestionar su ciclo de vida. Todos estos agentes AMS deben ser registrados para así obtener una AID (Agente identificador) válida.

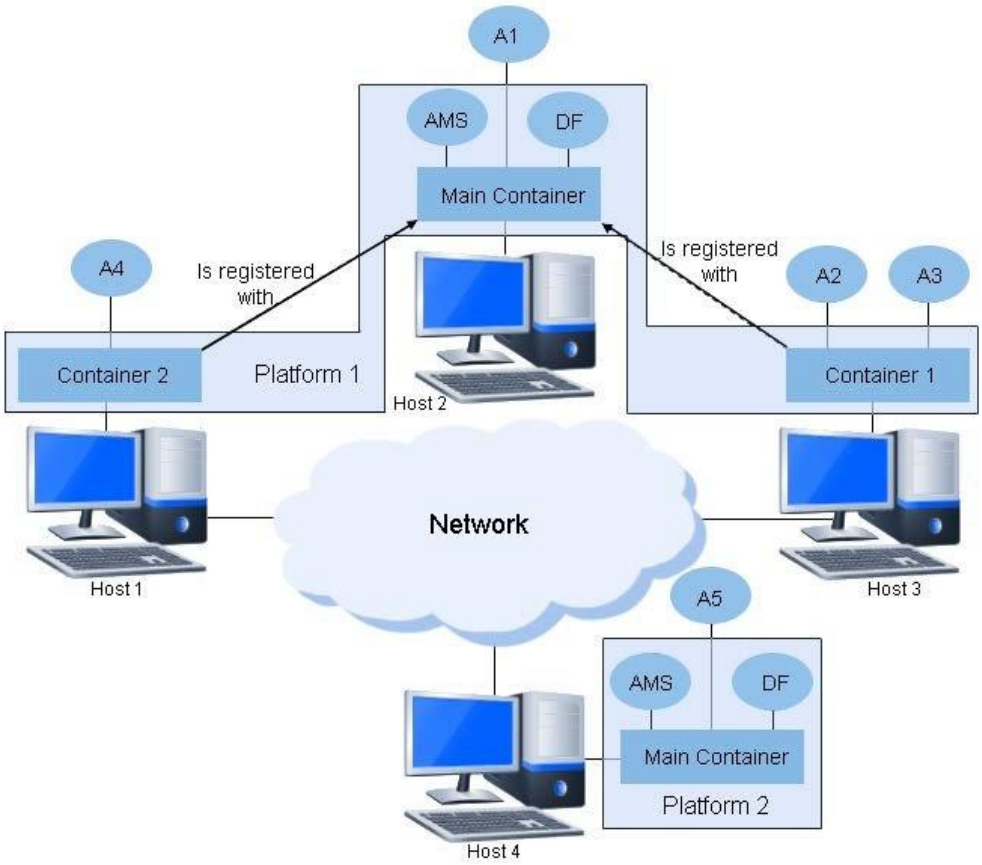

Figura 6 Principales elementos de la arquitectura de JADE [39]

JADE contiene varios servicios, como Agent Mobility Service el cual permite la movilidad entre plataforma, es decir, tolerar a los agentes de software moverse entre los distintos contenedores dentro de una misma plataforma [41]. Dentro de sus funcionalidades figuran las siguientes:

- Contiene un sistema totalmente poblado de agentes, los cuales cada uno se ejecuta como subprocesos separados, es decir, la plataforma nos brinda una API única.
- Cumple a totalidad con las especificaciones FIPA (*Foundation for Intelligent Physical Agent*).
- La transmisión de mensajes asíncronos por medio de una API se realiza de forma eficiente.

- Implementa API simples, así como herramientas gráficas para llevar un control de los ciclos de vida de los agentes, tanto de manera local como remota (suspender, reanudar, migrar, clonar) [42].

1.2.2.6. SPADE

Smart Python Multi-agent Develpment (SPADE) se la define como una plataforma destinada a sistemas multi agentes, orientada en la tecnología de mensajería instantánea XMMP. Con el paso de los años se han ido introduciendo distintas características que le han permitido integrarse de gran forma a los sistemas multi agentes [43].

La plataforma SPADE permite obtener resultados de carácter flexible y distribuidos donde permite que dispositivos de los sistemas multi agentes se agreguen o se eliminen de una forma transparente. Dentro de esta plataforma, el sistema está al tanto de los agentes conectados en tiempo real, dentro de los cuales están los agentes detectores que se posicionan dentro del entorno, para obtener y calcular la posición de un objeto [44].

La plataforma SPADE (Figura 7) está modelada según la FIPA estándar para una plataforma multi agente. Consta de servicios básicos estándares como un sistema de gestión de agentes y un facilitador de directorio, diseñados como componentes del servidor. El sistema de comunicación central se fundamenta en la tecnología Jabber pero también es compatible con otros protocolos de transporte de mensajes como HTTP. El enrutador XML es el elemento principal de la plataforma, donde el resto de los componentes y el resto de los agentes están conectados. Es un servidor XMPP estándar que enruta todos los mensajes de su remitente al receptor establecido sin tener que intervenir el usuario. El componente que gestiona toda la comunicación dentro de la plataforma se denomina ACC (*Agent Communication Channel*), el cual recibe los mensajes que llegan a la plataforma y los redirige al elemento final correcto, ya sea un agente u otro componente.

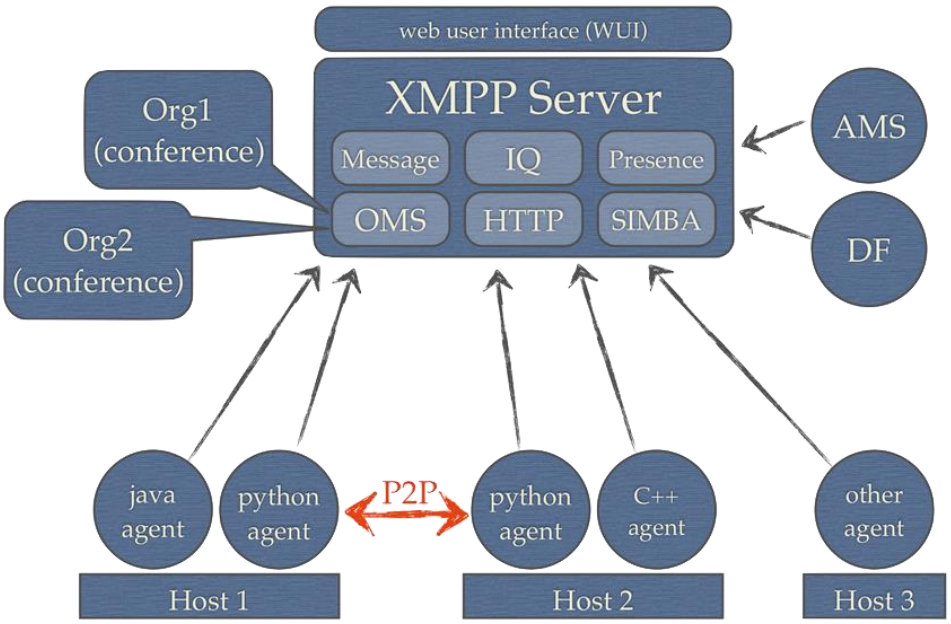

Figura 7 Modelo de plataforma SPADE [44]

Otras de las características de la plataforma SPADE es la creación de foros para los agentes, que estén conectados realizando procesos de varios agentes. Dentro de la plataforma SPADE se admiten dos tipos de idiomas de contenido, muy empleados hoy en día en el campo de agentes, los cuales son FIPA-SL (*Foundation for Intelligent physical agents*) y RDF (*Resource Description Framework*). Por medio de los diferentes servidores de SPADE, se admiten grandes cantidades de usuarios y mensajes, debido al enrutamiento de los mensajes Jabber, brindando a la plataforma un mayor rendimiento, así como una gran escalabilidad [45].

1.2.2.7. Protocolos de comunicación a nivel de agente

FIPA (*Fundation for Intelligent Physical Agents*) es un protocolo el cual establece estándares promoviendo la tecnología basada en agentes. Este protocolo consta de su propio lenguaje denominado lenguaje de comunicación del agente (ACL). Este lenguaje es muy importante en cuanto protocolo de red de contrato de alto nivel [46].

Se entiende como protocolo en general, a una serie de reglas a cumplir. En este caso, se encuentran los protocolos de comunicación. Este protocolo normalmente se encuentra en lenguaje de alto nivel, el cual tiene como objetivo detallar el

procedimiento de coordinación entre los agentes y un conjunto de métodos de asignación de recursos y trabajos.

Existen varios protocolos diseñados para la comunicación entre agentes. Los protocolos más empleados son los siguientes:

Tabla 1. Protocolos de interacción FIPA[48].

Protocolo de Interacción	**Compartir Tareas/Ifo**	**Empujar/Halar**	**Receptores 1-1 / 1-m**	**Otras características**
Solicitud	Tarea	Halar	1-1	Cancelable (por iniciador)
Solicitar cuando (alguna vez)	Tarea	Empujar	1-1	Cancelable
Consulta	Info	Halar	1-1	Cancelable
Contrato neto / CN iterado	Tarea	Empujar	1-m	Versión iterada cancelable
Subasta de inglés / Duth	Info	Halar	1-m	Cancelable
Corredor	Info	Halar	1-m	Cancelable
Recluta	Tarea	Halar	1-1	Cancelable
Suscribir	Info	Empujar	1-1	No cancelable
Proponer	Tarea	Halar	1—1	No cancelable

1.2.2.8. Aplicaciones

Las aplicaciones de los sistemas multiagentes son amplias y heterogéneas. Por ejemplificar, y definir el alcance de los agentes, a continuación, se describen 4 aplicaciones:

En [64], se planteó un marco basado en agentes para efectuar la Bolsa de Tareas (BoT), que significa un grupo de tareas independientes que están en la nube.

Mediante un grupo de agentes intermediarios se capta información sobre los recursos aptos y los proveedores de la nube para cada tarea en BoT.

Por otro lado, las redes sociales cada vez son más populares dentro de las personas, en especial a los usuarios de internet, los cuales la mayoría interactúa con una red social a diario y de manera constante. En [30], se propuso un sistema basado en agentes con el objetivo de pronosticar el comportamiento de los usuarios. Estos agentes estarían dentro de las redes sociales para poder clasificar temas y opiniones de datos para cada usuario específico, para después realizar la construcción del perfil de usuario.

En el campo de la electricidad, los sistemas multiagentes también han sido aplicados para solucionar problemas como, control de una micro-red en un contexto de operación de mercado, localización de fallas y restauración de servicio [29].

Finalmente, el área de robótica está siendo un área de estudio constante donde se han aplicado los agentes software. Un ejemplo es un sistema multi agente combinados con robots transportadores de objetos, robots jugadores de futbol y robots que evaden obstáculos. Los sistemas multi agentes se han aplicado como solución a problemas que necesitan varios grados de coordinación [49].

1.2.3. Modelado del IoT usando agentes

Los agentes inteligentes mantienen un fuerte vínculo con el Internet de las Cosas. Los agentes permiten tomar contacto con los sistemas naturales y artificiales caracterizados por ser complejo, dinámico y por llevar una autonomía relevante. Es por ello, que, en la actualidad, se ha aprovechado tanto para modelar, programar y simular aplicaciones o sistemas dentro del ámbito del Internet de las Cosas [50].

El agente se considera autónomo, en cuanto a la realización de sus tareas de forma individual, pero auto determinada, es por ello que la toma de decisiones debe ser también autónoma en lo que tiene que ver con recursos y ejecución de tareas. La toma de decisiones autónoma es una de las características más relevantes de los ecosistemas de IoT. De allí, que se han planteado dos enfoques para modelar el IoT basado en agentes. Dichos enfoques corresponden a la creación de agentes embebidos sobre los

objetos de IoT y el desarrollo se sistemas multiagentes para controlar los recursos de IoT

Agente embebido en objetos de IoT

El proceso de embeber un agente en la arquitectura de un objeto físico puede ser de dos maneras. En la primera forma, se debe introducir el agente y ocurre cuando el objeto es de un tipo de hardware que permite actualizarse, lo cual implica cargar un firmware para que pueda actualizarse dinámicamente. Mientras que la segunda manera de incorporar agentes en un objeto de IoT es a partir de mecanismos alternativos como un conjunto de sensores/actuadores formando entonces objetos de IoT con un agregado que son los componentes SBC (*Single Board Computer*). Por tanto, en este segundo caso, se utiliza un microprocesador más potente y con la capacidad de ejecutar algoritmos utilizados por agentes para el control de componentes físicos [55][56].

Modelado de IoT basado en sistemas multi agentes.

En este enfoque se debe modelar un sistema multi agente para gestionar objetos de IoT y sus recursos, es por ello, que este tipo de red adquiere habilidades sociales, autonomía e inteligencia para permitir a los objetos de IoT que realicen actividades colaborativas entre varios objetos. Componer la red de objetos IoT es posible gracias a la implementación de middlewares especializados en brindar una interfaz de comunicación de bajo nivel con los objetos de IoT [55][56].

