UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ÁREA TÉCNICA

TITULACIÓN DE INGENIERO EN SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Diseño de un framework de sistema multi agente basado en una ontología

TRABAJO DE FIN DE TITULACIÓN

AUTOR: Torres Aguilar, Yadira Margarita

DIRECTOR: Arias Tapia, Susana Alexandra, Mgs

LOJA – ECUADOR

2014

APROBACIÓN DEL DIRECTOR DEL TRABAJO DE FIN DE TITULACIÓN

Magister.

Susana Alexandra Arias Tapia

DOCENTE DE LA TITULACIÓN

De mi consideración:

El presente trabajo de fin de titulación denominado: *Diseño de un framework de sistema multi agente basado en una ontología* realizado por *Yadira Margarita Torres Aguilar* ha sido orientado y revisado durante su ejecución, por lo que se aprueba la presentación del mismo.

Loja, julio de 2014

f)

Susana Arias Tapia

DECLARACIÓN DE AUTORIA Y CESIÓN DE DERECHOS

"Yo, *Yadira Margarita Torres Aguilar* declaro ser autora del presente trabajo de fin de titulación denominado: *Diseño de un framework de sistema multi agente basado en una ontología* de la Titulación de *Ingeniero en Sistemas Informáticos y Computación* siendo *Susana Alexandra Arias Tapia* directora del presente trabajo; y eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones legales. Además certifico que las ideas, conceptos, procedimientos y resultados vertidos en el presente trabajo investigativo, son de mi exclusiva responsabilidad.

Adicionalmente declaro conocer y aceptar la disposición del Art. 67 del Estatuto Orgánico de la Universidad Técnica Particular de Loja que en su parte pertinente textualmente dice: "Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado que se realicen a través, o con el apoyo financiero, académico o institucional (operativo) de la Universidad"

f)

Yadira Torres Aguilar

1105030207

DEDICATORIA

A mis padres Margarita y Jorge, por ser bastones fundamentales en mi vida, quienes gracias a su apoyo, consejos y cariño me enseñan a escalar y cumplir día a día mis objetivos.

A Vane y mi Rapha de manera especial, por caminar junto a mí y estar siempre incondicionalmente.

Con mucho cariño.

Yadira.

AGRADECIMIENTO

A Susana Arias, por la confianza, el apoyo, las enseñanzas y la paciencia dedicada en la orientación de este trabajo.

A Charito Puertas, por todas las oportunidades y consejos brindados.

A ustedes, eternamente agradecida.

Yadira.

INDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ABREVIATURAS

SMA	**S**istema **M**ulti **A**gente
MAS	**M**ulti **A**gent **S**ystem
MaSE	**M**ulti **a**gent **S**ystem **E**ngineering (Ingeniería de sistemas multi agente)
REST	**R**epresentational **S**tate **T**ransfer **(**Transferencia de estado representacional)
RPC	**R**emote **P**rocedure **C**all (LLamada a procedimiento remote)
URI	**U**niform **R**esource **I**dentifier (Identificador de recursos uniforme)
SOAP	**S**imple **O**bject **A**ccess **P**rotocol (Objeto simple de protocolos de acceso)

RESUMEN

Los sistemas multi agente (SMA) se caracterizan por ser una composición de varios agentes trabajando para cumplir objetivos individuales y de grupo. Este enfoque es beneficioso debido a que se presentan situaciones en las que se necesita obtener cooperación de dos o más recursos, con varios actores, para lograr objetivos que no pueden alcanzarse sólo de manera individual.

El propósito de este proyecto es diseñar un framework de sistema multi agente basado en una ontología que presente los lineamientos necesarios para modelar y construir agentes. Además de construir una aplicación que modele el framework propuesto.

Se evaluó el mejor modelo para el framework, lo cual llevó al reúso de la metodología MaSE para mostrar los pasos necesarios en el análisis y diseño de SMA.

Para poner en marcha los requerimientos encontrados en el desarrollo de la metodología, se ofrece lineamientos para la construcción de los agentes mediante servicios REST, en un ambiente Java.

La aplicación desarrollada permite recopilar información de múltiples fuentes de información, compartir datos de manera eficiente y gestionar e integrar los resultados obtenidos para presentarlos al usuario.

Palabras claves: Multi agentes, agentes, ontologías, metodología, framework, MaSE, REST, SMA, agentes de información, recursos web, internet.

ABSTRACT

The (SMA) multi agent systems are characterized by a composition of several agents working to meet individual and group goals. This approach is beneficial because situations in which you need to obtain cooperation of two or more resources, with various stakeholders to achieve goals that can not be achieved solely individually presented.

The purpose of this project is to design a framework of multi agent system based on an ontology to present the necessary guidelines to model and build agents. In addition to building an application that models the proposed framework.

The best model for the framework, which led to the reuse of the MaSE methodology to show the steps in the analysis and design of SMA was evaluated.

To implement the requirements found in the development of the methodology, guidelines for building agents using REST services are offered in a Java environment.

The developed application allows you to collect information from multiple sources of information, to share data efficiently and to integrate and manage the results for presentation to the user.

Keywords: multi agent, agent, ontology, framework, methodology, MaSE, REST, SMA, information agents, web resources, internet.

INTRODUCCIÓN

La sección de inteligencia artificial de la Universidad Técnica Particular de Loja se ve atenta a la necesidad de diseñar un framework de sistema multi agente basado en una ontología, que muestre los procesos necesarios a aplicar en un entorno de sistemas multi agente (SMA) de acuerdo a dominios específicos y montar aplicaciones reales que ejecuten los procesos planteados en el framework.

El propósito de esta investigación es obtener un framework de SMA que sirva de base para realizar aplicaciones basadas en agentes, ofreciendo lineamientos claves, de forma que, aseguren el correcto desarrollo de los mismos en base a objetivos y el dominio de aplicación necesario. Además, desarrollar una aplicación en base al framework propuesto.

La aplicación web a implementar consiste en el desarrollo de un sistema multi agente basado en una ontología, que permita recopilar información de manera automática desde tres recursos web (Facebook, LinkedIn y Google) y un recurso físico (TestEstres_M). Incluye funciones de monitoreo y notificación al usuario de la información que va siendo recopilada.

El presente trabajo se estructura en 4 capítulos.

El capítulo 1 muestra un estudio de ontologías, frameworks y metodologías usadas para SMA, resalta la importancia de usar ontologías para fines de recopilar, compartir y gestionar información y muestra un número de trabajos existentes en desarrollo de frameworks para SMA.

El capítulo 2 es el framewrok propuesto. La metodología elegida para la composición del framework es MaSE, que ayuda a identificar componentes globales, comunes, roles y conversaciones, con el fin de construir un modelo que permita desarrollar sistemas multi agente. En el desarrollo, el framework modela a los SMA en *servicios de agentes*, cada agente contiene un proceso que se comunica en tres niveles para permitir la interacción. El primer nivel expone los parámetros que debe recibir cada agente, el segundo nivel crea una instancia única para cada petición y el tercer nivel contiene los procesos necesarios para llegar a cumplir los objetivos. El framework esta implementado en el lenguaje Java, bajo la concepción de Rest Ful Web Services.

El capítulo 3 muestra el desarrollo de la aplicación basada en los procesos del framework. Los agentes creados son 5: Facebook, LinkedIn, Google, Encuesta y Mailer, se explica la funcionalidad y objetivos que debe cumplir cada uno. Asimismo, el modelado y los entregables de cada agente y finalmente la integración de los agentes en el SMA.

El capítulo 4 muestra el resultado de la validación y pruebas realizadas con la aplicación con actividades de verificación y validación. Verificación, para demostrar que la aplicación

satisface los requisitos encontrados. Validación, para demostrar que la aplicación satisface el uso para la cual fue creada.

OBJETIVOS

GENERAL:

Diseñar un framework de sistema multi agente basado en una ontología para obtener un modelo que permita realizar aplicaciones basadas en agentes.

ESPECIFICOS:

- Seleccionar la metodología adecuada para la elaboración del framework.
- Ofrecer lineamientos de desarrollo para la producción de los agentes.
- Desarrollar e implementar una aplicación web multi agente basada en una ontología que permita:
    - Recopilar información de múltiples fuentes de información
    - Compartir datos de manera eficiente
    - Gestionar e integrar los resultados obtenidos para presentarlos al usuario

CAPÍTULO 1

ESTADO DEL ARTE

1. Introducción

El uso de herramientas, tecnología y software en los procesos diarios empezó desde pequeñas aplicaciones de escritorio, y luego tomó lugar con la información en la web trabajando e interactuando con aplicaciones totalmente distribuidas y procesos heterogéneos que cada vez requieren menor dificultad y menor tiempo en procesamiento. Es así, como los procesos estáticos son remplazados por procesos dinámicos y estos procesos dinámicos a su vez se complementan con aplicaciones que integran técnicas de inteligencia artificial, sistemas basados en el conocimiento y el uso de recursos de la web.

El acceso a la información en la web es cada vez más fácil, por eso, se deben desarrollar métodos óptimos para la recopilación de los datos. Estos métodos vienen apoyados del uso de agentes. “Los agentes son la mejor alternativa en la búsqueda de la información que proviene desde la web”. (Gutierréz, 2008)

El uso de agentes se presenta a lo largo del tiempo en una larga variedad de aplicaciones, siendo cada vez más tendente el empleo de estas tecnologías para diversos fines, es así como se especula que “internet estará invadido de agentes virtuales que operarán las 24 horas del día, los 365 días del año, con diversos propósitos”. (Gutierréz, 2008)

Cuando un agente opera con otro para cumplir sus fines hablamos de un SMA, los agentes inteligentes y SMA como indica Alejandro Zunino van teniendo una cabida aceptable, por ser métodos óptimos para analizar y diseñar soluciones softwares, mejorando la forma en la cual se materializa el software. Además que se los puede implementar para cualquier dominio de aplicación. (2000, pp. 1-3)

Un SMA presenta varias características a tomar en cuenta como la comunicación, cooperación y negociación, y, otras características comunes al dominio de aplicación que hacen que su complejidad aumente a medida que aumentan las comunicaciones. Por esta razón se deben diseñar modelos que aseguren la correcta producción de los agentes.

Entre la gran variedad de métodos para diseñar framework de SMA, se ha centrado el interés en el uso de metodologías. “La introducción de agentes inteligentes requiere de metodologías que asistan en todas las fases del ciclo de vida del sistema de agentes”. (Iglesias, 1998)

Las metodologías ofrecen el camino necesario para identificar requerimientos. En el estudio realizado por Vicente Julián & Vicente Botti se hace una comparación de diversas metodologías para SMA, resaltando sus características en dos fases: análisis y diseño. El análisis que ayuda a obtener un mejor entendimiento de los requerimientos necesarios, y el diseño que parte de los artefactos encontrados en la fase de análisis para poner a producción los agentes. (2003, p. 2) “El principal objetivo de las metodologías es ofrecer una

visión de un sistema como una organización computacional consistente en diferentes entidades interactuando". (Julián & Botti, 2003)

Este capítulo presenta el análisis de la importancia del uso de agentes en entornos compuestos por ontologías, funciones de recopilación, frameworks y metodologías.

1.1 Agentes

Un agente es:

Para (Hípola & Vargas, 1999):

> Un agente es una entidad software que basándose en su propio conocimiento, realiza un conjunto de operaciones destinadas a satisfacer las necesidades de un usuario o de otro programa, bien por iniciativa propia o porque uno de estos se lo requiere. (p. 1)

Según (Lara & Martínez, 2014):

> Un agente es una entidad autónoma capaz de almacenar conocimiento sobre sí misma y sobre su entorno, con unos objetivos y capacidad. Asimismo, un agente inteligente es un programa que basándose en su propio conocimiento, realiza un conjunto de operaciones para satisfacer las necesidades de un usuario o de otro programa, bien por iniciativa propia o porque alguno de estos se lo requiere. (p. 12)

Otras definiciones:

- "Un agente es una entidad que puede llevar a cabo algunas tareas que son, usualmente, para ayudarnos a los seres humanos. Los agentes pueden ser biológicos, robóticos o computacionales". (Coppin, 2004)
- "Los agentes inteligentes son unidades de software con la capacidad de tomar decisiones racionales cuando se les presenta una elección". (Guzmán, Sánchez, & Torres, 2006)

<u>Un agente en la web es:</u>

Para (Villalba, 2007):

> Un agente personal en la web semántica recibe tareas y preferencias por parte de una persona, busca información de fuentes web, se comunica con otros agentes, compara información sobre requerimientos y preferencias del usuario, selecciona ciertas opciones, y da respuestas al usuario. (p. 3)

Según (Miranda, 2009):

> Los agentes inteligentes en sistemas basados en web semántica son los encargados de interactuar con otros agentes que a su vez usaran ontologías para poder interpretar

significado, extraer automáticamente datos de las páginas web y así sacar conclusiones de ellos, tomar decisiones y en muchos casos negociar. (p. 35)

Para (Julian, Rebollo, & Carrascosa):

> El agente necesita ser capaz de almacenar, aprender y manipular las preferencias y gustos de cada usuario, así como sus cambios. Un agente de información debe mantener un conjunto de habilidades básicas, que se pueden resumir en cuatro: (1) habilidades para ejecutar sus tareas, (2) habilidades de conocimiento, (3) habilidades de comunicación y (4) habilidades de colaboración. (p. 3)

Otra definición:

- "Un agente es una entidad software que recoge, filtra y procesa información contenida en la web, realiza inferencias sobre dicha información e interactúa con el entorno sin necesidad de supervisión o control constante por parte del usuario". (Lara & Martínez, 2014, p. 50)

1.2 Multi agentes

Un multi agente es:

Para (D WeiB, 2000):

> Los sistemas multi agente, son aquellos consistentes de un conjunto de agentes interactuando entre sí. Los agentes pueden ubicarse en el mismo equipo o en equipos remotos sobre una red. Esto exige nuevas habilidades, no presentes en agentes aislados, relacionadas con la interacción, tales como coordinación, cooperación y negociación. (p. 1)

"La integración de un conjunto de agentes de diversos tipos para constituir un sistema multi agente necesita de tres conceptos importantes: comunicación, cooperación y negociación". (Jiménez, Ovalle, & Ochoa, 2008)

La cooperación de múltiples agentes se define según Adel Al-Jumaily & Mohamed Al-Jaafreh como "trabajando juntos para lograr algo", donde los agentes realizan operaciones de comunicación y negociación. Negociación significa una forma clave de interacción que permite que grupos de agentes lleguen a un acuerdo mutuo con respecto al objetivo. (2006, p. 371)

Es decir, los sistemas multi agentes se comunican, cooperan y negocian entre ellos para cumplir los requerimientos para los cuales han sido creados. Como menciona Ana Serrano & Sascha Ossowski, un SMA debe estar sometido a cambios locales que deben diseñarse

mediante reglas de comportamiento y sus resultados están influenciados por el comportamiento del resto de agentes. (2006, p. 2)

“El paradigma de agentes y sistemas multi agente constituye actualmente un área de creciente interés dentro de la Inteligencia Artificial, entre otras razones, por ser aplicable a la resolución de problemas complejos no resueltos de manera satisfactoria mediante técnicas clásicas”. (Julián & Botti, 2003)

Gerhar d WeiB, resalta el interés en los SMA en la idea que muchos problemas del mundo real se modelan mejor utilizando un conjunto de agentes en lugar de un único agente. En particular, el modelado de múltiples agentes hace que sea posible hacer frente a las limitaciones naturales como la capacidad de procesamiento de un único agente o la distribución física de los datos, para beneficiarse de las propiedades de robustez, tolerancia a fallos, paralelismo y escalabilidad. (2000)

Los SMA “han tenido una cabida aceptable y exitosa en el ámbito de la recuperación de información”. (Jiménez D. , 2013)

1.3 Agentes web para la recopilación de información

El proceso manual de recopilación de información implica tiempo, agilidad y espera debido a la cantidad de información que existe en la web. Por esta razón, se desarrolla agentes trabajando para los usuarios de manera interactiva, autónoma y automática, encontrando recursos en la web útiles para satisfacer necesidades.

“Cuando el ámbito de trabajo es la web, en función del tipo de acciones que realizan, se pueden encontrar agentes de búsqueda, filtrado, recuperación de información, agentes de notificación, agentes móviles, etc”. (Julian, Rebollo, & Carrascosa)

Los trabajos relacionados en referencia a agentes y funciones de recopilación en la web, son los siguientes:

- ***InfoMagnet***

Como se indica en InfoMagnet, es un buscador en internet dedicado a ofrecer información relevante, realizando búsquedas de manera más profunda, más amplia y más rápida. En base a las búsquedas, realiza una recopilación de información de más de 5 millones de páginas en la web. (2014, p. 1)

- ***BargainFinder***

Un agente de recuperación e integración de información alojado en la www. Sus funciones van alrededor de los usuarios y los medios multimedia. Realiza comparaciones acerca de los precios de compras entre un sin número de tiendas de música en línea.

Bruce Krulwich destaca que la web se vuelve cada día más compleja, descubrió que la mayoría de los usuarios presentaban dificultad para encontrar tiendas en línea, en el caso de tenerla, se encontraban con información poco relevante. Un problema asociado a la falta de conocimiento de las tiendas en línea, se lo atribuye al costo de la publicidad, los registros en línea y al interés constante que los usuarios presentan por encontrar información en menor tiempo. Los agentes creados para BargainFinder solucionan este problema, intentando aprender los perfiles de usuario para evitar al máximo que construya cadenas de búsquedas y ofrecerle información de interés. (2000, p. 1)

- ***NewsFinder***

Un agente de recopilación de información referente a noticias. Sus funciones consisten en recuperar las noticias en línea desde diversos repositorios y compararlas con los perfiles de usuario.

Bruce Krulwich, considera a los agentes de integración claves en la gestión de la información, NewsFinder mantiene perfiles de los intereses del usuario y periódicamente envía información de artículos que responden a sus necesidades. Este enfoque es beneficioso debido a que, en la web, los artículos normalmente tienen protocolos de acceso complejos y se requiere que los usuarios se desplacen sobre una cantidad significativa de artículos en un gran número de medios de comunicación, cada uno con un método de acceso diferente para ver sus noticias. (2000, p. 5)

- ***Letizia***

(Lieberman) indica:

> El agente hace un seguimiento del comportamiento del usuario y los intentos de anticipar los elementos de interés, realizando una exploración y recopilación autónoma de enlaces desde la posición actual del usuario. El agente automatiza una estrategia de navegación que consiste en una búsqueda denominada best-first, infiriendo el interés de los usuarios según el comportamiento en la navegación.

- ***CiteSeer***

Kurt Bollacker, Steve Lawrence, & Lee Giles mencionan que CiteSeer es un agente que recorre la web en busca de publicaciones de interés del usuario. Cumple tres funciones, la primera; localizar y recopilar automáticamente publicaciones de investigación. La segunda; analizar los documentos y la tercera; una interfaz de base de datos que soporta búsquedas por palabras claves y por enlaces de citas. (1998, pp. 116-123)

- ***Market Maker***

“Es una herramienta para crear agentes basados en el comercio electrónico, soporta la identificación automática de productos, calificaciones de vendedor/comprador y negociaciones basadas en software de agentes para facilitar las transacciones. El agente realiza recopilaciones de intereses en base a los productos que un usuario compra en la web”. (Yi Yang, 1999)

- ***Google Alerts***

Como se señala en Google Alerts, se supervisa la web para encontrar contenidos interesantes en base a alertas que el usuario configura. Las fuentes donde se recopila la información son: noticias, blogs, web, video, libros y foros de debate. Sus funciones dentro de la recopilación son claras, un usuario especifica una alerta que desea supervisar y el agente se encarga de buscar por la www y notificar con periodicidad vía mail las novedades al usuario entregando contenido interesante. (2014, p. 1)

- ***Sistema multi agente para la recuperación de documentos digitales***

Presenta el proceso para la construcción de un sistema multi agente de recuperación de documentos digitales utilizando la metodología de modelado de sistemas multi agente MAS-CommonKADS, en un entorno Java.

En la investigación (Guzmán, Sánchez, & Torres, 2006) se resalta la importancia de los agentes en la recopilación de información:

> Una solución prometedora para el problema de localizar, recuperar y procesar grandes cantidades de información es el uso de agentes. (FIPA, 2005; Iglesias, 1998) (**…**) esto permite que las tareas de almacenamiento, clasificación, búsqueda, recuperación, filtrado e interpretación de información se optimicen y/o automaticen logrando con esto una comunicación más rápida y segura entre los usuarios y el propio sistema. (p. 2)

Jaime Guzmán, Alejandro Sánchez & Durley Torres establecen que la visión general de los agentes de recopilación de información es proporcionar información útil de un usuario según sus requerimientos, evitando la información no relevante. (2006, p. 2)

- ***Agentes inteligentes: Recuperación autónoma de información en la web***

Se indican las características y el análisis de las propiedades y habilidades deseables para agentes dedicados a la recuperación de información en la web.

(Berrocal, Figuerola, Zazo, & Rodríguez, 2003), mencionan:

> Una de las tareas obvias que podría ser abordaba mediante agentes es la exploración automática de la web, con el fin de recuperar páginas relevantes para unas necesidades informativas determinadas. De esta forma, la formulación de tales necesidades seria parte de las especificaciones iniciales proporcionadas al agente;

éste exploraría la red, eligiendo los enlaces más prometedores, accediendo a nuevas páginas, recopilando las que pudiesen satisfacer las especificaciones iniciales, y así sucesivamente. (p. 14)

- ***Plan de trabajo para la investigación y desarrollo de agentes***

Se ofrece una visión general de las actividades de investigación y desarrollo en el campo de agentes y SMA. Indica la clasificación y las relaciones que tienen las distintas clases de agentes, describiendo como parte de esta clasificación a los agentes de recopilación automática de información desde la web.

Se resume la necesidad de usar agentes que trabajen autónomamente con los recursos disponibles en la web debido a que, según Nicholas Jennings, Katia Sycara & Michael Wooldridge la información va creciendo todos los días, por eso es necesario que se gestione. Toda la información que se encuentra disponible en la web representa un problema debido al constante crecimiento. Hay dos visiones que pueden solucionar este problema, la primera; el filtrado de información ya que de toda la información disponible, solo una pequeña cantidad es relevante. La segunda; diseñar mecanismos para que la recopilación de información satisfaga los requerimientos. Los agentes de software y los SMA representan una nueva forma de analizar, diseñar e implementar los recursos. (pp. 7-9)

- ***Diseño de agentes para recuperar información para el enriquecimiento de ontologías dirigidas a epidemiología: el caso de la tuberculosis.***

Joanna Alvarado, Ari Barrera, Miguel González & María Junco explican que la web muestra un sin número de información pero no asegura que los resultados sean lo que esperemos y obliga al usuario a buscar y navegar manualmente consumiendo su tiempo. (2014, p. 33)

La solución planteada es el desarrollo de un SMA que busca información en la web acerca de la tuberculosis para enriquecer una ontología. Realiza el uso de la metodología GAIA, en el lenguaje de programación JADE usando un entorno Java.

El sistema se trata de 3 agentes que recopilan información desde recursos web como: Facebook, Twitter y Blogs. La información recopilada se almacena en una ontología, facilitando su acceso. Los agentes usan recursos como redes sociales, ya que al ser los medios de interacción más frecuentes permiten obtener información precisa.

“La búsqueda de información útil para el enriquecimiento de la ontología se realiza por medio de un sistema multi agente; dicho sistema se compone de 3 agentes: RSATFB (Recolector del sistema multi agente en Facebook), RSATTW (Recolector del Sistema multi agente en Twitter) y RSATBLOG (Recolector del Sistema multi agente en los BLOGs). Estos agentes recopilan la información que los usuarios proporcionan en estos sitios y la analizan

para determinar qué información podría ser relevante para la actualización de la ontología de Tuberculosis". (Alvarado, Barrera, González, & Junco, 2014, pp. 33-34)

- ***Agentes móviles para la recuperación de información en bibliotecas digitales***

La investigación realizada por Roberto Mendoza hace mención del crecimiento exponencial de información en la web, y de búsquedas poco relevantes, ya que internet muestra información de manera desorganizada y no estructurada. (2009, p. 4) Por esta razón se usan sistemas que ayuden a los usuarios en las tareas más simples.

Se enfoca en solucionar el desaprovechamiento de los recursos digitales en la web, especialmente los libros, diseñando una solución a partir de agentes móviles con funciones de recuperación automática de recursos en la web.

El propósito es "desarrollar un prototipo software con agentes móviles para la búsqueda y recuperación automática de información en un entorno de biblioteca digital que muestre los documentos de acuerdo a la semántica de la consulta especificada por el usuario". (Mendoza, 2009)

1.4 Agentes y ontologías en la gestión de la información

"La incorporación del enfoque semántico en la web asegura que todos los resultados de búsqueda sean exactos (W3C)" (Alvarado, Barrera, González, & Junco, 2014)

Cuando la representación de conocimiento se realiza mediante ontologías, la información tiene estructura y representación. Como explica Rodrigo Villalba la evolución de la web da un paso rápido hacia la web semántica, con esta inclusión, se puede compartir e integrar información y servicios. (2007, p. 2)

"Las ontologías son utilizadas por las personas, las bases de datos, y las aplicaciones que necesitan compartir un dominio de información" (Fernández, Carbonell, Pérez, & Villalón, 2009). Incluyen definiciones de conceptos básicos del dominio, y las relaciones entre ellos, útiles para los ordenadores ya que codifican el conocimiento de un dominio haciéndolo reutilizable. En la investigación de Raquel Arenas & Marta Prieto acerca de la web semántica, se trata a las ontologías como tecnologías imprescindibles para conseguir la web semántica. Su estructura de información organizada hace que los agentes puedan entenderla e interpretarla. (2004, pp. 1-2) Además, en relación a agentes y ontologías se expresa que "gracias al conocimiento almacenado en las ontologías, las aplicaciones podrán extraer automáticamente datos de las páginas web, procesarlos y sacar conclusiones de ellos, así como tomar decisiones y negociar con otros agentes o personas". (Arenas & Prieto, 2004)

La necesidad de comunicación evoluciona hacia la necesidad de razonamiento lógico, de tener redes de datos entrelazados que representen dominios de conocimiento y, puedan ser entendibles, indiferentes del lenguaje de representación y la tecnología de gestión.

(Berners-Lee, Hendler, & Lasilla, 2001) definen acerca de la web semántica:

> Propone superar las limitaciones de la web actual mediante la introducción de descripciones explícitas del significado, la estructura interna y la estructura global de los contenidos y servicios disponibles en la www. Frente a la semántica implícita, el crecimiento caótico de recursos, y la ausencia de una organización clara de la web actual.

En el análisis de Pablo Castells de la definición de Tim Berners-Lee, James Handler & Ora Lassila, deducen que el propósito de usar ontologías es tener clases y relaciones que sean definidas exclusivamente por una ontología y de esta forma se haga uso de las características expuestas y se creen estándares en diversos dominios, para que finalmente los resultados se integren, exista colaboración y se tenga una red de nodos, descentralizados, compatibles y accesibles. (2012, pp. 5-6)

El punto a favor acerca de la "web semántica es que todos los recursos en la web pueden consumirse mediante aplicaciones". (Castells, 2012) (Berners-Lee, Hendler, & Lasilla, 2001) La **figura 1** muestra una representación elaborada por (Abián, 2013) en donde se observa el ambiente que ocupan las ontologías en relación a recursos, aplicaciones, usuarios y tecnologías.

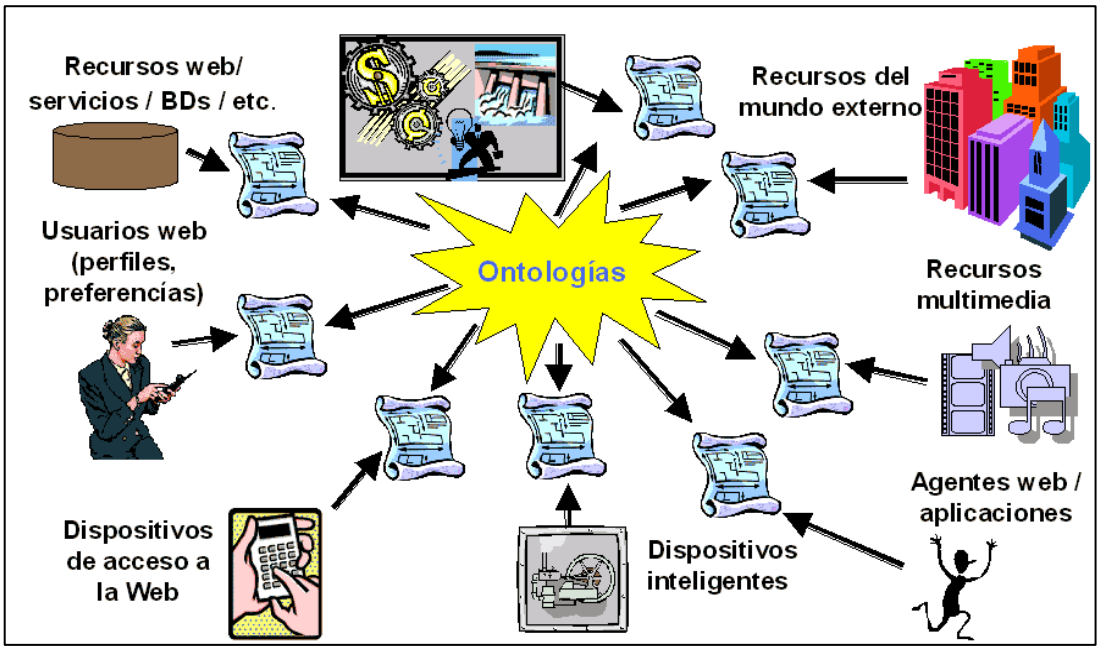


Figura 1: Desenvolvimiento de ontologías por aplicaciones y recursos; agentes

Fuente: (Abián, 2013)

Los trabajos relacionados en cuanto al uso de ontologías en la gestión de recopilación de información son los siguientes:

- ***La evolución de la web semántica***

Para Pablo Castells, la web semántica es un área potente en la inteligencia artificial. Su evolución e integración de nuevas tecnologías hacen que se estandarice mediante el uso de contenido enriquecido y la inclusión de agentes para el consumo de información. Por ser internet un instrumento de uso cotidiano se tiene la concepción de la web como una plataforma universal para el despliegue de aplicaciones. (pp. 1-2)

Actualmente, se requiere que los datos en la web representen fuentes robustas de información. Consumir y analizar datos de recursos accesibles y al alcance para representar patrones de interés del comportamiento de los usuarios para la toma de decisiones. La gestión de recopilación está realizada por aplicaciones software o agentes a través de los mecanismos que expone la web semántica.

Para (Castells, 2012):

> La web semántica propone introducir descripciones explícitas sobre el significado de los recursos, para permitir que las propias máquinas tengan un nivel de comprensión de la web suficiente como para hacerse cargo de una parte, la más costosa, rutinaria, o físicamente inabarcable, del trabajo que actualmente realizan manualmente los usuarios que navegan e interactúan con la web. (p. 1)

- ***La web semántica: contenido colaborativo***

Rodrigo Villalba describe tres etapas importantes en la evolución de la web: 1) generación de contenido estático, 2) generación de contenido dinámico e interactivo y 3) generación de contenido colaborativo. (p. 3) La última hace referencia a la web semántica, donde el usuario participa directamente en la generación de información.

El desborde de información existente en la web, genera a su vez contenido de importancia y del cual se puede realizar análisis en diversos ámbitos. En sí, la gestión de la información en la web semántica implica desarrollar agentes que naveguen por internet consumiendo recursos útiles para los usuarios.

La investigación de (Villalba, 2007), considera la importancia de los agentes, ya que:

> Se necesita de agentes o representantes software capaces de navegar y realizar operaciones por nosotros para ahorrarnos trabajo y optimizar los resultados. Para conseguir esta meta, la web semántica propone describir los recursos de la web con representaciones procesables (es decir, entendibles) no sólo por personas, sino por

programas que puedan asistir, representar, o reemplazar a las personas en tareas rutinarias o inabarcables para un humano. (p. 1)

- ***Ontologías, metadatos y agentes: recuperación "semántica" de la información***

Eduardo Peis, Yusef Hassam, Eduardo Herrera, & Enrique Herrara explican que el crecimiento exponencial de informacion y la falta de una estructura jerarquizada para la misma, presentan problemas en la recopilacion de informacion desde la web, sumado a que los actuales buscadores son incapaces de ofrecer informacion realmente precisa en sus resultados. Esta impresición se debe a la falta un significado semántico. (p. 1)

Para dar un entendimiento de la web no solo a personas, sino también a máquinas, aplicaciones y sistemas es imprescindible el uso de la web semántica.

La web semántica supone una de las mayores ventajas en las áreas como la recopilación de información. Los agentes serían los encargados de buscar información relevante a través de consultas dadas, explorando e interactuando sobre internet.

Asimismo, (Peis, Hassam, Herrera, & Herrera, 2003) indican:

> Un agente inteligente recorrerá la web a través de los enlaces entre recursos en busca de aquella información que le sea solicitada, pudiendo además interactuar con el entorno para el cumplimiento de tareas encomendadas. Por ejemplo, un agente, ante una consulta dada, podría consultar autónomamente un buscador, y a partir de sus resultados, explorar la web hasta encontrar la información solicitada, pudiendo finalmente llevar a cabo una acción sobre dicho recurso. (p. 5)

1.5 Conceptualización de ontologías mediante elementos de la web semántica

La web semántica guarda una estructura clara para la elaboración de sus vocabularios, dentro de esta estructura se encuentran:

<u>Los metalenguajes y estándares de representación:</u>

- **XML** (Extensible Markup Lenguaje): Su propósito es expresar información de forma estructurada, fácil de distribuir, que pueda ser, reutilizada, procesada y entendida por diversas fuentes. Creada por la W3C como una arquitectura abierta cliente-servidor. Define la sintaxis utilizada para definir otros lenguajes de etiquetas estructurados. Se orienta a organizar y esquematizar.

  Entre sus principales características están:

  - **Arquitectura abierta y extensible**: no necesita versiones, funciona en los navegadores sin problema y está adaptada de tal forma, que pueda funcionar en futuros proyectos.

- **Consistente, homogéneo y amplio**: se puede adaptar a otras tecnologías como los RDF.
- **Manipulación de los datos**: Solo es necesario crear un cliente Web para receptar los datos en formato XML.
- **Respuestas precisas y acertadas**: Las búsquedas XML devuelven un lenguaje de marcado capaz de ser comprensible sin necesidad de conocer conceptos básicos.

```
<?xml version="1.0" encoding="UTF-8" ?>
<album>
    <autor>Alejandro Sanz</autor>
    <titulo>La música no se toca</titulo>
    <formato>MP3</formato>
   <localizacion>España </localizacion>
</album>
```

Figura 2: Ejemplo de representación de XML

Fuente: La autora.

- **XML SCHEMA** (Lenguaje de esquema XML): Creada también por la W3C, es una extensión del formato XML, se usa para describir la estructura XML, mucho más preciso y abarca restricciones. Contiene atributos y elementos que se identifican a un etiquetado y que pueden relacionarse entre sí.

```
<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="note">
<xs:complexType>
<xs:sequence>
<xs:element name="to" type="xs:string"/>
<xs:element name="from" type="xs:string"/>
</xs:sequence>
</xs:complexType>
```

Figura 3: Ejemplo de representación de XML SCHEMA

Fuente: La autora.

- **RDF** (Resource Description Framework): Creado por la W3C, es una familia de especificaciones aplicable a metadatos, capaz de expresar cualquier dominio de conocimiento, trabajando con información distribuida.

  Representa información sobre los recursos en la web empleando URIs para que puedan ser identificados, usados y consumidos por aplicaciones.

Las URIs identifican el nombre de un recurso, pueden ser clasificados como un identificar URL o un nombre URN, o ambos. El URL representa al localizador de un recurso y la URN al nombre de dicho recurso.

Como ejemplo: http://ejemplo.org/Yadira. La parte en amarillo representa al localizador y la parte en morado al nombre del recurso.

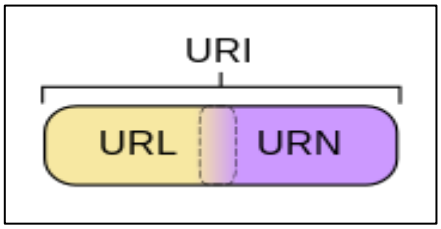


Figura 4: Diagrama de Euler para identificar una URI

Fuente: La autora.

Reglas del RDF (Zambrano, 2009):

- **Un recurso** es cualquier cosa que puede tener un URI, esto incluye todas las páginas web, todos los elementos individuales de cada documento XML y mucho más.
- **Una propiedad** es un recurso que tienen un nombre y que puede usarse como una propiedad, por ejemplo autor o título.
- **Una sentencia** consiste en la combinación de un recurso, una propiedad y un valor. Estas partes son conocidas como el sujeto, predicado y el objeto de la sentencia.

Todos los recursos tienen propiedades y valores que se asocian mediante tripletas. Pero existe otra forma de notación que es mostrar una sentencia mediante grafos dirigidos, con arcos y nodos. Los sujetos y objetos son nodos, mientras que los predicados son arcos.

Las tripletas están formadas por tres partes:

1. Un sujeto
2. Un predicado
3. Un objeto

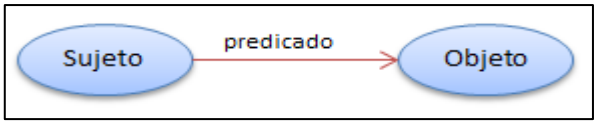


Figura 5: Esquema de una sentencia RDF

Fuente: La autora.

El siguiente ejemplo hace más visible el esquema RDF:

Number	Subject	Predicate	Object
1	http://www.w3.org/	http://purl.org/dc/elements/1.1/title	"World Wide Web Consortium"

Figura 6: Descomposición RDF en tripletas

Fuente: La autora.

Ahora su representación en lenguaje semántico:

```
1: <?xml version="1.0"?>
2: <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3:   xmlns:dc="http://purl.org/dc/elements/1.1/">
4:   <rdf:Description rdf:about="http://www.w3.org/">
5:     <dc:title>World Wide Web Consortium</dc:title>
6:   </rdf:Description>
7: </rdf:RDF>
```

Figura 7: Lenguaje RDF para la representación de metadatos

Fuente: La autora.

Y visualizamos el grafo generado del lenguaje:

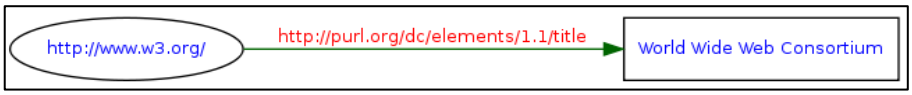


Figura 8: Grafo de esquema RDF para describir el título de una URL

Fuente: La autora.

- **RDF SCHEME** (Esquema RDF): es una extensión del lenguaje RDF y proporciona elementos para describir su vocabulario.

  Aunque es menos potente que OWL, se debe entender que un esquema RDF's está compuesto por elementos RDF.

  RDF's proporciona las directrices para distribuir el lenguaje RDF, que a su vez tiene una estructura basada en el lenguaje XML.

  El esquema es el siguiente:

  1. **Classes**
     a. rdfs:Resource
     b. rdfs:Class
     c. rdfs:Literal
     d. rdfs:Datatype
     e. rdf:XMLLiteral

f. rdf:Property

2. **Properties**

   a. rdfs:range

   b. rdfs:domain

   c. rdf:type

   d. rdfs:subClassOf

   e. rdfs:subPropertyOf

   f. rdfs:label

   g. rdfs:comment

3. **Otro vocabulario**

   a. rdf:Statement

   b. rdf:subject

   c. rdf:predicate

   d. rdf:object

```
<rdfs:Class rdf:about="http://www.w3.org/2000/01/rdf-schema#Resource">
        <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
        <rdfs:label>Resource</rdfs:label>
        <rdfs:comment>The class resource, everything.</rdfs:comment>
</rdfs:Class>
```

Figura 9: Ejemplo de RDF SCHEMA

Fuente: La autora.

- **OWL** (Web Ontology Language): Construida sobre lenguajes RDF y OWL. Arquitectura semántica compatible con ontologías para publicar y compartir datos en la Web.

```
<owlx:Class owlx:name="Sportsman" owlx:complete="false">
<owlx:Class owlx:name="#Person" />
</owlx:Class>
```

Figura 10: Ejemplo de OWL

Fuente: La autora.

Y el lenguaje que permite el consumo de datos semánticos:

- **SPARQL** (SPARQL Protocol and RDF Query Language): “Permite la consulta de datos semánticos, que usan arquitecturas RDF, OWL. El 15 de Enero del 2008 la W3C lo recomendó oficialmente. Estandarizado por Data Access Working Group

(DAWG) de W3C. Considerado componente de la Web Semántica". (Qaissi, 2009)

Las consultas SPARQL se realizan sobre esquemas ontológicos, aunque son muy parecidos al SQL QUERY, SPARQL no realiza consultas a las tablas comunes que encontramos en bases de datos relacionales, sino a recursos RDF, mediante tripletas.

El estándar establecido por la W3C es el siguiente:

```
SELECT <<nombredelRecurso>>
WHERE
{
 ?Sujeto  ?Predicado ?Objeto .
}
```

Figura 11: Estándar sentencias SPARQL

Fuente: La autora.

El siguiente ejemplo muestra cómo realizar la consulta a un recurso dado el esquema.

```
PREFIX  dc: <http://purl.org/dc/elements/1.1/>
SELECT  ?title
WHERE   {<http://ejemplo.org/libros> dc:title ?title }
```

Figura 12: Ejemplo de Consulta SPARQL

Fuente: La autora.

- Las consultas SPARQL cubren tres objetivos (Qaissi, 2009):
    1. Extraer información en forma de URIs y literales
    2. Extraer subestructuras RDF
    3. Construir nuevas estructuras RDF partiendo de resultados de consultas.

- ***Especificaciones SPARQL.***

Son 4 los tipos de especificaciones de un lenguaje SPARQL (Qaissi, 2009):

1. **SPARQL Query Language**: Núcleo de SPARQL o componente principal. Describe la sintaxis de sus sentencias y su concordancia.
2. **SPARQL Protocol for RDF:** Formato utilizado para devolver los resultados de las consultas (SELECT o ASK), a partir de un esquema RDF-XML.
3. **SPARUL (SPARQL Update):** Hace actualizaciones del contenido RDF. Desarrollado por Hewlett-Packard. No reconocido como estándar.

4. **SPARQL Query Results XML Format**: Describe el acceso remoto a datos y la transmisión de consultas del cliente a los procesadores. Utiliza WSDL (Web Services Description Language) para implementar protocolos remotos en la consulta a bases de datos basada en RDF.

1.5.1 *Etiquetados de ontologías.*

Las ontologías pueden definirse bajo etiquetas RDF u OWL. “Para desarrollar aplicaciones basadas en RDF, OWL o lenguajes similares se precisan librerías para leer y procesar las ontologías definidas en estos lenguajes”. (Castells, 2012)

Una de las librerías más populares para la creación de definiciones semánticas es Jena. Como lo afirma (Castells, 2012) y (Drake & López, 2009) Jena permite leer, recorrer y modificar RDF como OWL desde un programa Java.

“Jena permite además guardar las ontologías tanto en RDF textual como en formato de base de datos, lo que es importante para grafos muy grandes”. (Bravo & Redondo, 2004)

Las ontologías funcionan bajo etiquetas semánticas pero se apoyan de formatos que ofrecen estructuras para dotarlas de jerarquía y organización.

- ***RDF vs XML.***

Muestran datos estructurados, sin embargo, muchas aplicaciones trabajan sobre el esquema XML. Se debe saber cuándo utilizar el uno o el otro. La **tabla 1** presenta la comparación de XML y RDF.

Tabla 1: Comparación entre XML y RDF

XML	RDF
Lee datos.	Intenta hacer algo inteligente con estos datos.
Es un formato de serialización, de tal manera que codifica su información para ser comprensible entre máquinas.	Es un modelo de datos, que tiene un conjunto establecido de normas para representar información.
Es una opción, un mecanismo de lectura y estructura de datos.	Representa el contenido de la información.
Tiene una estructura y significado sintáctico.	Tiene un significado semántico.

Fuente: La autora.

Como ratifica Héctor Flórez, RDF es un lenguaje utilizado en la Web Semántica. (p. 12)

“RDF es usado por agentes inteligentes para facilitar el intercambio de información y compartir conocimientos”. (Arenas & Prieto, 2004) Describe un modelado de recursos en la web para representar cualquier dominio de conocimiento, archivos, imágenes, texto, destinado a informar. Debido a que “RDF no se asocia a ningún dominio en particular, lo podemos emplear en cualquier campo” (Arenas & Prieto, 2004) y XML un lenguaje para serialización y jerarquía de datos, por qué define esquemas de etiqueta.

- ***RDF vs OWL.***

Trabajan con esquemas semánticos y representan dominios de conocimiento. La **tabla 2** muestra las comparaciones.

Tabla 2: Comparación entre RDF y OWL

RDF	OWL
Representa dominios de conocimiento mediante la estandarización de un formato de datos en forma de tripletas.	Es similar, pero va más allá, permite decir muchas cosas más acerca del modelo de datos, como restricciones, comparaciones, limitaciones, teniendo más información.
Se puede representar cualquier cosa, sin restricciones por ejemplo; un recurso puede también ser una instancia.	Indica cómo se puede y cómo no se puede utilizar un vocabulario.
Algunos tipos de inferencia se ejecutan muy rápidamente.	Como tiene un vocabulario más detallado necesita de mayor cantidad de procesamiento.

Fuente: La autora.

OWL define vocabulario más detallado de dominios de conocimiento, no solo ofreciendo anotaciones, sino también, restricciones, limitaciones, comparaciones y más. Aunque se puede representar cualquier cosa como dominio de conocimiento, sin embargo OWL nos muestra la forma de cómo hacerlo y como no.

“OWL está diseñado para ser usado en aplicaciones que necesitan procesar el contenido de la información en lugar de únicamente representar información para los humanos. OWL

facilita un mejor mecanismo de interpretabilidad de contenido web, que los mecanismos admitidos por XML, RDF, y RDF Schema". (Arenas & Prieto, 2004)

En la web semántica se puede usar ambos formatos para describir ontologías.

1.6 Recursos web para recopilación de información

El consumo de información de redes sociales ayuda a obtener información de los usuarios, que puede ser fundamental para encontrar patrones que definan información en diversos ámbitos, por ejemplo, comportamientos que representen las causas o consecuencias de las acciones que efectúan los usuarios. Estos datos recopilados de los recursos web sociales, sin embargo, a primera vista no definen la certeza o veracidad de las acciones, comportamientos o afirmaciones para los cuales son recopilados. Para esto, precisan ser tratados necesariamente por expertos en el área de interés, mediante el diseño de mecanismos, técnicas y algoritmos que surjan de su análisis.

Las redes sociales Facebook, LinkedIn, Google tienen métodos que permiten el consumo de información mediante el uso de Apis.

Los trabajos relacionados que resaltan la importancia de los recursos web para la recopilación de información son los siguientes:

- ***Historia del concepto de red social***

La investigación realizada por Gina Cardozo, demuestra la evolución de las redes sociales y la importancia que tienen actualmente. Considera a las redes sociales como redes de conocimiento, de intercambio de información. El fenómeno de las redes sociales se ha visto magnificado gracias a las nuevas tecnologías, el acceso a internet cada vez más fácil, la necesidad de interacción de los usuarios, de forma que estas características han ido elevando potencialmente su uso y generando información a niveles más profundos de una forma nunca antes imaginada. (2009, p. 5)

"Las redes sociales definen una estructura que organiza la manera en que se relacionan las personas y la cualidad en que se mueve, crea y transforma la información entre ellas". (Cardozo, 2009)

- ***Diseño de agentes para recuperar información para el enriquecimiento de ontologías dirigidas a epidemiología: el caso de la tuberculosis.***

Joanna Alvarado, Ari Barrera, Miguel González & María Junco utilizan recursos web sociales como fuentes de recopilación de información, estos recursos son Facebook, Twitter y Blogs. (2014, p. 35) "Los agentes recopilan la información que los usuarios proporcionan en estos sitios, la analizan para determinar qué información es irrelevante o útil para la actualización

de la ontología de Tuberculosis a partir de los términos que sean extraídos de estos sitios web". (Alvarado, Barrera, González, & Junco, 2014)

El objetivo de la investigación de Joanna Alvarado, Ari Barrera, Miguel González & María Junco es obtener agentes que monitoreen los recursos en internet, verificando su disponibilidad; así como determinar nuevas páginas que se podrían incorporar para obtener información de acuerdo con la ontología especificada. (2014, p. 44)

1.7 Frameworks de SMA{ XE "**Framework de SMA**" }

"Un framework puede ser definido a través de un conjunto de clases genéricas que establecen la estructura de comportamiento abstracta de aplicaciones pertenecientes a un determinado dominio". (Johnson & Foote, 1998) La estructura de comportamiento abstracta de un framework tiene la característica de poder ser utilizada como base para implementar aplicaciones del dominio del framework.

Los trabajos realizados en referencia a frameworks de SMA, se centran en diversas características de acuerdo a los patrones requeridos en su modelo de dominio, por ejemplo: Bubble; simulación de agentes. Aglest; movilidad. Jade; middle-ware. Jafmas; coordinación.

- ***Bubble***

Un framework para el desarrollo de SMA. "Calificada como un proto framework que codifica en Java los componentes y comportamientos comunes a entidades colaborativas, posibilitando a través de ellos la construcción de otros frameworks más específicos de agentes". (Amandi, 2001)

(Díaz, Trilnik, Campo, & Clausse, 2001) definen:

> Bubble, para construir modelos de simulación de procesos colectivos utilizando un enfoque multi agente. El objetivo del framework es permitir definir y organizar conjuntos de agentes reactivos, que interaccionen entre ellos a través de eventos, dando lugar a determinados comportamientos colectivos que tienen aplicación en la simulación de procesos. Se ha desarrollado como ejemplo base, una simulación de flujo de fluidos en 2 fases, donde se analizan las perspectivas del enfoque. (pp. 2-3)

Trabaja en estados internos y en su arquitectura contempla la comunicación, la organización y el comportamiento, no tiene características distribuidas y al ser un sistema reactivo presenta complejos patrones de comportamiento.

La comunicación se realiza mediante eventos efectuando una seria de estados de activación y desactivación.

- ***Aglets***

Un framework basado en agentes móviles. Trabaja bajo el estándar de estados pero a diferencia de Bubble define 4 estados para la comunicación: *(a) Creación o Clonación; (b) Liberación de recursos; (c) Movilidad; y (d) Persistencia.* Cuenta también con un monitor de red y un administrador de seguridad para lograr la independencia.

(Lange & Mitsuru, 1998) mencionan:

> La estructura definida en los Aglets permite que estos actúen autónomamente colaborando con otros Aglets. La colaboración es una necesidad en los agentes porque en general ninguno está aislado ni puede realizar tareas complejas solo. La movilidad y el pasaje de mensajes son características importantes para la colaboración. Son objetos móviles Java capaces de visitar diferentes sitios en una red. Los Aglets son autónomos, ya que cada uno contiene un thread activo de ejecución y es capaz de reaccionar a mensajes enviados a él.

La comunicación en Aglets se realiza mediante estados. Los estados necesitan de procesos y roles para ejecutarse, pero siempre consultando la jerarquía en la cual se encuentran, esta característica hace que la comunicación sea compleja.

- ***Jade***

Es un entorno que "simplifica la implementación de sistemas multi agente mediante una capa de soporte (middle-ware) que respeta las especiaciones FIPA y con un conjunto de herramientas para el desarrollo y debugging" (Castelfranchi & Lespérance, 2000). "La plataforma puede ser distribuida en varias máquinas y la configuración puede ser controlada mediante una interfaz gráfica remota. La configuración puede incluso ser cambiada en tiempo de ejecución moviendo agentes de una máquina a otra, cuando es necesario". (JADE, 2002)

A diferencia de los framework anteriores no usa estados en la comunicación. Incorpora el paso de mensajes enviando una cola de mensajes ACL, en una lista en la cual los agentes acceden mediante una combinación de varios modos: blocking (bloqueo), polling (división), timeout (tiempo de espera) y pattern matching (coincidencia de patrones).

- ***Jafmas***

Como se menciona en (JAFMAS, 2002)

> Jafmas, provee una metodología genérica para desarrollar sistemas multi agente basados en los actos del habla junto con un conjunto de clases para soportar la implementación de estos agentes en Java. La intención del framework es asistir a los

desarrolladores principiantes y expertos a estructurar sus ideas en aplicaciones de agentes concretas.

La metodología está basada en cinco etapas:

1. Identificación de agentes
2. Definición de las conversaciones de cada agente
3. Determinación de las reglas que gobiernan las conversaciones de cada agente
4. Analizar la coherencia entre todas las conversaciones en el sistema
5. Implementación

Jafmas expone ideas esenciales de orientar las construcciones hacia dominios específicos basándose en metodologías y modelando las metodologías en aplicaciones concretas. Las cinco etapas de Jafmas son genéricas, se pueden modelar a cualquier dominio. No expone arquitecturas específicas, adecuando esta característica de acuerdo a las necesidades.

El enfoque adoptado en Jafmas es proveer de una metodología genérica para el desarrollo de sistemas multi agente.

La **tabla 3** muestra un resumen del estudio de frameworks de SMA.

Tabla 3: Cuadro resumen de framework de SMA

Framework	Enfoque	Comunicación
Bubble	simulación de agentes	Reactivo, comunicación mediante eventos en estados (activación- des activación)
Aglest	movilidad	4 estados para la comunicación: (a) Creación o Clonación; (b) Liberación de recursos; (c) Movilidad; y (d) Persistencia.
Jade	middle-ware	Comunicación mediante el paso de mensajes enviando una cola de mensajes (ACL), de varios modos: blocking, polling, timeout y pattern matching.

Jafmas	coordinación	Comunicación mediante conversaciones y siguiendo los pasos establecidos de una metodología genérica para producir los agentes.

Fuente: La autora.

1.8 Metodologías de SMA

Una metodología define un conjunto de métodos y procesos que se siguen para asegurar el desarrollo de un sistema software. Las metodologías orientadas a agentes aseguran que el diseño, análisis y construcción del SMA se dé de la forma más adecuada.

Las metodologías permiten:

- Analizar el problema que se va a resolver
- Concretar un proceso para el desarrollo
- Evaluar el proceso de desarrollo
- Poner en marcha

La característica de un SMA, de acuerdo a Jorge Gómez, viene integrada por tres áreas de conocimiento; 1) ingeniería del software, que permite realizar una estructura del proceso de desarrollo, 2) inteligencia artificial, que dota a las programas de una capacidad para tratar situaciones imprevistas y ayuda en la toma de decisiones y 3) programación concurrente y distribuida para coordinar las tareas ejecutadas de todos los agentes. Debido a esta combinación de tecnologías, el desarrollo de SMA, se complica. (2003, p. 3)

Por esta razón, es necesario usar herramientas que aseguren la creación de los agentes, el uso de metodologías en agentes vendría a solucionar estos inconvenientes.

Para (Férnandez, 2004):

> Una metodología de ingeniería software es un proceso para la producción organizada del software, empleando una colección de técnicas predefinidas y convenciones en las notaciones. Una metodología se presenta normalmente como una serie de pasos, con técnicas y notaciones asociadas a cada paso. Los pasos de la producción del software se organizan normalmente en un ciclo de vida consistente en varias fases de desarrollo.

Dentro de esta área de investigación varias son las metodologías existentes orientadas a los SMA, cada una con características propias que pueden ser reutilizadas de acuerdo a los

objetivos de cada agente. Además, facilitan su aplicación ofreciendo métodos, herramientas y artefactos para el desarrollo

Las características de las metodologías de los SMA se enfocan a dos puntos en particular con fines muy específicos:

1. Identificar los diferentes subsistemas que forman parte del sistema global.
2. Identificar las posibles interacciones y dependencias entre ellos.

Se han listado seis posibles retos que puede presentar un SMA y por qué es necesario usar metodologías (Sycara, 1998):

1. Como descomponer los problemas y asignar tareas a los agentes individuales.
2. Como comunicar el control y la comunicación del agente.
3. Como hacer que varios agentes actúen de una manera coherente.
4. Como hacer que los agentes individuales razonen acerca de otros agentes en su estado de coordinación.
5. Como conciliar objetivos que producen conflictos en el estado de coordinación del agente.
6. Como ingeniar sistemas multi agentes prácticos.

Las razones por las cuales se dice que un sistema multi agente abarca un alto grado de complejidad son las siguientes:

1. “Hay que dividir la tarea en componentes autónomos (agentes) que aplicarán distintas estrategias para resolver los sub problemas”. (Gallego, Llorens, & Rizo, 2004)
    - Para este fin se necesita definir cuáles son los agentes que participan en la recopilación de la información.
2. “Hay que resolver el problema de la comunicación entre los distintos agentes para que puedan coordinarse y entenderse”. (Gallego, Llorens, & Rizo, 2004)
    - Una vez definido cuáles son los agentes que participan en el SMA, se debe definir y diseñar de qué forma se comunican y cómo colaboran unos entre otros.
3. “Se trata de un entorno distribuido de proceso paralelo, lo cual dificulta muchísimo la comprensión del funcionamiento del sistema”. (Gallego, Llorens, & Rizo, 2004)
    - Cada agente tiene una forma distinta de resolver sus problemas, esto implica que cada uno posee una arquitectura distinta, y se debe crear una

arquitectura que soporte la comunicación de todos los agentes funcionando como un todo para sus objetivos globales.

4. Es necesario definir a los agentes en términos de sus comportamientos y roles, de modo que se les acaba confiriendo un cierto grado de inteligencia, adaptado a sus necesidades. (Gallego, Llorens, & Rizo, 2004)

    - Debido a que existen varios agentes que colaboran para cumplir objetivos individuales y de grupo, se debe crear una estructura jerárquica para diferenciar que procesos son prioritarios y quien los ejecuta.

