UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA: INGENIERÍA DE SISTEMAS

Trabajo de titulación previo a la obtención del título de: INGENIEROS DE SISTEMAS

TEMA:

DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA MULTIAGENTE PARA DISTRIBUIR Y PARALELIZAR REGISTROS MÉDICOS

AUTORES:

RAFAEL ALEJANDRO SÁNCHEZ GUAYASAMÍN

ADRIÁN ALEXIS CASTRO VINUEZA

TUTOR:

WASHINGTON ARSENIO RAMÍREZ MONTALVÁN

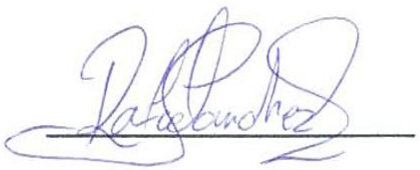
Quito, abril de 2016

Cesión de derechos de autor

Nosotros Adrián Alexis Castro Vinueza con documento de identificación N° 172109083-3 y Rafael Alejandro Sánchez Guayasamín con documento de identificación N° 172099419-1, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación intitulado: DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA MULTIAGENTE PARA ADMINISTRAR Y PARALELIZAR REGISTROS MÉDICOS, mismo que ha sido desarrollado para optar por el título de: Ingenieros de Sistemas, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestras condiciones de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.

Adrián Alexis Castro Vinueza
172109083-3



Rafael Alejandro Sánchez Guayasamín
172099419-1

Quito, abril 2016

Declaratoria de coautoría del docente tutor

Yo, declaro que bajo mi dirección y asesoría fue desarrollado el trabajo de titulación DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA MULTIAGENTE PARA DISTRIBUIR Y PARALELIZAR REGISTROS MÉDICOS realizado por RAFAEL ALEJANDRO SÁNCHEZ GUAYASAMÍN y ADRIÁN ALEXIS CASTRO VINUEZA, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana para ser considerados como trabajo final de titulación.

Quito, abril 2016

Washington Arsenio Ramírez Montalván
C.C 171080468-1

DEDICATORIA

A Dios por darme el regalo de la vida y compartirla con mis seres queridos.

A mis padres Carlos y Natacha, Dios me dio la vida y ustedes me dieron la libertad de vivirla, gracias por enseñarme que el trabajo y el conocimiento deben ser parte de mi existencia y decisiones. Muchas gracias por confiar en mí y mostrarme que con fe en Dios y en lo que soy, lo puedo y lo hago.

A mis hermanos Renato y Salomé, gracias por cuidarme y siempre estar junto a mí brindándome sus consejos y apoyo incondicional, muchas veces poniéndose en papel de padres.

A toda mi familia en especial a mis tíos, que siempre están ahí cuando más los necesito y siempre se han preocupado por mí.

A Elizabeth, por ser una persona muy importante en mi vida, y ser otra razón más para luchar y seguir adelante.

Gracias a todos, porque sin ustedes no sería nada.

Rafael Alejandro Sánchez Guayasamín

Dedico a Dios por permitirme llegar hasta éste momento de mi vida profesional.

A mi papá Iván por sacrificar mucho por darme todo y por sus enseñanzas.

A mi mamá Rosario por estar a mi lado a cada instante con sus consejos y apoyo.

A mi hermana Irene que ha estado cuando más la he necesitado.

A Rocío que ha sido el apoyo necesario para seguir adelante y la fortaleza para continuar día a día.

A mi tía Paulina que se encuentra a lado de Dios que me guio y acompañó a cada momento.

Adrián Alexis Castro Vinueza

AGRADECIMIENTO

A la Universidad Politécnica Salesiana por aportar con todo lo que necesitábamos en el desarrollo de la titulación.

A las autoridades, docentes, y personal administrativo de la institución que nos han apoyado en nuestra formación y que han provisto de las herramientas necesarias para enfrentarnos a los problemas profesionales que seguramente deberemos superar.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ANEXOS

RESUMEN

El campo de la salud es muy amplio, se encuentra caracterizado por múltiples sistemas de información y heterogeneidad de datos, por lo que la comunicación debe darse de una manera consolidada, logrando altos estándares de eficiencia de los servicios médicos.

Existen varias metodologías para implementar interoperabilidad entre sistemas de información en salud, sin embargo persisten problemas relacionados con la distribución, tolerancia a fallos y estándares no compatibles.

El presente trabajo de titulación tiene como objetivo hacer frente a los problemas antes descritos automatizando las tareas de registros médicos, creando nuevas estrategias de manejo de datos, permitiendo el procesamiento y distribución de archivos en paralelo, se construyó el aplicativo dividiéndolo en tres fases.

En la primera fase JAVA EE se encuentra la interfaz del usuario, es decir el espacio en el que el médico se encarga del ingreso y modificación de los registros e imágenes médicas del paciente DICOM; en la segunda fase WSIG, se expone los servicios proporcionados por los agentes JADE y en la tercera fase en el Sistema multiagente está la lógica de negocio, en la que a través de mensajes HL7, se determina como los archivos van a ser creados, mostrados, guardados, cambiados y distribuidos desde la capa de datos Hadoop hacia el usuario final.

Las pruebas fueron realizadas en un ambiente distribuido, donde finalmente con una configuración mínima de hardware, se demostró la factibilidad del sistema, abriéndose la posibilidad de que con determinados ajustes, según las necesidades, se pueda implementar en una o varias localidades médicas.

ABSTRACT

Health field is very wide and is characterized by multiple information systems and heterogeneous data, so that communication should be consolidated, achieving high standards of efficiency of medical services.

There are several methodologies to implement interoperability between health information systems, but still problems related to distribution, fault tolerance and incompatible standards.

This project aims face the problems described above automating tasks of medical records, creating new strategies for managing data, enabling the processing and distribution of files in parallel, It was divided the application constructed in three phases.

In first phase JAVA EE is the user interface, is the space in which the doctor is responsible for the entry and modification of records and DICOM medical images of the patient; In second phase WSIG, the services provided by the JADE agents and in third phase the multi-agent system is the business logic, which through HL7 messages, is determined the way that files will be created, shown is exposed, saved, changed and distributed from the data layer Hadoop to the end user.

The tests were performed in a distributed environment, where finally with minimal hardware configuration, the feasibility of the system was demonstrated, opening the possibility that with certain adjustments, as needed, can be implemented in one or more medical locations.

INTRODUCCIÓN

En la actualidad en el área de la tecnología de la información cada vez tiene más relevancia dentro del campo médico, la aparición de nuevas generaciones de servicios como expedientes electrónicos, tomografía computarizada, exámenes de laboratorio, y en fin un sin número de equipos y servicios médicos están conectados parcial o totalmente a un programa o programas de cómputo.

El conjunto de estos programas llevan el nombre de Sistema de Información en Salud (HIS); la falta de un estándar de comunicación limita la interoperabilidad entre estos sistemas impidiendo así en muchas formas la mejora en la gestión hospitalaria y limitando sus funciones.

Los HIS tienen una parte fundamental en el sistema multiagente. En (Corchado, Alonso, & Tapia, 2011) utilizan la tecnología de sistemas multiagentes en el área de medicina, el cual ha permitido la creación de una aplicación para la ayuda de enfermeras a cargo de pacientes con problemas de Alzheimer.

Por lo cual, en el capítulo uno se realizó un estudio del problema y análisis de la situación actual de los sistemas de información en salud, como el estudio de las tecnologías de sistemas multiagentes (SMA), HL7, DICOM, HADOOP y su campo de aplicación para evaluar cómo estas tecnologías nos pueden ayudar con una solución al problema. En el capítulo dos se realizó la fase de análisis y diseño de las herramientas de desarrollo a utilizar. En el capítulo tres se efectuó el desarrollo del producto tecnológico que sirva como base para enfrentar los problemas planteados en el capítulo uno. Al identificar la funcionalidad de nuestro producto tecnológico, se efectuará la simulación por medio de un software que permita llevar pruebas de stress sobre el producto. Finalmente en el capítulo cuatro se realizó las conclusiones y recomendaciones del trabajo realizado.

Antecedentes

En años recientes, dos tecnologías diferentes han sido el sujeto de múltiples investigaciones dentro del campo de la salud, específicamente en lo relacionado a los HIS: computación en la nube o “cloud computing” en inglés y los sistemas multiagentes.

En este contexto, un sistema móvil que permite a guardar, modificar y recuperar expedientes médicos electrónicos en la nube, es propuesto en (Doukas, Pliakas, & Maglogiannis, 2010), en el cual una aplicación móvil basada en clientes Android permite a los usuarios recuperar remotamente información de salud e imágenes médicas.

Otro sistema, es el enunciado por (Rolim, Koch, Westphall, Werne, Fracalossi, & Salvador, 2010) y se trata de una red de sensores wireless usados para automatizar el proceso de recolección de datos. La información recolectada es distribuida a través de una solución de cloud computing para el personal médico.

Un enfoque alternativo es propuesto en (Koufi, Malamateniou, & Vassilacopoulos, 2009), donde los datos y la interoperabilidad del servicio es obtenida a través de un sistema distribuido orientado a agentes. Finalmente, (Capozzi & Lanzola, 2010) y (Lyell & Liu, 2012) usan la tecnología de sistemas multiagentes para apoyar el monitoreo de cuidado en casa y tratamiento de los pacientes.

Justificación

El campo de la medicina se enfrenta a un número creciente de desafíos. La incidencia de errores médicos está aumentando; muchas instalaciones médicas no tienen el personal necesario, y sirven a áreas cada vez más grandes; los costos de salud crecen constantemente.

Por otra parte, el presupuesto no es suficiente, un manejo eficiente de los recursos médicos a menudo son difíciles de conseguir, y las instalaciones médicas están siempre bajo la presión de proveer mejores servicios con menos recursos.

Los sistemas de información en salud actualmente son los encargados de enfrentar estos desafíos. Los sistemas HIS pueden proveer una mejor coordinación entre el personal médico y las instalaciones, a la vez pueden reducir los costos de salud mejorando en muchas formas la gestión hospitalaria.

Desafortunadamente, debido a la complejidad inherente a su dominio de aplicación, HIS es fragmentado en varios sistemas que difícilmente llevan un estándar de comunicación, una definición de procesos y una homogeneidad en sus datos; volviéndolos así sistemas independientes y limitándolos en su funcionalidad.

Además, se debe tomar en cuenta que la interoperabilidad entre distintas instalaciones médicas es otro factor importante a ser investigado, ya que esta

conllevara a cientos o quizás miles de procesos individuales, realizando tareas individuales o cooperativas requiriendo una gran potencia de cálculo computacional.
El objetivo de este proyecto técnico es hacer frente a estos problemas de una manera eficiente, utilizando la tecnología de Sistemas Multiagentes en conjunto con estándares internacionales de comunicación tales como HL7 y DICOM; sacando provecho de HADOOP en el procesamiento de la información en paralelo, este tipo de procesamiento en la mayoría de los casos no ha sido reconocido como recurso para ser utilizado con este propósito.
Como resultado obtendremos un aplicativo que permita la interoperabilidad entre la información de los pacientes y un repositorio Open Source de imágenes médicas que maneje el estándar DICOM; en esta interfaz de comunicación podremos aplicar todos los temas antes mencionados; a su vez, este aplicativo quedará como punto de partida para nuevas investigaciones e implementaciones dentro de otras áreas de los HIS.

Objetivo general

Desarrollar e implementar un prototipo que permita verificar la factibilidad del diseño de un sistema para automatizar la interoperabilidad entre el control de historias clínicas y un repositorio de imágenes médicas, lo que permitirá probar suposiciones formuladas por analistas con respecto a los Sistemas multiagentes, distribución y paralelización de información, así como estándares de salud HL7 y DICOM.

Objetivos específicos

- Investigar las tecnologías sistemas multiagentes, sistemas de información en salud, e información en paralelo.
- Desarrollar un prototipo de Sistema multiagentes para el manejo de imágenes médicas DICOM utilizando estándares internacionales de salud HL7.
- Diseñar un aplicativo web, el cual será capaz de realizar visualización, modificación, borrar e ingresar registros médicos de un paciente.
- Mejorar los tiempos de respuesta implementado agentes en los diferentes procesos de comunicación entre los sistemas de información.
- Distribuir la información para su almacenamiento, recuperación de imágenes médicas utilizando tecnologías Open Source.

- Realizar pruebas de estrés sobre el aplicativo y el monitoreo de estado del software, las observaciones y conclusiones serán analizadas y anotadas en el documento final.

Alcance

- Se realizará un trabajo de investigación sobre las siguientes tecnologías: Sistemas Multiagentes JADE; Distribución y paralelización de información; Estándares internacionales de comunicación entre Sistemas de Información Médica.
- El fin de la investigación será desarrollar un producto que permita automatizar y optimizar la interoperabilidad entre sistemas de información médica.
- Se desarrollará un aplicativo web, el cual será capaz de realizar visualización, modificación, borrar e ingresar registros médicos de un paciente.
- Automatizar y optimizar la comunicación entre los registros médicos e imágenes médicas provenientes de un repositorio Open Source, utilizando estándares HL7 y estándares DICOM.
- Implementar servicios de almacenamiento y recuperación de imágenes médicas instalando una herramienta Open Source.
- Obtener imágenes médicas DICOM para pruebas, en el link http://www.osirix-viewer.com/datasets/ se encontrará imágenes con licencia exclusiva para investigación y enseñanza.
- Se realizarán pruebas de estrés sobre el aplicativo y el monitoreo de estado del software, las observaciones y conclusiones serán analizadas y anotadas en el documento final.
- Los entregables serán:
    - Documento Final de Proyecto de Titulación, el cual incluye la información más relevante de la investigación: Introducción; estudio del problema; antecedentes; justificación; objetivo general y específico; proceso de estudio; pruebas y conclusiones; anexos y referencias.
    - Manual Técnico de nuestro aplicativo.
    - Manual de Usuario de nuestro aplicativo.
    - El aplicativo se encuentra instalado en un servidor dentro del Grupo de Investigación de la Universidad Politécnica Salesiana.

- Fuentes del Aplicativo.
- Tanto las fuentes como la documentación mencionada será entregada en un CD.

CAPITULO 1

1. ESTADO DEL ARTE

Debido a que los sistemas multiagentes responden dinámicamente a los cambios de su entorno, estos son y han sido aplicados a múltiples campos, tales como simulaciones de tráfico, simulación poblacional, simulación de evacuación en los aeropuertos entre otros (Sano & Fukuta, 2013).

En este sentido, el presente trabajo de investigación tiene como objetivo desarrollar un sistema interoperativo entre diferentes sistemas de información, es decir un sistema multiagente que permita mejorar el servicio de salud en el país, por lo que se hace necesario indagar sobre trabajos similares que se han realizado en otros lugares.

En Italia (Palazzo & Paolo, 2013), desarrollaron una arquitectura informática orientada a lograr altos estándares de calidad y eficiencia en el uso de datos de salud, como la obtención de la historia clínica, el resumen del paciente, diagnósticos, y exámenes realizados.

Para desarrollar este sistema se utilizó el lenguaje de marcado XML y una ontología agente basada en los conceptos de HL7, cuyo modo de funcionamiento es el siguiente: 1) Se inicia sesión en una aplicación móvil para la plataforma de cada distrito ingresando el nombre y la contraseña del usuario. 2) En este momento se establece una conexión segura con la plataforma utilizando el protocolo TSL para garantizar el acceso seguro a los datos personales y sensibles de los pacientes. 3) A continuación, el médico o personal a cargo en este caso el cliente del aplicativo, pide el resumen del ciudadano paciente y de sus registros de salud relativas escribiendo su código de impuestos o cédula. 4) El agente de puerta de enlace consulta las diferentes entidades distribuidas para encontrar la ubicación de los datos necesarios e informar al cliente donde puede recuperar los registros de salud. 5) Por último, en la aplicación cliente se reúnen todos estos datos preguntando directamente a los agentes Document Handler de las plataformas que mantienen los registros del paciente.

Como puede evidenciarse el proceso manual es sencillo, pero la creación de la plataforma es la que debe cumplir con todos los estándares de calidad que requieren los usuarios.

El trabajo realizado por Prashant Sethia, cuyo tema es High Performance Multi-agent System based Simulations (Prashant, 2011) propone un sistema multiagente y el uso de big data que se encargarán de proporcionar un nuevo marco para las aplicaciones en ejecución con miles de nodos y petabytes de datos, soportando cargas de trabajos masivas múltiples sistemas que apoyarán en la organización de miles de servicios que exige la vida real, es un sistema que simula a gran escala la necesidad de los clientes en una gran urbe.

Se basa en el análisis de comportamientos emergentes en simulaciones sociales, permitiendo una búsqueda rápida de las personas y/o situaciones, con un mínimo de fallo, lo que permitirá a las autoridades a solucionar los problemas en el menor tiempo posible.

Los autores (Isern, Sánchez, & Moreno, 2009) quiénes realizan una revisión bibliográfica de algunas de las aplicaciones de sistemas multiagentes en el área médica y concluye que actualmente existe un creciente interés de los investigadores para desarrollar prototipos o software superiores a los clásico o ya establecidos que presentan deficiencias en su comportamiento y su accionar.

Sin embargo, indica que hasta ahora son solo prototipos y que los aspectos de seguridad, reutilización, modularidad, personalización y mantenimiento todavía debe trabajarse, sobre todo en entornos reales, en todo caso propone que se desarrolle agentes multi propósitos, considerando que son entidades que debe ser capaces de percibir la física o mundo virtual alrededor de lo real, utilizando sensores que sean capaces de reconocer y filtrar eventos o informaciones que lleven a programaciones de aplicaciones de interfaces o servicios de forma directa.

En América Latina existen pocos ejemplos de propuesta de sistemas de arquitectura multiagente para el sector sanitario, los autores (Betanco, Cruz, & Yamaira, 2013) proponen una arquitectura general para la construcción de sistemas multiagente, para el sector sanitario, con el objeto de proporcionar un veloz procesamiento de datos y el apoyo en la toma de decisiones médicas.

Para lograr este propósito utilizan la metodología MasCommonKads (Iglesias, Garijo, González, & Velasco, 1996), que tiene como base la construcción de sistemas inteligentes, utilizando la plataforma Java Agent DEvelopment Framework (JADE) 4.1 (Bellifemine, Caire, & Greenwood, 2007), soportado por la herramienta

Enterprise Architect 7.1 (Sparx Systems, 2015), Drools Expert 5.4.0 (Proctor, 2010) como motor de inferencia, el entorno de desarrollo Eclipse 3.6 (Eclipse Foundation, 2004) y PostgreSQL 9.1 (PostgreSQL Global Development Group, 1996).

Como conclusiones del trabajo se expresa que no todos los agentes se encuentren en contacto con el usuario y que existen agentes intermediarios para organizar eficientemente la información.

En general, la arquitectura propuesta para este trabajo consta de dos contenedores: el contenedor 1 (principal) con los agentes que proporciona JADE (AMS DF) y el contenedor 2 que es el agente planificador y los agentes específicos según los requerimientos.

Desarrollan el modelado e implementación de una aplicación informática basada en un sistema multiagente como herramienta de apoyo a programas de prevención de enfermedades de transmisión sexual (Callejas, Parada, & Alarcón, 2012).

Para desarrollar esta arquitectura se utilizó la metodología Ingenias y se usó la plataforma JADE, la diferencia con otros estudios es que elaboran un módulo consulta en donde los usuarios seleccionan los síntomas que tienen o que requieren consultar y los agentes se comunican y emiten un diagnóstico que sirve de referente para determinar la enfermedad y a la vez se informa de los procesos que debe seguir el paciente para controlar la enfermedad.

Los módulos del sistema WEB Multiagente son: 1) El módulo administrador, el que gestionará la información de enfermedades y síntomas que se presentan, y 2) el módulo consulta.

Nuestro aporte en este trabajo es proponer una arquitectura multiagente capaz de permitir el acceso a los reportes médicos e imágenes médicas de los pacientes de una manera distribuida, eficiente y confiable. Además de beneficiarnos de la escalabilidad, robustez y alta modularidad que nos ofrece las nuevas infraestructuras orientadas a la carga y procesamiento masivo de datos.

1.1. Marco teórico

1.1.1. Agentes

Es “una abstracción de software, una idea o concepto similar a los métodos, funciones y objetos en la programación orientada a objetos”, refiriéndonos al campo de esta investigación se diría que la definición contiene tres elementos que lo

conforman que son: Un efecto, una acción y una respuesta de otro (Alegsa, 1998 - 2015)

De acuerdo con Shoham, enunciada por (Aguilar, Rios Bolivar, Hidrobo, & Cerrada, 2012) “Usualmente, cuando la gente usa el término agente se refiere a una entidad que funciona continua y autónomamente, en un entorno en el cual otros procesos ocurren y existen otros agentes”, es decir, los agentes son entes que interactúan entre sí.

Otra definición indica que es una entidad que percibe sus entorno y actúa bajo estas percepciones (Russell & Norvig, 2008).

En síntesis, es un sistema informático que se encuentra situado en un ambiente, en donde los involucrados, agentes, pueden realizar acciones en forma autónoma para lograr objetivos específicos según la ramas de formación en las que se quiere implementar.

1.1.1.1. Descripción práctica de un agente

Un agente de conformidad con las funciones para los que fue diseñado puede ser caracterizado por:

Las tareas: Constituyen las actividades que ejecuta el agente de conformidad con los objetivos y las metas a cumplir.

Los conocimientos: Son las reglas que sigue el agente para realizar sus tareas, los cuales pueden ser adquiridas por el diseñador del sistema o incorporadas según las necesidades y la evolución del mismo, además se consideran otras fuentes de conocimiento o de sus propios mecanismos de aprendizaje.

La comunicación que utiliza en su intercomunicación con el ambiente y con los otros agentes por lo que es necesario definir el tipo de lenguaje y los protocolos de comunicación.

1.1.1.2. Características de agentes

Entre las principales características de un agente se pueden citar las siguientes (Ternero, 2009 - 2010):

- *Reactividad.* Es reactivo ya que mantiene una interacción permanente con su medio o con el entorno y es coherente con los cambios que ocurren en ese contexto. Esta respuesta inmediata es porque tiene la capacidad de percibir una

señal en el entorno de forma precisa y su acción es rápida en tiempo real, en donde no se aplican reglas complejas, es decir el sistema es sencillo y eficaz.

- *Proactividad.* La proactividad es la capacidad que tiene el individuo para tomar iniciativas que no estaban programadas con el fin de alcanzar un propósito o un fin en sí mismo, este atributo en la sistema multiagente se hace evidente cuando se diseña un sistema simple, en base a estímulos y respuestas, construyendo todos los escenarios que sean posibles.

  Los agentes, en este caso, actúan por las personas y deben saber tomar decisiones que permitan alcanzar las metas propuestas o los objetivos del sistema, por lo que deben ser diseñados de manera tal que tomen iniciativas, planifiquen acciones y los realicen de forma puntual y efectiva.
- *Comunicación.* Es la capacidad de conversar, intercambiar información entre varias personas e individuos, en este caso, la comunicación se realizará entre los agentes diseñados en el sistema por lo que se requiere crear un lenguaje informático que permita este intercambio de información, entre la "comunidad de agentes".

  Este lenguaje generalmente está compuesto de dos partes: el envoltorio compuesto por los agentes emisores, los agentes receptores, el lenguaje y el protocolo y; la segunda compuesta por el contenido del mensaje, que a su vez está compuesta por una etapa pre-formativa que indica la acción del mensaje y la frase construida de acuerdo a la ontología que son las clasificaciones o la categorización o agrupación de la información en clases.
- *Sociabilidad.* El sistema multiagentes tiene una característica especial de mantener la sociabilidad entre los agentes involucrados, lo que significa que deben poseer la capacidad de obrar y cooperar recíprocamente entre ellos mediante una comunicación directa, puesto que la ser un sistema unos agentes necesariamente tendrán que comunicarse con los otros, puesto que para realizar una acción deben cumplir con protocolos en los cuales unos tienen responsabilidades que deben ser cumplidas de manera cooperativa y colaborativa. En síntesis, en un sistema informático, multiagente, no existen logros individuales, los logros o los objetivos son sociales.

- *Movilidad.* Exactamente como sucede en un sistema humano, los agentes no son permanentes, deben tener la capacidad de movilizarse, dentro de su entorno, es decir, pueden migrar, cambiar de ubicación dentro de una red de nodos de procesamiento de datos, por lo que se debe contar con sistemas de nombramiento y localización de conformidad con los sistemas distribuidos.
- *Autonomía.* Se considera que es la principal característica de los agentes, ya que ellos deben tener la capacidad de actuar sin la intervención humana ni de otros sistemas extraños, lo que ayuda que tengan un comportamiento efectivo propio y reaccionar a los estímulos externos basados en los estímulos internos.
- *Veracidad.* Los agentes por su diseño y objetivo de creación, no pueden comunicar información falsa, por lo que es imposible que mientan y lo contrario los agentes proveen resultado veraz de actuación.
- *Benevolencia.* Los agentes al prestar sus servicios en post de los demás agentes, están demostrando benevolencia, solo actúan en conflicto si no se cumple con los objetivos para los que fueron diseñados, por lo que todos los agentes tienen un espíritu de colaboración y trabajo en conjunto.
- *Racionalidad.* El sistema multiagente actúa bajo un principio de racionalidad, ya que no ejecuta acciones que no le lleven a cumplir sus objetivos, en vista de que ellos tienen un conjunto de objetivos predefinidos y lo llevan a cabo con puntualidad.
- *Inteligencia.* Los agentes del sistema utilizan redes neuronales artificiales, para que logren desarrollar la capacidad de acumular conocimiento, ordenar ideas, llegar a conclusiones, tomar decisiones y actuar de forma autónoma frente a los requerimientos de la red.
- *Adaptativo.* Los agentes desarrollan la capacidad de cambiar su comportamiento basado en experiencias previas y las percepciones que han hecho sobre su entorno, por medio de sensores y actuar racionalmente sobre ese entorno.

1.1.1.3. Taxonomía de un agente

Taxonomía de un agente.

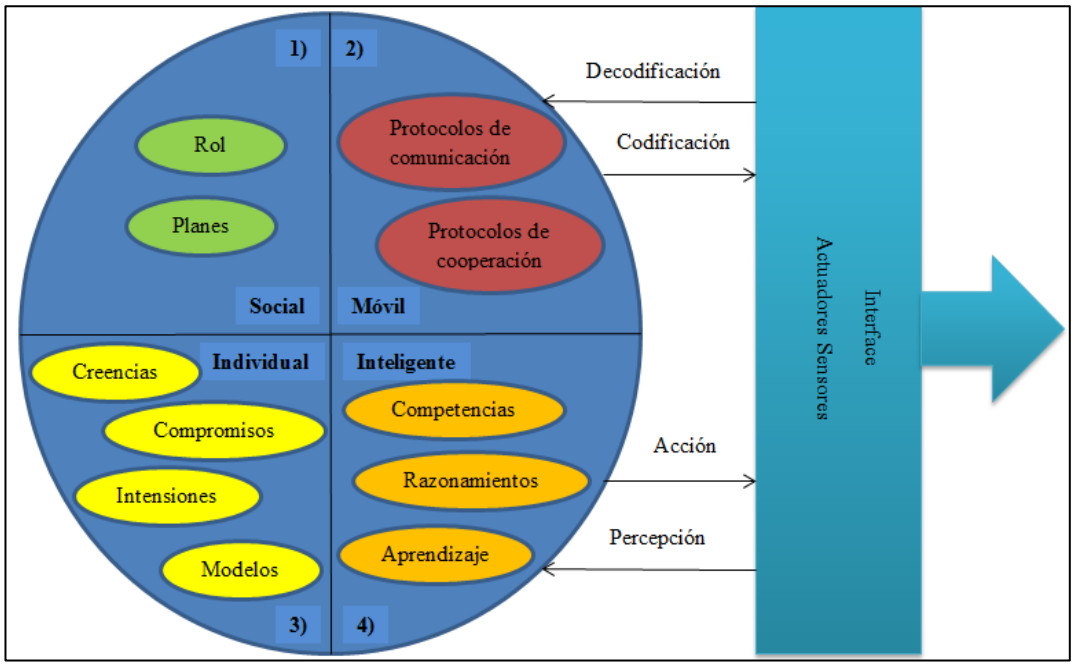


Figura 1. Bloque de una taxonomía basado en características de un agente

Fuente: (Aguilar, Rios Bolivar, Hidrobo, & Cerrada, 2012)

Elaborado por: (Sánchez & Castro)

En la Figura 1 los elementos y las propiedades de un agente dividida en cuatro cuadrantes (Aguilar, Rios Bolivar, Hidrobo, & Cerrada, 2012):

1) En el cuadrante superior izquierdo nos enseña que los agentes pueden interactuar con otros agentes, podemos encontrar el rol de un agente entre su entorno, para efectuar el agente realizará diferentes planes para su finalidad, entre sus características pueden ser: benevolencia, sociabilidad, movilidad, reactividad, comunicación.
2) En el cuadrante superior derecho podemos encontrar los agentes móviles que permiten que los agentes puedan relacionarse entre su entorno, para su intercomunicación tiene sus diferentes protocolos de comunicación y las herramientas de coordinación, que nos permiten realizar sus entornos de ejecución, entre sus características pueden ser: adaptativo, inteligencia, racionalidad, reactividad, comunicación, sociabilidad, movilidad, benevolencia.