1.3. Marco Legal

La presente investigación se centra en la generación de tecnología que aporta al crecimiento tecnológico del país. Para simplificar lo expuesto se presenta el siguiente análisis:

- El Plan Nacional de Gobierno electrónico 2018-2021 en base a la Carta Iberoamericana de Gobierno electrónico (2017), formula 12 principios que precautelan el derecho de los ciudadanos a relacionarse con el Estado electrónicamente. Entre estos principios figuran los siguientes:

- “Principio de adecuación tecnológica: Garantiza que las administraciones elegirán las tecnologías más adecuadas para satisfacer sus necesidades, por lo que se recomienda el uso de estándares abiertos y de software libre en razón de la seguridad, sostenibilidad a largo plazo y la socialización del conocimiento.”
- La Constitución de la República del Ecuador (2008) que garantiza la soberanía nacional y define los sectores estratégicos entre los cuales están las tecnologías como software y hardware:
- “Art.322. Se reconoce la propiedad intelectual de acuerdo con las condiciones que señale la ley. Se prohíbe toda forma de apropiación de conocimientos colectivos, en el ámbito de las ciencias, tecnologías y saberes ancestrales…”
- “Art. 385. El sistema nacional de ciencia, tecnología, innovación y saberes ancestrales, en el marco del respeto al ambiente, la naturaleza, la vida, las culturas y la soberanía, tendrá como finalidad desarrollar tecnologías e innovaciones que impulsen la producción nacional, eleven la eficiencia y productividad, mejoren la calidad de vida y contribuyan a la realización del buen vivir.

- Norma técnica ecuatoriana NTE INEN-ISO/IEC 29363: Tecnología de la información – Interoperabilidad de servicios web; mediante su promulgación en el Registro oficial, promueve un justo equilibrio de intereses entre proveedores y consumidores.

CAPÍTULO II: METODOLOGÍA

2.1. Tipo de investigación

La presente investigación tiene un carácter cuantitativo-cualitativo (mixto) ya que se utilizaron y se combinaron las características de cada una de ellas para obtener una mejor perspectiva de la investigación realizada.

La investigación es cuantitativa ya que implica el uso de herramientas informáticas, estadísticas, y matemáticas para obtener resultados, el objetivo del cual es, cuantificar el problema y entender qué tan generalizado está mediante la búsqueda de resultados proyectables a una población mayor. Por otro lado, este tipo de investigaciones son más exploratorias que depende de la recolección de datos, de conductas u observaciones.

Presenta una metodología cualitativa ya que tiene como objetivo la descripción de las cualidades de un fenómeno, tratando de descubrir la mayor cantidad de cualidades como sea posible. Además, hace énfasis en la validez de las investigaciones mediante la proximidad a la realidad empírica que proporciona esta metodología.

La investigación de tipo "mixta" brindó una mayor "exploración y explotación" de los datos para así poder conseguir el mayor éxito a la hora de presentar resultados. El estudio tuvo como finalidad identificar las diferencias y semejanzas entorno a un evento o en un mismo contexto. Se contextualizaron las herramientas para realizar comparaciones, evaluar resultados obtenidos mediante el uso de métricas de evaluación.

La investigación propuesta fue también exploratoria porque el tema abordado fue poco estudiado. El objetivo fue investigar y encontrar pruebas relacionadas con el uso de marcos de trabajo orientados a agentes como JADE y SPADE para el modelado de ecosistemas inteligentes de IoT. Dentro de la plataforma JADE, se investigó acerca de los "componentes activos" denominadas agentes y la comunicación bidireccional entre dichas entidades. Además, se investigó acerca de la comunicación en JADE, así como el comportamiento de los agentes en ambientes del hogar digital conectado implementado con tecnologías IoT. En este caso, se investigó cómo implementar un escenario de hogar digital conectado mediante la plataforma OpenHAB hacia JADE.

Asimismo, se empleó una segunda plataforma SPADE, y se realizó en mismo proceso llevado a cabo con JADE, a fin de determinar el alcance de Python y Java en el desarrollo de ecosistemas inteligentes de IoT.

2.2. Método de investigación

El diseño de la investigación fue experimental ya que se empleó estrategias para responder a los interrogantes sobre la comunicación que debe tener el sistema multi agente con la plataforma OpenHAB, tanto en JADE como SPADE y así poder alcanzar los objetivos planteados. También fue experimental ya que el investigador provocó una situación para introducir determinadas variables de estudio modificadas por él, para controlar y observar lo que sucede en situaciones controladas, es decir, establecer la causa y el efecto de un fenómeno. Al utilizar este tipo de investigación, se obtuvo un fuerte control sobre las variables y por ende ayudó a analizar el alcance de los sistemas agentificados de IoT usando JADE y SPADE.

Por otro lado, el método sintético permitió cumplir con una serie de actividades inherentes a la revisión y lectura de diversos documentos donde se encontraron ideas explicitas relacionadas a sistemas multi agentes, además del proceso de comunicación de los agentes con las distintas plataformas de interés. Por lo tanto, se realizaron continuas interpretaciones con el claro propósito de revisar aquellas apreciaciones o investigaciones propuestas por diferentes investigadores relacionadas con el tema de interés, para luego dar la respectiva argumentación a los planteamientos, en función a las necesidades encontradas en la indagación.

2.3. Variables sujetas a investigación

El tiempo de respuesta y la cantidad de usuarios concurrentes son indicadores relevantes de rendimiento en sistemas distribuidos, así como en sistemas multi agentes. Hay otros factores que se incluyen por el nivel del agente y que tienen efecto sobre el rendimiento que son, la coordinación, el modelo del conocimiento y el modelo de racionalidad del agente. Dentro de un modelo de varios agentes, las variables que afectan al rendimiento incluyen el número de agentes, el número de tareas que los agentes realizan, la organización de los agentes y el tipo de protocolos que se emplean para llevar a cabo procesos de comunicación [58].

La seguridad es otro aspecto muy importante dentro de la comunicación entre agentes, es por ello, que se debe implementar métodos que garanticen las comunicaciones seguras entre ellos. Existen arquitecturas de sistemas multi agentes que permiten a los agentes del sistema la funcionalidad necesaria para implementar cualquier protocolo de seguridad, proporcionando así seguridad al usuario del servicio. Estos sistemas se caracterizan por ofrecer a cualquier aplicación distribuida servicios de carácter dinámicos y flexibles de seguridad, así como la libertad a los agentes intermediarios para usar un protocolo de seguridad en cada aplicación, para brindar integridad de los datos o la confidencialidad de la información [59].

Para asegurar este aspecto, se tiene que basar en estándares, es decir, es una variedad de recomendaciones a cumplir para conseguir la entrega de un producto. Basándose en la ISO/IEC 25000 se establecen algunas características de software a cumplir, así como la calidad de software que contiene métricas de calidad. Dentro de esta norma se encuentran beneficios como el aumento de eficiencia en los requerimientos del software, así como la mejora de calidad de un producto basándose en evaluaciones internas [60]. Se escogió adecuación funcional, la cual representa la capacidad del producto software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando el producto se usa en las condiciones específicas. Esta característica se subdivide a su vez en las siguientes sub-características que son, la completitud funcional, que es el grado en el cual el conjunto de funcionalidades cubre todas las tareas y los objetivos del usuario especificado, y la corrección funcional, que tiene la capacidad del sistema para proveer resultados correctos con el nivel de precisión requerido.

La calidad del producto de software se divide en 3 fases según SQuaRE (Requisitos y evaluación de calidad del sistema y software) que son calidad interna, es decir, el producto de software en desarrollo; calidad externa, el producto de software en funcionamiento, y por último la calidad en uso que no es más que el producto de software en uso. El modelo de calidad del producto analiza una serie de características como eficiencia en el rendimiento, compatibilidad, confiabilidad, seguridad, y en esta investigación nos centramos en el tiempo de procesamiento [61].

Una de las motivaciones de realizar este proyecto mediante sistemas multiagentes es comprobar cuan factible es utilizar agentes inteligentes en sistemas embebidos como una computadora o una Raspberry Pi, por eso en esta investigación se centró en el

tiempo de ejecución de los agentes tanto en JADE como SPADE para medir la factibilidad técnica, y determinar así, cuál de las dos tecnologías orientadas al desarrollo de agentes es la más idónea para implementas sistemas de agentes bajo hardware con restricciones de recursos de cómputo.

Finalmente, la integridad de los datos hace referencia al hecho de que dichos datos recibidos sean los mismos que se enviaron en su tiempo. Es posible conocer si un agente o un mensaje enviado por un agente ha sido alterado de alguna forma, sea por errores de comunicación como por fuentes externas malintencionadas [62].

2.4. Técnicas de investigación

El componente cualitativo de la investigación se llevó a cabo mediante el uso de la técnica de análisis documental. A través de una revisión de la literatura que exploró fuentes de información científica de impactos (i.e., Scopus, IEEE Xplore, ACM), se determinó el alcance de las tecnologías involucradas en esta investigación, esto es, desarrollo basado en agentes, IoT y agentificación del IoT usando agentes.

Por otro lado, en lo relacionado a la parte cuantitativa de la investigación, donde se validó el funcionamiento del escenario sujeto a estudio (hogar digital implementado con IoT agentificado), y se midieron las variables sujetas a estudio, previamente descritas, se empleó la técnica de la experimentación en laboratorio. Dicha técnica se materializó mediante el diseño de un experimento y un *test*. El formato de dicho instrumento es descrito en el Anexo 1.

2.5. Tecnologías usadas

Para el desarrollo del estudio fue necesario crear un ecosistema de IoT a partir del cual se desarrolló un hogar digital conectado. Asimismo, se desarrolló un ecosistema de agentes software a partir de los cuales se llevaron a cabo las acciones de control de los objetos de IoT. Para el desarrollo de estos ecosistemas se emplearon las siguientes tecnologías:

- OpenHAB. La versión usada fue OpenHAB 2.5.7, cuya descarga fue realizada en el sitio oficial de la plataforma, https://www.openhab.org/. Previamente se debe

instalar la plataforma Java 8 para que funcione correctamente.

- JADE. Dentro de esta plataforma de desarrollo de agentes, que facilita el desarrollo de sistemas multi-agente bajo el estándar FIPA, se escogió para trabajar el entorno Apache NetBeans. La versión de este marco de trabajo se encuentra disponible en: https://jade.tilab.com/
- SPADE. Plataforma libre para la creación de sistemas multi agentes que trabaja con el lenguaje de programación Python, donde se escogió para programar la plataforma PyCharm. La versión empleada para desarrollar los agentes SPADE, compatible con versiones de Python superior a 3.6, se encuentra disponible en: https://spade-mas.readthedocs.io/en/latest/readme.html.
- Java. Se utilizó la versión Java 8 para poder trabajar correctamente con la plataforma OpenHAB, descargada e instalada del sitio oficial, https://www.java.com/es/download/.
- Python. Lenguaje de programación que cuenta con documentación en español de forma gratuita para ayudar a programar pequeñas aplicaciones o grandes proyectos con interfaz gráfica. Para instalar Python se debe ir al gestor de software de distribución e instalar la versión Python 3.6, además se recomienda instalar Pip (Instalación de Paquetes) para facilitar la obtención de librerías.
- PhyCharm. Se escogió este IDE por sus beneficios a la hora de programar, ya que facilita un editor de código inteligente, una indicación de errores instantáneo, además de tener un depurador y ejecutor de pruebas integrados. Se utilizó la versión 2020.2, descargada de la página https://www.jetbrains.com/es-es/pycharm/download/#section=linux
- Apache NetBeans. Se trata del entorno de desarrollo integrado libre, realizado principalmente destinado para el lenguaje de programación Java. Se utilizó la versión Apache NetBeans 11.2, la cual fue descargada del sitio oficial, https://netbeans.apache.org/.

CAPÍTULO III: RESULTADOS

3.1. Escenario

Este objetivo del sistema propuesto es desarrollar un sistema inteligente basado en IoT dirigido por agentes que fuera capaz de brindar confort control lumínico, de seguridad y térmico en un hogar inteligente. El escenario de IoT, desarrollado en la herramienta OpenHAB, consistió en un ecosistema compuesto por sensores y actuadores automáticos que estuvieron distribuidos en una vivienda de dos 2 pisos, un jardín y un pasillo.

En lo referente al control lumínico consiste en manejar si hay presencia en una habitación y depende si es de día o de noche poder tomar decisiones de apagar o encender las luces automáticamente.

Por otro lado, en lo relacionado al control de seguridad, éste permite poder tomar decisiones de cerrar las ventanas o puertas en caso de no haber presencia humana en la casa o también abrirlas en caso de si haberlo tomando en cuenta la hora del día, esto es día o noche. Además, este confort contempla poder tomar decisiones de seguridad en caso de fugas de gas como es el caso de la cocina.

Finalmente, el control térmico se enfoca en aplicar medidas de control mediante sensores que leen la temperatura en toda la casa y en base a ellos manipular la temperatura del ambiente e incluso manejar los aires acondicionados de tal forma que sea un confort ideal para las personas que se encuentren en la misma.