"La necesidad de darle un carácter de ingeniería al desarrollo de sistemas multi agente, ha generado la aparición de diversas metodologías para el análisis y diseño de software basado en agentes". (Guzmán, Sánchez, & Torres, 2006)

En la **figura 13** se indica la clasificación de metodologías para SMA, según (Pavon, 2003)**.**

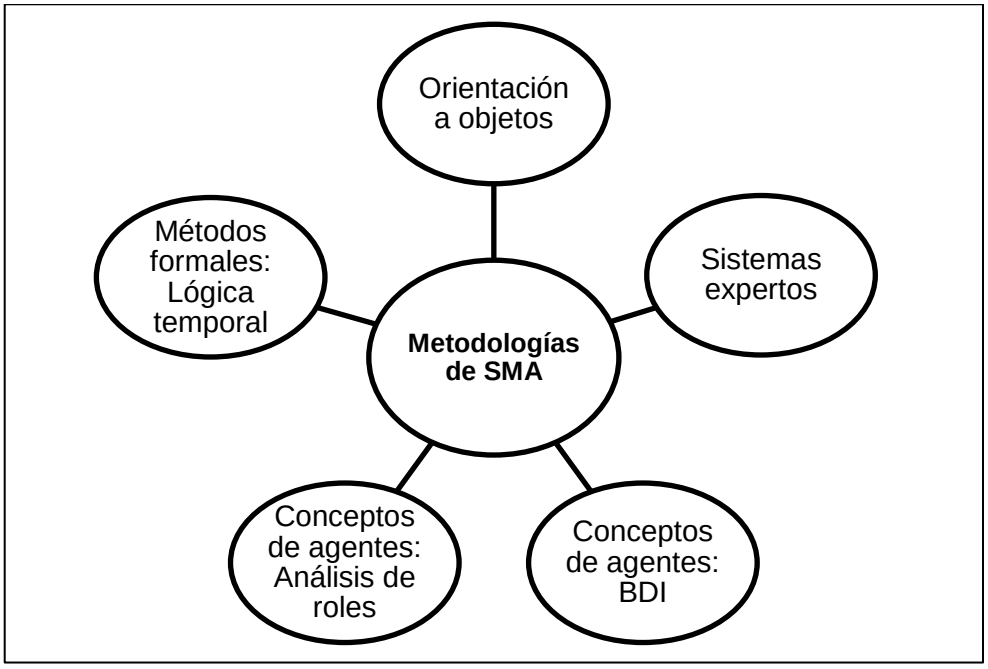


Figura 13: Clasificación de metodologías para SMA

Fuente: (Pavon, 2003)

Cada metodología tiene características de acuerdo a áreas concretas, si la experiencia del usuario está en el área de los objetos y existen clases definidas que se conceptualizan como un objeto, se debe usar metodologías orientadas a objetos debido a que poseen niveles de abstracción para simplificar la programación, en cambio, si se necesita jerarquías de agentes que contienen roles específicos que cumplen cada uno a nivel de autorizaciones, se usa metodologías enfocadas al análisis de roles, seguidamente si se desea construir agentes en referencia a lógicas de programación, usando lenguajes computacionales, el

criterio de selección va hacia métodos formales y para finalizar si se necesita a detalle obtener el razonamiento de un experto en un dominio concreto se debe usar sistemas expertos.

Para Juan Pavón, una de las grandes ventajas del uso de metodologías orientadas a objetos es que son más fáciles de aceptar por los ingenieros de software, ya que la mayoría de sistemas actualmente se construyen pensando en programación orientada a objetos, en este punto priman ciertas características; los recursos, la experiencia y las herramientas que ofrecen, por ejemplo Multi Agente System Engineering (MaSE), es una metodología que ofrece grandes beneficios al desarrollador por presentar arquitecturas concretas dependientes del dominio de aplicación. (2003, pp. 50-51)

De acuerdo a la clasificación presentada en la **figura 13,** en la **tabla 4** se muestra la categorización de metodologías de SMA.

Tabla 4: Metodologías para desarrollo de SMA{ XE "Tabla 4\: Metodologías para desarrollo de SMA" }

Orientadas a Objetos	Análisis de roles	Métodos Formales	Sistemas Expertos	Librerías de componentes
MaSE, Kendall, ODAC, Adelfe, Styx.	Gaia, AAll	SMART, DESIRE, Concurrent METATEM	MAS-CommonKADS, CoMoMAS	Vowel Engineering

Fuente: La autora.

La **tabla 5** presenta una comparación de metodologías de acuerdo a la clasificación presentada en la **tabla 4.**

Las metodologías a evaluarse son las siguientes, seleccionando una metodología de cada clasificación:

- ***MaSE***
- ***GAIA***
- ***Concurrent METAMEM***
- ***MAS-CommonKADS***
- ***Vowel Engineering***

Tabla 5: Comparación de metodologías para desarrollo de SMA de acuerdo a la clasificación

ORIENTADAS A OBJETOS	ANÁLISIS DE ROLES	MÉTODOS FORMALES	SISTEMAS EXPERTOS	LIBRERÍAS DE COMPONENTES
MASE	**GAIA**	**Concurrent METATEM**	**MAS-CommonKADS**	**Vowel Engineering**
Se divide en dos fases cada una consistente de diversas tareas. Las fases son: fase de análisis y fase de diseño.	Separa los conceptos que son entidades puramente abstractas, de lo que son conceptos concretos, o lo que es lo mismo, entidades que tienen un reflejo directo de sí mismas en la implementación del sistema, consideradas en dos fases de diseño y análisis.	Cada agente se programa mediante un conjunto de lógicas temporales en base a especificaciones de la conducta que deben exhibir los agentes.	Emplea 7 modelos diferentes, cada uno está basado en una teoría distinta, de acuerdo a la definición del sistema.	Se divide en subgrupos aplicando una técnica basada en vocales: A, E, I, O. A: Agente E: Entorno I: Interacciones O: Organizaciones
Se representa mediante diagramas de clases de agentes, diagramas de secuencia, diagramas de conversaciones. Son dos lenguajes de conceptualización,	La especificación y el diseño de los sistemas son independientes.	Se representa mediante fórmulas lógicas temporales basadas en el teorema de separación de Gabbay. Gabbay describe que cualquier fórmula lógica	Aplica varios modelos UML para la representación de diagramas.	Se representa mediante autómatas o por complejos sistemas de conocimiento.

uno de análisis y otro de diseño.		temporal arbitraria puede ser escrita en una lógica equivalente (pasado -> futuro).		
El desarrollo de los agentes se lo hace a concepción del desarrollador pero siguiendo las etapas de la metodología. Se basa en la creación de los agentes basados en roles. Es decir se crea agente por agente para al final crear el sistema final	El desarrollo aplica técnicas clásicas del paradigma orientado a objetos.	La ejecución es un proceso de hacer coincidir continuamente reglas contra una historia, y disparando esas reglas cuando los antecedentes están satisfechos. Cualquier consecuente futuro en tiempo instanciados convierte compromisos que posteriormente deben ser satisfechos, generando de forma iterativa un modelo para la fórmula integrada por las reglas del programa.	El nivel de detalle alcanzado en la descripción no es realizable sin el apoyo de herramientas de soporte. Lo que propone MAS-CommonKADS, entre otras cosas, no es una notación sino una lista detallada de elementos y relaciones a identificar en el sistema. Un desarrollador puede seguir la lista y generar la documentación requerida de forma manual, sin embargo el proceso es demasiado costoso y dado a errores.	Se desarrolla el sistema mediante un orden determinado que propone la metodología.
El desarrollo tiene un nivel de	El desarrollo es complejo,	El desarrollo tiene un nivel de	El desarrollo es exhaustivo a la	El desarrollo es incompleto

complejidad medio, además se adapta fácilmente a diversos problemas.	requiere que se conozca a profundidad la metodología dado el nivel de abstracción y el esfuerzo a invertir para pasar de una especificación GAIA hasta su implementación es alto.	complejidad muy alto, debido a que no existe documentación clara de cómo concebir la lógica temporal en los agentes, lo que requiere un estudio exhaustivo y un alto coste de comprensión de la lógica impuesta.	hora de detallar el sistema y además el proceso de desarrollo en la mayoría de los casos es más complejo que un conjunto de pasos.	*Vowel Enginering* solo proporciona las vocales como resumen de elementos a considerar en el desarrollo y un conjunto de tecnologías aplicadas.
Encaminado a la concurrencia y a la programación orientada a objetos. Los agentes no son considerados entes autónomos, proactivos y sociales, sino simples procesos que se comunican para conseguir el objetivo global del sistema. Los agentes son sólo una abstracción conveniente, que puede o no poseer inteligencia. En este sentido, los componentes inteligentes y no	Se utiliza para grandes aplicaciones, implementadas en diferentes tecnologías, bajo diferentes lenguajes, los servicios que proveen los agentes son estáticos y no varían en el tiempo.	Se basa en reglas de programa, mediante lógica temporal para predecir cuales son los comportamientos que deben ir presentando los agentes.	Aprovecha las ideas orientadas a objetos pero se necesita de un alto grado de conocimiento y experiencia en la aplicación de la metodología.	El objetivo es lograr librerías de componentes que den soluciones al diseño de cada uno de estos aspectos. A, E, I, O, U.

inteligentes se gestionan igualmente dentro del mismo armazón.

Fuente: La autora.

1.9 Ontologías utilizadas en la aplicación

- **FOAF**

Friend **O**f **A** **F**riend denominado “amigo de un amigo”. Es un esquema que define al dominio persona y las relaciones que tiene. Se trata de un estándar básico capaz de delimitar a un recurso como una persona, un documento y proceso con más detalle. No describe atributos físicos sino más bien los aspectos cualitativos relacionales en torno a una persona, como a quien conoce o que publicaciones ha realizado.

Como se menciona en (Foaf, 2014):

> El amigo de un amigo es la creación de una web de páginas legibles por máquina que describen las personas, las relaciones entre ellos y las cosas que crean y hacen, sino que es una contribución al sistema de información vinculada conocida como la web. (p. 1)

Es un proyecto de web semántica para crear un estándar que permite conceptualizar información referente a una persona, sus intereses y poder establecer una definición global de personas. “FOAF te permite compartir e información inter-conexión de diversas fuentes”. (Foaf, 2014)

De distribución libre y el más usado hoy en día por presentar información relevante. Está definido por el lenguaje RDF dentro del esquema XML.

- **Test_StressM**

Es un vocabulario que conceptualiza información en referencia al estrés que tiene una persona, exponiendo un cuestionario para predecir las causas físicas, psicológicas y comportamentales y en que niveles. El área de estudios a la que se enfoca es la académica.

Según (Ramos, 2013) está basada en las encuesta SISCO de estrés académico y contiene clases que representan dominios de conocimiento que son:

- SISCO: Es la encuesta que fue aplicada a los estudiantes de la titulación.
- Option_Other: Cuestionarios de preguntas para determinar el nivel de estrés de las personas y que no han sido consideradas en la encuesta.

- Questionary: Cuestionario de preguntas que se aplicaron basados en los parámetros de la encuesta SISCO.
- Situations: Conjunto de estímulos estresores.
- Strategy: Actividades para afrontar el estrés.
- Symptoms: Síntomas que presenta una persona con estrés.
- Test: Y el test de estrés de los estudiantes que se sometieron a la encuesta.

1.10 Apis y tecnologías usadas en el desarrollo de la aplicación

Apis

Llamadas también interfaz de programación de aplicaciones. Las Apis usadas en la aplicación son las siguientes:

- **Jena**

Un Api que permite el consumo de datos semánticos, de distribución libre y construida sobre Java.

Escribe datos sobre RDF y también tiene soporte para OWL. Además realiza consultas SPARQL.

Su arquitectura incluye:

  - Api para trabajar (leer, procesar, escribir) ontologías RDF y OWL
  - Motor de inferencia para razonar sobre ontologías RDF y OWL
  - Estrategias de almacenamiento flexible para almacenar tripletas RDF en memoria o fichero
  - Motor de queries compatible con especificación SPARQL

- **Graph Facebook Api**

Es un Api que permite recuperar datos de Facebook. Se debe crear una cuenta como *Facebook Developer* (https://developers.facebook.com/) y desarrollar una aplicación para consultar la información de interés de acuerdo a los permisos que se otorgue.

Los resultados de las consultas se presentan en formato JSON.

- **LinkedIn Api**

Es un Api que permite recuperar datos de LinkedIn. Se necesita crear una cuenta en *LinkedIn Developers* (https://developer.LinkedIn.com/) y desarrollar una aplicación para consultar la información de interés referente a intereses, trabajos y conexiones de acuerdo a los permisos que se otorgue.

Los resultados de las consultas se presentan en formato JSON.

- **Google Web Search Api**

Es un Api de Google que permite el consumo de servicios usando el esquema de recopilación de información del Api de Google. Se necesita crear una cuenta en Gmail para acceder a las características de *Google Developer* (https://developers.google.com/) y gestionar los servicios a consumir.

Los resultados de las consultas se presentan en formato JSON o XML.

- **JAX-RS: Java API para RESTful Web Service**

Es un lenguaje de programación de Java Api para la creación de servicios web de acuerdo a los estándares REST.

El Api proporciona las siguientes anotaciones:

  - **@Path** para especificar el recurso
  - **@GET**, **@PUT**, **@POST** y **@DELETE** para especificar las peticiones del recurso.
    - @GET para recuperar información de un recurso
    - @PUT para actualizar información de un recurso
    - @PUT para crear información de un recurso
    - @DELETE para eliminar información de un recurso
  - **@Produces** la respuesta del recurso
  - **@Consumes** los tipos de solicitud

Tecnologías

Las tecnologías usadas en el desarrollo de la aplicación son las siguientes:

- **Java**

Es un lenguaje de programación robusto que actualmente es el más común. De propósito general, concurrente, orientado a objetos, arquitectónico, diseñado para ofrecer al desarrollador un medio comprensible y escalable de distribución libre. Corre bajo cualquier dispositivo sobre Java Virtual Machine (JVM).

Entre las principales características:

  - Es orientado a objetos
  - Fácil de usar
  - Soporte para múltiples sistemas operativos.

- Soporte para trabajo en red.
- Muchas Apis externas se compilan en Java.

- **REST**

REST es comúnmente usado en sistemas distribuidos y entornos www. “Mantener los principios de la filosofía REST supone poner a disposición de los usuarios un API que es funcional y accesible mediante llamadas a métodos web”. (Higes, 2011)

Cada petición es llamada mediante una URI única por cada servicio. “REST, gracias a su simplicidad, se está imponiendo haciendo que muchos de los grandes agentes de internet estén migrando sus tecnologías SOAP a REST. Dicha arquitectura puede resumirse en dos grandes características: entidades y principios REST”. (Higes, 2011)

“REST no es exactamente un protocolo de comunicaciones, sino más bien una técnica de diseño de las comunicaciones en sistemas distribuidos”. (Fielding, 2000) SOAP está especialmente indicado para diseñar las comunicaciones bajo el esquema RPC, es decir, invocando la ejecución de funciones remotas. “REST un método para comunicar servicios más eficiente y rápido que SOAP” (Pautasso, Zimmermann, & Leymann, 2008) está orientado al acceso y manipulación de recursos, a diferencia de SOAP. “REST es mucho más sencillo de implementar, siendo más escalable que SOAP” (Espí, 2013) y “permite el acceso a todas las acciones a través de URL” (Camacho, s.f)

Las entidades REST son:

- Recurso: Es cualquier cosa que puede ser nombrada, posee un identificador único URI y todos los recursos comparten una interfaz homogénea para ser fáciles de gestionar, un recurso tiene múltiples representaciones.
- URI: Es un identificador único para cada recurso por ejemplo: www.people.com/name.
- Representación: Son varias las representaciones para un mismo recurso, es por eso que cada servicio debe contemplar parámetros únicos necesarios para cumplir con las peticiones, por ejemplo; mediante el paso de parámetro nombre se obtiene la información acerca de cierta persona.
- Conectores: Encapsulan las actividades de un recurso. Es un mecanismo abstracto que hace posible la comunicación, coordinación y cooperación entre componentes.

Por su parte, cada recurso tiene n representaciones posibles, puede ser XML, JSON, que son los más utilizados.

“REST permite cualquier formato de mensaje en la transferencia de información. Aunque en sus inicios se utilizaba XML, en la actualidad está ganando terreno el empleo de JSON como formato más ligero y ágil a la hora de ser procesado”. (Espí, 2013)

- **Rest Ful Web Services**

Es una arquitectura que trata de exponer servicios mediante un URI. Trabaja a nivel de servidor, y presenta para el cliente lenguajes de marcado y jerarquización, entre los más comunes y usados, XML y JSON.

Los vocabularios de recuperación son los siguientes:

- GET
- POST
- PUT
- DELETE

El más popular GET, que sirve para recuperar recursos.

Ha sido un gran avance para las aplicaciones, por qué las consultas ya no dependen de sentencias SQL u otros métodos de consulta, sino, expone una URI que debe ser consumida mediante un lenguaje cliente simple, como JQuery o Javascript.

CAPÍTULO 2

FRAMEWORK DE SISTEMA MULTI AGENTE

2. Introducción

En este capítulo se describe las etapas para el diseño del framework.

Primero, se apoya en los pasos de análisis y desarrollo de requerimientos de la metodología MaSE de Scott DeLoach. MaSE ayuda a identificar componentes globales, comunes, roles y conversaciones, con el fin de construir un modelo que permita desarrollar SMA, obteniendo un patrón definido para la inclusión de nuevos agentes en la fase de análisis y finalmente un modelo de interacción de todos estos agentes en la fase de diseño. El reúso de MaSE permite modelar aplicaciones en diversos dominios.

Seguidamente está la producción de los agentes. Antes de empezar con la programación, se presenta una etapa llamada *lineamientos para producir los agentes*, en donde se da un entendimiento de cómo ir asociando los diagramas en los 3 niveles de programación.

El desarrollo del framework modela a los SMA en *servicios de agentes*. El primer nivel expone los parámetros que debe recibir cada agente para cumplir una conversación, el segundo nivel crea instancias de clases de objetos para cumplir con las peticiones, y el tercer nivel contiene los métodos y clases necesarios para llegar a cumplir los objetivos, representa también los conceptos de la ontología. El tercer nivel se lo representa mediante clases de objeto, ya que al ser MaSE una metodología orientada a objetos, es preciso concebir el desarrollo orientado a objetos.

La implementación se la realiza en el lenguaje Java, bajo servicios RESTful.

La **figura 14** muestra la composición del framework:

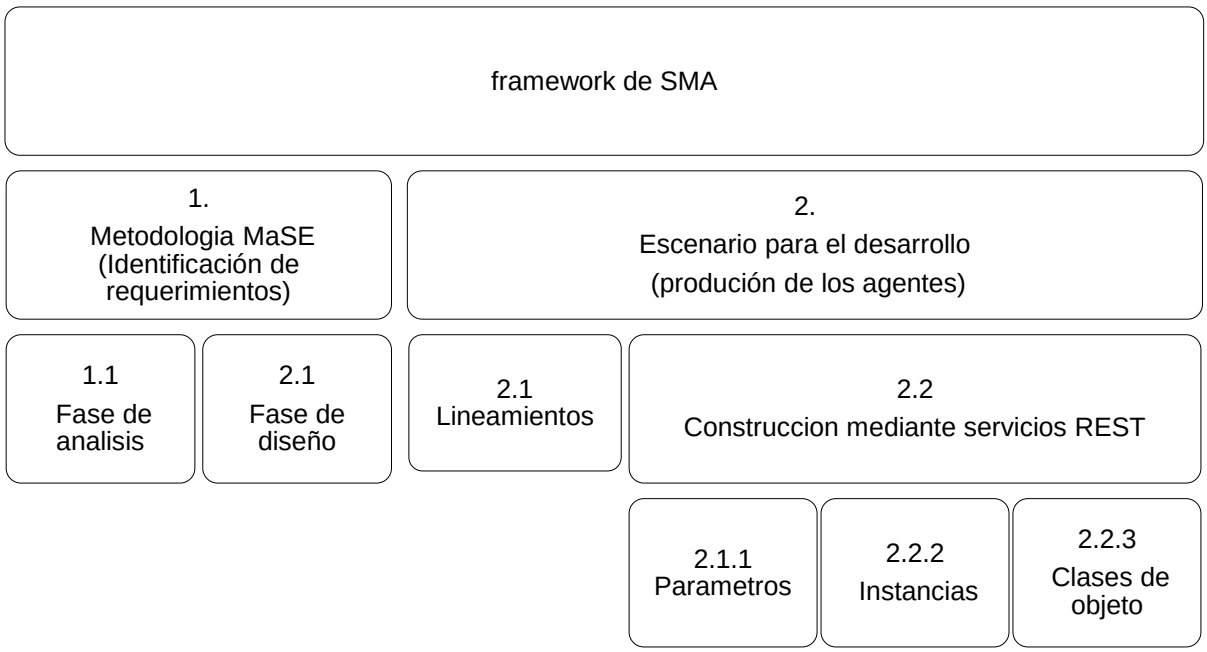


Figura 14: Composición del framework de SMA

Fuente: La autora.

2.1 Metodología del framework de sistema multi agente: MaSE

La metodología elegida es MaSE (Multi agent System Engineering) o Ingeniería de sistemas multi agente, es la etapa de identificación de requerimientos dentro del framework.

MaSE es:

Según (Julián & Botti, 2003):

> MaSE (Multiagent System Engineering) es una metodología desarrollada en el Air Force Institute of Technology. Dicha metodología trata de cubrir todas las etapas en el proceso de construcción de un sistema multi agente, partiendo de la especificación del mismo hasta su implementación.

Según (DeLoach, 2001):

> MaSE (Multi-agent systems Software Engineering) parte del paradigma orientado a objetos y asume que un agente es sólo una especialización de un objeto. La especialización consiste en que los agentes se coordinan unos con otros vía conversaciones y actúan proactivamente para alcanzar metas individuales y del sistema.

Esta forma de definir la metodología considera que los agentes son una abstracción conveniente, que pueden poseer o no inteligencia. Así se maneja componentes “inteligentes y no inteligentes dentro del mismo marco”. (Gómez, 2003) Y se puede implementar la cantidad de agentes que sea necesario desde diversas perspectivas arquitectónicas pero bajo el mismo diseño.

Una característica importante que resalta Scott DeLoach, es que la metodología se distingue en que primero diseñamos los componentes generales del sistema antes de definir realmente el propio sistema. (2000)

El uso de la metodología MaSE permite diseñar primero cada uno de los agentes y luego realizar su integración, esta acción permite obtener un “comportamiento coordinado de un sistema de agentes individuales para luego proveer un comportamiento como sistema multi agente” (DeLoach, 2000) debido a que “un agente sólo se especifica como una entidad de comunicación activa que desempeña funciones dentro de los grupos”. (Ferber & Gutknecht, 1999)

Como explican (Di Leo, Jacobs, & Scott, 2002) la metodología también permite integrar ontologías:

> MaSE amplia la investigación para el uso de ontologías para la especificación de dominio de la información. Las extensiones permiten al diseñador especificar el flujo de información mediante el *uso de los objetos de la ontología como parámetros en*

conversaciones de agente. El desarrollador puede garantizar la funcionalidad del sistema mediante la verificación de que cada agente tiene la información necesaria para cumplir con los objetivos del sistema.

La **figura 15** muestra la metodología MaSE.

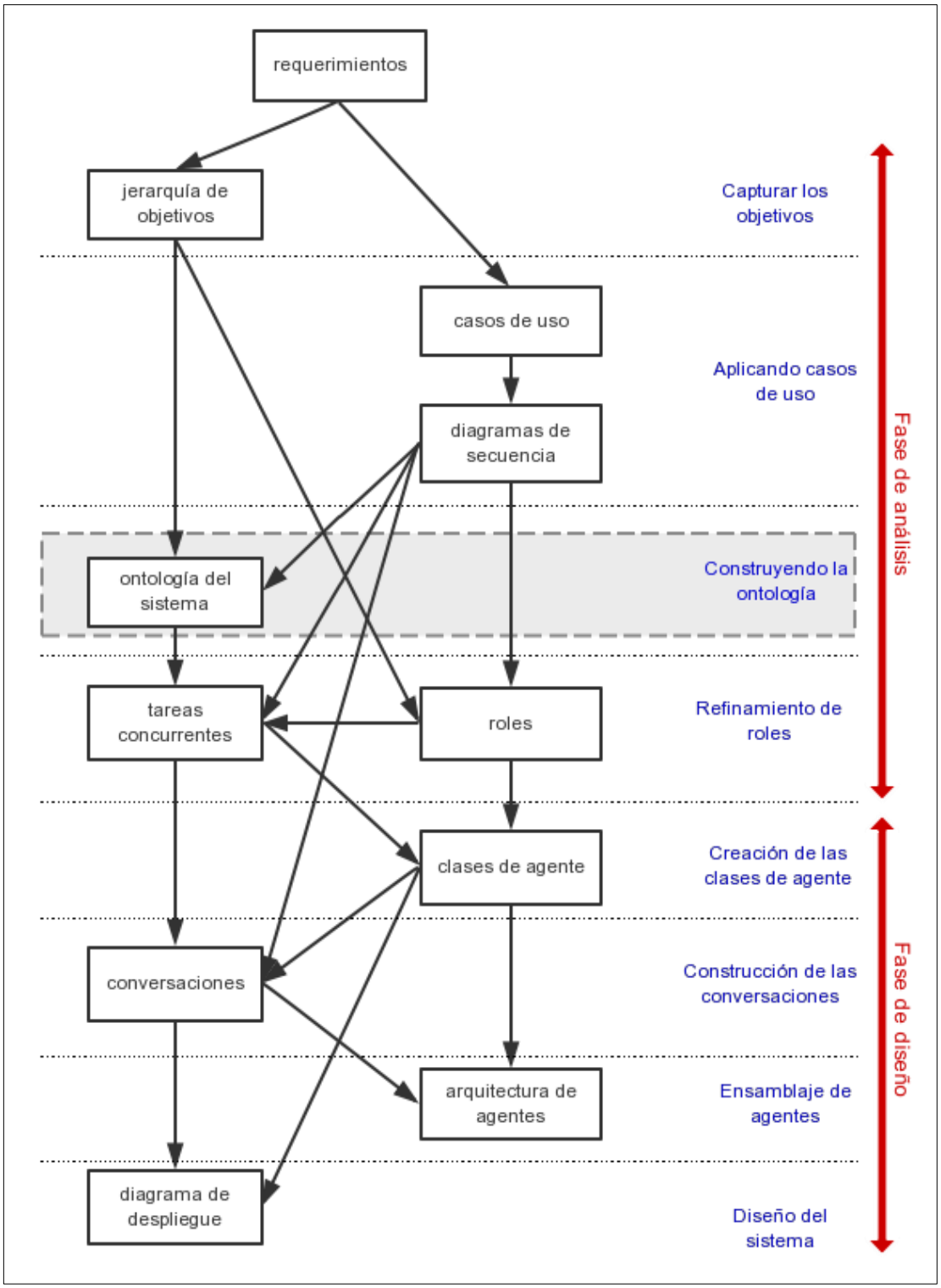


Figura 15: Metodología MaSE

Fuente: (DeLoach, 2000)

La metodología MaSE es la alternativa más completa de todas las consideradas cuando no se especifica un dominio de aplicación en común. MaSE, parte de un esquema principal que son dos fases perfectamente estructuradas y complementarias. El proceso de desarrollo está bien descrito y detallado, los modelos gráficos tienen una expresividad adecuada y están muy bien coordinados, y todo en su conjunto está basado en ideas de agentes reales. Cuenta con la parte de representación, que permite trabajar con todos los modelos de forma gráfica, interactiva, además de ser capaz de convertir modelos de análisis en modelos de diseño.

2.1.1 *Fase de Análisis.*

Fase de análisis, que es la primera fase en MaSE consta de cuatro pasos: 1) capturar los objetivos, 2) capturar los casos de uso, 3) obtener la ontología del sistema y 4) refinar los roles.

“El propósito de la Fase de Análisis es producir un conjunto de roles cuyas tareas describan que es lo que el sistema debe hacer para cumplir con las exigencias globales”. (DeLoach, Wood, & Sparkman, 2000)

- **Capturar los objetivos**

En esta fase surgen preguntas como: ¿Qué se quiere lograr con el sistema?, ¿Cuántos agentes se necesita?, ¿Qué deben hacer los agentes?, las respuestas dan un conjunto de objetivos.

“En MaSE se decide basar la fase de análisis en objetivos, porque los objetivos del sistema son más estables que las funciones, procesos, o estructuras de información que con frecuencia cambian con el tiempo”. (Kendall, Palanivelan, & Kalikivayi, 1998) Este cambio depende de cómo se encuentre el entorno y el desempeño del sistema en el mismo.

Los objetivos del sistema se pueden extraer desde un documento de especificación de requerimientos, pero no se toma en cuenta las actividades, tareas o funciones que conllevan. Además, también se puede abstraer estos objetivos de un conjunto de requisitos funcionales. Una vez que se han los ha identificado y ordenado jerárquicamente se debe ir por los sub objetivos.

El entregable de este paso es un *diagrama de objetivos*.

La estructura básica que se debe tomar en cuenta para capturar los objetivos es:

1. <u>Identificar objetivos</u>

El primer paso al capturar objetivos es identificarlos, de tal forma, que se plasmen los principales requisitos funcionales del sistema. Además de identificar cuáles son los

escenarios en los que estos objetivos deben desenvolverse. Informar, detectar, almacenar y notificar pueden ser unos de los objetivos que el sistema puede cumplir.

2. Estructurar objetivos

Una vez identificados los objetivos el siguiente paso es estructurarlos. Partiendo de los objetivos generales y descomponiendo en objetivos específicos. La estructura que se lleva en este punto es que de cada objetivo identificado en el sistema se debe descomponer en sub objetivos que colaboran funcionalmente con el principal.

El objetivo principal es la necesidad del agente, el por qué necesitamos realizar la creación, el fin, el propósito a cumplir, se lo especificar como el objetivo raíz, de él, se desprenden sub objetivos necesarios para cumplir los propósitos. Los objetivos deben tener jerarquía y ser organizados mediante números. El principal será el objetivo 1, y de ahí se descomponen los sub objetivos como 1.1, 1.2.

Los objetivos giran en torno a todo tipo de necesidades del sistema, puede haber objetivos orientados a reportes, monitoreo, notificaciones, restricciones, almacenamiento, filtrado, petición… Se debe elegir qué objetivo pertenece a otro y ver como se organizan jerárquicamente. Para sistemas complejos se crea un diagrama de objetivos por cada agente. Entendiendo que la integración de estos diagramas es la composición de objetivos del sistema multi agente.

Este punto se lo puede ejemplificar con un diagrama de objetivos como se muestra en la **figura 16**, el ejemplo trata de un sistema multi agente para detección y notificación de login erróneo y violación de login.

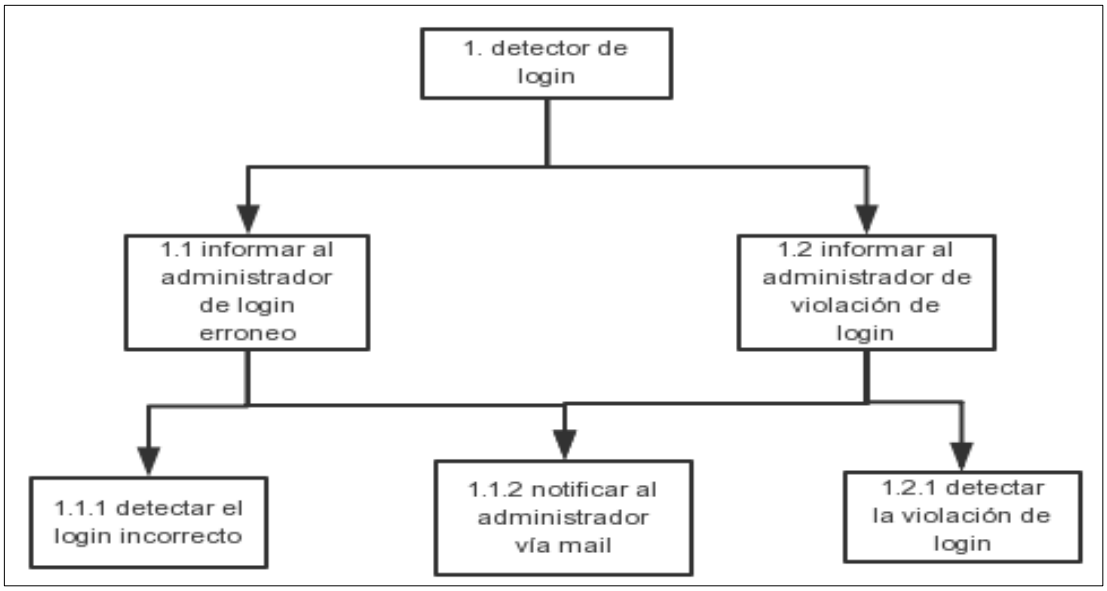


Figura 16: Diagrama de objetivos MaSE

Fuente: La autora.

- **Capturar los casos de uso**

En esta fase se diseñan los casos de uso, que es una especificación de todos los objetivos encontrados en la etapa anterior, para definir el comportamiento deseado del sistema.

Se muestra lo que cada objetivo debe cumplir, con qué tipo de rol y que comunicación tiene este objetivo en relación con otros. Se define esta fase como un conjunto de eventos que especifican como se compone el sistema.

De cada objetivo se descomponen tareas por cumplir. Se debe estructurar los casos de uso en diagramas de secuencia mostrando la sucesión de eventos entre roles y, como resultado, define la mínima comunicación necesaria entre ellos.

Se desprenden dos tareas importantes para cumplir este punto:

1. Crear los casos de uso

El primer paso al identificar los casos de uso, es crearlos. Este paso surge como una necesidad de describir con más detalle las funcionalidades en MaSE, mejor conocidas como objetivos del sistema. Si ocurre algún error en esta etapa, se debe regresar a la anterior, para identificar qué objetivos se estableció mal, o que objetivos faltaron.

“El objetivo de crear casos de uso es para identificar vías de comunicación, no para definir todas las posibles combinaciones de eventos y de datos en el sistema”. (Self & DeLoach, 2003)

Un caso de uso siempre tiene asociados 1 o más objetivos. Los casos de uso deben ser positivos o negativos, porque también es importante conocer cómo debe comportarse el sistema frente a posibles errores.

De los objetivos expuestos en la etapa anterior, **figura 16**:

1. Detector de login

1.1 Informar al administrador de login erróneo

1.1.1 Detectar el login incorrecto

1.1.2 Notificar al administrador vía mail

1.2 Informar al administrador de violación de login

1.2.1 Detectar la violación de login

1.2.2 Notificar al administrador vía mail

Del ejemplo se puede encontrar n casos de uso. Para el objetivo seleccionado se considera el caso de uso: Detectar el login erróneo e informar al administrador vía mail el problema. El caso de uso lleva asociados los objetivos 1.1, 1.1.1 y 1.1.2.

2. Crear los diagramas de secuencia

Los diagramas de secuencia especifican como ocurren los eventos en el sistema y se realizan por cada caso de uso identificado.

Caso de uso 1: Detectar el login erróneo e informar al administrador vía mail el problema

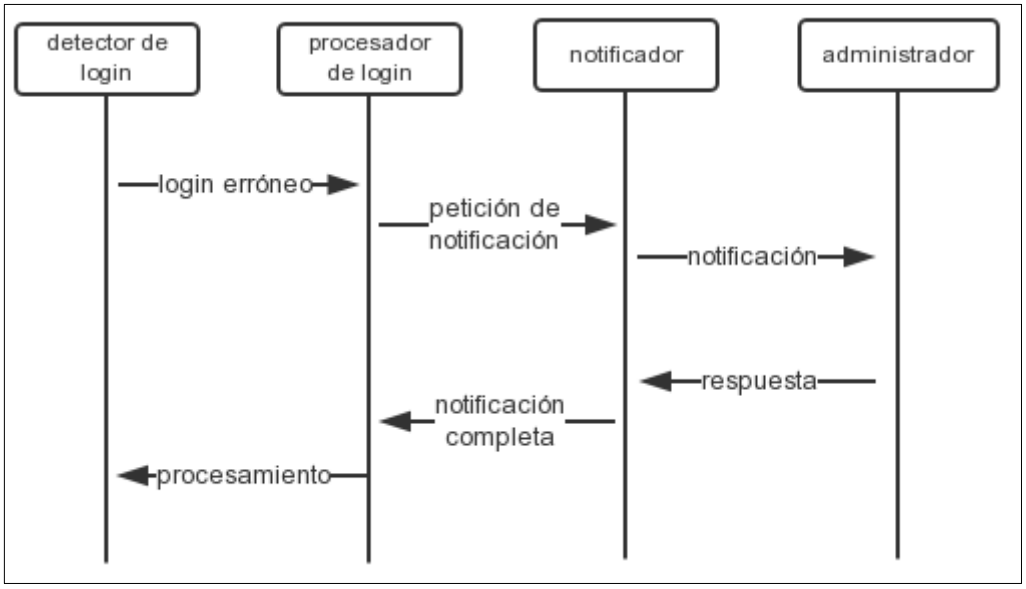


Figura 17: Diagrama de secuencia MaSE

Fuente: La autora.

En el diagrama de secuencia de la **figura 17** se observa el proceso que se debe realizar para cumplir con el caso de uso. El detector de login notifica al procesador el login erróneo, el cual ejecutara los procesos necesarios para enviar el mensaje al notificador informando la causa del problema. Finalmente, el notificador envía el mensaje al administrador, para cumplir con la secuencia de envió.

Una vez que el administrador ha leído el mensaje se procesa la información y se ejecuta la respuesta, el notificador envía el mensaje al detector de login, según el mensaje del administrador se enviará las acciones a ejecutar para que el detector de login interrumpa el login erróneo.

- **Obtener la ontología del sistema**

"La ontología especifica el dominio de información del sistema multi agente" (Di Leo, Jacobs, & Scott, 2002)

Cuando realizamos aplicaciones software usando base de datos se realiza la especificación del modelo entidad relación del sistema. En esta paso, MaSE considera conveniente e importante realizar la especificación del modelo de dominio de información.

El paso de mensajes entre agentes se realiza mediante conversaciones. Cada conversación define parámetros necesarios para entender las peticiones. Estos parámetros son objeto de algún tipo, y sin una especificación de dominio de información, la metodología no puede dirigirse a la información contenida en estos parámetros.

Como definen Di Leo, Jacobs, & Scott, el diseño de la ontología en este paso es necesario debido a que se puede usar términos del diagrama de objetivos, casos de uso y diagramas de secuencia como posibles conceptos de la ontología. La ontología resultante se puede utilizar para crear tareas en la etapa de refinación de roles en MaSE. (2002)

Las tareas en el refinamiento de roles, indican el paso de parámetros, por lo que se coloca después de la construcción de la ontología para poder especificar el tipo de los parámetros como las clases de la ontología.

Se toma en cuenta:

- Las clases en una ontología describen los objetos en la vista del sistema del dominio de información.
- Los objetos de la ontología pueden funcionar como parámetros para las conversaciones.
- Las funciones y objetos constantes definen las propiedades y relaciones entre estos objetos.
- Los axiomas describen cualquier relación o restricción adicional a los conceptos en la ontología

Las tareas a realizar en este punto son las siguientes:

1. Determinar el propósito y el alcance de la ontología

En esta tarea se describe por qué se está realizando la ontología, los posibles usuarios que van a participar y la información que contiene.

El propósito describe por que la ontología existe y el alcance define el nivel de detalle con que la ontología representa los objetos.

La ontología ejecuta acciones asociadas a consultas, almacenamiento o eliminación.

2. Recopilar información

Se crea una lista de conceptos y términos que la ontología debe describir. La lista surge del análisis de la jerarquía de objetivos, los casos de uso y diagramas de secuencia realizados en la etapa anterior.

Las acciones que se realizan en el diagrama de despliegue pueden ser parte de la información que se trasmite al sistema.

3. Construir y validar la ontología

De la lista de términos anteriores, se agrupa la información en clases. Al crear la ontología se debe especificar solo los conceptos y términos necesarios para cumplir con los objetivos del sistema.

“La ontología no debe especificar todos los atributos de un ser humano, tales como la altura, la edad y el peso, cuando el sistema sólo requiere el nombre de un humano para funcionar (…) se debe determinar también si alguna de las ontologías existentes pueden satisfacer las necesidades del sistema. El beneficio de usar una ontología existente es que el sistema es interoperable, en términos de paso de datos, con cualquier otro sistema que utiliza el mismo modelo de datos.”. (Di Leo, Jacobs, & Scott, 2002)

Después, se debe validar si la ontología creada o escogida cumple con las necesidades del sistema, esta validación se puede realizar, haciendo un análisis de los objetivos, diagramas de secuencia y casos de uso de las etapas anteriores.

- **Refinar los roles**

El tercer paso de la fase de análisis es transformar los diagramas de secuencia y casos de uso, en roles y sus funciones. Se debe asegurar que se ha identificado todos los roles necesarios, para después desarrollar las tareas que definen el comportamiento de los mismos y los patrones de comunicación. “Los objetivos del sistema estarán satisfechos si cada objetivo se asocia a una función y cada rol es interpretado por una clase de agente”. (DeLoach, Wood, & Sparkman, 2000)

Cada objetivo debe tener un rol y cada rol debe estar desempeñado por al menos un agente. Pese a que, generalmente, cada objetivo es relacionado a un rol individual, existen situaciones en las que conviene asignar más de un objetivo a un mismo rol, por motivos de eficiencia.

El resultante de esta fase es un diagrama de roles, que se lo observa en la **figura 18**:

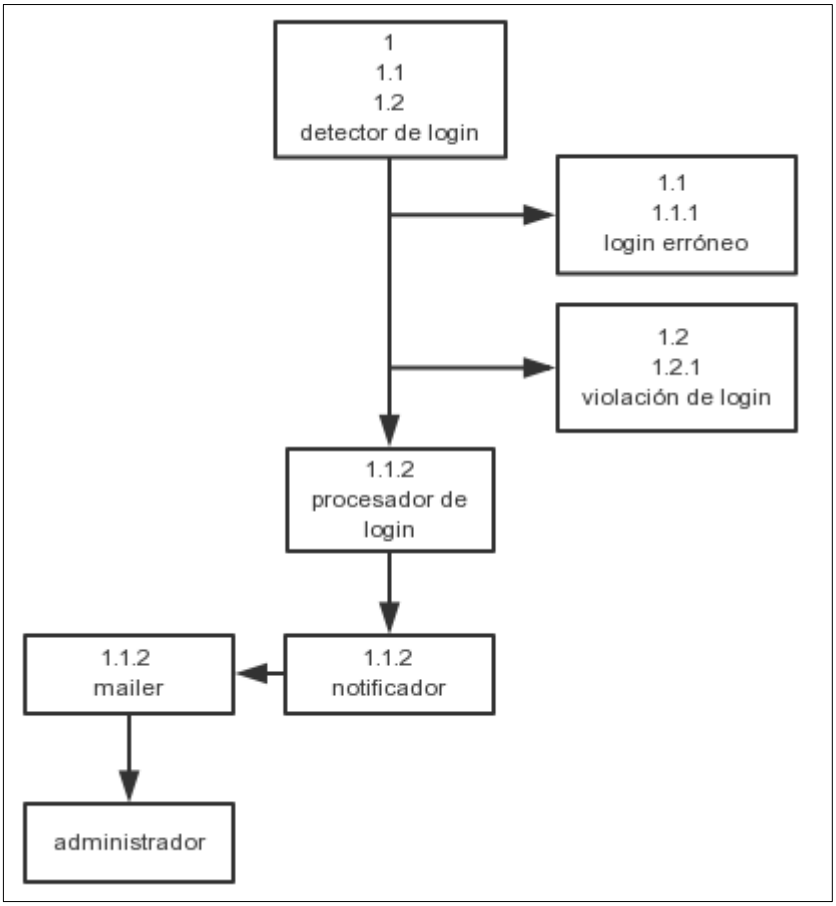


Figura 18: Diagrama de roles MaSE

Fuente: La autora.

2.1.2 *Fase de diseño.*

Fase de diseño, que es la segunda fase en MaSE tiene cuatro pasos: 1) creación de las clases de agentes, 2) construcción de conversaciones, 3) ensamblaje de las clases de agente y 4) diseño final del sistema.

El producto de estas etapas serán diagramas de clases de agentes, que enumeran los agentes del sistema y sus posibles conversaciones, diagramas de conversaciones donde se detalla las conversaciones que el sistema debe cumplir, diagrama de arquitectura para saber las clases a modelar y diagrama de despliegue que el sistema global mostrando su localización y composición.

La fase de diseño se adecua desde la fase de análisis donde se recopila la información, ahora se debe diseñar la arquitectura necesaria para desarrollar las funciones y roles que se han establecido.

“El objetivo es convertir los roles y funciones de manera que permitan la implementación del modelo. Consiste en diagramas de clases y conversaciones entre los agentes, por lo que se

puede construir y finalmente implementar los agentes". (C. S, Araújo, Gomes, da Costa, & Werneck, 2009)

Cuatro son los pasos que se deben seguir en esta fase:

- **Creación de clases de agente**

En ese paso se crean clases de agentes de las funciones definidas en la fase de análisis, el resultado final es un diagrama de clases de agente.

"Una clase de agente es una plantilla para un tipo de agente en el sistema y es análoga a una clase de objeto en la orientación a objetos. Un agente es una instancia real de una clase de agente". (DeLoach, Wood, & Sparkman, 2000)

Las clases de agente son los roles que va a desempeñar el agente, unido a una pequeña representación de las conversaciones que va a tener el agente con otros agentes, para cumplir sus objetivos tanto individuales como globales.

Por cada función que se haya identificado en la fase de análisis debe existir una correspondencia con una clase de agente. Los agentes de la misma clase pueden también desempeñar diferentes funciones, pero deben contemplar un mismo rol. Es decir, un agente A tiene que cumplir un rol A desempeñando las funciones A, B pertenecientes a una sola clase de agente.

En esto paso también se identifica las conversaciones en la que los agentes participan, aunque no se definen todos los detalles porque estos están plasmados en el siguiente paso de la fase de diseño.

El resultante de este paso es un diagrama de clases de agente, la diferencia entre los diagramas de clases, es que los diagramas de clases de agente vienen identificados por roles.

Para el ejemplo establecido de la **figura 16**, se tiene los siguientes agentes identificados:

Agente A: encargado de realizar la notificación de login.

Agente B: encargado de realizar la comprobación de login incorrecto o violación de login.

Los agentes y sus roles se representan en el diagrama de clases de agente de la **figura 19**.

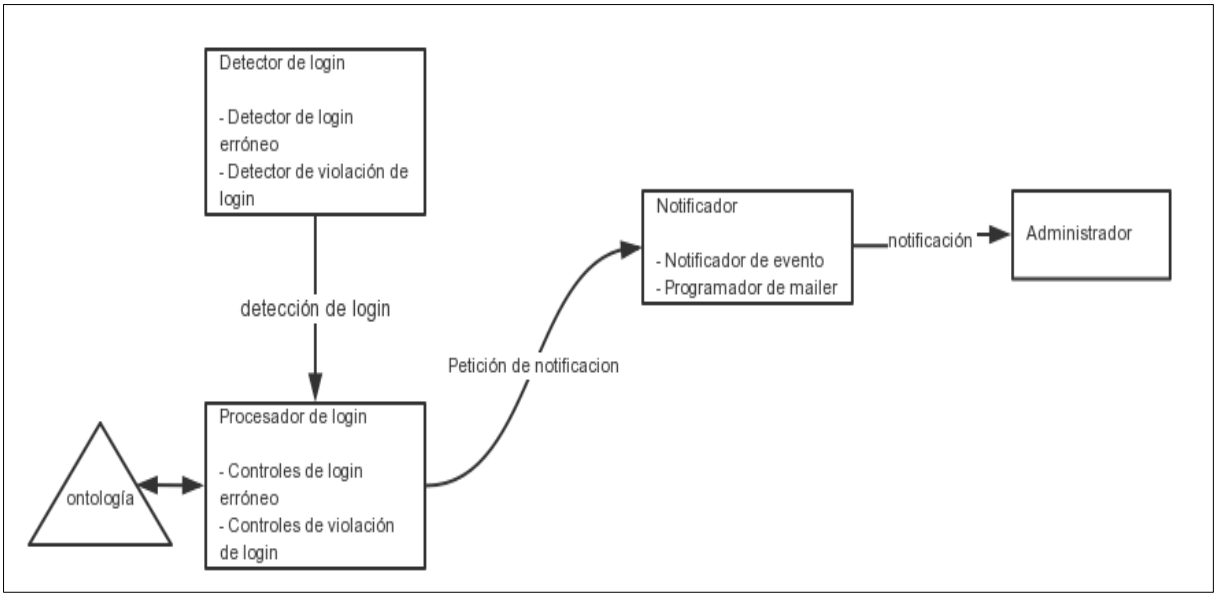


Figura 19: Diagrama de clases de agente MaSE

Fuente: La autora.

Como se explicó con anterioridad el diagrama también debe llevar una identificación de las conversaciones que surgen en el agente, en el esquema de la **figura 19** se muestra las posibles conversaciones: detección de login, petición de notificación y notificación. Que se irán complementando en el siguiente paso construcción de conversaciones.

- **Construcción de conversaciones**

Las conversaciones para los agentes son la vía de comunicación y permiten entender cómo estos interactúan para cumplir con sus funciones. En la fase anterior no se especifica a detalle cómo es la conversación entre los agentes, simplemente se asevera que un agente necesita de una.

Se pueden entender las conversaciones como un autómata finito de estados, un autómata tiene inicio, fin y las aristas o rutas que son los pasos a seguir para pasar de un estado de inicio a un estado 1, del estado 1 al estado 2, y así hasta llegar al estado de fin. Una conversación MaSE define un protocolo de coordinación entre dos agentes. Específicamente, una conversación consiste en dos diagramas de clases de comunicación, uno para el emisor y otro para el receptor.

Para el ejemplo de la **figura 19**, se corresponde el diagrama de conversaciones de la **figura 20**:

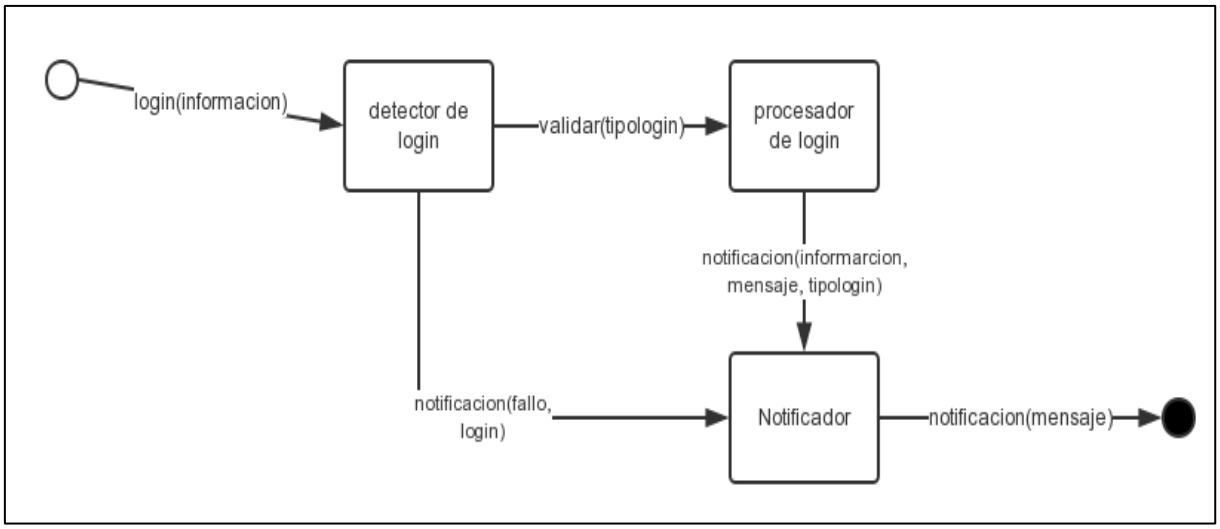


Figura 20: Diagrama de conversaciones MaSE

Fuente: La autora.

Como se mencionó anteriormente Mase considera importante también especificar las acciones a realizar ante posibles errores y ofrecer los controles necesarios mediante conversaciones. En conversaciones el diseñador se enfrenta a un dilema entre tener muchas conversaciones sencillas o ideas complejas. “Si el sistema tiene un gran número de simples comunicaciones, éstas deben implementar una serie de conversaciones más pequeñas”. (DeLoach, Wood, & Sparkman, 2000)

- **Ensamblaje de las clases de agente**

En la fase de diseño, etapa ensamblaje de las clases de agentes ya se requiere ir definiendo la estructura interna del agente. Esta etapa tiene consigo dos sub tapas:

1. Definir la arquitectura del agente.
2. Definir los componentes que constituyen la arquitectura del agente.

MaSE no propone una arquitectura para el agente, permite que las arquitecturas se desarrollen de acuerdo a los requerimientos que se presenten, es decir, es flexible, se puede adaptar a arquitecturas existentes o crear nuevas arquitecturas. Además los componentes se pueden reutilizar en el caso de ser existentes. Y si la complejidad es grande se puede tener sub arquitecturas. La arquitectura interna y la definición de los componentes debe ser consistente con las conversaciones definidas en el paso anterior.

En la **figura 21** a continuación se observa el diagrama de arquitectura.

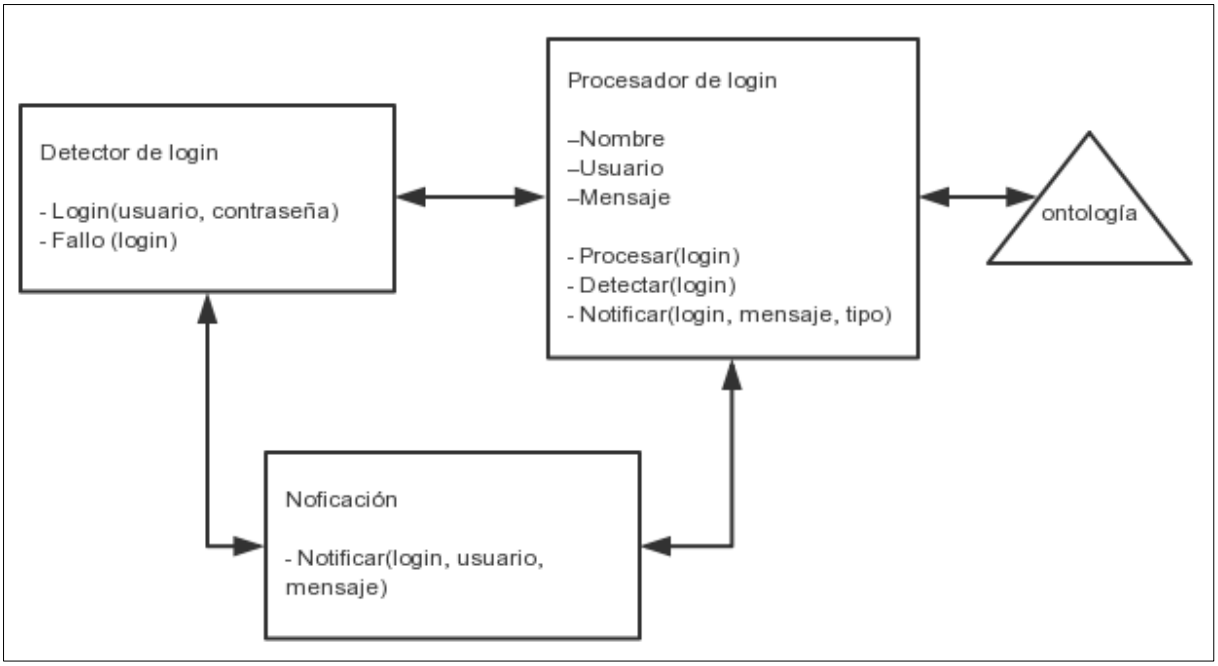


Figura 21: Diagrama de arquitectura MaSE

Fuente: La autora.

La arquitectura de los agentes está formada de componentes, en la **figura 21** se observan 3 agente a crearse y los métodos necesarios para cumplir sus objetivos.

- **Diseño final del sistema**

El último paso en la metodología es el diseño del sistema. En este paso se recopila toda la información anterior, los diagramas de clases, los diagramas de conversaciones, la arquitectura… para finalmente tener el diagrama de despliegue. La integración de los agentes en el SMA.

El diagrama de despliegue muestra los números, los tipos y donde se van a colocar los agentes en relación al sistema global. El diseño del sistema es el paso más simple dentro de la metodología, debido a que en las etapas anteriores se plasma ya las funciones que cada agente debe cumplir.

Los diagramas de despliegue de agentes, describen un sistema completo basado en agentes. Al llegar a este paso en MaSE se ha comprendido a totalidad como el sistema multi agente funciona en sí, cuál es su coordinación en relación a los demás agentes y la interacción que deben tener.

En la **figura 22** se observa un ejemplo de diagrama de despliegue.