3) En el cuadrante inferior izquierdo se presenta lo individual que un agente puede llegar a ejecutar tareas de un proceso, esto conlleva a los objetivos de creación que tiene un agente, sus intenciones, creencias y compromisos, entre sus características pueden ser: autonomía, proactividad, movilidad, veracidad.
4) En el cuadrante inferior derecho se encuentran los elementos que permiten al agente tener un comportamiento autónomo e inteligente, el agente puede llegar a realizar razonamiento para poder usar de manera eficiente el conocimiento que tiene almacenado, puede llegar a tener aprendizaje de tal manera que puede adquirir conocimiento de su entorno y es vital para la toma de decisiones inteligentes, entre sus características pueden ser: autonomía, inteligencia, adaptativo, racionalidad, benevolencia, veracidad, movilidad, comunicación, proactividad, reactividad.

1.1.1.4. Arquitectura de un agente

Los tipos de arquitectura según la forma de abordar la construcción del agente pueden ser los siguientes (Caparrini, 2014):

- *Arquitectura de agentes reactiva.* Son sistemas que tratan jerarquías de tareas en función de niveles de abstracción, se generan sin la necesidad de aplicar modelos simbólicos y se podrán utilizar en algunos sistemas complejos, la característica principal es que se desarrollan en tiempo real.
- *Arquitectura de agentes deliberativa.* En ésta arquitectura se puede llegar a expresar el ambiente, comportamiento en técnicas de conocimiento representados simbólicamente, las decisiones son en base a la deducción lógica.
- *Arquitectura de agentes híbrida.* Adoptan agentes reactivo y deliberativo, del modelo reactivo se podría indicar que reacciona en eventos de entorno sin la necesidad de invertir en razonamientos; y del modelo deliberativo se encarga de planificar, efectúa tareas de abstracción superior, son agrupados en dos niveles, los de información a nivel abstracto, representación simbólica del ambiente y los de comportamiento social y planificación del alto nivel.

Se puede clasificar otro tipo los agentes: *Agente reactivo simple***,** que cumple con la función de percepción – entrada – acción y entorno; *Agente reactivo basado en el modelo,* se compone de un agente entrada – acción – predicción – estado entorno;

*Agente por objetivos***,** está compuesto por un agente entrada – objetivos – acción – predicción – estado entorno; *Agente basado en la utilidad***,** es un agente diseñado según la necesidad (Hyacinth S & Divine T, 1996) y (Weiss, 1999).

1.1.2. Sistemas multiagentes

Un sistema multiagente (SMA) es una técnica, una ciencia; es una ciencia ya que se fundamenta en postulados epistemológicos tanto de la lógica formal de la teoría de sistemas, como del constructivismo como corriente que explica la realidad desde varios puntos de vista y que concibe que un sistema es un ser vivo que interactúa entre sus elementos o en este caso de los diferentes agentes que constituyen la red de un sistema.

Es una técnica ya que un sistema multiagente es un sistema distribuido entre nodos en este contexto, es importante hablar que es un agente, para entender lo que es un sistema multiagente (Sethia, 2011).

1.1.2.1. El estudio de sistemas multiagente

El estudio de los sistemas multiagente se encarga de tratar sobre el desarrollo y análisis de resoluciones a problemas sofisticados y arquitecturas de control tanto para sistemas con un único agente como sistemas multiagente.

Los temas de investigación en SMA son: ingeniería de software orientado a agentes; creencias, deseos e intenciones (en inglés Beliefs, Desires and Intentions, BDI); cooperación y coordinación; organización; comunicación; negociación; resolución de problemas distribuida; aprendizaje de multiagentes; comunidades científicas fiabilidad y tolerancia a fallos (Ecured, 2014)

1.1.2.2. Metodologías de agentes

Entre las más significativas metodologías utilizadas en los sistemas multiagentes pueden ser (Botti & Julián, 1995):

- GAIA de Michael Wooldridge y Nick Jennings, se basa en la organización de roles, que permiten la comunicación, permanecen comunicados con otros roles e interaccionan entre ellos.
- MASE de Scott A. Deloach, expone agentes como extensiones de objetos, utiliza la herramienta AgentTool.

- MADKiT por Jacques Ferber, hecho en JAVA, se basa en el paradigma Agente-Role-Organización.
- INGENIAS del grupo GRASIA de la UCM, forma parte de la metodología MESSAGE, provee un conjunto de herramientas para modelar y crear código de sistemas multiagente.
- Mas-CommonKADS de Carlos Iglesias en la UPM, es una amplificación de la metodología CommonKADS, la diferencia está que soporta metodología orientada a objetos, de igual manera sobrelleva lenguajes de especificación de protocolos como SDL. Cabe destacar que hay otras metodologías de agentes, se describieron algunas de las principales.

1.1.3. Big Data

Éste sistema no es nuevo, fue Doug Laney del Meta Grup hoy denominado Gartner, quien dio la importancia de indagar el incremento de datos a gran velocidad, variedad y volumen (SAS, 2015).

En la actualidad las empresas proveedoras de datos masivos tratan de encontrar soluciones para atender la gran cantidad de datos, cabe destacar que el almacenamiento de datos crece cada cuarenta meses desde los años ochenta, este crecimiento en algunas ocasiones llegan a afectar a los motores de búsqueda de internet, en los cuales no se ha logrado controlar la información.

Big Data se refiere a la acumulación de grandes cantidades de datos sean estructurados o no estructurados, los grandes datos más importantes pueden conllevar que las empresas puedan realizar análisis para mejorar su rendimiento y sean procesados en un tiempo razonable. Se debe tener en cuenta que los problemas en este tratamiento se concentran en la recolección, almacenamiento, búsqueda, análisis y visualización de los datos (SAS, 2015).

Con el crecimiento de información se puede realizar una infografía, esto permite una visualización amigable de resultados estadísticos en gráfico o mapas que en tablas con números y conclusiones que para varios no significa mucho, esto puede crear un escenario más amigable, menos complicada y con una gran capacidad de informar de manera clara y precisa (Infochannel, 2012).

- **Aportes Big Data en la investigación.-** Los principales son los siguientes: i) *El Language, Interaction and Computation Laboratory (CLIC)*, se encargan de investigar la información verbal y no verbal con técnicas computacionales y cognitivas; ii) *Lineberger Comprehensive Cancer Center - Bioinformatics Group*, su función es la de analizar la información producida por investigadores relacionados con el cáncer; iii) *El PSG College of Technology – India*, se encargan de llevar a cabo un análisis de secuencias de proteínas, con lo cual ayuda a demostrar los enlaces evolutivos y pronosticar estructuras moleculares (IBM, 2012). Hay muchos aportes en la actualidad sobre el uso de Big Data en la actualidad, se acaban de mencionar unas pocas.
- **Componentes de una plataforma big data.-** Hay diferentes tipos de plataformas que ayudan a los problemas de manejo a grandes cantidades de datos de información, la plataforma base que las organizaciones toman en cuenta es Hadoop.

1.1.4. Hadoop

Hadoop es un framework creado por Apache de código abierto para elaborar aplicaciones sobre grandes clusters (Merchan, Plaza, & Moreno, 2011).

1.1.4.1. Modos de funcionamiento de Hadoop

Hadoop funciona de tres modos (Merchan, Plaza, & Moreno, 2011):

- *Modo local (Standalone).* Por defecto, está en modo no-distribuido el cual se puede usar para ejecutar los ejemplos que hay por defecto de hadoop.
- *Modo psuedo-distribuido.* Se puede ejecutar en un único modo, hadoop puede ejecutar diferentes procesos de JAVA.
- *Modo distribuido.* Se puede ejecutar en total capacidad hadoop, los servicios de Hadoop son ejecutados en diferentes nodos de un clúster. Hadoop básicamente es escalable, económico, eficiente, confiable, es fácil de usar, tiene tolerancia a fallos.

1.1.4.2. Características de Hadoop

Entre las características de Hadoop tenemos (IBM, 2012) y (TurnkeyConsulting, 2014):

- *Datos estructurados.* Su longitud, formato, fechas, como cadena de caracteres, están completamente bien estructurados, se pueden decir que las bases de datos relacionales y hojas de cálculo son ejemplos principales de éste tipo.
- *Datos no estructurados.* No tienen una estructura específica, la información es recolectada tal y como fueron recogidos. Sus datos no pueden ser guardados en una tabla dado que la información no puede ser desglosada en forma elemental. Documentos de texto, archivos del tipo pdf, e-mails, son ejemplos básicos de éste tipo.
- *Datos semiestructurados.* Posee datos que contienen marcadores que ayudan en la separación de sus elementos, los cuales contienen sus propios metadatos semiestructurados, HTML, XML, serían los ejemplos principales de estos tipos de datos.
- *Machine to Machine (M2M).* En este tipo de datos la información se comparte con dispositivos que puedan capturar un evento específico, entre los ejemplos podemos mencionar a la comunicaciones inalámbricas como el Wi–Fi, el Bluetooth, GPS.
- *Análisis de los datos obtenidos.* Para analizar los datos se utilizan diferentes técnicas como:
- *Asociación.* Cuando se relaciona diferentes variables, la causalidad es la premisa que permite encontrar una predicción en el comportamiento de otras variables.
- *Minería de datos.* Se trata de encontrar comportamientos predictivos, engloba el conjunto de técnicas que combina métodos estadísticos y de machine learning con almacenamiento en bases de datos. Está estrechamente relacionada con los modelos utilizados para descubrir patrones en grandes cantidades de datos.
- *Agrupación.* Este tipo de análisis divide a los grandes grupos en partes más pequeños para lo cual es indispensable encontrar similitudes entre estos grupos y el descubrimiento de nuevos, conociendo cuáles son las cualidades que lo definen. Es una metodología apropiada para encontrar relaciones entre resultados y hacer una evaluación preliminar de la estructura de los datos analizados. Existen diferentes técnicas y algoritmos de clustering.

- *Análisis de texto.* Esta metodología permite extraer información de estos datos y así modelar temas y asuntos o predecir palabras, los datos provienen de emails, búsquedas web o contenidos.

1.1.4.3. Arquitectura de Hadoop

Partes básicas de una arquitectura Hadoop.

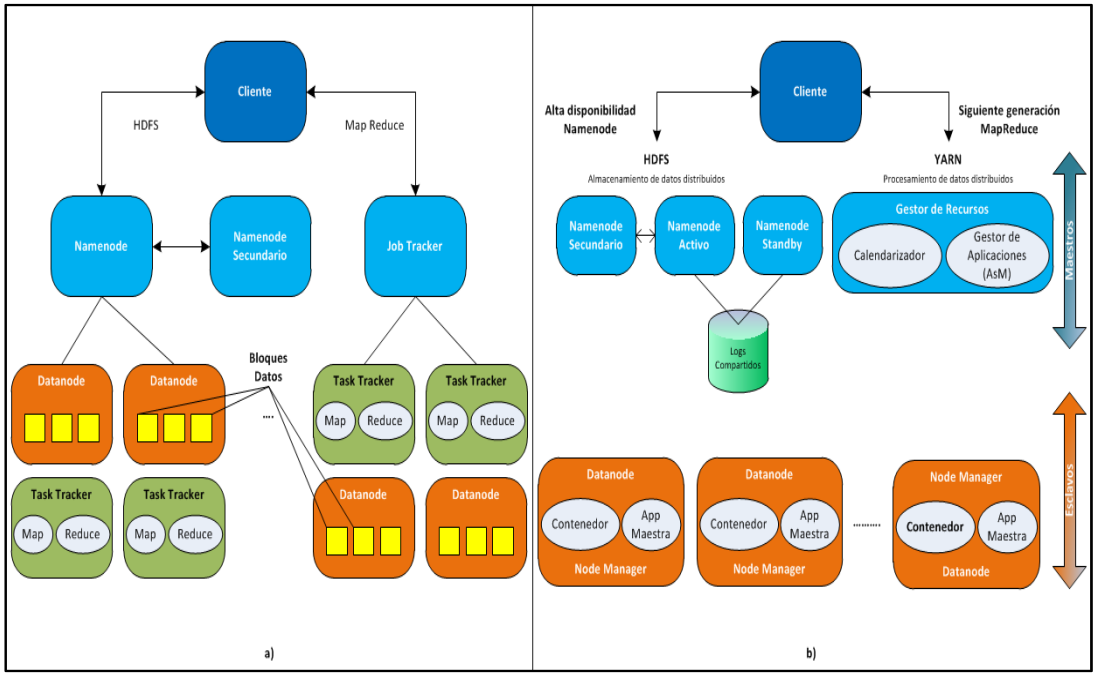


Figura 2. a) Arquitectura de Hadoop versión 1.x, b) Arquitectura de Hadoop versión 2.x

Fuente: (Edureka, 2014)

Elaborado por: (Sánchez & Castro)

En la Figura 2 a), se representa la arquitectura Hadoop en sus primeras versiones en la cual se puede observar el funcionamiento que se realizaba con HDFS y MapReduce. En la Figura 2 b), se representa la arquitectura Hadoop en sus versiones a partir de la 2.x con la cual se aumenta en el funcionamiento Yarn, disponible para su descarga en https://hadoop.apache.org/releases.html#Download

Las arquitecturas están compuestas de cuatro módulos principales que serían: i) *Hadoop Distributed File System (HDFS)*, es el sistema de archivos distribuido que fue planteado para ejecutarse a nivel de hardware, es aplicado cuando se obtienen grandes agrupaciones de datos*;* ii) *Hadoop MapReduce*, es el motor de hadoop, funciona en nodos que realizan las opciones de mapeo donde se recoge los datos, reduce que se encarga de procesar y generar resultados; iii) *Hadoop Common:* ayuda

a la incorporación de librerías que soporta sub-proyectos de Hadoop; iv) *Yarn:* gestiona recursos y trabaja en planificación y monitoreo de forma separada.

- **Hadoop Distributed File System (HDFS)**

Arquitectura HDFS.

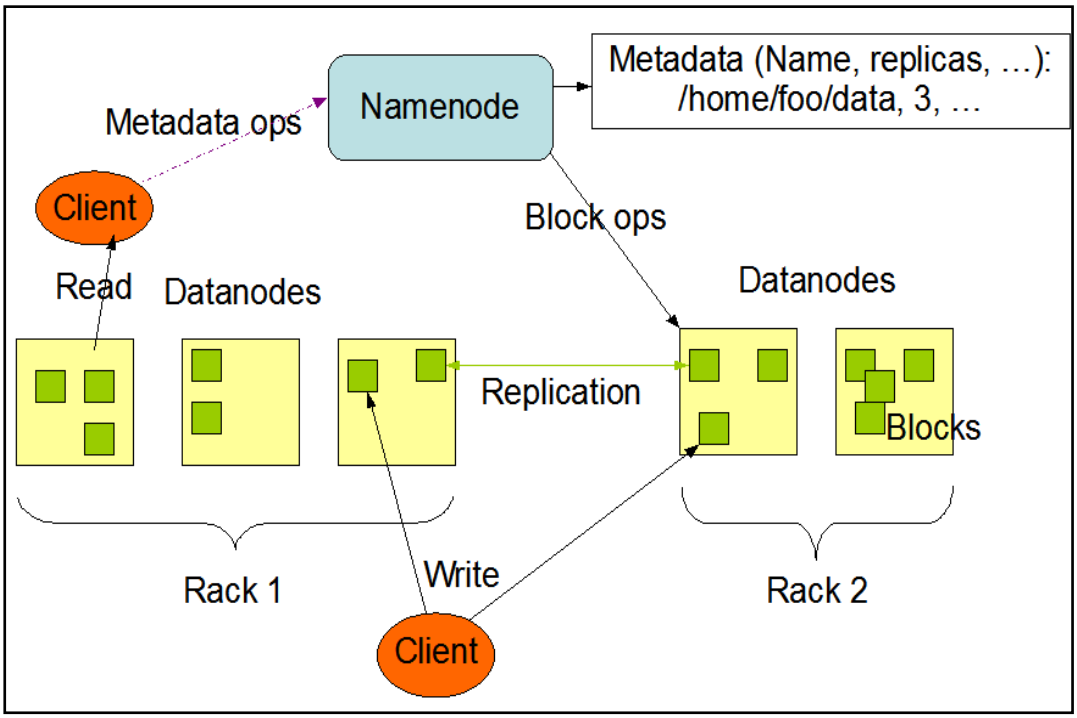


Figura 3. Partes de una arquitectura basada en HDFS

Fuente: (Hadoop, 2014)

En la Figura 3, se puede observar que la arquitectura contiene los nodos de información llamados *DataNodes*, cada nodo se encarga de enviar bloques de información a través de la red, pueden enviar la información por medio de HTTP, que se pueden obtener por medio de un navegador web, son coordinados por el *NameNode* (ApacheHDFS, 2015).

El *NameNode* es el encargado de guardar las alteraciones del sistema de ficheros en logs, almacena metadatos con la información necesaria de donde están almacenados, el número de réplicas que permiten la tolerancia a fallos.

El *NameNode* se encarga de tomar decisiones por cada replicación de los bloques, cada *DataNode* recibe un *Heartbeat* esto quiere decir que funciona el respectivo *DataNode*, a la vez recibe un *Blockreport* de los *DataNodes* que se encuentran en los

clústeres, al recibirlo se obtiene una lista de todos los bloques del *DataNode* (ApacheHDFS, 2015).

- **MapReduce**

Arquitectura MapReduce.

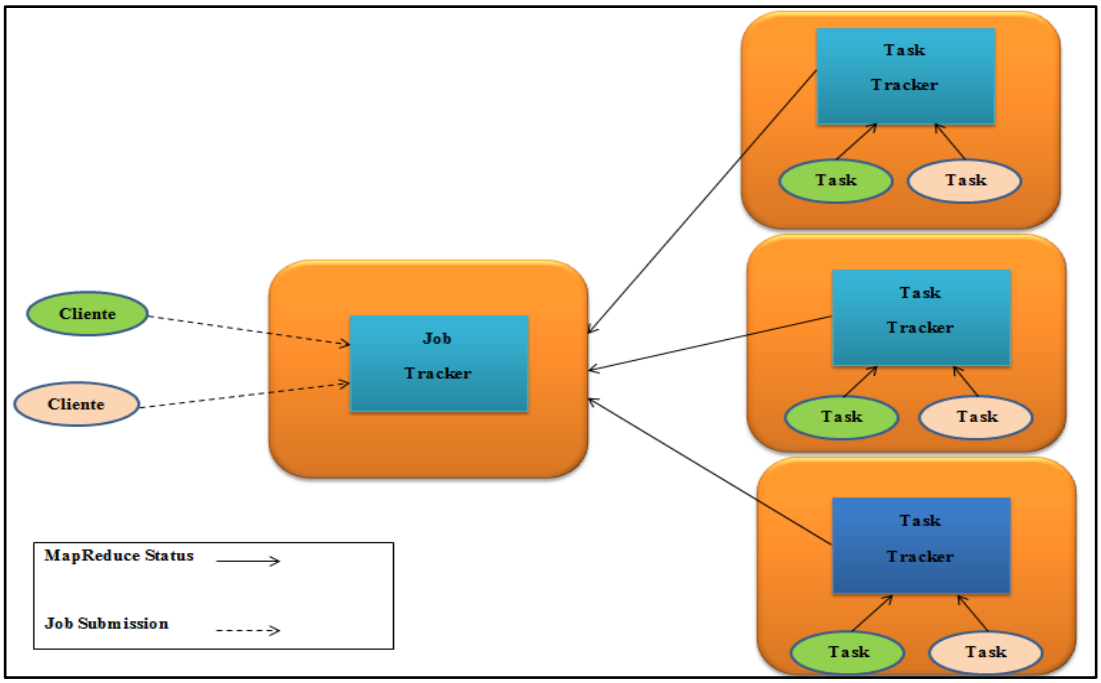


Figura 4. Funcionamiento de MapReduce

Elaborado por: (Sánchez & Castro)

En la Figura 4, se puede observar la Arquitectura de base MapReduce, el cual nos permite obtener tolerancia a fallos, esto quiere decir que si un nodo falla en el proceso, la tarea es reasignada a otro nodo que haya, tomar en cuenta que se puede obtener un máximo de clusters 4000 nodos y un máximo de 40000 de task concurrentes (Hortonworks, 2011).

En base a la Figura 4, podemos determinar los componentes principales como son: *JobTracker*, se encarga de gestionar los recursos del clúster, luego se encarga de enviar la carga de trabajo al nodo *TaskTracker*.

Se puede ofrecer soporte al procesamiento en paralelo, escribe y ejecuta aplicaciones que procesan gran cantidad de información en paralelo. Consta de dos fases, la fase Map, la fase Reduce. A continuación se detalla el funcionamiento de las fases antes mencionadas (Merchan, Plaza, & Moreno, 2011).

Funcionamiento MapReduce.

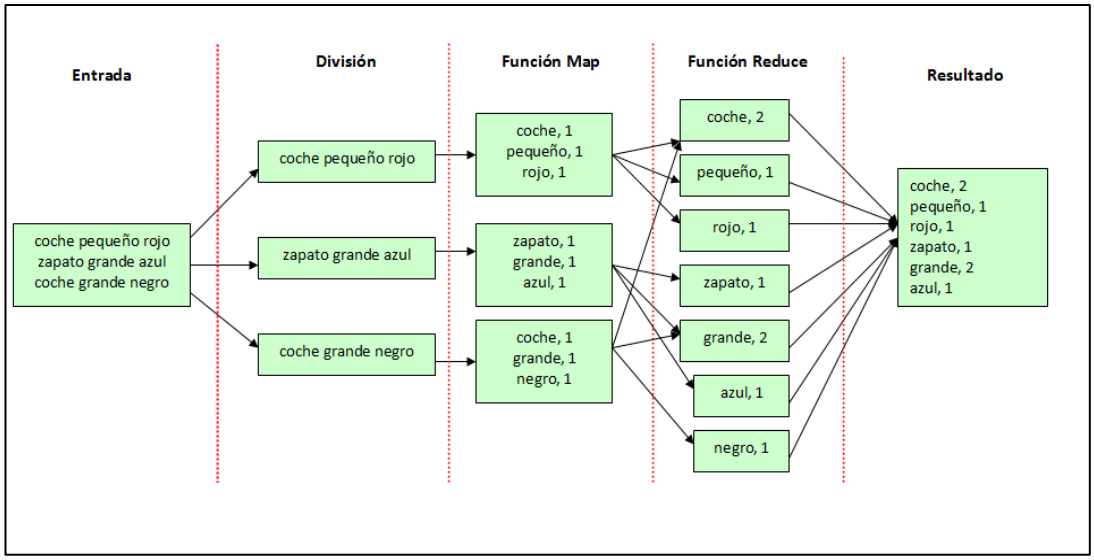


Figura 5. Esquema del funcionamiento MapReduce

Fuente: (Montero, 2015)

Para poder entender el funcionamiento de Map y Reduce en la Figura 5 se puede obtener un ejemplo de su funcionamiento.

El archivo de *Entrada (input)* contiene los datos de texto en diferentes líneas, la fase map se encarga de transformar los datos de entrada en datos intermedios, que pasan a ser divididos, en la parte de *División* se toma cada línea de texto y los separa en bloques, los cuales pasan a la *Función Map* que se encarga de escribir salidas en repositorios temporales y mapea los datos en pares clave/valor correspondiente a cada proceso de la palabra, en esta fase cada palabra se le asigna un valor, luego de esto pasa a la *Función Reduce*, donde todos los valores que están asociados con la misma clave se agrupan y se ponen a disposición para el proceso reduce, el cual se ejecuta en paralelo, esto no hace sino aumentar el "valor" de una clave para determinar la frecuencia de esta palabra y se obtiene el *Resultado final*, (Montero, 2015).

- **Hadoop Yarn**

Arquitectura Yarn.

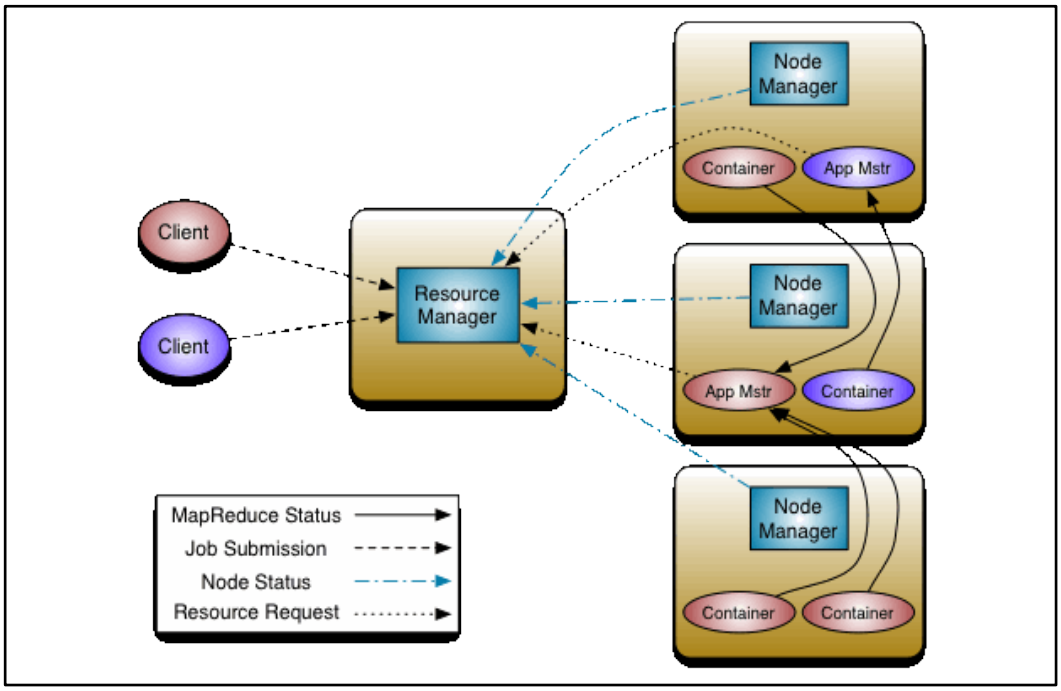


Figura 6. Estructura de la arquitectura Yarn

Fuente: (Hadoop, Apache, 2015)

En la Figura 6, se demuestra las partes fundamentales de la arquitectura Yarn en la cual contiene *JobTracker* (NameNode) se divide en dos formas: i) *ResourceManager*, se encarga de toda la gestión de recursos, ii) *ApplicationMaster*, se encarga de la planificación/monitoreo de las tareas.