3.2 Ecosistema de IoT

Mediante la plataforma Openhab 2.5.8 se implementó el ecosistema IoT compuestos por Planta baja o piso uno, segundo piso, pasillo y un jardín. Como se observa en la Figura 8 del diseño del hogar digital se puede visualizar todo lo antes mencionado con respecto al escenario ya descrito.

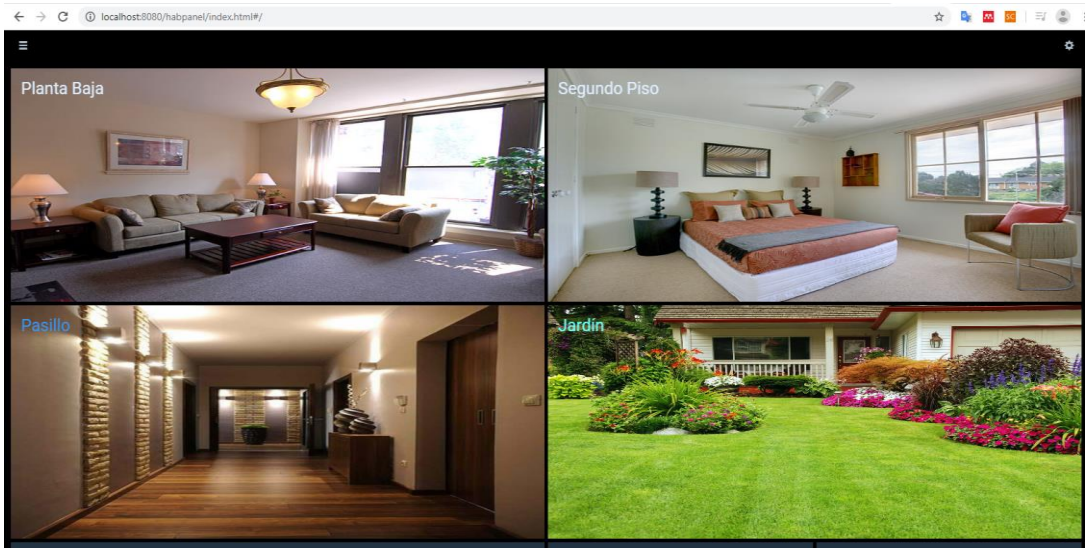

Figura 8. Diseño del hogar digital

Para implementar el ecosistema se procedió a descargar de la web la herramienta OpenHAB en la versión antes mencionada, luego se procedió a editar la interfaz del el diseño planteado, y a continuación, se crearon los objetos de IoT en cada una de las habitaciones que formaron el escenario. Para ejecutar estos procesos se ejecutó OpenHAB desde la consola de comando en el sistema operativo Windows. Accediendo al servidor local de OpenHAB en el puerto 80 (http://localhost:8080) fue posible visualizar el escenario antes mostrado en la Figura 1.

La primera planta o planta baja del hogar domótico estuvo compuesto por una cocina, con sus respectivos objetos lumínicos, por un sensor de temperatura y una persiana. Esta planta constó también de una sala y un baño social incluyendo en ambos, sensores de temperatura, objetos lumínicos, persianas y un aire acondicionado/calefacción. Dentro de la sala, existió un sensor lumínico que se enciende cuando detecta poca luz solar y se apaga cuando hay bastante luz solar. También existe un sensor térmico que se enciende siempre y cuando la temperatura esté por encima del valor establecido y se apaga cuando el valor esté por debajo de dicho umbral. Estos mismos objetivos de control tiene el baño de la planta baja. Asimismo, en la cocina se incluyeron sensores de temperatura, ventana, una puerta, sensor de gas, y bombillas de luces. En la sala, se encontraron también bombillas de luces, sensores de temperatura, sensores lumínicos, ventanas y sensores de apertura de puertas. Finalmente, en el baño se incorporaron también bombillas de luces, espejos, sensores de temperatura, ventanaas y puertas. Todos estos objetos se ilustran en la Figura 8.

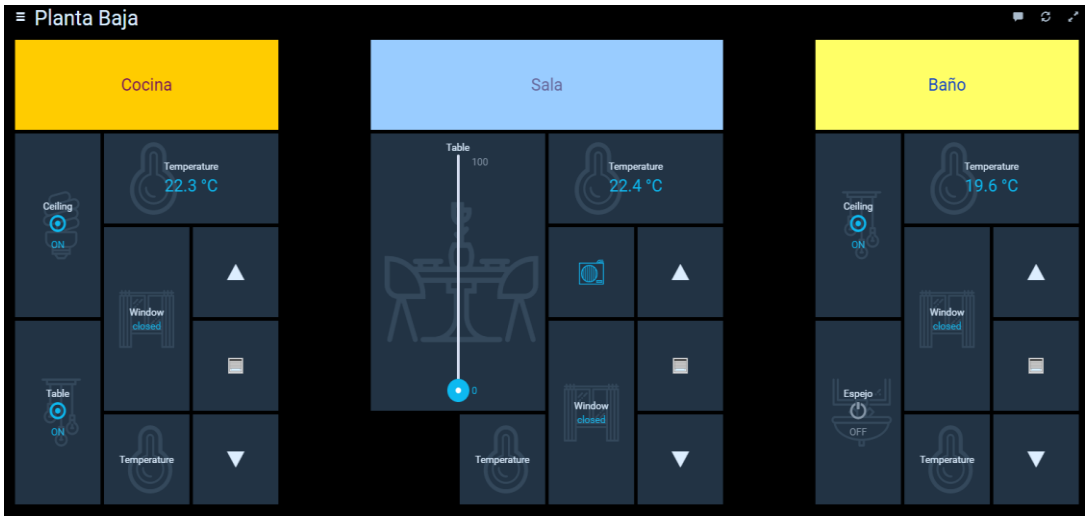

Figura 9. Planta baja del hogar

El segundo piso del hogar domótico estuvo compuesto por dos habitaciones, una máster y una para un bebé, además consta de una oficina, un baño y un pasillo. Todos estos habitáculos tuvieron implementados sensores y objetos lumínicos, sensores de temperatura, aire acondicionado/calefacción y persianas. El cuarto máster incluyó bombillas de luces, sensores de temperatura, sensores lumínicos, ventanas y una puerta. Asimismo, la habitación del bebé estuvo integrada por bombillas de luces, sensores de temperatura, sensores lumínicos, un baño y una puerta. Finalmente, la oficina integró bombillas de luces, sensores de temperatura, ventana, puerta y un sensor lumínico.

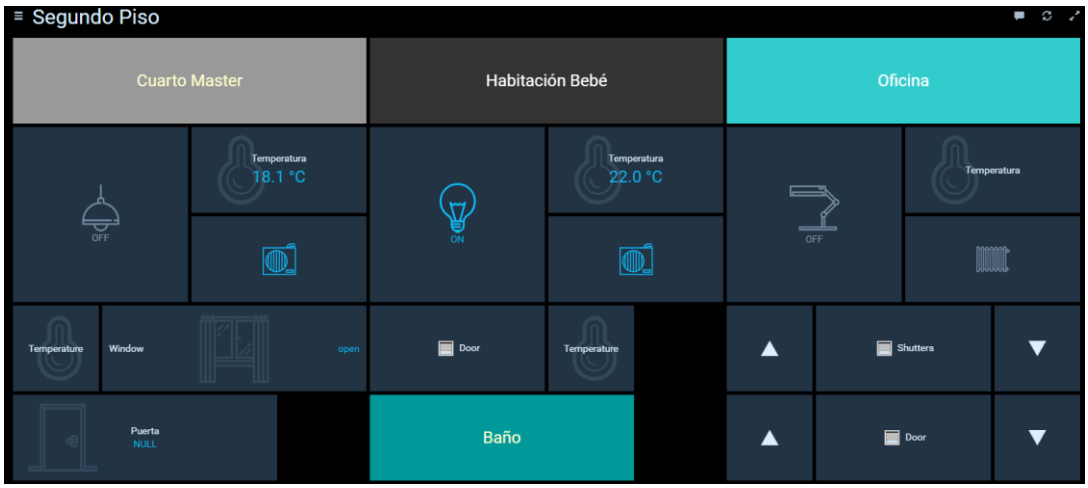

Figura 10. Segundo piso del hogar domótico

En lo que respecta al jardín, éste también tuvo dos espacios físicos, una terraza y un garaje. La interfaz de usuario para monitorear el estado de los objetos instalados se muestra en la Figura 11.

Figura 11. Jardín y garaje del hogar domótico

3.3. Desarrollo del ecosistema de IoT en OpenHAB

El archivo esencial del sistema OpenHAB se encuentra en el directorio *config*. En este directorio guarda archivos de configuración como es el *sitemaps* e *items*. La función y la estructura de estos dos archivos se describen como sigue:

Directorio sitemaps

Un archivo de tipo *sitemap* se utiliza para seleccionar y preparar estos elementos con el fin de componer una presentación orientada al usuario de esta configuración para varias interfaces de usuario. Este archivo, llamado *demo.sitema*p se ubicó en el directorio C:\OpenHab 2.0\conf\sitemaps\. En la Figura 12, líneas 3-7, se muestran los marcos usados para organizar los pisos del hogar digital en la interfaz de monitoreo y control de usuario. Los marcos siguientes permitieron organizar los datos referentes a la climatología de la zona.

```
sitemap demo label="Main Menu"
{
    Frame {
        Group item=gFF label="First Floor" icon="firstfloor"
        Group item=gGF label="Ground Floor" icon="groundfloor"
        Group item=gC label="Cellar" icon="cellar"
        Group item=Garden icon="garden"
    }
    Frame label="Weather" {
        Text item=Weather_Temperature valuecolor=[Weather_LastUpdate=="NULL"="lightgray",Weather_LastUpdate>90="lightgray",>25="orange",>15="green",>5="orange",<
            Frame {
                Text item=Weather_Temp_Max valuecolor=[>25="orange",>15="green",>5="orange",<=5="blue"]
                Text item=Weather_Temp_Min valuecolor=[>25="orange",>15="green",>5="orange",<=5="blue"]
                Text item=Weather_LastUpdate visibility=[Weather_LastUpdate>30] valuecolor=[Weather_LastUpdate>120="orange", Weather_LastUpdate>300="red"]
            }
            Frame {
                Switch item=Weather_Chart_Period label="Chart Period" icon="chart" mappings=[0="Hour", 1="Day", 2="Week"]
                Chart item=Weather_Chart period=h refresh=600000 visibility=[Weather_Chart_Period==0, Weather_Chart_Period=="NULL"]
                Chart item=Weather_Chart period=D refresh=3600000 visibility=[Weather_Chart_Period==1]
                Chart item=Weather_Chart period=W refresh=3600000 visibility=[Weather_Chart_Period==2]
            }
        }
        Text label="Astronomical Data" icon="sun" {
            Text item=Sun_Elevation
            Text item=Sun_Azimuth
            Text item=Sunrise_Time
            Text item=Sunset_Time
            Text item=Moon_Elevation
            Text item=Moon_Azimuth
            Text item=Moon_Phase
        }
    }
    Frame label="Demo" {
        Text item=CurrentDate
        Text label="Group Demo" icon="firstfloor" {
            Switch item=Lights mappings=[OFF="All Off"]
            Group item=Heating
            Group item=Windows
            Text item=Temperature
        }
        Text label="Widget Overview" icon="chart" {
            Frame label="Binary Widgets" {
                Switch item=DemoSwitch label="Toggle Switch"
                Switch item=DemoSwitch label="Button Switch" mappings=[ON="On"]
            }
```

Figura 12. Finalización del contenido sitemap

Directorio ítem

Un ítem en el entorno de la herramienta OpenHAB, es un elemento que representa todas las propiedades y capacidades de la domótica del usuario. Estos elementos tienen la característica de que a través de openHAB se los pueda conectar al mundo exterior a través de enlaces. Esto significa que se puede conectar de manera sencilla para el control de los objetos de IoT reales. El archivo ítem creado en este proyecto se almacenó en la ruta c:\openhab 2.0\conf\items, el cual, se describieron cada uno de los objetos (sensores y actuadores) incorporados en el hogar digital antes descrito, agrupados mediante coordinadores principales y subgrupos de éstos mismos. Cada una de las secciones de la casa domótica estuvo creada a través de un subgrupo.