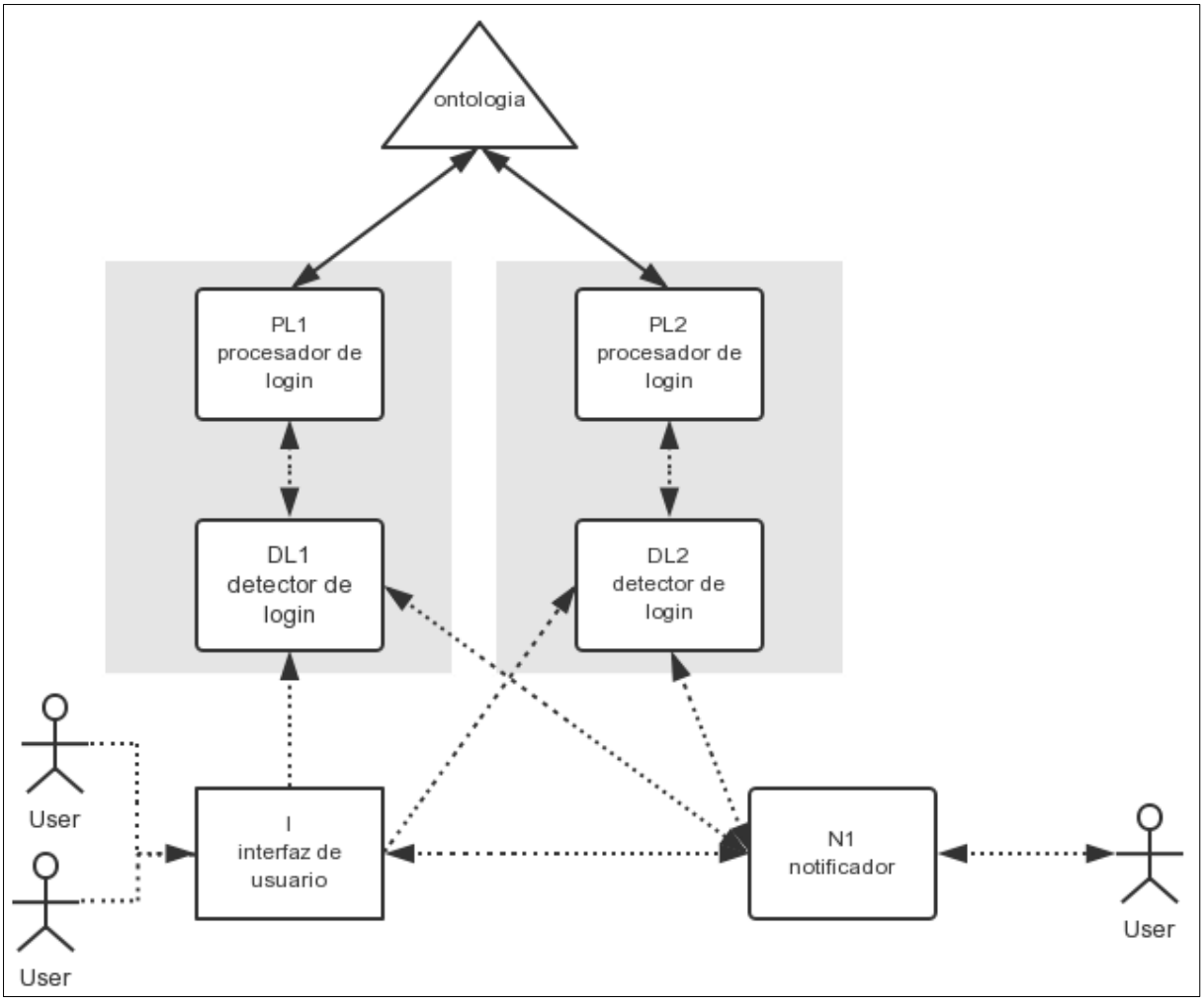


Figura 22: Diagrama de despliegue MaSE

Fuente: La autora.

En la **figura 22** se define la localidad de los sistemas en relación a los agentes y la interacción que tiene cada uno. Para el ejemplo resuelto en la metodología se supone que se tiene dos tipos de agente de detección y procesamiento geográficamente separados y un agente que realiza tareas de notificación.

Los usuarios acceden desde la interfaz de usuario para realizar el login, después el detector de login envía al procesador de login la información necesaria para que pueda comprobar si se está realizando un login erróneo o una violación de login. Los datos se procesan en la ontología. Seguidamente la información regresa al detector de login, el cual envía la información al notificador para que pueda alertar al administrador.

Finalmente el administrador enviará información para que se realicen las ejecuciones respectivas que puedan evitar la pérdida de información.

2.2 Escenario para el desarrollo de los agentes

Una vez ejecutada la metodología con sus fases de análisis y diseño con MaSE, los diagramas están listos para llevarse a la programación. Como MaSE es una metodología orientada a objetos se debe concebir el desarrollo orientado a objetos.

La **figura 23** muestra el escenario para producir los agentes.

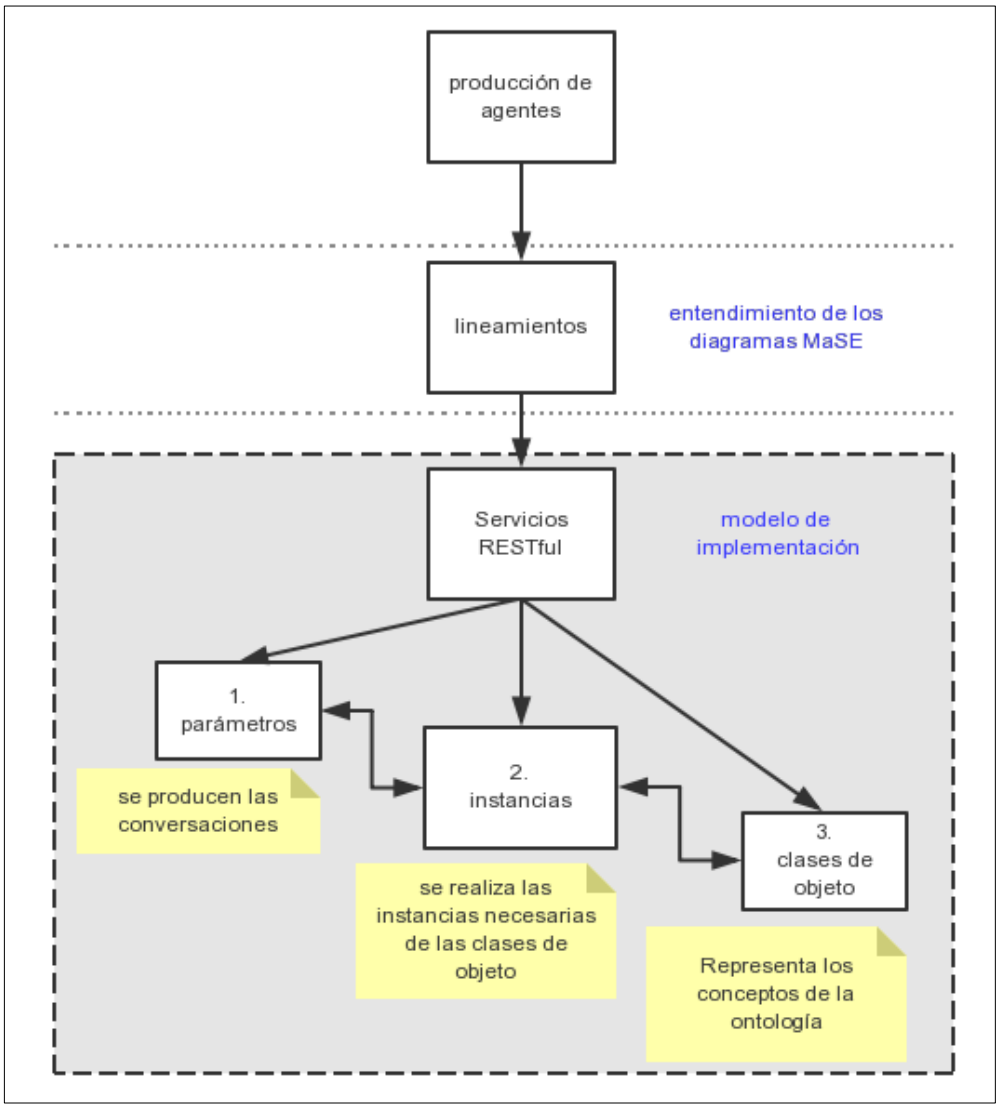


Figura 23: Escenario para la producción de agentes

Fuente: La autora.

Los lineamientos permiten que el usuario identifique que recursos de los diagramas encontrados en MaSE se usa para poder concebir un desarrollo mediante servicios RESTful, bajo el modelo de implementación; parámetros, instancias y clases de objeto.

El modelo de implementación se adecua para el desarrollo de las acciones que ejecuta propiamente cada agente y para las interacciones que realiza con otros agentes.

2.2.1 *Modelo de Implementación.*

Cada elemento de información, conversación o proceso que realizan los agentes es tomado como un recurso, bajo el esquema RESTful. Un recurso tiene asociado un nombre, una URI, y también puede tener un conjunto de parámetros únicos.

La implementación se realiza en tres niveles: 1) parámetros, 2) instancias y 3) métodos.

- Parámetros

Procesa las necesidades individuales y globales de los agentes.

Para procesar necesidades individuales, el agente en este nivel decide cuales son las tareas a realizar y con qué información. La información que resulte fundamental para cumplir con los objetivos individuales representa parámetros.

Cuando se requiere interacción con dos o más agentes para cumplir objetivos globales, las comunicaciones se realizan mediante conversaciones. Para cumplir una conversación se evalúa cuáles son los mensajes que se requiere. La información que resulte fundamental para dar a entender un mensaje en una conversación, representa parámetros.

En este nivel también se define el nombre que se le da a los recursos y el conjunto de parámetros que tiene asociados.

- Instancias

Representa un objeto de una clase de objeto. Se pueden tener n objetos de clases de objetos. Sirven para procesar las peticiones de los agentes.

En este nivel se realiza un control de las instancias de objetos necesarias para cumplir con una petición, evaluando que parámetros el agente recibe del nivel anterior.

- Clases de objeto

Representan los procesos a llevar a cabo para realizar las peticiones y para cumplir con los objetivos de los agentes.

Están compuestas por atributos y métodos. Los atributos son los datos de una clase de objeto, son características que describen a la clase y los métodos son las definiciones del comportamiento de esa clase, todas las cosas que la clase puede hacer.

En este nivel se hace la interacción con la ontología del sistema, ya que las clases de objeto conceptualizan los conceptos de una ontología.

La **figura 24** esquematiza como se produce la ejecución de una tarea en un agente, bajo el modelo; parámetros, instancias y métodos.

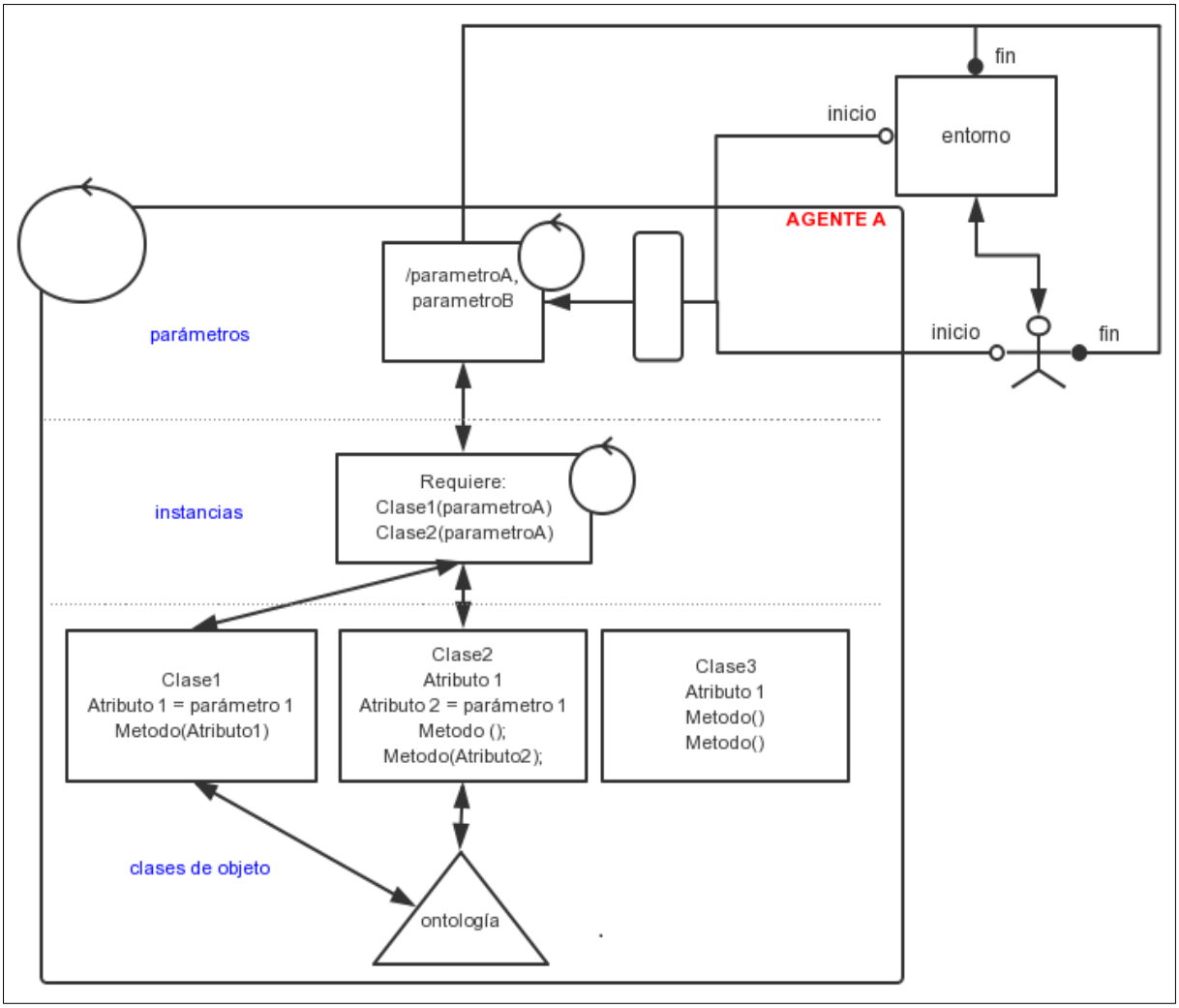


Figura 24: Proceso de un agente; parámetros, instancias, métodos.

Fuente: La autora.

En la **figura 24** se observa que cuando un agente ejecuta procesos de forma individual influyen tres factores: el entorno, el usuario o un control propio que realiza el agente.

El entorno, el usuario o el agente realizan una activación de un proceso. El agente recibe la activación del proceso y realiza un control de los parámetros que requiere para procesar la petición. En el siguiente nivel se instancia la petición y se realiza objetos de las clases necesarias para cumplir la ejecución del proceso. Cuando existen procesos como guardar, consultar o eliminar, las clases de objeto son las encargadas de realizar interacción con la ontología, ya que modelan los conceptos de la ontología como atributos de sus clases. Finalmente el agente ejecuta el proceso y si se trata del usuario o un entorno notifica la finalización.

El proceso que realizó el agente en la **figura 24** representa un recurso. En la **figura 25** se muestra el esquema del recurso.

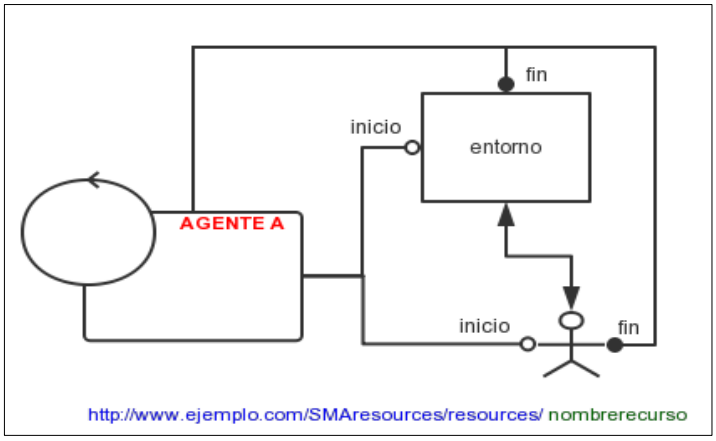


Figura 25: Representación de un recurso sin paso de parámetros

Fuente: La autora.

En la **figura 25** la parte azul representa la URI del recurso y la parte en verde el nombre. El agente mediante la URI activa el proceso correspondiente que debe ejecutar para enviar una respuesta al entorno o al usuario.

La **figura 26** esquematiza como se produce una conversación entre dos agentes, bajo el modelo; parámetros, instancias y métodos.

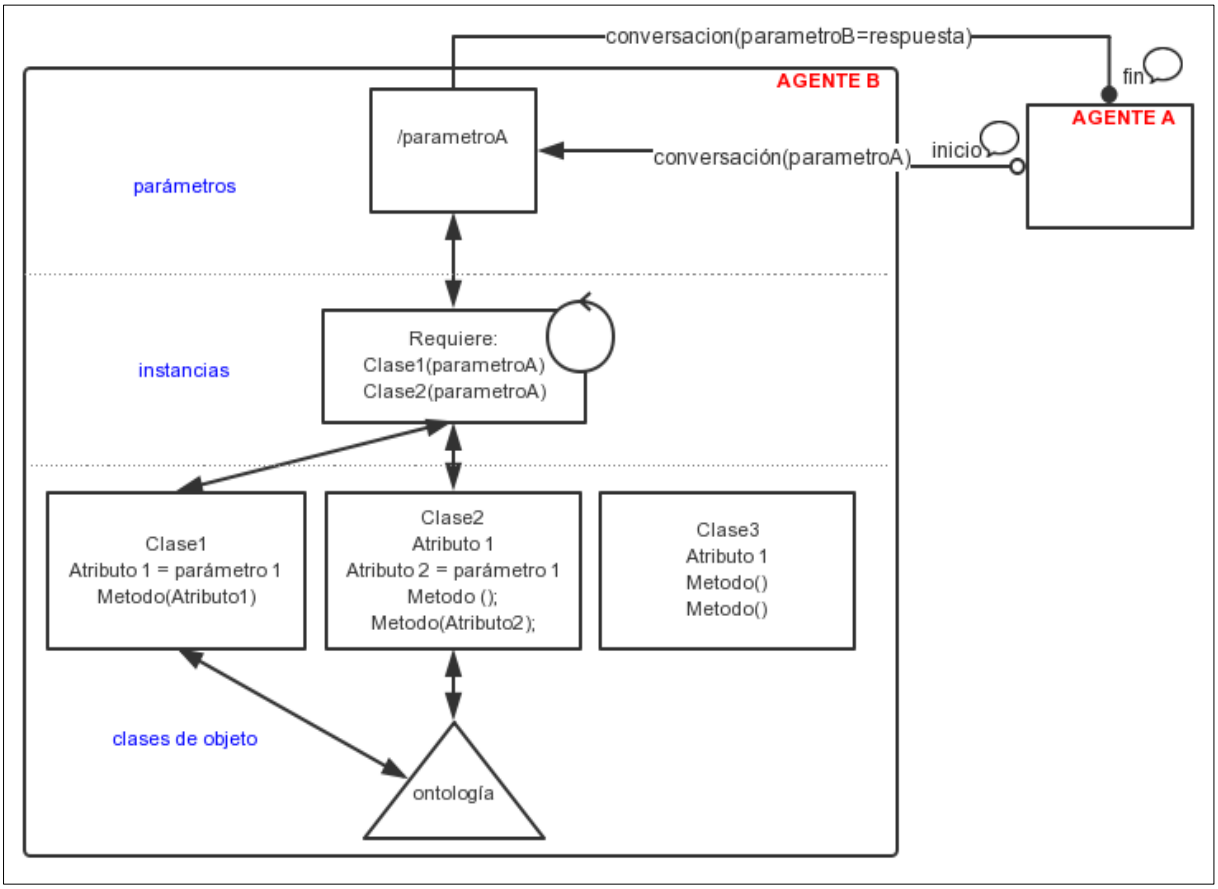


Figura 26: Conversación entre dos agentes; parámetros, instancias, métodos.

Fuente: La autora.

En la **figura 26** el agente A inicia una conversación con el agente B, mediante el parámetro A. El agente B procesa la información, lee el parámetro A enviado, en este caso no realiza controles de los parámetros que necesita, ya que el agente B envía en el mensaje los parámetros para que el agente A instancie la petición. En el siguiente nivel, se realizan las instancias necesarias de las clases de objetos para cumplir con la conversación, si requiere información de la ontología se realiza la interacción con la misma. Y finalmente el agente B realiza los procesos necesarios para dar respuesta al agente A.

Las conversaciones en los agentes representan recursos. En la **figura 27** se observa el esquema del recurso, para que el agente A solicite información al agente B.

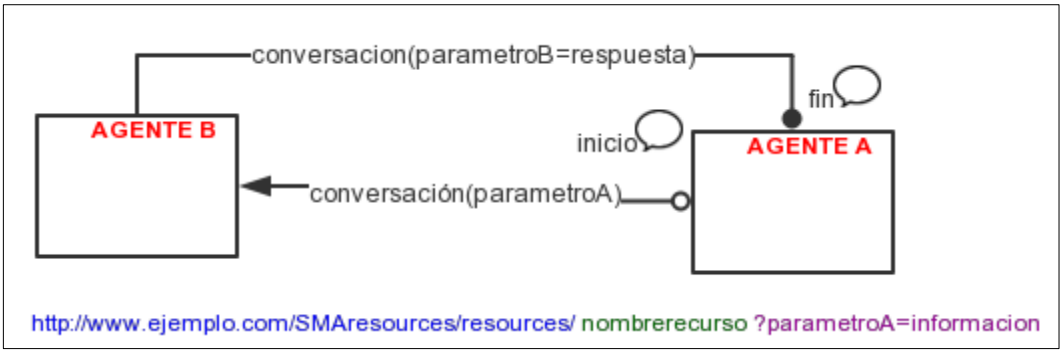


Figura 27: Representación de un recurso con paso de parámetros.

Fuente: La autora.

En la **figura 27** la parte en azul representa el URI del recurso, la parte en verde el nombre del recurso y la parte en morado los parámetros. EL agente A envía la URI y solicita al agente B que ejecute el proceso correspondiente.

2.2.2 *Lineamientos para el entendimiento de los diagramas.*

Para tener un entendimiento de los diagramas, se toma como referencia la fase de diseño en MaSE, porque muestra la interacción e integración de los agentes y ya se ha realizado un proceso de análisis que permite tener un concepto global del sistema.

Aunque los diagramas ya plasman la arquitectura de los agentes y la composición de las clases, no muestran cómo llevar estas clases al modelo en 3 niveles, ni tampoco conciben en su totalidad los métodos necesarios, ya que un método puede dividirse en n sub métodos para cumplir un objetivo.

Para los diagramas obtenidos en la identificación de requerimientos con MaSE, bajo cualquier dominio de aplicación, se debe tener las referencias que se explica a continuación.

Para un mejor entendimiento se toma como referencia los diagramas resultantes del ejemplo modelado con MaSE. El ejemplo se trata acerca de un SMA detector de login erróneo y violación de login.

Fase de diseño

REF 1. Diagrama de arquitectura

- Modela las clases de objeto de los agentes.

Para obtener las clases de objeto es necesario asociar los atributos y métodos a una sola clase. La asociación se la realiza de forma que los atributos y metidos tengan una relación funcional con la clase de objeto. Como ejemplo, los atributos nombre, apellido, edad, serán asociados a la clase de objeto Persona.

Una vez finalizada la asociación de los atributos y métodos tendremos las correspondientes clases de objeto de los agentes. Esta correspondencia se la realiza para cada agente del diagrama de arquitectura. No es necesario modificar el UML original si se tiene experiencia en la identificación de las clases de objeto.

El diagrama de arquitectura encontrado en MaSE generaliza los atributos y métodos necesarios para que el agente cumpla sus objetivos, pero no concibe el modelo de implementación propuesto. Esta generalización se la observa en el ejemplo de la **figura 28**.

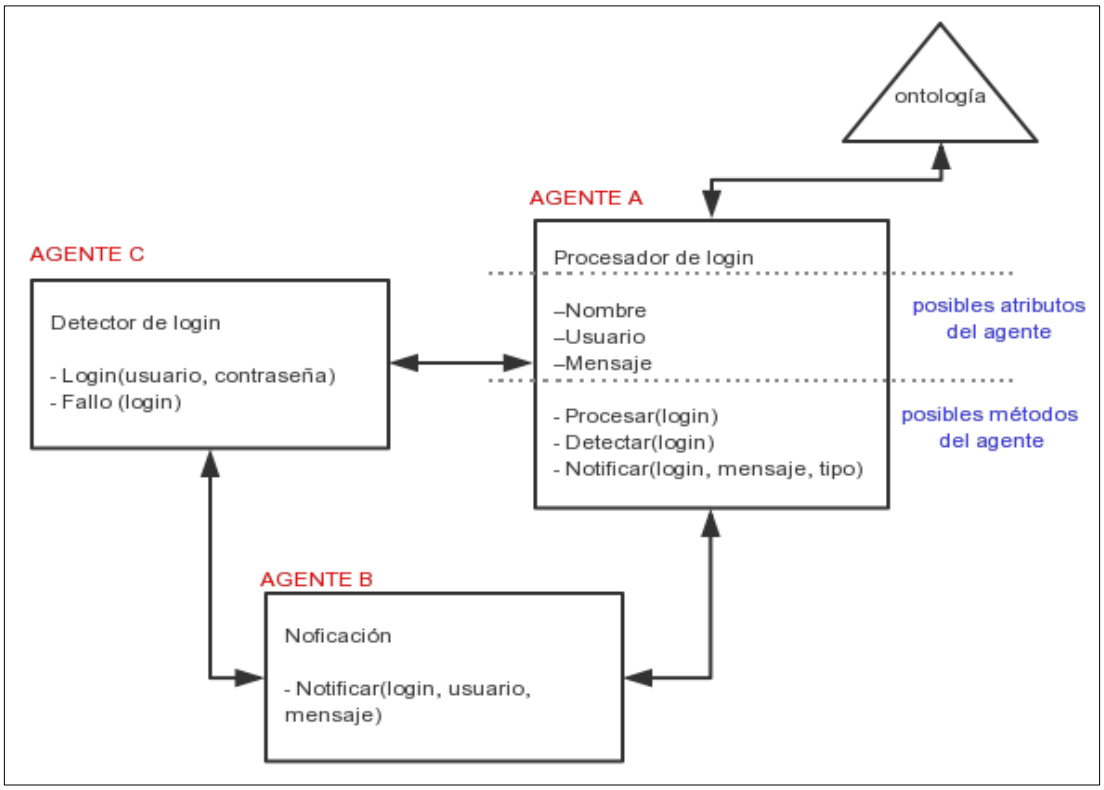


Figura 28: Ejemplo de diagrama de arquitectura – vista general de agentes.

Fuente: La autora.

En la **figura 28** se observa que el agente procesador de login necesita de los atributos; nombre usuario y mensaje, y los métodos; procesar, detectar y notificar, para el cumplimiento de sus objetivos.

Al realizar la correspondencia de las clases, el agente procesador de login ahora trabaja con dos conceptos; Login y Persona. Login tiene asociado los atributos usuario y contraseña y Persona los atributos nombre y mensaje. Ahora se deben diseñar los métodos correspondientes para cumplir con las funciones procesar, detectar y notificar.

Para cumplir con funciones de notificación, Persona tiene asociado un método *NotificarMensaje* y Login *NotiticarLogin*. Para la detección, Login tiene asociado el método *ValidarLogin* y para el procesamiento, Persona tiene asociado el método *ProcesarNombre*.

Después de haber realizado la identificación de los atributos y métodos de los agentes ya se ha conseguido las clases de objeto necesarias para el agente *Procesador de Login*, que representan el tercer nivel del modelo de implementación. En las referencias de los diagramas siguientes se conseguirá obtener los niveles restantes.

Para el ejemplo de la **figura 28** de la vista generalizada de los agentes, después de la identificación de los atributos y métodos de las clases de objeto, se obtiene el ejemplo de la **figura 29**, que muestra las clases de objeto implementadas.

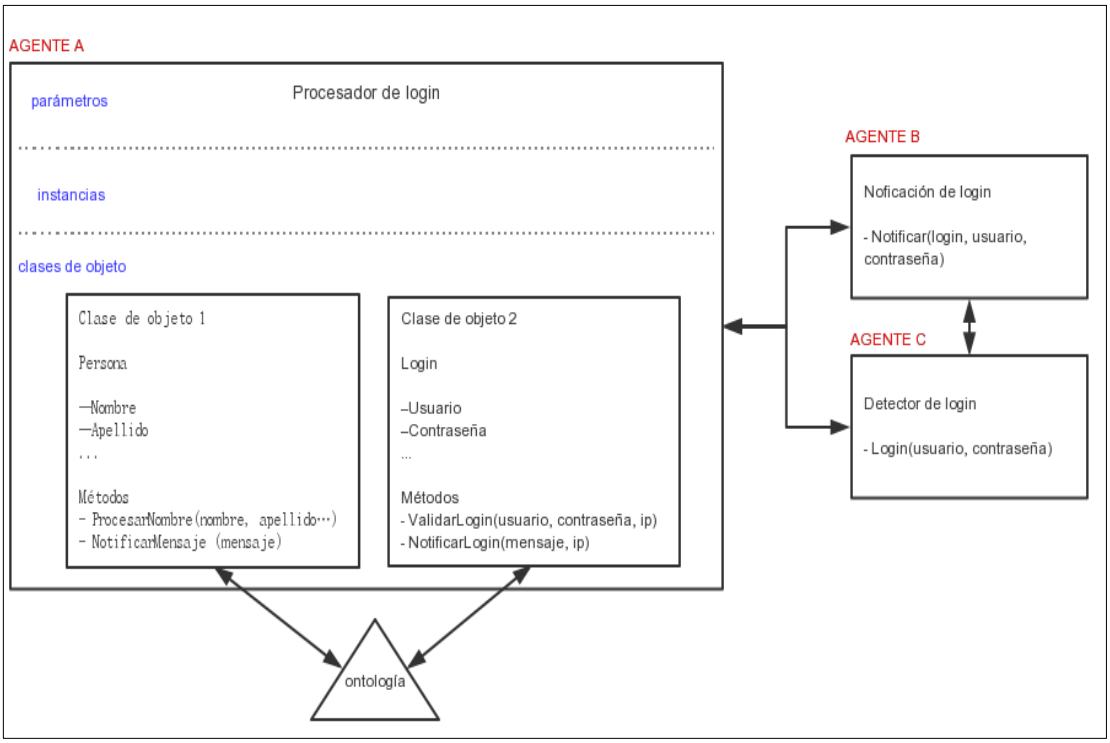


Figura 29: Ejemplo de diagrama de arquitectura – vista en tres niveles de implementación.

Fuente: La autora.

En la **figura 29** el agente procesador de login, ahora cuenta con dos clases de objeto, clase *persona* y *login* para el nivel *clases de objeto*, ya que aún no se cuenta con información de los niveles anteriores.

Estas clases de objeto pueden representar los conceptos de la ontología.

El tercer nivel de implementación es donde se manipula la ontología para acciones de consultar, guardar o eliminar la información que surge de los agentes.

En la fase de análisis en MaSE, ya se obtiene la ontología que representa el dominio de información del sistema. Los atributos de la clase de objeto pueden representar las propiedades de dato de una ontología. Por ejemplo para la clase Persona, la ontología tendrá una clase Persona que contenga las propiedades de dato nombre y apellido. El nivel clases de objeto es el encargado de manipular y ejecutar procesos en la ontología.

REF 2. Diagrama de conversaciones

- Modelan los parámetros y las instancias de un agente

Los diagramas representan las conversaciones que realizan los agentes para cumplir sus objetivos globales.

Para poder obtener los correspondientes parámetros e instancias de objeto se debe identificar las conversaciones asociadas a cada uno de los agentes presentes en el diagrama de conversaciones.

Para el agente comprobador de login, en la **figura 30** se puede observar que se realiza la conversación notificación, con los parámetros información, mensaje y tipologin al notificador.

Información, representa información en relación al usuario que realizo el login.

Mensaje, representa información en relación al login.

TipoLogin, representa información en relación al tipo de login, donde establece si es login correcto o erróneo.

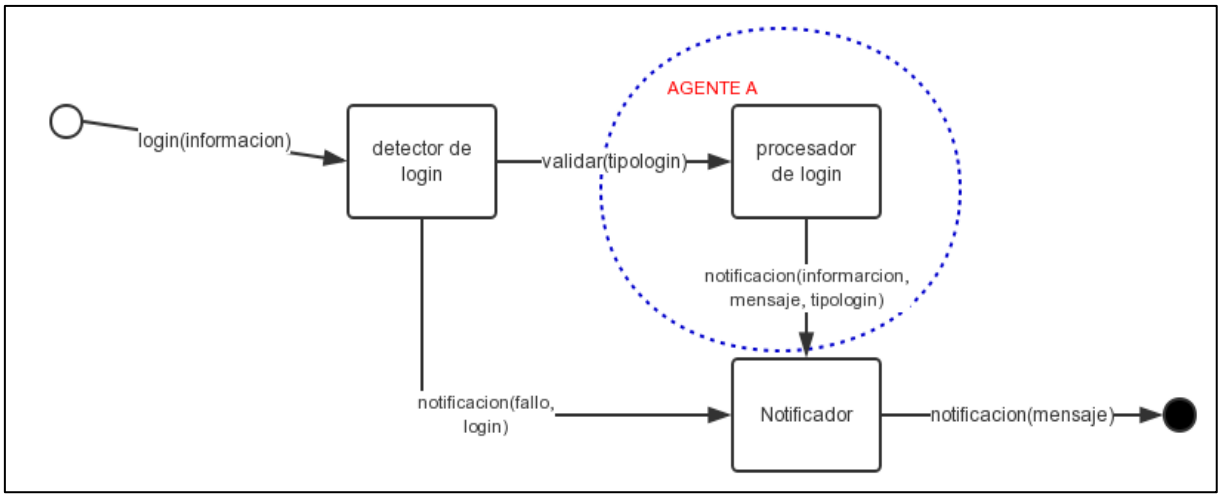


Figura 30: Ejemplo de diagrama de conversaciones.

Fuente: La autora.

En el diagrama de arquitectura de la **figura 30**, ya se identificó las clases de objeto para el agente procesador de login; persona y login. La conversación *notificación* asocia a las dos clases de objetos.

Del ejemplo de la **figura 30**, para llegar a cumplir el del paso de mensaje del dato *información* se requiere la clase de objeto *Persona* con el método *ProcesarNombre*. Para cumplir el paso de mensaje y *tipologin* se requiere la clase de objeto *Login*¸los metodos *NotiticarLogin* y *ValidarLogin* respectivamente. En esta identificación de asociaciones ya se ha conseguido el nivel instancias de objetos.

Para el nivel parámetros la conversación notificación ya muestra los datos que se debe usar. La necesidad del agente notificador es obtener datos de información, mensaje y el tipo de login, cada información requerida debe tener asociado una instancia de objeto de las clases del agente procesador de login, que de cómo resultante la información requerida por el agente notificador.

En la conversación *notificación*, para cumplir con el mensaje *información*, es necesario obtener una instancia de la clase *Persona* que involucre el método *NotificarMensaje*, y que este método tenga un resultante que corresponde a la información del usuario. Para el mensaje, es necesario obtener una instancia de la clase *Login* con el método *NotificarLogin* y que dé como resultado información en relación al login. Para el mensaje *tipologin* se usa la misma instancia de la clase *Login* pero incluyendo el método *ValidarLogin*, esta segunda instancia debe contener información en la relación al tipo de login encontrado, si fue un login erróneo o violación de login.

El la **figura 31** podemos observar el resultante de la identificación de parámetros e instancias para el agente procesador de login.

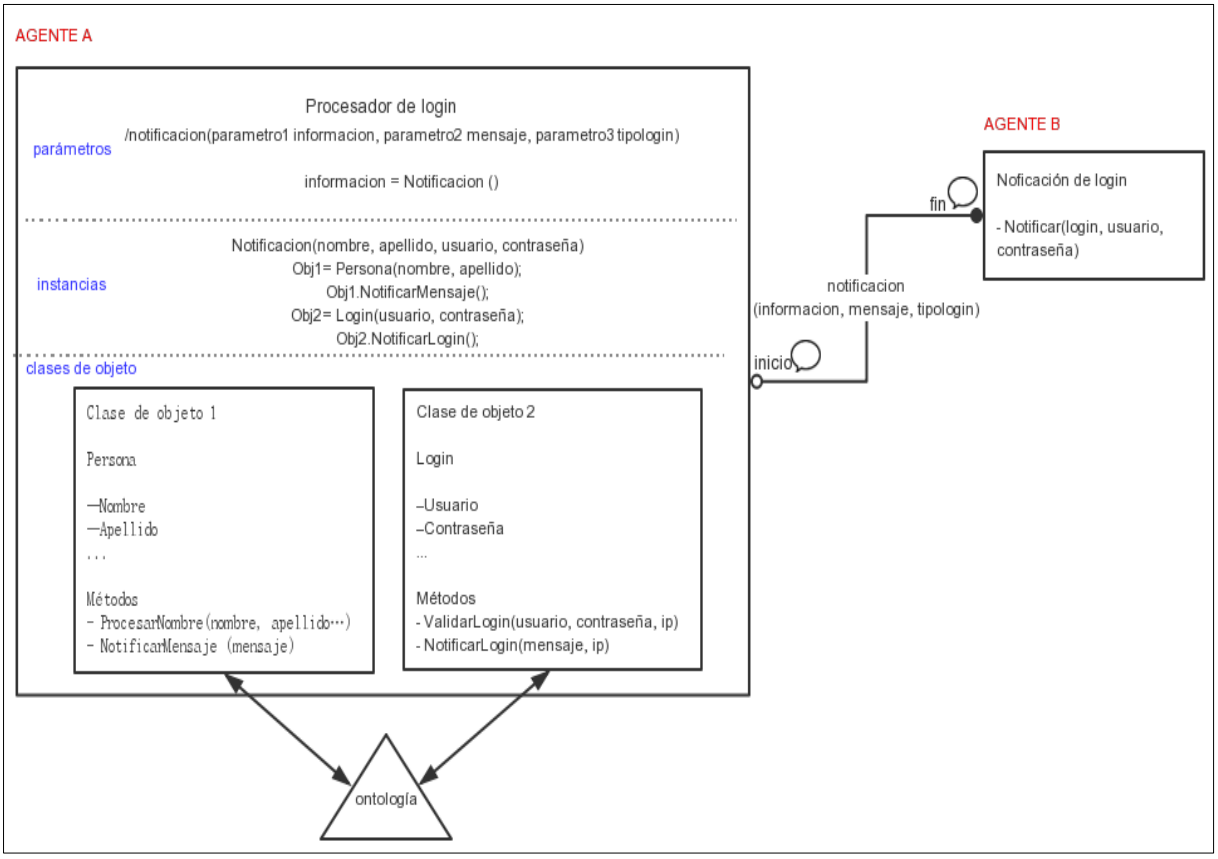


Figura 31: Ejemplo de diagrama de conversaciones; parámetros, instancias y métodos.

Fuente: La autora.

La **figura 31** es el proceso completo de identificación de los 3 niveles de programación, haciendo el entendimiento de los diagramas obtenidos de la fase de diseño en MaSE.

Cada conversación o acción que realiza un agente representa un recurso, cuando ya se obtenido para cada tarea los parámetros, instancias y clases de objeto necesarias, ya se ha obtenido un recurso.

Para el ejemplo de la **figura 31**, el recurso correspondiente se representa en la **figura 32.**

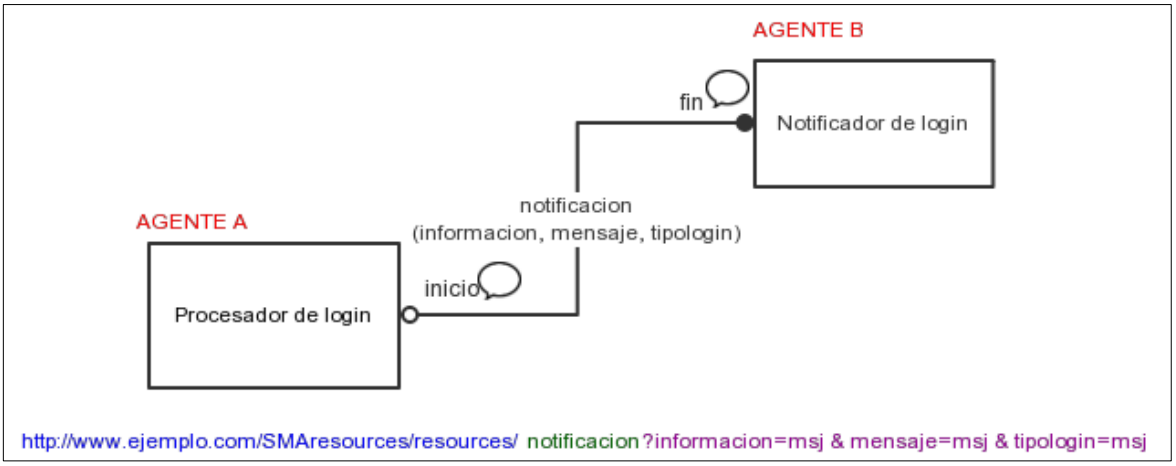


Figura 32: Ejemplo de diagrama de conversaciones; identificación de un recurso.

Fuente: La autora.

En la **figura 32** la parte azul representa el URI del recurso, la parte verde el nombre del recurso en este caso el nombre asignado *notificación* es el mismo que se ejecuta en la conversación, los parámetros son la parte en morado y son los mismos correspondientes a la conversación, *información*, *mensaje* y *tipologin*. El agente B mediante la llamada del recurso solicita la información al agente A.

2.2.3 Esquema de *implementación.*

En este punto ya se comprende cómo identificar el modelo para implementar los agentes, ahora se da los lineamientos para producirlos en el lenguaje Java.

- **Parámetros**

La **figura 33** muestra el esquema de una clase básica para conseguir el nivel parámetros.

```
package <paquete>;
  //librerias necesarias en la implementación
  import java.io.IOException;
  import javax.ejb.EJB;
  import javax.ejb.Stateless;
  import javax.ws.rs.GET;
  import javax.ws.rs.Path;
  import javax.ws.rs.Produces;
  import javax.ws.rs.QueryParam;
  //
  @author @<autor>
  @Stateless
  @Path("/MiRecurso")
  public class AlgunRecurso {
    @EJB
    private instancianivel2 nombreinstancia;
    @GET
    @Produces("text/plain")
    public String NombreClase(
              @QueryParam("Parametro1") String MiParametro1,
             @QueryParam("Parametro2") Int MiParametro2
    ) {
  return nombreinstancia.informacion(MiParametro1, MiParametro2);
  }
```

Figura 33: Clase básica para nivel parámetros.

Fuente: La autora.

En la **figura 33**:

- **@Stateless** es un bean de sesión sin estado para objetos distribuidos que permite la concurrencia.
- **@Path** define el nombre del recurso.

Cuando los agentes realizan conversaciones, @**Path** será definido por el nombre de la conversación.

Cuando el agente necesite cumplir sus tareas propias, @**Path** será definido por el nombre de la tarea.

- @**GET**, @**PUT**, @**POST** y @**DELETE** para especificar las peticiones del recurso.
    - @GET para recuperar información de un recurso
    - @PUT para actualizar información de un recurso
    - @PUT para crear información de un recurso
    - @DELETE para eliminar información de un recurso
- @**QueryParam** son los parámetros que participan en las conversaciones.
- @**Produces** representa la salida de la información, esta puede ser:
    - text/plain, para salidas de información en texto plano
    - application/json, para salidas de información en formato JSON
    - text/html, para salidas de información en formato HTML
    - application/xml, para salidas de información en formato XML

En la **figura 34**, se observa los import necesarios para poder manipular las peticiones de los recursos bajo la arquitectura REST.

```
@GET          import javax.ws.rs.GET;
@Produces     import javax.ws.rs.Produces;
@Path         import javax.ws.rs.Path;
@PathParam    import javax.ws.rs.PathParam;
@QueryParam   import javax.ws.rs.QueryParam;
@POST         import javax.ws.rs.POST;
@Consumes     import javax.ws.rs.Consumes;
@FormParam    import javax.ws.rs.FormParam;
@PUT          import javax.ws.rs.PUT;
@DELETE       import javax.ws.rs.DELETE;
```

Figura 34: Import requeridos para clases REST

Fuente: La autora.

- **Instancias**

La **figura 35** muestra el esquema de una clase básica para conseguir el nivel instancias.

```
package <paquete>;
//librerias necesarias en la implementación
import java.io.IOException;
import javax.ejb.Singleton;
//
@author @<autor>
@Singleton
public class NombreClase {
//inicialización de variables de la clase
Private String msj;
 public String ConstructorClase(MiParametro1, MiParametro2) {
    //Instancias de clases de objeto necesarias
Clase objeto = new Clase();
msj = objeto.Metodo(MiParametro1, MiParametro2);
 return msj;
  }
}
```

Figura 35: Clase básica para nivel instancias.

Fuente: La autora.

En la **figura 35**:

- **@Singleton** es un bean de sesión diseñado para el acceso simultáneo a las instancias de las clases de manera concurrente.

- **Clases de objeto**

La **figura 36** muestra el esquema de una clase básica para conseguir el nivel instancias.

```
package services;
import <librerías>
 @author @<autor>
public class NombreClase {
//variablesglobales
public static String MiVariable1;
public static String MiVariable2;
 public static String MiClase (
String MiVariable1)   {
    try {
    } catch (Exception e) {
    }
    return  msj;
  }

public static String MiClase2 (
String MiVariable2)   {
    try {
    } catch (Exception e) {
    }
    return  msj;
  }
}
```

Figura 36: Clase básica para nivel clases de objeto.

Fuente: La autora.

En la **figura 36**:

- **import** permita usar paquetes para poder realizar manipulación de los objetos

  En este nivel se realiza la interacción con la ontología, para esto se usará el api Jena con sus librerías necesarias que se observan en la **figura 37.**

```
import com.hp.hpl.jena.ontology.DatatypeProperty;
import com.hp.hpl.jena.ontology.Individual;
import com.hp.hpl.jena.ontology.ObjectProperty;
import com.hp.hpl.jena.ontology.OntClass;
import com.hp.hpl.jena.ontology.OntModel;
import com.hp.hpl.jena.ontology.OntModelSpec;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.util.iterator.ExtendedIterator;
```

Figura 37: Librerías necesarias para poder conseguir la interacción con la ontología.

Fuente: La autora.

- Las variables globales son manipuladas por todos los métodos de la clase.

  Para poder interactuar con la ontología se debe realizar la declaración del modelo de la ontología como variable global de la clase, como se observa en la **figura 38**.

```
public static OntModel model = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);
```

Figura 38: Formato declaración del modelo de la ontología.

Fuente: La autora.

- Se realiza los métodos para cumplir los objetivos.

  En los métodos se hace la lectura y la manipulación de la ontología del sistema como se observa a continuación en la **figura 39**.

```
model.read("Ruta de la ontología", "RDF/XML");
model.write("Ruta de la ontología", "RDF/XML");
```

Figura 39: Formato de lectura y escritura de la ontología

Fuente: La autora.

En la **figura 39** el formato de lectura, puede ser "RDF/XML", "N-TRIPLE", "TURTLE" o "TTL" y "N3".

Luego de haber identificado los 3 niveles del modelo de implementación, ya se ha conseguido la representación de un recurso.

Suponiendo que el URL que maneja el MAS es www.ejemplo.com, tomando como ejemplo la clase obtenida del nivel de parámetros de la **figura 35,** luego de haber identificado y

obtenido las 3 clases del modelo de implementación, se obtiene el recurso resultante de la **figura 40**.

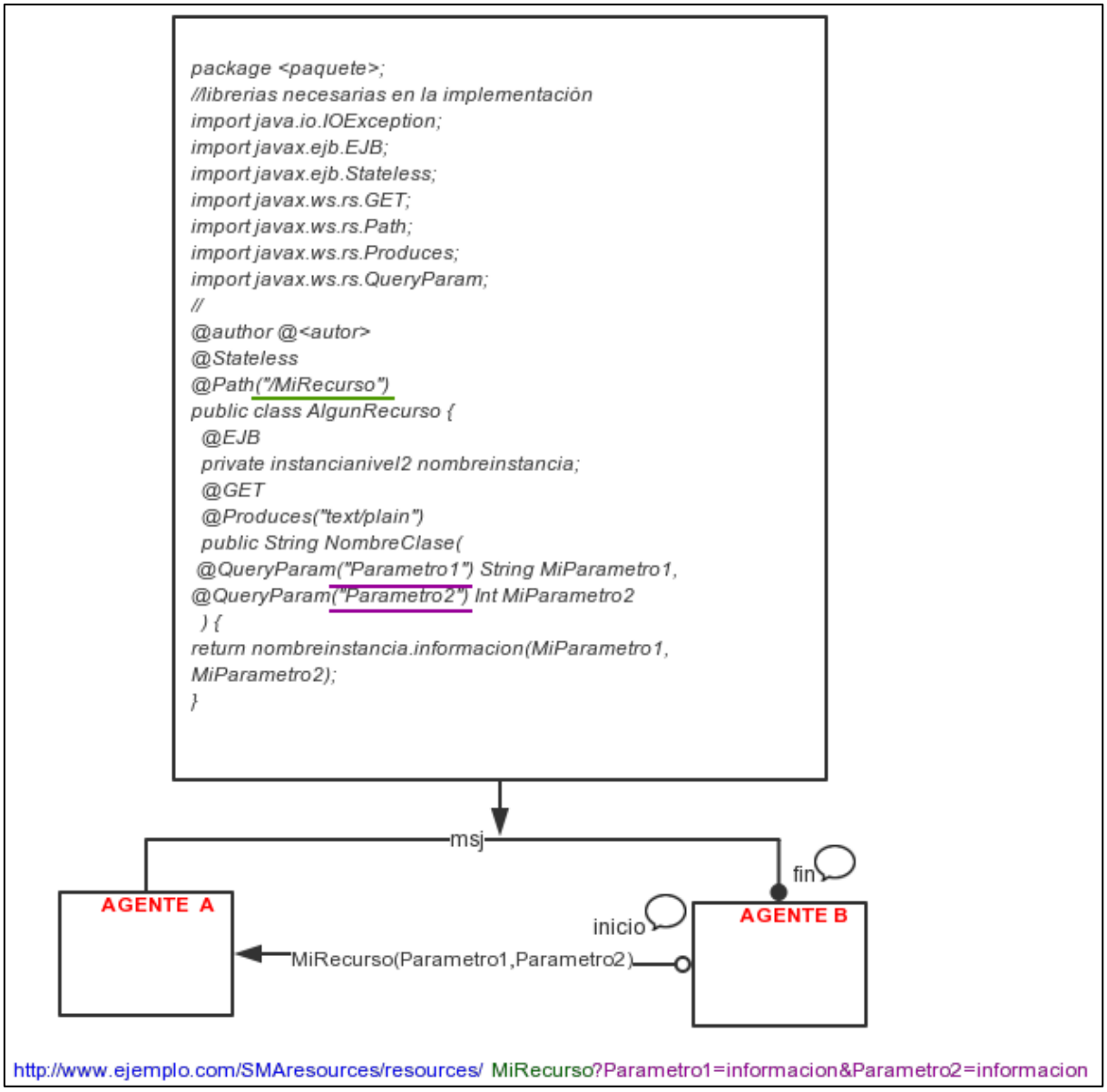


Figura 40: Formato de lectura y escritura de la ontología

Fuente: La autora.

En la **figura 40**, la parte azul corresponde al URI del recurso, la parte verde corresponde al nombre del recurso que es el mismo que está definido en la variable @Path, la parte en morado es los parámetros del recurso que corresponden a la definición del @QueryParam en la clase.

El proceso que realizan los agentes es el siguiente, el agente B inicia una conversación con el agente A, mediante la llamada a la URL del recurso, el agente A lee los parámetros que ha enviado el agente B, para realizar la ejecución de los procesos necesarios para cumplir con los objetivos.

Finalmente el agente A envía la respuesta al agente B.

El proceso de identificación de las clases del modelo de implementación y de los recursos que comprenden estas clases debe realizarse por cada agente y función presente en el SMA.

Las acciones que realizan los agentes para cumplir con sus objetivos individuales representan recursos y también las conversaciones que realizan con otros agentes para cumplir con sus objetivos globales.

CAPÍTULO 3

APLICACIÓN WEB MULTI AGENTE BASADA EN UNA ONTOLOGÍA

3. Descripción de la aplicación

El propósito es desarrollar una aplicación web multi agente basada en una ontología que modele los procesos del framework y que permita:

- Recopilar información de múltiples fuentes de información
- Compartir datos de manera eficiente
- Gestionar e integrar los resultados obtenidos para presentarlos al usuario

La aplicación recopila de manera automática información en referencia a perfiles personales desde recursos en la web, como; Facebook, LinkedIn y Google y desde un recurso físico que es una ontología existente; TestEstres_M.

La **figura 41** muestra una vista general de la información que se recopila desde los recursos web y el recurso físico.

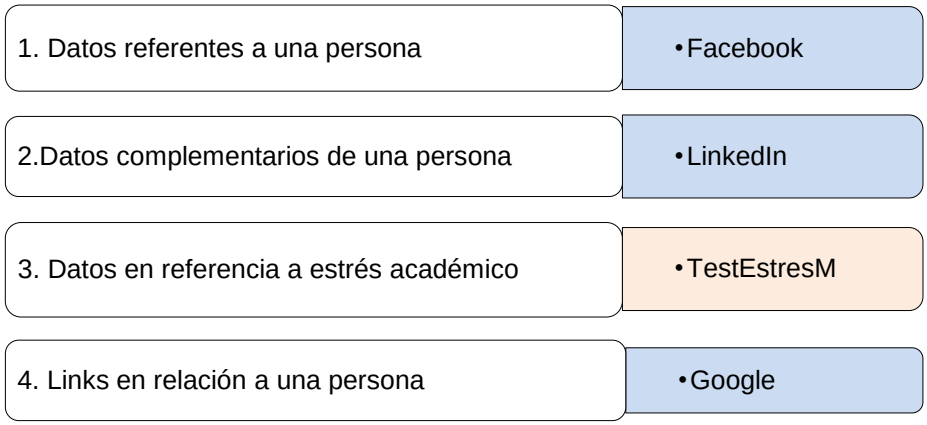


Figura 41: Datos que se recopila desde los recursos web y físico

Fuente: La autora.

En la **figura 41** se observa que la información a recopilar de Facebook corresponde a datos referentes a una persona, LinkedIn datos complementarios de una persona, Google links en relación a una persona y de TestEstresM datos en referencia a estrés académico de una persona.

La información se almacena en una ontología. La ontología es una combinación de Test_EstresM y Foaf, de forma que se tenga una representación de conocimiento más completa de datos en referencia a perfiles personales. La ontología tiene el nombre de **People.**

El objetivo de esta recopilación de información es proveer de datos, validos, fiables y por tanto de utilidad científica que puedan ser utilizados como recursos para realizar

investigaciones, que determinen el entorno en los cuales las personas se desenvuelven y que puedan ser causa o consecuencia de las acciones que realiza un usuario.

La **figura 42** muestra el listado de agentes a construir a partir de las necesidades de recopilación.

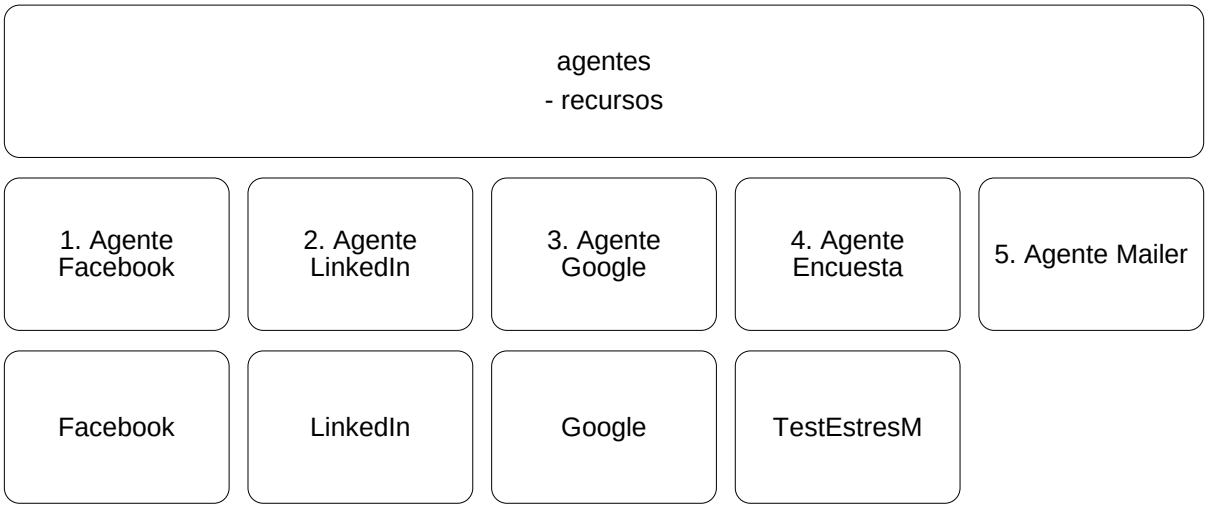


Figura 42: Agentes y utilización de recursos

Fuente: La autora.

Se realiza un agente por cada recurso identificado, ya que los métodos de recopilación, identificación y asociación de información con los conceptos de la ontología son distintos para cada agente, en relación con sus recursos asociados. Cada recurso web tiene un mecanismo y formato de salida distinto para permitir la recopilación.

En la **figura 43** se observa la composición final del sistema multi agente.