Se puede determinar que el *ResourceManager* y el *NodeManager* de cada nodo constituyen el entorno de trabajo, el *ResourceManager* se encarga de repartir y gestionar los recursos entre todas las aplicaciones del sistema, el *ApplicationMaster* se encarga de la negociación de recursos con el *ResourceManager* y los *NodeManager* para poder ejecutar y controlar las tareas (ApacheYarn, 2015).

1.1.4.4. Distribuciones Hadoop

Big Data tiene algunas distribuciones que son de códigos abiertos y gratuitos, a continuación se describen las distribuciones disponibles como (IBM, 2012) y (TurnkeyConsulting, 2014):

- *Avro.* Proporciona servicios de serialización, creado por Apache y licencia libre. Al guardar los datos en un archivo, el esquema define el archivo y es guardado internamente; esto ayuda para cualquier aplicación pueda ser leído posteriormente.
- *Cassandra.* Se puede llegar a decir que es una base de datos no relacional distribuida, creada en Java. Soporta grandes volúmenes de datos en forma distribuida, creado por Apache y licencia libre.
- *Chukwa.* Planteado para analizar, y a la vez se encarga de coleccionar ampliamente logs, contiene toolkit para visualizar los respectivos efectos del análisis y monitoreo, creado por Apache.
- *Flume.* Lleva datos citados en la fuente hacia otra localidad. Creado por Apache, contiene tres objetos principales: i) *Unsource* es una fuente de datos, ii) Undecorator es la operación del flujo de datos encargado de convertir la información, iii) Sink es el trayecto de una acción en particular.
- *Sqoop.* Transfiere eficientemente bulk data entre Hadoop y sistemas de almacenamiento con datos estructurados, se puede importar base de datos enteros a HDFS, basado en Apache.

2.1.4.5 Proyectos relacionados con Hadoop

Entre los proyectos disponibles se pueden citar (IBM, 2012) y (TurnkeyConsulting, 2014):

- *HBase.* Es una base de datos no relacional, no resiste SQL, se debe tomar en cuenta que las filas y las columnas de las tablas respectivamente admiten atributos, los cuales pueden llegar a ser agrupados, HBase fue creado por Apache.
- *Hive.* Ayuda a la administración de datos almacenados en ambientes distribuidos. El lenguaje utilizado se denomina HQL (Hive Query Language), el cual tiene la opción de enviar procesos MapReduce que se producen en clusters de Hadoop, es basado en Apache.
- *Lucene.* Usado principalmente para realizar búsquedas sobre textos, es un proyecto de Apache. Contiene librerías para indexación y búsqueda de texto. Utilizado en la implementación de motores de búsqueda.

- *Oozie.* Ayuda a la simplificación del flujo de trabajo que se dan entre los procesos. El usuario es capaz de detallar acciones, y los resultados expresados por aquellas acciones, es de código abierto.
- *PIG.* Fue creado para la ayuda de Hadoop para analizar los conjuntos de datos y no gastar tiempo en la construcción de programas MapReduce, maneja cualquier tipo de dato, basado en Apache.
- *Mahout.* Se encarga de aprender automáticamente con data mining empleando Hadoop, bajo licencia Apache.
- *ZooKeeper*. Contiene una infraestructura centralizada, facilita servicios que permiten la sincronización de clúster y está bajo licencia Apache.

En síntesis, Hadoop es una plataforma de Big Data, que radican en ambiente con proyectos que unidos pueden llegar a simplificar, como nos permitirán administrar, y examinar gran cantidad de información.

1.1.5. Estándares para mensajes e imágenes médicas.

A partir de las dos últimas décadas, las organizaciones de atención de salud y los hospitales en particular, han comenzado a automatizar las áreas de manejo de su información. Al principio, estos esfuerzos se ligaron con reducir la burocracia, mejorar la contabilidad y mejor la toma de decisiones administrativas.

En los últimos años, el objetivo es fomentar un mejor servicio clínico y auxiliar, a través de sistemas secundarios que están directamente relacionados con el paciente (por ejemplo, los servicios de los centros de emergencia). Hoy en día no es raro que la mayoría de los hospitales tengan instalado sistemas informáticos para mantener la información del paciente, la transferencia de datos clínicos, de laboratorio y exámenes radiológicos.

La necesidad de la creación de un sistema completo informativo hospitalario, está constituido por los sistemas de sub heterogéneos, que hace necesario el uso de modelos. Los modelos se refieren a la comunicación, el intercambio de datos, los modelos informativos, la terminología y la seguridad. El protocolo en la terminología de la tecnología de la información son las reglas o modelos que permiten a los ordenadores comunicarse entre ellos e intercambiar información con los menos errores posibles.

2.1.5.1 HL7

Health Level Seven Internacional (HL7) fundada en 1987 por American National Standards Institute (ANSI), Normas acreditados en desarrollo Organizaciones (SDO) cometido exclusivamente al ámbito de la salud. El dominio que HL7 se concentra en datos clínicos y administrativos (Meo, Quattrone, & Ursino, 2011).

El nombre se deriva de la interconexión de sistemas abiertos (OSI) modelo de red como la capa 7 se refiere al intercambio de información, por ejemplo clave2/valor2 → clave2, valor2

Estándares HL7 definen mensajes y protocolos de intercambio de mensajes para apoyar la práctica clínica que mejoran la prestación de atención, optimizar el flujo de trabajo, reducir la ambigüedad y mejorar la transferencia de conocimientos entre todos los interesados.

HL7 internacional cree que el intercambio de datos entre el cuidado de la salud aplicaciones es esencial para el logro de ese objetivo (HL7, 2007 - 2016).

Estándares HL7 trabajan en el supuesto de que a medida que un evento se desencadena en un entorno de atención de salud, por ejemplo, ingreso del paciente, se requiere el flujo de datos entre los distintos sistemas.

HL7 es el modelo más ampliamente generalizado en el intercambio de información a través de mensajes en entorno clínico. Es compatible con la automatización de los procesos que se deben cumplir en el marco de Organismo de Salud (HL7, 2007 - 2016).

Casi todos los instrumentos de diagnóstico (equipo médico) pueden "hablar" HL7 y casi todos los sistemas médicos informativos envían y reciben los mensajes HL7, utilizando las reglas de intercambio de mensajes de (protocolo) de HL7.

El estándar HL7 admite dos protocolos de mensajes: Versión 2 y Versión 3 (HL7, 2007 - 2016).

HL7 V2.x es un protocolo para el intercambio de datos clínicos a través de mensajes. Los mensajes se codifican como cadenas de texto ASCII con delimitadores. También se ha desarrollado para la codificación de mensajes V2.x en XML. Además el HL7 versión 2, en el siguiente enlace podremos encontrar la dirección para su respectiva descarga

http://www.hl7.org/implement/standards/product_brief.cfm?product_id=213; Se basa

en datos Electrónicos Formato de Intercambio (EDI), pero las especificaciones están también disponibles en formato XML. La versión 2 fue desarrollado con el supuesto de que un evento en el mundo de la salud, llamado el "evento de disparo", ejecuta el intercambio de mensajes entre un par de aplicaciones.

Cuando se produce un evento en un sistema compatible HL7, un mensaje HL7 se prepara mediante la recopilación de los datos necesarios del subyacente sistema de aplicación y se pasa al solicitante. Por ejemplo, un evento de disparo, llamado ADT A01- Admisión / descarga / transferencia (admisión de un paciente hospitalizado en un centro): ocurre cuando un paciente es admitido a un centro y esto puede provocar que los datos sobre el paciente puedan ser recogidos y enviados a las aplicaciones que interactúan entre sí (HL7, 2007 - 2016).

2.1.5.2 DICOM

Digital Imaging and Communication in Medicine (DICOM), fue desarrollado por Asociación Nacional de Fabricantes Electrónicos (NEMA) y por el Colegio Estadounidense de Radiología (ACR), es un estándar para imágenes médicas, son usadas alrededor de todo el mundo en dispositivos de radiología, cardiología, oftalmología, odontología, para sacar como ejemplo, rayos X, CT, MRI, ultrasonidos, entre otros (Orza, Cordos, Vlaicu, & Meza, 2010).

DICOM puede almacenar imágenes 2D, posiciones 3D, el tamaño físico de los objetos que se encuentran en la imagen, de los cuales esto permite mejorar su procesamiento. A la vez puede llevar información de datos médicos, con mensajes que contengan más de 2000 atributos estandarizados que conservan datos e imágenes médicas del paciente.

Estándar DICOM es el componente obligatorio para la integración del procesamiento de imágenes y sistemas de distribución en la asistencia sanitaria. Proporciona soluciones de comunicación para aplicaciones de software médicos.

- **Funciones principales de DICOM**

Realizará almacenamiento, visualización, envío, creación, recuperación, consultas, procesamiento e impresión de imágenes médicas. DICOM tiene su propio lenguaje, integración con el estándar HL7, describir el formato de archivos y la especificación de información del paciente, formato digital de una imagen.

El formato DICOM es del tipo .dcm, en el cual DICOM puede tener varios formatos en los cuales pueden ser archivos planos como pueden tener varios fotogramas, DICOM puede soportar formatos como JPEG, JPEG 2000, LZW, JPEG LOSSLESS (Orza, Cordos, Vlaicu, & Meza, 2010) y (Herranz, 2013).

- **Estructura de un archivo DICOM.**

DICOM tiene su propio lenguaje, el cual puede llegar a contener información física como pueden ser: nombre del paciente, tipo de estudio, entre otros.

Un archivo DICOM contiene una Cabecera (Header), que nos permite guardar la información del nombre del paciente, el tipo de exploración, como los datos de la imagen con información en tres dimensiones. Contiene un DATA SET, donde va la imagen o las imágenes especificadas (Herranz, 2013).

El Data Set está formado por un Data Element el cual está constituido por: *TAG*, es la etiqueta de identificación, es el identificador único, tiene un número de grupo y un número del elemento, los cuales son en hexadecimal; *Representación del Valor (VR)*, valor de representación que indica el tipo de dato almacenado, estará definido por un diccionario de datos; *Longitud del valor (Value Lenght)*, indica la longitud del dato; *Campo de Valor (Value Field)*, van la información del paciente, análisis hechos, imágenes obtenidas, depende mucho del valor de la etiqueta (Herranz, 2013).

En la Figura 7, se puede visualizar la estructura de un archivo DICOM.

Estructura de un archivo DICOM.

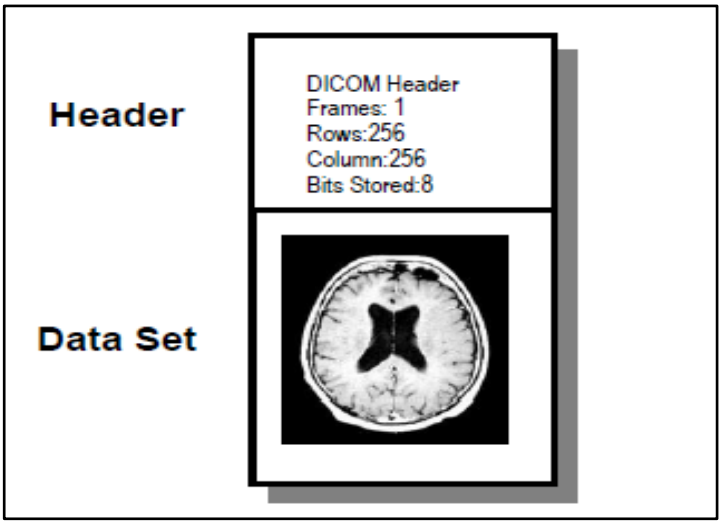


Figura 7. Ejemplo de un archivo DICOM.

Fuente: (Fic)

CAPITULO 2

2. ANALISIS Y DISEÑO

2.1. Análisis y diseño

Como buena práctica de la ingeniería de software y siguiendo el ciclo del desarrollo, se empezará realizado el análisis de los requerimientos del sistema.

En este sentido, este capítulo identificará las tareas y procesos que debe cumplir el sistema, diseñará interfaces gráficas que ayuden a cumplir con este cometido y al mismo tiempo identificará las restricciones que cumple el software.

Independientemente del método o enfoque utilizado para gestionar el desarrollo, la ingeniería de software clasifica los requerimientos en funcionales y no funcionales.

2.1.1. Requerimientos funcionales

Se define como tareas observables o funciones que deben llevarse a cabo, sus alcances y limitaciones. Los requerimientos funcionales son:

- *Interacción con los usuarios.* El sistema controla eficientemente el registro de admisiones de pacientes y el examen físico regional, para esto necesita controlar las acciones que deben realizar los usuarios con determinados perfiles o roles.
- *Funcionalidad global.* El software dispone de módulos que permiten el registro de admisiones de pacientes así como el mantenimiento de imágenes médicas, para lo cual consta con funcionalidades de ingreso, actualización y en determinadas secciones la eliminación de información.
- *Alcance.* El software automatiza el proceso de registro de admisiones de pacientes así como el mantenimiento del examen físico regional creación, actualización y eliminación de imágenes médicas.

El alcance del proyecto se limitará a las admisiones del paciente (formulario HCU-001 anverso) y el examen físico regional (formulario HCU-002 anverso sección 7). Este proyecto no abarcará más formularios.

2.1.2. Requerimientos no funcionales

Define la interacción con el usuario, la apariencia, el mantenimiento del sistema y la operatividad. Los requerimientos no funcionales en el proyecto técnico son:

- El sistema se despliega en un entorno web Java EE, para un mayor acceso de usuario, sin limitaciones a una ubicación física, se utilizará el framework JavaServer Faces y el compilador java.
- Se ofrece una interfaz amigable, y de navegación intuitiva.
- El acceso al aplicativo, y a determinadas opciones del menú serán controlados por perfiles de usuario.
- Se trata en lo posible de remplazar las tareas manuales, por automáticas. Se disminuirá así el consumo de papel y se mejorará el tiempo de respuesta.
- El sistema usa una base de datos relacional MySQL y un sistema de archivos distribuido Hadoop.
- Las pruebas de estrés al software son llevadas a cabo por la herramienta JMeter.
- El monitoreo de eficiencia sobre el sistema de archivos distribuidos son realizados con la herramienta Ganglia.

2.2. Identificación de actores

Es necesario identificar los roles y las actividades de los actores en el aplicativo como se muestra en la tabla 1.

Tabla 1. Identificación de actores.

Actor	Descripción
Paciente	Solicita atención en una localidad médica, no tiene contacto directo con el sistema.
Recepcionista.	Encargado de solicitar los datos personales de los pacientes e ingresarlos en los formularios del sistema, el rol "Recepcionista" con respecto al sistema puede ser realizado por el recepcionista en turno o bien por una persona autorizada para este cometido.
Médico.	Puede revisar y actualizar la información de los pacientes, así como visualizar, cargar y eliminar las imágenes médicas, el rol "Médico" con respecto al sistema puede ser realizado por el médico en turno o bien por una persona autorizada para este cometido.

Nota: Descripción de actores con su actividad respectiva

Elaborado por: (Sánchez & Castro)

2.3. Identificación de casos de uso

Para una representación visual entre los actores y sus actividades se emplean los casos de uso a continuación, tenemos los casos de uso generales (tabla 2) y los casos de uso por rol (tabla 3).

Tabla 2. Casos de Uso generales.

Actor	**Función**	
	Nro.	**Descripción**
Recepcionista / Médico.	CU1	Iniciar sesión.
	CU2	Cerrar sesión.

Nota: Identificación y función de los actores en general

Elaborado por: (Sánchez & Castro)

Tabla 3. Casos de Uso por rol.

Actor	**Función**	
	Nro.	**Descripción**
Recepcionista / Médico, Paciente.	CU1	Registro de admisiones primera vez.
	CU2	Registro de admisiones subsecuentes.
Médico.	CU3	Actualización examen físico regional.
	CU4	Administración imágenes médicas.

Nota: Identificación y función de los actores por su rol

Elaborado por: (Sánchez & Castro)

2.3.1. Caso de uso: Iniciar sesión

A continuación tenemos la representación visual del proceso de inicio de sesión (figura 8) y una descripción detallada de este (tabla 4).

Caso de uso: Iniciar sesión.

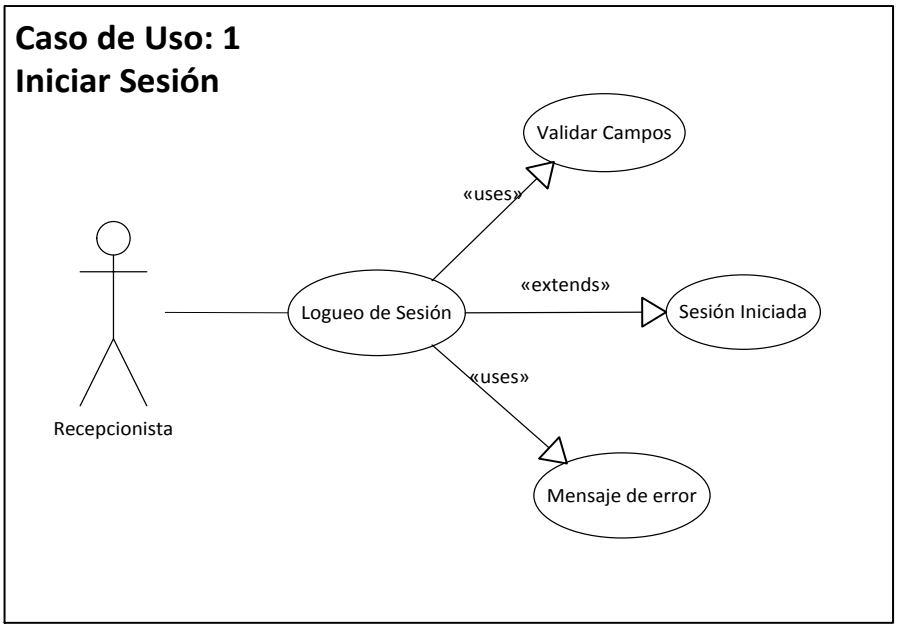


Figura 8. Descripción del proceso de inicio de sesión

Elaborado por: (Sánchez & Castro)

Tabla 4. Caso de uso: Iniciar sesión.

Nro. Caso Uso	CU1
Nombre	Iniciar sesión.
Descripción	Ingresar usuario y contraseña para iniciar sesión en el sistema.
Actores	Recepcionista
Precondiciones	Usuario existente en el sistema. No haber iniciado una sesión. Ingresar nombre de usuario correcto. Ingresar contraseña correcta.
Postcondiciones	Recepcionista logueado en el sistema.
Flujo normal	1. El recepcionista inicia el sistema. 2. El recepcionista ingresa su usuario y contraseña. 3. El sistema valida los campos ingresados.

	4. El sistema autentica al usuario. 5. El sistema crea una nueva sesión. 6. El sistema re direcciona a la página principal de la aplicación.
Flujo alternativo	1. El recepcionista ingresa datos inválidos. 2. El sistema muestra el mensaje "Usuario o contraseña incorrecta".
Excepciones	Recepcionista no creado en el sistema.

Nota: Detalles del proceso para iniciar sesión

Elaborado por: (Sánchez & Castro)

2.3.2. Caso de uso: Registro admisiones primera vez

A continuación tenemos la representación visual del proceso de registro de admisiones de pacientes por primera vez (figura 9) y una descripción detallada de este (tabla 5).

Caso de uso: Registro admisiones primera vez.

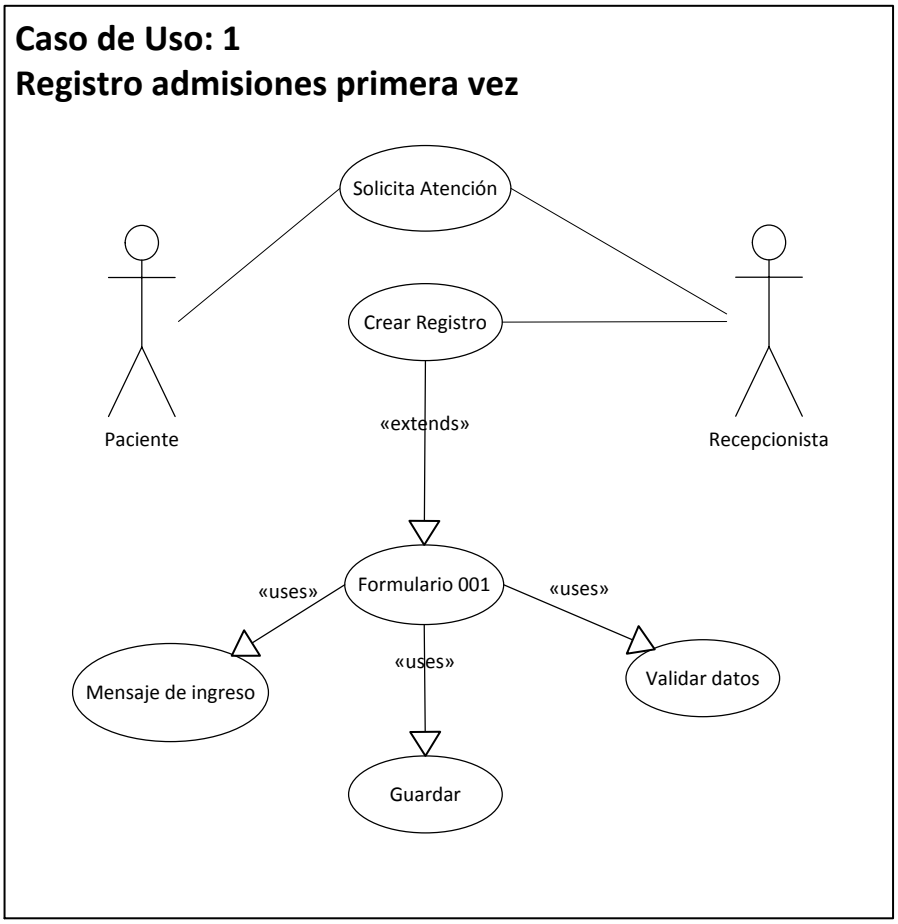


Figura 9. Descripción del proceso para admisiones por primera vez

Elaborado por: (Sánchez & Castro)

Tabla 5. Caso de uso: Registro admisiones primera vez.

Nro. Caso Uso	CU1
Nombre	Registro admisiones primera vez.
Descripción	Creación del registro de la primera admisión de los pacientes en el sistema.
Actores	Recepcionista, Paciente
Precondiciones	Ser usuario recepcionista. Iniciar sesión. El paciente no debe tener admisiones existentes en el sistema.

Postcondiciones	El registro de admisión del paciente es creado.
Flujo normal	1. El paciente llega a la localidad médica y solicita atención. 2. El recepcionista ingresa al sistema. 3. El recepcionista selecciona la opción Admisión – Primera Vez. 4. El recepcionista toma los datos del paciente y los registra en el formulario. 5. El recepcionista selecciona la opción guardar. 6. El sistema validad los datos ingresados. 7. El sistema almacena los registros en la base de datos. 8. El sistema retorna el mensaje de ingreso exitoso.
Flujo alternativo	1. El recepcionista ingresa datos inválidos. 2. El sistema retorna un mensaje de error dependiendo de que validación haya fallado.
Excepciones	Paciente creado en el sistema.

Nota: Detalles del proceso para admisiones por primera vez de un paciente

Elaborado por: (Sánchez & Castro)

2.3.3. Caso de uso: Registro admisiones subsecuentes

A continuación tenemos la representación visual del proceso de registro de admisiones de pacientes subsecuentes (figura 10) y una descripción detallada de este (tabla 6).

Caso de uso: Registro admisiones subsecuentes.

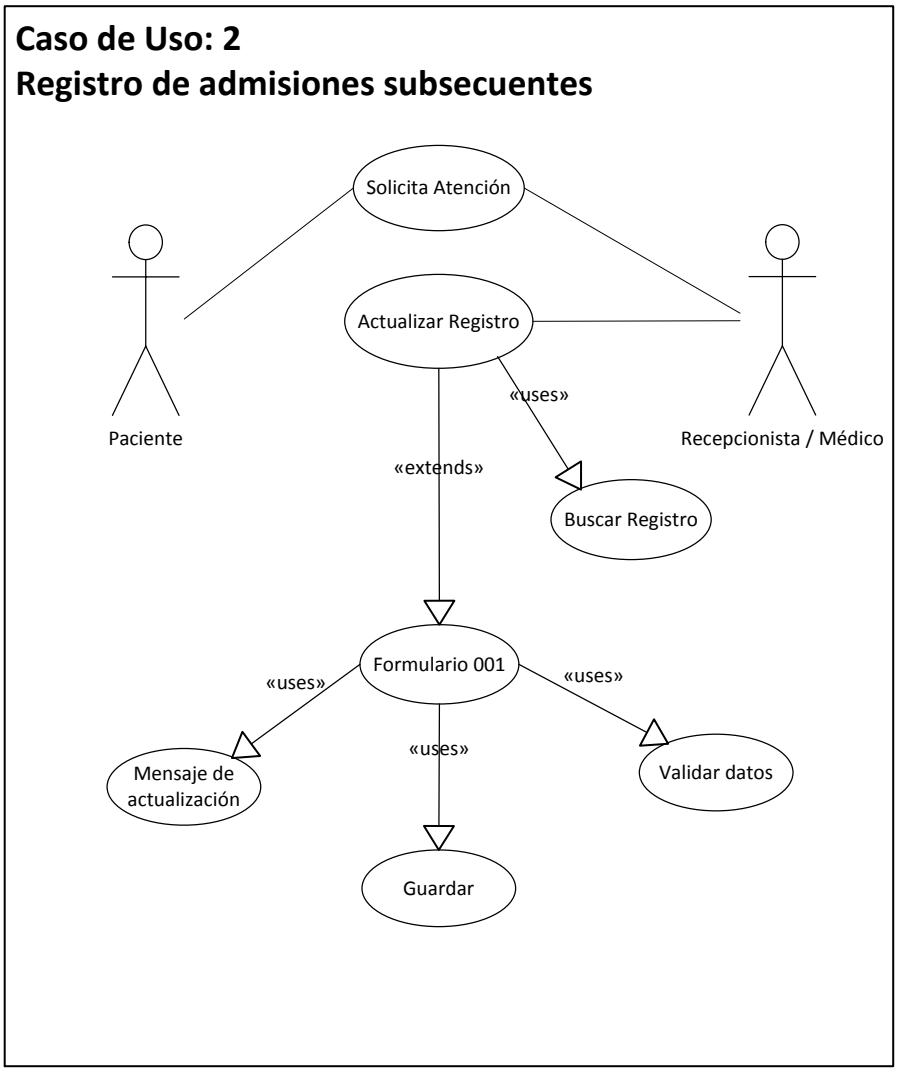


Figura 10. Descripción del proceso para admisiones subsecuentes

Elaborado por: (Sánchez & Castro)

Tabla 6. Caso de uso: Registro admisiones subsecuentes.

Nro. Caso Uso	CU2
Nombre	Registro admisiones subsecuentes.
Descripción	Actualizar el registro de admisión de los pacientes, y llevar un log de los cambios realizados.
Actores	Recepcionista/Médico , Paciente

Precondiciones	Ser usuario recepcionista. Iniciar sesión. El paciente debe tener el registro de al menos una admisión en el sistema.
Postcondiciones	El registro de admisión del paciente es actualizado, y el log de los cambios es creado.
Flujo normal	1. El paciente llega a la localidad médica y solicita atención. 2. El recepcionista ingresa al sistema. 3. El recepcionista selecciona la opción Admisión – Subsecuentes. 4. El recepcionista ingresa la cédula o el número de historia clínica del paciente. 5. El sistema realiza una búsqueda de una previa admisión. 6. El recepcionista toma los datos del paciente y los actualiza en el formulario. 7. El recepcionista selecciona la opción guardar. 8. El sistema validad los datos ingresados. 9. El sistema almacena los registros en la base de datos. 10. El sistema retorna el mensaje de actualización exitoso.
Flujo alternativo	1. El recepcionista ingresa datos inválidos. 2. El sistema retorna un mensaje de error dependiendo de que validación haya fallado.
Excepciones	Admisión del paciente no realizada en el sistema.