El archivo ítems creado empezó estableciendo grupos por cada piso con iniciales abreviadas en inglés (gFF pertenece al primer piso, gGF pertenece al segundo piso, gC pertenece al pasillo y por último Garden pertenece al jardín). Dichos grupos se muestran en la Figura 13.

```
Group gFF          "First Floor"   <firstfloor>    ["FirstFloor"]
Group gGF          "Ground Floor"  <groundfloor>   ["GroundFloor"]
Group gC           "Cellar"        <cellar>        ["Basement"]
Group Garden       "Garden"        <garden>        ["Garden"]
Group Weather      "Weather"       <sun>

Group Status
Group Shutters

Group FF_Living    "Living Room"   <video>     (gFF)   ["LivingRoom"]
Group FF_Kitchen   "Kitchen"       <kitchen>   (gFF)   ["Kitchen"]
Group FF_Toilet    "Toilet"        <bath>      (gFF)   ["Bathroom"] { synonyms="Toilets,WC,Restroom" }
Group FF_Corridor  "Corridor"      <corridor>  (gFF)   ["Corridor"]
Group FF_Outside   "Outside smart home"   <outdoorlight> (gFF)   ["outdoorligh"]

Group GF_Bath      "Bathroom"      <bath>      (gGF)   ["Bathroom"]
Group GF_Office    "Office"        <office>    (gGF)   ["Room"] { synonyms="Study" }
Group GF_Son       "Oliver's Room" <boy_1>     (gGF)   ["Bedroom"] { synonyms="Oli's Room" }
Group GF_Daughter  "Amelia's Room" <girl_1>    (gGF)   ["Bedroom"] { synonyms="Amy's Room" }
Group GF_Bed       "Bedroom"       <bedroom>   (gGF)   ["Bedroom"] { synonyms="Master Bedroom" }
Group GF_Corridor  "Corridor"      <corridor>  (gGF)   ["Corridor"]
```

Figura 13. Finalización del contenido sitemap

Luego se procedió definir los objetos de IoT que se utilizaron en cada uno de los pisos. En este caso, se dividió lo objetos en grupos de acuerdo con sus características. Se inició por los objetos de tipo bombillas de luz, que fueron de tipo switch, como se ilustra en la Figura 14. Esto significa que dichos objetos tenían dos estados, encendido y apagado.

```
/* Lights */
Dimmer Light_FF_Living_Table_12       "Table"         (FF_Living, Lights)     ["Light", "Lighting"]
Switch Light_FF_Corridor_Ceiling_14   "Ceiling"       (FF_Corridor, Lights)   ["Light"]
Switch Light_FF_Kitchen_Ceiling_11    "Ceiling"       (FF_Kitchen, Lights)    ["Light"]
Switch Light_FF_Kitchen_Table_11      "Table"         (FF_Kitchen, Lights)    ["Light"]
Switch Light_FF_Corridor_Wardrobe_14  "Wardrobe"      (FF_Corridor, Lights)   ["Light"]
Switch Light_FF_Toilet_Ceiling_13      "Ceiling"      (FF_Toilet, Lights)     ["Light"]
Switch Light_FF_Toilet_Mirror_13       "Mirror"       (FF_Toilet, Lights)     ["Light"]

Switch Light_GF_Bath_Ceiling_24       "Ceiling"       (GF_Bath, Lights)       ["Light"]
Switch Light_GF_Bath_Mirror_24        "Mirror"        (GF_Bath, Lights)       ["Light"]
Switch Light_GF_Corridor_Ceiling_25   "Corridor"      (GF_Corridor, Lights)   ["Light"]
Switch Light_GF_Office_Ceiling_23     "Ceiling"       (GF_Office, Lights)     ["Light"]
Switch Light_GF_Son_Ceiling_22        "Ceiling"       (GF_Son, Lights)        ["Light"]
Switch Light_GF_Daughter_Ceiling_21   "Ceiling"       (GF_Daughter, Lights)   ["Light"]
Switch Light_GF_Bed_Ceiling_21        "Ceiling"       (GF_Bed, Lights)        ["Light"]
```

Figura 14. Finalización del contenido sitemap

A continuación, se crearon los sensores de iluminación. Éstos fueron creados por cada una de las habitaciones en donde se disponía de un objeto para medir el nivel de iluminación. En la Figura 15 se muestran los objetos de este tipo que fueron creados e integrados en el archivo ítems del proyecto.

```
/* ilumination*/
Switch Ilumination_GF_Ilumination_23   "Ilumination"         (GF_Office, Lights)    ["Light"]
Switch Ilumination_GF_Ilumination_22   "Ilumination"         (GF_Son, Lights)    ["Light"]
Switch Ilumination_GF_Ilumination_21   "Ilumination"         (GF_Daughter, Lights)    ["Light"]
Switch Ilumination_GF_Ilumination_12   "Ilumination"         (FF_Living , Lights)    ["Light"]
Switch Ilumination_FF_Ilumination_43   "Ilumination"         (FF_Outside, Lights)    ["Light"]
```

Figura 15. Sensores Lumínicos

Seguido, se crearon los sensores de movimiento denominados “motion” los cuales sirven para detectar si hay movimiento en las habitaciones donde se instalaron. La lista de los sensores de este tipo se muestra en la Figura 16.

```
/* motion*/
Switch motion_GF_motion_12      "motion"         (FF_Living,motion)    ["motion"]
Switch motion_GF_motion_11      "motion"         (FF_Kitchen,motion)    ["motion"]
Switch motion_GF_motion_25      "motion"         (FF_Corridor,motion)    ["motion"]
Switch motion_GF_motion_23      "motion"         (GF_Office,motion)    ["motion"]
Switch motion_GF_motion_22      "motion"         (GF_Son,motion)    ["motion"]
Switch motion_GF_motion_21      "motion"         (GF_Daughter,motion)    ["motion"]
Switch motion_GF_motion_32      "motion"         (GF_Corridor,motion)    ["motion"]
```

Figura 16. Sensores de movimiento

Otro de los objetos creados dentro de algunas de las habitaciones del hogar digital es la calefacción (“Heating”). Estos objetos permitieron realizar el control de la temperatura por cada habitación donde se encontraban instalados. En la Figura 17 se muestran los elementos de este tipo que fueron creados.

```
/* Heating */
Switch Heating_FF_Corridor14          "Corridor"      <heating>   (FF_Corridor, Heating)  ["HVAC"]
Switch Heating_FF_Toilet13            "Toilet"        <heating>   (FF_Toilet, Heating)    ["HVAC"]
Switch Heating_FF_Living12            "Livingroom"    <heating>   (FF_Living, Heating)    ["HVAC"]
Switch Heating_FF_Kitchen11           "Kitchen"       <heating>   (FF_Kitchen, Heating)   ["HVAC"]

Switch Heating_GF_Bath24              "Bath"          <heating>   (GF_Bath, Heating)      ["HVAC"]
Switch Heating_GF_Office23            "Office"        <heating>   (GF_Office, Heating)    ["HVAC"]
Switch Heating_GF_Son22               "Oliver's Room" <heating>   (GF_Son, Heating)       ["HVAC"]
Switch Heating_GF_Daughter21          "Amelia's Room" <heating>   (GF_Daughter, Heating)  ["HVAC"]
```

Figura 17. Sensores Térmicos

Otros de los objetos creados fueron las ventanas y las persianas. Las ventanas (Windows) se instalaron en la cocina, terraza, baños y oficina. En estos mismos espacios se instalaron también los objetos de tipo persianas enrollables denominada “Rollershutters”. Ambos tipos de objetos se muestran en la Figura 18, en el mismo orden en que fueron descritos.

```
/* Rollershutters */
Rollershutter Shutter_GF_Toilet13         "Toilet"        (GF_Toilet, Shutters)   ["Blinds"]
Rollershutter Shutter_GF_Kitchen11        "Kitchen"       (GF_Kitchen, Shutters)  ["Blinds"]
Rollershutter Shutter_GF_Living12         "Livingroom"    (GF_Living, Shutters)   ["Blinds"]

Rollershutter Shutter_FF_Bed21            "Bedroom"       (FF_Bed, Shutters)      ["Blinds"]
Rollershutter Shutter_FF_Bath24           "Bath"          (FF_Bath, Shutters)     ["Blinds"]
Rollershutter Shutter_FF_Office_Window23  "Office Window" (FF_Office, Shutters)   ["Blinds"]
Rollershutter Shutter_FF_Office_Door23    "Office Door"   (FF_Office, Shutters)   ["Blinds"]

Switch Shutter_all

/* Windows */
Contact Window_FF_Frontdoor14     "Frontdoor [MAP(en.map):%s]"     <frontdoor>  (FF_Corridor, Windows)    ["FrontDoor"
Contact Window_FF_Kitchen11       "Kitchen [MAP(en.map):%s]"                    (FF_Kitchen, Windows)     ["Window"]
Contact Window_FF_Living12        "Terrace door [MAP(en.map):%s]"  <door>       (FF_Living, Windows)      ["Door"]
Contact Window_FF_Toilet13        "Toilet [MAP(en.map):%s]"                     (FF_Toilet, Windows)      ["Window"]

Contact Window_GF_Bath24          "Bath [MAP(en.map):%s]"                       (GF_Bath, Windows)        ["Window"]
Contact Window_GF_Bed21           "Bedroom [MAP(en.map):%s]"                    (GF_Bed, Windows)         ["Window"]
Contact Window_GF_Office_Window23 "Office Window [MAP(en.map):%s]"              (GF_Office, Windows)      ["Window"]
Contact Window_GF_Office_Door23   "Balcony Door [MAP(en.map):%s]"  <door>       (GF_Office, Windows)      ["Door"]
```

Figura 18. Objetos "Rollershutters"

A continuación, se crearon los sensores de temperatura que permitieron medir la temperatura en el escenario modelado. Como se muestra en la Figura 19, estos sensores se desplegaron en el corredor, baños y habitaciones.

```
/* Indoor Temperatures */
Number Temperature_FF_Corridor14  "Corridor [%.1f °C]"        <temperature>   (Temperature, FF_Corridor)  ["Temperatur
Number Temperature_FF_Toilet13    "Toilet [%.1f °C]"          <temperature>   (Te9 "Measurement", "CurrentTemperature"
Number Temperature_GF_Bath24      "Bath [%.1f °C]"            <temperature>   (Temperature, GF_Bath)      ["Temperatur
Number Temperature_GF_Office23    "Office [%.1f °C]"          <temperature>   (Temperature, GF_Office)    ["Temperatur
Number Temperature_GF_Son22       "Oliver's Room [%.1f °C]"   <temperature>   (Temperature, GF_Son)       ["Temperatur
Number Temperature_GF_Daughter21  "Amelia's Room [%.1f °C]"   <temperature>   (Temperature, GF_Daughter)  ["Temperatur
Number Temperature_GF_Bed2       "Bedroom [%.1f °C]"         <temperature>   (Temperature, GF_Bed)       ["Temperature
```

Figura 19. Sensores de temperatura

Por último, un sensor de gas fue ubicado en la cocina. Este sensor se complementa de un objeto de tipo alarma que se enciende cuando el sensor de humo se activa ante la presencia de incendios. La descripción es ambos ítems se muestra en la Figura 20.

```
/* gas*/
Number Gas_GF_gas_3  "Gas [%.1f ºC ]"     <gas>   (gas, FF_Kitchen)  ["Temperature", "Measurement"]


/* alarma*/
Switch Heating_GF_Living_Alarma_e43       "Alarma"    <soundvolume>   (GF_Outside, Heating)    ["alarma"]
```

Figura 20. Sensores auxiliares

3.4. Agentificación del ecosistema de IoT

El proceso de ligar un agente con cada uno de los objetos de IoT, o agentificación del IoT, se realizó con el objetivo de que los objetos de IoT, tradicionalmente objetos pasivos, se convirtieran en objetos proactivos que actuaran en función de los cambios del entorno (ecosistema de IoT, y para este caso particular el hogar digital conectado).

Como se observa en la Figura 21, los usuarios dejan de controlar directamente los objetos. Su comportamiento está principalmente dirigido por agentes, que son entidades autónomas. Esto les permite a los agentes que tengan la capacidad de controlar las acciones propias en el hogar digital conectado en base a metas y objetivos que persiguen en cada momento.

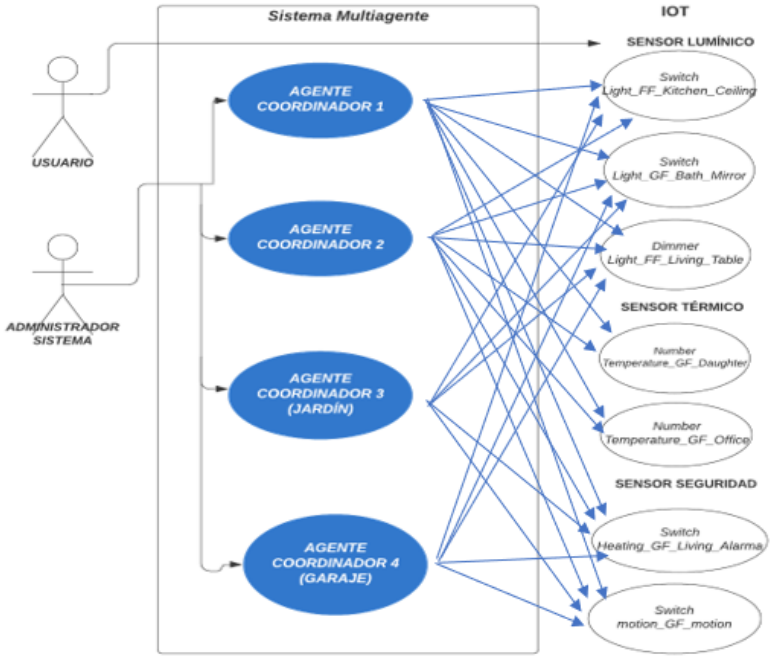

Figura 21. Diagrama UML del sistema domótico

Además, los agentes brindan también racionalidad ya que es posible que ejecuten comportamientos de acuerdo con datos que proporcionan otros agentes vinculados a objetos de IoT. Para el caso del confort lumínico, el agente coordinador de llevar a cabo este control sobre el escenario puede evaluar el nivel de iluminación de las salas de la casa y en base a si existe presencia de personas, encender las bombillas de forma automática, en concordancia con la estación del año u hora del día. En el caso del confort

térmico, se evalúa la temperatura ambiente de casa habitáculo mediante los sensores instalados, para encender el aire acondicionado o calefacción dependiendo del valor de temperatura obtenido en un rango de valores dados por el usuario. Finalmente, en el caso del confort de seguridad, se evalúa el control de movimiento en cada habitación mediante el sensor de movimiento, para en caso de detectar presencia, se activen las luminarias. En todos los casos, la proactividad inherente a los agentes logra también mantener un control exhaustivo en un ambiente cambiante como es un hogar domótico basados en IoT.