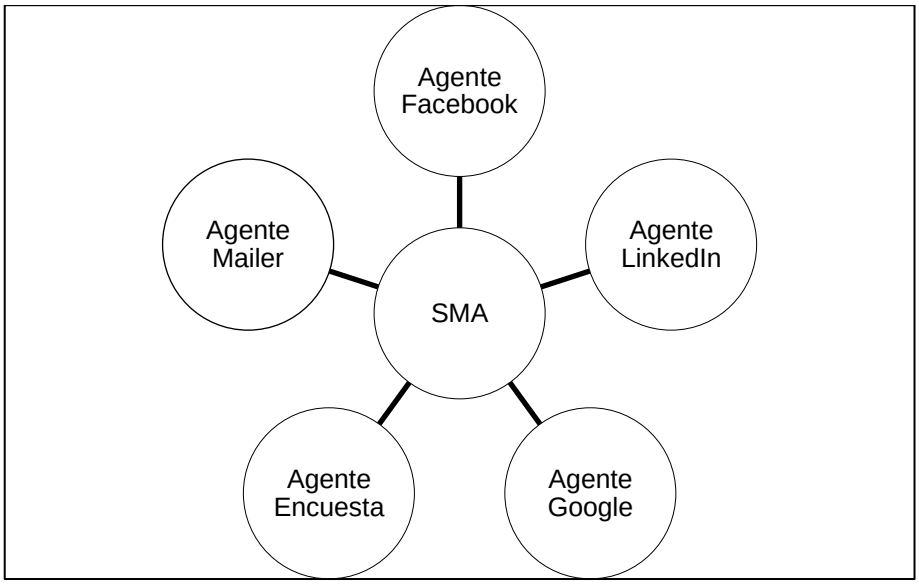


Figura 43: Composición aplicación de SMA

Fuente: La autora.

Las funciones principales de los agentes se las observa en la **figura 44.**

Agente	•Proceso general
1. Agente Mailer	•Notifica al usuario para que use la aplicación
2. Agente Facebook	•Recopila información general de una persona
3. Agente LinkedIn	•Recopila información complementaria de una persona
4. Agente Google	•Recopila informacion en relación a una persona
5. Agente Encuesta	•Recopila información en referencia al estrés

Figura 44: Proceso de recopilación de información SMA.

Fuente: La autora.

3.1 Fase de análisis

Se describe la fase de análisis y los diagramas resultantes del modelado con Mase, para conseguir la aplicación de sistema multi agente basada en una ontología que recopile información automáticamente desde recursos web como Facebook, LinkedIn y Google y un recurso físico Test_EstresM.

3.1.1 *Capturar los objetivos.*

Agente Facebook

Recopila información referente a datos de la persona desde la red social Facebook y realiza controles de las sesiones que se ejercen en los agentes para ayudar a la asociación de la información.

Los datos que se recopilan son los siguientes:

- Nombre completo
- Primer nombre
- Primer apellido
- Email
- Cumpleaños
- Género

- Estado
- Nickname
- Trabajo

Objetivo General:

Recopilar información referente a los datos generales de una persona desde la red social Facebook.

Objetivos Específicos:

1. Iniciar una sesión en Facebook por medio del agente.
2. Recopilar información de Facebook.
3. Guardar los datos en la ontología.
4. Notificar al usuario que los datos son almacenados.
5. Crear una sesión de usuario con los datos recopilados de Facebook.

Agente LinkedIn

Es el encargado de recopilar información referente a datos complementarios de la persona desde la red social LinkedIn.

Los datos que se recolectan son los siguientes:

- Industria a la que pertenece
- Trabajo actual
- Lugar de residencia
- Código de país de residencia
- Intereses
- Nombres de las personas a las cuales se encuentra conectado y conoce

Además debe realizar un monitoreo para identificar que el usuario aún se encuentra vinculado a la sesión que se crea desde el agente Facebook.

Objetivo General:

Recopilar información referente a los datos complementarios de una persona desde la red social LinkedIn.

Objetivos Específicos:

1. Iniciar una sesión en LinkedIn por medio del agente.

2. Recopilar información de LinkedIn.
3. Asociar la información proveniente de LinkedIn con la información que se recopila desde Facebook y guardar los datos en la ontología.
4. Notificar al usuario que los datos son almacenados.
5. Monitorear la sesión de la aplicación con los datos de sesión de usuario que realiza el agente Facebook.

Agente Google

Es el encargado de recopilar información referente a links que relacionen a una persona, por ejemplo, imágenes, publicaciones, artículos o links de perfiles sociales de una persona.

Los datos que se recopilan son los siguientes:

- Datos de la web: como documentos, perfiles sociales o links.
- Datos de noticias: links en donde se habla acerca de la persona.
- Imágenes

Además debe realizar un monitoreo para identificar que el usuario aún se encuentra vinculado a la sesión que se crea desde el agente Facebook.

Objetivo General:

Recopilar información de interés acerca de una persona desde el motor de búsqueda Google.

Objetivos Específicos:

1. Recopilar información de Google.
2. Asociar la información proveniente de Google con la información que se recopila desde Facebook y guardar los datos en la ontología.
3. Notificar al usuario que los datos son almacenados.
4. Monitorear la sesión de la aplicación con los datos de sesión de usuario que realiza el agente Facebook.

Agente Encuesta

Es el encargado de realizar búsquedas de la información del usuario en la ontología Test_EstresM para determinar si ya ha realizado la encuesta de estrés o sino aplicarle la misma.

Objetivo General:

Buscar si el usuario ya ha realizado la encuentra SISCO, si no existe su información aplica la encuesta.

Objetivos Específicos:

1. Buscar en Test_EstresM si el usuario ha realizado la encuesta.
2. Aplicar la encuesta SISCO al usuario.
3. Recopilar información de la encuesta.
4. Asociar la información proveniente de la Encuesta con la información que se recopila desde Facebook y guardar los datos en una ontología.
5. Notificar al usuario que los datos son almacenados.
6. Monitorear la sesión de la aplicación con los datos de sesión de usuario que realiza el agente Facebook.

Agente Mailer

Es el encargado de notificar al usuario y administrador las novedades de los agentes en cuanto a gestión de información y sesiones. Realiza las notificaciones de la información que es recopilada de los recursos y monitorea con periodicidad la inclusión de nuevos usuarios en la aplicación, realizando comparaciones para no realizar notificaciones dobles. Además cuando ha finalizado el proceso de recopilación informa al usuario de los datos obtenidos.

Permite al administrador obtener una interfaz de consultas para la búsqueda de la información de los usuarios que han participado del SMA.

La notificación al usuario se realiza vía mail, se ingresa en un archivo de texto las personas que se desea obtener la información, de esta forma el agente Mailer monitorea en un intervalo de tiempo de 30 minutos si ha existido la incorporación de nuevos usuarios y realiza las respectivas notificaciones. El mail que se envía al usuario presenta el siguiente mensaje:

> *Los datos a recuperar sirven para obtener una descripción de tu perfil personal, sitios web sociales, links de interés que te relacionen y conexiones con otras personas.*
> *Además aplicaremos un test de estrés para conocer las causas y consecuencias de tus principales preocupaciones.*
> *Los datos que van a ser recopilados en la aplicación, sólo tienen fines académicos.*
> *Acepta recuperar tu información haciendo click en el siguiente enlace: http://localhost:8080/DataMAS/paso1.html*

Figura 45: Formato mail agente Encuesta.

Fuente: La autora.

Además ofrece un buscador de personas, en donde se puede consultar la información de los usuarios que han usado el SMA.

Objetivo General:

Notificar al usuario y administrador las novedades de los agentes en cuanto a gestión de información y sesiones.

Objetivos Específicos:

1. Notificar al administrador el estado de las sesiones.
2. Notificar al usuario para que use la aplicación.
3. Notificar al usuario los datos recopilados.
4. Permitir que el usuario y el administrador realicen una búsqueda de los datos recopilados.

Entregables: diagramas de objetivos

La **figura 46** presenta el diagrama de objetivos del agente Facebook.

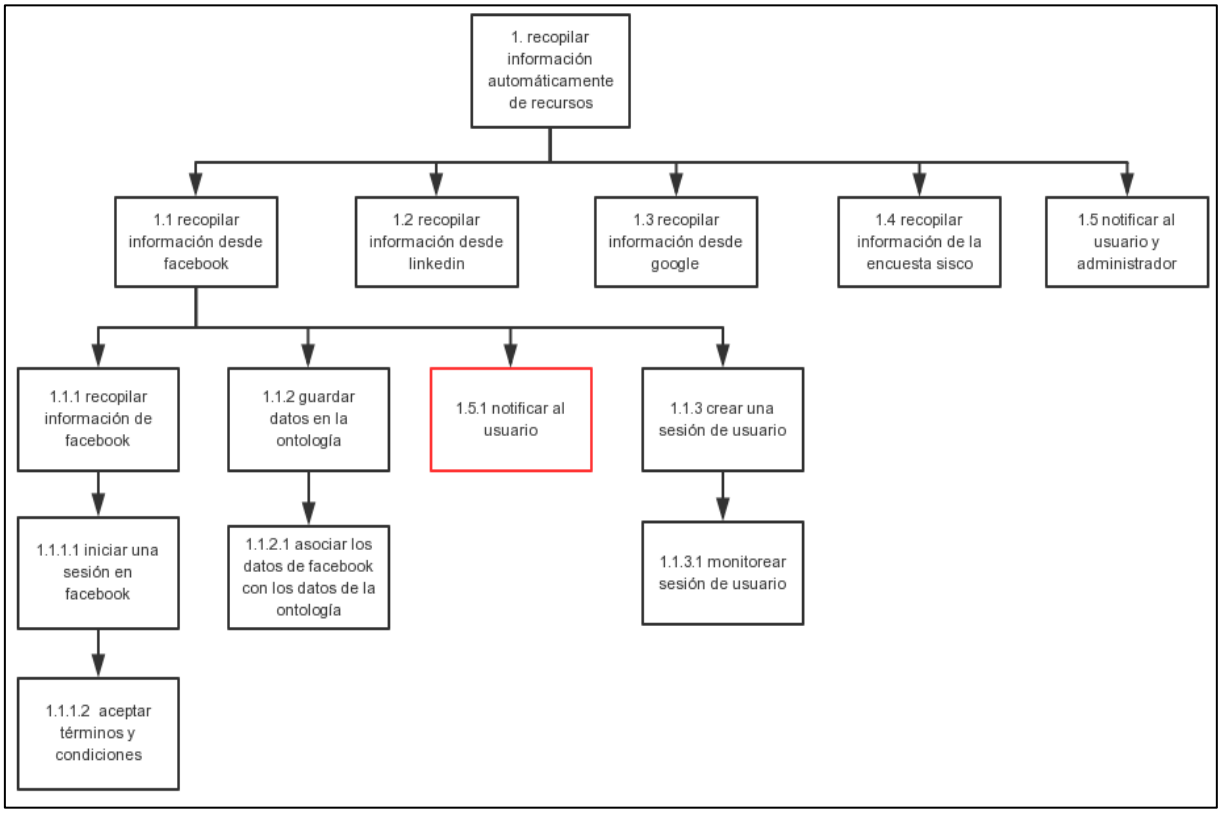


Figura 46: Diagrama de objetivos agente Facebook.

Fuente: La autora.

La **figura 47** presenta el diagrama de objetivos del agente LinkedIn.

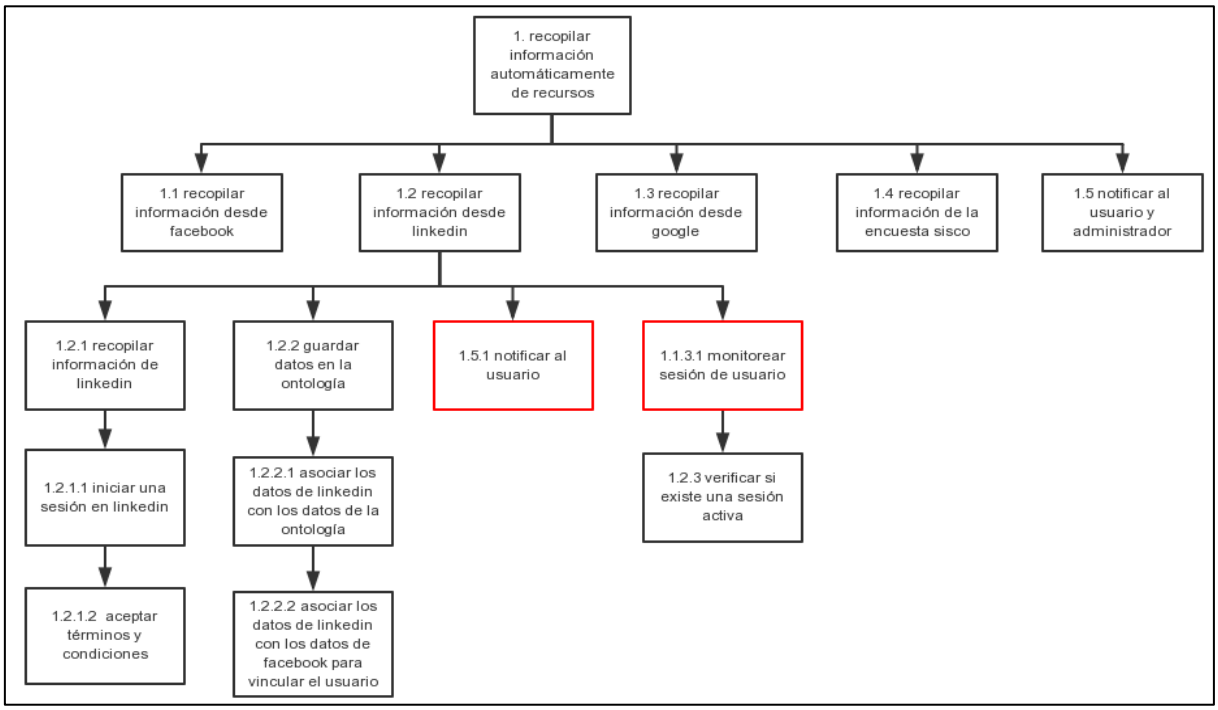


Figura 47: Diagrama de objetivos agente LinkedIn.

Fuente: La autora.

La **figura 48** presenta el diagrama de objetivos del agente Google.

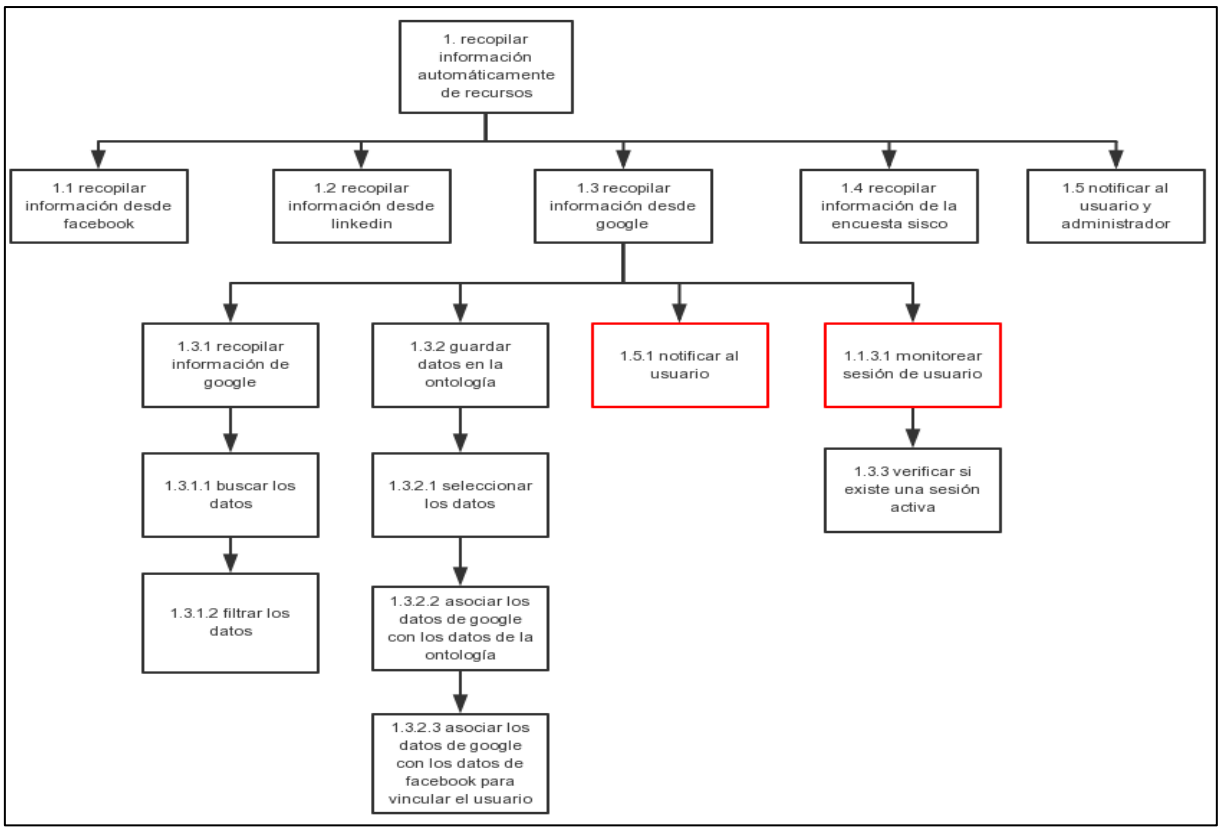


Figura 48: Diagrama de objetivos agente Google.

Fuente: La autora.

La **figura 49** presenta el diagrama de objetivos del agente Encuesta.

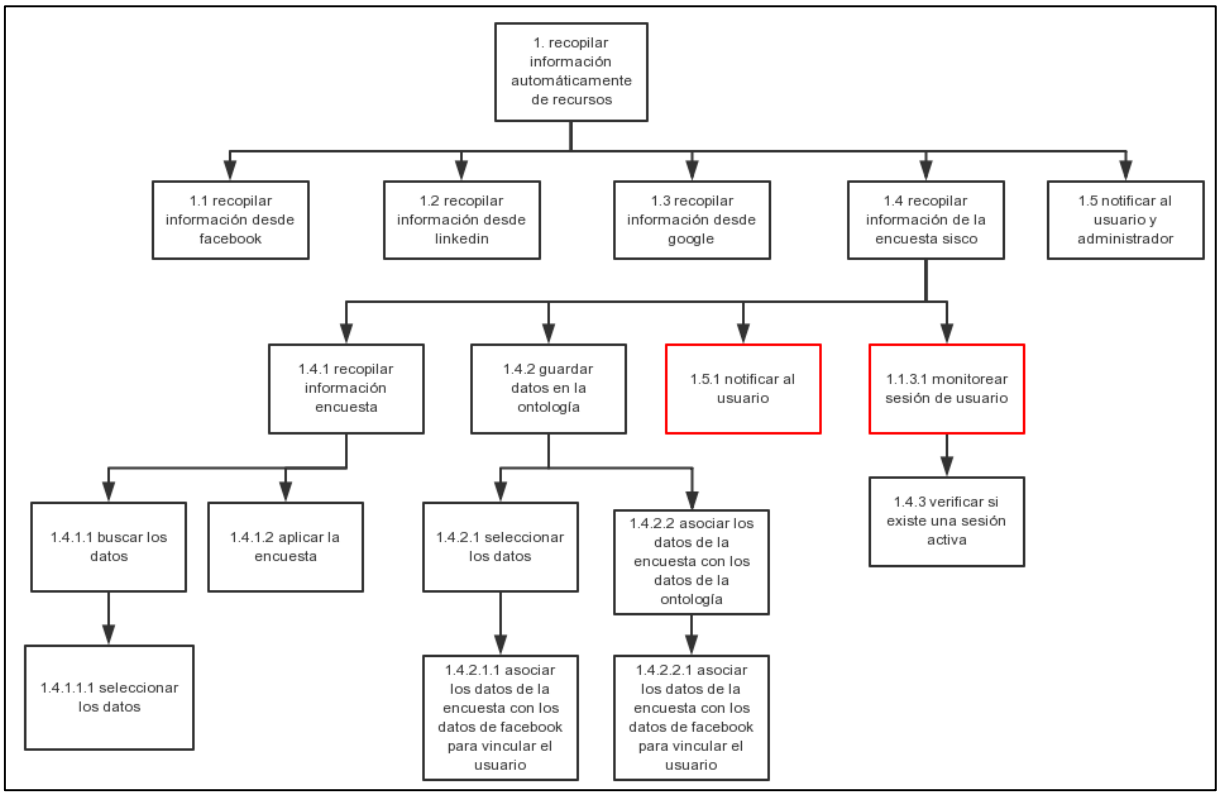


Figura 49: Diagrama de objetivos agente Encuesta.

Fuente: La autora.

La **figura 50** presenta el diagrama de objetivos del agente Mailer.

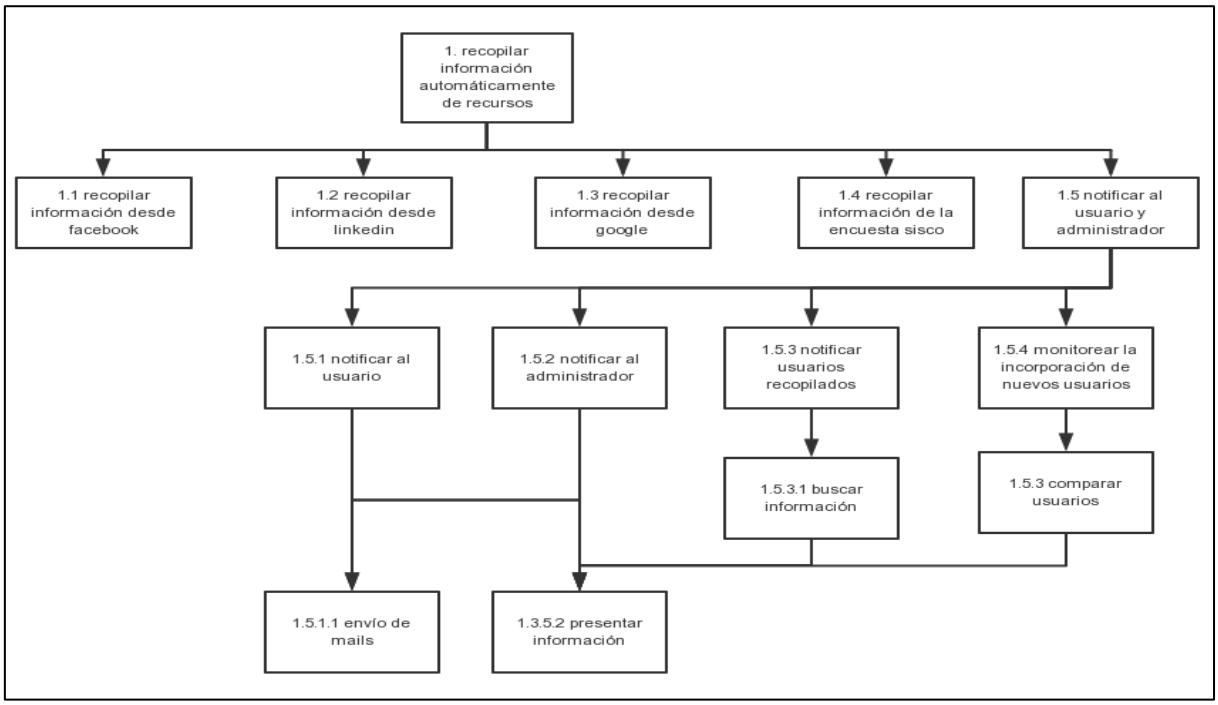


Figura 50: Diagrama de objetivos agente Mailer.

Fuente: La autora.

3.1.2 *Capturar los casos de uso.*

Agente Facebook

Caso de uso 1: Iniciar una sesión en Facebook por medio del agente Facebook.

Objetivos asociados:

- 1.1.1.1
- 1.1.1.2
- 1.1.3
- 1.5.1

Proceso:

- Si el usuario tiene una sesión de Facebook activa, el agente Facebook muestra un mensaje para el usuario acepte los términos y condiciones de uso.
- Si el usuario no tiene una sesión activa de Facebook, realiza el login a Facebook ingresando usuario y contraseña desde la interfaz del agente Facebook y seguidamente se muestra un mensaje para que acepte los términos y condiciones de uso.
- Si el usuario tiene una sesión activa en Facebook y previamente ya ha aceptado los términos y condiciones de uso, inicia sesión.
- Si el usuario tiene una sesión activa en Facebook y acepta los términos y condiciones de uso, inicia sesión.
- Si el usuario no acepta los términos y condiciones de uso no inicia sesión.

Caso de uso 2: Recopilar información de Facebook y almacenarla en la ontología.

Objetivos asociados:

- 1.1.1
- 1.1.1.1
- 1.1.1.2
- 1.1.2
- 1.1.2.1
- 1.5.1
- 1.1.3.1

Proceso:

- Si el usuario tiene una sesión activa en Facebook y previamente ya ha aceptado los términos y condiciones de uso, se recopilan sus datos y se almacenan en la ontología.
- Si el usuario tiene una sesión activa en Facebook y acepta los términos y condiciones de uso, se recopilan sus datos y se almacenan en la ontología.
- Si el usuario no acepta los términos y condiciones de uso no se recopilan sus datos y no se almacenan en la ontología.

Caso de uso 3: Crear una sesión de usuario con los datos de Facebook.

Objetivos asociados:

- 1.1.3
- 1.1.3.1
- 1.1.2

Proceso:

- Si los datos fueron almacenados en la ontología, el agente Facebook crea una sesión de usuario.

Caso de uso 4: Monitorear que los otros agentes mantengan activa las sesiones de usuario.

Objetivos asociados:

- 1.1.3
- 1.1.3.1
- 1.1.2

Proceso:

- Si los otros agentes no mantienen activa la sesión de usuario no pueden ejecutar sus procesos.
- Si los otros agentes no mantienen activa la sesión de usuario, el agente Facebook es el encargado de solicitar el inicio de sesión.

Agente LinkedIn

Caso de uso 1: Iniciar una sesión en Linkedin por medio del agente Linkedin.

Objetivos asociados:

- 1.2.1.1
- 1.2.1.2
- 1.1.3.1
- 1.5.1

Proceso:

- El agente Facebook realiza el monitoreo de sesión al agente LinkedIn, para comprobar que tiene activa la sesión con el agente Facebook, si no tiene activa:
    - El agente Facebook presenta una interfaz para que el usuario se loguee nuevamente con usuario y contraseña al agente Facebook.
    - No puede realizar el proceso con el agente LinkedIn.
- El agente Facebook realiza el monitoreo de sesión al agente LinkedIn, para comprobar que tiene activa la sesión con el agente Facebook, si tiene activa:
    - Si el usuario tiene una sesión de LinkedIn activa, el agente LinkedIn muestra un mensaje para el usuario acepte los términos y condiciones de uso.
    - Si el usuario no tiene una sesión activa de LinkedIn, realiza el login a LinkedIn ingresando usuario y contraseña desde la interfaz del agente LinkedIn y seguidamente se muestra un mensaje para que acepte los términos y condiciones de uso.
- Si el usuario no ha recopilado información desde Facebook con el agente Facebook, no puede realizar el proceso del agente LinkedIn.

Caso de uso 2: Recopilar la información de LinkedIn y almacenarla en la ontología.

Objetivos asociados:

- 1.2.1
- 1.2.1.2
- 1.2.2
- 1.2.2.1
- 1.2.2.2
- 1.1.3.1
- 1.5.1
- 1.2.3

Proceso:

- Si el usuario tiene una sesión activa en LinkedIn y previamente ya ha aceptado los términos y condiciones de uso, se recopilan sus datos, se realiza la asociación de la

información recopilada con el agente Facebook que pertenece al usuario y se almacena la información en la ontología.

- Si el usuario tiene una sesión activa en Facebook y acepta los términos y condiciones de uso, se recopilan sus datos, se realiza la asociación de la información recopilada con el agente Facebook que pertenece al usuario y se almacena la información en la ontología.
- Si el usuario no acepta los términos y condiciones de uso no se recopilan sus datos y no se almacena en la ontología.

Agente Google

Caso de uso 1: Recopilar información de Google por medio del agente Google.

Objetivos asociados:

- 1.3.1
- 1.3.1.1
- 1.3.1.2
- 1.5.1
- 1.1.3.1

Proceso:

- El agente Google necesita que la sesión de usuario que provee el agente Facebook este activa.
- El usuario buscar los datos mediante parámetros, el agente Google filtra los datos y los presenta en 3 secciones: imágenes, web y noticias.
- Si en alguna de las secciones no se encontró información, el agente Google no presenta la información de esa sección.

Caso de uso 2: Guardar los datos seleccionados en la ontología.

Objetivos asociados:

- 1.3.2
- 1.3.2.1
- 1.3.2.2
- 1.5.1
- 1.1.3.1

Proceso:

- El agente Google necesita que la sesión de usuario que provee el agente Facebook este activa.
- De los datos presentados, el usuario selecciona los links de interés y de relación a su persona. El agente Google procesa los datos y almacena en la ontología.

Agente Encuesta

Caso de uso 1: Recopilar información de la encuesta SISCO y almacenar en la ontología.

Objetivos asociados:

- 1.4.1
- 1.4.1.1
- 1.4.1.1.1
- 1.4.1.2
- 1.4.2
- 1.4.2.1
- 1.4.2.1.1
- 1.4.2.2
- 1.4.2.2.1
- 1.5.1
- 1.1.3.1

Proceso:

- El agente Encuesta necesita que la sesión de usuario que provee el agente Facebook este activa.
- El agente Encuesta busca si el usuario ha realizado anteriormente la encuesta.
- Si ha realizado visualiza las coincidencias y selecciona su información,
- Si no ha realizado el agente Encuesta le muestra la opción de realizar la encuesta.

Agente Mailer

Caso de uso 1: Realizar notificaciones periódicas al usuario para que usen el SMA.

Objetivos asociados:

- 1.5.1

- 1.5.1.1
- 1.5.3
- 1.5.4

Proceso:

- El agente Mailer en un intervalo de 30 minutos realiza un monitoreo para observar si se ha incluido nuevos usuarios.
- El agente Mailer realiza una comparación de usuarios para saber cuáles ya han sido notificados.
- Si existen nuevos usuarios que no han sido notificados el agente Mailer notifica vía mail para que el usuario experimente la aplicación.
- Si existen nuevos usuarios que ya han sido notificados el agente Mailer no realiza ninguna acción.
- El agente Mailer terminado el proceso de recopilación enviar un mail al usuario notificando los datos que los agentes han recopilado de cada uno.

Caso de uso 2: Presentar la información al usuario.

Objetivos asociados:

- 1.5.1
- 1.5.3.2

Proceso:

- El agente Mailer es el encargado de realizar las notificaciones de los datos que los agentes han recopilado de los usuarios.

Caso de uso 3: Buscar en la ontología información de los datos recopilados.

Objetivos asociados:

- 1.5.3
- 1.5.3.1

Proceso:

- El usuario o administrador especifica un parámetro de búsqueda en el Agente Mailer.
- El agente Mailer procesa el parámetro.
- El agente Mailer envía la información.

Entregables: diagramas de secuencia

La **figura 51** presenta el diagrama de secuencia del agente Facebook para el caso de uso: Iniciar una sesión en Facebook por medio del agente Facebook.

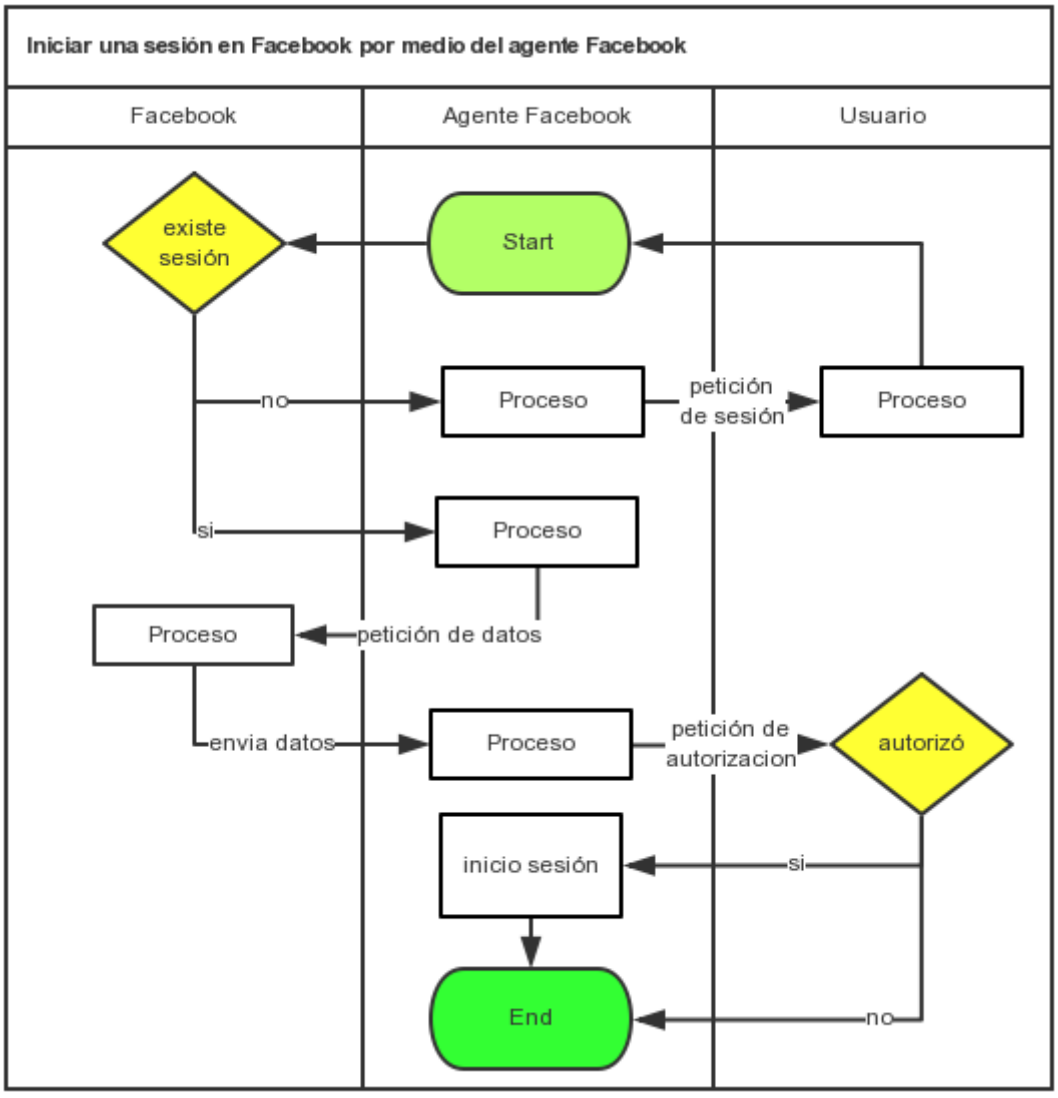


Figura 51: Diagrama de secuencia caso de uso 1 – Agente Facebook.

Fuente: La autora.

La **figura 52** presenta el diagrama de secuencia del agente Facebook para el caso de uso: Recopilar información de Facebook y almacenarla en la ontología.

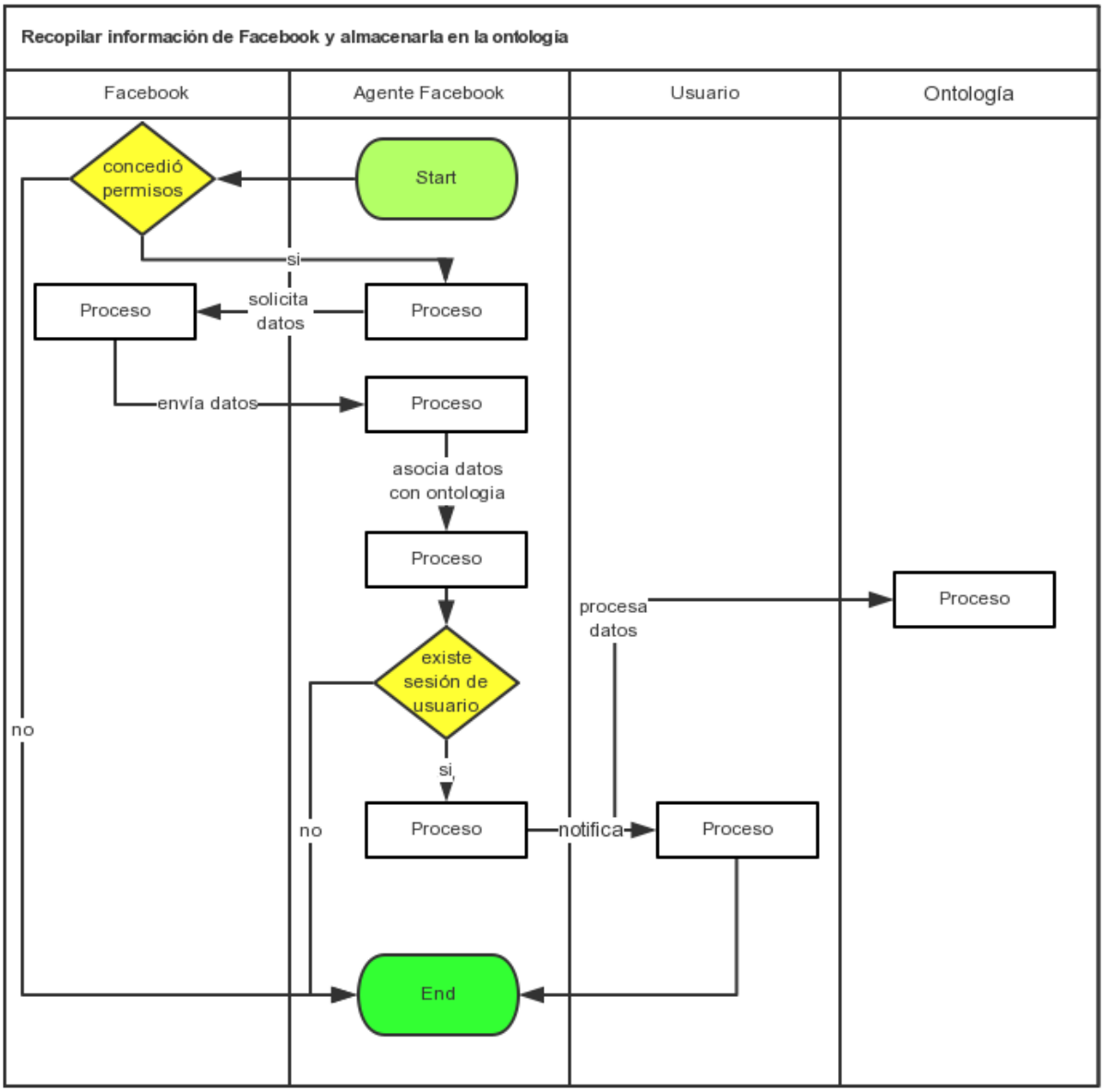


Figura 52: Diagrama de secuencia caso de uso 2 – Agente Facebook.

Fuente: La autora.

La **figura 53** presenta el diagrama de secuencia del agente Facebook para el caso de uso: Crear una sesión de usuario con los datos de Facebook.

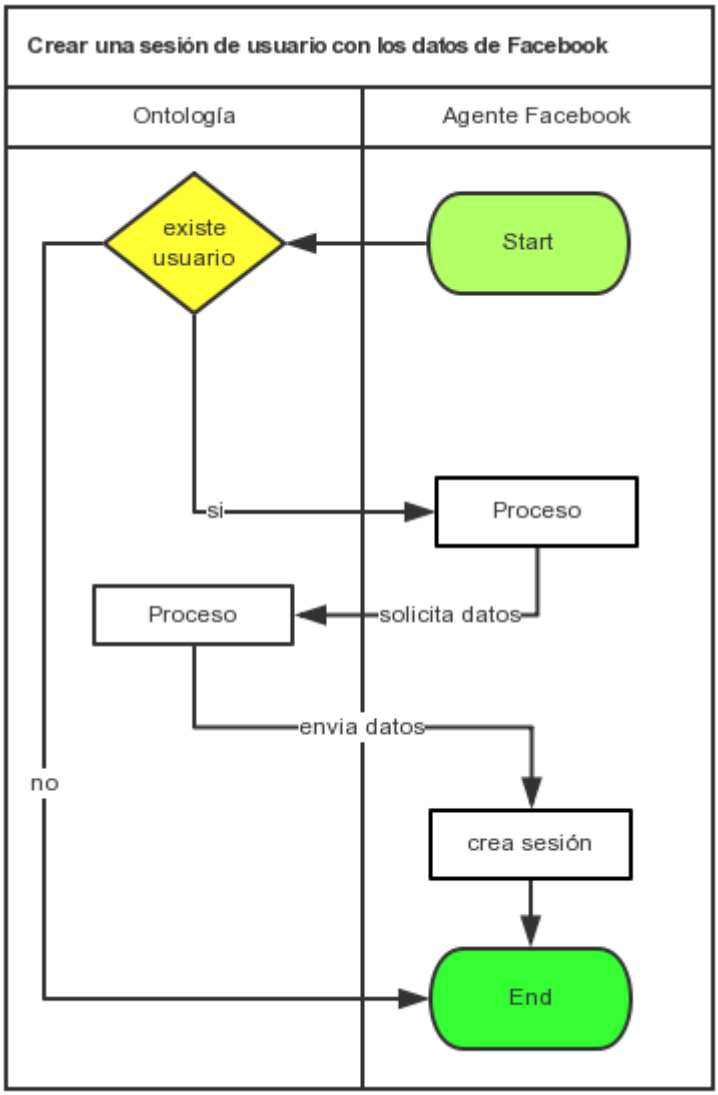


Figura 53: Diagrama de secuencia caso de uso 3 – Agente Facebook.

Fuente: La autora.

La **figura 54** presenta el diagrama de secuencia del agente Facebook para el caso de uso: Monitorear que los agentes mantengan activas las sesiones.

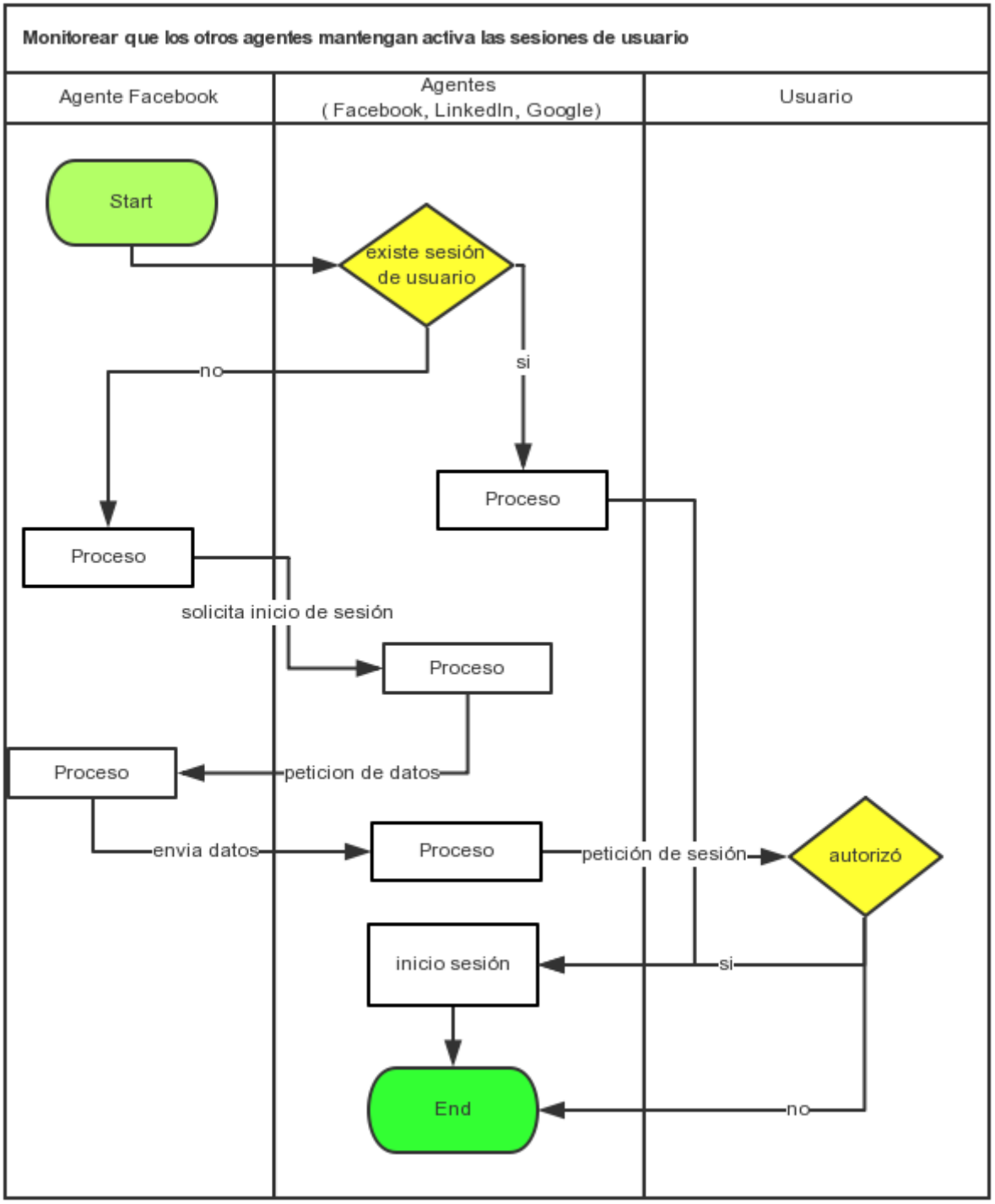


Figura 54: Diagrama de secuencia caso de uso 4 – Agente Facebook.

Fuente: La autora.

La **figura 55** presenta el diagrama de secuencia del agente LinkedIn para el caso de uso: Iniciar una sesión en LinkedIn por medio del agente LinkedIn.

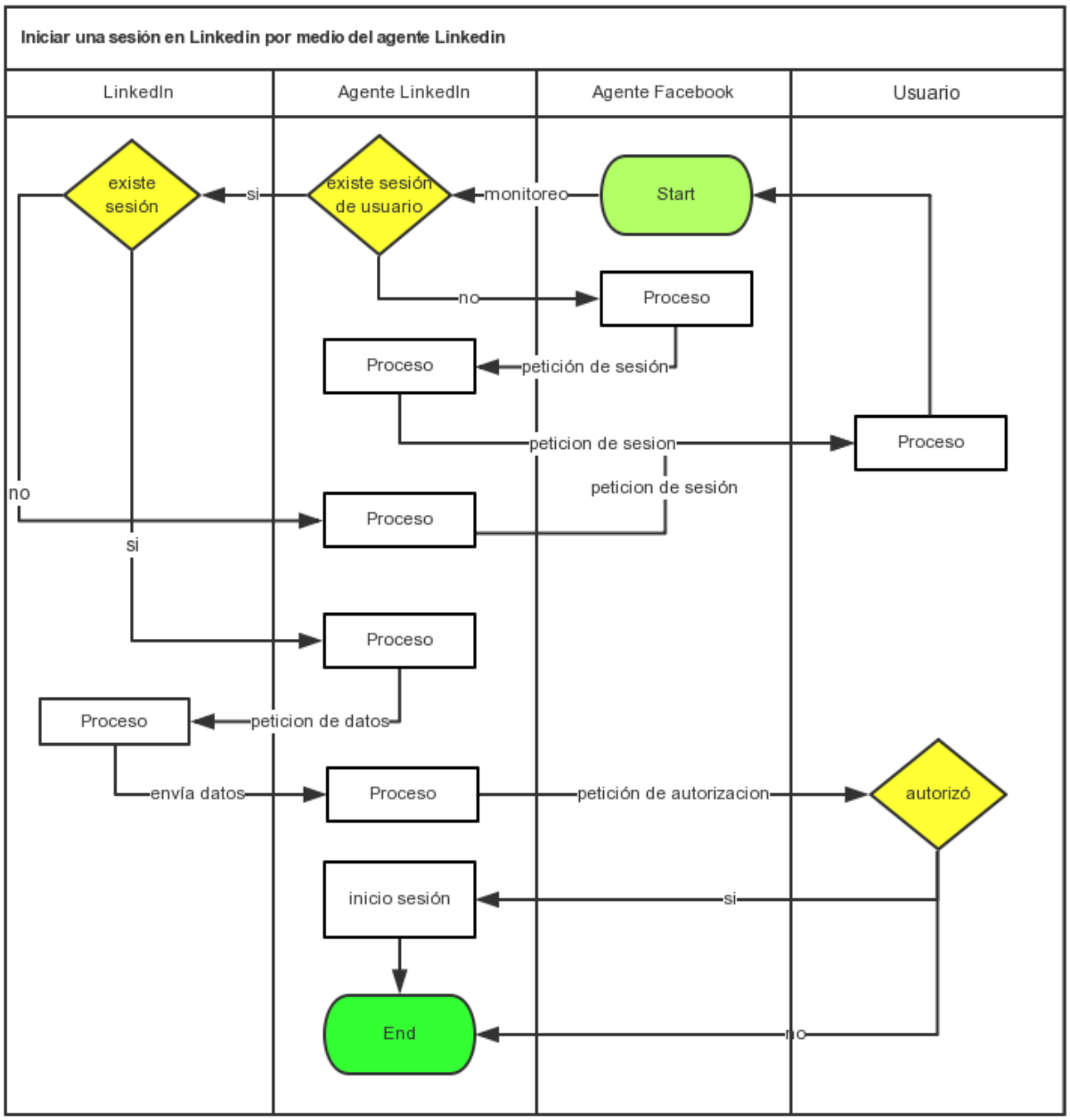


Figura 55: Diagrama de secuencia caso de uso 1 – Agente LinkedIn.

Fuente: La autora.

La **figura 56** presenta el diagrama de secuencia del agente LinkedIn para el caso de uso: Recopilar información de LinkedIn y almacenarla en la ontología.

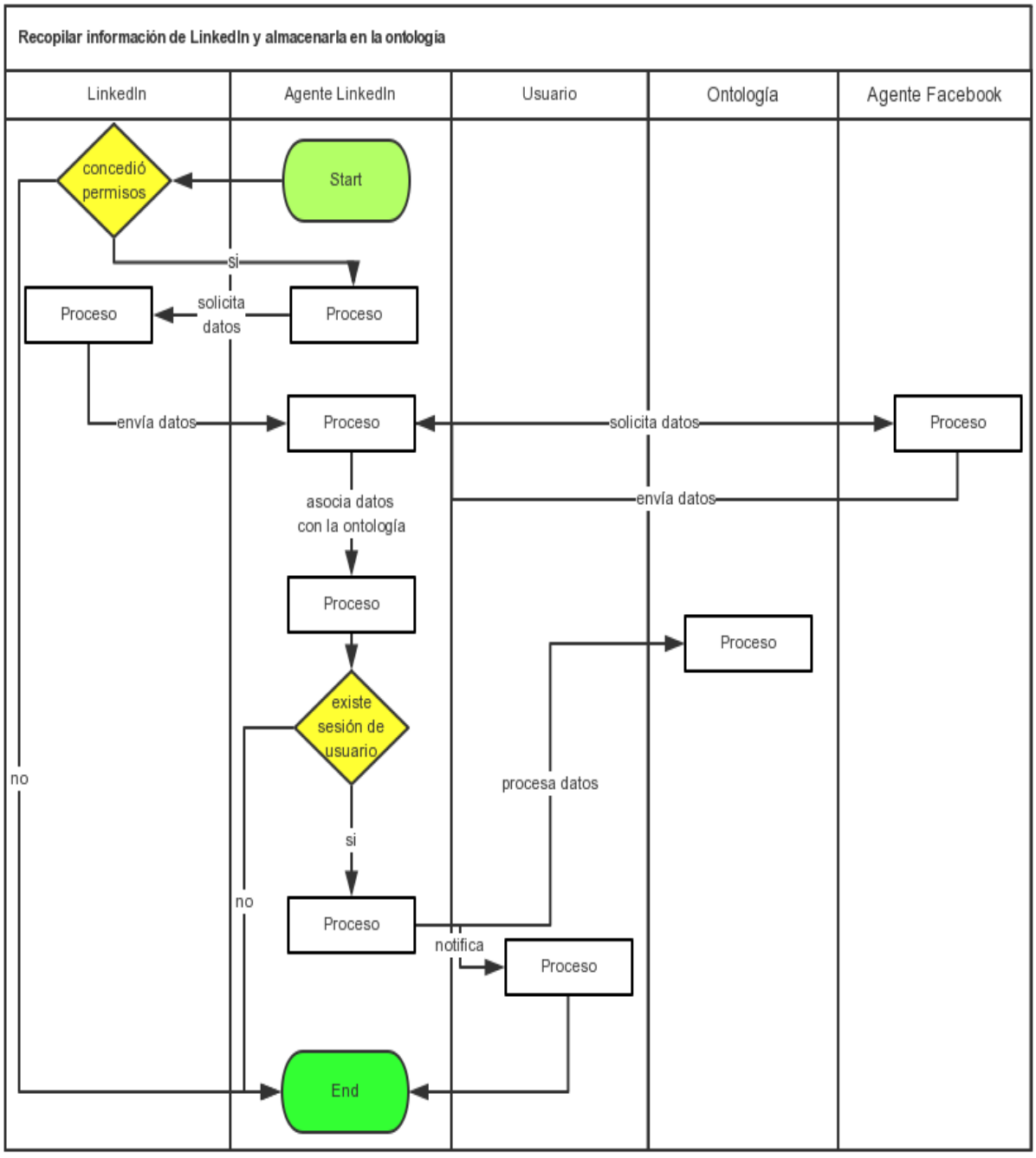


Figura 56: Diagrama de secuencia caso de uso 2 – Agente LinkedIn.

Fuente: La autora.

La **figura 57** presenta el diagrama de secuencia del agente Google para el caso de uso: Recopilar información de Google, por medio del agente Google.

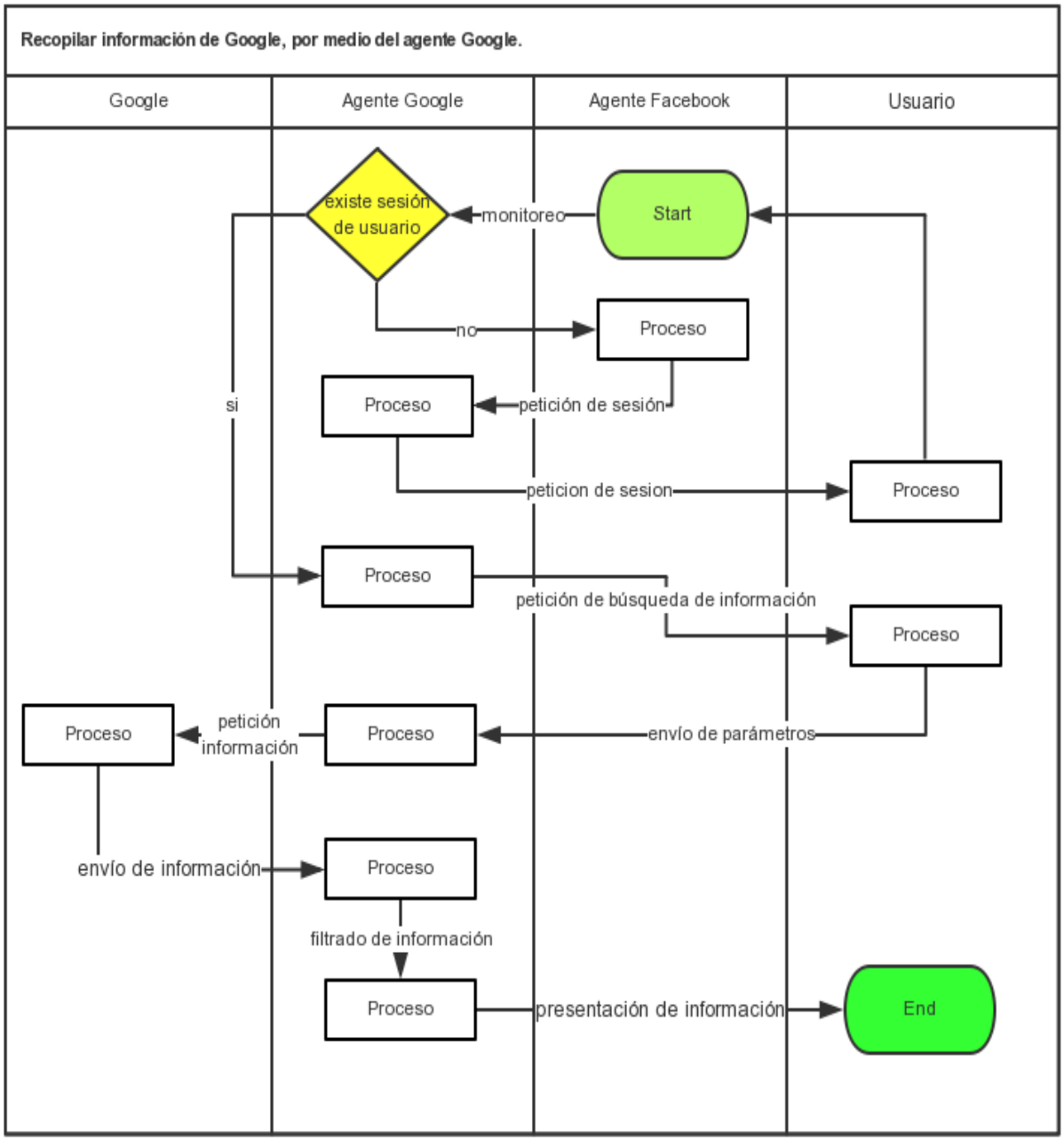


Figura 57: Diagrama de secuencia caso de uso 1 – Agente Google.

Fuente: La autora.

La **figura 58** presenta el diagrama de secuencia del agente Google para el caso de uso: Guardar los datos seleccionados en la ontología.