Nota: Detalles del proceso para admisiones subsecuentes de un paciente

Elaborado por: (Sánchez & Castro)

2.3.4. Caso de uso: Actualización examen físico regional

A continuación tenemos la representación visual del proceso de actualización examen físico regional (figura 11) y una descripción detallada de este (tabla 7).

Caso de uso: Actualización examen físico regional.

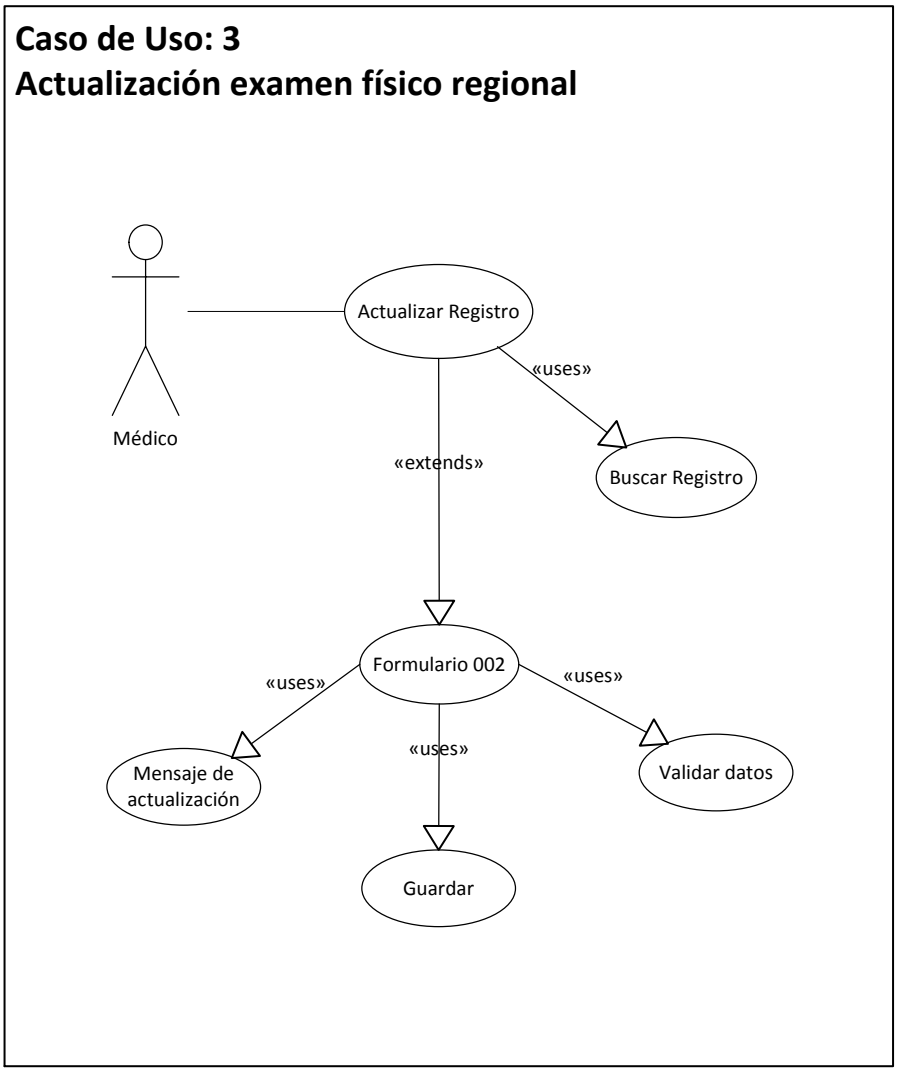


Figura 11. Descripción del proceso para actualizar examen físico regional

Elaborado por: (Sánchez & Castro)

Tabla 7. Caso de uso: Actualización examen físico regional.

Nro. Caso Uso	CU3
Nombre	Actualización examen físico regional.
Descripción	Actualizar el examen físico regional de los pacientes.
Actores	Médico.
Precondiciones	Ser usuario médico. Iniciar sesión. El paciente debe tener el registro de al menos una admisión en el sistema.
Postcondiciones	El examen físico regional del paciente es actualizado.
Flujo normal	1. El médico ingresa al sistema. 2. El médico selecciona la opción Examen Físico – Examen Físico Regional. 3. El médico ingresa la cédula o el número de historia clínica del paciente. 4. El sistema realiza una búsqueda del examen físico regional. 5. El médico actualiza la evidencia patológica del paciente. 6. El médico selecciona la opción guardar. 7. El sistema validad los datos ingresados. 8. El sistema almacena los registros en la base de datos. 9. El sistema retorna el mensaje de actualización exitoso.
Flujo alternativo	1. El médico ingresa datos inválidos. 2. El sistema retorna un mensaje de error dependiendo de que validación haya fallado.
Excepciones	Admisión del paciente no realizada en el sistema.

Nota: Detalles del proceso para actualización de un examen físico regional de un paciente

Elaborado por: (Sánchez & Castro)

2.3.5. Caso de uso: Administración de imágenes médicas

A continuación tenemos la representación visual del proceso de administración de imágenes médicas (figura 12) y una descripción detallada de este (tabla 8).

Caso de uso: Administración de imágenes médicas.

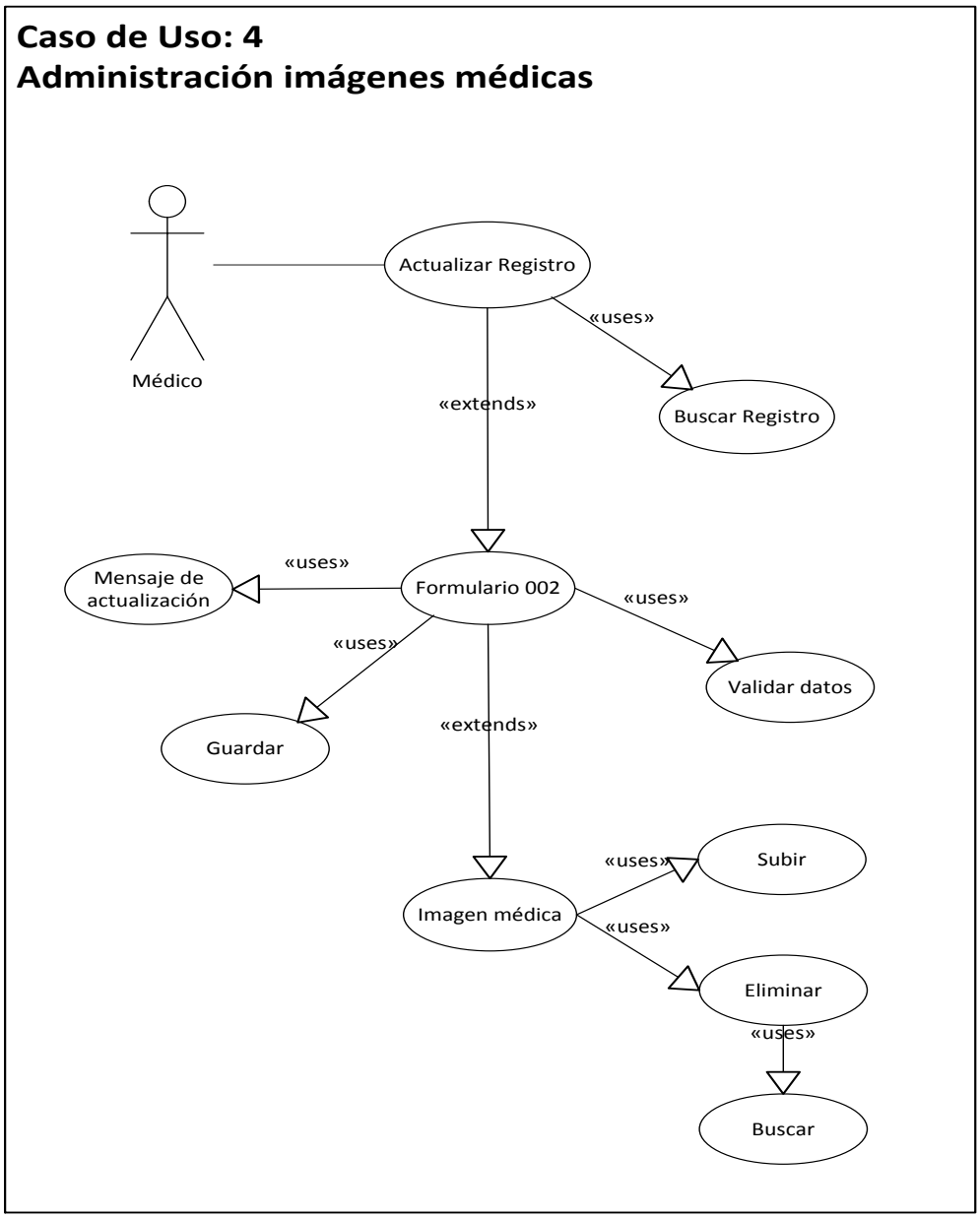


Figura 12. Descripción del proceso para la administración de imágenes médicas

Elaborado por: (Sánchez & Castro)

Tabla 8. Caso de uso: Administración de imágenes médicas.

Nro. Caso Uso	CU4
Nombre	Administración de imágenes médicas.
Descripción	Subir, actualizar o eliminar imágenes médicas en el examen físico regional.
Actores	Médico.
Precondiciones	Ser usuario médico. Iniciar sesión. El paciente debe tener el registro de al menos una admisión en el sistema.
Postcondiciones	La imagen médica actualizada.
Flujo normal	1. El médico ingresa al sistema. 2. El médico selecciona la opción Examen Físico – Examen Físico Regional. 3. El médico ingresa la cédula o el número de historia clínica del paciente. 4. El sistema realiza una búsqueda del examen físico regional. 5. El médico sube la imagen médica. 6. El médico actualiza la imagen médica. 7. El médico elimina la imagen médica. 8. El médico selecciona la opción guardar. 9. El sistema validad los datos ingresados. 10. El sistema almacena los registros en la base de datos. 11. El sistema retorna el mensaje de actualización exitoso.
Flujo alternativo	1. El médico ingresa datos inválidos. 2. El sistema retorna un mensaje de error dependiendo de que validación haya fallado.

Excepciones	Admisión del paciente no realizada en el sistema.

Nota: Detalles del proceso para administración de imágenes médicas de un paciente

Elaborado por: (Sánchez & Castro)

2.4. Diseño de la base de datos relacional

Para este cometido, se escogió por su facilidad de uso, extremada potencia, seguridad y escalabilidad a MySQL. Nuestra base de datos tendrá dos finalidades:

1. Almacenará los datos de los formularios 001 (Admisión) y 002 (Consulta Externa, sección 7 examen físico regional). Para esto se ha diseñado las siguientes tablas:
   - *hcu_form001*. Contendrá un único registro de los datos de la admisión (formulario 001) más reciente del paciente.
   - *hcu_form001_log.* Contendrá el log de los cambios realizados en cada admisión del paciente (formulario 001). Se relacionará con la tabla *hcu_form001* a través de su cédula.
   - *hcu_form002.* Contendrá un único registro de la consulta externa del paciente (formulario 002, sección 7 examen físico regional). Se relacionará con la tabla *hcu_form001* a través de su cédula.
2. Almacenará los datos de autenticación y autorización de usuarios en el aplicativo:
   - *users.* Contendrá usuarios y contraseñas necesarios para la autenticación.
   - *roles.* Contendrá roles permitidos.
   - *users_roles.* Contendrá la relación entre usuarios y los roles que desempeñe.

En la figura 13 tenemos la representación visual del diseño de la base de datos.

Diseño de base de datos.

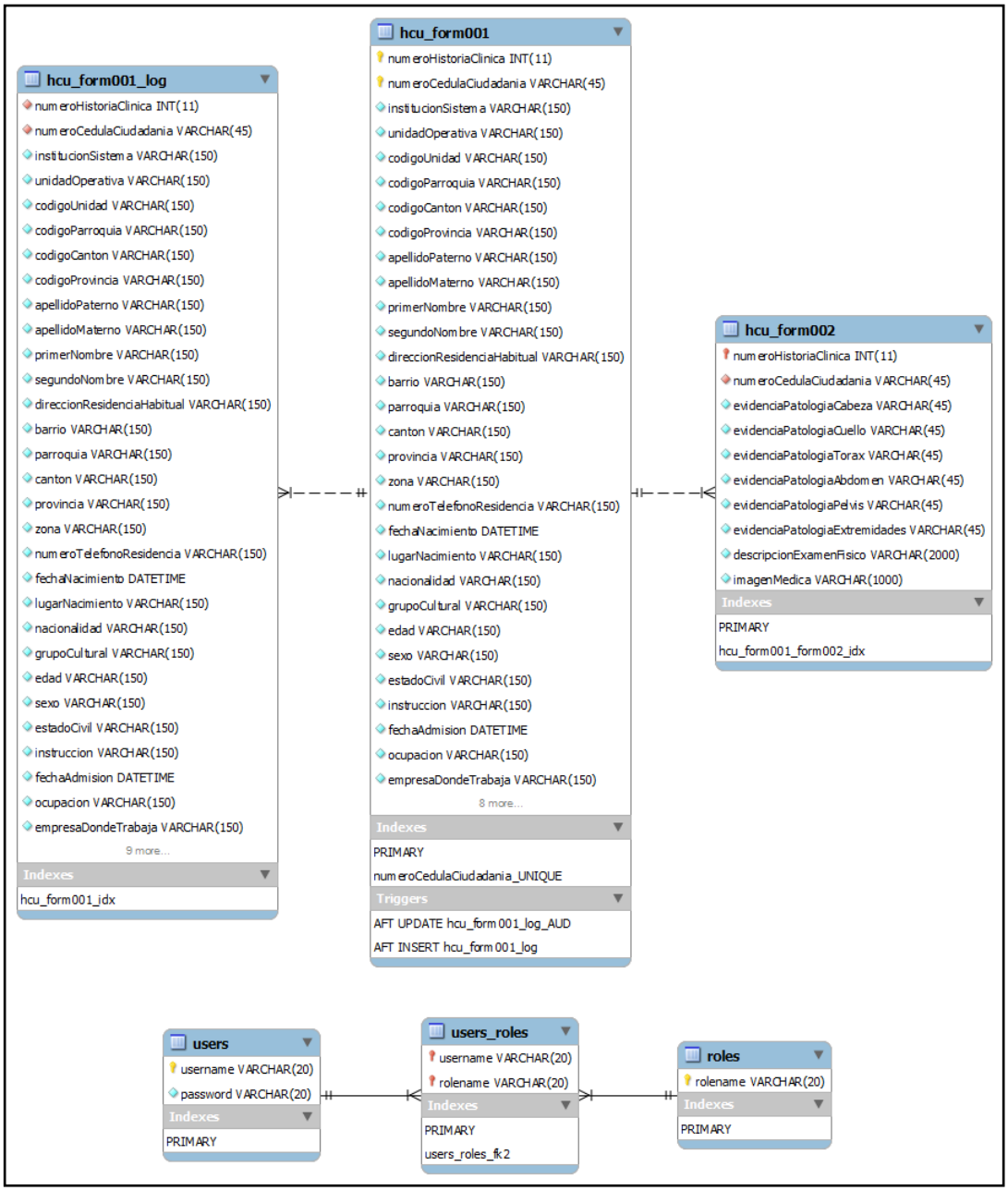


Figura 13. Descripción de la base de datos relacional

Elaborado por: (Sánchez & Castro)

2.5. Diseño del sistema de archivos distribuidos Hadoop

Por su escalabilidad, costo efectivo, flexibilidad, velocidad y su resistencia a fallos se optó por Hadoop.

Como podemos ver en la figura 14 el sistema constará de un nodo maestro (*NameNode*) y un nodo esclavo (*DataNode*). El *NameNode* manejará todas las

peticiones que vienen del sistema multiagente precisamente del "Agente Orden Imagen" (Figura 21, sección F3.3 Agente Orden Imagen) que veremos más adelante en la etapa de construcción.

Independientemente del tipo de petición, sea de creación, recuperación o eliminación, el *NameNode* será el encargado de ejecutar las tareas MapReduce, este algoritmo paralelizará los archivos en este caso las imágenes médicas DICOM y las distribuirá en los *DataNode*.

Es importante mencionar que dentro del sistema de archivos distribuidos de Hadoop, se llevará la siguiente jerarquía: /pacientes/<número documento>/<número de historia clínica>/<imagen 1>;<imagen 2>;…<imagen n>. Esta jerarquía la podemos ver en la figura 14 bajo el HDFS.

Diseño sistema de archivos distribuido.

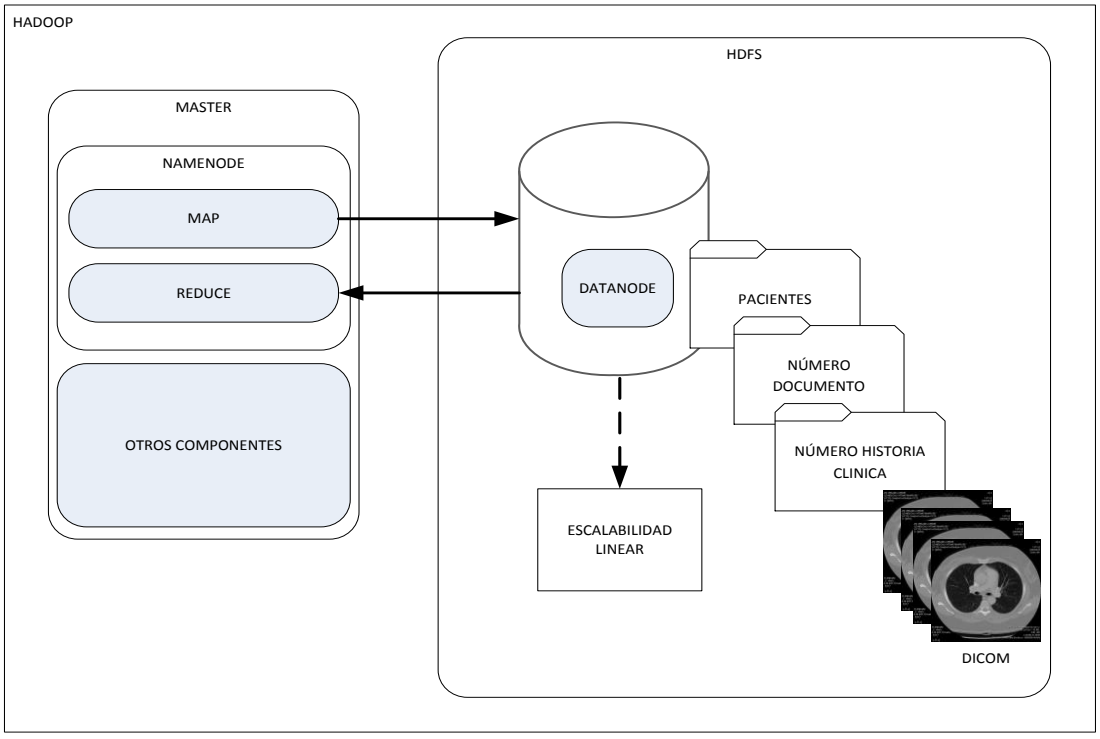


Figura 14. Descripción del sistema de archivos distribuidos

Elaborado por: (Sánchez & Castro)

2.6. Diseño de interfaces

A continuación, se presentará de una manera esquemática las interfaces que se utilizan, se realizaron con la herramienta de modelado de interfaces Balsamiq Mockups (balsamiq, 2008).

2.6.1. Inicio de Sesión

En la figura 15 se presenta la pantalla designada para realizar el proceso de inicio de sesión.

Pantalla inicio de sesión.

Figura 15. Diseño de inicio de sesión

Elaborado por: (Sánchez & Castro)

2.6.2. Pantalla Principal

En la figura 16 se presenta la plantilla genérica que se usa en el sistema, el área de trabajo es la que muchas veces cambiará, pero el esquema básico se mantiene.

Pantalla Principal.

Figura 16. Diseño de inicio de la pantalla principal

Elaborado por: (Sánchez & Castro)

2.6.3. Formulario Admisión primera vez

En la figura 17 se presenta la pantalla designada para el registro de admisiones por primera vez (formulario 001).

Pantalla formulario admisión primera vez.

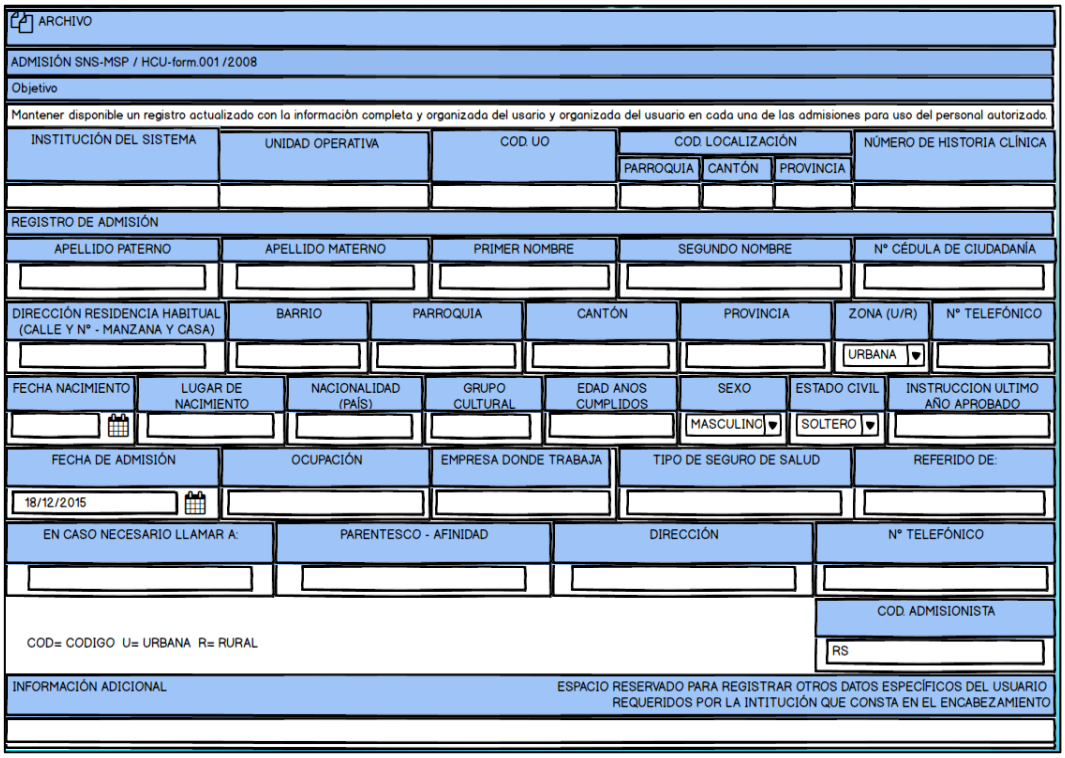


Figura 17. Diseño del formulario de admisión por primera vez

Elaborado por: (Sánchez & Castro)

2.6.4. Formulario Admisión subsecuente

En la figura 18 se presenta la pantalla designada para el registro de admisiones subsecuentes (formulario 001).

Pantalla formulario admisión subsecuente.

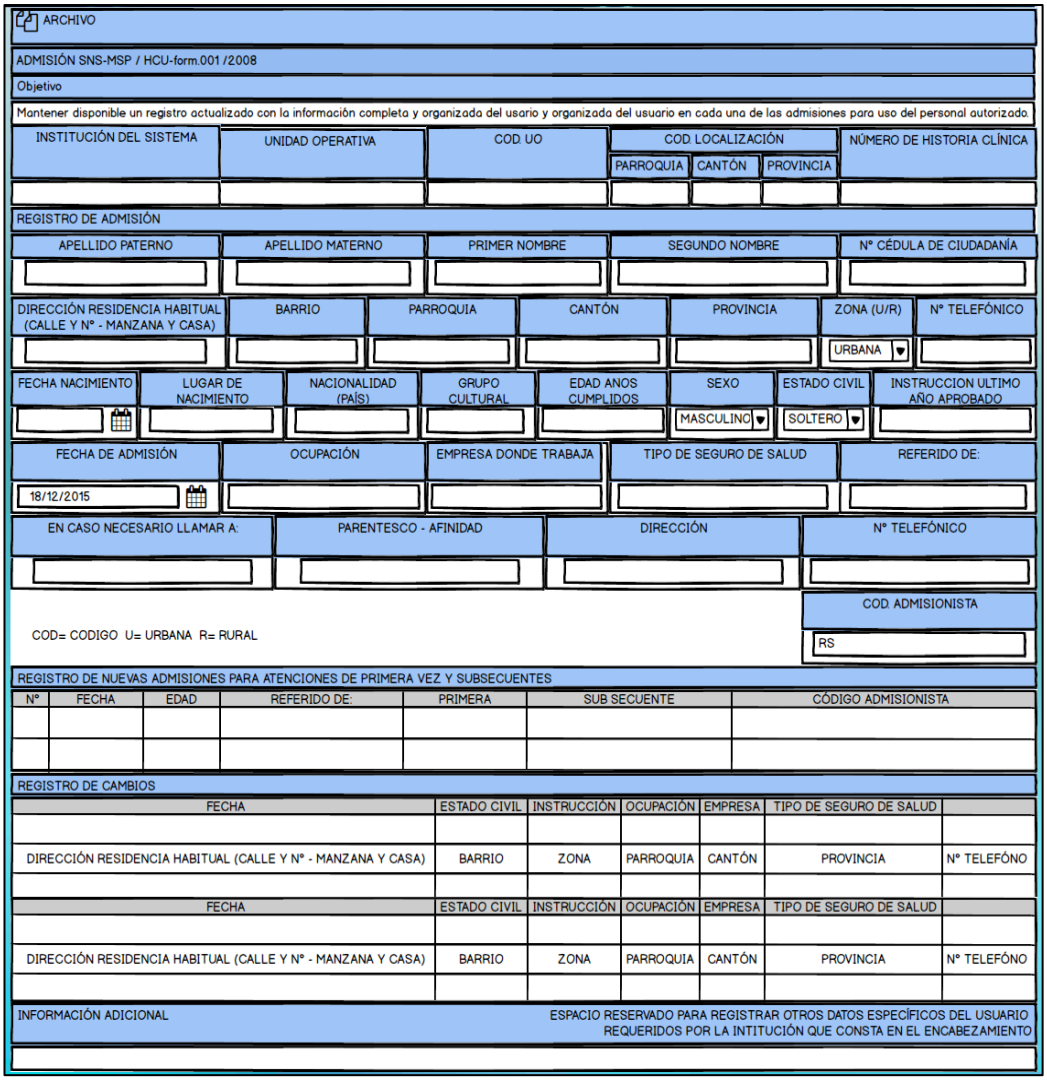


Figura 18. Diseño del formulario de admisión subsecuente

Elaborado por: (Sánchez & Castro)

2.6.5. Búsquedas formularios

En la figura 19 se presenta la pantalla designada para realizar la búsqueda de registros (formulario 001 o formulario 002).

Pantalla búsqueda formularios.