3.5. Sistema multiagente

Los agentes que se implementaron en este sistema se agrupan en coordinadores principales que en este caso fueron cuatro: el coordinador del piso 1 (Figura 22.a), el coordinador del piso 2 (Figura 22.b), el coordinador del jardín (Figura 22.c) y el coordinador del garaje (Figura 22.d). Dentro de cada coordinador principal se generaron subgrupos de coordinadores como coordinador de la sala o del cuarto máster. Y como coordinadores esenciales se tiene el lumínico, el térmico y el de seguridad.

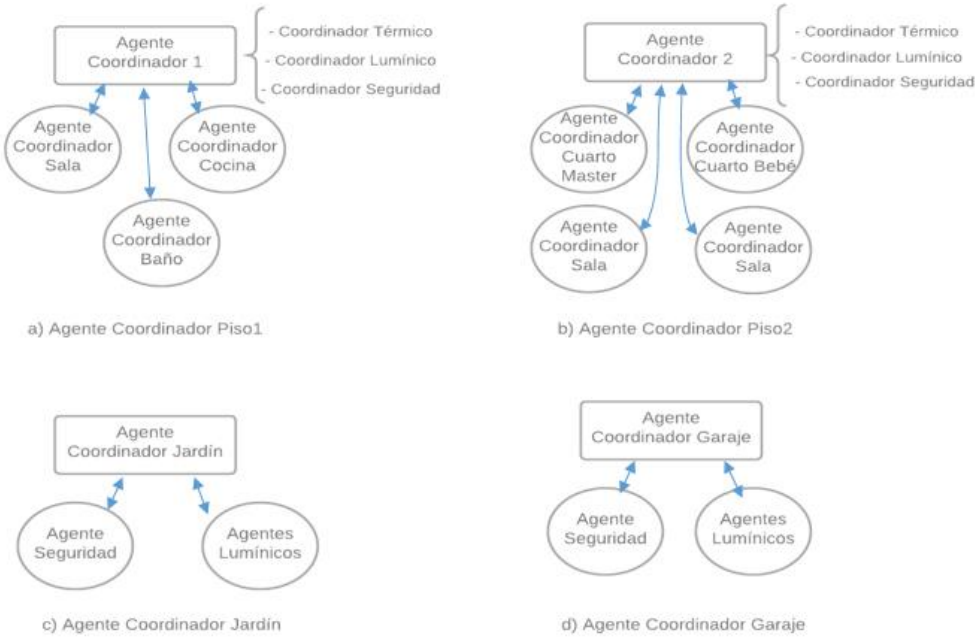

Figura 22. Interacciones de los agentes del sistema multiagente

De manera más específica, los agentes previamente ilustrados, describen su comportamiento como sigue:

3.6. Comportamiento del agente coordinador uno

El objetivo del agente coordinador del primer piso es el de brindar un confort tanto térmico, lumínico y de seguridad al hogar inteligente de tal manera que el usuario se mantenga cómodo y en óptimas condiciones.

Como se muestra en la Figura 23, el agente coordinador 1, solicita a los distintos subgrupos de agentes el valor que necesita en diferentes circunstancias, ya sea lumínico, térmico o de seguridad. Dependiendo de los valores obtenidos el agente realiza distintas acciones como encendido de luminarias, encendido de aire acondicionado y encendido de alarmas. Estos valores arrojados por los agentes son previamente captados por los objetos de IoT que se encuentran en el hogar inteligente. Este proceso es detallado más adelante.

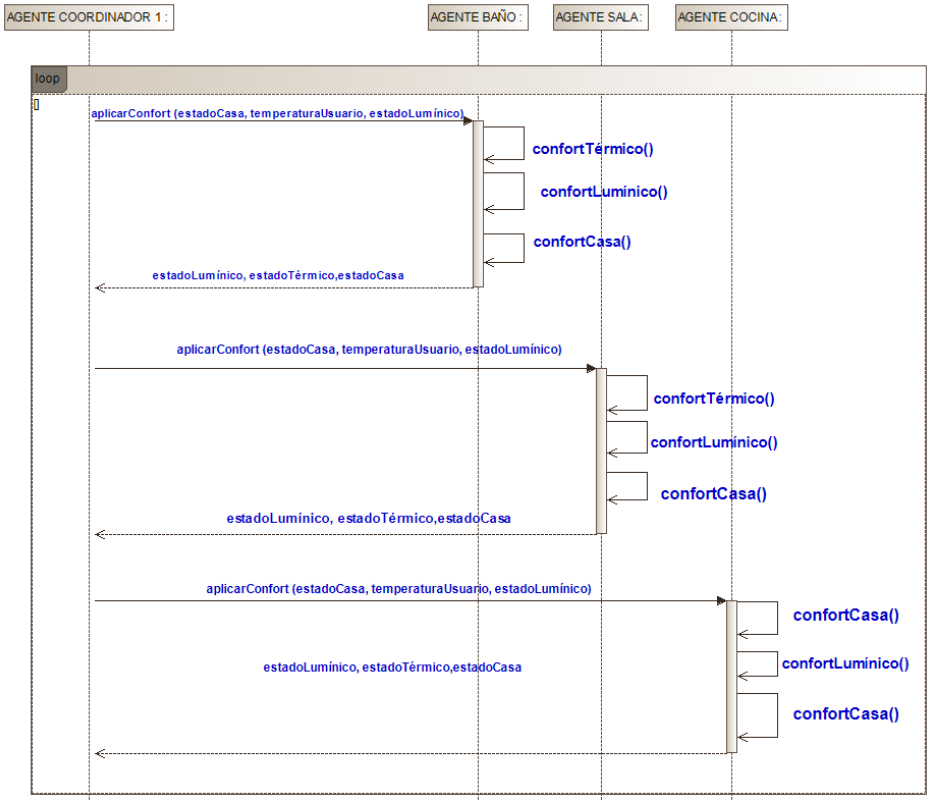

Figura 23. Diagrama de secuencia Coordinador 1

El agente coordinador 2, el agente Jardín y el agente Garaje tienen un comportamiento parecido al agente coordinador 1, con la diferencia que estos dos no

brindan un confort térmico, pero si generan un confort lumínico y de seguridad para el usuario.

3.7. Comportamiento del agente Cocina

El agente cocina contiene múltiples objetos IoT con el objetivo de brindar un mayor confort al usuario en el ámbito lumínico, térmico y de seguridad, tal como se muestra en la Figura 22. En primer lugar, dicho agente solicita al agente de iluminación el estado lumínico de la casa, y posteriormente, solicita el estado de la casa para saber si hay gente o no en el hogar inteligente. Una vez el agente haya obtenido esos valores, solicita y obtiene los valores de los sensores de seguridad del hogar para encender la luminaria de la casa en el caso de que se detecte movimiento en su interior, caso contrario, el estado de la luminaria permanecerá apagado.

En el ámbito lumínico, el agente cocina solicita y recibe los valores de los sensores lumínicos y dependiendo de la luminosidad que hay en el hogar, es decir, si el nivel de iluminación es inferior o igual a 60, se encenderán las bombillas y por consecuente se cerrarán las ventanas, caso contrario, las bombillas se apagarán y se abrirán las ventanas. Se ha trabajado con un rango de iluminación que va de 0 a 100.

En el caso del confort térmico, el agente cocina solicita al sensor de temperatura los valores de temperatura del hogar, en el caso de que el valor de temperatura sea inferior o igual a 25 grados, el aire acondicionado se apagará, caso contrario el aire acondicionado se encenderá.

El último lugar, tenemos el detector de gas dentro de la cocina, el cual el agente cocina activará una alarma en caso de que se detecte fuga de gas.

3.8. Desarrollo del sistema agentificado con agentes JADE

Dentro del desarrollo del hogar inteligente en Jade, se creó un agente coordinador por cada uno de los pisos para que tome decisiones en cada una de ellas. Dichos coordinadores implementaron el comportamiento previamente descritos en la Figura 21. A continuación, se describe la clase que implementó el comportamiento del coordinador 1 y el agente cocina, ambos comportamientos descritos mediante un diagrama de secuencia ilustrado

en la Figura 21 y Figura 22. Sin embargo, se ilustra cómo los agentes que formaron parte del proyecto han sido organizados (Figura 23).

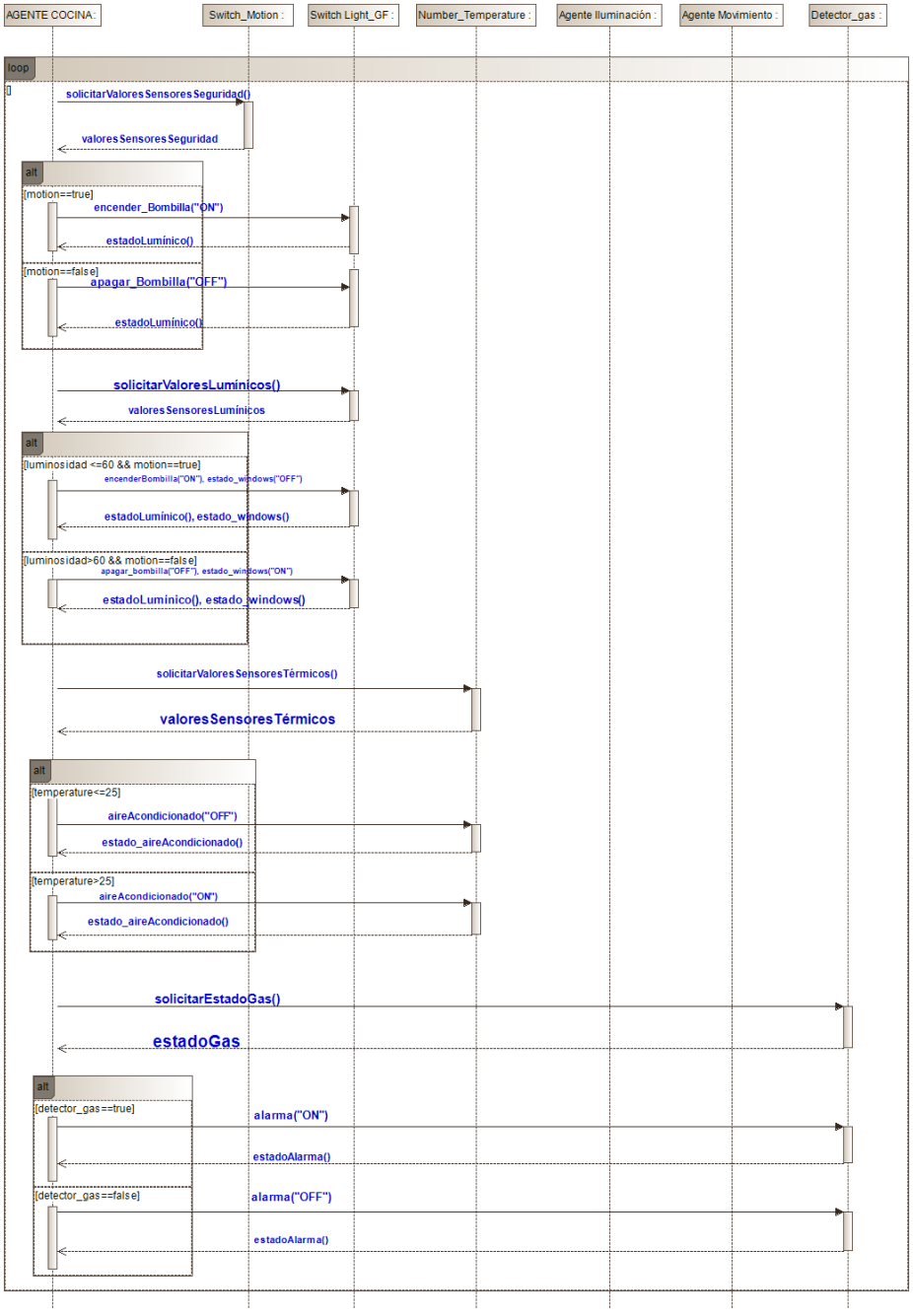

Figura 24. Diagrama de secuencia Agente Cocina

El proyecto incluyó clases que implementaron agentes y comportamientos. En el paquete “Clases”, se almacenaron los agentes actuadores, sensores y coordinadores; estos tres son los tipos de agentes que se implementaron en el proyecto. En el paquete “Plantilla” se implementaron los objetos de tipo actuadores y sensores, los cuales

recopilan la información global de todos los agentes que se crean por cada uno de los componentes que habitan en cada piso y salas. Éstos mismos son los que realizan el envío de información a los coordinadores de cada piso para poder sesionarla. Los agentes coordinadores de las habitaciones de la casa se organizaron en el paquete "Home". Finalmente, los agentes coordinadores de confort se almacenaron en el paquete "Confort".