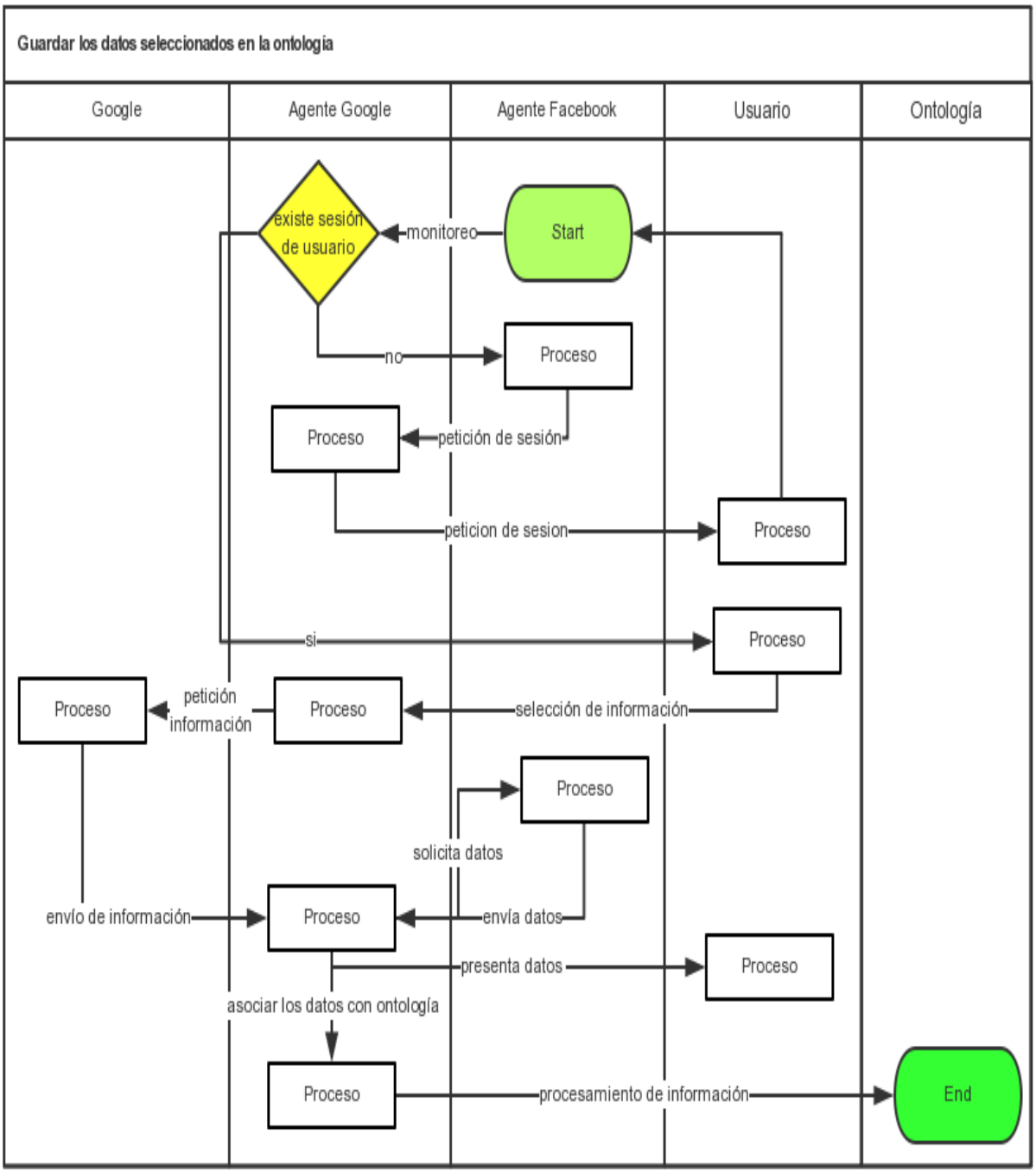


Figura 58: Diagrama de secuencia caso de uso 2 – Agente Google.

Fuente: La autora.

La **figura 59** presenta el diagrama de secuencia del agente Encuesta para el caso de uso: Recopilar información de la encuesta SISCO y almacenar en la ontología.

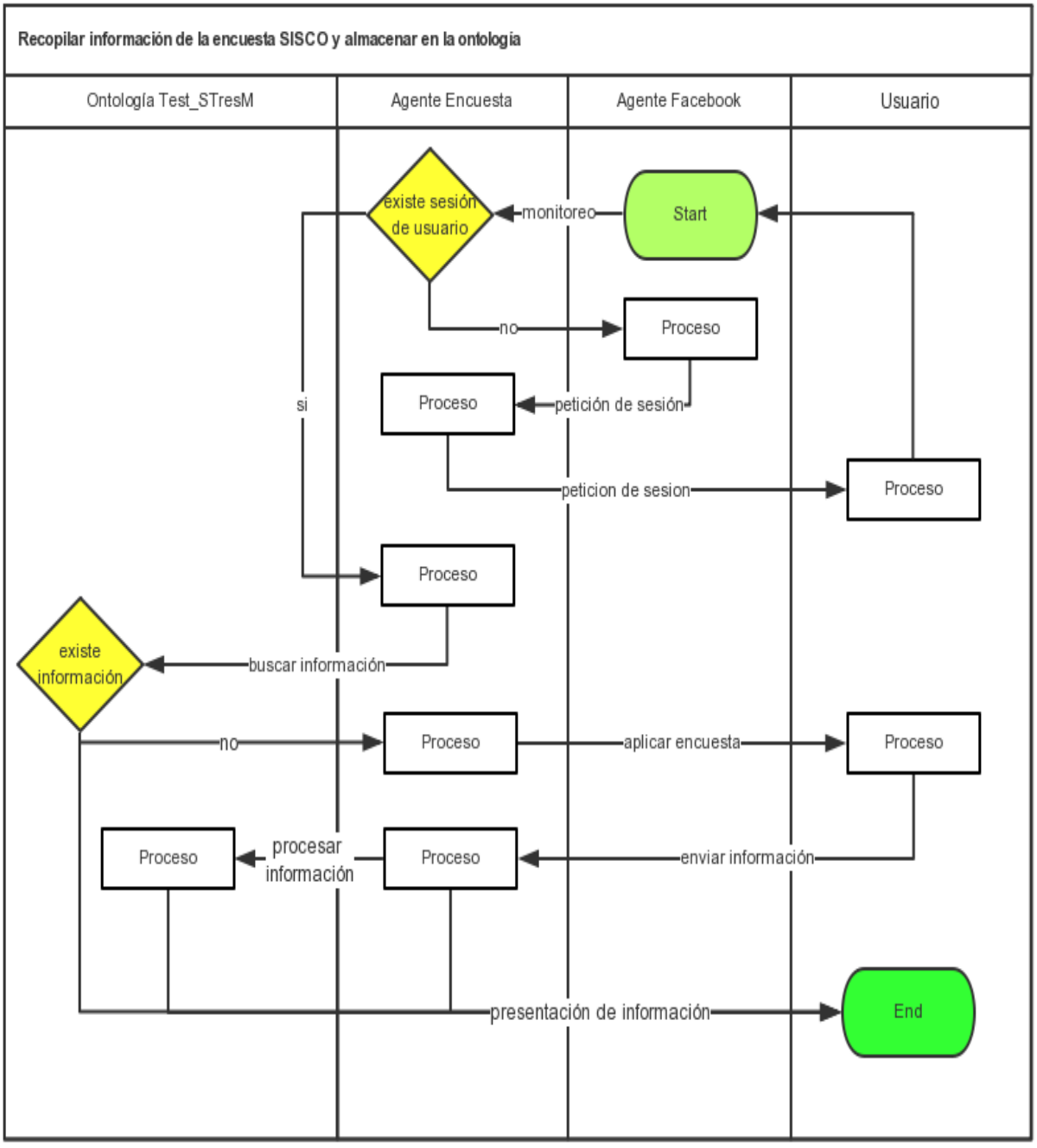


Figura 59: Diagrama de secuencia caso de uso 1 – Agente Encuesta

Fuente: La autora.

La **figura 60** presenta el diagrama de secuencia del agente Mailer para el caso de uso: Realizar notificaciones periódicas al usuario para que usen el SMA.

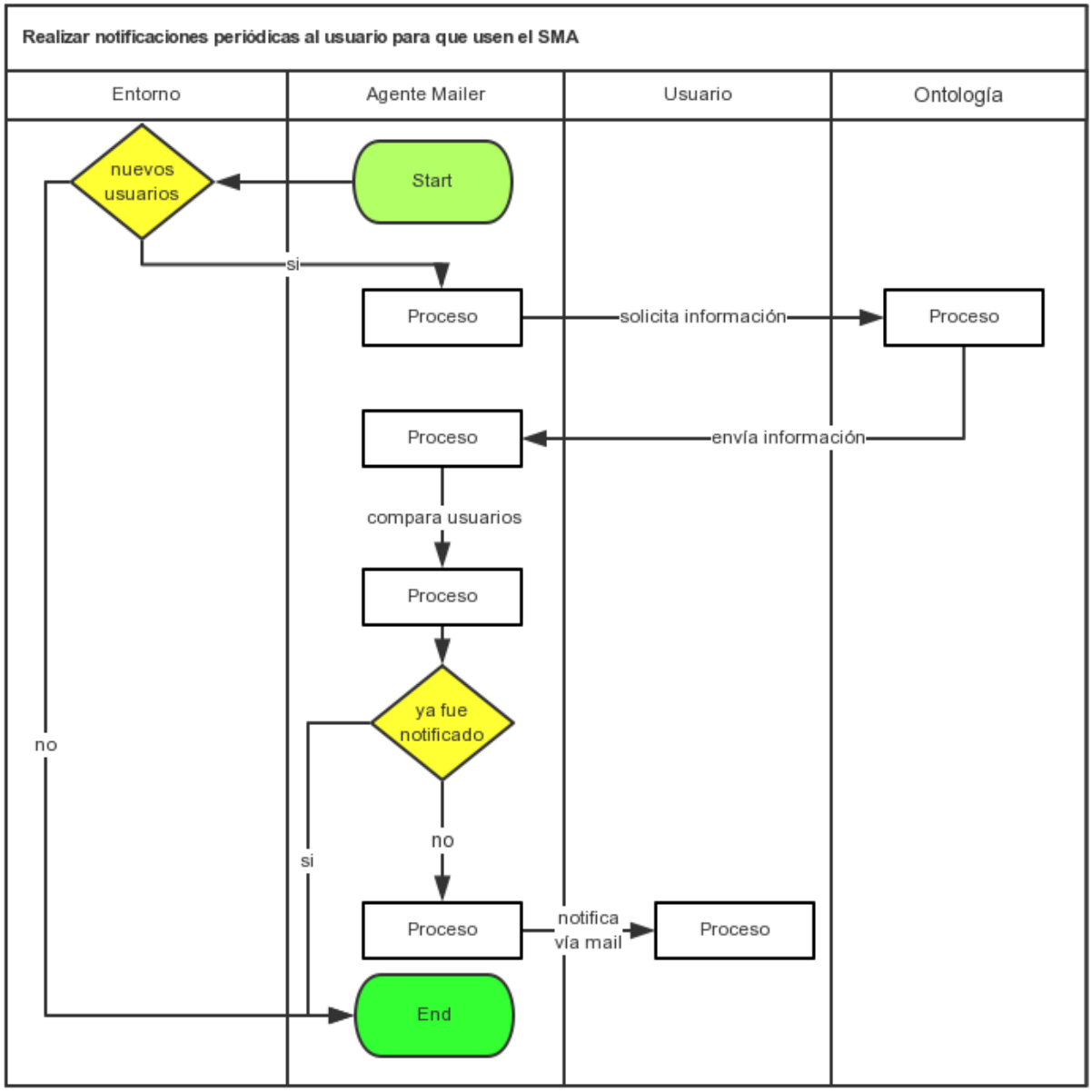


Figura 60: Diagrama de secuencia caso de uso 1 – Agente Mailer.

Fuente: La autora.

La **figura 61** presenta el diagrama de secuencia del agente Mailer para el caso de uso: Presentar información al usuario.

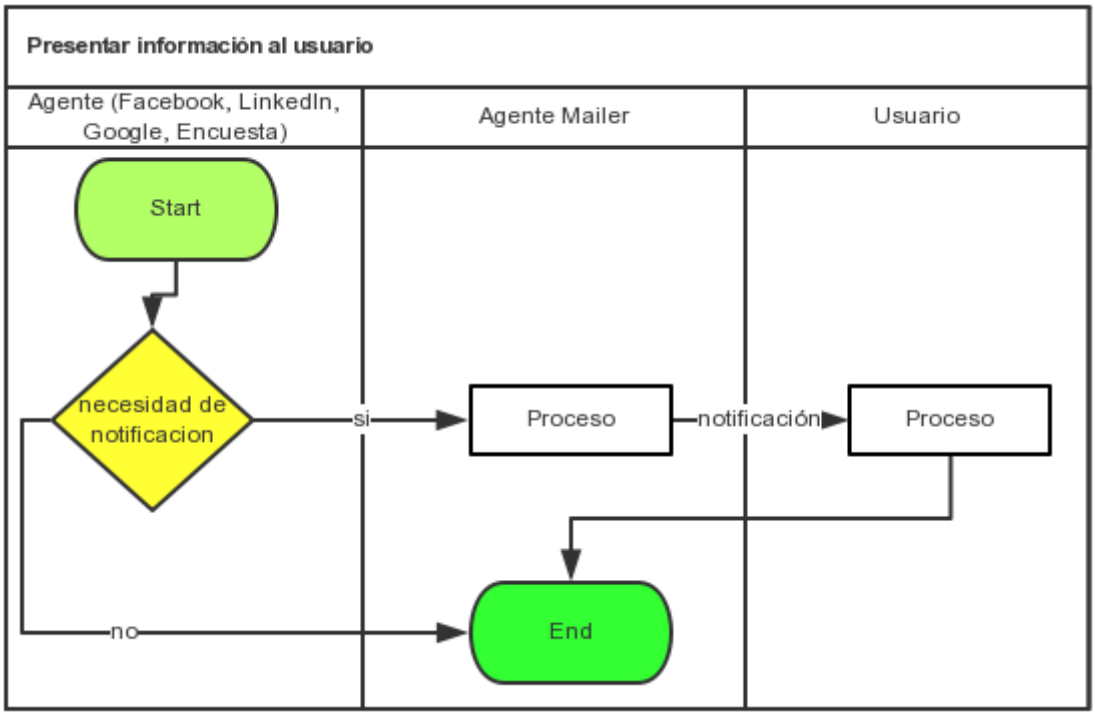


Figura 61: Diagrama de secuencia caso de uso 2 – Agente Mailer.

Fuente: La autora.

La **figura 62** presenta el diagrama de secuencia del agente Mailer para el caso de uso: Buscar en la ontología información de los datos recopilados.

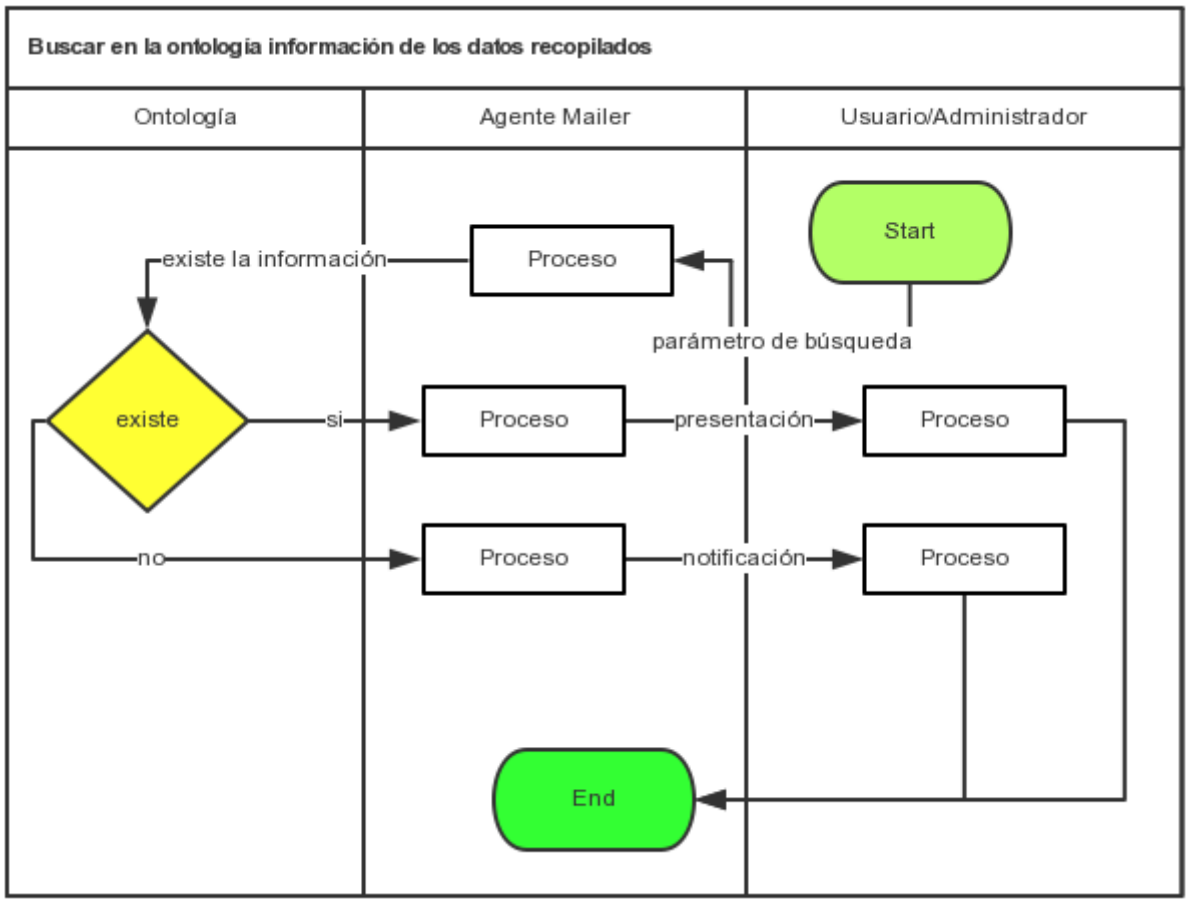


Figura 62: Diagrama de secuencia caso de uso 3 – Agente Mailer.

Fuente: La autora.

3.1.3 *Obtener la ontología del sistema.*

Se cuenta con la ontología Test_EstresM que modela un dominio de aplicación para determinar o inferir el posible estrés académico**.** La ontología cuenta con una población de datos de alrededor de 200 estudiantes que se les aplico la encuesta SISCO.

Dentro de la información que se encuentra almacenada, se habla de síntomas, situaciones y estrategias que el usuario realiza cuando se encuentra frente a una situación de estrés. Sin embargo no se cuenta con información que constituya un perfil personal del usuario, ya que no solo las causas académicas pueden representar estrés sino también las causas personales, como trabajo, relaciones y más.

El propósito es tener una base de datos condesada de información acerca de las personas, para que a un futuro pueda realizarse un análisis que provea información que sea la causa o consecuencia de las acciones que realiza un usuario. Además, que esta base de datos se alimente de información automáticamente, ya que los procesos manuales implican tiempo, agilidad y espera.

Para complementar la información de Test_StressM se representó una nueva ontología llamada People, que es la combinación de Test_StresM y Foaf.

Foaf, es un estándar global de vocabularios ontológicos para representar perfiles personales. Entre los beneficios del uso de Foaf es que en el lenguaje Java, existen Apis y librerías que permiten manejar la ontología de forma dinámica porque ya cuentan con las representaciones de sus conceptos, un ejemplo de estas Apis es Jena.

Además, los conceptos que modela Foaf, tienen una relación directa con la información a recopilar desde los recursos web.

- **FOAF**

Clases:

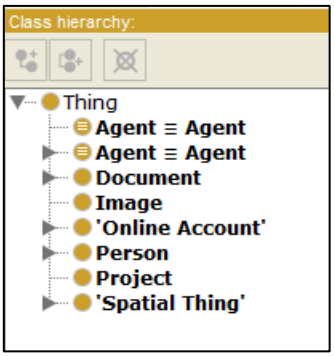


Figura 63: Clases de ontología FOAF.

Fuente: La autora.

La **figura 63** muestra las clases o también comúnmente llamados recursos que contiene un vocabulario, que vienen siendo todas aquellas cosas que pertenecen a un dominio de conocimiento por ejemplo Persona es un dominio que contiene información para definir el nombre, apellido, edad, educación, entre otros.

Subclases:

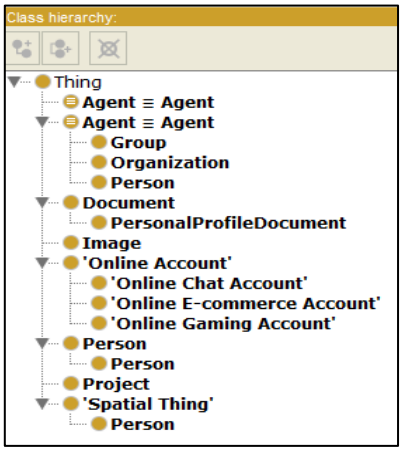


Figura 64: Subclases de ontología FOAF.

Fuente: La autora.

La **figura 64** muestra las subclases de una clase. Cuando se habla de subclases se refiere a la descomposición que puede tener un recurso. En este caso, se observa que; un grupo, organización y persona son parte del recurso Agente.

Foaf ofrece una gama de información amplia para representar a personas, sus relaciones, intereses, trabajo, proyectos y otras características.

Propiedades de relación:

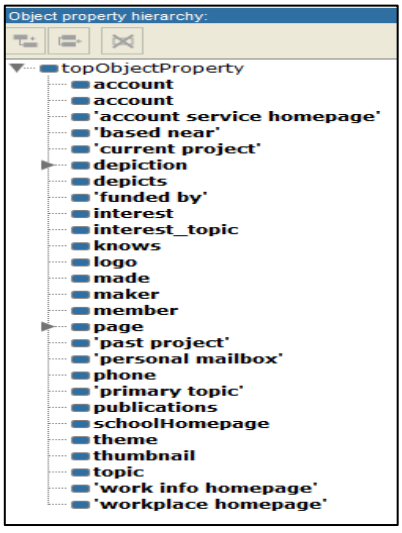


Figura 65: Propiedades de relaciones FOAF.

Fuente: La autora.

La **figura 65** muestra las relaciones o propiedades que tiene un vocabulario. Por ejemplo para la propiedad Knows, existe la relación Person - Knows – Person (Una persona conoce a una persona).

Propiedades de dato:

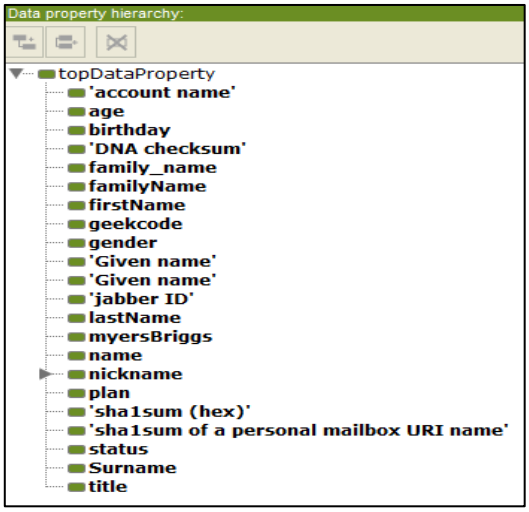


Figura 66: Propiedades de datos FOAF.

Fuente: La autora.

La **figura 66** presenta los literales del vocabulario que también son conceptos o recursos de la ontología. Y ayudan a dar propiedades y características a las clases. Por ejemplo; la propiedad de dato FirstName, name, Surname, gender pertenecen al dominio Persona, por que una persona cuenta con un primer nombre y otras características para dar un entendimiento del dominio de información.

- **Test_EstressM**

Clases:

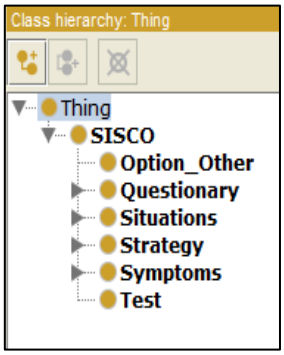


Figura 67: Clases de ontología Test_EstresM.

Fuente: La autora.

La **figura 67** muestra las clases del vocabulario Test_EstresM.

Subclases:

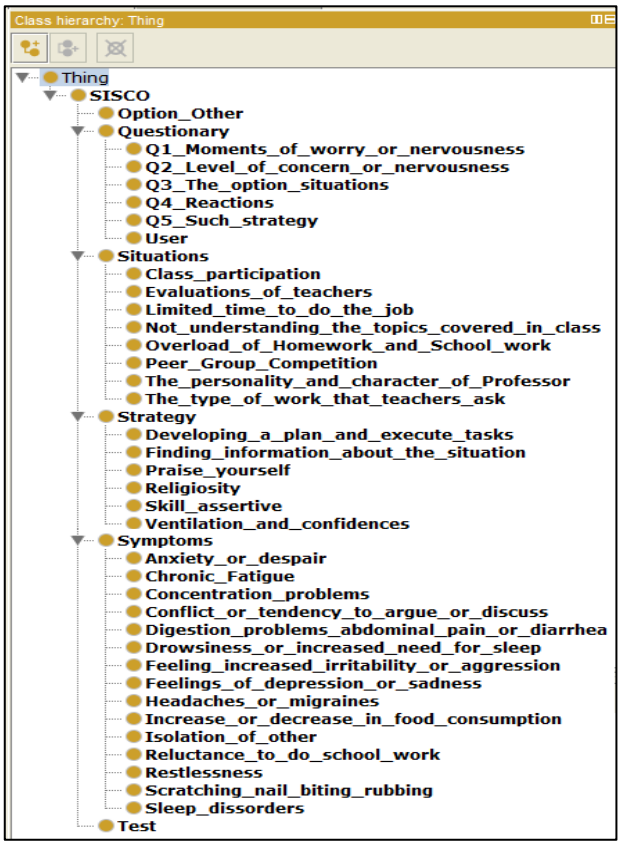


Figura 68: Subclases de ontología Test_EstresM

Fuente: La autora.

La **figura 68** muestra las sub clases del vocabulario Test_EstresM.

Propiedades de relación:

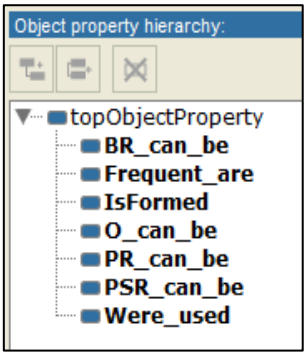


Figura 69: Propiedades de relación Test_EstresM

Fuente: La autora.

La **figura 69** muestra las propiedades de relación de la encuesta Test_EstresM.

Se observa que un Test está formado por un Cuestionario. La representación se presenta en la **figura 70**.

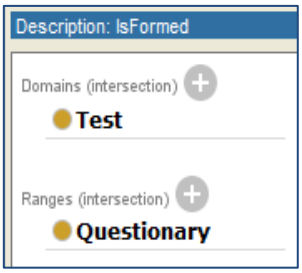


Figura 70: Relación test-Questionary Test_EstresM

Fuente: La autora.

Propiedades de dato:

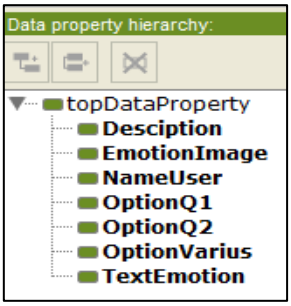


Figura 71: Propiedades de dato Test_EstresM

Fuente: La autora.

La **figura 71** muestra las propiedades de dato de Test_EstresM.

Literales:

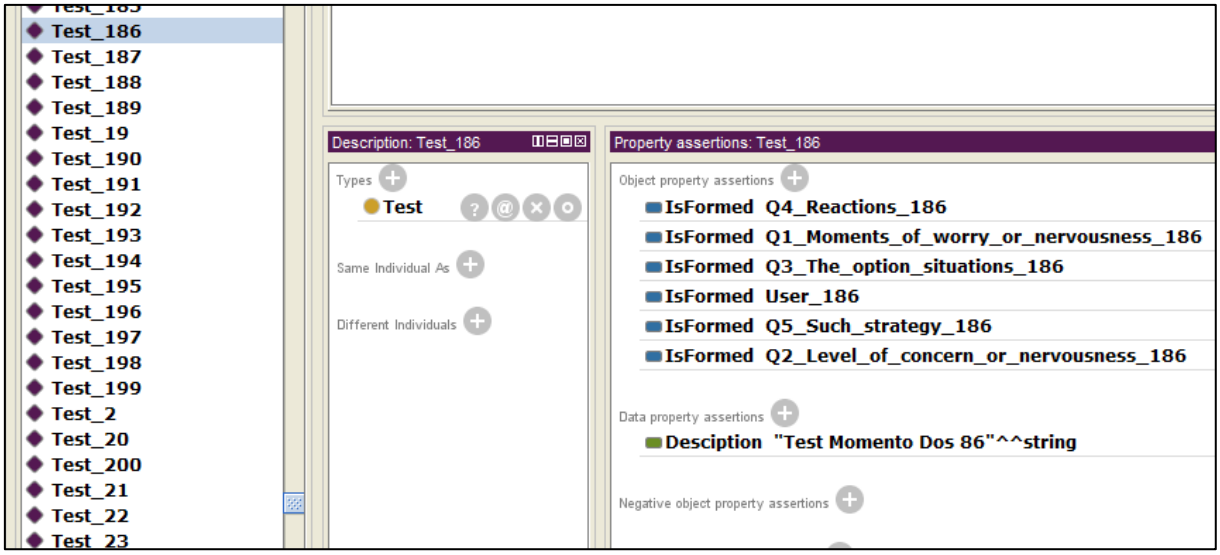


Figura 72: Vista de literales de Test_EstresM

Fuente: La autora.

La **figura 72** muestra una vista de los literales almacenados en Test_EstresM.

- **PEOPLE**

Es una combinación producto de la unión de dos clases de ontologías Test_EstresM y Foaf.

La **figura 73** muestra el diseño final de la ontología luego de haber realizado la combinación de Foaf y TestEstres_M.

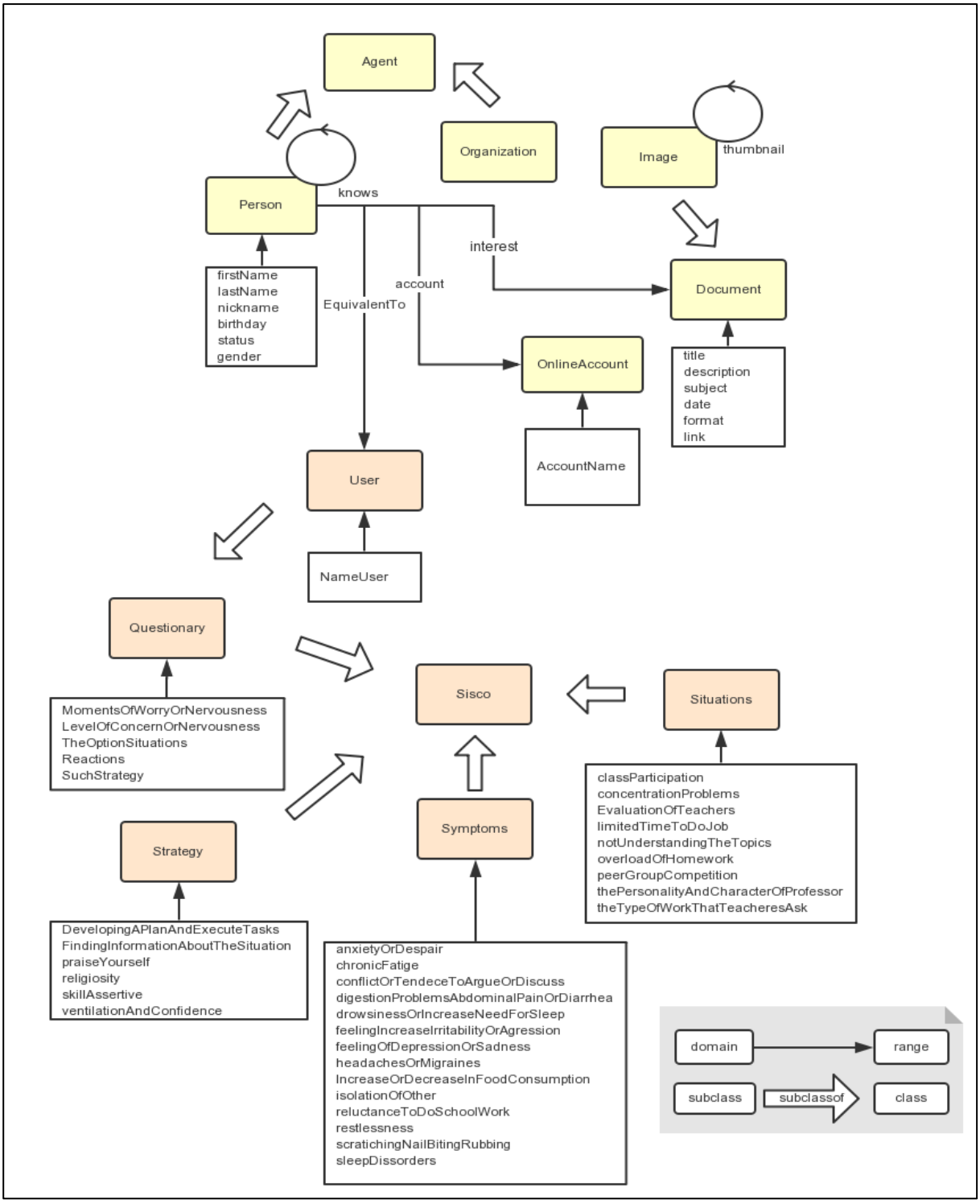


Figura 73: Diseño de ontología People

Fuente: La autora.

El objetivo es dar un mejor entendimiento de los usuarios. Como se explicó Test_EstresM tiene una población de información en referencia a un usuario, pero no define las características del usuario, para poder tener un mejor entendimiento del dominio Persona y otras características que pueden influir para presentar estrés.

En la **figura 74** se observa que Test_EstresM contiene una subclase Usuario que pertenece a la clase principal Cuestionario. Esta clase únicamente está definida por la propiedad de dato nombre, lo que limita la representación de la información de una persona.

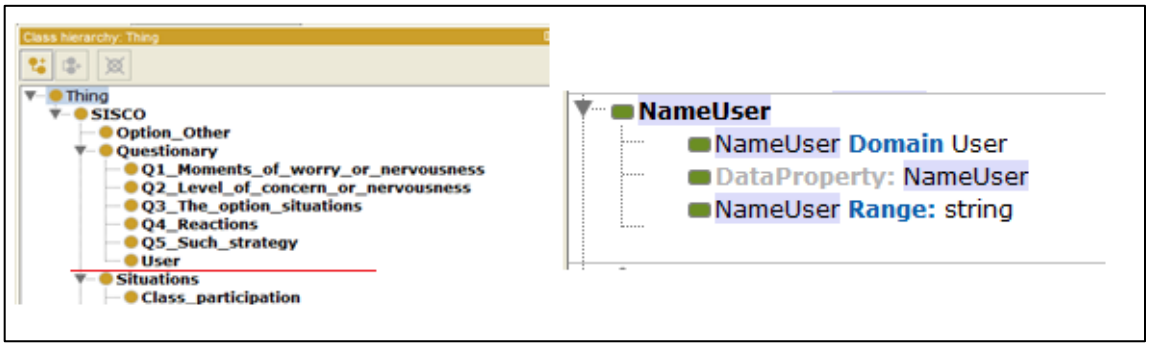


Figura 74: Propiedades de dato Test_EstresM – representación de usuario

Fuente: La autora.

Para ayudar a complementar la información de persona, en la **figura 75** se observa la asociación de Usuario de Test_EstresM con la clase Persona de Foaf.

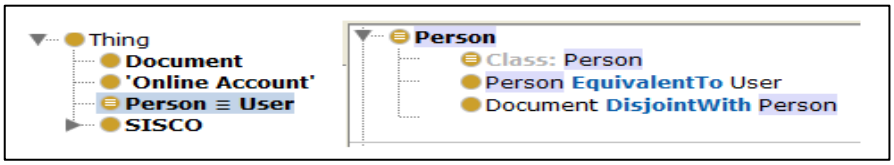


Figura 75: Asociación de FOAF – Test_EstressM

Fuente: La autora.

En la **figura 75** se observa que el dominio Persona es igual al dominio Usuario.

En la **figura 75** también se observa la descomposición de tripletas. Persona es un tipo de Clase, Persona es equivalente a Usuario y Persona es disjunta de Documento, lo que significa que aunque ambos conceptos son clases, no son iguales.

En la fase de análisis en MaSE, la tarea *Obtener la ontología del sistema*, se especifica que es necesario representar únicamente los conceptos que vayan ser usados en el SMA. En la explicación de la información que se va a recopilar desde los recursos web con la asociación de los agentes ya se manifiesta la correspondencia de la información y se expone los datos que son usados de Foaf.

Después de realizar la combinación de las ontologías se tiene las correspondientes clases, subclases, propiedades de objeto y propiedades de dato de la ontología que se presentan a continuación.

Clases

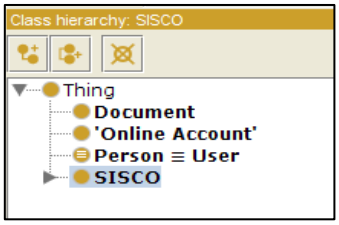


Figura 76: Clases People

Fuente: La autora.

En la **figura 76** se observa las clases de People luego de haber realizado la combinación de Foaf y TestEstres_M.

Subclases

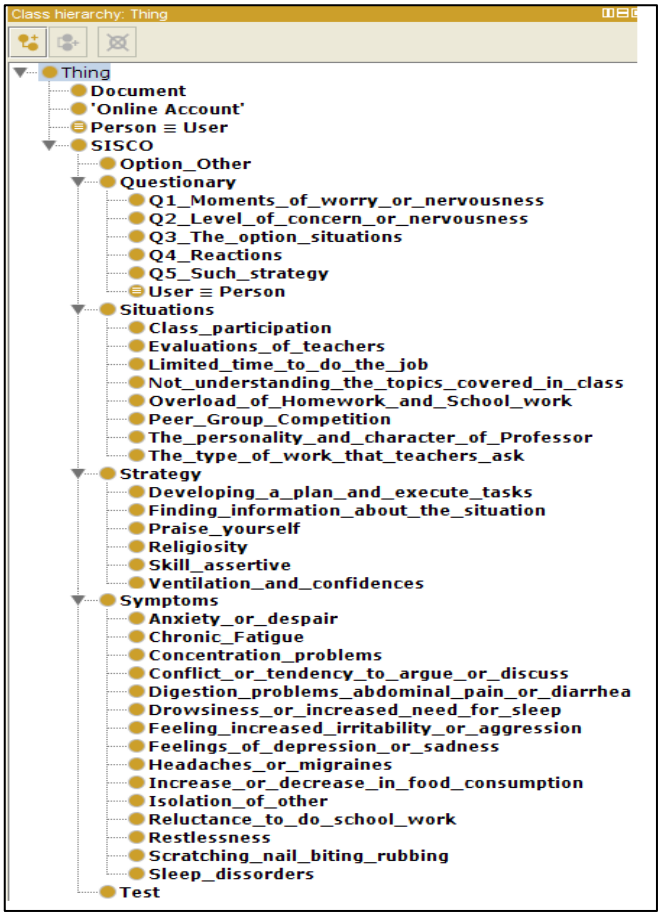


Figura 77: Subclases People

Fuente: La autora.

En la **figura 77** se observa las subclases de People luego de haber realizado la combinación de Foaf y TestEstres_M.

Propiedades de relación

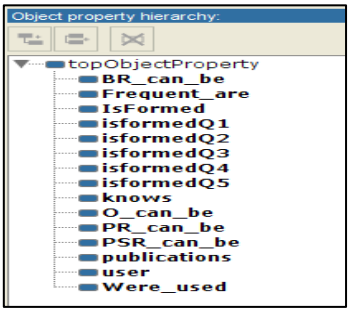


Figura 78: Propiedades de relación People

Fuente: La autora.

En la **figura 78** se observa las propiedades de relación de People.

Propiedades de dato

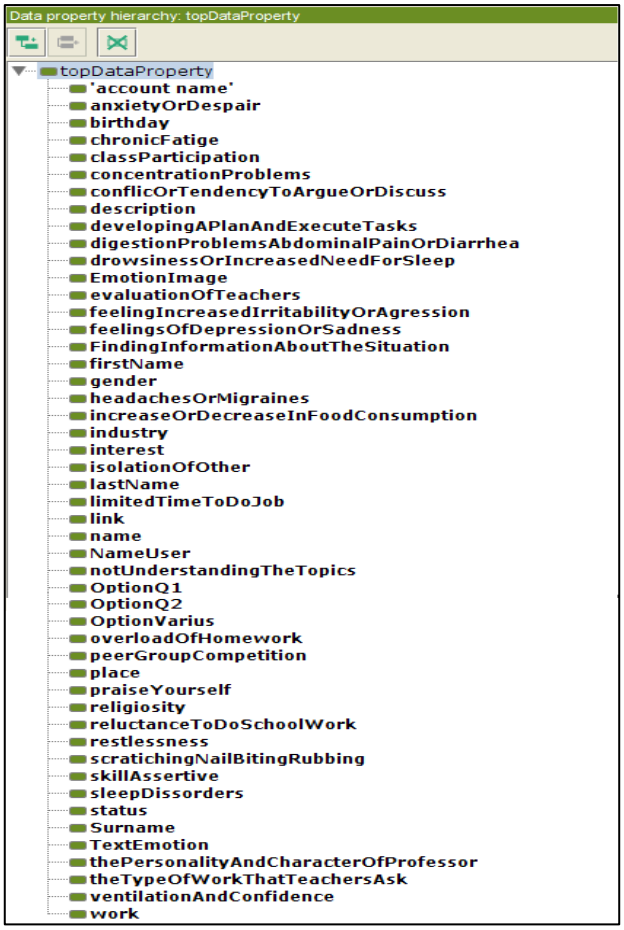


Figura 79: Propiedades de dato

Fuente: La autora.

En la **figura 79** se observa las propiedades de dato de la ontología People.

- **Mapeo de información**

El agente Facebook, Linkedin y el agente Google son los encargados de realizar la recopilación de información de diversos recursos web. Esta información se asocia con la ontología.

Para el agente Facebook:

Tabla 6: Asociación de conceptos de Facebook con ontología

Datos de Facebook	Asociación con conceptos de la ontología
Nombre completo	<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/**name**" vs:term_status="testing" rdfs:label="name" rdfs:comment="A name for some thing."> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/> <rdfs:domain rdf:resource="http://www.w3.org/2002/07/owl#Thing"/> <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/> <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/"/> <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#label"/> </rdf:Property>
Primer nombre	<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/**firstName**" vs:term_status="testing" rdfs:label="firstName" rdfs:comment="The first name of a person."> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/> <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/> <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/> <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/"/> </rdf:Property>
Primer apellido	<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/**lastName**" vs:term_status="testing" rdfs:label="lastName" rdfs:comment="The last name of a person."> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/> <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/> <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/> <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/"/> </rdf:Property>
Email	<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/**accountName**" vs:term_status="testing" rdfs:label="account name" rdfs:comment="Indicates the name (identifier) associated with this online account."> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/> <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/OnlineAccount"/> <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/> <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/"/> </rdf:Property>

Cumpleaños	<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/**birthday**" vs:term_status="unstable" rdfs:label="birthday" rdfs:comment="The birthday of this Agent, represented in mm-dd string form, eg. '12-31'."> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/> <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Agent"/> <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/> <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/"/> </rdf:Property>
Género	<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/**gender**" vs:term_status="testing" rdfs:label="gender" rdfs:comment="The gender of this Agent "> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/> <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Agent"/> <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/> <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/"/> </rdf:Property>
Estado	<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/**status**" vs:term_status="unstable" rdfs:label="status" rdfs:comment="A string expressing what the user is happy for the general public (normally) to know about their current activity."> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/> <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Agent"/> <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/> <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/"/> </rdf:Property>
Nickname	<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/**surname**" vs:term_status="archaic" rdfs:label="Surname" rdfs:comment="The surname of some person."> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/> <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/> <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/> <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/"/> </rdf:Property>
Trabajo	<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/**work**" vs:term_status="testing" rdfs:label="current project" rdfs:comment="A current project this person works on."> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/> <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/> <rdfs:range rdf:resource="http://www.w3.org/2002/07/owl#Thing"/> <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/"/> </rdf:Property>

Fuente: La autora.

Para el agente LinkedIn:

Tabla 7: Asociación de conceptos de LinkedIn con ontología

Datos de Linkedin	Asociación con conceptos de la ontología
Industria a la que pertenece	<rdfs:Class rdf:about="http://xmlns.com/foaf/0.1/**Organization**" rdfs:label="Organization" rdfs:comment="An organization." vs:term_status="stable"> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/> <rdfs:subClassOf rdf:resource="http://xmlns.com/foaf/0.1/Agent"/> <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/"/> <owl:disjointWith rdf:resource="http://xmlns.com/foaf/0.1/Person"/> <owl:disjointWith rdf:resource="http://xmlns.com/foaf/0.1/Document"/> </rdfs:Class>
Trabajo actual	<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/**currentProject**" vs:term_status="testing" rdfs:label="current project" rdfs:comment="A current project this person works on."> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/> <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/> <rdfs:range rdf:resource="http://www.w3.org/2002/07/owl#Thing"/> <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/"/> </rdf:Property>
Lugar de residencia	<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/**place**" vs:term_status="archaic" rdfs:label="Surname" rdfs:comment="The place of some person live"> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/> <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/> <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/> <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/"/> </rdf:Property>
Intereses	<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/**interest**" vs:term_status="testing" rdfs:label="interest" rdfs:comment="A page about a topic of interest to this person."> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/> <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Agent"/> <rdfs:range rdf:resource="http://xmlns.com/foaf/0.1/Document"/> <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/"/> </rdf:Property>
Personas que conoce	<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/**knows**" vs:term_status="stable" rdfs:label="knows" rdfs:comment="A person known by this person (indicating some level of reciprocated interaction between the parties)."> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/> <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/> <rdfs:range rdf:resource="http://xmlns.com/foaf/0.1/Person"/> <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/"/> </rdf:Property>

Fuente: La autora.

Para el agente Google:

Tabla 8: Asociación de conceptos de Google con ontología

Datos de Linkedin	Asociación con conceptos de la ontología
Link en relación	<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/**publications**" vs:term_status="testing" rdfs:label="publications" rdfs:comment="A link to the publications of this person."> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/> <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/> <rdfs:range rdf:resource="http://xmlns.com/foaf/0.1/Document"/> <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/"/> </rdf:Property>

Fuente: La autora.

3.1.4 *Refinar los roles.*

En este paso se transforma los diagramas de secuencia y casos de uso, en roles y sus funciones.

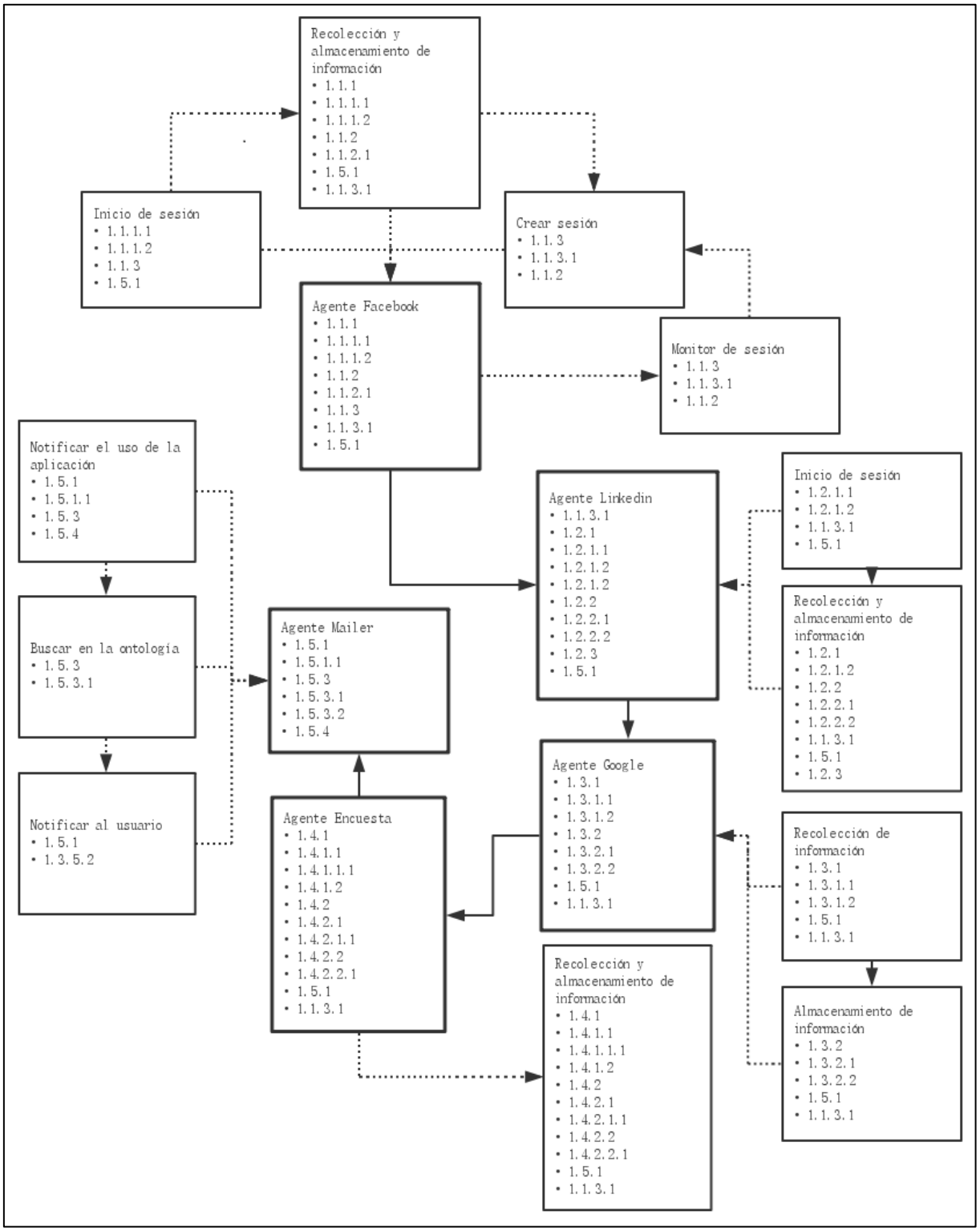


Figura 80: Diagrama de roles - aplicación

Fuente: La autora.

3.2 Fase de diseño

Se muestra el modelado de la fase de diseño y los diagramas resultantes encontrados con MaSE, en esta fase se obtienen diagrama de clases de agente, diagrama de conversaciones y diagrama de arquitectura.

3.2.1 *Creación de las clases de agentes.*

Describe un diagrama de clases de agentes de las funciones encontradas y los roles. Además las posibles conversaciones que surgen con otros agentes.
La **figura 81** muestra el diagrama de clases de agente resultante.

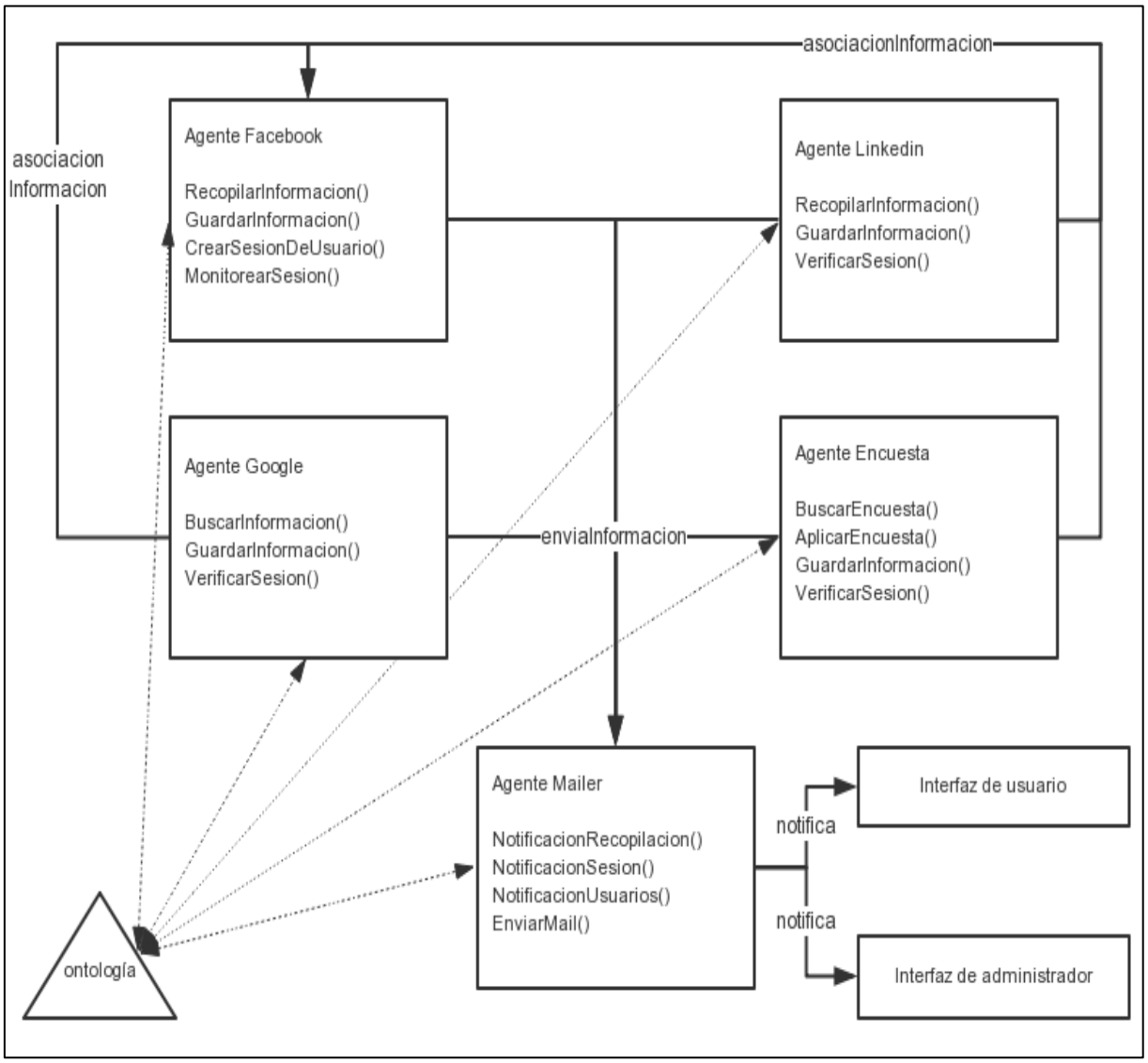


Figura 81: Diagrama de clases de agente - aplicación

Fuente: La autora.

3.2.2 *Construcción de conversaciones.*

Muestra las conversaciones que realizan los agentes para cumplir sus objetivos globales.

La **figura 82** muestra el diagrama de conversaciones resultantes para el monitoreo de sesión del agente Facebook.

El monitoreo de sesión consiste en comprobar que todos los agentes se han logueado con sus datos de Facebook. Mantener esta sesión activa sirve para realizar la asociación de la información proveniente de todos los recursos.

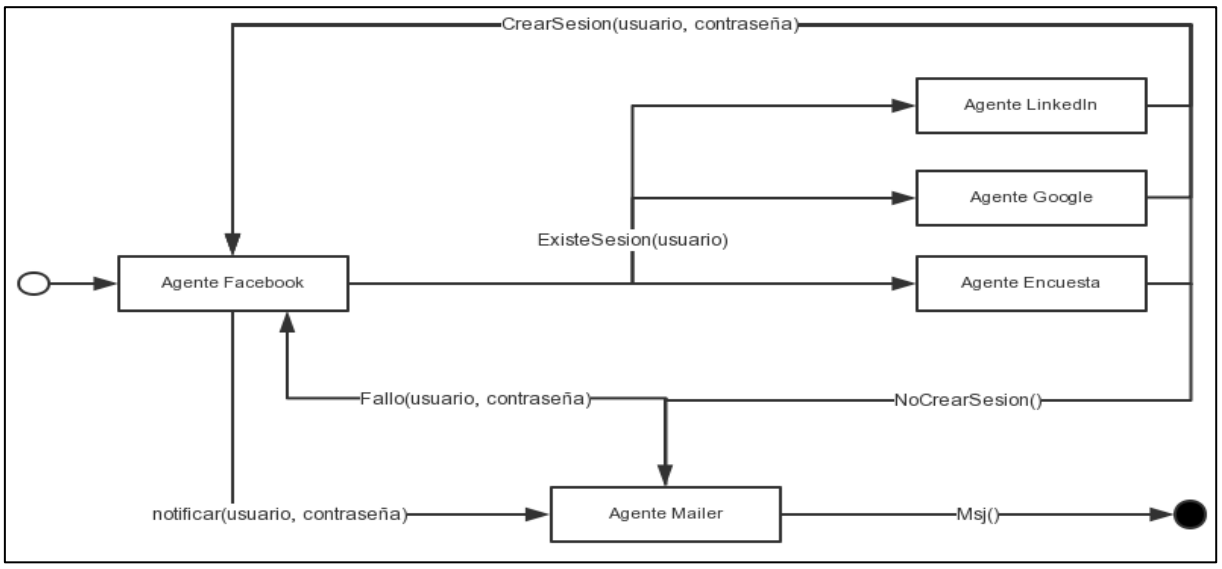


Figura 82: Diagrama de clases de agente - aplicación

Fuente: La autora.

La **figura 83** muestra el diagrama de conversaciones resultante para la verificación de la sesión que realizan los agentes y la asociación de la información con los datos de Facebook.