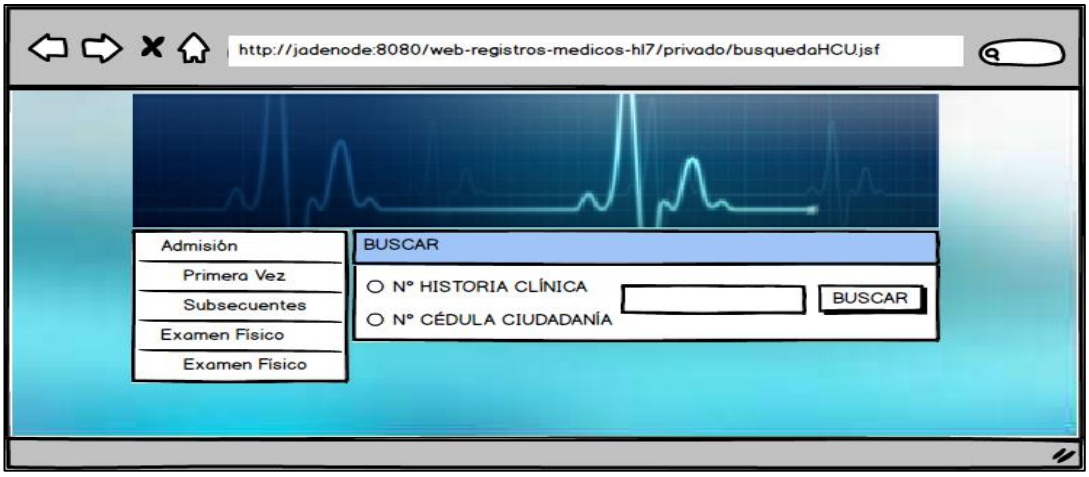


Figura 19. Diseño de búsqueda de formularios

Elaborado por: (Sánchez & Castro)

2.6.6. Formulario examen físico

En la figura 20 se presenta la pantalla designada para la actualización del examen físico (formulario 002, sección 7).

Pantalla formulario examen físico.

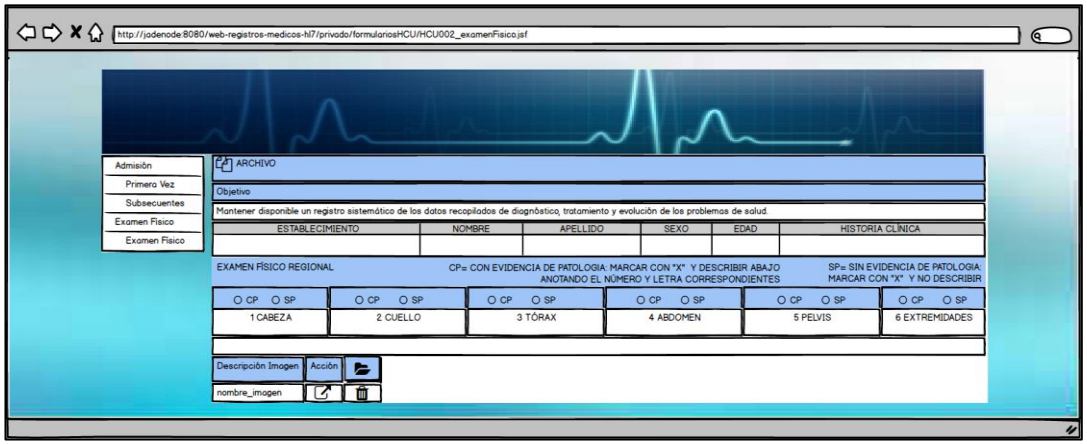


Figura 20. Diseño del formulario para un examen físico

Elaborado por: (Sánchez & Castro)

CAPITULO 3

3. CONSTRUCCIÓN Y PRUEBAS

3.1. Construcción

Arquitectura Sistema multiagente.

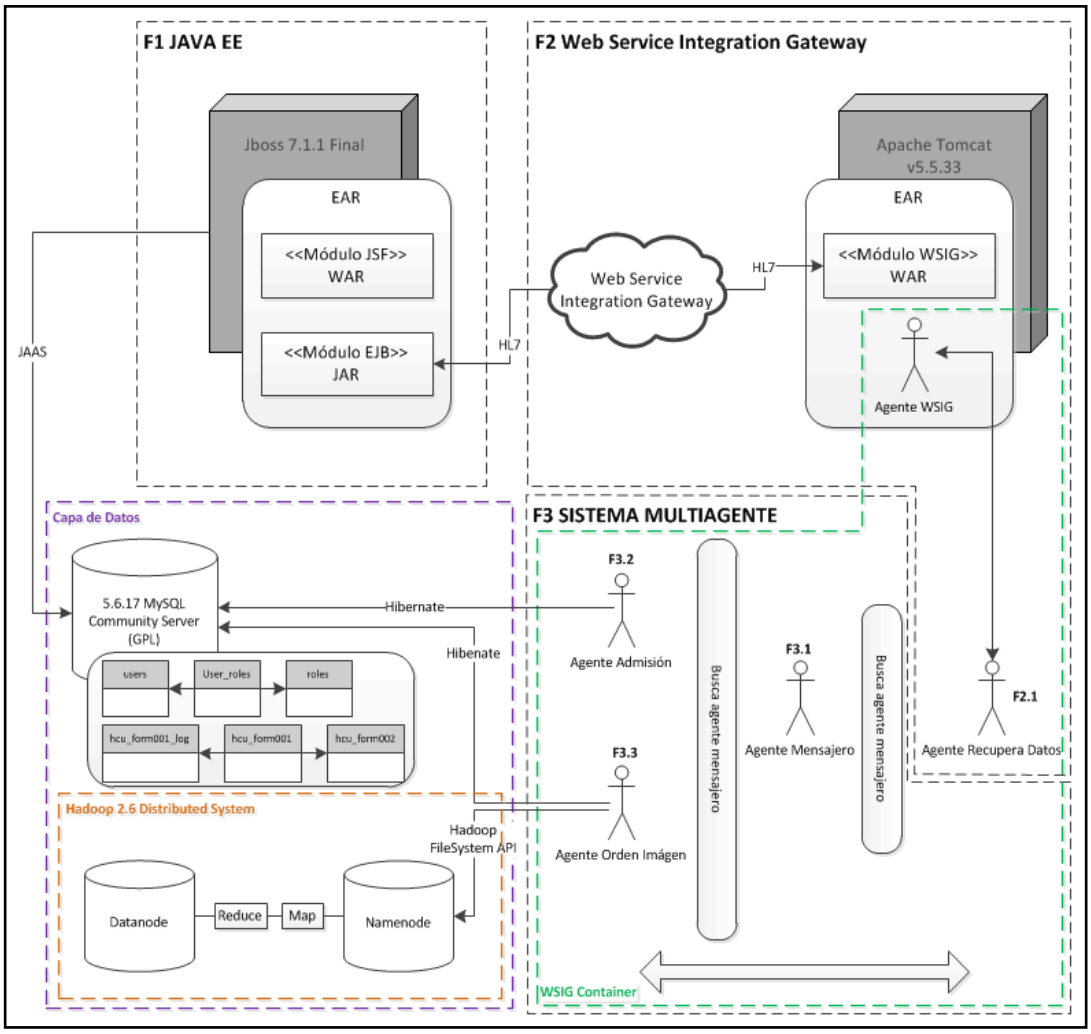


Figura 21. Descripción de la arquitectura del sistema multiagente

Elaborado por: (Sánchez & Castro)

Para la construcción del aplicativo fue necesario dividir el trabajo en tres fases. 1) JAVA EE, 2) Web Service Integration Gateway, 3) Sistema multiagente.

3.1.1. Elaboración Aplicación Empresarial Java (JAVA EE)

Esta fase del sistema representada visualmente en la figura 21 sección F1 JAVA EE, se encuentra la interfaz del usuario final, en este caso el médico o encargado del

ingreso y modificación de los datos del paciente, se encuentra la información y opciones necesarias para realizar esta acción.

El desarrollo se lo ha realizado bajo el framework JavaServer Faces y se lo ha dividido en los módulos o capas a) Capa de presentación, b) Capa de negocio.

3.1.1.1. Capa de presentación

Tomando como modelo los formularios vigentes para el registro de admisiones de los pacientes, procedemos a construir el formulario 001 y la sección número 7 del formulario 002.

3.1.1.2. Ingreso al sistema

Ingreso al sistema.

Figura 22. Pantalla de ingreso al sistema

Elaborado por: (Sánchez & Castro)

Se construye la página `login.xhtml` (figura 22), que es llamada por la URL http://jadenode:8080/web-registros-medicos-hl7/publico/login/login.jsf, el sistema re direcciona automáticamente a esta página cada vez que no se encuentre una sesión abierta o ésta haya caducado.

El formulario consta con los campos de usuario y contraseña, ejecuta la acción `j_security_check` propia de la implementación del Servicio de Autorización y Autenticación Java (JAAS).

3.1.1.3. Formulario 001 admisión

Formulario 001.

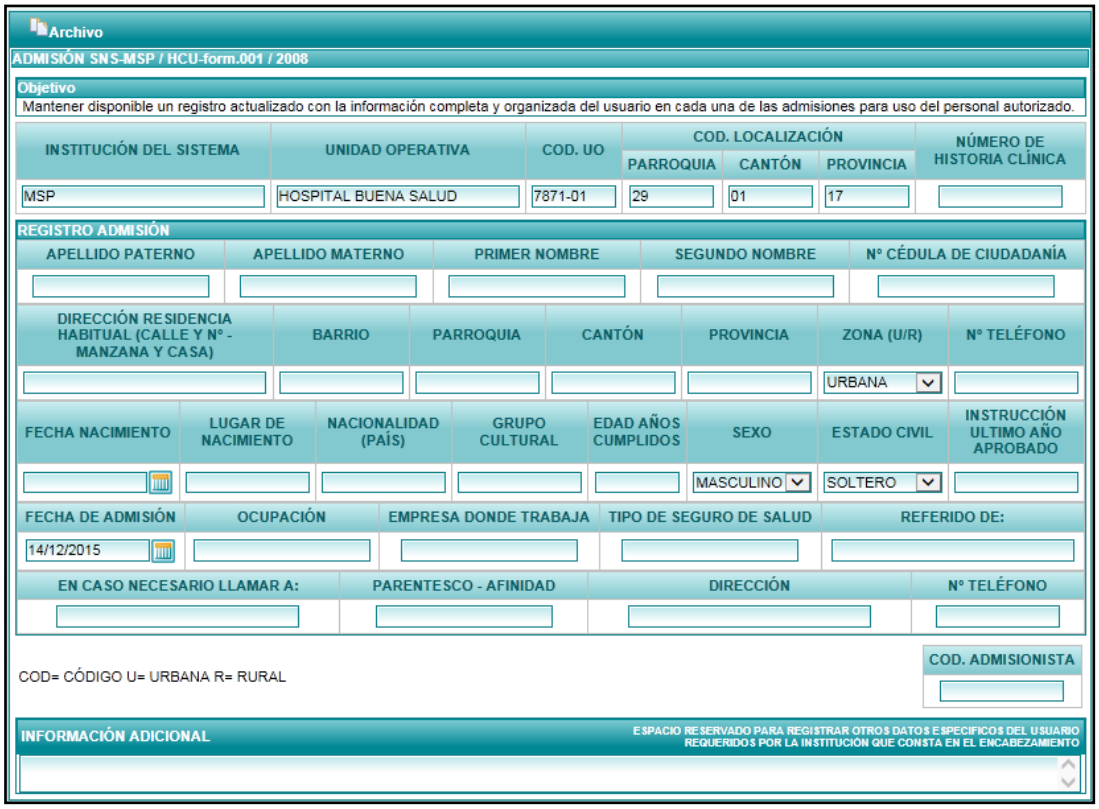


Figura 23. Pantalla de formulario de admisión

Elaborado por: (Sánchez & Castro)

Se construye la página `HCU001_admisionPaciente.xhtml` (figura 23), que es llamada por la URL http://jadenode:8080/web-registros-medicos-hl7/privado/formulariosHCU/_HCU001_admisionPaciente.jsf está dispone de los campos necesarios para la admisión por primera vez o subsecuente de un paciente en el sistema.

El formulario está atado al controlador (ManagedBean) `HCU001_admicionPacienteControlador.java`. Cuyos principales métodos son:

- `cambioEstadoFormulario(String,String,ArrayOfString)` Modifica el estado del formulario 0 Nuevo, 1 Guardado.
- `enviarDatosFormulario()` Envía los datos del formulario encapsulándolos en mensajes HL7 y con este parámetro hace un llamado al método

`recuperaDatosFormulario(String)` del Servicio Web llamado Web Service Integration Gateway (WSIG) que veremos más adelante.

- `inicio()` Método `@PostConstruct` se ejecuta cada vez que se crea una nueva instancia de la clase, se encarga de inicializar las variables y de evaluar si es un nuevo formulario.

3.1.1.4. Formulario 002 consulta externa

Formulario 002.

Figura 24. Pantalla de formulario consulta externa

Elaborado por: (Sánchez & Castro)

Se construye la página `HCU002_examenFisico.xhtml` (figura 24), que es llamada por la URL http://jadenode:8080/web-registros-medicos-hl7/privado/formulariosHCU/HCU002_examenFisico.jsf esta consta con los campos necesarios para llevar un registro de datos encargados del diagnóstico, procedimiento y la evolución de las complicaciones registradas en la salud, en este caso nos centramos en el examen físico regional.

El formulario está atado al controlador (ManagedBean) `HCU002_examenFisicoControlador.java` Cuyos principales métodos son:

- `cambioEstadoFormulario(String,String,ArrayOfString)` Modifica el estado del formulario 0 Nuevo, 1 Guardado.
- `eliminarImagen()` Hace un llamado al método enviarDatosFormulario(String) enviándole un parámetro que identifica la imagen a ser eliminada.

- `enviarDatosFormulario(String)` Envía los datos del formulario encapsulándolos en mensajes HL7 y con este parámetro hace un llamado al método `recuperaDatosFormulario(String)` del Servicio Web WSIG.
- `guardarDatosFormulario()` Hace un llamado al método `enviarDatosFormulario(String)` enviándole un parámetro para que guarde los datos del formulario.
- `guardarImagen()` Hace un llamado al método `enviarDatosFormulario(String)` enviándole un parámetro que identifica la imagen a ser guardada.
- `inicio()` Método `@PostConstruct` se ejecuta cada vez que se crea una nueva instancia de la clase, se encarga de inicializar las variables y de evaluar si es un nuevo formulario.
- `initListaImagenesMedicas()` Modifica la lista de imágenes que se puede visualizar.
- `listenerFileUpload()` Reacciona a los eventos de subida de archivos, transforma las imágenes a cadena de bits.
- `recuperaImagen()` Hace un llamado al método `enviarDatosFormulario(String)` enviándole un parámetro que identifica la imagen a ser recuperada. Luego de recuperar la imagen el sistema hace uso de la librería `pixelmed.jar` (librería que implementa el código para leer y crear datos DICOM) (PixelMed Publishing, 2000-2015) para poder visualizar la imagen médica DICOM como se muestra en la figura 25.

Imagen médica recuperada

Figura 25. Pantalla de recuperación de una imagen médica

Elaborado por: (Sánchez & Castro)

3.1.1.5. Búsqueda de formularios existentes

Búsqueda de formularios existentes.

Figura 26. Pantalla de búsqueda de un formulario

Elaborado por: (Sánchez & Castro)

Se construye la página `busquedaHCU.xhtml` (figura 26), que es llamada por la URL http://jadenode:8080/web-registros-medicos-hl7/privado/busquedaHCU.jsf esta consta con un campo que permitirá la búsqueda tanto de formularios 001 como 002, la búsqueda puede ser por el número de historia clínica o por número de cédula de ciudadanía.

El formulario está atado al controlador (ManagedBean) `BusquedaHCUControlador.java`. Cuyos principales métodos son:

- `buscarFormularioHCU()` Envía los datos de la búsqueda a ser procesado. Hace un llamado al método `recuperaDatosFormulario(String)` del Servicio Web WSIG.
- `init()` Método `@PostConstruct` se ejecuta cada vez que se crea una nueva instancia de la clase, se encarga de inicializar los parámetros a buscar.

3.1.1.6. Capa de negocio

En la capa de negocio se construye la clase `ServicioFormulariosHl7.java` y se genera el cliente del Servicio Web WSIG, este cliente se encarga de consumir al servicio web atreves de la clase antes mencionada.
La clase consta de los siguientes métodos:

- `enviarFormularioHl7(String)` Envía el mensaje HL7 al Servicio Web WSIG.
- `enviarFormularioHl7(String, String)` Envía el mensaje HL7 y la imagen DICOM al Servicio Web WSIG.

3.1.2. Elaboración del Web Services Integration Gateway (WSIG)

Arquitectura WSIG.

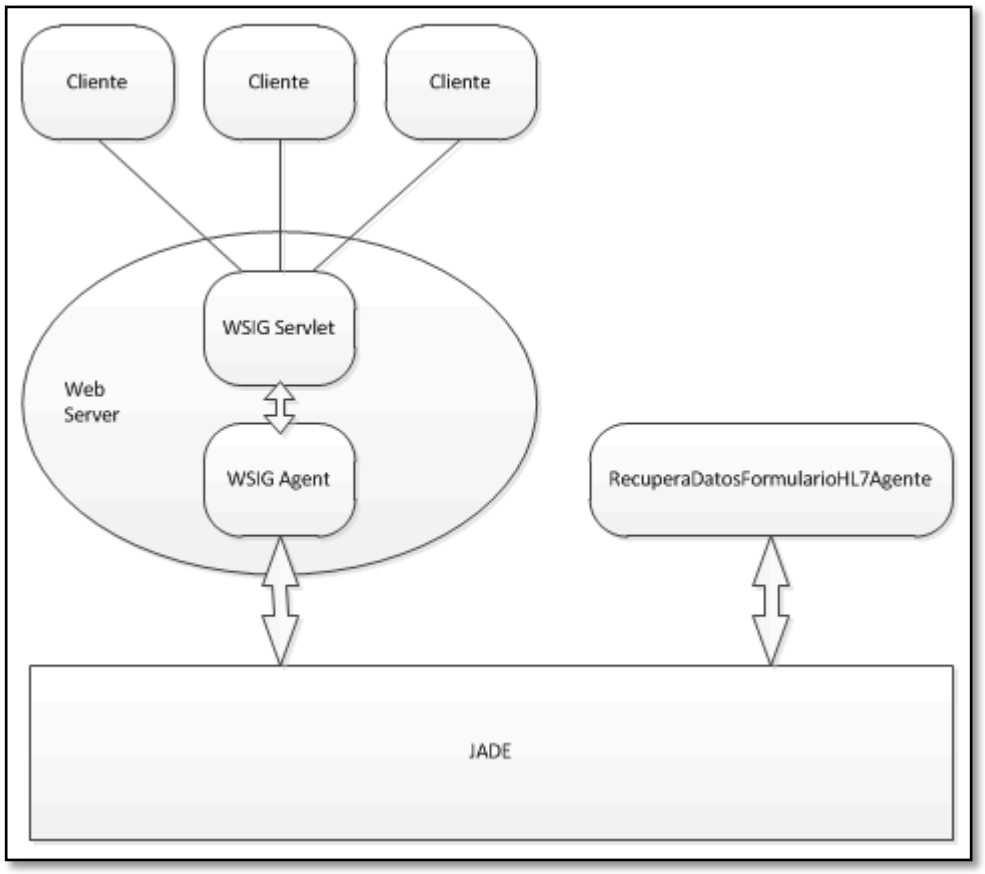


Figura 27. Descripción de la arquitectura WSIG

Elaborado por: (Sánchez & Castro)

WSIG es una aplicación web cuyo objetivo es exponer los servicios proporcionados por los agentes JADE, se encuentra representado visualmente en la figura 21 sección F2 Web Service Integration Gateway.

Su arquitectura (figura 27), consta principalmente de a) WSIG Servlet, b) WSIG Agent y c) Agente Recupera Datos Formulario HL7.

3.1.2.1. WSIG Servlet

La clase `WSIGServlet.java` sirve como front-end frente a la red en la cual se encuentre, es responsable de servir las peticiones HTTP/SOAP entrantes, enviarlo al WSIG Agent y convertir el resultado de la acción en un mensaje SOAP para devolverlo al cliente. Consta de los siguientes métodos:

- `doPost(HttpServletRequest, HttpServletResponse)` Sirve las peticiones HTTP/SOAP.
- `elaborateWSDLRequest(String, HttpServletResponse)` Recupera el servicio WSIG.
- `elaborateWSIGAgentCommand(String, HttpServletResponse)` Inicia el WSIG Agent.
- `executeOperation(AgentAction, WSIGService)` Ejecuta la acción del WSIG Agent.
- `extractSOAPMessage(HttpServletRequest)` Extrae el mensaje SOAP.
- `fillHttpResponse(byte[], String, HttpServletResponse)` Llena la respuesta HTTP.
- `fillSoapHttpResponse(SOAPMessage , HttpServletResponse)` Llena la respuesta SOAP.

3.1.2.2. WSIG Agent

WSIG Agent.

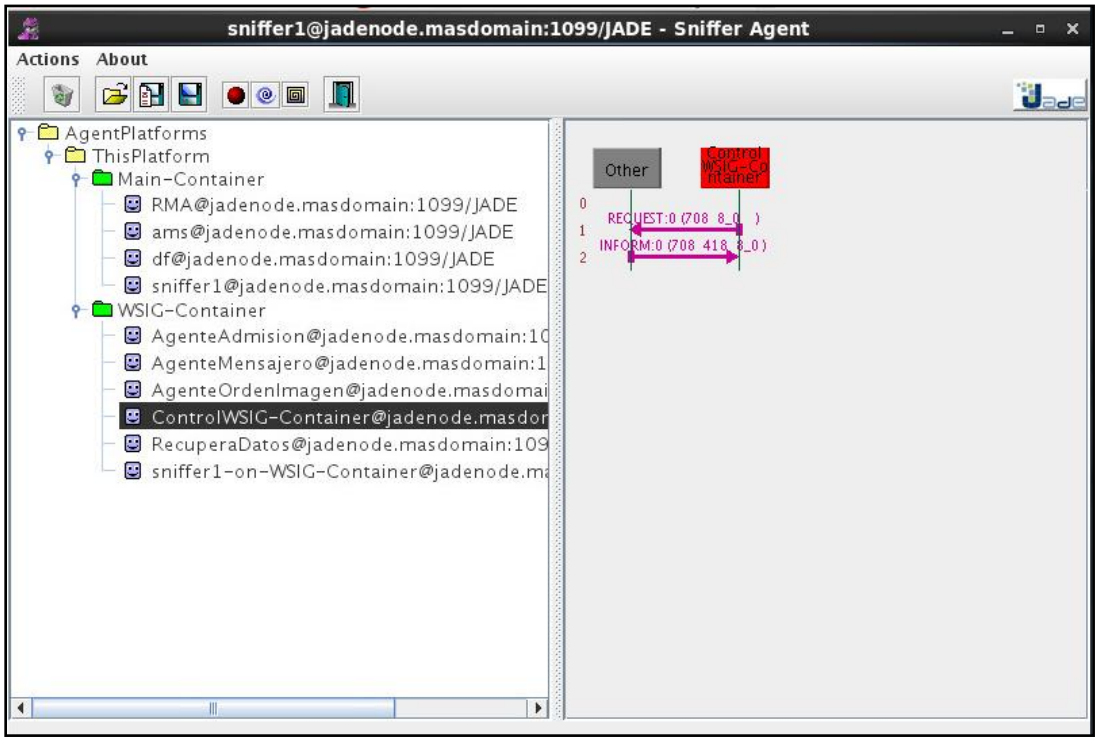


Figura 28. Interacción de un agente

Elaborado por: (Sánchez & Castro)

La clase `WSIGAgent.java` sirve como gateway entre la Web y el mundo de los Agentes, es responsable de reenviar las acciones de los agentes recibidas del WSIG Servlet a los agentes capaces de servir estas acciones y de informar el resultado, en la figura 28 se puede visualizar al agente y sus interacciones en el sistema multiagente. Consta de los siguientes métodos:

- `createWSIGService(AID, ServiceDescription)` Crea el WSDL correspondiente a cada servicio de agente registrado.
- `registerAgent(DFAgentDescription)` Registra el servicio de un agente.
- `deregisterAgent(DFAgentDescription)` Des registra el servicio de un agente.

3.1.2.3. Agente Recupera Datos Formulario HL7

Representado visualmente en la figura 21 sección F2.1 Agente Recupera Datos, este agente es el encargado de extraer la información de los formularios recibida en formato de mensajes HL7 y de reenviar los datos al "Agente Mensajero", a su vez recibir la respuesta de este y reenviarla al WSIG Agent para que la despache. Consta de las siguientes clases:

- `Hl7Ontologia.java`
- `Hl7Vocabulario.java`
- `RecuperaDatosFormularioHL7Agente.java`
- `RecuperarDatosFormularioAccion.java`

3.1.2.3.1. Hl7Ontologia.java

Clase encargada de definir una correcta sintaxis al contenido de los mensajes entre el agente `RecuperaDatosFormularioHL7Agente.java` y el agente `AgenteMensajero.java` que veremos más adelante. La clase consta de los siguientes métodos:

- `getIntance()` Retorna una instancia de la ontología.
- `Hl7Ontologia()` Se encarga de configurar un set de conceptos y símbolos utilizados en la comunicación entre los agentes mencionados.

3.1.2.3.2. Hl7Vocabulario.java

Interfaz encargada de representar vocabulario utilizado en la comunicación entre los agentes mencionados, se implementará en la ontología. Agrupa todos los símbolos usados para nombres de conceptos, predicados, acciones de agentes. El vocabulario consta de los siguientes nombres de conceptos y acciones de agentes:

- *NOMBRE_ONTOLIGIA* Nombre de la ontología.
- *CADENA_DATOS_FORMULARIO* Cadena de datos recibida de los formularios.
- *CADENA_DATOS_IMAGEN_IN* Cadena de datos que contiene la imagen a ser guardada.
- *RESULTADO* Esquema conceptual el cual es entregado como resultado.
- *MENSAJEHL7* Mensaje empaquetado en HL7.
- *LISTALOGMENSAJESHL7* Lista de mensajes empaquetados en HL7.
- *CADENA_DATOS_IMAGEN_OUT* Cadena de datos que contiene la imagen a ser recuperada.

- *CODIGOERROR* Código de error por consulta.
- *RECUPERA_DATOS_FORMULARIO* Acción del agente.

3.1.2.3.3. RecuperaDatosFormularioHL7Agente.java

Agente Recupera Datos.

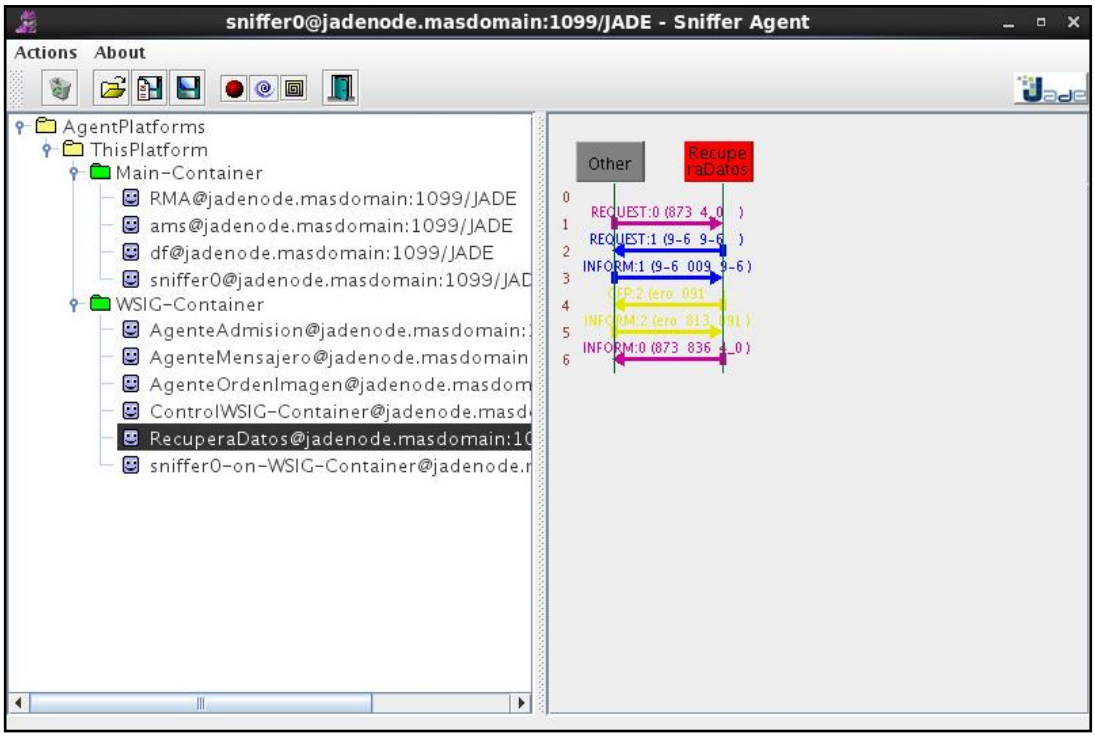


Figura 29. Interacción del agente recupera datos

Elaborado por: (Sánchez & Castro)

Agente encargado receptar información del WSIG Agent, para interactuar con el "Agente Mensajero", la figura 29 se lo representa visualmente dentro del Sistema multiagente. Implementa las siguientes funcionalidades:

- `setup()` Inicializa los comportamientos del agente, además se encarga de registrar los servicios proporcionados en el catálogo "Yellow Pages" donde sus servicios podrán ser encontrados.
- `ObtieneDatosFormularioHL7` Implementa el comportamiento `CyclicBehaviour` de escucha de mensajes HL7 consta de los siguientes métodos:
  - `action()` Identifica si el mensaje contiene datos del formulario.
  - `servirRecuperaDatosFormularioAccion(RecuperarDatosFormularioAccion,Action,ACLMessage)` Si el mensaje tiene el formato de los mensajes HL7, este

método se encarga de darle al agente el comportamiento `EnviarMensajeHl7AgenteMensajero`.