A continuación, se describe el código a partir del cual se creó el agente coordinador de cocina en JADE.

```
1.  public class CoordinatorKitchen extends Agent{
2.  public int auxiliar=0;
3.  public ArrayList<Coordinator> listacoordinador = new
   ArrayList();
4.  ACLMessage  lista;
5.  private boolean fin=false;
6.  public class ReceptorComportamiento extends SimpleBehaviour{
7.  public void action(){
8.  block();
9.  fin=false;
10.      llenarlista();}
11.      public boolean done(){
12.      return fin;}
13.      public void llenarlista(){
14.       lista = receive();
15.       if(lista!=null){
16.       auxiliar++;
17.      System.out.println("el valor auxiliar es: "+auxiliar);
18.      System.out.println("Lista room1 es: "+lista.getContent());
19.      //extraer la lista que envio el OfthingSensor
20.      java.lang.reflect.Type listType = new
         TypeToken<ArrayList<Sensor>>() {}.getType();
21.      ArrayList<Coordinator> myModelList = new
         Gson().fromJson(lista.getContent(), listType);
22.      listacoordinador=myModelList;}}//fin llenar listas/}/emisor
23.      public class EmisorComporamiento extends CyclicBehaviour{
24.      int contador=1;
25.      private Sensor sensor = new Sensor();
26.      AID id= new AID(); //List<Sensor> listacoordinador;
27.      ACLMessage  ms;
28.      public void action(){
29.      block(999000000);
30.      System.out.println("preparando room1 para emitir");
31.      if(contador<2){
32.      if(auxiliar>=4){
33.      System.out.println("El agente ya está listo");
34.      for (Coordinator sensores : listacoordinador) {
35.      System.out.println("*********");
36.      System.out.println(sensores.getName());
37.      System.out.println("*********");}
38.      opesensor();
39.      contador++;}
40.      public void opesensor(){
```

```
41.       id.setLocalName("home"+contador);
42.       System.out.println("Enviare mensaje al"+id.getLocalName());
43.       ms = new ACLMessage(ACLMessage.INFORM);
44.       ms.setSender(getAID());
45.       ms.addReceiver(id);
46.       //crear una lista a string
47.       String listToJson;
48.       listToJson = new Gson().toJson(listacoordinador);
49.       ms.setContent(listToJson);
50.       send(ms); }
51.       public void setup(){
52.       addBehaviour(new ReceptorComportamiento());
53.       //System.out.println("voy a enviar al home");
54.       addBehaviour(new EmisorComporamiento());
```

Para crear el agente coordinador de la cocina (línea 1), se necesitó heredar de la clase jade.core.Agent. El agente coordinador, como todos los agentes se extiende de la clase Agent, e implementa un compartamiento atómico (Simple Behaviour) que realiza la captura de los paquetes de los diferentes sensores que se encuentran en el hogar domótico. Una vez creado el capturador de paquetes, el método action (línea 7) repite la tarea de capturar paquetes, y almacenarlos en un objeto ArrayList para su uso posterior. Se crea una clase Emisor Comportamiento de tipo CyclicBehaviour, la cual representa un comportamiento cíclico que se debe ejecutar indefinidamente mientras el agente se esté ejecutando. Por último, se crea un objeto (línea 40-52) para almacenar los comportamientos de los sensores en una lista, el cual enviará toda la lista a cada sensor. También, se debieron inicializar los métodos "ReceptorComportamiento" y "EmisorComportamiento"

3.9. Desarrollo del sistema agentificado con agentes SPADE

Con el objetivo de describir el modelo de programación de agentes en SPADE, a continuación, se describe una porción de código que implementa el agente coordinador de cocina, ya descrito anteriormente; pero con JADE.

```
1.      from spade.agent import Agent
2.      from spade.behaviour import OneShotBehaviour
3.      from spade.behaviour import CyclicBehaviour
4.      from spade.message import Message
5.      from Clases.Coordinador import Coordinador
6.      def coordinador2():
7.      for coordinador in listaauxiliar1:
8.      if listaauxiliar1.idc.__eq__("1"):
9.      Coordinador.setIds(listaauxiliar1.ids)
```

```
10.     Coordinador.setName(listaauxiliar1.Name)
11.     Coordinador.setUri(listaauxiliar1.Uri)
12.     Coordinador.settype(listaauxiliar1.type)
13.     Coordinador.setDescripcion(listaauxiliar1.Descripcion)
14.     Coordinador.setMeasure(listaauxiliar1.Measure)
15.     Coordinador.setResourse(listaauxiliar1.Resourse)
16.     #añadir a la lista del coordinador1
17.     listacoordinado1.append(Coordinador)
18.     class Coordinadorkitchen(Agent):
19.     global listacoordinado1
20.     global listaauxiliar1
21.     # Receptor
22.     class ReceptorComportamiento(OneShotBehaviour):
23.     async def run(self):
24.     print("Recibiendo ")
25.     msg = await self.receive(timeout=10)
26.     if msg:
27.     print("Mensaje recibido con contenido: 28.{}".format(msg.body))
28.     #convertir una cadena a una lista
29.     listaauxiliar1(msg)
30.     coordinador2()
31.     else:
32.     print("No se recibió ningún mensaje")
33.     class EmisorComportamiento(CyclicBehaviour):
34.     async def run(self):
35.     print("Enviando lista de sensores al coordinador2")
36.     msg = Message(to="receiver@coordinatorr2")
37.     msg.set_metadata("performative", "inform")
38.     msg.set_metadata("ontology", "myOntology")
39.     msg.set_metadata("language", "OWL-S")
40.     msg.body = listaauxiliar1
41.     #Enviar Mensaje
42.     await self.send(msg)
43.     print("Los sensores y actuadores han sido enviados!")
44.     async def setup(self):
45.     print("SenderAgent started")
46.     self.b = self.ReceptorComportamiento()
47.     self.b = self.EmisorComportamiento()
48.     self.add_behaviour(self.b)
```

Para crear el agente coordinador de la cocina en SPADE (línea 1), se procedió a importar la librería de spade, luego los comportamientos a utilizar y posteriormente se declararon los mensajes con los que se comunicaran los agentes. Una vez que se establezcan las librerías con sus componentes a utilizar se procedió a instanciar la clase Coordinador la cual tiene atributos (id del sensor, id del coordinador, nombre, descripción, uri, recurso, medida y tipo) que permitirán manipular los datos de los actuadores y sensores que habitan en la cocina.

Se establecen los comportamientos a utilizar por el agente que son OneShotBehaviour (línea 2) y CyclicBehaviour (línea 3) para la recepción y emisión de mensajería.

El método coordinador 2 (línea 6) realiza la función de obtención de los elementos de la lista que recibe el agente coordinador de la cocina, los compara si pertenecen a la cocina para luego agregarlos a la lista del coordinador de la cocina.

El agente CoordinadorKitchen implementa un comportamiento OneShotBehaviour (Simple Behaviour) ReceptorComportamiento (línea 22), el cual realiza la acción de receptar los actuadores y sensores que se crearon para luego manipularlos en su ambiente de cocina. El agente CoordinadorKitchen también implementa un comportamiento CyclicBehaviour (línea 33) para el envío de los actuadores y sensores al coordinador2 mediante el uso de json convirtiendo las listas en cadenas para poder ser enviadas.

Por último en el método setup (línea 44-48) del coordinador se agregan sus comportamientos para luego proceder a su ejecución.

3.10. Comparación del sistema IoT agentificado con JADE y SPADE

Como parte del trabajo, se planteó una evaluación de los sistemas desarrollados en JADE y SPADE. La evaluación se centró en tres aspectos, esto es, seguridad del sistema, calidad de las acciones de control y el rendimiento (o desempeño) del sistema. A continuación, se describen detalles de la evaluación realizada. Como se muestra en la Figura 25, a partir de la evaluación de las métricas mencionadas se plantearon las conclusiones y recomendaciones para mejorar los sistemas desarrollados.

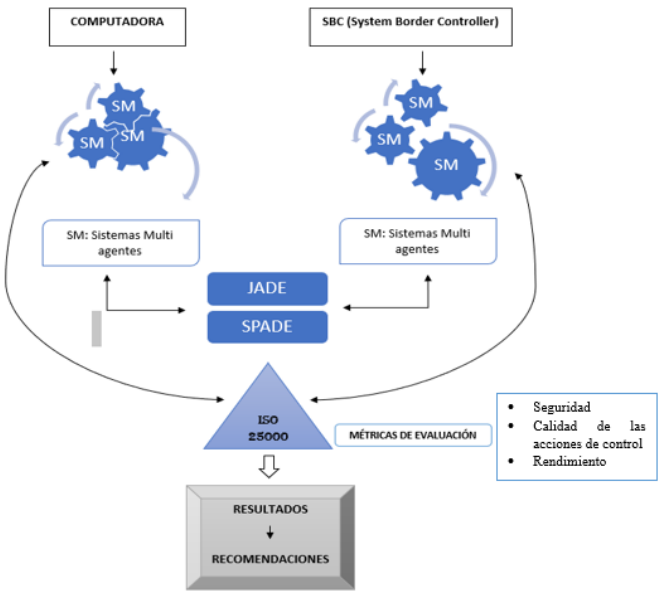

Figura 25. Diseño del funcionamiento de la investigación realizada

3.10.1. Seguridad

La seguridad es uno de los principales inconvenientes del IoT, es por ello, que se debe tener la capacidad de proteger información y datos de tal forma que sean incapaces de ser leídos o modificados por otras personas o sistemas no autorizados, dentro de la cual la autenticidad, la confidencialidad y la responsabilidad son sus principales características.

Dentro de los servicios de la plataforma openHAB, se encuentra openHAB Cloud, el cual es un servicio en la nube complementario y un backend para el software de automatización del hogar inteligente. El backend de openHAB brinda acceso remoto y seguro permitiendo a los usuarios de la plataforma monitorear, controlar y dirigir de forma remota sus hogares inteligentes mediante el Internet.

OpenHAB tiene un servidor web incorporado, que escucha en el puerto 8080 para HTTP y 8443 para solicitudes HTTPS. El protocolo HTTPS (protocolo de transferencia de hipertexto) es un protocolo de comunicación de Internet que protege la integridad y la confidencialidad de los datos de los usuarios entre sus ordenadores y el sitio web. Es por eso, que openHAB se encarga de la seguridad de los proyectos, manejando todos los mecanismos relacionados con la seguridad en la nube, brindando al usuario un confort en un ámbito importante como es la seguridad e integridad de los datos. No se ha ahondado esfuerzos para gestionar el aspecto de la seguridad debido a que se aprovecha la gestión que hace openHAB a través de su servicio de nube. Tampoco era un objetivo de este trabajo proponer o mejorar los mecanismos de seguridad ni para OpenHAB ni para los agentes.

3.10.2. Calidad de las acciones de control

Por medio los agentes inteligentes, los usuarios del hogar inteligente lograron tener un confort tanto lumínico, térmico y de seguridad, gracias a la comunicación exitosa que logran los diferentes agentes para obtener óptimos resultados teniendo un control y confort total del hogar. Esto permite obtener una calidad de acciones de control, que como se observa en la Tabla 2. En cada habitáculo los agentes actúan con varias acciones dependiendo de los valores obtenidos por los sensores implementados. Cabe recalcar que todas las acciones de control visualizados en la Tabla 2, son iguales tanto para JADE como para SPADE.