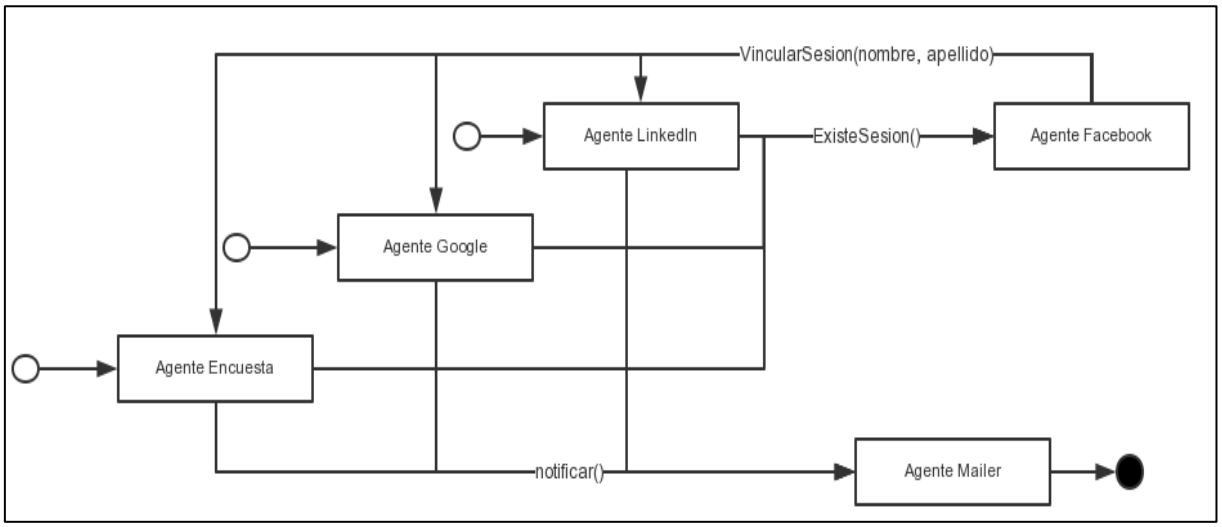


Figura 83: Diagrama de conversaciones – asociación de información

Fuente: La autora.

3.2.3 *Ensamblaje de las clases de agente.*

Se obtiene los diagramas de arquitectura resultantes del SMA.

Agente Facebook

La **figura 84** muestra el diagrama de arquitectura del agente Facebook.

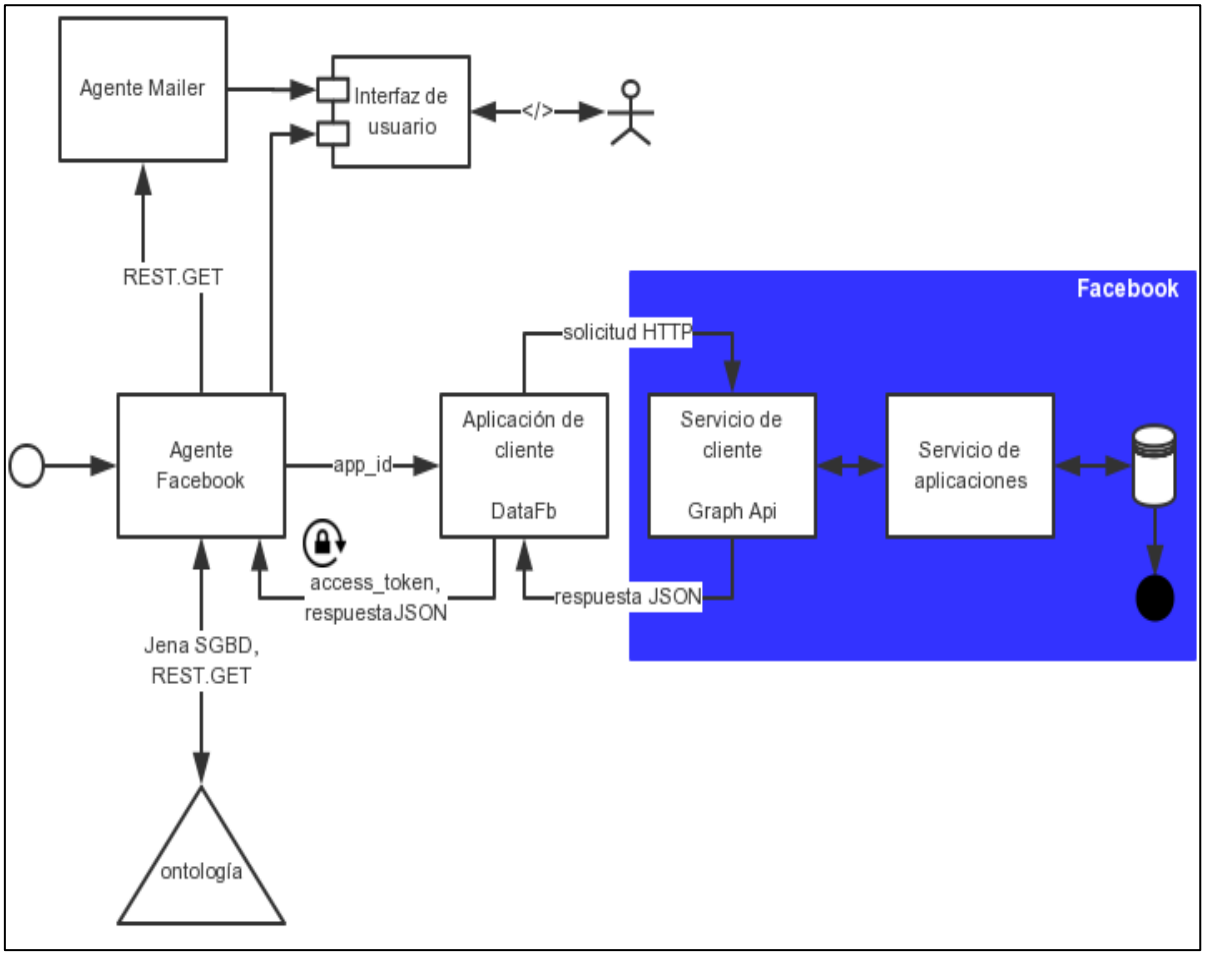


Figura 84: Diagrama de arquitectura agente Facebook

Fuente: La autora.

En la **figura 84** la porción en azul corresponde al proceso que realiza Facebook para permitir la recopilación de información. Facebook cuenta con un servicio cliente llamado Graph Api, este servicio permite el consumo de los datos de Facebook mediante la creación de una aplicación de cliente, que se la ha llamado DataFb.

La aplicación cliente DataFb realiza una solicitud HTTP al servicio cliente Graph Api, este envía una respuesta en formato JSON. Ahora los datos se encuentran dentro de la aplicación cliente DataFb para permitir el consumo.

Para consumir estos datos se necesita obtener un token de acceso y crear un Id de aplicación, que son establecidos cuando se realiza la aplicación cliente.

El agente Facebook realiza la petición de información y envía el id de aplicación a la aplicación cliente DataFb, está verifica si los datos son correctos y envía la aprobación del token de acceso y la respuesta de la información en formato JSON.

El agente Facebook ahora cuenta con los datos en formato JSON y crea un recurso REST que usa el método GET para enviar los parámetros y almacenar los datos en la ontología. En el recurso se ejecutan procesos de gestión de información con la Api Jena, que permite manipular ontologías.

Cuando la información se ha almacenado, se envía la respuesta del agente Facebook al agente Mailer, mediante la creación de un recurso REST que usa el método GET para que notifique al usuario que los datos han sido almacenados. A su vez el agente Facebook presenta los datos al usuario, en la interfaz de usuario mediante un esquema HTML.

La **tabla 9** muestra las tecnologías, especificaciones y versiones usadas para la construcción del agente.

Tabla 9: Especificaciones agente Facebook

Tecnología	**Especificación**	**Versión**
Lenguaje de programación:	*JAVA	-
	*JavaScript	-
IDE utilizado:	NETBEANS	7.3.1
Bibliotecas:	JQuery	-
Api:	*Jena Api	2.6.4
	*Graph Api	-

Fuente: La autora.

La **figura 85** muestra los parámetros necesarios para conseguir el acceso a la aplicación cliente DataFb desde el agente Facebook.

```
config: {
        app_id: 'XXXXXXXXXXX',
        use_xfbml: true,
        extendPermissions: 'publish_stream',
        locale: 'es_ES'
    }
```

Figura 85: Parámetros para conseguir acceso a DataFb desde agente Facebook.

Fuente: La autora.

Agente LinkedIn

La **figura 86** muestra el diagrama de arquitectura del agente LinkedIn.

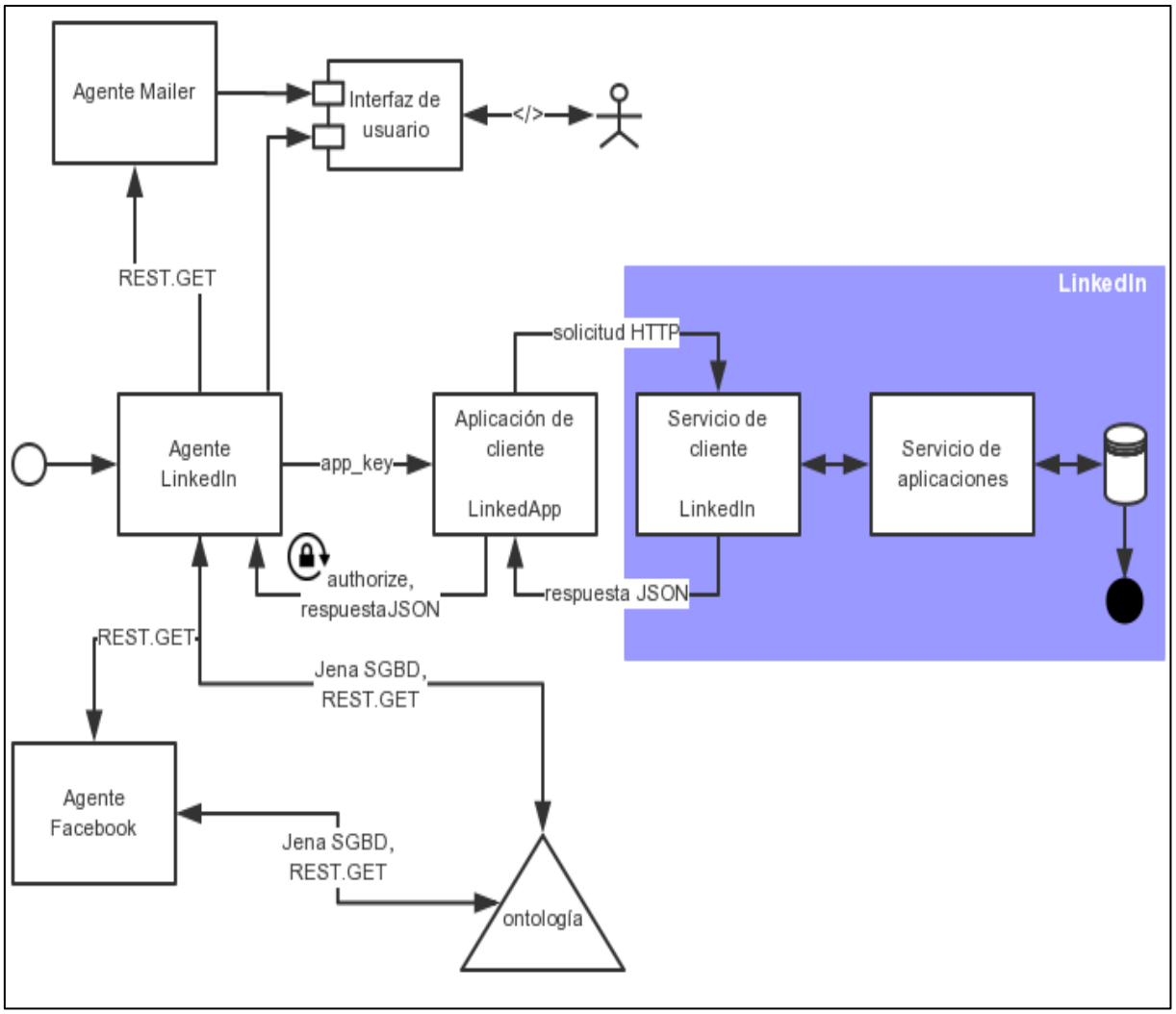


Figura 86: Diagrama de arquitectura agente LinkedIn

Fuente: La autora.

En la **figura 86** la porción en azul corresponde al proceso que realiza LinkedIn para permitir la recopilación de información. Para consumir los datos desde LinkedIn se necesita crear una aplicación cliente, que se llama LinkedApp. El servicio de cliente LinkedIn permite la recolección de datos.

La aplicación cliente LinkedApp realiza una solicitud HTTP al servicio cliente LinkedIn, este envía una respuesta en formato JSON. Ahora los datos se encuentran dentro de la aplicación cliente LinkedApp para permitir el consumo.

Para consumir estos datos se necesita obtener una autorización y crear una llave de aplicación, que son establecidas cuando se realiza la aplicación cliente.

El agente LinkedIn realiza la petición de información y envía la llave de aplicación a la aplicación cliente LinkedApp, está verifica si los datos son correctos y envía la aprobación del llave de acceso y la respuesta de la información en formato JSON.

El agente LinkedIn una vez que ha recibido los datos desde la aplicación cliente LinkedApp, crea un recurso REST que usa el método GET, para verificar que la sesión se encuentre activa con el agente Facebook, a su vez también crea un recurso REST, que usa el método GET para asociar la información proveniente de LinkedIn con la información que se ha almacenado de Facebook, el agente Facebook consulta la información en la ontología mediante la manipulación del api de Jena para obtener los datos.

Cuando se realizado la asociación de los datos, el agente LinkedIn cuenta con los datos en formato JSON y crea un recurso REST que usa el método GET para enviar los parámetros y almacenar los datos en la ontología. En el recurso se ejecutan procesos de gestión de información con la Api Jena que permite manipular ontologías.

Cuando la información se ha almacenado, se envía la respuesta del agente LinkedIn al agente Mailer, mediante la creación de un recurso REST que usa el método GET para que notifique al usuario que los datos han sido almacenados. A su vez el agente LinkedIn presenta los datos al usuario, en la interfaz de usuario mediante un esquema HTML.

La **tabla 10** muestra las tecnologías, especificaciones y versiones usadas para la construcción del agente.

Tabla 10: Especificaciones agente LinkedIn.

Tecnología	**Especificación**	**Versión**
Lenguaje de programación:	*JAVA	-
	*JavaScript	-
IDE utilizado:	NETBEANS	7.3.1
Bibliotecas:	JQuery	-
Api:	*Jena Api	2.6.4
	*LinkedIn Api	-

Fuente: La autora.

La **figura 87** muestra los parámetros necesarios para conseguir el acceso a la aplicación cliente LinkedApp desde el agente LinkedIn.

```
config: {
        api_key: XXXXXXXXXXX
        authorize: true
        credentials_cookie: true
        credentials_cookie_crc: true
        onLoad: onLinkedInLoad
  }
```

Figura 87: Parámetros para conseguir acceso a LinkedApp desde agente LinkedIn.

Fuente: La autora.

Agente Google

La **figura 88** muestra el diagrama de arquitectura del agente Google.

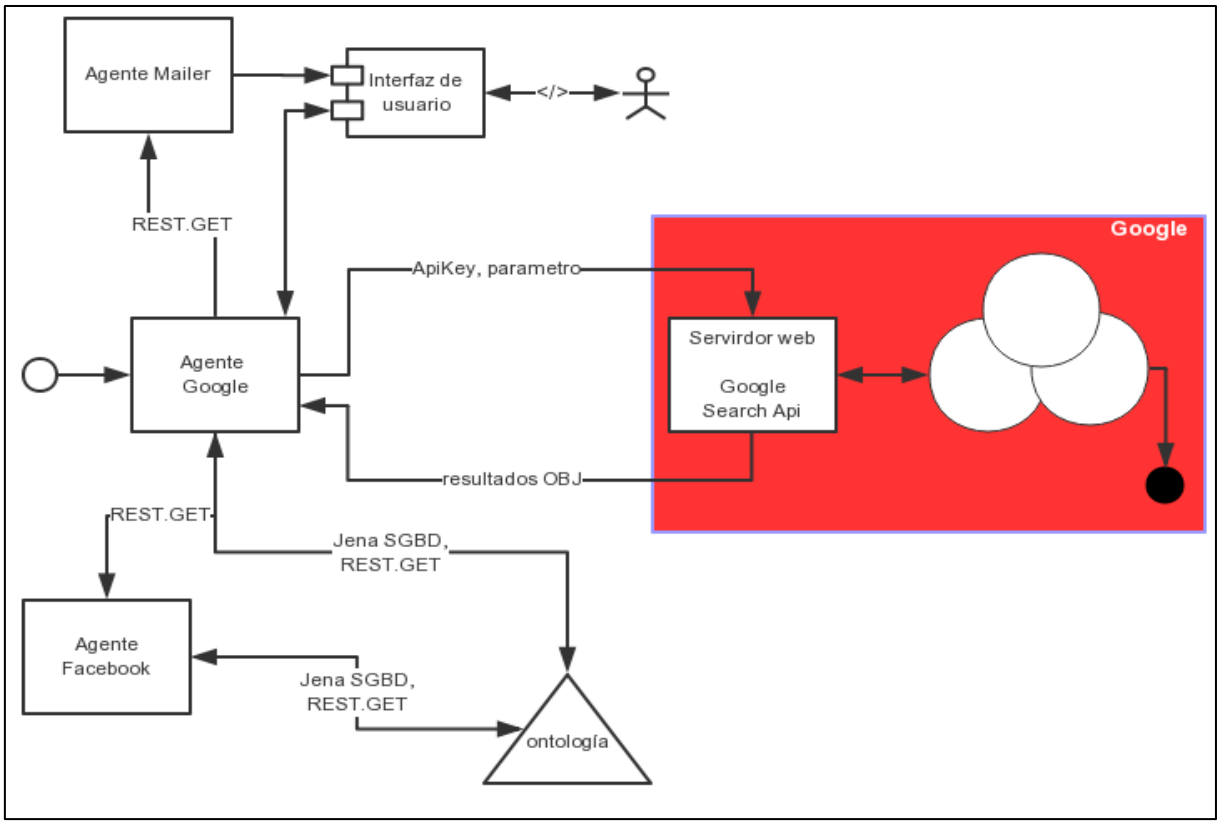


Figura 88: Diagrama de arquitectura agente Google.

Fuente: La autora.

En la **figura 88** se muestra el diagrama de arquitectura resultante para el agente Google.

Para conseguir el consumo de datos se necesita crear un acceso en Google Search Api, lo que da un api Key que es una llave de acceso al servicio web.

El agente Google envía un parámetro y la llave de acceso al servidor web de Google, este procesa la información y responde un objeto de datos.

El agente Google una vez que ha recibido los datos crea un recurso REST que usa el método GET, para verificar que la sesión se encuentre activa con el agente Facebook. El agente Google manipula este objeto de datos y presenta al usuario la información. El usuario podrá observar en la interfaz de usuario una lista de recomendaciones en 3 secciones; web, noticias e imágenes, de las cuales elige las que tengan relación con él y hace la selección.

El agente Facebook procesa esta selección de información y crea un recurso REST que usa el método GET para enviar los parámetros y almacenar los datos en la ontología. En el

recurso se ejecutan procesos de gestión de información con la Api Jena que permite manipular ontologías.

Cuando la información se ha almacenado, se envía la respuesta del agente Google al agente Mailer, mediante la creación de un recurso REST que usa el método GET para que notifique al usuario que los datos han sido almacenados. A su vez el agente Google presenta los datos al usuario, en la interfaz de usuario mediante un esquema HTML.

La **tabla 11** muestra las tecnologías, especificaciones y versiones usadas para la construcción del agente.

Tabla 11: Especificaciones agente Google.

Tecnología	**Especificación**	**Versión**
Lenguaje de programación:	*JAVA	-
	*JavaScript	-
IDE utilizado:	NETBEANS	7.3.1
Bibliotecas:	JQuery	-
Api:	*Jena Api	2.6.4
	*Google Web Search Api	-

Fuente: La autora.

La **figura 89** muestra los parámetros necesarios para conseguir el consumo del servicio web de Google.

```
config: {
 google.loader.ServiceBase = 'http://www.google.com/uds';
 google.loader.GoogleApisBase = 'http://ajax.googleapis.com/ajax';
 google.loader.ApiKey = 'XXXXXXXXXXX';
 google.loader.KeyVerified = true;
 google.loader.LoadFailure = false;
 google.loader.Secure = false;
 google.loader.GoogleLocale = 'www.google.com';
 google.loader.ClientLocation = null;
 google.loader.AdditionalParams = '';
 }
```

Figura 89: Parámetros para conseguir acceso a las búsquedas de Google.

Fuente: La autora.

Agente Encuesta

La **figura 90** muestra el diagrama de arquitectura del agente Encuesta.

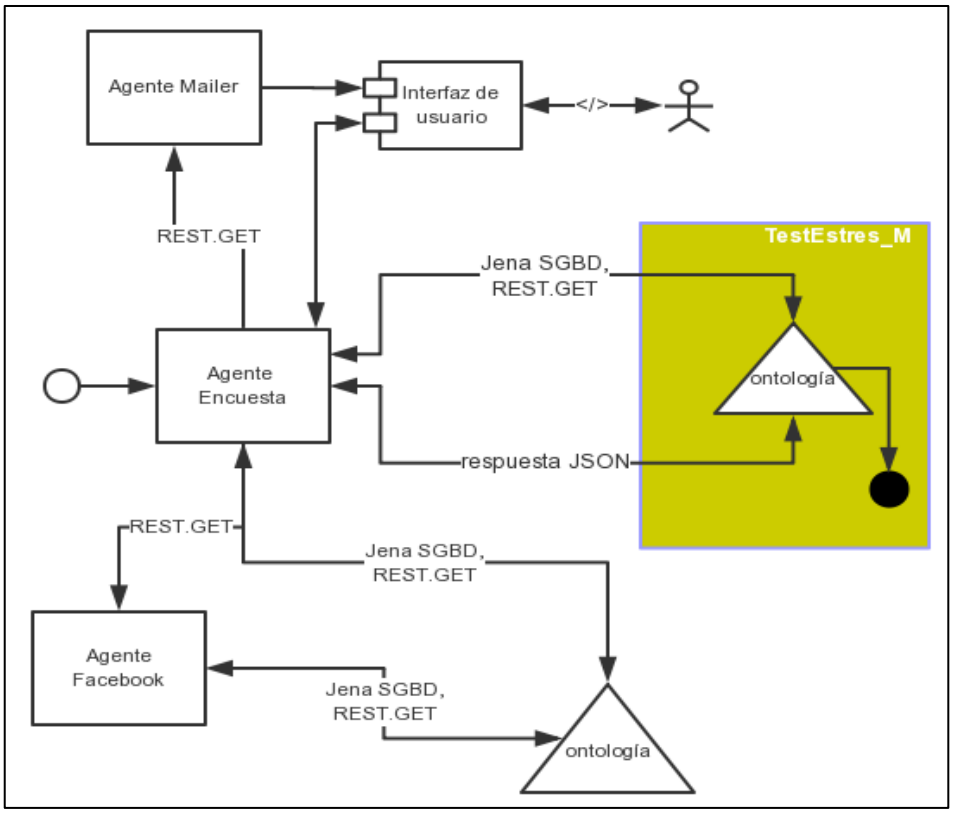


Figura 90: Diagrama de arquitectura agente Encuesta.

Fuente: La autora.

En la **figura 90** el agente Encuesta crea un recurso REST con el método GET para mediante la manipulación de datos de la ontología con Jena, buscar si el usuario ya ha realizado la encuesta SISCO, la respuesta del recurso se da en formato JSON. A su vez crea un recurso con el agente Facebook o una conversación para verificar la sesión de usuario.

Si es que no existe información de la Encuesta realizada, el agente Encuesta presenta una encuesta al usuario en la interfaz de usuario con un esquema HTML.

El usuario contesta las preguntas y el agente Encuesta procesa la información y crea un recurso REST que usa el método GET para enviar los parámetros y almacenar los datos en la ontología. En el recurso se ejecutan procesos de gestión de información con la Api Jena que permite manipular ontologías.

Cuando la información se ha almacenado, se envía la respuesta del agente Encuesta al agente Mailer, mediante la creación de un recurso REST que usa el método GET para que notifique al usuario que los datos han sido almacenados.

La **tabla 12** muestra las tecnologías, especificaciones y versiones usadas para la construcción del agente.

Tabla 12: Especificaciones agente Encuesta.

Tecnología	**Especificación**	**Versión**
Lenguaje de programación:	*JAVA	-
	*JavaScript	-
IDE utilizado:	NETBEANS	7.3.1
Bibliotecas:	JQuery	-
Api:	*Jena Api	2.6.4

Fuente: La autora.

Agente Mailer

La **figura 91** muestra el diagrama de arquitectura del agente Mailer.

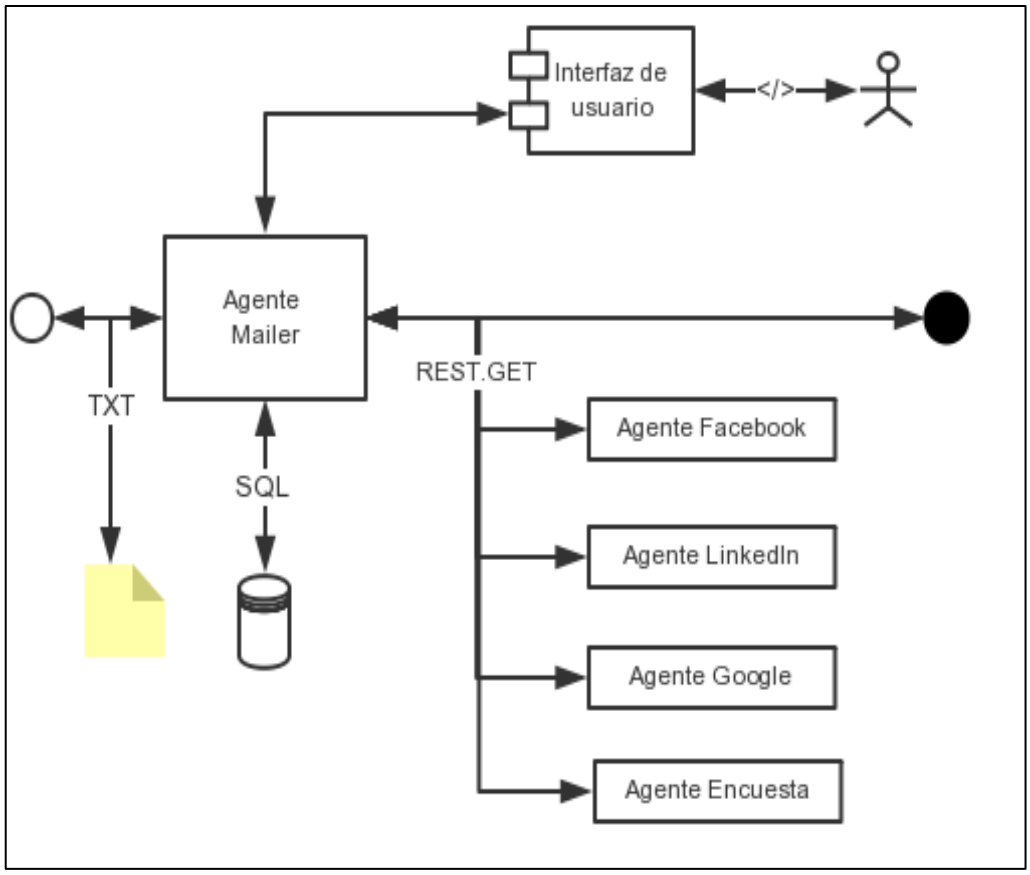


Figura 91: Diagrama de arquitectura agente Mailer.

Fuente: La autora.

El agente Mailer realiza la consulta a un archivo de texto para verificar si es que existen nuevos usuarios, si esto sucede, el agente Mailer procesa la información en una base de datos mediante una sentencia SQL para llevar un registro de los usuarios notificados y no volver a enviarles invitaciones.

Además, el agente Mailer mediante la creación de un recurso REST el método GET es el encargado de realizar la notificación al usuario de los datos que los agentes están recopilando.

Finalmente el agente Mailer mediante la creación de un recurso REST el método GET notifica al usuario vía mail, la información que todos los agentes obtuvieron.

La **tabla 13** muestra las tecnologías, especificaciones y versiones usadas para la construcción del agente.

Tabla 13: Especificaciones agente Mailer

Tecnología	**Especificación**	**Versión**
Lenguaje de programación:	*JAVA	-
	*JavaScript	-
	*PHP	-
IDE utilizado:	NETBEANS	7.3.1
Bibliotecas:	JQuery	-
Api:	*Jena Api	2.6.4

Fuente: La autora.

3.2.4 *Diseño final del sistema.*

Se muestra el diagrama de despliegue final.

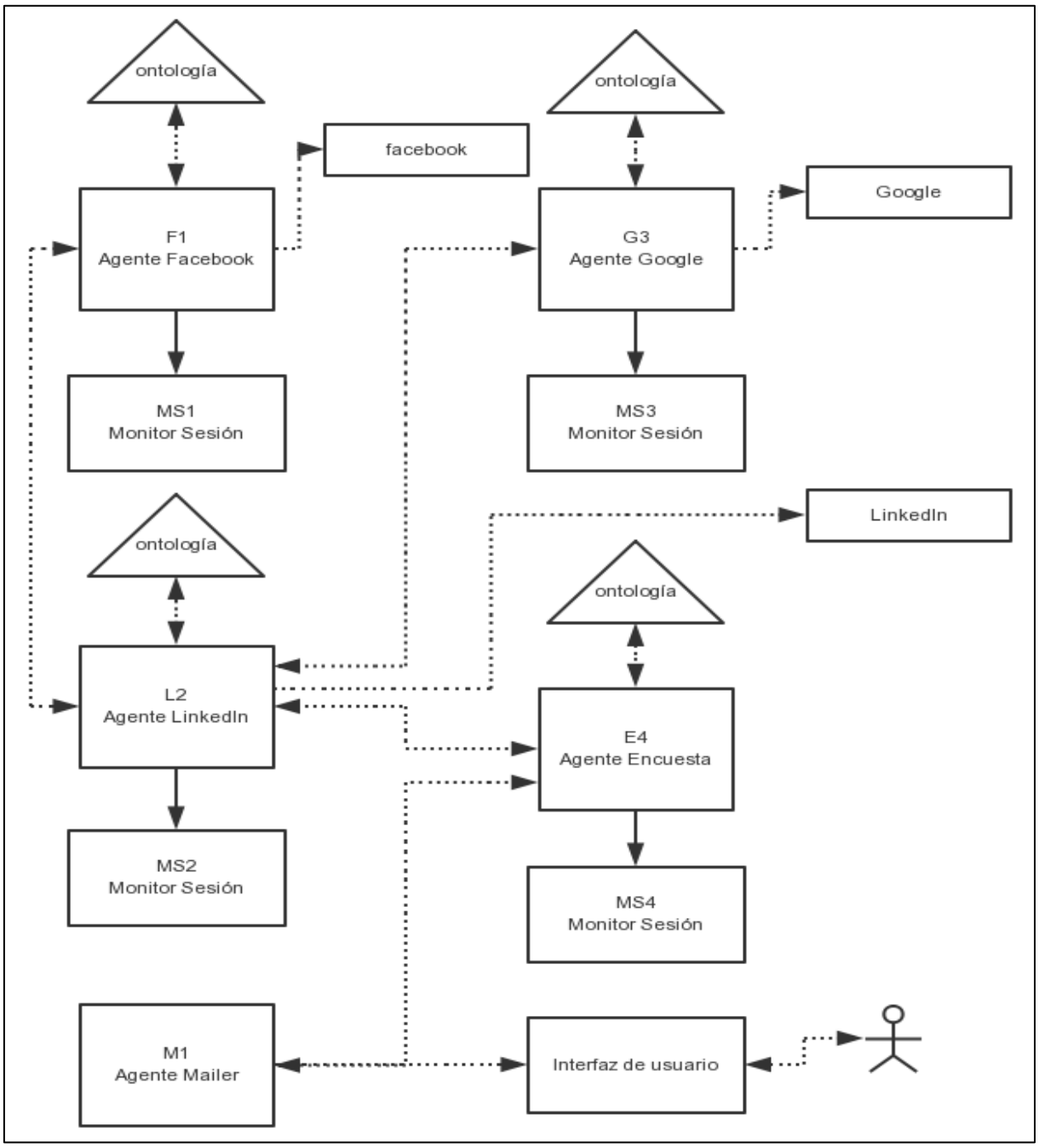


Figura 92: Diagrama de despliegue aplicación final.

Fuente: La autora.

En la **figura 92** se observa que los agentes no se encuentran geográficamente separados. El agente Facebook, LinkedIn y Google se conectan a recursos en la web, Facebook, LinkedIn y Google respectivamente. Todos los agentes menos en agente Mailer cumplen funciones de monitoreo de sesión.

Además, todos los agentes se conectan a una ontología para la gestión de la información.

3.2.5 *Interfaz de usuario.*

Agente Facebook

Figura 93: Interfaz de usuario agente Facebook.

Fuente: La autora.

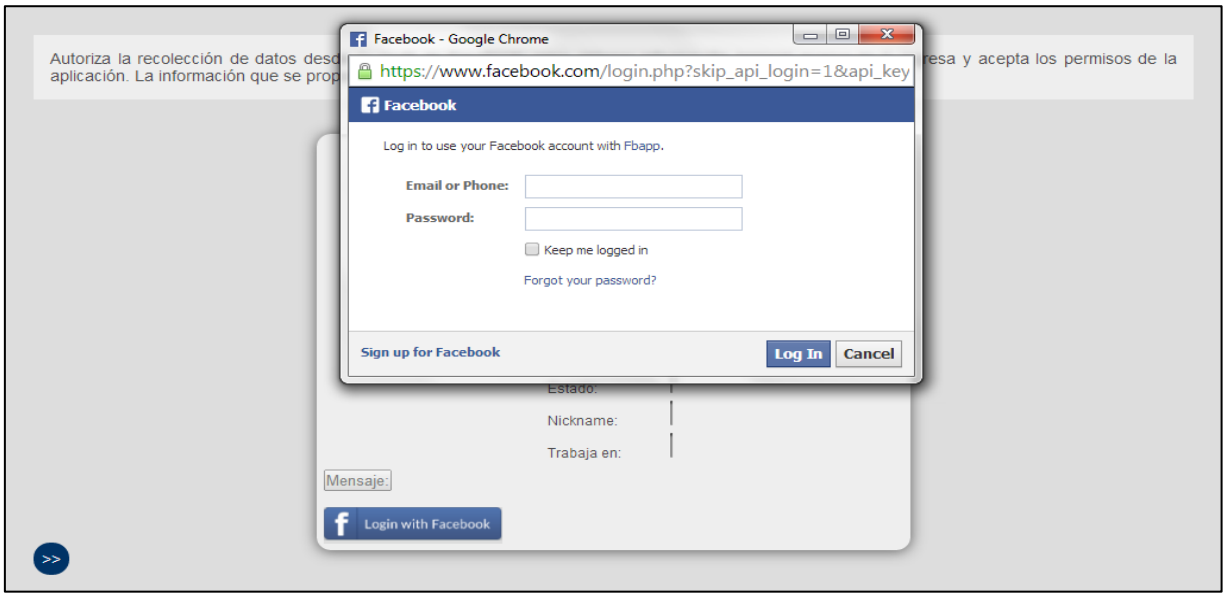


Figura 94: Interfaz de usuario agente Facebook: autorización de recopilación.

Fuente: La autora.

Figura 95: Interfaz de usuario agente Facebook: autorización incorrecta.

Fuente: La autora.

Figura 96: Interfaz de usuario agente Facebook: autorización correcta –recopilación de información.

Fuente: La autora.

Agente LinkedIn

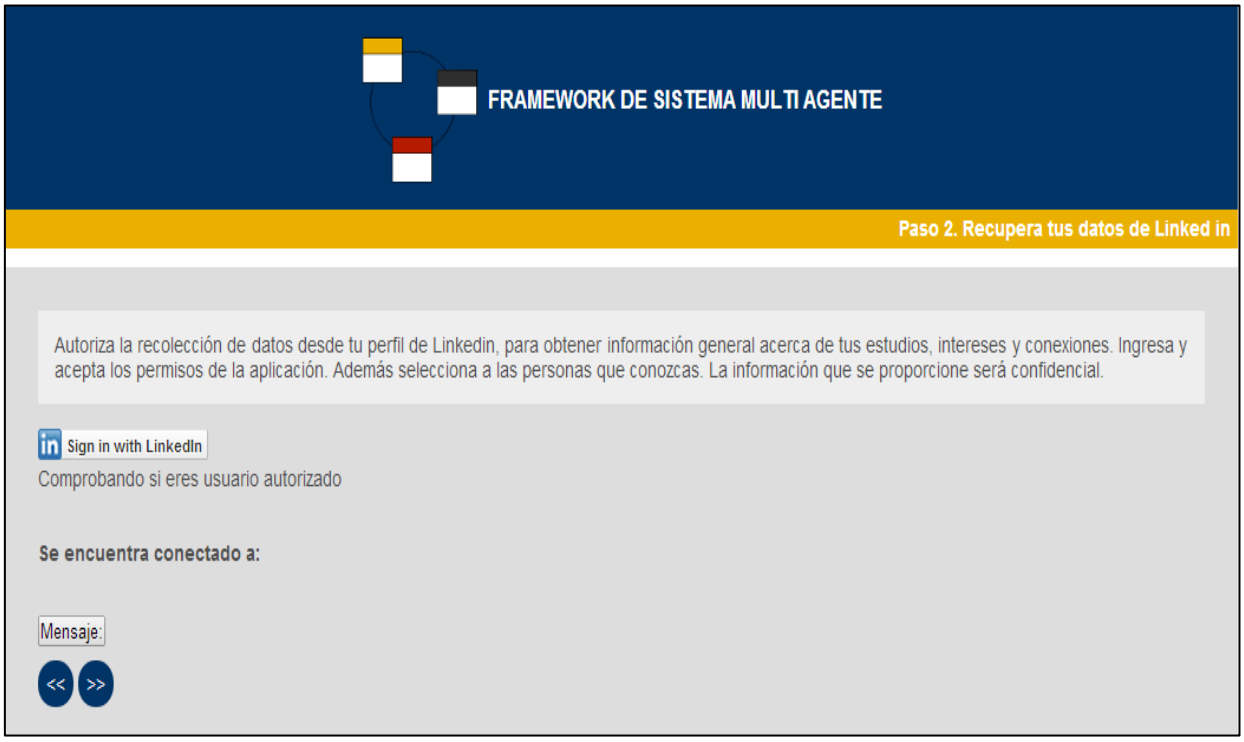


Figura 97: Interfaz de usuario de agente LinkedIn.

Fuente: La autora.

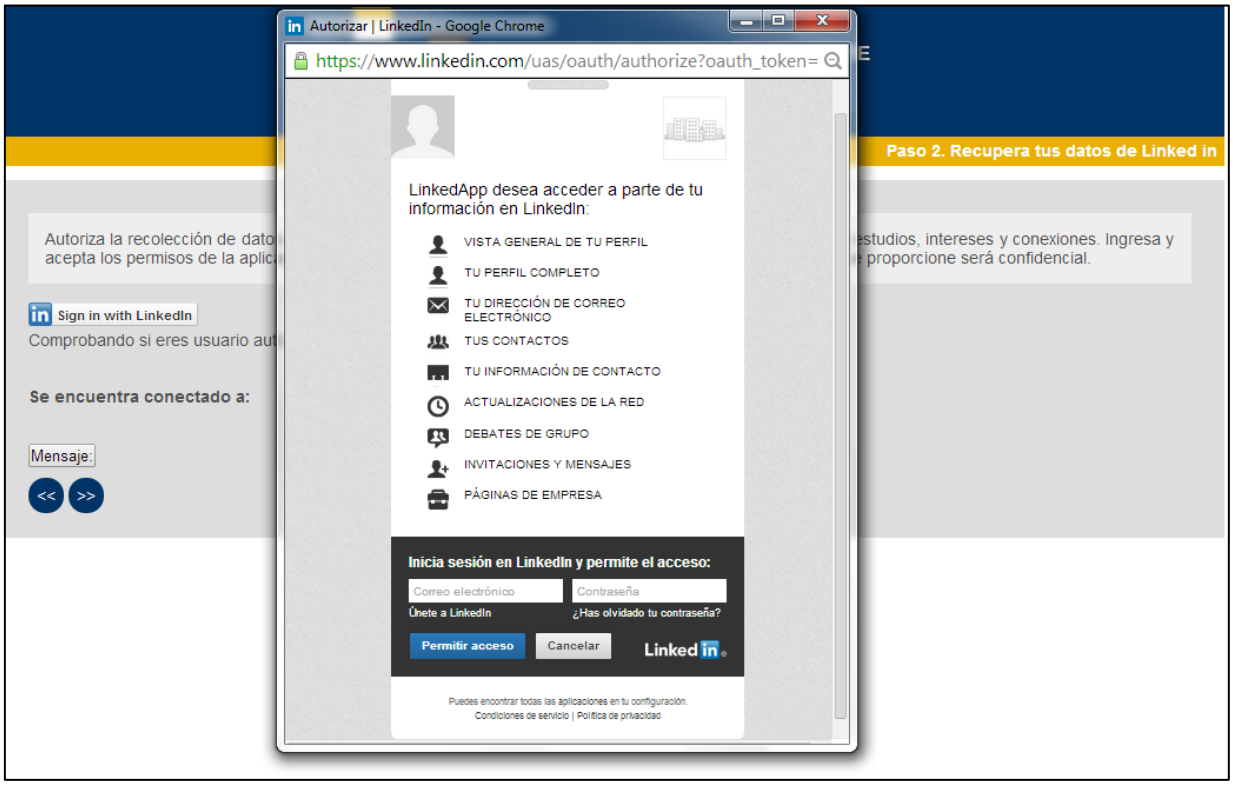


Figura 98: Interfaz de usuario de agente LinkedIn – pedido de autorización.

Fuente: La autora.

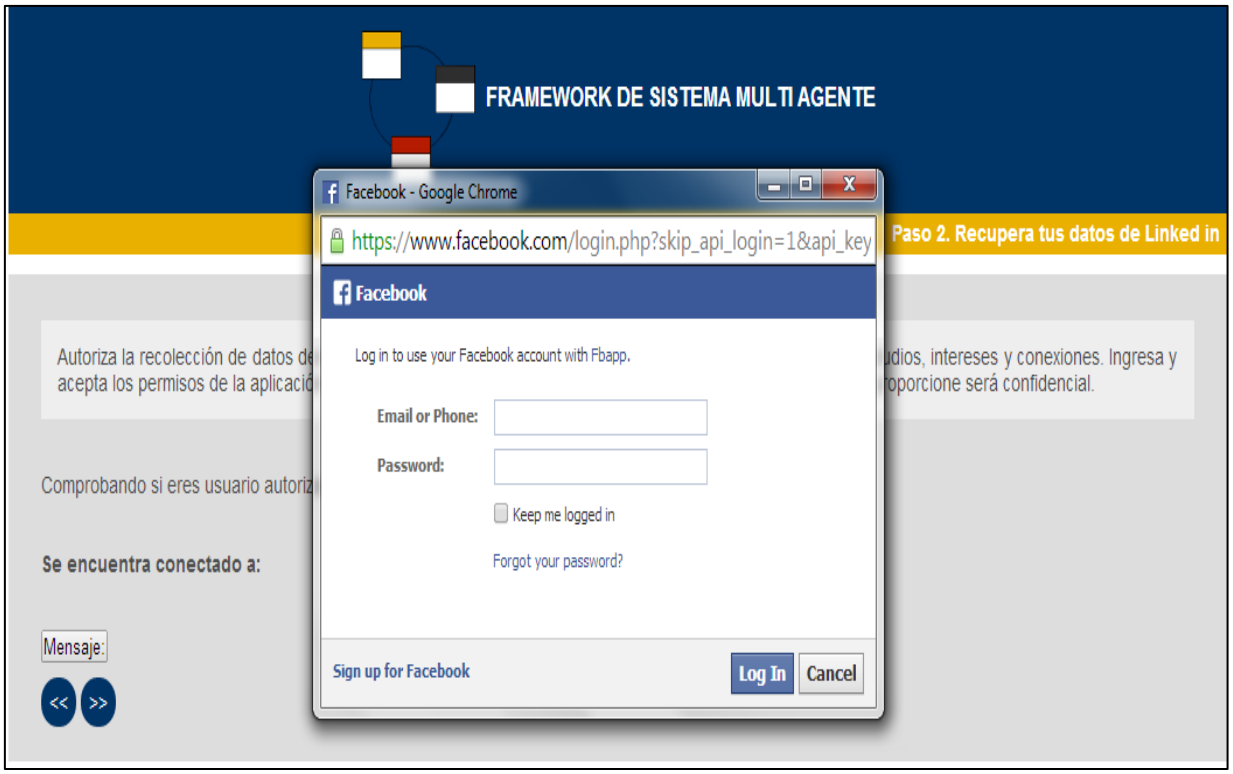


Figura 99: Interfaz de usuario de agente LinkedIn – control de sesiones.

Fuente: La autora.

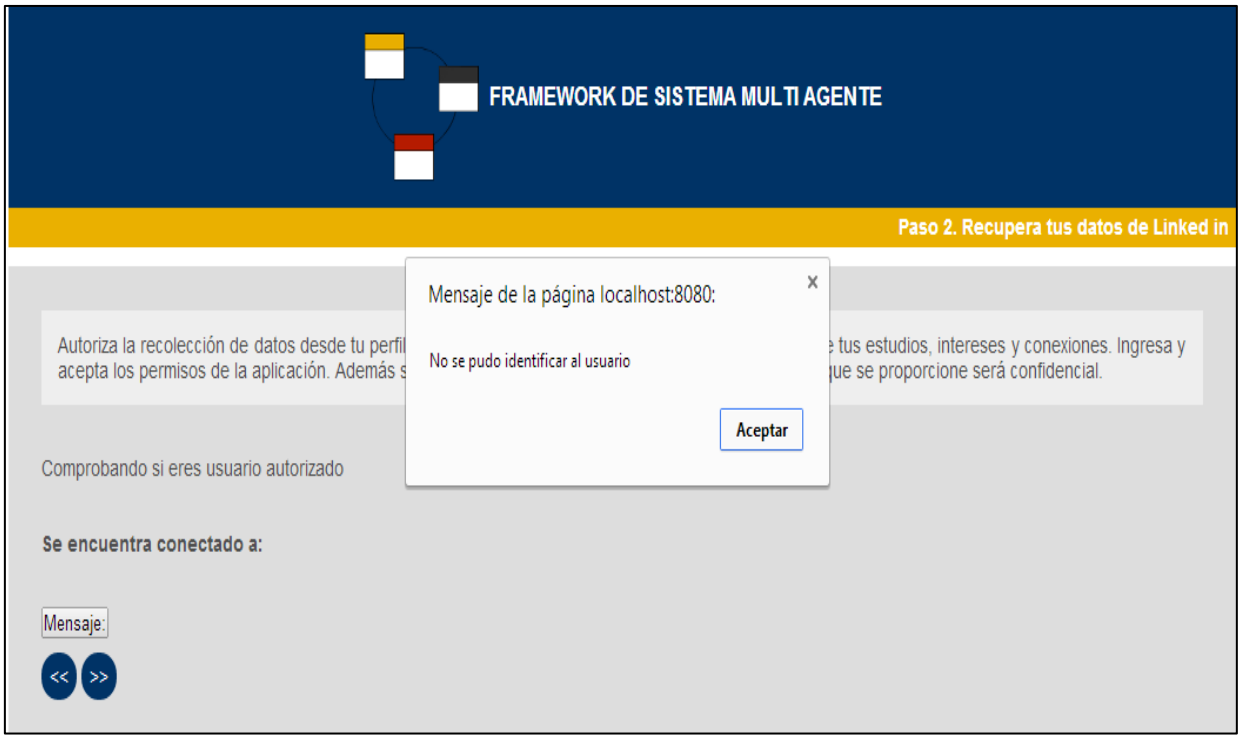


Figura 100: Interfaz de usuario de agente LinkedIn – sesión incorrecta.

Fuente: La autora.

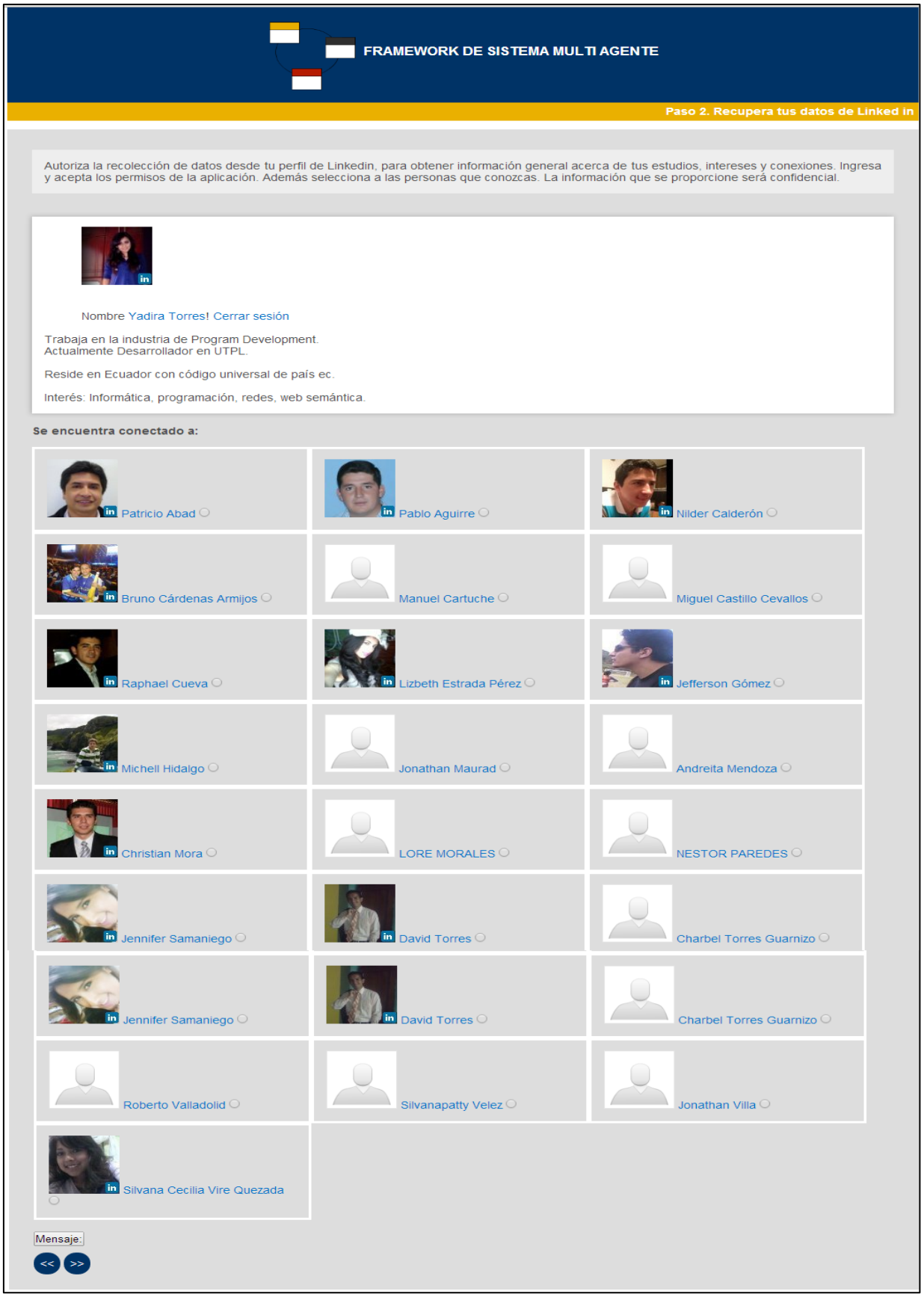


Figura 101: Interfaz de usuario de agente LinkedIn – recopilación de datos.

Fuente: La autora.

Agente Google

Figura 102: Interfaz de usuario de agente Google.

Fuente: La autora.

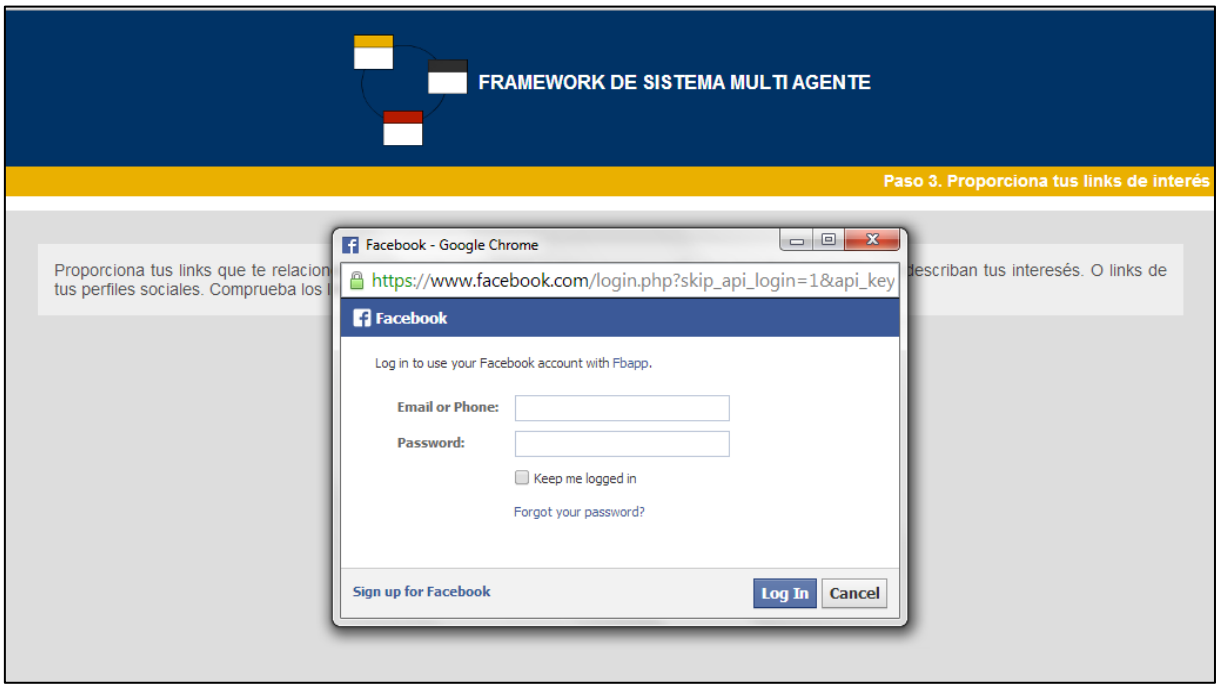


Figura 103: Interfaz de usuario de agente Google – control de sesiones.

Fuente: La autora.

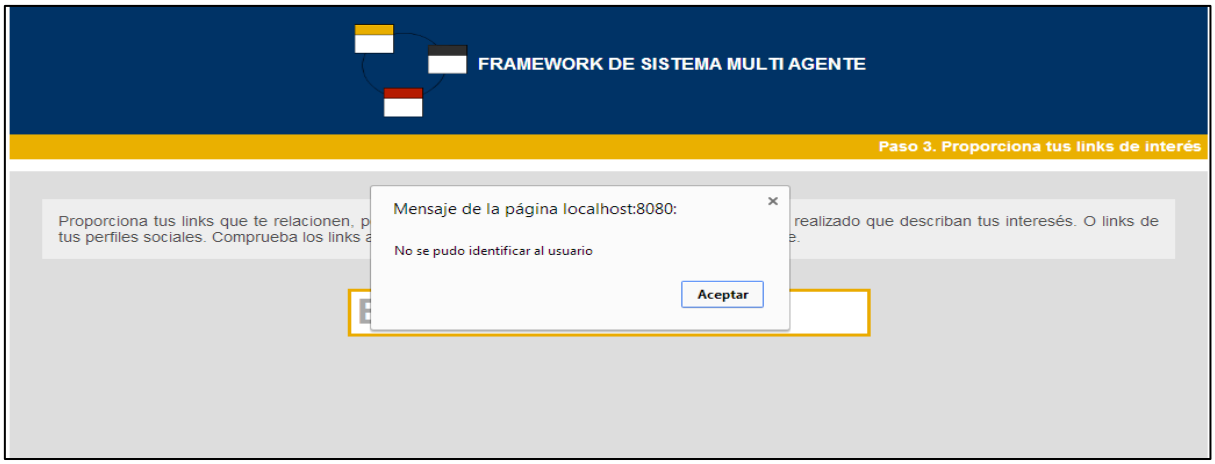


Figura 104: Interfaz de usuario de agente Google – sesión incorrecta.

Fuente: La autora.

Figura 105: Interfaz de usuario de agente Google – filtrado de parámetros, eliminación de campos no encontrados.

Fuente: La autora.

Figura 106: Interfaz de usuario de agente Google – recopilación de datos.

Fuente: La autora.