- `EnviarMensajeHl7AgenteMensajero` Se encarga de la comunicación con el "Agente Mensajero" consta de los siguientes métodos:
  - `action()` Implementa la comunicación en dos pasos:
    - Busca los "Agentes Mensajeros" en el sistema, si encuentra al menos uno le envía el mensaje HL7 y queda en espera a su respuesta, si no encuentra mensajero cierra la comunicación.
    - Recibe la respuesta a cargo del "Agente Mensajero" y la reenvía al WSIG Agent.
  - `done()` Determina si el comportamiento del agente ha terminado o no.
  - `notificacion(Action,ACLMessage,int,Object)` Envía notificaciones según los parámetros de envío ingresados.

3.1.2.3.4. RecuperarDatosFormularioAccion.java

Define las acciones que el agente va a entender, estas serán implementadas en la ontología en la clase `Hl7Ontologia.java`.

En la figura 30 como resultado, podemos visualizar el servicio web expuesto del agente.

Servicio Web

```
<?xml version="1.0" encoding="UTF-8"?>
- <wsdl:definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:impl="urn:RecuperaDatosFormularioHL7"
  targetNamespace="urn:RecuperaDatosFormularioHL7" name="RecuperaDatosFormularioHL7">
    - <wsdl:types>
        - <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:impl="urn:RecuperaDatosFormularioHL7"
          targetNamespace="urn:RecuperaDatosFormularioHL7">
              <xsd:annotation/>
            - <xsd:element name="recuperaDatosFormulario">
                - <xsd:complexType>
                    - <xsd:sequence>
                          <xsd:element type="xsd:string" name="cadenaDatosFormulario" minOccurs="0"/>
                          <xsd:element type="xsd:string" name="cadenaDatosImagenIn" minOccurs="0"/>
                      </xsd:sequence>
                  </xsd:complexType>
              </xsd:element>
            - <xsd:element name="recuperaDatosFormularioResponse">
                - <xsd:complexType>
                    - <xsd:sequence>
                          <xsd:element type="impl:resultado" name="recuperaDatosFormularioReturn"/>
                      </xsd:sequence>
                  </xsd:complexType>
              </xsd:element>
            - <xsd:complexType name="resultado">
                - <xsd:sequence>
                      <xsd:element type="xsd:string" name="mensajeHl7"/>
                      <xsd:element type="impl:ArrayOfString" name="listaLogMensajesHl7"/>
                      <xsd:element type="xsd:string" name="cadenaDatosImagenOut" minOccurs="0"/>
                      <xsd:element type="xsd:int" name="codigoError"/>
                  </xsd:sequence>
              </xsd:complexType>
            - <xsd:complexType name="ArrayOfString">
                - <xsd:sequence>
                      <xsd:element type="xsd:string" name="string" minOccurs="0" maxOccurs="unbounded"/>
                  </xsd:sequence>
              </xsd:complexType>
          </xsd:schema>
      </wsdl:types>
    - <wsdl:message name="recuperaDatosFormularioRequest">
          <wsdl:part name="parameters" element="impl:recuperaDatosFormulario"> </wsdl:part>
      </wsdl:message>
    - <wsdl:message name="recuperaDatosFormularioResponse">
          <wsdl:part name="parameters" element="impl:recuperaDatosFormularioResponse"> </wsdl:part>
      </wsdl:message>
    - <wsdl:portType name="RecuperaDatosFormularioHL7Port">
        - <wsdl:operation name="recuperaDatosFormulario">
              <wsdl:input message="impl:recuperaDatosFormularioRequest"> </wsdl:input>
              <wsdl:output message="impl:recuperaDatosFormularioResponse"> </wsdl:output>
          </wsdl:operation>
      </wsdl:portType>
    - <wsdl:binding type="impl:RecuperaDatosFormularioHL7Port" name="RecuperaDatosFormularioHL7Binding">
          <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        - <wsdl:operation name="recuperaDatosFormulario">
              <wsdlsoap:operation soapAction="urn:RecuperaDatosFormularioHL7Action"/>
            - <wsdl:input>
                  <wsdlsoap:body use="literal"/>
              </wsdl:input>
            - <wsdl:output>
                  <wsdlsoap:body use="literal"/>
              </wsdl:output>
          </wsdl:operation>
      </wsdl:binding>
    - <wsdl:service name="RecuperaDatosFormularioHL7Service">
        - <wsdl:port name="RecuperaDatosFormularioHL7Port" binding="impl:RecuperaDatosFormularioHL7Binding">
              <wsdlsoap:address location="http://jadenode.masdomain:8180/web-services-integration-
                gateway/ws/RecuperaDatosFormularioHL7"/>
          </wsdl:port>
      </wsdl:service>
  </wsdl:definitions>
```

Figura 30. Detalle del servicio web del agente expuesto

Elaborado por: (Sánchez & Castro)

3.1.3. Sistema multiagente

Diagrama de secuencia sistema multiagente.

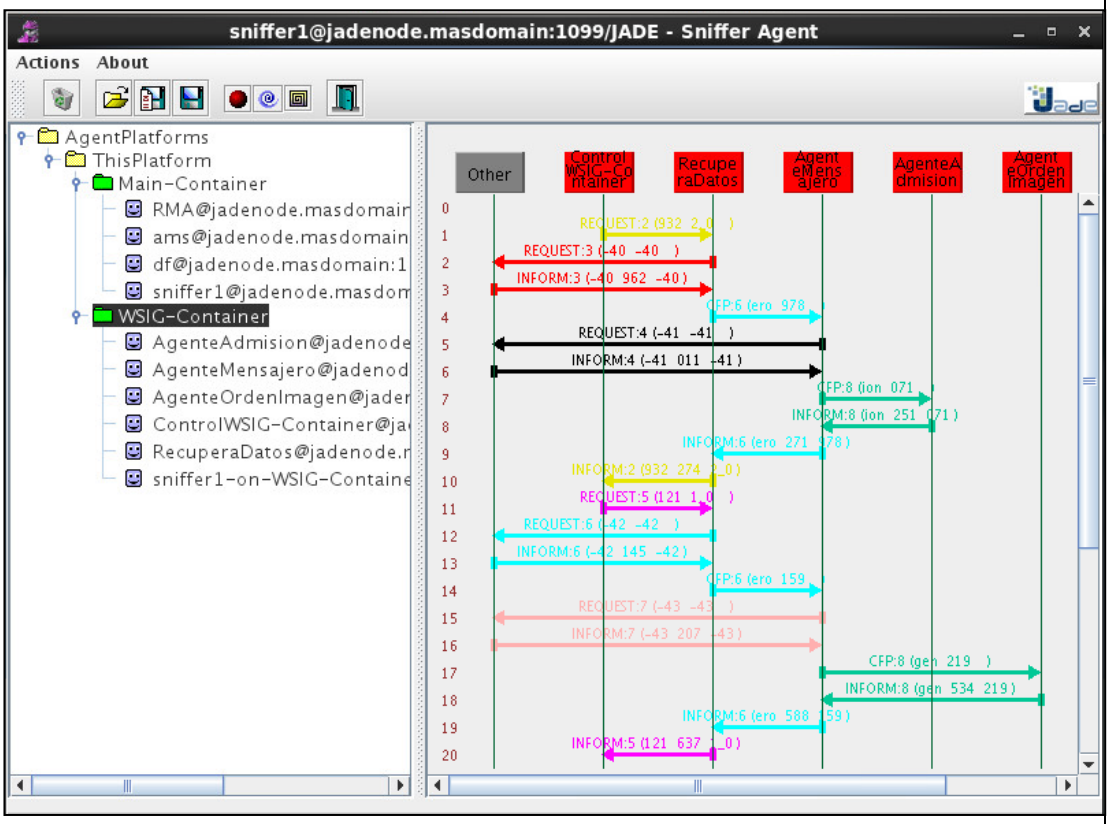


Figura 31. Interacción del sistema multiagente

Elaborado por: (Sánchez & Castro)

Representado como diagrama de secuencia en la figura 31 y como componente del sistema por la figura 21 sección F3 SISTEMA MULTIAGENTE, se encarga de la lógica del negocio, determina como los datos van a ser creados, mostrados, guardados, cambiados y distribuidos. Es sistema cuenta con tres principales componentes: a) Agente Mensajero, b) Agente Admisión y c) Agente Orden Imagen.

3.1.3.1. Agente Mensajero

Encargado de recibir mensajes HL7 analizarlos y reenviarlos al "Agente Admisión" o al "Agente Orden Imagen", según el contenido de estos. Su representación dentro del sistema se encuentra en la figura 21 sección F3.1 Agente Mensajero. Implementa las siguientes funcionalidades:

- `setup()` Inicializa los comportamientos del agente, además se encarga de registrar los servicios proporcionados en el catálogo "Yellow Pages" donde sus servicios podrán ser encontrados.
- ServicioMensajeria Implementa el comportamiento `CyclicBehaviour` de escucha de mensajes HL7 consta de los siguientes métodos:
  - `action()` Recibe el mensaje HL7, lo analiza y determina qué tipo de mensaje es, si es ADT_A01 el agente implementa el comportamiento EnviarMensajeHl7AgenteAdmision, si es OMI_O23 el agente implementa el comportamiento EnviarMensajeHl7AgenteOrdenImagen.
- EnviarMensajeHl7AgenteAdmision Se encarga de la comunicación con el "Agente Admisión" consta de los siguientes métodos:
  - `action()` Implementa la comunicación en dos pasos:
    - Busca los "Agentes Admisión" en el sistema, si encuentra al menos uno le envía el mensaje HL7 y queda en espera a su respuesta, si no encuentra mensajero cierra la comunicación.
    - Recibe la respuesta a cargo del "Agente Admisión" y la reenvía al "Agente Recupera Datos".
  - `done()` Determina si el comportamiento del agente ha terminado o no.
  - `notificaResultado(ACLMessage,int,Object)` Envía notificaciones según los parámetros de envío ingresados.
- EnviarMensajeHl7AgenteOrdenImagen Se encarga de la comunicación con el "Agente Orden Imagen" consta de los siguientes métodos:
  - `action()` Implementa la comunicación en dos pasos:
    - Busca los "Agentes Orden Imagen" en el sistema, si encuentra al menos uno le envía el mensaje HL7 y queda en espera a su respuesta, si no encuentra mensajero cierra la comunicación.
    - Recibe la respuesta a cargo del "Agente Orden Imagen" y la reenvía al "Agente Recupera Datos".
  - `done()` Determina si el comportamiento del agente ha terminado o no.
  - `notificaResultado(ACLMessage,int,Object)` Envía notificaciones según los parámetros de envío ingresados.

3.1.3.2. Agente Admisión

Se encarga de recibir el mensaje ADT_A01 evaluar su contenido para tomar decisiones lógicas e interactuar con la capa de datos, el resultado de este proceso lógico es notificado al "Agente Mensajero", su representación dentro del sistema se encuentra en la figura 21 sección F3.2 Agente Admisión. Consta de las siguientes funcionalidades:

- `setup()` Inicializa los comportamientos iniciales del agente, además se encarga de registrar los servicios proporcionados en el catálogo "Yellow Pages" donde sus servicios podrán ser encontrados.
- ServicioAdmision Implementa el comportamiento `CyclicBehaviour` de escucha de mensajes HL7 consta de los siguientes métodos:
  - `action()` Recibe el mensaje ADT_A01, extrae su contenido y lo mapea en una clase pojo, obtiene el tipo de evento a ser realizado y de acuerdo a este se realizan las siguientes transacciones persistentes a la base de datos MYSQL:
    - Si es una nueva admisión, se crea un nuevo registro en la tabla `hcu_form001`.
    - Si es una admisión subsecuente, se actualiza el registro en la tabla `hcu_form001`.
    - Si es una modificación en la consulta externa, se actualiza el registro en la tabla `hcu_form002`.
    - Si es una búsqueda, primero se determina el formulario a buscar y dependiendo de este, se realiza en la tabla `hcu_form001` o en `hcu_form002`.
    - Si es el formulario es de admisión, se realiza una búsqueda del registro de cambios en la tabla `hcu_form001_log`.
  - Se notifica al "Agente Mensajero" el resultado de las transacciones realizadas.
  - `done()` Determina si el comportamiento del agente ha terminado o no.
  - `notificarResultado(ACLMessage,int,String,int,jade.util.leap.List)` Envía notificaciones según los parámetros de envío ingresados.

Es importante mencionar que todas las transacciones son gestionadas por el framework de persistencia Hibernate (Red Hat projects, 2016). El diseño de las tablas, se encuentra más detallado en la sección 3.4. (Diseño de la base de datos relacional).

3.1.3.3. Agente Orden Imagen

Se encarga de recibir el mensaje OMI_O23 evaluar su contenido para tomar decisiones lógicas e interactuar con la capa de datos, el resultado de este proceso lógico es notificado al "Agente Mensajero", su representación dentro del sistema se encuentra en la figura 21 sección F3.1 Agente Mensajero. Consta de las siguientes funcionalidades:

- `setup()` Inicializa los comportamientos del agente, además se encarga de registrar los servicios proporcionados en el catálogo "Yellow Pages" donde sus servicios podrán ser encontrados.
- `ServicioOrdenImagen` Implementa el comportamiento `CyclicBehaviour` de escucha de mensajes HL7 consta de los siguientes métodos:
- `action()` Recibe el mensaje OMI_023, extrae su contenido y lo mapea en una clase pojo, obtiene el tipo de evento a ser realizado y de acuerdo a este se realiza los siguientes procesos de grabado en el Sistema Distribuido de Archivos de Hadoop a través del API de Hadoop para Java:
  - `fs.create(Path)` Crea archivos en la ruta determinada por el parámetro ingresado.
  - `fs.open(Path)` Recupera archivos en la ruta determinada por el parámetro ingresado.
  - `fs.exists(Path)` Determina si el archivo en la ruta determinada por el parámetro ingresado existe.
  - `fs.delete(Path)` Crea archivos en la ruta determinada por el parámetro ingresado.
- Se notifica al "Agente Mensajero" el resultado de las transacciones realizadas.
- `done()` Determina si el comportamiento del agente ha terminado o no.
- `notificarResultado(ACLMessage,int,String,int,String)` Envía notificaciones según los parámetros de envío ingresados.

El proceso que se lleva a cabo detrás de estas peticiones a Hadoop, se encuentra mejor detallado en la sección 3.5 (Diseño del sistema de archivos distribuidos Hadoop).

3.2. Pruebas

Con el fin de medir la factibilidad de implementación en condiciones conocidas y aportar valor en la calidad del desarrollo de software. Se ha contemplado un plan de pruebas carga y comportamiento funcional utilizando la herramienta JMeter (Apache JMeter Team, 1999), complementado con un monitoreo de la eficiencia de Sistema multiagente en tiempo real con Ganglia (Ganglia Project, 2011).

JMeter fue diseñado originalmente para probar aplicaciones web, permite grabar las iteraciones realizadas en el aplicativo y luego reproducirlas simulando cargas pesadas sobre los servidores.

El resultado de las pruebas expone el correcto funcionamiento de las funcionalidades principales del sistema multiagente.

3.2.1. Pruebas módulo autenticación de usuarios

La prueba está realizada sobre el proceso de autorización y autenticación, entre el servidor Jboss y la base de datos MySQL. Se realizaron las pruebas con los datos de la tabla 9.

Tabla 9. Parámetros prueba autenticación usuarios.

Iteración #	**Usuarios concurrentes**	**Repeticiones**	**Transacciones**
1	50	100	5000
2	100	30	3000
3	200	40	8000

Nota: Detalle de los datos usados para realización de autenticación de usuarios

Elaborado por: (Sánchez & Castro)

3.2.1.1. Iteración 1

Como se muestra en la figura 32, para la primera prueba se configuraron 50 usuarios concurrentes, cada 1 segundo, realizando 100 repeticiones.

Configuración de usuarios concurrentes.

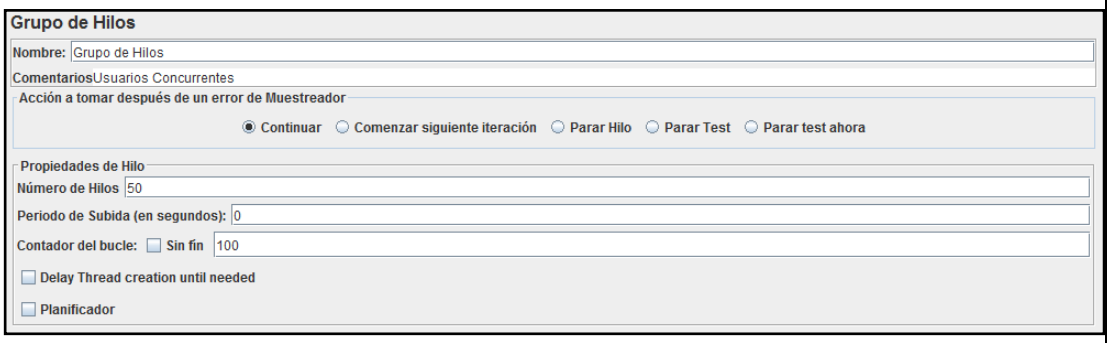


Figura 32. Configuración de los datos de usuarios para realizar la prueba

Elaborado por: (Sánchez & Castro)

Los valores totales se obtenidos se muestran en la figura 33.

Informe agregado.

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec
1 /web-registr...	5000	155	99	353	426	627	9	3519	0,00%	30,9/sec	118,5
3 /web-registr...	5000	37	17	93	137	249	4	752	0,00%	31,0/sec	89,3
4 /web-registr...	5000	365	313	714	860	1395	16	3768	0,00%	31,0/sec	277,2
6 /web-registr...	5000	36	17	91	129	245	4	523	0,00%	31,0/sec	89,4
7 /web-registr...	5000	38	20	90	136	252	4	795	0,00%	31,1/sec	161,1
8 /web-registr...	5000	37	19	89	130	267	4	505	0,00%	31,1/sec	142,2
9 /web-registr...	5000	28	13	68	111	225	3	687	0,00%	31,1/sec	10,3
10 /web-regis...	5000	28	13	72	108	217	3	672	0,00%	31,1/sec	12,7
Total	40000	91	24	271	438	771	3	3768	0,00%	247,4/sec	898,0

Figura 33. Resultado de la prueba realizada para la autenticación de usuarios

Elaborado por: (Sánchez & Castro)

Como se puede observar, el tiempo promedio para realizar la autenticación de usuario es de 91 milisegundos, realizándose un total de 40000 peticiones al servidor.

El tiempo total utilizado para los 50 usuarios concurrentes se puede calcular con la siguiente fórmula:

Tiempo Total = #Muestras * Media = 40000 * 91 = 3640000 milisegundos

El tiempo promedio total requerido por cada usuario concurrente, se puede calcular de la siguiente manera:

((Tiempo Total/1000/60))/usuarios recurrentes = (3640000/1000/60)/50 = 1,213 minutos

En la figura 34 podemos observar el tiempo de respuesta dependiendo de los procesos realizados, podemos destacar en la gráfica que el proceso que consume más tiempo j_security_check.

Gráfico tiempo de respuesta.

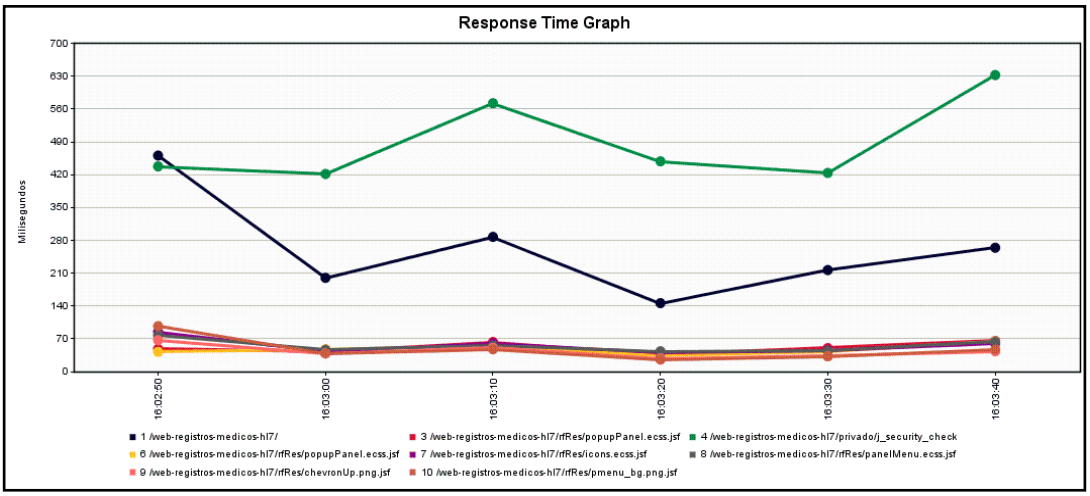


Figura 34. Representación gráfica de los tiempos de respuesta a procesos realizados

Elaborado por: (Sánchez & Castro)

Se valoraron los resultados a través de un intervalo de confianza para una distribución normal del 95%. La fórmula es la siguiente:

$[TP – Z\ 0.95 * D/\sqrt{n};\ TP + Z\ 0.95 * D/\sqrt{n}]$

Dónde:

- Tiempo Promedio (TP) de respuesta es: 91
- Desviación (D) es: 170
- Tamaño de la muestra (n) es: 40000
- Z 0.95 es: 1,96

El intervalo resultante es:

$[91 – 1,96 * 170/\sqrt{40000};\ 91 + 1,96 * 170/\sqrt{40000}]$

$[91 – 1,96 * 0,85;\ 91 + 1,96 * 0,85] = [89,33;\ 92,66]$ en milisegundos.

La figura 35 muestra los resultados conseguidos al ejecutar las peticiones HTTP definidas, brindando en un gráfico de líneas los valores de tiempo en milisegundos, desviación típica, rendimiento, media y mediana.

Gráfico de Resultados.

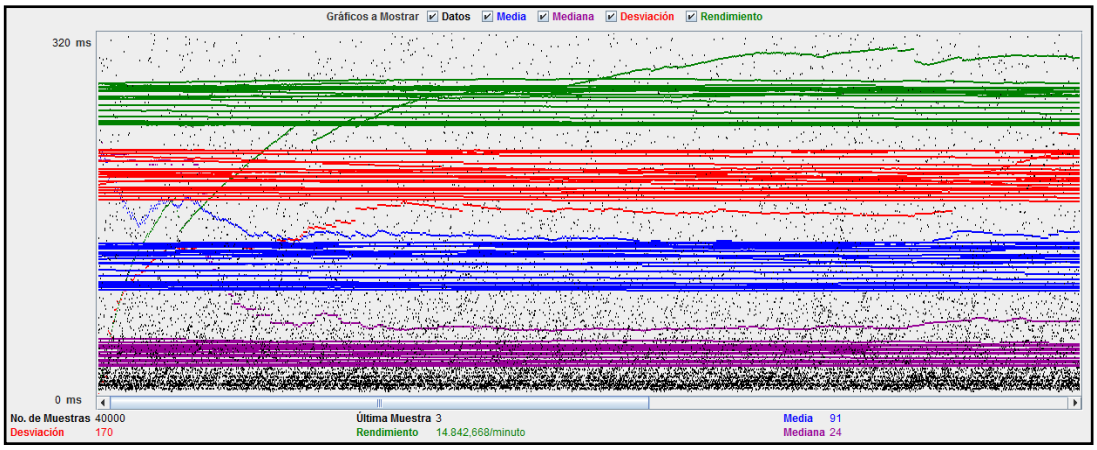


Figura 35. Resultados al ejecutar peticiones HTTP

Elaborado por: (Sánchez & Castro)

Por lo tanto, en la tabla 10 podemos resumir los resultados de la primera iteración.

Tabla 10. Resultados.

Usuarios concurrentes	50
Repeticiones	100
Transacciones	40000
Tiempo Total	3640000 milisegundos
Tiempo de respuesta promedio	89 – 93 milisegundos
Desviación estándar	170

Nota: Resultados totales de la interacción 1

Elaborado por: (Sánchez & Castro)

3.2.1.2. Iteración 2

Como se muestra en la figura 36, para la segunda prueba se configuraron 100 usuarios concurrentes, cada 1 segundo, realizando 30 repeticiones.

Configuración de usuarios concurrentes.

Figura 36. Configuración de los datos de usuarios para realizar la prueba

Elaborado por: (Sánchez & Castro)

Los valores totales se obtenidos se muestran en la figura 37.

Informe agregado.

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec
1 /web-registros-m...	3000	569	535	937	1103	1444	11	2015	0,00%	38,7/sec	148,3
3 /web-registros-m...	3000	113	67	276	386	617	4	1402	0,00%	38,7/sec	111,5
4 /web-registros-m...	3000	1051	982	1632	1935	2687	62	4074	0,00%	38,7/sec	346,1
6 /web-registros-m...	3000	119	76	296	387	646	4	1215	0,00%	40,1/sec	115,3
7 /web-registros-m...	3000	114	70	274	380	628	4	1003	0,00%	40,1/sec	208,1
8 /web-registros-m...	3000	114	73	271	368	591	3	1514	0,00%	40,2/sec	183,8
9 /web-registros-m...	3000	93	56	229	317	546	3	1738	0,00%	40,3/sec	13,4
10 /web-registros-...	3000	98	60	232	336	611	3	1180	0,00%	40,3/sec	16,5
Total	24000	284	108	851	1130	1734	3	4074	0,00%	308,8/sec	1121,1

Figura 37. Resultado de la prueba realizada para la autenticación de usuarios

Elaborado por: (Sánchez & Castro)

Como se puede observar, el tiempo promedio para realizar la autenticación de usuario es de 284 milisegundos, realizándose un total de 24000 peticiones al servidor.

El tiempo total utilizado para los 100 usuarios concurrentes se puede calcular con la siguiente fórmula:

Tiempo Total = #Muestras * Media = 24000 * 284 = 6816000 milisegundos

El tiempo promedio total requerido por cada usuario concurrente, se puede calcular de la siguiente manera:

((Tiempo Total/1000)/60)/usuarios recurrentes = ((6816000/1000)/60)/100 = 1,136 minutos

En la figura 38 podemos observar el tiempo de respuesta dependiendo de los procesos realizados, podemos destacar en la gráfica que el proceso que consume más tiempo j_security_check.

Gráfico tiempo de respuesta.