Tabla 2 Acciones de control ejercidas sobre el hogar inteligente

	Acciones de Control	
Habitáculos	**Reglas**	**Funcionalidad**
Cocina	***Confort Lumínico*** Agente: Switch_Light_Kitchen Objetivo: Encender/Apagar luces dependiendo de la luminosidad. Regla 1: Luminosidad <=60 AND Motion ==true THEN Encender Bombilla Regla 2: Luminosidad >60 THEN Apagar Bombilla ***Confort Térmico*** Agente: Number_Temperature_Kitcken Objetivo: Encender/Apagar el aire acondicionado dependiendo de la temperatura interna que haya. Regla 1: Temperatura >25 THEN Encender Aire Acondicionado Regla 2: Temperatura <=25 THEN Apagar Aire Acondicionado ***Confort Seguridad*** Agente: Switch_motion_GF_1 Objetivo: Encender/Apagar luces y alarma en caso de detectar movimiento dentro de la casa o cuando se detecte gas. Regla 1: Motion ==true THEN Encender Bombillas && Cerrar Ventanas Regla 2: Detector_gas==true THEN Encender Alarma	Cumple objetivo
Sala	***Confort Lumínico*** Agente: Dimmer Light_FF_Living_Table Objetivo: Encender/Apagar luces dependiendo de la luminosidad. Regla 1: Luminosidad <=60 AND Motion ==true THEN Encender Bombilla Regla 2: Luminosidad >60 THEN Apagar Bombilla ***Confort Térmico*** Agente: Number_Temperature_FF_Living Objetivo: Encender/Apagar el aire acondicionado dependiendo de la temperatura interna que haya. Regla 1: Temperatura >25 THEN Encender Aire Acondicionado Regla 2: Temperatura <=25 THEN Apagar Aire Acondicionado ***Confort Seguridad*** Agente: Switch_motion_GF_2 Objetivo: Encender/Apagar luces y alarma en caso de detectar movimiento dentro de la casa Regla 1: Motion ==true THEN Encender Bombillas	Cumple objetivo

Baños	***Confort Lumínico*** Agente:Switch Light_FF_Bath Objetivo: Encender/Apagar luces dependiendo de la luminosidad. Regla 1: Luminosidad <=60 AND Motion ==true THEN Encender Bombilla Regla 2: Luminosidad >60 THEN Apagar Bombilla ***Confort Seguridad*** Agente: Switch_motion_GF_3 Objetivo: Encender/Apagar luces y alarma en caso de detectar movimiento dentro de la casa Regla 1: Motion ==true THEN Encender Bombillas	Cumple objetivo
Cuarto Máster	***Confort Lumínico*** Agente: Switch_Light_Daughter_Bed Objetivo: Encender/Apagar luces dependiendo de la luminosidad. Regla 1: Luminosidad <=60 AND Motion ==true THEN Encender Bombilla && Cerrar persianas Regla 2: Luminosidad >60 THEN Apagar Bombilla && Abrir persianas ***Confort Térmico*** Agente: Number_Temperature_Daughter_Bed Objetivo: Encender/Apagar el aire acondicionado dependiendo de la temperatura interna que haya. Regla 1: Temperatura >25 THEN Encender Aire Acondicionado Regla 2: Temperatura <=25 THEN Apagar Aire Acondicionado ***Confort Seguridad*** Agente: Switch_motion_GF_4 Objetivo: Encender/Apagar luces y alarma en caso de detectar movimiento dentro de la casa o cuando se detecte gas. Regla 1: Motion ==true THEN Encender Bombillas && Cerrar Ventanas	Cumple objetivo
Cuarto Bebé	***Confort Lumínico*** Agente: Switch_Light_Son_Ceiling Objetivo: Encender/Apagar luces dependiendo de la luminosidad. Regla 1: Luminosidad <=60 AND Motion ==true THEN Encender Bombilla && Cerrar persianas Regla 2: Luminosidad >60 THEN Apagar Bombilla && Abrir persianas ***Confort Térmico*** Agente: Number_Temperature_Son_Ceiling Objetivo: Encender/Apagar el aire acondicionado dependiendo de la temperatura interna que haya.	Cumple objetivo

	Regla 1: Temperatura >25 THEN Encender Aire Acondicionado Regla 2: Temperatura <=25 THEN Apagar Aire Acondicionado ***Confort Seguridad*** Agente: Switch_motion_GF_5 Objetivo: Encender/Apagar luces y alarma en caso de detectar movimiento dentro de la casa o cuando se detecte gas. Regla 1: Motion ==true THEN Encender Bombillas && Cerrar Ventanas	
Oficina	***Confort Lumínico*** Agente: Switch_Light_Office Objetivo: Encender/Apagar luces dependiendo de la luminosidad. Regla 1: Luminosidad <=60 AND Motion ==true THEN Encender Bombilla && Cerrar persianas Regla 2: Luminosidad >60 THEN Apagar Bombilla && Abrir persianas ***Confort Térmico*** Agente: Number_Temperature_Office Objetivo: Encender/Apagar el aire acondicionado dependiendo de la temperatura interna que haya. Regla 1: Temperatura >25 THEN Encender Aire Acondicionado Regla 2: Temperatura <=25 THEN Apagar Aire Acondicionado ***Confort Seguridad*** Agente: Switch_motion_GF_6 Objetivo: Encender/Apagar luces y alarma en caso de detectar movimiento dentro de la casa o cuando se detecte gas. Regla 1: Motion ==true THEN Encender Bombillas && Cerrar Ventanas	Cumple objetivo
Garaje	***Confort Lumínico*** Agente:Switch Light_G_Garage Objetivo: Encender/Apagar luces dependiendo de la luminosidad. Regla 1: Luminosidad <=60 AND Motion ==true THEN Encender Bombilla Regla 2: Luminosidad >60 THEN Apagar Bombilla ***Confort Seguridad*** Agente: Switch_motion_GF_7 Objetivo: Encender/Apagar luces y alarma en caso de detectar movimiento dentro de la casa Regla 1: Motion ==true THEN Encender Bombillas	Cumple objetivo
Jardín	***Confort Lumínico*** Agente:Switch Light_G_Garden	

	Objetivo: Encender/Apagar luces dependiendo de la luminosidad. Regla 1: Luminosidad <=60 AND Motion ==true THEN Encender Bombilla Regla 2: Luminosidad >60 THEN Apagar Bombilla ***Confort Seguridad*** Agente: Switch_motion_GF_8 Objetivo: Encender/Apagar luces y alarma en caso de detectar movimiento dentro de la casa Regla 1: Motion ==true THEN Encender Bombillas	Cumple objetivo

Tabla 3. Acciones de Control del hogar inteligente

3.10.3. Rendimiento de procesamiento

Los resultados que se muestran en la Tabla 4, se obtuvieron después de veinte ejecuciones de cada tipo de agentes en el mismo escenario. Se empleó una computadora HP con un procesador i5 1.6GHz, 4GB de RAM, sistema operativo Windows 10 Home. 64 bits, Java SE 1.8.0, JADE v4.3. Para implementar el sistema en JADE se utilizó Java SE Developmet Kit 11.0.9 y para el marco de trabajo SPADE se utilizó la versión Python 3.6.

Para evaluar el rendimiento del sistema del hogar inteligente, la Tabla 4 muestra el comportamiento del sistema en términos de la integridad de las comunicaciones entre los distintos agentes y el ecosistema de servicios.

Las primeras tres columnas de la Tabla 4, muestran las diferentes entidades involucradas en el proceso de solicitud/respuesta que se realizan en el sistema del hogar inteligente. Los resultados obtenidos de la experimentación muestran el tiempo total requeridos para la ejecución de cada uno de los procesos de solicitud / respuesta llevados a cabo para cumplir los objetivos del agente de confort correspondiente (Lumínico, Térmico y de Seguridad). Cada experimentación fue realizada durante 20 iteraciones. La comunicación agente-agente de los agentes lumínicos requirió en promedio 4.32 ms. En el caso de los agentes térmicos, el promedio que se requirió para la comunicación es de 6.72 ms. Y finalmente, en el caso de los agentes de seguridad, para su comunicación se requerían 3.96 ms de promedio.

Tabla 4. Evaluación del proceso ejecutado por los agentes de confort

#	Entidad Solicitante	Entidad Responsable	Tiempo de Ejecución JADE (ms)	Tiempo de Ejecución SPADE (ms)
1	Agente de Confort Lumínico	Switch Light_FF_Kitchen_Ceiling	4.35 ms	4.52 ms
2	Agente de Confort Lumínico	Switch Light_FF_Kitchen_Table	4.33 ms	4.44 ms
3	Agente de Confort Lumínico	Switch Light_FF_Toilet_Ceiling	4.35 ms	4.37 ms
4	Agente de Confort Lumínico	Switch Light_FF_Toilet_Mirror	4.30 ms	4.29 ms
5	Agente de Confort Lumínico	Switch Light_FF_Corridor_Ceiling	4.29 ms	4.28 ms
6	Agente de Confort Térmico	Number Temperature_FF_Corridor	6.69 ms	6.88 ms
7	Agente de Confort Térmico	Number Temperature_FF_Toilet	6.72 ms	6.79 ms
8	Agente de Confort Térmico	Number Temperature_GF_Bath	6.67 ms	6.75 ms
9	Agente de Confort Térmico	Number Temperature_GF_Kitchen	6.82 ms	6.88 ms
10	Agente de Confort Seguridad	Contact Window_FF_Kitchen	5.28 ms	5.33 ms
11	Agente de Confort Seguridad	Contact Window_FF_Toilet	5.23 ms	5.39 ms
12	Agente de Confort Seguridad	Contact Window_FF_Living	5.20 ms	5.30 ms
13	Agente de Confort Seguridad	Number Gas_GF_gas	2.86 ms	2.89 ms
14	Agente de Confort Seguridad	Switch motion_GF_motion	2.33 ms	2.37 ms
15	Agente de Confort Seguridad	Switch motion_GF_motion_kitchen	2.42 ms	2.47 ms

CAPÍTULO V: DISCUSIÓN

La plataforma OpenHab brinda una gran cantidad de ventas en cuanto a la creación de un sistema domótico, ofreciendo una amplia interoperabilidad entre las plataformas existentes sin generar un gasto económico alto, dando la posibilidad al usuario de desarrollar sistemas flexibles para satisfacer distintas necesidades del usuario. Además, hay que tener en cuenta que es una plataforma cuyo software es de código abierto centrada en la automatización de sistemas de vivienda, dispositivos y tecnologías dentro de una misma solución. OpenHAB se caracteriza por ser una plataforma flexible y personalizable para los usuarios, que tiene la capacidad de comunicarse electrónicamente con dispositivos inteligentes y no tan inteligentes, realizando acciones definidas por el usuario y proporciona páginas web con información definida por el usuario, además de herramientas definidas para interactuar con todos los dispositivos. Cabe recalcar que openHAB tiene capacidad de integrar otras tecnologías como los agentes inteligentes tanto el lenguaje de programación Java como en Python, y pueden ser implementados tanto el sistema operativo Windows como en Linux.

A través de la API REST de openHAB, otros programas pueden acceder fácilmente a la mayoría de los aspectos del sistema openHAB. Esto incluye, por ejemplo, el acceso a todos los datos relacionados con Elementos, Cosas y Vinculaciones, así como la capacidad que tiene para generar acciones que pueden modificar el estado de los Elementos o influir en el comportamiento de otros elementos de openHAB. Gracias a la API se pueden realizar varias acciones como, recuperar datos de openHAB de aplicaciones externas, activar eventos en openHAB desde aplicaciones externas, conocer estados y parámetros de los elementos de la plataforma e interactuar con openHAB desde otros programas en varios lenguajes de programación.

Las reglas de control son un pilar importante dentro de la automatización del hogar, ya que el usuario las define a su conveniencia pudiendo así determinar horarios o condiciones para que se realice una acción. Estas reglas consisten en encender las bombillas (luminaria) de los habitáculos del hogar dependiendo del valor de luminosidad que se obtiene de los sensores, brindando así un confort lumínico en toda la casa. También se trata de obtener un confort térmico, que consiste en encender o apagar el aire

acondicionado dependiendo de la temperatura interna del hogar por medio de los sensores de temperatura. Y por medio de reglas de control, también de obtiene un confort de seguridad a través de los sensores de movimiento, detección de gas y alarmas, que se activan cuando se detecta presencia de humanos o cuando se detecta fuga de gas en la cocina del hogar.

En cuanto a las métricas evaluadas, la seguridad es un tema de gran importancia en la actualidad ya que ocurren robos de información diariamente a nivel mundial. Es por eso, que openHAB cuenta con una conexión segura, creando una conexión VPN a la red doméstica del usuario. Además, openHAB contiene una instancia llamada openHAB Cloud la cual crea una conexión tipo túnel que reenvía todas las solicitudes a través de este túnel, es decir, la plataforma se encarga de la seguridad del sistema. La calidad y la eficiencia de las acciones de control del sistema, permiten alcanzar un confort total al usuario, ya que basándose en las evaluaciones realizadas a dichas acciones, se obtuvieron resultados positivos a la hora de ejecutar los agentes inteligentes implementados en el hogar inteligente, generando así un control seguro, rápido y de calidad.

CAPÍTUCULO VI: CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

Esquematizando los resultados obtenidos en esta investigación, se puede percibir que se cumplieron con los objetivos mencionados al principio, lográndose palpar de las fuentes bibliográficas, de cómo se maneja la plataforma de OpenHab y sus distintas funcionalidades con la comunicación exitosa de los agentes mediante el entorno NetBeans y el lenguaje de programación Java.

La implementación del lenguaje de programación Java para trabajar con la plataforma OpenHab es óptima ya que implementa la tecnología de JADE que permite conectar a la plataforma por medio de librerías REST de una manera sencilla y la manipulación de agentes por mensajería de subida para la comunicación de todos estos agentes para conseguir el confort en un hogar.

La implementación de la plataforma OpenHab permitió la correcta monitorización y control del hogar domótico aplicando la inteligencia con la programación de automatizaciones de todo nivel para así conseguir una mejora en la calidad de vida y confort de los usuarios, donde con el paso de los años se convertirán en imprescindible para los seres humanos.