Agente Encuesta

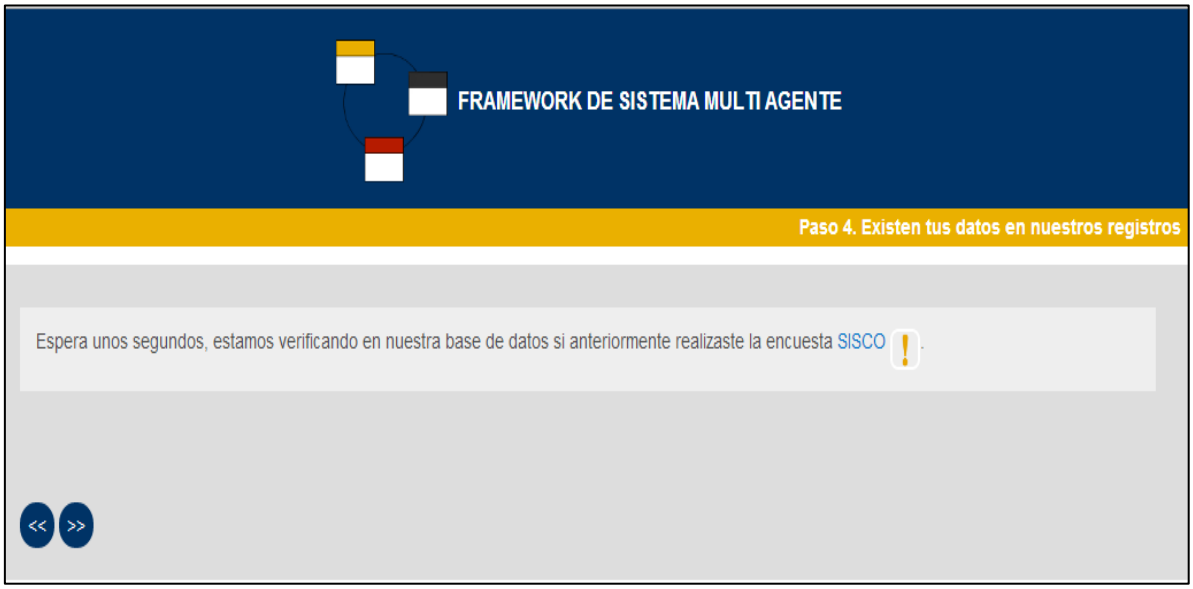


Figura 107: Interfaz de usuario de agente Encuesta- búsqueda de datos.

Fuente: La autora.

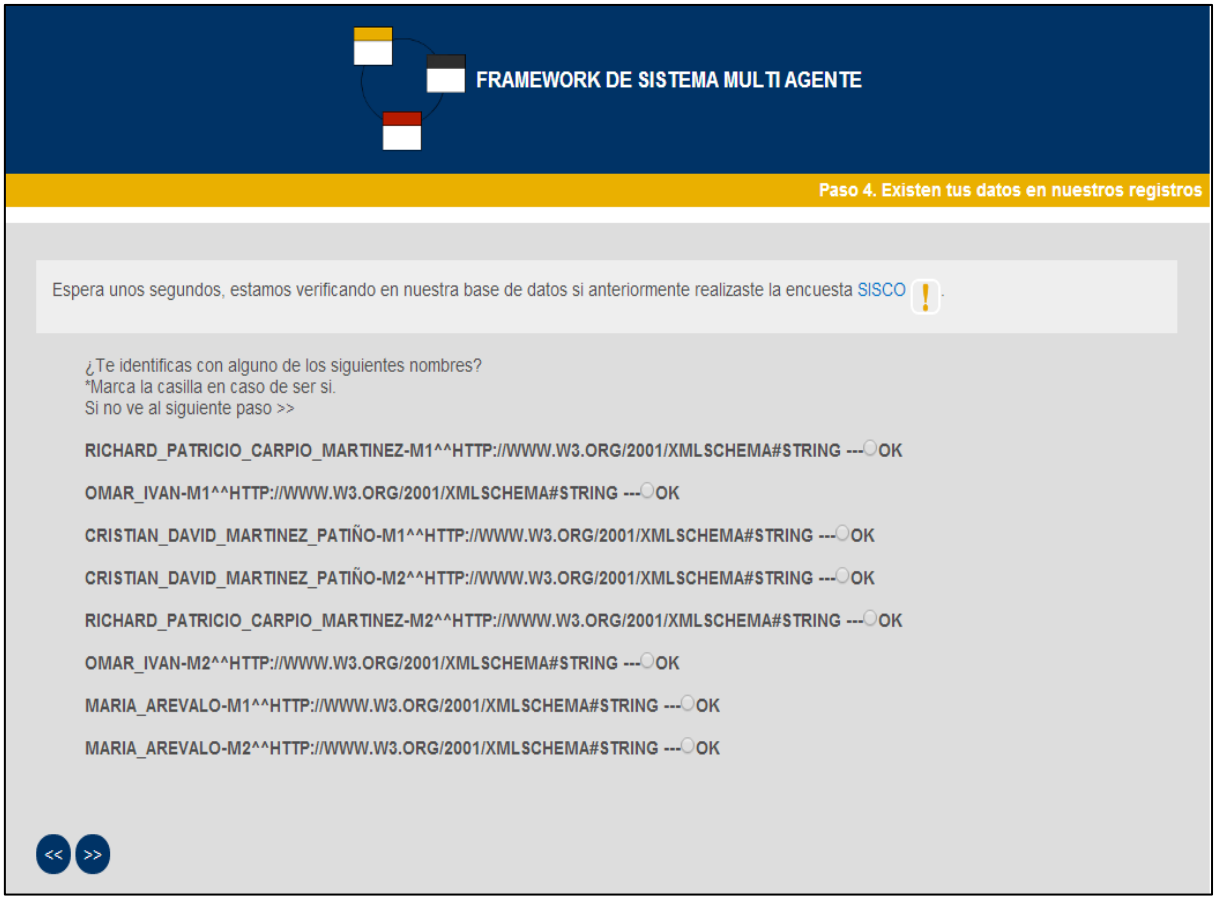


Figura 108: Interfaz de usuario de agente Encuesta – respuesta datos encontrados.

Fuente: La autora.

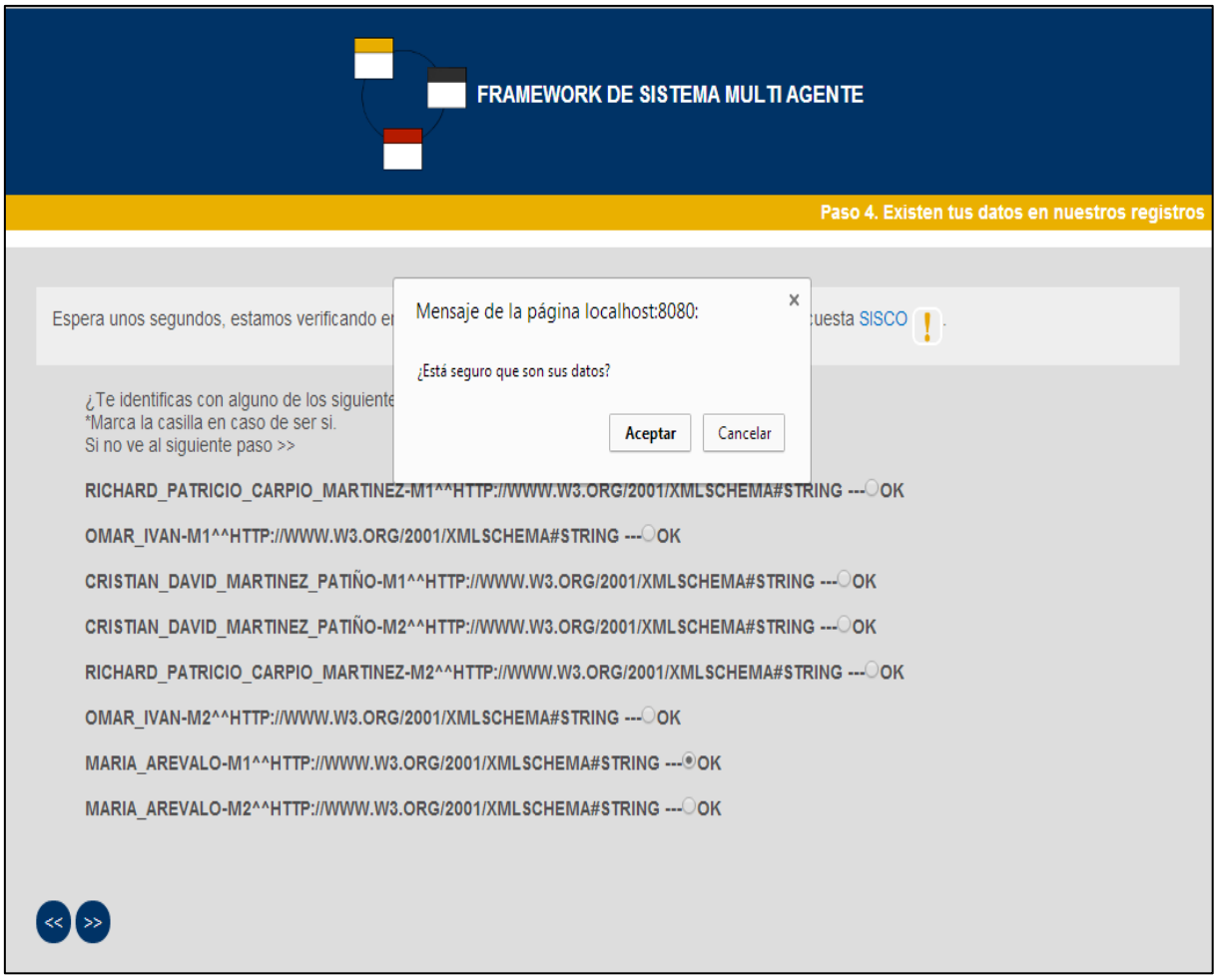


Figura 109: Interfaz de usuario de agente Encuesta – encontrando una coincidencia.

Fuente: La autora.

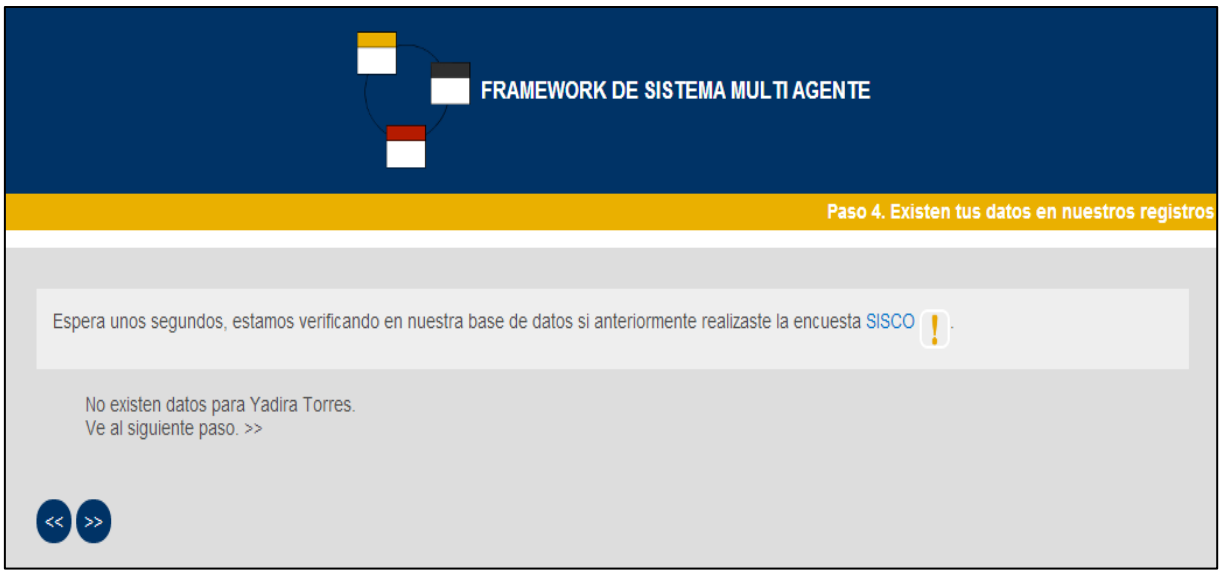


Figura 110: Interfaz de usuario de agente Encuesta – respuesta datos no encontrados.

Fuente: La autora.

Figura 111: Interfaz de usuario de agente Encuesta – aplicando encuesta.

Fuente: La autora.

Figura 112: Interfaz de usuario de agente Encuesta – respuesta no.

Fuente: La autora.

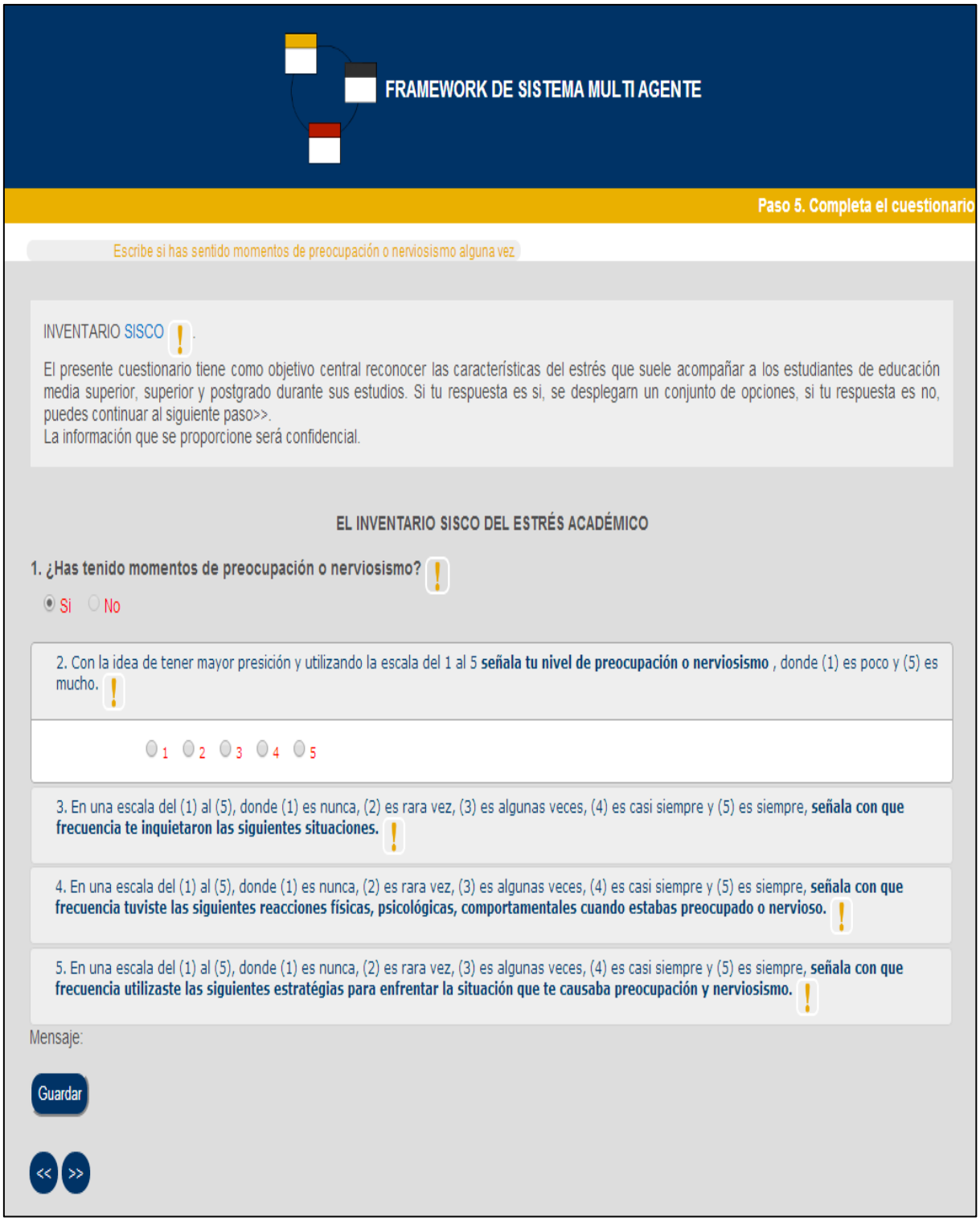


Figura 113: Interfaz de usuario de agente Encuesta – respuesta si.

Fuente: La autora.

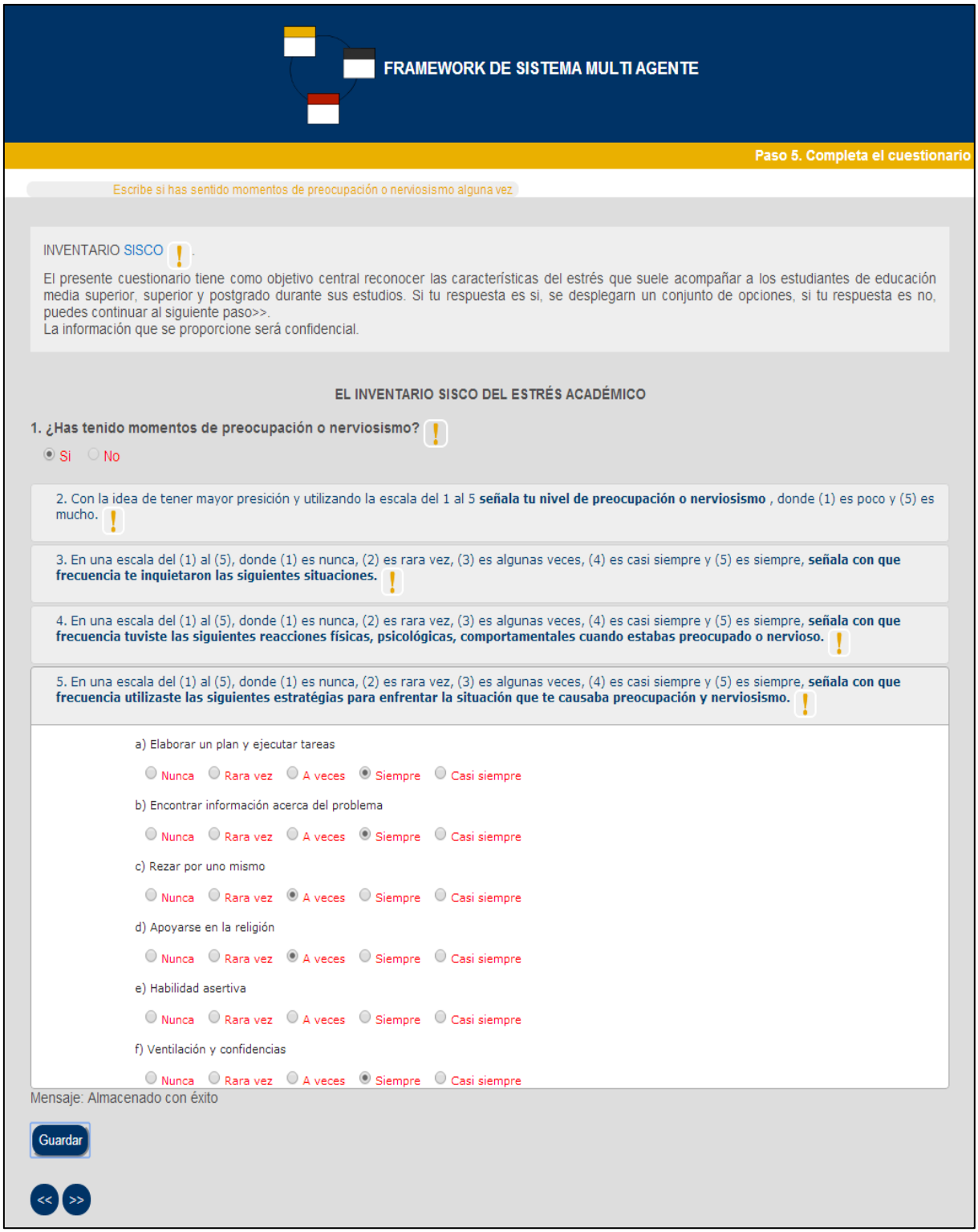


Figura 114: Interfaz de usuario de agente Encuesta – respuesta si, selección de opciones y almacenamiento de información.

Fuente: La autora.

Agente Mailer

Figura 115: Interfaz de usuario de agente Google.

Fuente: La autora.

Figura 116: Interfaz de usuario de agente Mailer – buscador de personas.

Fuente: La autora.

Figura 117: Interfaz de usuario de agente Mailer – buscador de personas con un parámetro.

Fuente: La autora.

Figura 118: Interfaz de usuario de agente Mailer – buscador de personas consultando más información.

Fuente: La autora.

VALIDACIÓN Y PRUEBAS

4. Parámetros para las pruebas.

El objetivo de este capítulo es realizar pruebas para verificar la validación de la aplicación sometida a diversos ambientes.

Se consideran dos parámetros a evaluar:

1. **Verificación**: demuestra que la aplicación realizada y sus componentes, satisfacen los requisitos especificados.
2. **Validación**: demuestra que la aplicación realizada y sus componentes, satisfacen el uso para el cual fueron creados. Comprueban la calidad de la aplicación. Se realiza la evaluación de la aplicación mediante:
    1. Análisis de la usabilidad
    2. Análisis del rendimiento
    3. Análisis de la concurrencia

La **tabla 14** muestra las características del servidor de pruebas que se utilizó.

Tabla 14: Características del servidor de prueba

Características	
Tipo:	Portátil
Marca:	Lenovo
Procesador:	Intel Core I7
Memoria RAM:	4GB
Frecuencia	1.8GHz

Fuente: La autora.

- **Alcance**

En cuanto a verificación:

Las pruebas se enfocan en verificar si se cumplen los casos de uso planteados en el desarrollo de cada agente. Además si los casos de uso ofrecen las salidas y mensajes esperados.

En cuanto a validación:

Además de realizar la validación del mismo, evaluación mediante; el análisis de la utilidad, rendimiento y concurrencia.

4.1 Verificación

Los puntos aplicados en este apartado verifican que, la aplicación:

Según (Drake & López, 2009):

- Cumple con los casos de uso propuestos por cada agente y además:
    - Ofrece las salidas esperadas a los casos de uso.
    - Ofrece los mensajes esperados a las salidas de los casos de uso.

Tabla 15: Lista de verificación a la aplicación final.

Lista de verificación						
Caso de uso	**Verificación**		**Salida esperada**	**Mensaje esperado**	**Verificación**	
	si	**no**			**si**	**no**
Agente Facebook						
Iniciar una sesión en Facebook por medio del agente Facebook.	x		Iniciar sesión	-	x	
			No iniciar sesión	No se pudo identificar al usuario	x	
Recopilar información de Facebook y almacenarla en la ontología.	x		Guardar información	Almacenado con éxito	x	
			No guardar información	No se pudo almacenar la información	x	
Crear una sesión de usuario con los datos de Facebook.	x		Crear sesión	-	x	
			No crear sesión	No se pudo identificar al usuario	x	
Monitorear que los otros agentes mantengan activa las sesiones de usuario.	x		Sesión usuario	-		x
			No sesión usuario	No se pudo identificar al usuario		x
Agente LinkedIn						
Iniciar una sesión en Linkedin por medio del agente Linkedin.	x		Iniciar sesión	-	x	
			No iniciar sesión	No se pudo identificar al usuario	x	
Recopilar la información de LinkedIn y almacenarla en la ontología.	x		Guardar información	Almacenado con éxito		x
			No guardar información	No se pudo almacenar la información	x	
Agente Google						
Recopilar información de Google por medio del agente Google.	x		-	-	x	
Guardar los datos seleccionados en la ontología.	x		Guardar información	Almacenado con éxito		x

			No guardar información	No se pudo almacenar la información	x	
Agente Encuesta						
Recopilar información de la encuesta SISCO y almacenar en la ontología.	x		Recopilar información	Está seguro de sus datos	x	
			No se encuentra información	No se encuentra información siga al siguiente paso	x	
Agente Mailer						
Realizar notificaciones periódicas al usuario para que usen el SMA.	x		Nuevo usuario	Notificado con éxito	x	
			Nuevo usuario	No se pudo enviar el mail	x	
Presentar la información al usuario.	x		-	-	x	
Buscar en la ontología información de los datos recopilados.	x		Presentar información	Almacenado con éxito		x
			Presentar información	No se pudo almacenar la información	x	

Fuente: La autora.

La primera verificación de la aplicación demostró que el agente Facebook no estaba realizando de forma adecuada el control de sesiones en los otros agentes, por lo tanto el Agente LinkedIn, Google, Mailer y Encuesta no realizan de forma adecuada la asociación de información con los datos que recopila el agente Facebook, ya que no puede asociar la información a un persona en común.

Para solucionar este inconviente se añadió una condición a todos los agentes, que se puede observar en la **figura 119**.

```
fb.ready(function() {
        if (fb.logged)
        {
//sentencia
        } else {
 alert("Necesita mantener su sesión activa de Facebook. Vuelva a loguearse.");
            login();
        }
    });
```

Figura 119: Condicion para controlar que la sesión de Facebook se encuentre activa.

Fuente: La autora.

La **tabla 16** muestra la lista de verificación final.

Tabla 16: Lista de verificación de la aplicación final.

Lista de verificación						
Caso de uso	**Verificación**		**Salida esperada**	**Mensaje esperado**	**Verificación**	
	si	**no**			**si**	**no**
Agente Facebook						
Iniciar una sesión en Facebook por medio del agente Facebook.	x		Iniciar sesión	-	x	
			No iniciar sesión	No se pudo identificar al usuario	x	
Recopilar información de Facebook y almacenarla en la ontología.	x		Guardar información	Almacenado con éxito	x	
			No guardar información	No se pudo almacenar la información	x	
Crear una sesión de usuario con los datos de Facebook.	x		Crear sesión	-	x	
			No crear sesión	No se pudo identificar al usuario	x	
Monitorear que los otros agentes mantengan activa las sesiones de usuario.	x		Sesión usuario	-	x	
			No sesión usuario	No se pudo identificar al usuario	x	
Agente LinkedIn						
Iniciar una sesión en Linkedin por medio del agente Linkedin.	x		Iniciar sesión	-	x	
			No iniciar sesión	No se pudo identificar al usuario	x	
Recopilar la información de LinkedIn y almacenarla en la ontología.	x		Guardar información	Almacenado con éxito	x	
			No guardar información	No se pudo almacenar la información	x	
Agente Google						
Recopilar información de Google por medio del agente Google.	x		-	-	x	
Guardar los datos seleccionados en la ontología.	x		Guardar información	Almacenado con éxito	x	
			No guardar información	No se pudo almacenar la información	x	
Agente Encuesta						
Recopilar información de la encuesta SISCO y almacenar en la ontología.	x		Recopilar información	Está seguro de sus datos	x	
			No se encuentra información	No se encuentra información siga al siguiente paso	x	
Agente Mailer						

Realizar notificaciones periódicas al usuario para que usen el SMA.	x		Nuevo usuario	Notificado con éxito	x	
			Nuevo usuario	No se pudo enviar el mail	x	
Presentar la información al usuario.	x		-	-	x	
Buscar en la ontología información de los datos recopilados.	x		Presentar información	Almacenado con éxito	x	
			Presentar información	No se pudo almacenar la información	x	

Fuente: La autora.

Una vez realizada la verificación final de los casos de uso, se observa han sido ejecutados con éxito.

4.2 Validación

Para la evaluación de la aplicación, se realizó:

- Análisis de la usabilidad
- Análisis del rendimiento
- Análisis de concurrencia

Análisis de la usabilidad

Para realizar la validación de la usabilidad, se usa el método de entrevistas de usuario en donde se expone una serie de preguntas y respuestas para evaluar atributos de usabilidad como:

Según (Vega, Rodríguez, & Justo, 2009):

- Privacidad, el usuario debe confiar en que sus datos personales se usan de forma adecuada.
- Facilidad de aprendizaje, está dado por la facilidad con que los usuarios puedan ejecutar las tareas planteadas en la aplicación.

Prueba 1.

Herramienta: Encuesta

Muestra: 30 personas.

Objetivo: Medir el porcentaje de confianza de los usuarios hacia la aplicación, basado en la notificación del uso de la aplicación vía correo electrónico.

Resultados:

- Se notificó a 30 usuarios a que usen la aplicación, la notificación se realizó mediante *el agente Mailer* vía correo electrónico, **solo 22 aceptaron el uso de la aplicación**, la **tabla 17** explica el grado de confianza obtenido.

Tabla 17: Análisis de la usabilidad: factor privacidad.

Invitaciones			
# invitaciones	**Esperadas**	**Reales**	**% confianza en uso**
30	30	22	73,33%

Fuente: La autora.

- Consecuentemente se les invito a llenar el cuestionario de experiencia de usuario, para este punto (privacidad) se concentra en las preguntas:

Pregunta 1

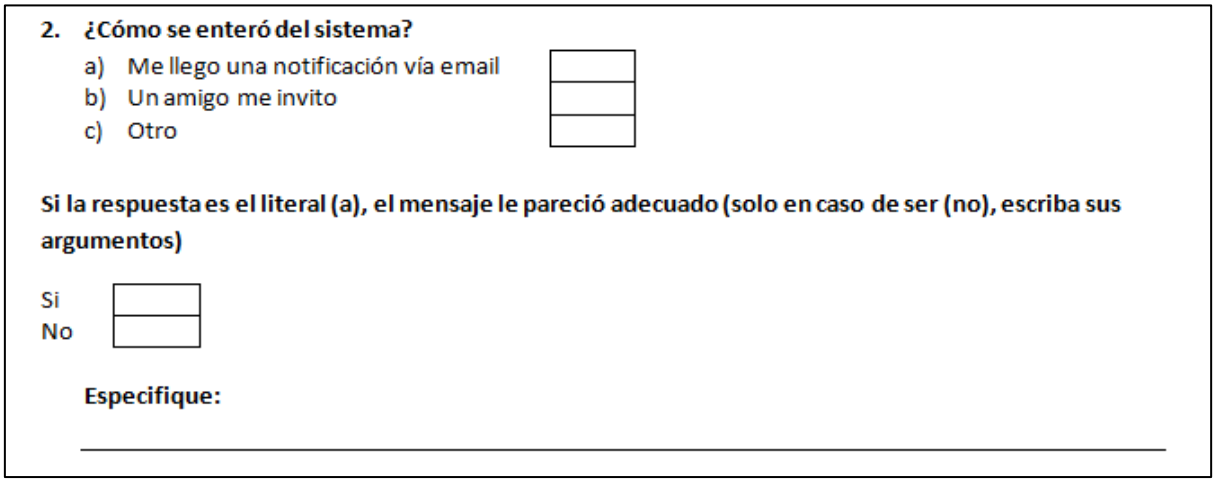


Figura 120: Validación privacidad pregunta 1.

Fuente: La autora.

Los 30 usuarios coinciden en la notificación de la invitación vía mail. El 26.77% argumenta que actualmente muchas aplicaciones manejan mensajes considerados Spam para invitar a usar aplicaciones, por lo que no confían en aceptar el uso de la aplicación vía email, a pesar de que el mensaje de invitación les pareció adecuado.

El 73.33% restante creyó conveniente y confiable el mensaje de invitación.

Pregunta 2

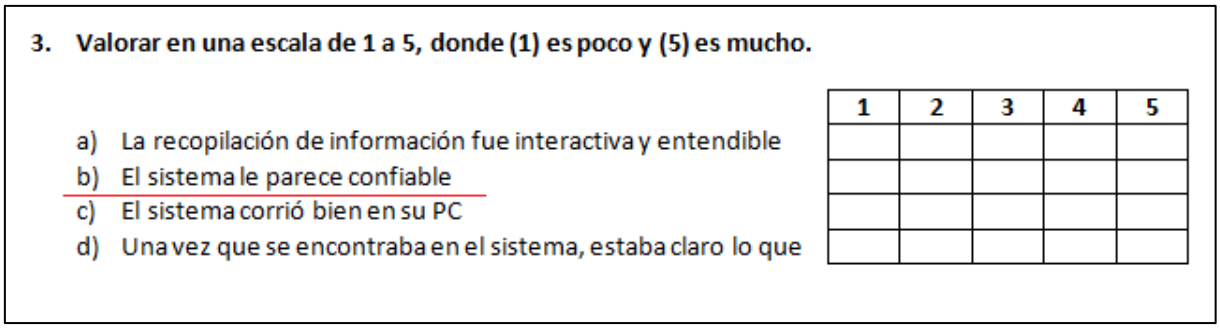


Figura 121: Validación privacidad pregunta 2.

Fuente: La autora.

Los 30 usuarios contestaron la pregunta, y el apartado b. 15 seleccionaron 5, 7 seleccionaron 4, que son los que participaron del proceso y se considera un puntaje aceptable. Entre los 8, dos seleccionaron 2, y 6 seleccionaron 3, por los argumentos antes expuestos en la pregunta 2.

Prueba 2.

Herramienta: Encuesta

Muestra: 30 personas.

Objetivo: Medir la facilidad de aprendizaje que tiene la aplicación para los usuarios.

Resultados:

- Se notificó a 30 usuarios a que usen la aplicación, la notificación se realizó mediante *el agente Mailer* vía correo electrónico. Consecuentemente se les invito a llenar el cuestionario de experiencia de usuario, para este punto (facilidad de aprendizaje) nos concentramos en las preguntas:

Pregunta 1

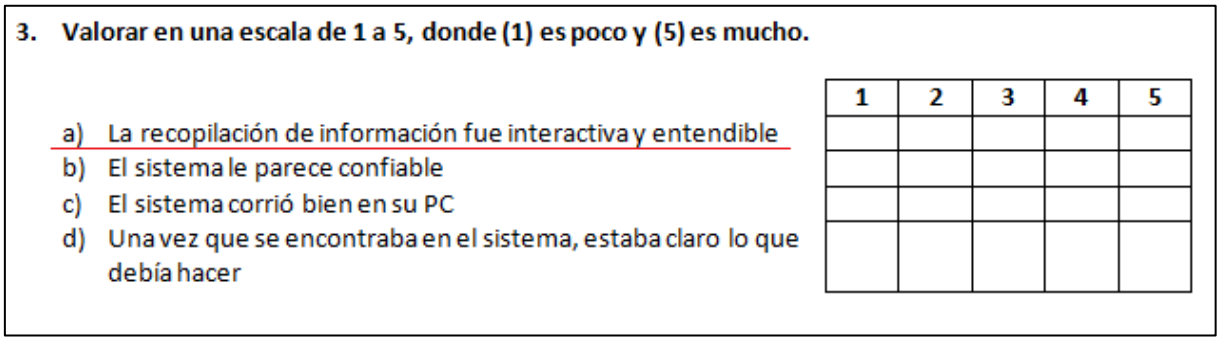
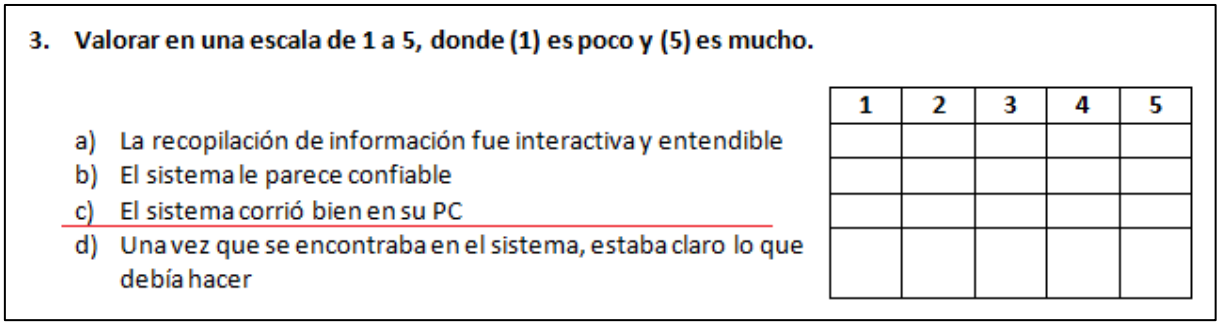


Figura 122: Validación facilidad de aprendizaje pregunta 1.

Fuente: La autora.

24 usuarios seleccionaron 5, 4 usuarios seleccionaron 4 y 2 usuarios seleccionaron 3. Se considera entre 4 y 5 un nivel alto de aceptación, lo que da un porcentaje de 93.3% de facilidad de aprendizaje encontrado en la pregunta 1, apartado a.

Pregunta 2



3. **Valorar en una escala de 1 a 5, donde (1) es poco y (5) es mucho.**

		1	2	3	4	5
a)	La recopilación de información fue interactiva y entendible					
b)	El sistema le parece confiable					
c)	El sistema corrió bien en su PC					
d)	Una vez que se encontraba en el sistema, estaba claro lo que debía hacer					

Figura 123: Validación facilidad de aprendizaje pregunta 2.

Fuente: La autora.

25 usuarios seleccionaron 5, 2 seleccionaron 4, 2 selecciono 3 y 1 selecciono 2. Se considera entre 4 y 5 un nivel alto de aceptación, lo que da un porcentaje de 90% de facilidad de aprendizaje encontrado en la pregunta 2, apartado c.

Se obtiene un promedio final de 91.65% de facilidad de aprendizaje para la aplicación.

Análisis del rendimiento

"Son necesarias para saber que la aplicación procesa la carga esperada" (Sommerville, 2006). Las herramientas usadas analizan el rendimiento de nuestra aplicación considerando factores, como tiempo de carga, velocidad de acceso, velocidad de respuesta. La puntuación va de 0 a 100 puntos. Cuanta más alta sea la puntuación, mejor. Por ejemplo, una puntuación de 85 indicaría que el rendimiento de la página es bueno.

Las herramientas usadas son las siguientes:

- Yahoo Slow
- Google Page Speed
- Web Page Test

Prueba 1.

Herramienta: Yahoo Slow, Google Page Speed, Web Page Test.

Objetivo: Medir el rendimiento de la aplicación, basado en parámetros de tiempo de carga, velocidad de acceso y velocidad de respuesta.

Resultados:

- Se midió el rendimiento de la aplicación con 3 herramientas, que nos ofrecían un puntaje sobre 100.

Tabla 18: Análisis del rendimiento.

Rendimiento	
Herramienta	**/ 100**
Yahoo Slow	86
Google Page Speed	85
Web Page Test	85

Fuente: La autora.

Yahoo Slow muestra un puntaje de 86/100, Google Page Speed muestra un puntaje de 85/100, y Web Page Test un puntaje de 85/100. El promedio obtenido es 85/100 en cuanto a rendimiento. Una puntuación de 85/100 indica que el rendimiento de la página es bueno.

Análisis de concurrencia

Creadas para comprobar que el sistema puede soportar un volumen de carga aceptable, y ver la capacidad de concurrencia realizadas varias peticiones. “La utilización de una herramienta puede ser útil para mejorar y agilizar las pruebas a realizar, por ejemplo, a la hora de simular una cantidad determinada de usuarios accediendo a un sitio al mismo tiempo, o la carga de datos en un sistema, etc”. **(Whittaker, 2000)**

La herramienta usada es la siguiente:

- Load Impact

Es un servicio web de pago para realizar pruebas de carga mediante visitas concurrentes. Tiene una opción gratuita con la que podemos lanzar una prueba con hasta 50 usuarios simultáneos y es la que se usa para evaluar la concurrencia de la aplicación.

Prueba 1.

Herramienta: Load Impact.

Muestra: Concurrencia entre 2 y 50 usuarios virtuales.

Objetivo: Medir la concurrencia de la aplicación sometida a diversos accesos.

Resultados:

- Se realizó la prueba con la herramienta Load Impact, que permite un acceso de hasta 50 usuarios concurrentes.
- El resultado del análisis se lo obtiene en la **tabla 19,** que se observa a continuación.

Tabla 19: Análisis de la concurrencia.

Usuarios virtuales activos	Conexiones activas (Número de conexiones TCP activas)	Ancho de banda (Rendimiento actual del sistema)	Datos recibidos (Número total de bytes recibidos durante la prueba)	Requerimientos (Número actual de requerimientos)	Requerimientos/s (requerimientos por segundo)	Tiempo
2	8	2497.3Kbit/s	919.42KiB	38	13	9.87s
4	16	1821.1Kbit/s	6.32MiB	248	9	9.98s
6	24	2192.8Kbit/s	9.48MiB	372	9	10.01s
10	40	1068.1Kbit/s	19.33MiB	767	8	11.98s
13	55	4020.3Kbit/s	30.25MiB	1191	20	8.86s
17	69	5.1Mbit/s	50.78MiB	2002	26	9.62s
20	81	7.5Mbit/s	65.97MiB	2598	36	9.34s
24	96	11.7Mbit/s	95.71MiB	3745	48	9.99s
28	113	10.3Mbit/s	122.63MiB	4825	48	11.01s
32	131	13.9Mbit/s	162.29MiB	6367	58	10.03s
37	152	17.9Mbit/s	213.18MiB	8368	79	10.02s
39	154	13.6Mbit/s	229.53MiB	9026	65	11.35s
40	165	18.7Mbit/s	236.34MiB	9276	80	9.54s
43	170	12.6Mbit/s	277.57MiB	10881	53	10.05s
46	192	24.6Mbit/s	308.76MiB	12106	106	10.15s
48	197	16Mbit/s	335.31MiB	13160	68	10.21s
49	196	16.9Mbit/s	350.71MiB	13757	76	10.91s
50	201	16.2Mbit/s	356.64MiB	14006	79	10.18s

Fuente: La autora.

En la **tabla 19** se observa el comportamiento concurrente de acuerdo a diversos factores:

- Conexiones TCP activas
- Ancho de banda
- Datos recibidos
- Requerimientos totales
- Requerimientos que se cumplen por segundo
- Y los segundos que toman en cumplir los requerimientos.

La siguiente gráfica es el resultante entre usuarios activos (derecha), tiempo de carga (izquierda), tiempo de la prueba (inferior):

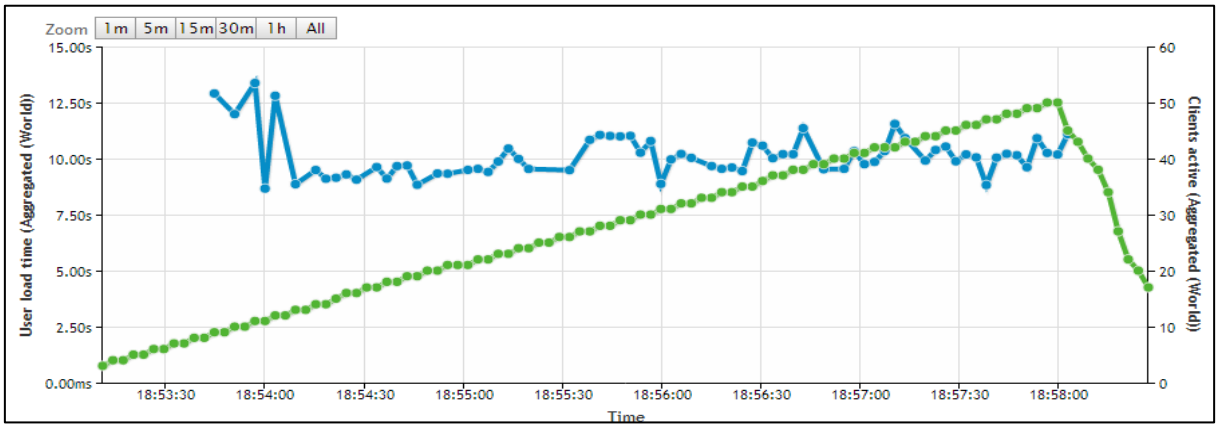


Figura 124: Validación tiempo de carga vs tiempo de prueba.

Fuente: La autora.

Ahora se evalua la interacción de 7 usuarios virtuales:

- 10 usuarios: 11.98s

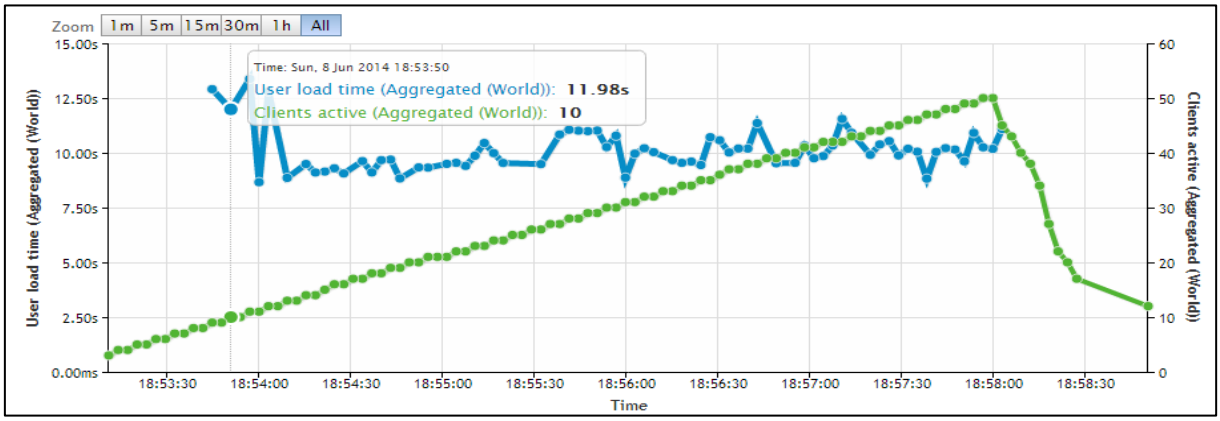


Figura 125: Validación concurrencia 7 usuarios.

Fuente: La autora.

- 20 usuarios: 9.34s

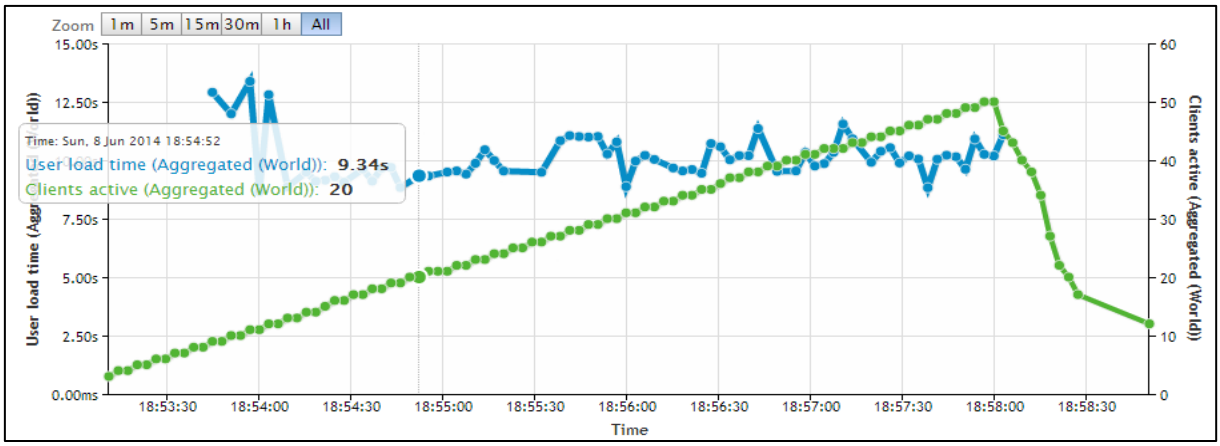


Figura 126: Validación concurrencia 20 usuarios.

Fuente: La autora.

- 30 usuarios: 10.26s

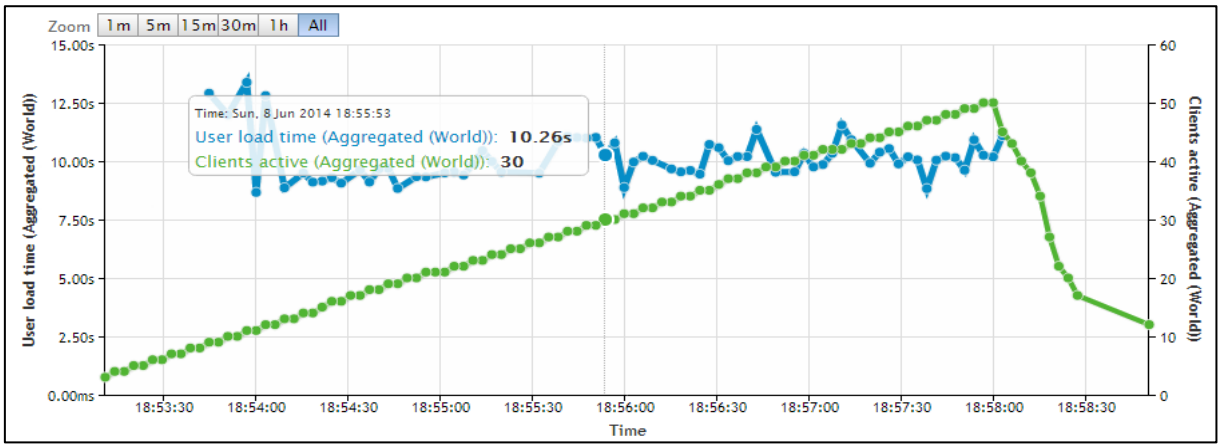


Figura 127: Validación concurrencia 30 usuarios.

Fuente: La autora.

- 34 usuarios: 9.60s

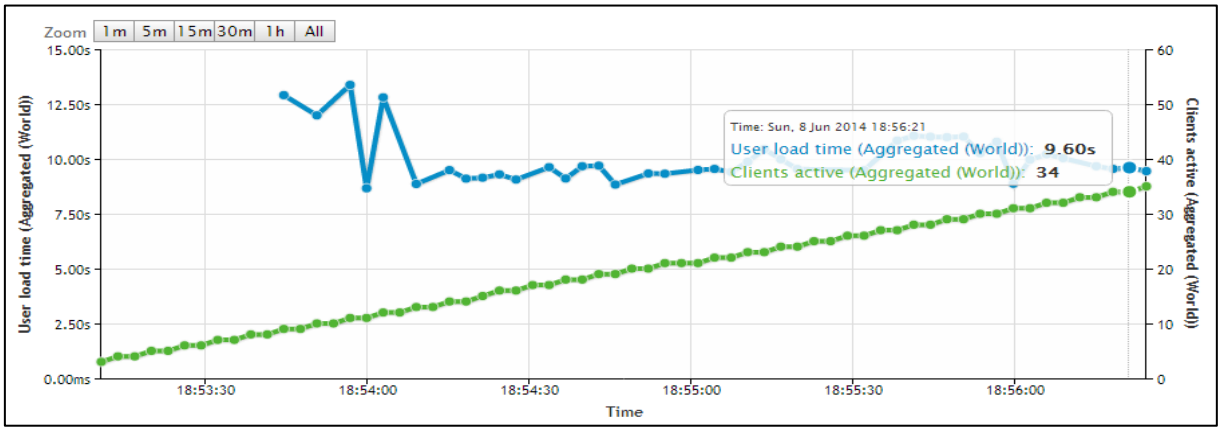


Figura 128: Validación concurrencia 34 usuarios.

Fuente: La autora.

- 40 usuarios:9.54s

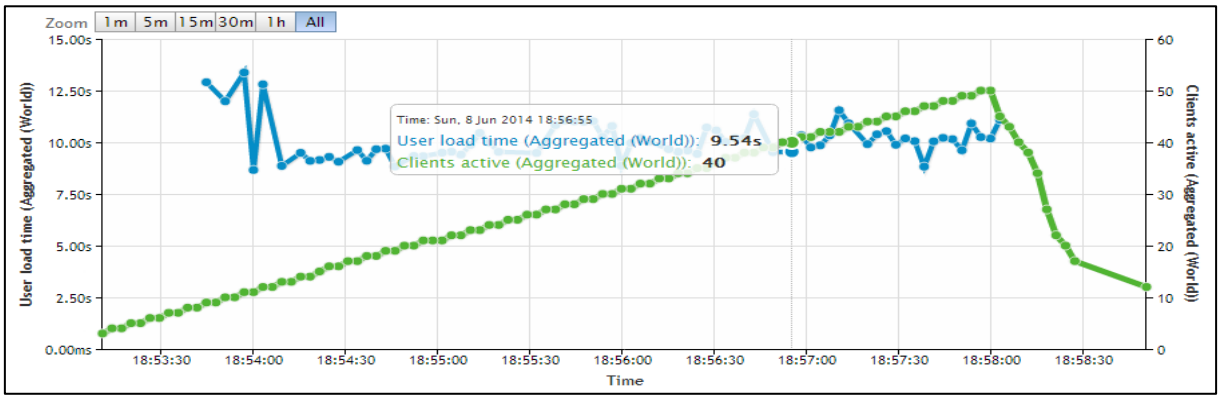


Figura 129: Validación concurrencia 40 usuarios.

Fuente: La autora.

- 45 usuarios:10.54s

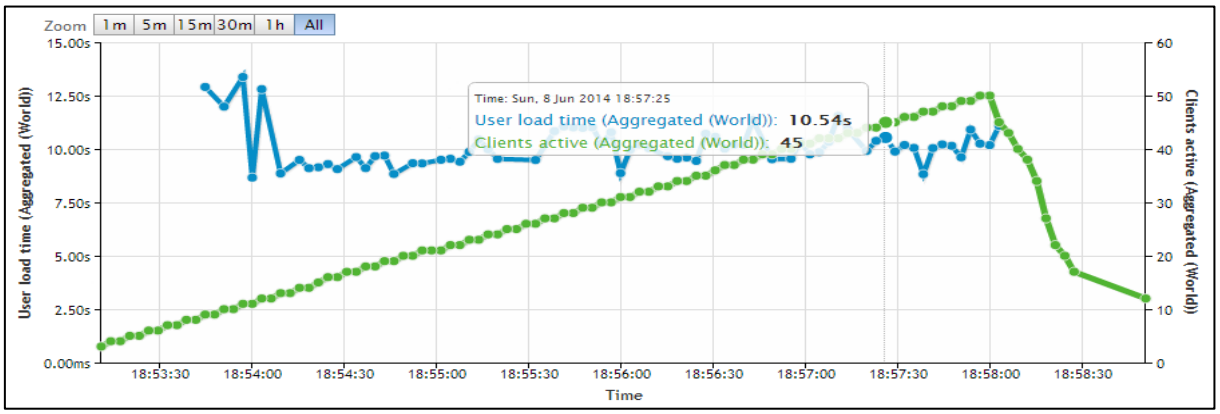


Figura 130: Validación concurrencia 45 usuarios.

Fuente: La autora.

- 50 usuarios: 10.18s

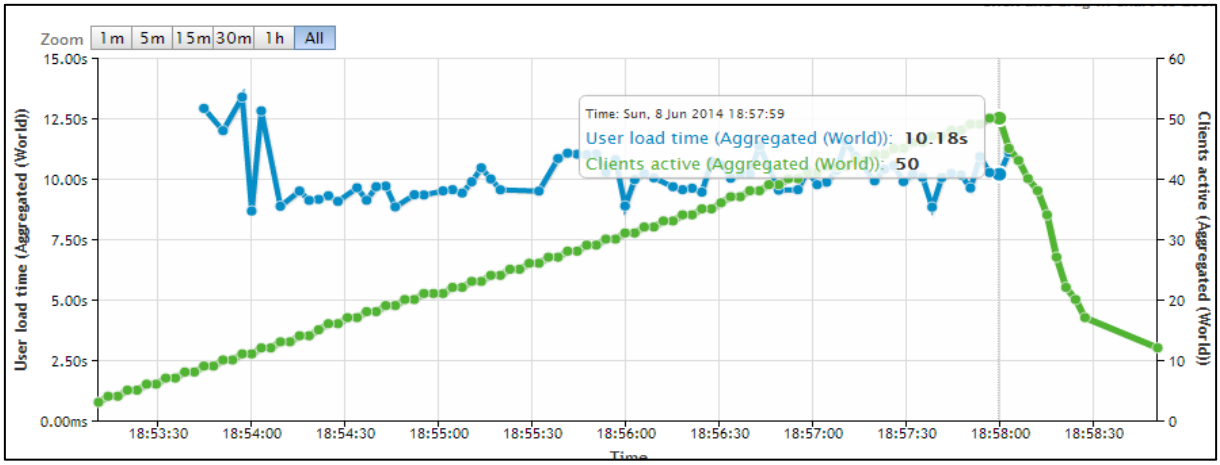


Figura 131: Validación concurrencia 50 usuarios.

Fuente: La autora.

Lo que da un promedio en el acceso concurrente de hasta 50 usuarios de:

11.98s + 9.34s + 10.26s + 9.60s + 9.54s + 10.54s + 10.18s = 10.20s

Considerado un buen resultado.

CONCLUSIONES

- La evaluación de los framework Bubble, Aglest, Jade y Jafmas permite conocer que la característica más importante de los agentes es la comunicación, debido a que los agentes necesitan un mecanismo adecuado de interacción para cumplir con los objetivos globales.
- La evaluación de los framework Bubble, Aglest, Jade y Jafmas guían a pensar en dos características para desarrollar framework de SMA; 1) el enfoque y 2) los componentes. El enfoque del framework desarrollado es en metodologías genéricas y con componentes MaSE para el análisis y diseño, y REST para la implementación.
- Se presenta el desarrollo de una aplicación para ayudar al mejor entendimiento del framework.
- Las metodologías para SMA tienen varias clasificaciones de acuerdo a áreas concretas, en el desarrollo del framework se realiza el reúso de MaSE, una metodología orientada a objetos, ya que la abstracción simplifica el proceso de programación.
- Los 3 niveles del modelo de implementación 1) parámetros, 2) instancias y 3) métodos, permiten tener una estructura jerarquizada para el cumplimiento de objetivos de los agentes.
- En este proyecto se hace uso de REST para la comunicación. Como resultado se ha obtiene concurrencia y mayor disponibilidad, ya que los agentes acceden, se comunican o conversan con otros agentes a través de recursos, de forma íntegra sin la necesidad de estados.
- La interacción de las personas con la red social Facebook hace que sea el primer medio de preferencia en la web, en la pruebas de la aplicación todos los usuarios contaban con una cuenta.
- En el agente Mailer, las notificaciones vía mail para el uso de la aplicación no han alcanzado la confianza necesaria debido al exceso de spam que se filtra en los correos.
- El agente Google se encarga de recopilar información en referencia a 1) web,2) noticias e 3) imágenes, para esta primera peculiaridad los datos que se recopilaron en las pruebas hacen más énfasis en los links de perfiles personales de diversas redes sociales que documentos científicos.