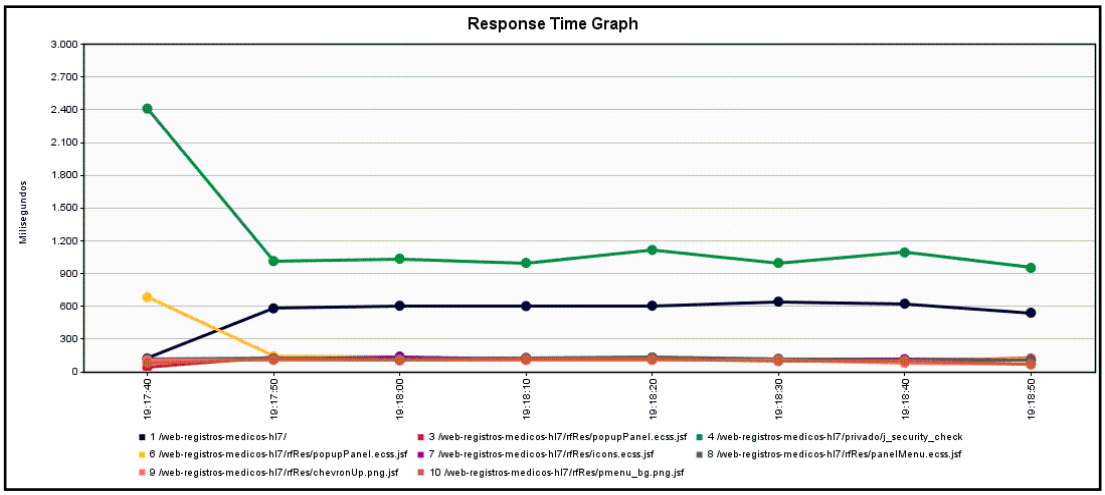


Figura 38. Representación gráfica de los tiempos de respuesta a procesos realizados

Elaborado por: (Sánchez & Castro)

Se valoraron los resultados a través de un intervalo de confianza para una distribución normal del 95%. La fórmula es la siguiente:

$[TP – Z\ 0.95 * D/\sqrt{n};\ TP + Z\ 0.95 * D/\sqrt{n}]$

Dónde:

- Tiempo Promedio (TP) de respuesta es: 284
- Desviación (D) es: 397,95
- Tamaño de la muestra (n) es: 24000
- Z 0.95 es: 1.96

El intervalo resultante es:

$[284 – 1{,}96 * 397.95/\sqrt{24000};\ 284 + 1{,}96 * 397.95/\sqrt{24000}]$

$[284 – 1{,}96 * 2{,}5626;\ 284 + 1{,}96 * 2{,}5626] = [278{,}97;\ 289{,}02]$ en milisegundos.

La figura 39 muestra los resultados conseguidos al ejecutar las peticiones HTTP definidas, brindando en un gráfico de líneas los valores de tiempo en milisegundos, desviación típica, rendimiento, media y mediana.

Gráfico de Resultados.

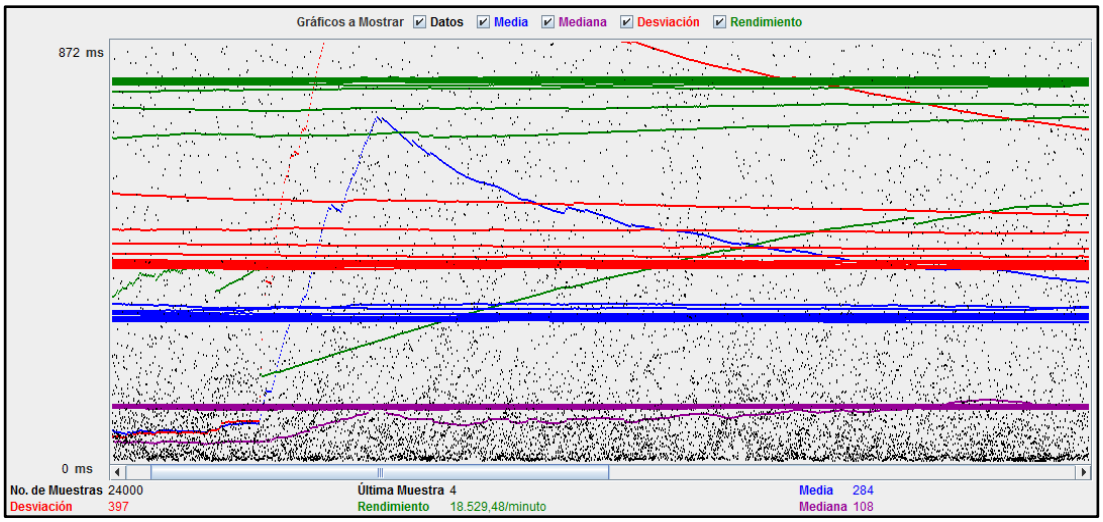


Figura 39. Resultados al ejecutar peticiones HTTP

Elaborado por: (Sánchez & Castro)

Por lo tanto, en la tabla 11 podemos resumir los resultados de la segunda iteración.

Tabla 11. Resultados.

Usuarios concurrentes	100
Repeticiones	30
Transacciones	24000
Tiempo Total	6816000 milisegundos
Tiempo de respuesta promedio	279 – 289 milisegundos
Desviación estándar	397

Nota: Resultados totales de la interacción 2

Elaborado por: (Sánchez & Castro)

3.2.1.3. Iteración 3

Como se muestra en la figura 40, para la tercera prueba se configuraron 200 usuarios concurrentes, cada 1 segundo, realizando 40 repeticiones.

Configuración de usuarios concurrentes.

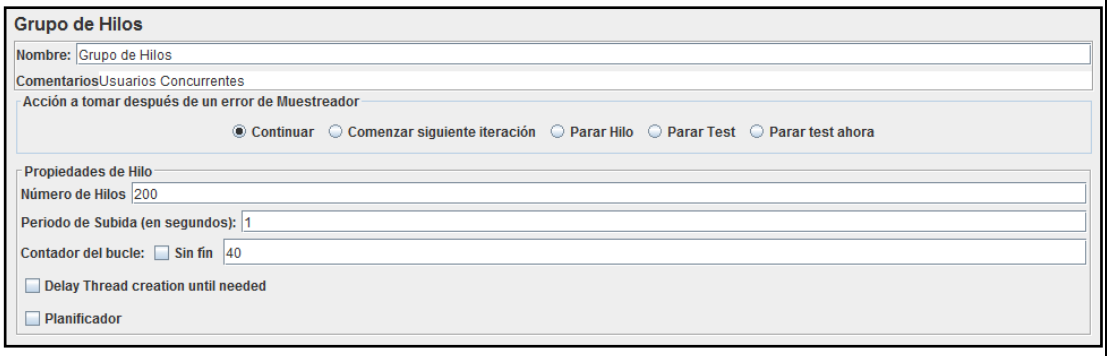


Figura 40. Configuración de los datos de usuarios para realizar la prueba

Elaborado por: (Sánchez & Castro)

Los valores totales se obtenidos se muestran en la figura 41.

Informe agregado.

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec
1 /web-regis...	8000	1388	1498	2507	2847	3587	10	5923	0,00%	20,5/sec	78,4
3 /web-regis...	8000	332	220	821	1070	1666	4	3198	0,00%	20,5/sec	59,1
4 /web-regis...	8000	2948	3311	4769	5207	6156	16	10750	0,00%	20,5/sec	183,5
6 /web-regis...	8000	345	228	872	1092	1569	4	4558	0,00%	20,7/sec	59,6
7 /web-regis...	8000	299	155	784	1016	1501	5	3878	0,00%	20,7/sec	107,4
8 /web-regis...	8000	298	144	784	1025	1646	4	3446	0,00%	20,8/sec	95,0
9 /web-regis...	8000	268	108	733	987	1586	3	3281	0,00%	20,8/sec	6,9
10 /web-regi...	8000	260	97	725	945	1437	3	3657	0,00%	20,8/sec	8,5
Total	64000	767	295	2545	3638	4943	3	10750	0,00%	163,7/sec	594,2

Figura 41. Resultado de la prueba realizada para la autenticación de usuarios

Elaborado por: (Sánchez & Castro)

Como se puede observar, el tiempo promedio para realizar la autenticación de usuarios es de 767 segundos, realizándose un total de 64000 peticiones al servidor.

El tiempo total utilizado para los 200 usuarios concurrentes se puede calcular con la siguiente fórmula:

Tiempo Total = #Muestras * Media = 64000 * 767 = 49088000 milisegundos

El tiempo promedio total requerido por cada usuario concurrente, se puede calcular de la siguiente manera:

((Tiempo Total/1000)/60)/usuarios recurrentes = ((49088000/1000)/60)/200 = 4,09 minutos

En la figura 42 podemos observar el tiempo de respuesta dependiendo de los procesos realizados, podemos destacar en la gráfica que el proceso que consume más tiempo j_security_check.

Gráfico tiempo de respuesta.

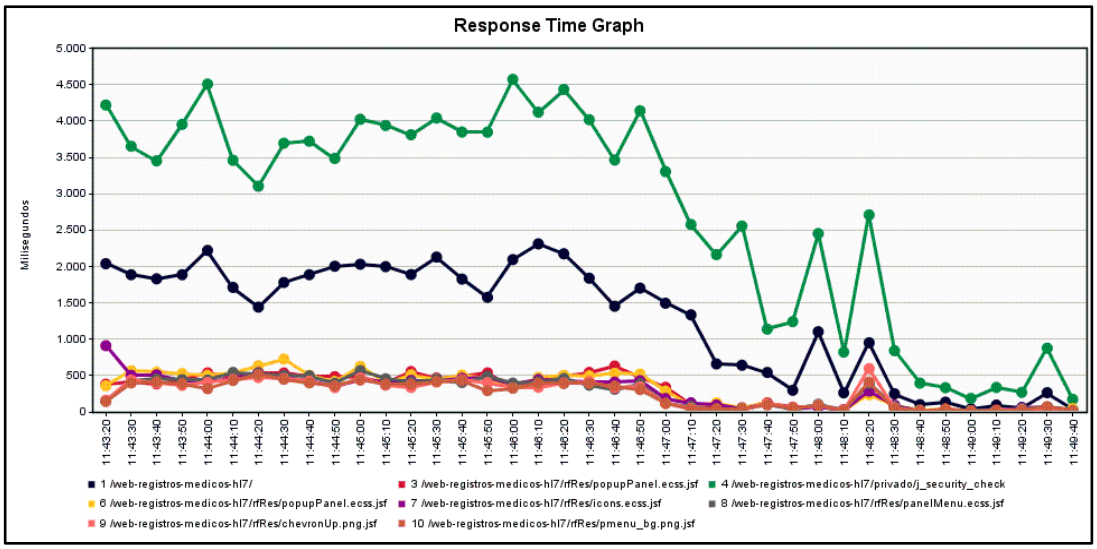


Figura 42. Representación gráfica de los tiempos de respuesta a procesos realizados

Elaborado por: (Sánchez & Castro)

Se valoraron los resultados a través de un intervalo de confianza para una distribución normal del 95%. La fórmula es la siguiente:

$[TP - Z\ 0.95 * D/\sqrt{n};\ TP + Z\ 0.95 * D/\sqrt{n}]$

Dónde:

- Tiempo Promedio (TP) de respuesta es: 767
- Desviación (D) es: 1162,42
- Tamaño de la muestra (n) es: 64000
- Z 0.95 es: 1,96

El intervalo resultante es:

$[767 - 1{,}96 * 1162{,}42/\sqrt{64000};\ 767 + 1{,}96 * 1162{,}42/\sqrt{64000}]$

$[767 - 1{,}96 * 4{,}5949;\ 767 + 1{,}96 * 4{,}5949] = [757{,}99;\ 776{,}00]$ en milisegundos.

La figura 43 muestra los resultados conseguidos al ejecutar las peticiones HTTP definidas, brindando en un gráfico de líneas los valores de tiempo en milisegundos, desviación típica, rendimiento, media y mediana.

Gráfico de Resultados.

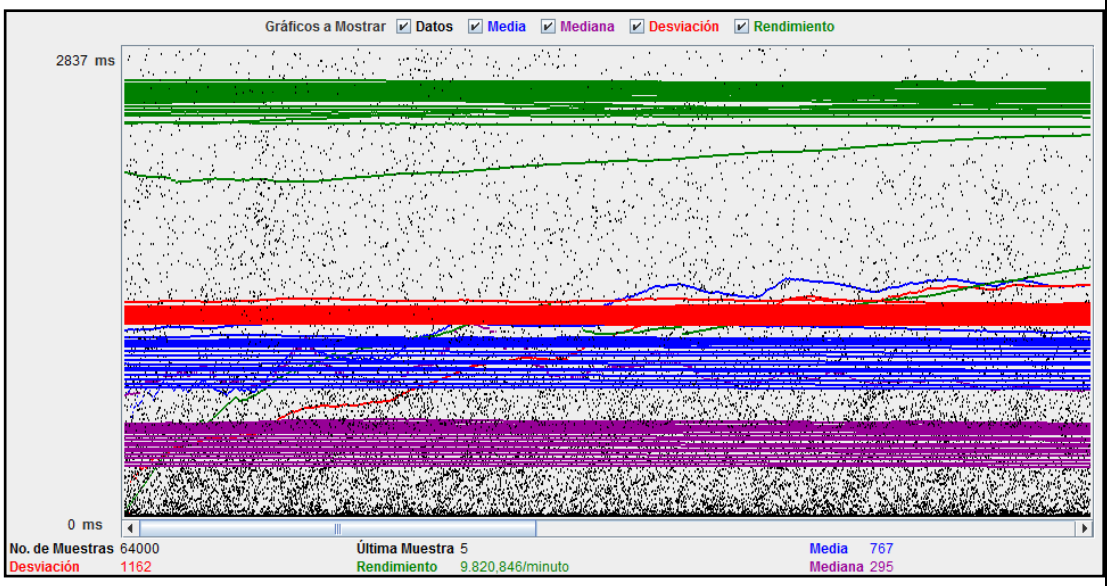


Figura 43. Resultados al ejecutar peticiones HTTP

Elaborado por: (Sánchez & Castro)

Por lo tanto, tabla 12 podemos resumir los resultados de la tercera iteración.

Tabla 12. Resultados.

Usuarios concurrentes	200
Repeticiones	40
Transacciones	64000
Tiempo Total	49088000 milisegundos
Tiempo de respuesta promedio	758 – 776 milisegundos
Desviación estándar	1162

Nota: Resultados totales de la interacción 3

Elaborado por: (Sánchez & Castro)

3.2.2. Pruebas módulo registros médicos pacientes

La prueba está realizada sobre el proceso de actualización de registros médicos en MySQL. Se realizaron las pruebas con los datos de la tabla 13.

Tabla 13. Parámetros prueba autenticación usuarios.

Iteración #	**Usuarios concurrentes**	**Repeticiones**	**Transacciones**
1	50	5	250
2	100	2	200
3	1000	1	1000

Nota: Detalle de los datos usados para realización de registros médicos de pacientes

Elaborado por: (Sánchez & Castro)

3.2.2.1. Iteración 1

Como se muestra en la figura 44, para la primera prueba se configuraron 50 usuarios concurrentes, cada 1 segundo, realizando 10 repeticiones.

Configuración de usuarios concurrentes.

Figura 44. Configuración de los datos de usuarios para realizar la prueba

Elaborado por: (Sánchez & Castro)

Los valores totales se obtenidos se muestran en la figura 45.

Informe agregado y resultados.

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec
1 /web-registr...	250	42	8	173	269	474	6	561	0,00%	36,9/min	2,5
2 /web-registr...	250	131	25	468	624	840	13	1057	0,00%	36,9/min	5,5
3 /web-registr...	250	104	15	427	501	793	9	1256	0,00%	36,9/min	6,4
4 /web-registr...	250	53945	70072	76986	77796	79983	3140	82234	0,00%	33,8/min	,4
5 /web-registr...	250	272	171	616	702	1570	34	5076	0,00%	34,1/min	71,1
6 /web-registr...	250	25184	11575	70940	74730	76007	44	76732	0,00%	34,1/min	86,4
Total	1500	13280	78	70718	74962	77649	6	82234	0,00%	3,4/sec	169,9

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Media de Bytes
1 /web-registro...	250	42	6	561	96,71	0,00%	36,9/min	2,53	4212,0
2 /web-registro...	250	131	13	1057	211,23	0,00%	36,9/min	5,50	9157,8
3 /web-registro...	250	104	9	1256	196,25	0,00%	36,9/min	6,45	10727,8
4 /web-registro...	250	53945	3140	82234	26465,16	0,00%	33,8/min	0,35	637,1
5 /web-registro...	250	272	34	5076	395,99	0,00%	34,1/min	71,07	128172,9
6 /web-registro...	250	25184	44	76732	29456,97	0,00%	34,1/min	86,44	155878,2
Total	1500	13280	6	82234	25995,74	0,00%	3,4/sec	169,87	51464,3

Figura 45. Resultado de la prueba realizada para el registro médico de pacientes

Elaborado por: (Sánchez & Castro)

Como se puede observar, el tiempo promedio para realizar el registro del paciente es de 13280 segundos, realizándose un total de 1500 peticiones al servidor.

El tiempo total utilizado para los 50 usuarios concurrentes se puede calcular con la siguiente fórmula:

Tiempo Total = #Muestras * Media = 1500 * 13280 = 19920000 milisegundos

El tiempo promedio total requerido por cada usuario concurrente, se puede calcular de la siguiente manera:

((Tiempo Total/1000)/60)/usuarios recurrentes = ((19920000/1000)/60)/50 = 6,64 minutos

En la figura 46 podemos observar el tiempo de respuesta dependiendo de los procesos realizados, podemos destacar en la gráfica que el proceso que consume más tiempo busquedaHCU la cual realiza la transacción con la base de datos.

Gráfico tiempo de respuesta.

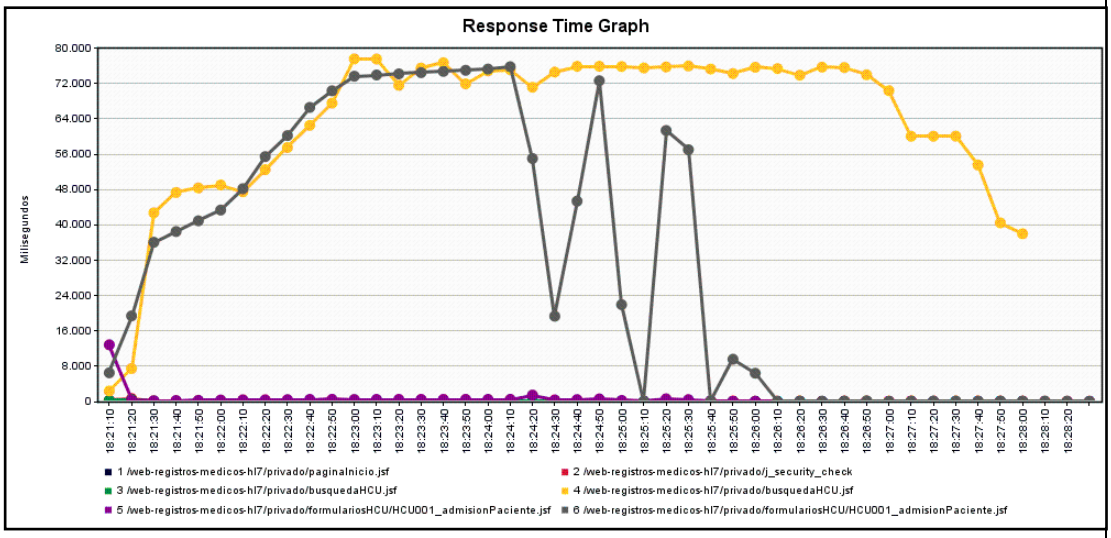


Figura 46. Representación gráfica de los tiempos de respuesta a procesos realizados

Elaborado por: (Sánchez & Castro)

Se valoraron los resultados a través de un intervalo de confianza para una distribución normal del 95%. La fórmula es la siguiente:

[TP – Z 0.95 * D/√n; TP + Z 0.95 * D/√n]

Dónde:

- Tiempo Promedio (TP) de respuesta es: 13280
- Desviación (D) es: 25995,74
- Tamaño de la muestra (n) es: 1500
- Z 0.95 es: 1,96

El intervalo resultante es:

[13280 – 1,96 * 25995,74/√1500; 13280 + 1,96 * 25995,74/√1500]

[13280 – 1,96 * 671,2071; 13280 + 1,96 * 671,2071] = [11964,43; 14595,5657] en milisegundos. = [12; 15] en segundos.

La figura 47 muestra los resultados conseguidos al ejecutar las peticiones HTTP definidas, brindando en un gráfico de líneas los valores de tiempo en milisegundos, desviación típica, rendimiento, media y mediana.

Gráfico de Resultados.

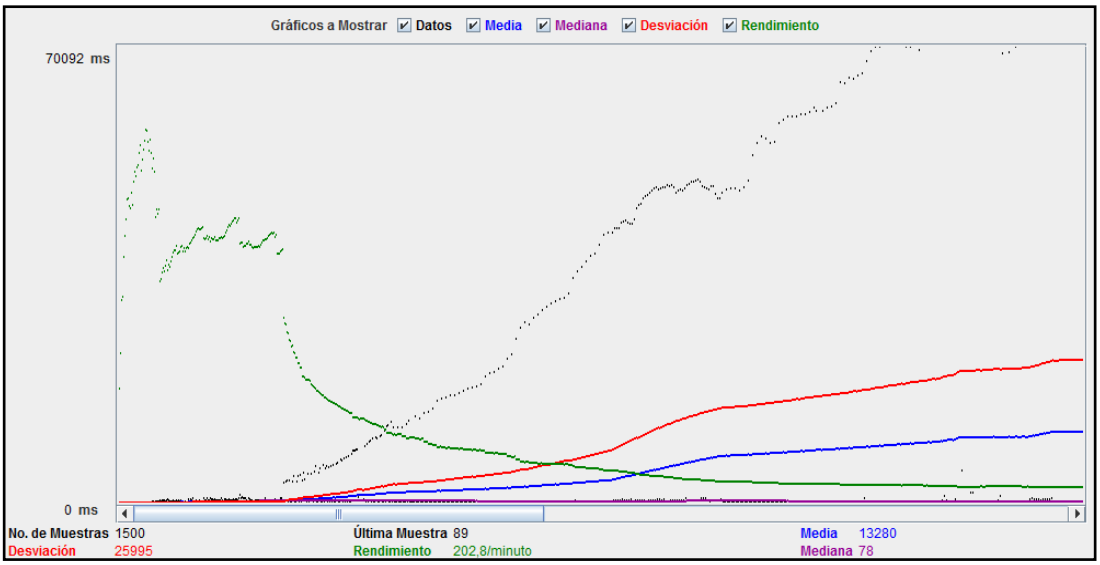


Figura 47. Resultados al ejecutar peticiones HTTP

Elaborado por: (Sánchez & Castro)

Por lo tanto, en la tabla 14 podemos resumir los resultados de la primera iteración.

Tabla 14. Resultados.

Usuarios concurrentes	50
Repeticiones	5
Transacciones	1500
Tiempo Total	19920000 milisegundos
Tiempo de respuesta promedio	12 – 15 segundos
Desviación estándar	25995

Nota: Resultados totales de la interacción 1

Elaborado por: (Sánchez & Castro)

3.2.2.2. Iteración 2

Como se muestra en la figura 48, para la segunda prueba se configuraron 100 usuarios concurrentes, cada 1 segundo, realizando 2 repeticiones.

Configuración de usuarios concurrentes.

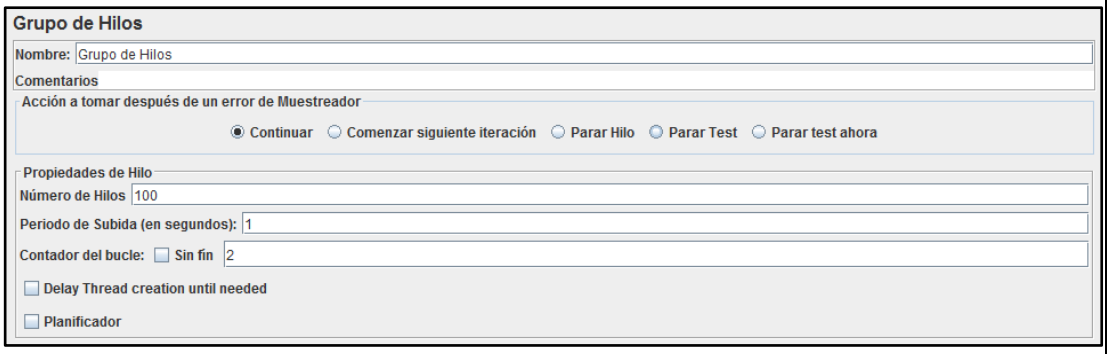


Figura 48. Configuración de los datos de usuarios para realizar la prueba

Elaborado por: (Sánchez & Castro)

Los valores totales se obtenidos se muestran en la figura 49.

Informe agregado.

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec
1 /web-registr...	200	712	245	1545	1636	1698	8	1834	0,00%	1,1/sec	4,5
2 /web-registr...	200	2212	398	4566	4686	4933	18	5022	0,00%	1,1/sec	9,8
3 /web-registr...	200	1631	339	3531	3682	3952	17	4018	0,00%	1,1/sec	11,8
4 /web-registr...	200	86370	41892	150937	151066	151155	8429	151213	0,00%	36,9/min	,4
5 /web-registr...	200	1770	437	6562	7021	8887	42	9521	0,00%	38,0/min	23,1
6 /web-registr...	200	36376	299	107378	117214	127910	51	131367	0,00%	39,1/min	51,6
Total	1200	21512	1512	107378	150283	151044	8	151213	0,00%	3,6/sec	83,8

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Media de Bytes
1 /web-registro...	200	712	8	1834	705,35	0,00%	1,1/sec	4,50	4212,0
2 /web-registro...	200	2212	18	5022	2157,34	0,00%	1,1/sec	9,83	9157,6
3 /web-registro...	200	1631	17	4018	1610,35	0,00%	1,1/sec	11,83	10727,9
4 /web-registro...	200	86370	8429	151213	62275,77	0,00%	36,9/min	0,39	641,4
5 /web-registro...	200	1770	42	9521	2392,86	0,00%	38,0/min	23,11	37351,6
6 /web-registro...	200	36376	51	131367	42760,63	0,00%	39,1/min	51,60	81087,0
Total	1200	21512	8	151213	44229,98	0,00%	3,6/sec	83,76	23862,9

Figura 49. Resultado de la prueba realizada para el registro médico de pacientes

Elaborado por: (Sánchez & Castro)

Como se puede observar, el tiempo promedio para realizar el registro del paciente es de 21512 segundos, realizándose un total de 1200 peticiones al servidor.

El tiempo total utilizado para los 100 usuarios concurrentes se puede calcular con la siguiente fórmula:

Tiempo Total = #Muestras * Media = 1200 * 21512 = 25814400 milisegundos

El tiempo promedio total requerido por cada usuario concurrente, se puede calcular de la siguiente manera:

((Tiempo Total/1000)/60)/usuarios recurrentes = ((25814400/1000) /60)/100 = 4,3 minutos

En la figura 50 podemos observar el tiempo de respuesta dependiendo de los procesos realizados, podemos destacar en la gráfica que el proceso que consume más tiempo busquedaHCU la cual realiza la transacción con la base de datos.

Gráfico tiempo de respuesta.