Cabe destacar la cantidad de propiedades que se pueden personalizar y ordenar al controlador que posee la plataforma OpenHab, así como el extenso abanico de dispositivos acordes con los enlaces (puede ser interfaz o extensiones HTTP e IFTTT) que brindan la facilidad de controlar y captar todo tipo de dispositivos IoT.

6.2. Recomendaciones

Se recomienda utilizar la plataforma OpenHab ya que una de las características más dinámicas y llamativas con las que cuenta esta plataforma es la facilidad de personalizar aspectos del diseño del hogar domótico, como, por ejemplo, agregar o eliminar elementos dentro de una habitación, colocación de iluminación o implementación de sensores internos o externos. El aprendizaje que genera esta plataforma desde que se decide instalar

es positivo, ya que el usuario puede mejorar el sistema que está realizando sin necesidad de esperar una actualización del sistema, sino que él mismo puede hacer todo el diseño del hogar y modificar prácticamente todo en cualquier momento y al gusto del usuario.

A los usuarios que deseen trabajar con la plataforma OpenHab y realizar sistemas multi agentes, se recomienda trabajar con el lenguaje de programación Java, porque existe más información acerca de la comunicación y relación de este lenguaje con la plataforma OpenHab.

BIBLIOGRAFÍA

[1] H. Abbas, S. Shaheen, and M. Amin, "Providing a transparent dynamic organization technique for efficient aggregation of multiple JADE agent platforms," in *2018 International Conference on Innovative Trends in Computer Engineering (ITCE)*, 2018, pp. 100–108.

[2] B. Xu, "Design of platform for performance testing based on JADE," *Proc. - 2014 6th Int. Conf. Meas. Technol. Mechatronics Autom. ICMTMA 2014*, pp. 251–254, 2014.

[3] B. K. de Freitas, L. Fritzen Venturini, M. A. Domingues, M. Augusto da Rosa, and D. Issicaba, "Exploiting PADE to the Simulation of Multiagent Restoration Actions," in *2019 11th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*, 2019, pp. 1–5.

[4] L. Ribeiro, A. Rocha, and J. Barata, "A study of JADE's messaging RTT performance using distinct message exchange patterns," in *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, 2013, pp. 7410–7415.

[5] I. Lee and K. Lee, "The Internet of Things (IoT): Applications, investments, and challenges for enterprises," *Bus. Horiz.*, vol. 58, no. 4, pp. 431–440, Jul. 2015.

[6] M. Conoscenti, A. Vetro, and J. C. De Martin, "Blockchain for the Internet of Things: A systematic literature review," in *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, 2016, pp. 1–6.

[7] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1125–1142, Oct. 2017.

[8] S. Li, & Li, D. Xu, and S. Zhao, "The internet of things: a survey."

[9] P. Roa, C. Morales, and P. Gutiérrez, "Norma ISO/IEC 25000," *Tecnol. Investig. y Acad.*, vol. 3, no. 2, pp. 27–33, 2015.

[10] F. Khodadadi, A. V. Dastjerdi, and R. Buyya, "Internet of Things: an overview," in *Internet of Things*, Elsevier, 2016, pp. 3–27.

[11] . V. K. B., S. L. Joshi, and S. H. Barshikar, "A Survey on Internet of Things," *Int. J. Comput. Sci. Eng.*, vol. 6, no. 12, pp. 492–496, 2019.

[12] J. Hangki, Y. Inhwan, and R. Intae, "The internet of everything based on energy efficient P2P transmission technology with Bluetooth low energy."

[13] P. P. Ray, "A survey on Internet of Things architectures," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 30, no. 3, pp. 291–319, 2018.

[14] X. Chen, K. Choi, and K. Chae, "A Secure and Efficient Key Authentication using Bilinear Pairing for NFC Mobile Payment Service," *Wirel. Pers. Commun.*, vol. 97, no. 1, 2017.

[15] N. Shah and P. S. Sundar, "Smart Electric Meter Using LoRA Protocols and lot

applications," in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 2018, pp. 1178–1180.

[16] E. Borgia, "The Internet of Things vision: Key features, applications and open issues," *Comput. Commun.*, vol. 54, pp. 1–31, Dec. 2014.

[17] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Futur. Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[18] Whitmore, A., Agarwal, A. & Da Xu, L. The Internet of Things—A survey of topics and trends. Inf Syst Front 17, 261–274 (2015).|

[19] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[20] A. Lele, "Internet of Things (IoT)," 2019, pp. 187–195.

[21] A. M. Abraham *et al.*, "Augmenting IoT-based Systems with Intelligence," in *2018 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 2018, pp. 1–6.

[22] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for Internet of Things: A Survey," *IEEE Internet Things J.*, vol. 3, no. 1, pp. 70–95, Feb. 2016.

[23] M. Ramljak, "Security analysis of Open Home Automation Bus system," in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2017, pp. 1245–1250.

[24] S. Werner, F. Pallas, and D. Bermbach, "Designing Suitable Access Control for Web-Connected Smart Home Platforms," 2018, pp. 240–251.

[25] A. Salihbegovic, T. Eterovic, E. Kaljic, and S. Ribic, "Design of a domain specific language and IDE for Internet of things applications," in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2015, pp. 996–1001.

[26] Y. Rizk, M. Awad, and E. W. Tunstel, "Decision Making in Multiagent Systems: A Survey," *IEEE Trans. Cogn. Dev. Syst.*, vol. 10, no. 3, pp. 514–529, Sep. 2018.

[27] D. Ye, M. Zhang, and A. V. Vasilakos, "A Survey of Self-Organization Mechanisms in Multiagent Systems," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 47, no. 3, pp. 441–461, Mar. 2017.

[28] A. Victor *et al.*, "Multi-agent cognitive system for optimal solution search," in *2018 International Conference on Development and Application Systems (DAS)*, 2018, pp. 53–56.

[29] A. Kantamneni, L. E. Brown, G. Parker, and W. W. Weaver, "Survey of multi-agent systems for microgrid control," *Eng. Appl. Artif. Intell.*, vol. 45, pp. 192–203, Oct. 2015.

[30] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-Agent Systems: A Survey," *IEEE Access*, vol. 6, pp. 28573–28593, 2018.

[31] S. Darmoul, S. Elkosantini, A. Louati, and L. Ben Said, "Multi-agent immune

networks to control interrupted flow at signalized intersections," *Transp. Res. Part C Emerg. Technol.*, vol. 82, pp. 290–313, Sep. 2017.

[32] H. E. Farag, E. F. El-Saadany, and L. El Chaar, "A multilayer control framework for distribution systems with high DG penetration," in *2011 International Conference on Innovations in Information Technology*, 2011, pp. 94–99.

[33] M. Robledo, R. Fuentetaja, A. G. Serrano, and J. Z. Hernández, "Propuesta de una arquitectura multiagente para la gestión de tráfico," *Traffic*, no. August 2014, 2002.

[34] J. M. Corchado, "2 Modelos y Arquitecturas de Agente," no. March, pp. 1–40, 1994.

[35] T. Balke and N. Gilbert, "How Do Agents Make Decisions? A Survey," *J. Artif. Soc. Soc. Simul.*, vol. 17, no. 4, 2014.

[36] I. F. Carlos, "Definición de una metodología para el desarrollo de sistemas multiagente," 1998.

[37] J. P. MÜLLER, "Architectures and applications of intelligent agents: A survey," *Knowl. Eng. Rev.*, vol. 13, no. 4, pp. 353–380, Feb. 1999.

[38] C. Biazus and M. Roisenberg, "The Development of a Hybrid, Distributed Architecture for Multiagent Systems," *2008 IEEE Conf. Intell. Syst.*, 2008.

[39] F. Bellifemine, F. Bergenti, G. Caire, and A. Poggi, "Jade — A Java Agent Development Framework," 2005, pp. 125–147.

[40] K. Kravari and N. Bassiliades, "A Survey of Agent Platforms," *J. Artif. Soc. Soc. Simul.*, vol. 18, no. 1, 2015.

[41] "Agent Mobility," in *Developing Multi-Agent Systems with JADE*, Chichester, UK: John Wiley & Sons, Ltd, pp. 115–130.

[42] "The JADE Platform," in *Developing Multi-Agent Systems with JADE*, Chichester, UK: John Wiley & Sons, Ltd, pp. 29–50.

[43] A. Ballén, N. Gelvez, and H. Espitia, "Prototype of a Recommendation System Based on Multi-agents in the Analysis of Movies Dataset," 2018, pp. 206–217.

[44] C. Peñaranda, V. Julian, J. Palanca, and V. Botti, "A Multi-Agent System to Improve Mobile Robot Localization," 2017, pp. 471–482.

[45] J. Palanca *et al.*, "A jabber-based multi-agent system platform Argumentation & Educational Recommender Systems & Persuasion View project A Jabber-based Multi-Agent System Platform *," 2006.

[46] M. Divya and K. L. Shunmuganathan, "A multi-agent based intelligent query processing system for Hadoop with FIPA-OS using cooperating agent in cloud environment," in *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, 2016, pp. 1–6.

[47] P. G. Balaji and D. Srinivasan, "An Introduction to Multi-Agent Systems," 2010, pp. 1–27.

[48] S. Poslad, "Specifying protocols for multi-agent systems interaction," *ACM Trans. Auton. Adapt. Syst.*, vol. 2, no. 4, 2007.

[49] A. Roggow, G. El-Howayek, and S. Khorbotly, “Autonomous identification of local agents in multi-agent robotic swarms,” in *2016 IEEE International Conference on Electro Information Technology (EIT)*, 2016, pp. 0170–0173.

[50] C. Savaglio, G. Fortino, M. Ganzha, M. Paprzycki, C. Bădică, and M. Ivanović, “Agent-Based Computing in the Internet of Things: A Survey,” 2018, pp. 307–320.

[51] T. Jung, N. Jazdi, and M. Weyrich, “A survey on dynamic simulation of automation systems and components in the Internet of Things,” in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017, pp. 1–4.

[52] C. Janiesch, M. Fischer, A. Winkelmann, and V. Nentwich, “Specifying autonomy in the Internet of Things: the autonomy model and notation,” *Inf. Syst. E-bus. Manag.*, vol. 17, no. 1, pp. 159–194, Mar. 2019.

[53] P. Semasinghe, S. Maghsudi, and E. Hossain, “Game Theoretic Mechanisms for Resource Management in Massive Wireless IoT Systems,” *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 121–127, Feb. 2017.

[54] D. Rivera, L. Cruz-Piris, G. Lopez-Civera, E. de la Hoz, and I. Marsa-Maestre, “Applying an Unified Access Control for IoT-based Intelligent Agent Systems,” in *2015 IEEE 8th International Conference on Service-Oriented Computing and Applications (SOCA)*, 2015, pp. 247–251.

[55] P. Pico-Valencia, J. A. Holgado-Terriza, and X. Quinonez-Ku, “A brief survey of the main internet-based approaches. An outlook from the internet of things perspective,” *Proc. - 3rd Int. Conf. Inf. Comput. Technol. ICICT 2020*, pp. 536–542, 2020.

[56] P. Pico-Valencia and J. A. Holgado-Terriza, “Agentification of the Internet of Things: A systematic literature review,” *Int. J. Distrib. Sens. Networks*, vol. 14, no. 10, 2018.

[57] B. Dolwithayakul, P. Boonnasa, S. Klomchit, and S. Tuwachaosuan, “Green public computer lab using single-board computer and interactive computer reservation system,” in *2015 International Computer Science and Engineering Conference (ICSEC)*, 2015, pp. 1–4.

[58] N. K. Nagwani, “Performance measurement analysis for multi-agent systems,” in *2009 International Conference on Intelligent Agent & Multi-Agent Systems*, 2009, pp. 1–4.

[59] L. Mengual, N. Barcia, J. Bobadilla, E. Jiménez, J. Setién, and J. Yágüez, “Arquitectura multi-agente segura basada en un sistema de implementación automática de protocolos de seguridad.”

[60] L. A. Espinel, “Estándares para la calidad de software,” *Tecnol. Investig. y Acad.*, vol. 5, no. 1, pp. 75–84, 2017.

[61] T. N. S. Vaca and A. E. O. Jácome, “Calidad de software del módulo de talento humano del sistema informático de la Universidad Técnica del Norte bajo la norma ISO/IEC 25000,” *Researchgate*, no. May, 2018.

[62] J. Luis, P. Luján, J. Luis, P. Yagüe, and E. Símó, “Sistema de Seguridad Variable

en una Arquitectura de Agentes Móviles Sistema de Seguridad Variable en una Arquitectura de Agentes Móviles," no. March 2015.

[63] D. Kovac, J. Hosek, P. Masek, and M. Stusek, "Keeping eyes on your home: Open-source network monitoring center for mobile devices," in *2015 38th International Conference on Telecommunications and Signal Processing, TSP 2015*, 2015, pp. 612–616.

[64] del Sol, Víctor *et al.*, "Sistema para la monitorización y asistencia en el hogar usando openhab y robótica móvil monitoring and home assistance using openhab and mobile robotics.". Universidad de Málaga, 2016.