- En la aplicación, el agente Facebook es el encargado de realizar las instancias de sesión de usuario, debido a que; Facebook, el recurso web desde donde recopila información, es más usado en la actualidad a diferencia de LinkedIn.
- La recopilación de información en el agente Google es más eficiente cuando añadimos a más del nombre otros parámetros de búsqueda, debido a que agregar palabras claves realiza búsquedas más precisas.
- La validación mediante la herramienta Load Impact nos muestra que en un escenario de acceso concurrente de 50 usuarios, el tiempo para cumplir los requerimientos se mantiene entre 9 y 11 segundos, lo cual es un resultado óptimo.
- En cuanto al rendimiento promedio obtenido (85/100) mediante las herramientas Google Page Speed, Yahoo Slow y Web Page Test considerando varios puntos; velocidad de carga, velocidad de descarga, tiempo, archivos html usados, comunicación, es aceptable. Google Page Speed indica que con una puntuación de 85/100 el rendimiento de la página es bueno.

RECOMENDACIONES

- En el modelado de la aplicación, la fase de análisis es la etapa en donde se debe prestar mas atención, ya que si existen errores que se han pasado por alto y aparecen en la implementación de los agentes debemos regresar a la etapa de análisis a verificar todos los diagramas.
- Para diseñar un framework de SMA se debe analizar el dominio de aplicación donde se va a desenvolver, ya que las metodologías genéricas pueden resultar muy generales cuando se tiene dominios de aplicación de SMA específicos, por ejemplo: agentes móviles, agentes geográficos, etc.
- Las notificaciones via mail en este proyecto no alcanzaron la confianza necesaria debido al creciente SPAM, al realizar SMA que requieran notificaciones vía mail se puede pensar en opciones alternativas.

TRABAJOS FUTUROS

- A partir del UML presentando en la fase de análisis y diseño con MaSE, y los lineamientos propuestos para la construcción, diseñar un generador de código automático para la producción de los agentes.
- Implementar aplicaciones de diversos dominios para probar la usabilidad del framework cuando se somete a otros ambientes distintos de la web.
- En base a la metodología estudiada y teniendo como dominio de aplicación la recopilación automática de información desde recursos web, diseñar una arquitectura de sistemas multi agente para la recuperación de información de recursos en la web. Este paso vendría a ser una extensión de la metodología MaSE en la fase de Diseño, el paso *Ensamblaje de Agentes.*
- Extender la funcionalidad de los agentes de forma que se puedan realizar recopilaciones de información desde repositorios universitarios y complementar la información existente.

BIBLIOGRAFÍA

Gutierréz, F. (2008). Comprenda las técnicas de internet. (117).

Zunino, A. (Marzo de 2000). Brainstorm/J: Un framework para agentes inteligentes. *Universidad Nacional de Buenos Aires* , 1-3.

Julián, V., & Botti, V. (2003). Estudio de métodos de desarrollo de sistemas multi- agente. *Asociación española para la inteligencia artificial* .

Iglesias, C. (Enero de 1998). Definición de una metodología para el desarrollo de sistemas multi agente. *Universidad Politécnica de Madrid* .

Julian, V., Rebollo, M., & Carrascosa, C. (n.d.). Agentes de información. *Facultad de informática* .

Infomagnet. (2014). *Infomagnet*. From http://www.infomagnet.info/

Krulwich, B. (2000). Information Integration Agents: BargainFinder and NewsFinder. *Andersen Consulting Center for Strategic Technology Research* .

Bollacker, K., Lawrence, S., & Giles, L. (1998). CiteSeer: An Autonomous Web Agent for Automatic Retrieval and Identification of Interesting Publications. *ACM DL* .

Alerts, G. (2014). *Google Alerts*. From https://www.google.com.ec/alerts

Lieberman, H. (n.d.). *Letizia*. From http://web.media.mit.edu/~lieber/Lieberary/Letizia/Letizia-Intro.html

Yi Yang, D. (1999). Market Maker: an Agent Mediated MarketPlace Infraestructure. *Electrical Engineering and Computer Science* .

Guzmán, J., Sánchez, A., & Torres, D. (2006). Sistema multi agente para la recuperación de documentos digitales. *Revista Colombiana de Tecnologías avanzadas* , 114.

Berrocal, J., Figuerola, C., Zazo, Á., & Rodríguez, E. (2003). Agentes inteligentes: Recuperación autónoma de información en la web. *Departamento de Informática y Automática* .

Jennings, N., Sycara, K., & Wooldridge, M. (1998). A Roadmap of Agent Research and Development. *ACM DL* .

Alvarado, J., Barrera, A., González, M., & Junco, M. (2014). Diseño de agentes para recuperar información para el enriquecimiento de ontologías dirigidas a la epidemiología: el caso de la tuberculosis. *Instituto Tecnológico y de Estudios Superiores de Monterrey* , 33-34.

Mendoza, R. (2009). Agentes móviles para la recuperación de información en bibliotecas digitales. *Universidad Mayor de San Andrés* .

Villalba, R. (2007). La web semántica. *Universidad Católica Nuestra Señora de Asunción* .

Fernández, A., Carbonell, S., Pérez, Y., & Villalón, T. (2009). Las ontologías. Nuevos retos. *DIALNET* .

Arenas, R., & Prieto, M. (2004). Web Semántica. *Instituto de Telemática* .

Berners-Lee, T., Hendler, J., & Lasilla, O. (2001). The Semantic Web.

Castells, P. (2012). La Web Semántica.

Abián, M. (2013). *Slideshare*. From http://www.slideshare.net/torrubia/ontologas-qu-son-y-a

Peis, E., Hassam, Y., Herrera, E., & Herrera, J. (2003). Ontologías, metadatos y agentes: recuperación "semántica" de la la información. *Universidad de Granada* .

Drake, J., & López, P. (2009). Ingeniería de programación. *Ingeniería software* .

Bravo, C., & Redondo, M. (2004). *Sistemas interactivos y colaborativos en la web.*

Cardozo, G. (Octubre de 2009). Historia del concepto de red social. *Universidad de Santo Tomás* .

Amandi, A. (2001). Desarrollo de sistemas multi agente. *Revista Iberoamericana de Inteligencia Artificial* (13), 33-35.

Díaz, A., Trilnik, F., Campo, M., & Clausse, A. (2001). Simulación basada en agentes reactivos. *Facultad de Ciencias Exactas - ISIST AN* .

Lange, D., & Mitsuru, O. (Agosto de 1998). Programming and Deploying Java Mobile Agents With Aglets. *Addison-Wesley Pub* .

Castelfranchi, C., & Lespérance, Y. (Julio de 2000). Developing multi-agent systems with jade, Agent Theories, Architectures. *Springer* .

JADE. (2002). Java agent development framework. *TILAB* .

JAFMAS. (2002). Java-based frame work for multi- agent systems. *University of Cincinnati* .

Pavon, J. (2003). Metodologías de desarrollo de sistemas multi agente.

Gómez, J. (2003). Metodologías para el desarrollo de sistemas multi agente, Departamento de Sistemas Informáticos y Programación. *Departamento de sistemas informáticos y programación* .

Férnandez, D. (2004). Desarrollo de una metodología para un nuevo paradigma de software. *Lenguajes y sistemas informáticos* .

Sycara, K. (1998). Multiagent Systems. *AIMagazine* .

Gallego, F., Llorens, F., & Rizo, R. (2004). Breve análisis de algunas metodologías de diseño de SMA. *Informática industrial e inteligencia artificial* .

DeLoach, S. (2000). Multiagent Systems Engineering: A methodology and Language for Designing Agent Systems. *Department of Electrical & Computer Engineering Air Force Institute of Technology* .

DeLoach, S. (2001). Proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS.

Ferber, J., & Gutknecht, O. (1999). A meta-model for the analysis and design of organizations in multi-agent systems. *Laboratorio de informática, Universidad Montpellier* .

Di Leo, J., Jacobs, T., & Scott, D. (2002). Integrating Ontologies into Multiagent Systems Engineering. *Department of Computing and Information Sciences* .

DeLoach, S., Wood, M., & Sparkman, C. (2000). Multi-agent Systems Engineering. *Air Force Institute of Technology Graduate School of Engineering and Management* .

Kendall, E., Palanivelan, U., & Kalikivayi, S. (1998). Capturing and Structuring Goals: Analysis Patterns, Proceedings of the Third European Conference on Pattern Languages of Programming and Computing.

Self, A., & DeLoach, S. (2003). Designing and Specifying Mobility within the Multi-agent Systems Engineering Methodology. *Air Force Personnel Center Randolph Air Force Base, Department of Computing and Information Sciences* .

C. S, R., Araújo, A., Gomes, C., da Costa, R., & Werneck, V. (2009). Modelagem de Requisitos Orientada a Agentes utilizando MaSE. *Instituto de matemática y estática* .

Higes, Á. (2011). Servicios de directorios inteligentes y modelos de confianza para la recomendación de servicios. *Escuela Técnica Superior de Ingeniería Informática* .

Fielding, R. (2000). Architectural styles and the design of network-based software architectures. *University of California* .

Pautasso, C., Zimmermann, O., & Leymann, F. (2008). Restful web services vs. big'web services: making the right architectural decision. *ACM: Proceedings of the 17th international conference on World Wide Web* .

Espí, J. (2013). Integración de procesos industriales mediante patrones SOA y tecnología RESTful. *Universidad de Alicante* .

Camacho, A. (s.f). *Procesador en línea de texto científico, Calisoft, La Habana.*

Johnson, R., & Foote, B. (1998). Designing reusable classes. Journal of Object-Oriented Programming.

Foaf. (2014). *Foaf*. From http://www.foaf-project.org

Zambrano, M. (2009). XML Y RDF en la Web Semántica. *Universidad de Cauca* .

Qaissi, H. (2009). Arquitecturas de bases de datos web, Tecnologías de la Web Semántica. *DIP-UPM* .

Hípola, P., & Vargas, B. (1999). Agentes inteligentes: definición y tipología. Los agentes de información.

Coppin, B. (2004). Artificial Intelligence Illuminated.

Lara, P., & Martínez, J. (2014). Agentes inteligentes en la búsqueda y recuperación de información.

Miranda, C. (2009). Sistema multiagentes para la descripción, localización, recuperación y acceso a documentos. *Grupo Ingeniería de software y nuevas tecnologías Universidad Autónoma del Caribe* .

Jiménez, J., Ovalle, D., & Ochoa, J. (2008). SMART: Sistema multi agente robótico. *Universidad Nacional de Colombia* .

Adel Al-Jumaily, A., & Al-Jaafreh, M. (2006). Multi – Agent System Concepts Theory and Application Phases. *University of Technology* .

Serrano, A., & Ossowski, S. (2006). Inteligencia artificial distribuida y sistemas multi agente. *Universidad Politécnica de Madrid, Universidad Juan Carlos de Madrid* .

D WeiB, G. (2000). Learning to Coordinate Actions in Multi-Agen t Systems. *Institut fur lnformatik, Technische Universitat Miinchen Arcisstr.*

Jiménez, D. (2013). Un Framework flexible para resolver problemas de optimización mediante sistemas multi agente. *Universidad Autónoma de Madrid* .

Flórez, H. (2006). Web Semántica aplicada al registro académico institucional. *Univerdidad Konrad Lorenz* .

Vega, R., Rodríguez, L., & Justo, Y. (2009). Procedimiento para realizar pruebas de usabilidad,. *Universidad de las Ciencias Informáticas* .

Sommerville, I. (2006). *Ingeniería del software.* Madrid.

Whittaker, J. (2000). What is software testing? And why is it so hard? *IEEE* .

Ramos, J. (2013). Técnicas de analisis de sentimientos para inferir el estrés académico de los estudiantes de la titulación de Sistemas Informáticos y Computación.

ANEXOS

ANEXO I: Encuesta experiencia de usuario

UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

SECCIÓN DEPARTAMENTAL DE INTELIGENCIA ARTIFICIAL

ENCUESTA DE APLICACIÓN DEL FRAMEWORK DE SISTEMA MULTI AGENTE

Objetivo: El presente cuestionario tiene como objetivo principal conocer su opinión acerca del uso de la aplicación del framework de sistema multi agente, que recopila información de manera automática desde recursos web y físicos.

DATOS GENERALES

Nombres y apellidos:

_____

SEÑALE LA RESPUESTA CORRECTA

1. **Señale con un (x) uno o más según crea conveniente.**

   ¿Cuáles son los recursos web por los cuáles usted interactúa más?

   a) Facebook [ ]
   b) LinkedIn [ ]
   c) Google [ ]
   d) Otros [ ]

   Especifique:

   _____

2. ¿Cómo se enteró del sistema?

a) Me llego una notificación vía email ☐

b) Un amigo me invito ☐

c) Otro ☐

Si la respuesta es el literal (a), el mensaje le pareció adecuado (solo en caso de ser (no), escriba sus argumentos)

Si ☐

No ☐

Especifique:

_____

3. Valorar en una escala de 1 a 5, donde (1) es poco y (5) es mucho.

	1	**2**	**3**	**4**	**5**
a) La recopilación de información fue interactiva y entendible					
b) El sistema le parece confiable					
c) El sistema corrió bien en su PC					
d) Una vez que se encontraba en el sistema, estaba claro lo que debía hacer					

4. Señale con un (x) uno o más ítems según crea conveniente.

¿En cuál de las siguientes fuentes, usted obtuvo la mayor cantidad de información?

a) Facebook ☐

b) LinkedIn ☐

c) Google ☐

5. **Señale con una (x) uno o más ítems según crea conveniente.**

   ¿Cuál de las siguientes características considera usted son relevantes en la aplicación de framework de sistema multi agente?

   a) Presentación de la información que se recopila desde los recursos web ☐
   b) Presentación de la descripción del proceso a realizarse en cada paso ☐
   c) Presentación de las actividades a realizarse en todo el sistema en forma de pasos ☐
   d) La invitación mediante un mail ☐
   e) El mail final que indica que información se ha recuperado de usted ☐

6. **Valorar en una escala del 1 al 5, donde (1) es poco y (5) es mucho. Señale con una (x) el ítem según crea conveniente.**

   ¿El tiempo que le tomo completar todos los requisitos del sistema fue adecuado?

1	2	3	4	5

   ¿Por qué?

   _____

7. **¿Considera usted que la información recopilada se puede utilizar para el análisis con fines científicos, en diversos ámbitos, como por ejemplo la psicología?**

Si ☐

No ☐

¿Por qué?

MUCHAS GRACIAS POR SU COLABORACIÓN

ANEXO II: Demostración del proceso de recopilación de información

a) Recuperar los datos de Facebook, para esto se debe realizar la autorización de los datos desde la aplicación. La autorización se la realiza una sola vez. Para aceptar la recopilación, damos click en **Login with Facebook.**

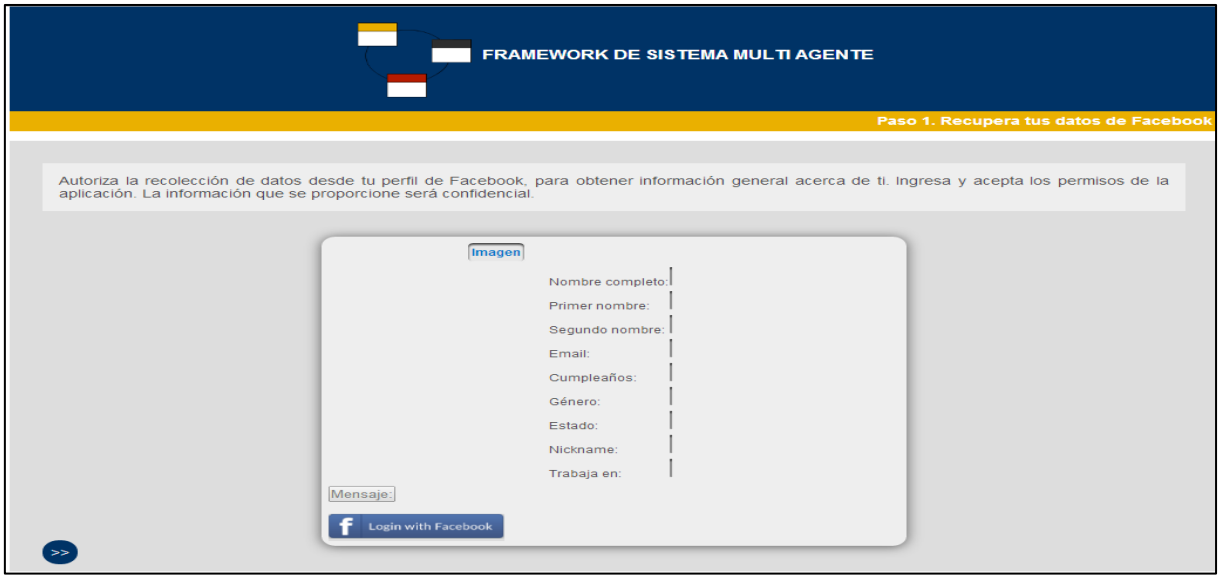


Figura 132: Recuperar datos desde Facebook.

Fuente: La autora.

b) Al hacer click en **Login with Facebook** aparece una pantalla donde se ingresa el usuario y contraseña.

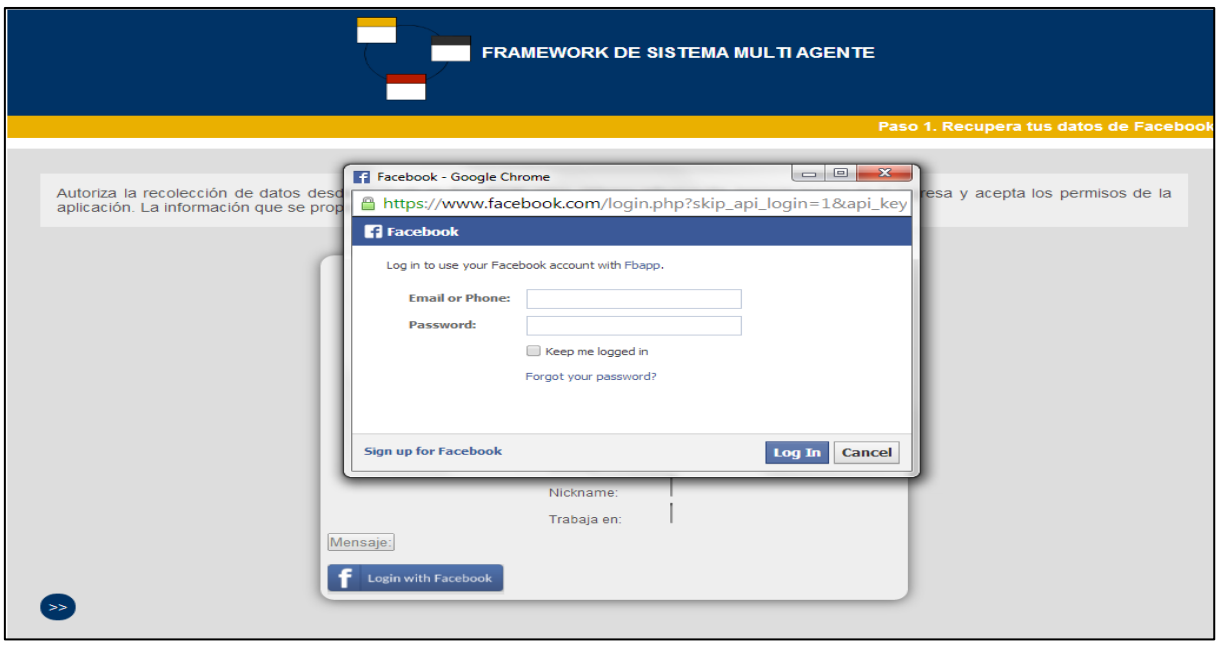


Figura 133: Autorizar la recopilación desde Facebook.

Fuente: La autora.

c) Se ingresa el email o teléfono y contraseña y click en **Log in**.

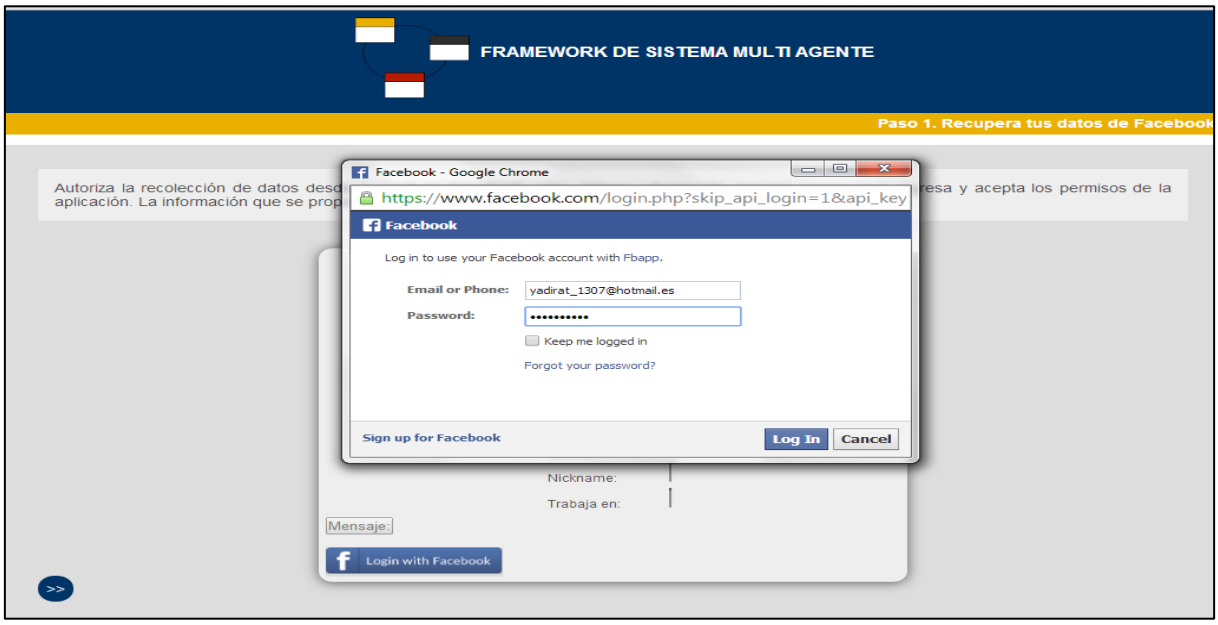


Figura 134: Ingresar el email y la contraseña de Facebook.

Fuente: La autora.

d) Si el ingreso **no** es exitoso, se observa el mensaje de la aplicación que muestra **No se pudo identificar al usuario.**

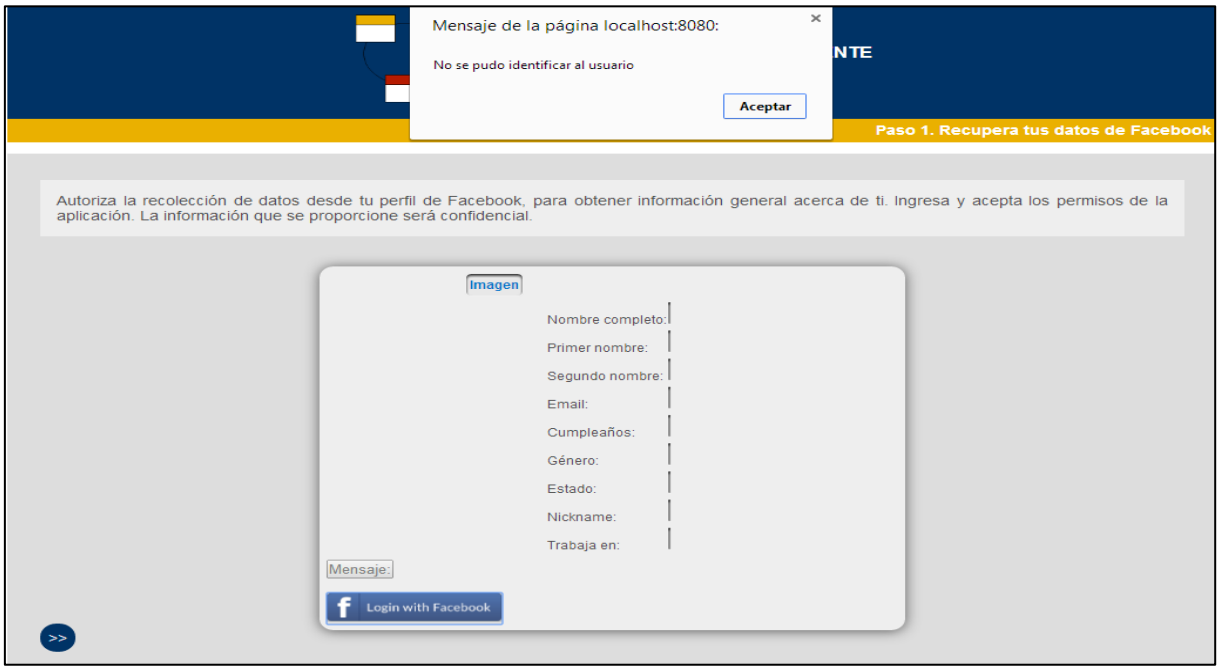


Figura 135: Fallo en autenticación.

Fuente: La autora.

e) Si el ingreso **si** es exitoso, se observa la recopilación de los datos, y el mensaje de **Almacenado con Éxito.**

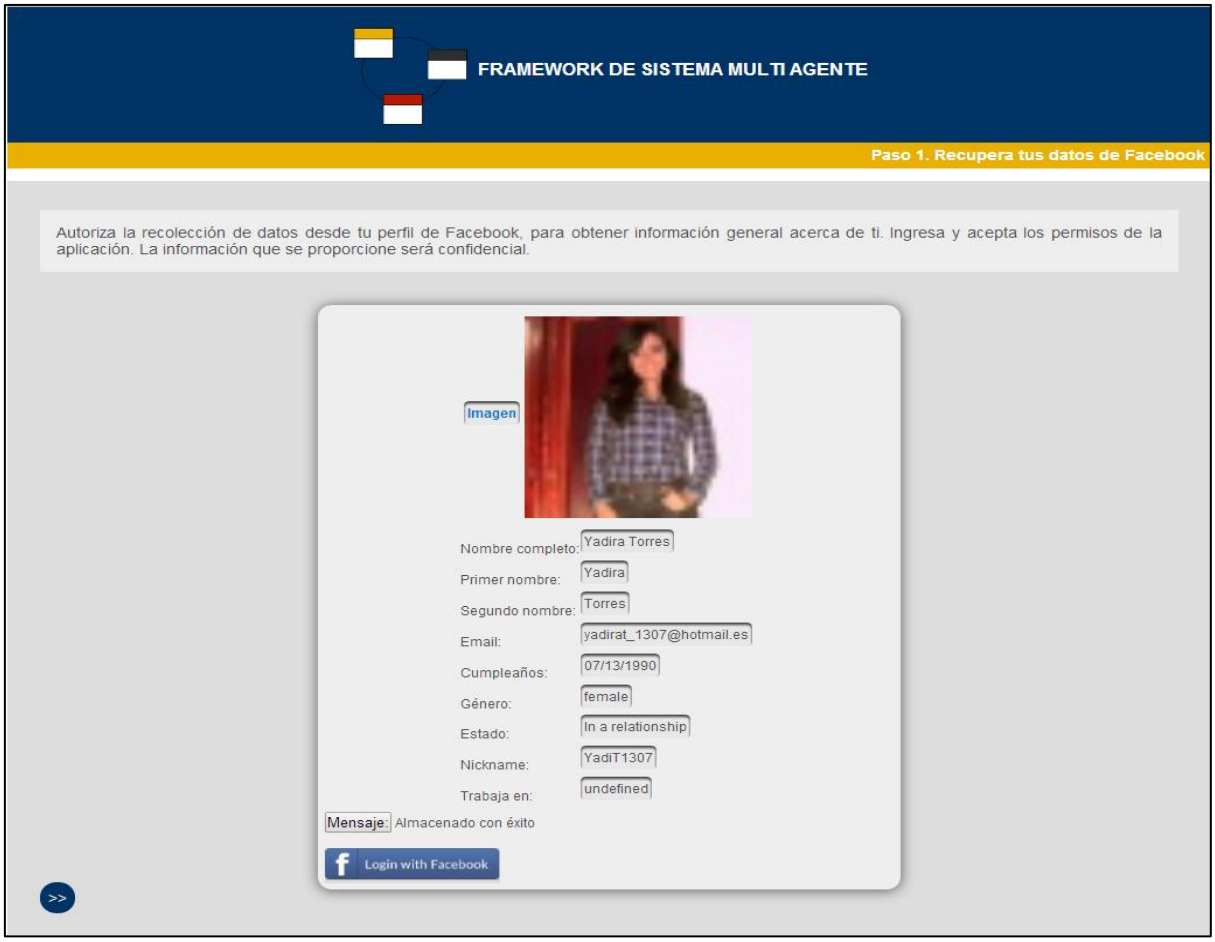


Figura 136: Recopilación correcta de información.

Fuente: La autora.

f) Click en la pestaña de siguiente y aparece la interfaz para recuperar datos de LinkedIn, para esto se realiza la autorización de los datos desde la aplicación. La autorización se la realiza una sola vez. Para aceptar la recopilación, click en **Sign in with LinkedIn.**

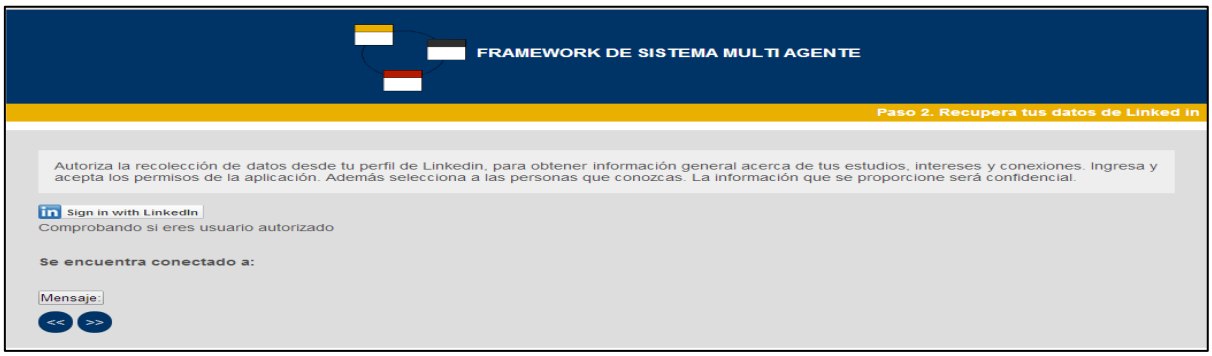


Figura 137: Recuperar datos desde LinkedIn.

Fuente: La autora.

g) Al hacer click en **Sign in with LinkedIn** aparece una pantalla donde se debe ingresar el usuario y contraseña.

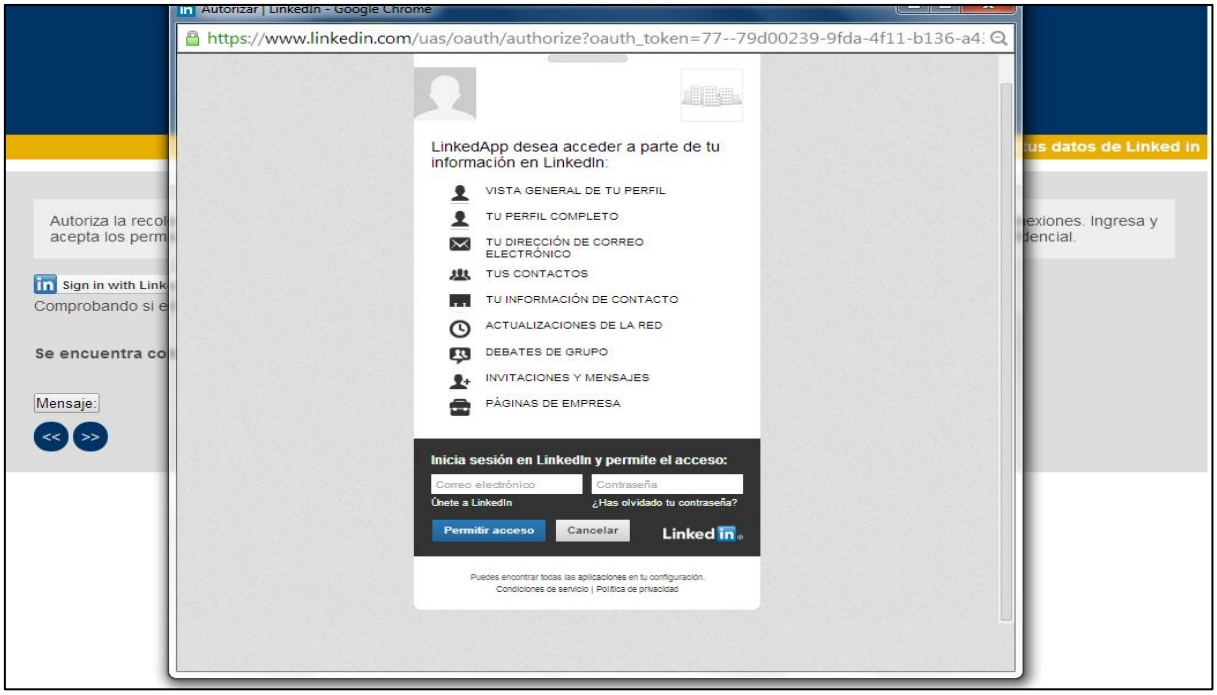


Figura 138: Autorizar la recopilación desde LinkedIn

Fuente: La autora.

h) Se ingresa el email y contraseña y click en **Permitir acceso**.

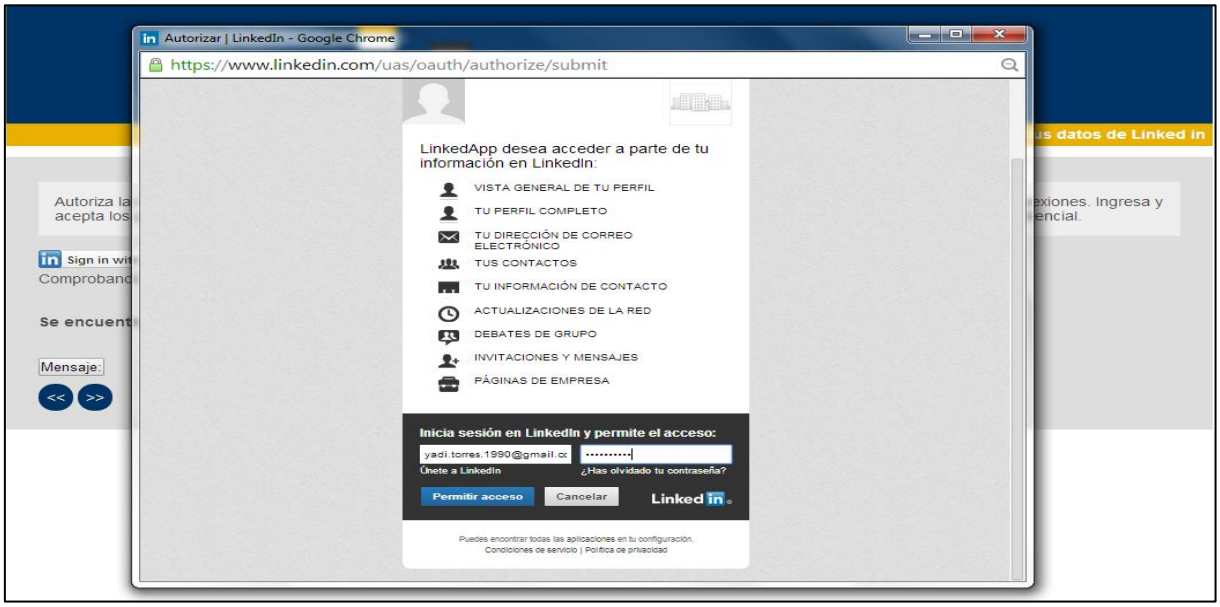


Figura 139: Ingresar el email y la contraseña de LinkedIn

Fuente: La autora.

i) Se podrá observar los datos que han sido recopilados y las conexiones desde LinkedIn.

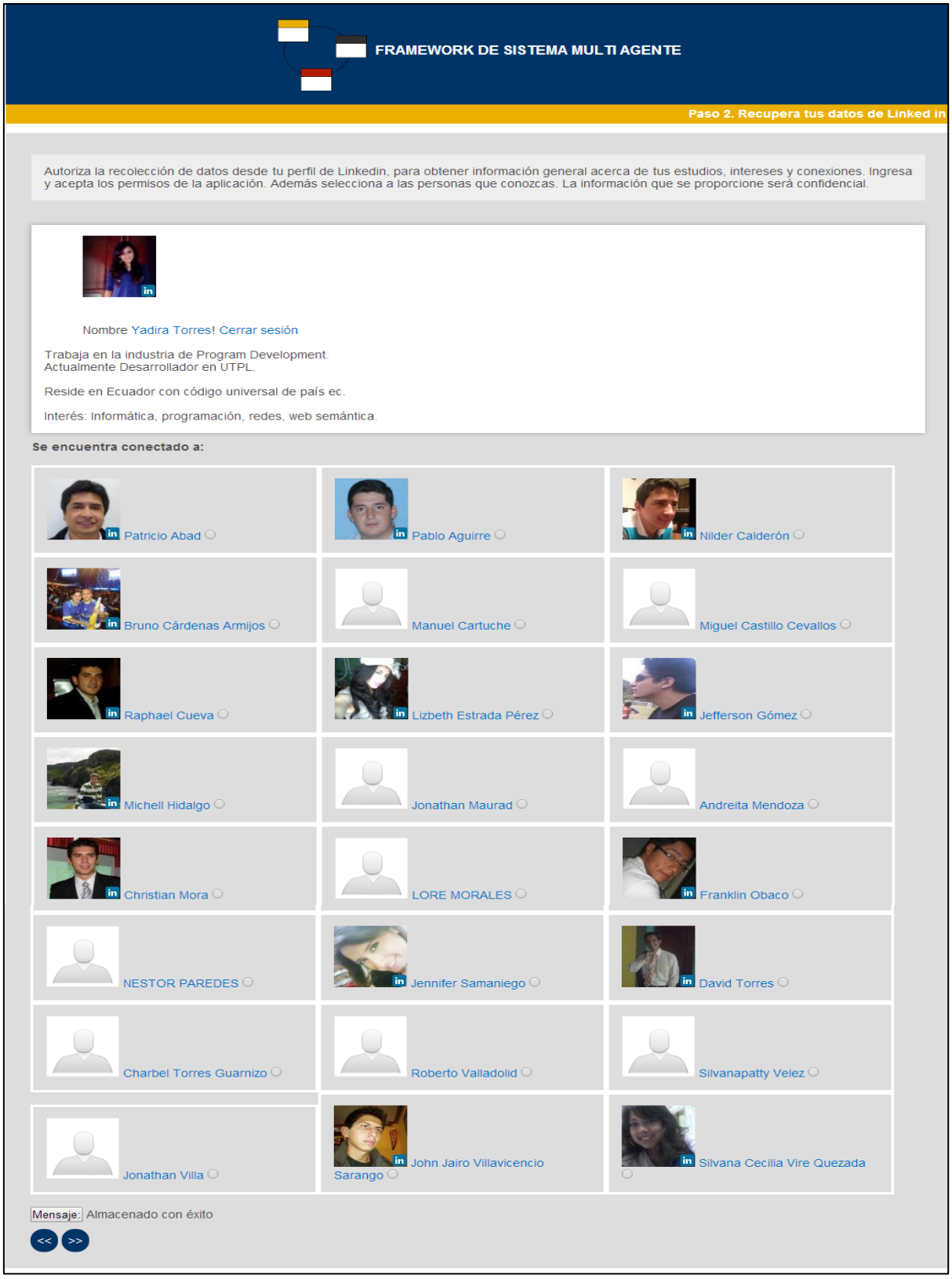


Figura 140: Recopilación de datos desde LinkedIn.

Fuente: La autora.

j) Las conexiones deben irse aprobando una por una, el usuario podrá seleccionar la cantidad conveniente de personas con las que tenga relación. Para eso se selecciona una conexión en particular y aparecerá el mensaje **Estas seguro de conocer a <nombre de la persona>** y click en **Aceptar**.

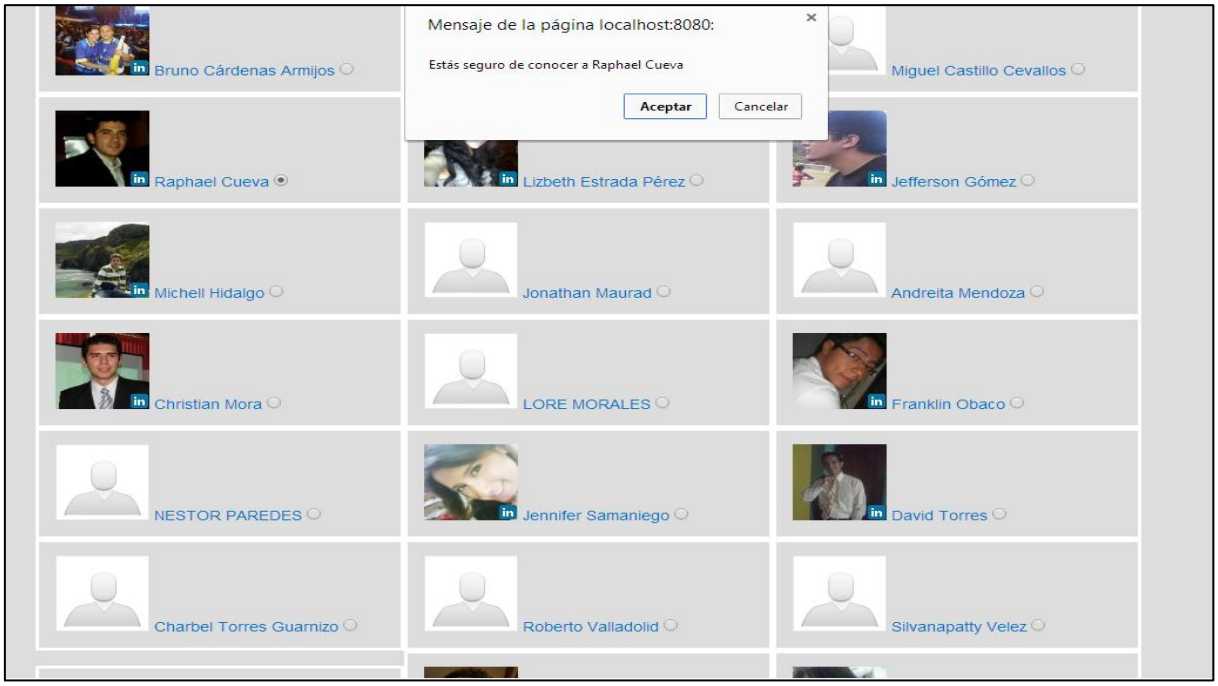


Figura 141: Creando conexiones desde LinkedIn

Fuente: La autora.

k) Al aceptar la conexión dará un mensaje de **Almacenado con Éxito.**

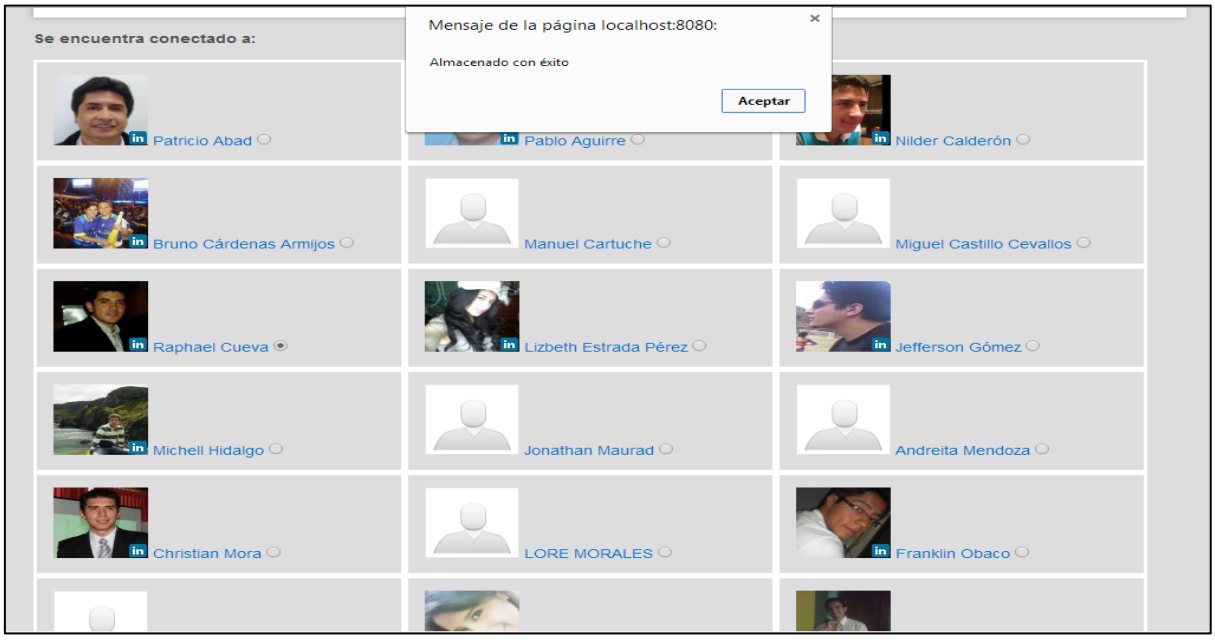


Figura 142: Mensaje de confirmación de almacenamiento de conexión.

Fuente: La autora.

l) Al hacer click en la pestaña de siguiente aparece la interfaz para realizar búsquedas acerca de links que tengan relación con el usuario. En la caja de texto, llamada **Buscador**, se escribe los parámetros para identificar los links en relación.

Figura 143: Recopilación de datos desde Google.

Fuente: La autora.

m) Se observa en las secciones los parámetros encontrados acerca de las personas, divididas en 3 secciones, web, noticias e imágenes.

Figura 144: Filtrado de parámetros.

Fuente: La autora.

n) Se selecciona la información que el usuario cree importante y encuentra relación con su perfil y aparece un mensaje de confirmación con el texto **Estas seguro del <link>** y seguidamente click en **Aceptar**.

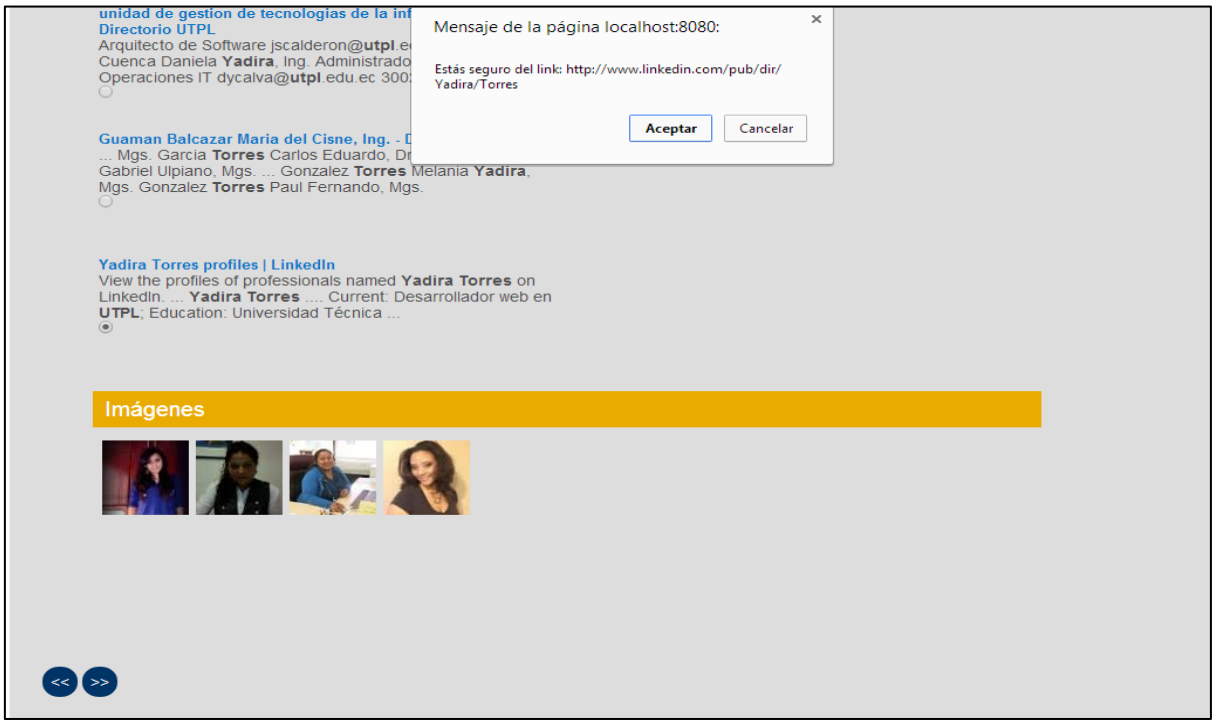


Figura 145: Selección de links en relación.

Fuente: La autora.

o) Aparece un mensaje de confirmación **Almacenado con éxito**

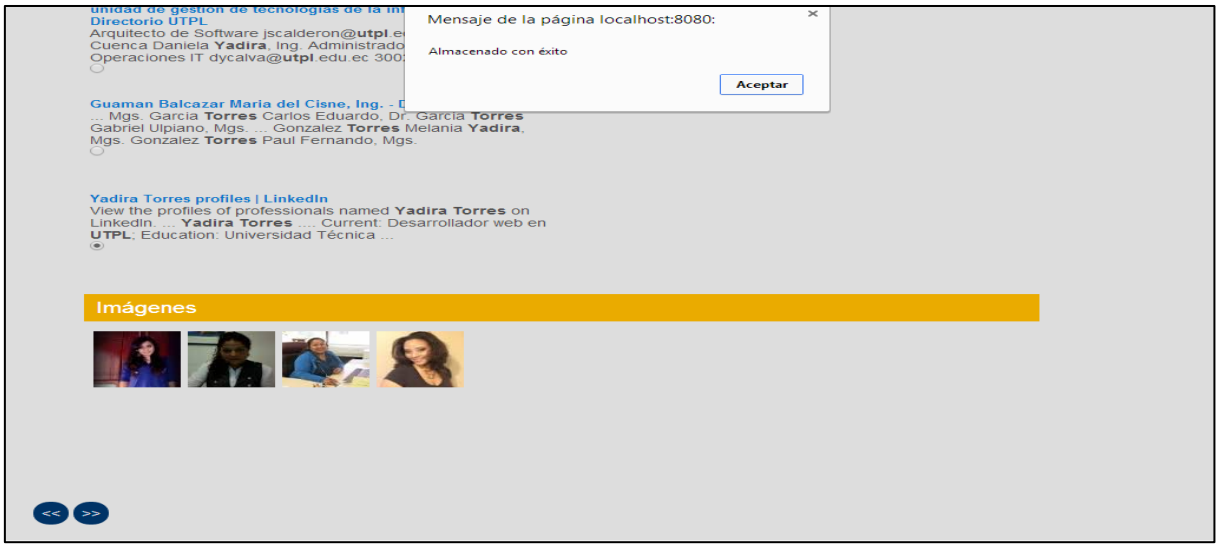


Figura 146: Confirmación de almacenamiento de información desde Google.

Fuente: La autora.

p) Click en la pestaña siguiente y aparece la interfaz para recuperar datos desde la ontología Test_EstresM, si es que los datos existen se da la opción de elegir entre

los resultados presentados, si los datos no existen, arroja un mensaje para ir al siguiente paso.

Figura 147: Búsqueda de información en Test_EstresM.

Fuente: La autora.

q) Click en la pestaña de siguiente y aparece la interfaz para llenar la encuesta. Las primera opción es **¿Has tenido momentos de preocupación o nerviosismo?,** en donde la respuesta **sí** o **no** definirá el proceso a llevarse a cabo.

Figura 148: Cuestionario.

Fuente: La autora.

r) Al hacer click en la opción **si** se desplegarán una seria de preguntas a contestar. Cuando se haya concluido el cuestionario click en **Guardar**.

Figura 149: Despliegue de opciones en cuestionario.

Fuente: La autora.

s) Click en la pestaña siguiente y aparece la interfaz en donde se notifica al usuario de los datos que fueron recopilados y se informa vía mail, en caso de no existir un mail durante el proceso de recopilación, se pide al usuario, que ofrezca un mail para poder enviar su información. Al llegar a este punto se ha concluido con el proceso de recopilación.

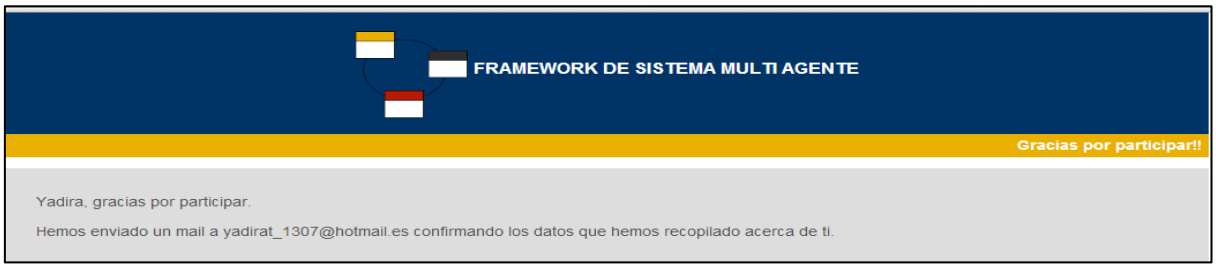


Figura 150: Finalización del proceso de recopilación.

Fuente: La autora.

ANEXO III: Código Agente Facebook, recopilar informacion de Facebook.

a) Parámetros

```
package services;
import java.io.IOException;
import javax.ejb.EJB;
import javax.ejb.Stateless;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import process.RemoveAccents;
/**
*
* @author ymtorres
*/
@Stateless
@Path("/dataFb")
public class FoafResource {

@EJB
private FoafBean nameBean;
@GET
@Produces("text/plain")
public String foaf(
@QueryParam("parrafo") String parrafo,
@QueryParam("fname") String fname,
@QueryParam("lname") String lname,
@QueryParam("email") String email,
@QueryParam("birthday") String birthday,
@QueryParam("gender") String gender,
@QueryParam("ciudad") String ciudad,
@QueryParam("estadoR") String estadoR,
@QueryParam("work") String work,
@QueryParam("surname") String surname
) throws IOException {
String id = "";
String person = RemoveAccents.remove(parrafo);
String[] parr = person.split(" ");
for (int i = 0; i < parr.length; i++) {
//System.out.println(parr[i]);
id = id + parr[i].charAt(0);
id = id + parr[i].charAt(1);
```

```
id = id + parr[i].charAt(2);
}
return nameBean.putData(id, person, fname, lname, birthday, gender, estadoR, email, surname);
}
}
```

b) Instancias

```
package services;
import java.io.IOException;
import javax.ejb.Singleton;
@Singleton
public class FoafBean {
   private String msj;
   public String putData(String idPerson, String nombre, String firstName, String lastname, String birthday, String gender, String status, String account, String surname) throws IOException {
      //instanciamos un objeto de la clase datafoaf
      datafoaf obj = new datafoaf();
      msj = obj.Individuals(idPerson, nombre, firstName, lastname, birthday, gender, status, account, surname);
      return msj;
   }
}
```

c) Clases de objeto

```
package services;
import com.hp.hpl.jena.ontology.DatatypeProperty;
import com.hp.hpl.jena.ontology.Individual;
import com.hp.hpl.jena.ontology.ObjectProperty;
import com.hp.hpl.jena.ontology.OntClass;
import com.hp.hpl.jena.ontology.OntModel;
import com.hp.hpl.jena.ontology.OntModelSpec;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.util.iterator.ExtendedIterator;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.PrintWriter;
import process.SourceFile;
/**
 *
 * @author ymtorres
 */
public class datafoaf {
   public static String msj;
```

```
    public static OntModel model = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);
    public static String Individuals(
            String IdPerson,
            String nameI,
            String firstNameI,
            String LastNameI,
            String birthdayI,
            String genderI,
            String statusI,
            String accountI,
            String surnameI
    ) throws FileNotFoundException //throws FileNotFoundException, IOException
    {
        try {
            model.read(SourceFile.ruta(), "RDF/XML");
            ExtendedIterator<ObjectProperty> i = model.listObjectProperties();
            ObjectProperty cls = i.next();
            String NS = cls.getNameSpace();

            //System.out.println(NS);
            OntClass PersonClass = model.getOntClass(NS + "Person");

            // System.out.println(PersonClass);
            Individual person = model.createIndividual(NS + IdPerson, PersonClass);
            DatatypeProperty nameC = model.getDatatypeProperty(NS + "name");
            DatatypeProperty firstName = model.getDatatypeProperty(NS + "firstName");
            DatatypeProperty lastName = model.getDatatypeProperty(NS + "lastName");
            DatatypeProperty birthday = model.getDatatypeProperty(NS + "birthday");
            DatatypeProperty gender = model.getDatatypeProperty(NS + "gender");
            DatatypeProperty status = model.getDatatypeProperty(NS + "status");
            DatatypeProperty account = model.getDatatypeProperty(NS + "accountName");
            DatatypeProperty surname = model.getDatatypeProperty(NS + "Surname");
//
            person.setPropertyValue(nameC, model.createLiteral(nameI));
            person.setPropertyValue(firstName, model.createLiteral(firstNameI));
            person.setPropertyValue(lastName, model.createLiteral(LastNameI));
            person.setPropertyValue(birthday, model.createLiteral(birthdayI));
            person.setPropertyValue(gender, model.createLiteral(genderI));
            person.setPropertyValue(status, model.createLiteral(statusI));
            person.setPropertyValue(account, model.createLiteral(accountI));
            person.setPropertyValue(surname, model.createLiteral(surnameI));
//
            File file = new File(SourceFile.ruta1());
```

```
            model.write(new PrintWriter(file), "RDF/XML");
            msj = "Almacenado con éxito";
        } catch (Exception e) {
            msj = "No se pudo almacenar " + e;
        }
        return msj;
    }
}
```