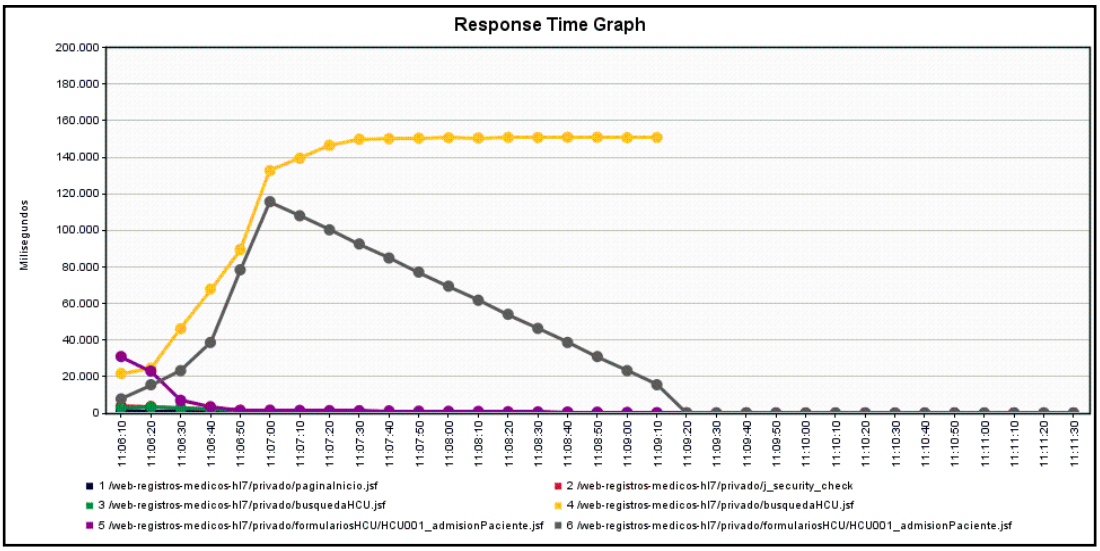


Figura 50. Representación gráfica de los tiempos de respuesta a procesos realizados

Elaborado por: (Sánchez & Castro)

Se valoraron los resultados a través de un intervalo de confianza para una distribución normal del 95%. La fórmula es la siguiente:

[TP – Z 0.95 * D/√n; TP + Z 0.95 * D/√n]

Dónde:

- Tiempo Promedio (TP) de respuesta es: 21512
- Desviación (D) es: 44229,98
- Tamaño de la muestra (n) es: 1200
- Z 0.95 es: 1,96

El intervalo resultante es:

[21512 – 1,96 * 44229/√1200; 21512 + 1,96* 44229/√1200]

[21512 – 1,96 * 1276,78; 21512 + 1,96 * 1276,78] = [1900; 24014,48] en milisegundos. = [19; 24] en segundos.

La figura 51 muestra los resultados conseguidos al ejecutar las peticiones HTTP definidas, brindando en un gráfico de líneas los valores de tiempo en milisegundos, desviación típica, rendimiento, media y mediana.

Gráfico de Resultados.

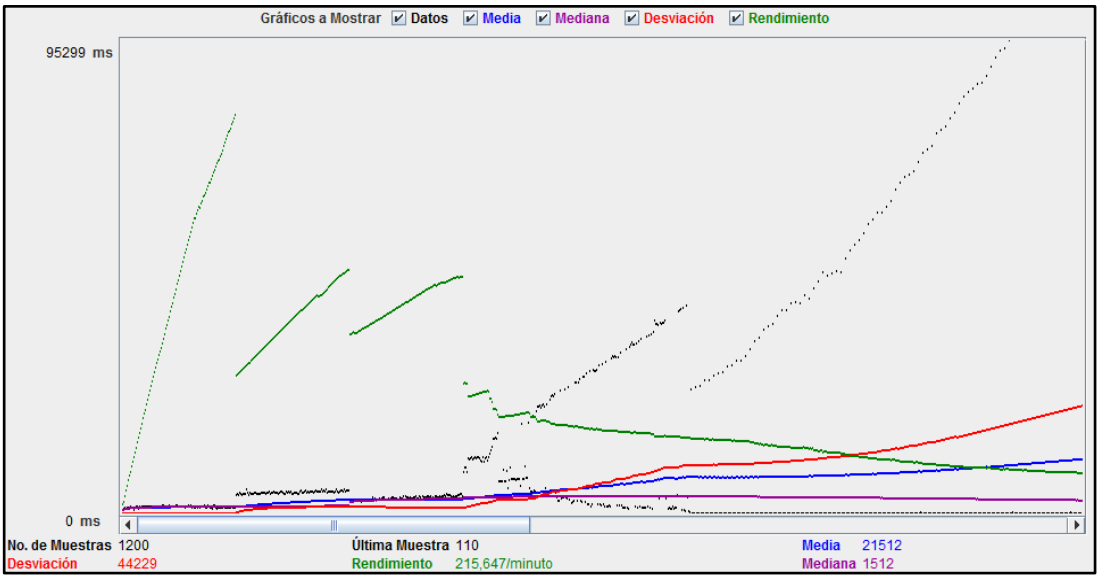


Figura 51. Resultados al ejecutar peticiones HTTP

Elaborado por: (Sánchez & Castro)

Por lo tanto, en la tabla 15 podemos resumir los resultados de la segunda iteración.

Tabla 15. Resultados.

Usuarios concurrentes	100
Repeticiones	2
Transacciones	1200
Tiempo Total	25814400 milisegundos
Tiempo de respuesta promedio	19 – 24 segundos
Desviación estándar	44229

Nota: Resultados totales de la interacción 2

Elaborado por: (Sánchez & Castro)

3.2.2.3. Iteración 3

Como se muestra en la figura 52, para la tercera prueba se configuraron 1000 usuarios concurrentes, cada 1 segundo, realizando 1 repetición.

Configuración de usuarios concurrentes.

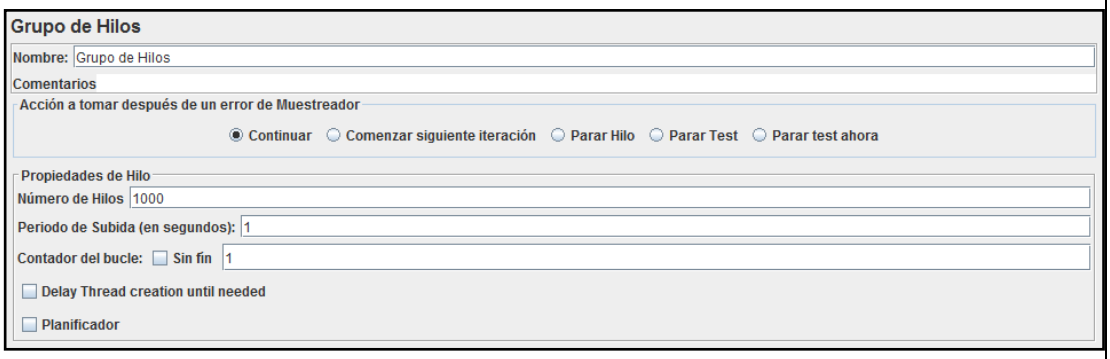


Figura 52. Configuración de los datos de usuarios para realizar la prueba

Elaborado por: (Sánchez & Castro)

Los valores totales se obtenidos se muestran en la figura 53.

Informe agregado.

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec
1 /web-registr...	1000	11443	8357	21113	21143	21216	1643	33018	48,80%	27,4/sec	88,7
2 /web-registr...	1000	12155	11459	21101	21142	21201	3	21228	48,80%	22,0/sec	126,0
3 /web-registr...	1000	9086	9518	14217	20998	21095	4	21163	48,80%	16,1/sec	105,1
4 /web-registr...	1000	73071	25067	218597	295069	483291	3	634030	48,80%	1,5/sec	2,1
5 /web-registr...	1000	8396	760	37938	43495	53303	2	59258	48,80%	1,4/sec	30,9
6 /web-registr...	1000	460719	32213	1823551	1881887	1977845	3	2103685	48,80%	27,7/min	14,9
Total	6000	95812	9140	203820	553741	1871871	2	2103685	48,80%	2,8/sec	32,6

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Media de Bytes
1 /web-registro...	1000	11443	1643	33018	7453,15	48,80%	27,4/sec	88,67	3318,3
2 /web-registro...	1000	12155	3	21228	6439,13	48,80%	22,0/sec	126,00	5861,8
3 /web-registro...	1000	9086	4	21163	5016,27	48,80%	16,1/sec	105,13	6688,6
4 /web-registro...	1000	73071	3	634030	105804,74	48,80%	1,5/sec	2,09	1404,4
5 /web-registro...	1000	8396	2	59258	14850,96	48,80%	1,4/sec	30,88	22362,3
6 /web-registro...	1000	460719	3	2103685	631884,38	48,80%	27,7/min	14,87	33036,3
Total	6000	95812	2	2103685	309238,11	48,80%	2,8/sec	32,57	12111,9

Figura 53. Resultado de la prueba realizada para el registro médico de pacientes

Elaborado por: (Sánchez & Castro)

Como se puede observar, el tiempo promedio para realizar el registro del paciente es de 95812 milisegundos, realizándose un total de 6000 peticiones al servidor.

El tiempo total utilizado para los 100 usuarios concurrentes se puede calcular con la siguiente fórmula:

Tiempo Total = #Muestras * Media = 6000 * 95812 = 574872000 milisegundos

El tiempo promedio total requerido por cada usuario concurrente, se puede calcular de la siguiente manera:

((Tiempo Total/1000)/60)/usuarios recurrentes = ((574872000/1000)/60)/1000 = 9 minutos

En la figura 54 podemos observar el que el proceso que consume más tiempo en las anteriores iteraciones busquedaHCU cae súbitamente, esto representa una caída del servidor.

Gráfico tiempo de respuesta.

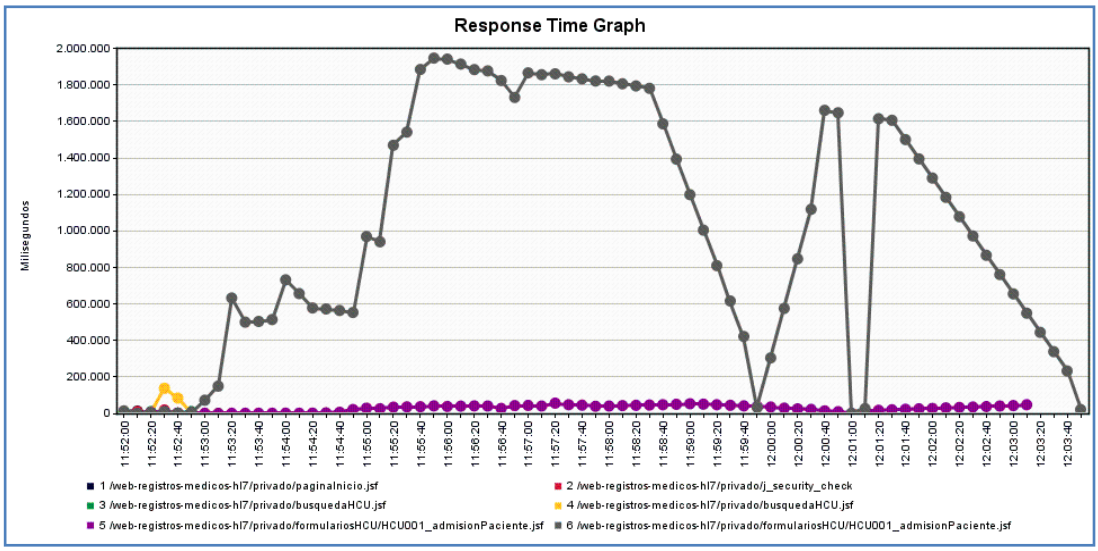


Figura 54. Representación gráfica de los tiempos de respuesta a procesos realizados

Elaborado por: (Sánchez & Castro)

Se valoraron los resultados a través de un intervalo de confianza para una distribución normal del 95%. La fórmula es la siguiente:

TP – Z 0.95 * D/√n; TP + Z 0.95 * D/√n]

Dónde:

- Tiempo Promedio (TP) de respuesta es: 95812
- Desviación (D) es: 309238,11
- Tamaño de la muestra (n) es: 6000
- Z 0.95 es: 1,96

El intervalo resultante es:

[95812 – 1,96 * 309238,11/√6000; 95812 + 1,96 * 309238,11/√6000]

[95812 – 1,96 * 3992; 95812 + 1,96 * 3992] = [87987; 103636,32] en milisegundos.

= [88; 103] en segundos.

La figura 55 muestra los resultados conseguidos al ejecutar las peticiones HTTP definidas, brindando en un gráfico de líneas los valores de tiempo en milisegundos, desviación típica, rendimiento, media y mediana.

Gráfico de Resultados.

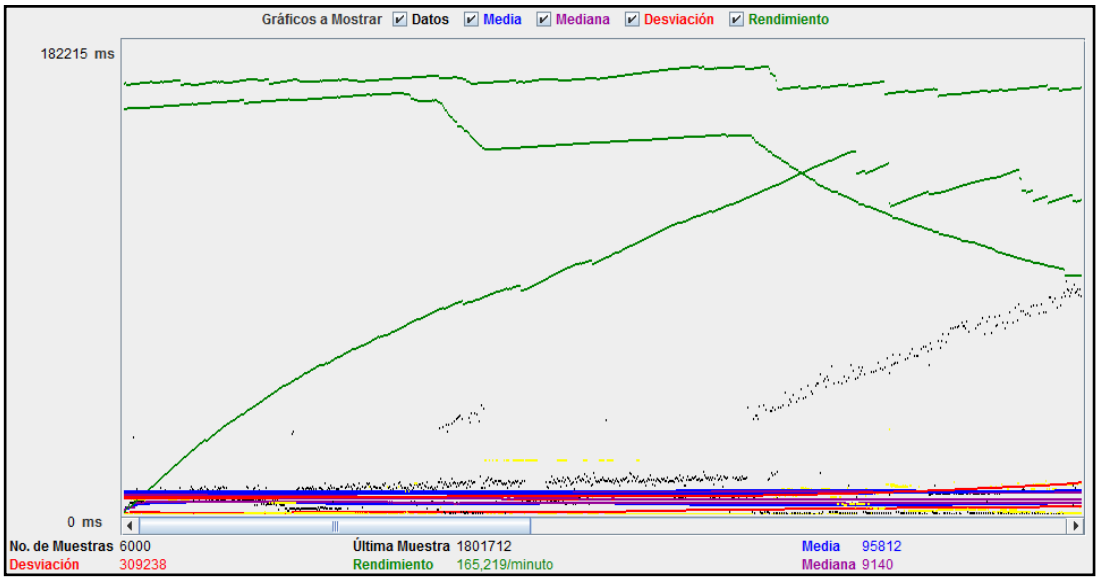


Figura 55. Resultados al ejecutar peticiones HTTP

Elaborado por: (Sánchez & Castro)

Por lo tanto, en la tabla 16 podemos resumir los resultados de la segunda iteración. Es importante mencionar que el servidor sufrió una caída y dejo de responder.

Tabla 16. Resultados.

Usuarios concurrentes	1000
Repeticiones	1
Transacciones	6000
Tiempo Total	574872000 milisegundos
Tiempo de respuesta promedio	88 – 103 segundos
Desviación estándar	309238

Nota: Resultados totales de la interacción 3

Elaborado por: (Sánchez & Castro)

3.2.3. Módulo imágenes médicas pacientes

La prueba está realizada sobre el proceso de recuperación de imágenes médicas DICOM provenientes de Hadoop, se realizó un monitoreo utilizando la herramienta Ganglia. Se realizaron las pruebas con los datos de la tabla 17.

Tabla 17. Parámetros prueba autenticación usuarios.

Iteración #	Usuarios concurrentes	Repeticiones	Transacciones	CPU	RAM
1	100	1	100	6	12GB

Nota: Detalle de los datos usados para realización de recuperación de imágenes médicas

Elaborado por: (Sánchez & Castro)

3.2.3.1. Iteración 1

Como se muestra en la figura 56, para la primera prueba se configuraron 100 usuarios concurrentes, cada 1 segundo, realizando 1 repeticiones, la prueba inicio a las 09:04:00 y finalizo 09:05:05.

Configuración de usuarios concurrentes.

Figura 56. Configuración de los datos de usuarios para realizar la prueba

Elaborado por: (Sánchez & Castro)

Primero mostramos en la figura 57 una vista comprensiva del uso de recursos de los diferentes nodos del sistema multiagente.

Uso de recursos monitor Ganglia.

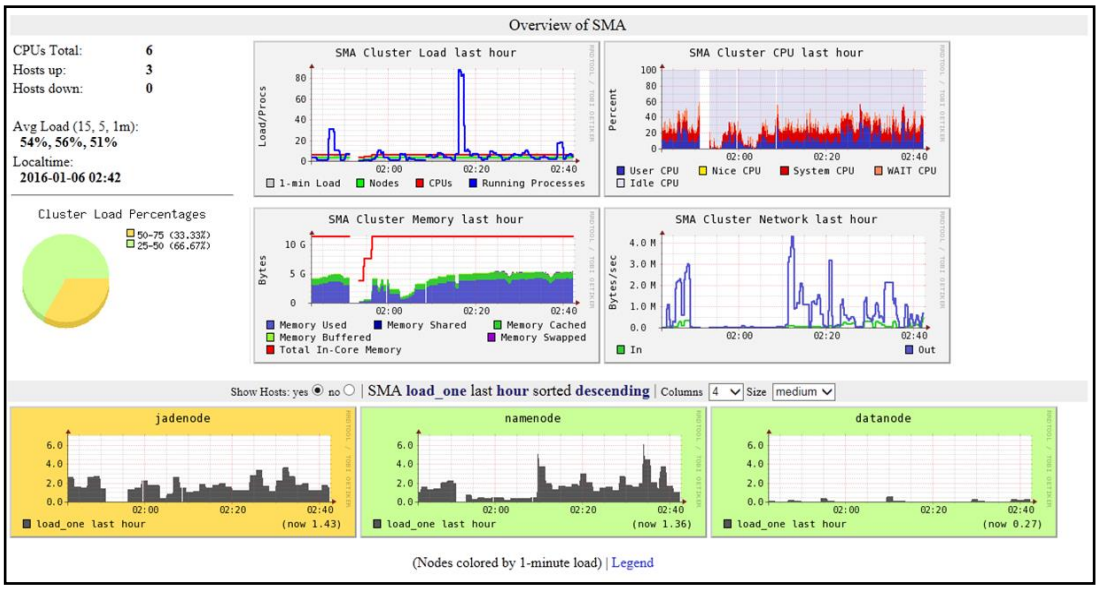


Figura 57. Representación gráfica de los recursos del sistema multiagente en cada nodo

Elaborado por: (Sánchez & Castro)

Los valores totales se obtenidos se muestran en la figura 58.

Informe agregado y resumen.

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec
1 /web-registros-m...	100	392	104	1306	1481	1543	17	4117	0,00%	24,3/sec	99,9
2 /web-registros-m...	100	2209	2309	3033	3231	3328	552	3480	0,00%	22,3/sec	199,1
3 /web-registros-m...	100	645	640	1030	1195	1377	64	1439	0,00%	42,1/sec	440,7
4 /web-registros-m...	100	20885	22343	34909	38005	40400	2082	43033	0,00%	2,2/sec	,8
5 /web-registros-m...	100	2377	583	5813	8083	11982	54	12688	0,00%	2,0/sec	64,1
6 /web-registros-m...	100	1850	111	6176	8004	9409	12	11351	0,00%	1,9/sec	34,6
7 /web-registros-m...	100	3595	3019	7637	9571	12857	227	17305	0,00%	1,8/sec	2016,4
8 /web-registros-m...	100	4291	3939	7524	8660	13961	568	16023	0,00%	1,8/sec	5285,1
9 /web-registros-m...	100	25993	27051	33991	34607	37057	14244	37617	0,00%	1,4/sec	201,5
Total	900	6915	2527	26441	31539	37122	12	43033	0,00%	11,6/sec	5569,0

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Media de Bytes
1 /web-registros-medic...	100	392	17	4117	625,73	0,00%	24,3/sec	99,91	4212,0
2 /web-registros-medic...	100	2209	552	3480	748,62	0,00%	22,3/sec	199,09	9157,7
3 /web-registros-medic...	100	645	64	1439	321,36	0,00%	42,1/sec	440,73	10727,7
4 /web-registros-medic...	100	20885	2082	43033	10932,68	0,00%	2,2/sec	0,80	367,0
5 /web-registros-medic...	100	2377	54	12688	2846,10	0,00%	2,0/sec	64,09	32822,7
6 /web-registros-medic...	100	1850	12	11351	2719,50	0,00%	1,9/sec	34,60	18766,0
7 /web-registros-medic...	100	3595	227	17305	3240,62	0,00%	1,8/sec	2016,36	1140881,0
8 /web-registros-medic...	100	4291	568	16023	2667,96	0,00%	1,8/sec	5285,05	3058803,0
9 /web-registros-medic...	100	25993	14244	37617	6758,56	0,00%	1,4/sec	201,52	143443,7
Total	900	6915	12	43033	10147,09	0,00%	11,6/sec	5568,96	491020,1

Figura 58. Resultado de la prueba realizada para la recuperación de imágenes médicas

Elaborado por: (Sánchez & Castro)

Como se puede observar, el tiempo promedio para recuperar la imagen médica es descendió a 6915 milisegundos, realizándose un total de 900 peticiones al servidor.

El tiempo total utilizado para los 100 usuarios concurrentes se puede calcular con la siguiente fórmula:

Tiempo Total = #Muestras * Media = 900 * 6915 = 6223500 milisegundos

El tiempo promedio total requerido por cada usuario concurrente, se puede calcular de la siguiente manera:

((Tiempo Total/1000)/60)/usuarios recurrentes = ((6223500/1000)/60)/100 = 1,03 minutos

En la figura 59 podemos observar el tiempo de respuesta dependiendo de los procesos realizados, podemos destacar en la gráfica que el proceso que consume más tiempo HCU002_examenFisico la cual está encargado de realizar recuperar una imagen médica DICOM desde Hadoop.

Gráfico tiempo de respuesta.

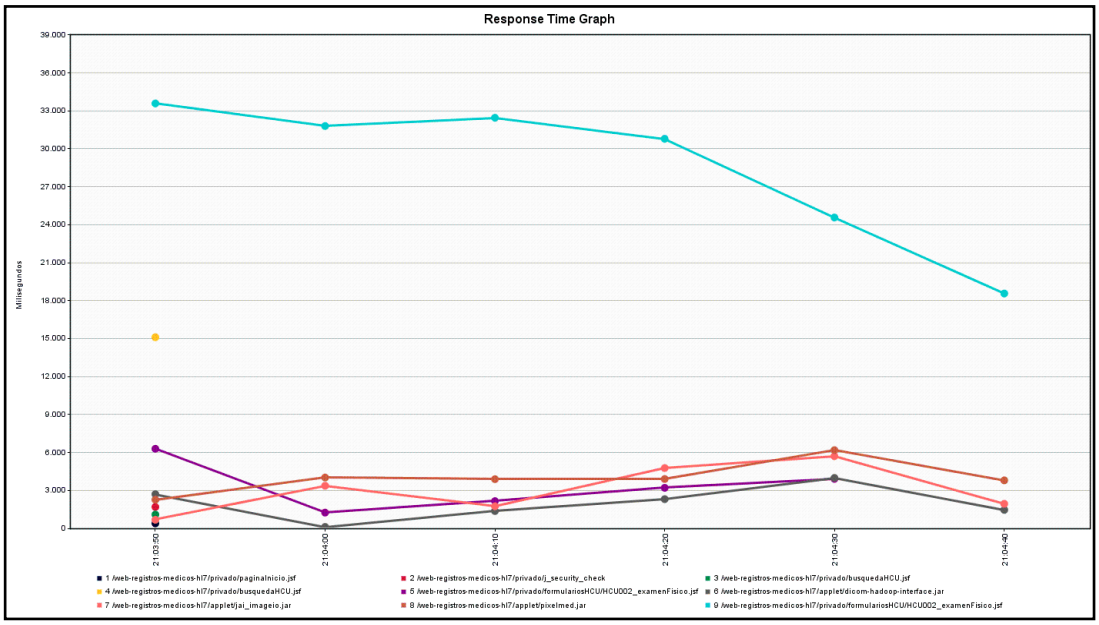


Figura 59. Representación gráfica de los tiempos de respuesta a procesos realizados

Elaborado por: (Sánchez & Castro)

Se valoraron los resultados a través de un intervalo de confianza para una distribución normal del 95%. La fórmula es la siguiente:

$[TP - Z\ 0.95 * D/\sqrt{n};\ TP + Z\ 0.95 * D/\sqrt{n}]$

Dónde:

- Tiempo Promedio (TP) de respuesta es: 6915
- Desviación (D) es: 10147,09
- Tamaño de la muestra (n) es: 900
- Z 0.95 es: 1,96 %

El intervalo resultante es:

[6915 – 1,96 * 10147,09/√900; 6915+ 1,96 * 10147,09/√900]

[6915 – 1,96 * 338,236; 6915 + 1,96 * 338,236] = [6252,05744; 7577,94256] en milisegundos. = [6; 7] en segundos.

La figura 60 muestra los resultados conseguidos al ejecutar las peticiones HTTP definidas, brindando en un gráfico de líneas los valores de tiempo en milisegundos, desviación típica, rendimiento, media y mediana.

Gráfico de Resultados.

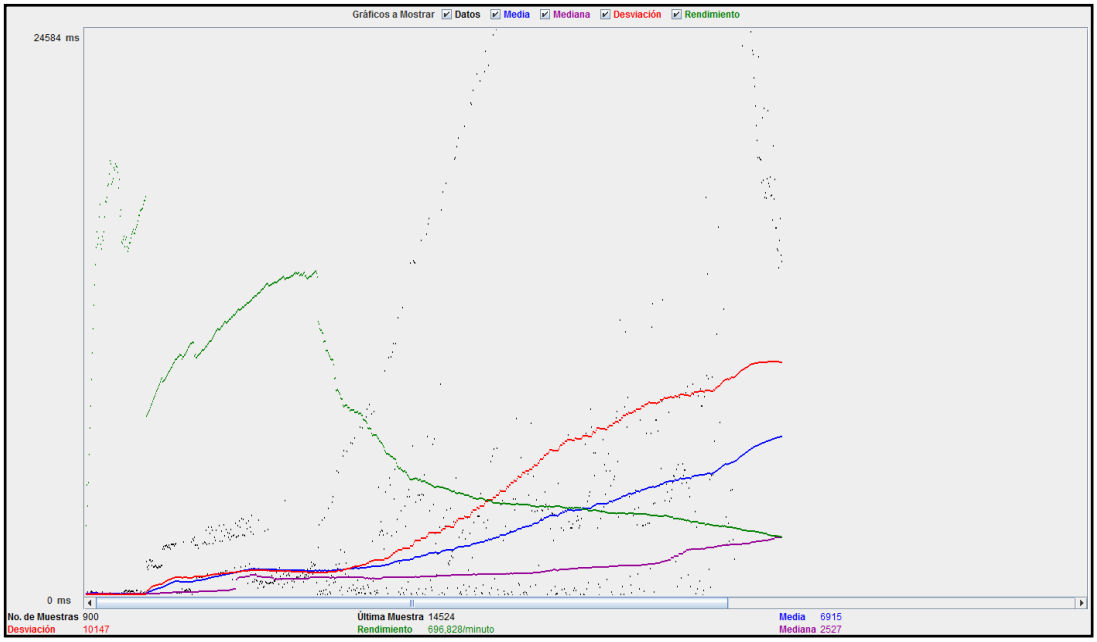


Figura 60. Resultados al ejecutar peticiones HTTP

Elaborado por: (Sánchez & Castro)

Por lo tanto, en la tabla 18 podemos resumir los resultados de la iteración.

Tabla 18. Resultados.

Usuarios concurrentes	100
Repeticiones	1
Transacciones	900
Tiempo Total	6223500 milisegundos

Tiempo de respuesta promedio	6 – 7 segundos
Desviación estándar	10147

Nota: Resultados totales de la interacción 1

Elaborado por: (Sánchez & Castro)

Los resultados de monitoreo junto con las especificaciones de hardware que se tuvieron para esta prueba fueron las siguientes.

El servidor NameNode posee la especificación de hardware de la tabla 19.

Tabla 19. Métricas NameNode.

Arquitectura	x86_64
Sistema operativo	Linux
Versión	2.6.32-431.el6.x86_64
Número CPUs	3 CPUs
Velocidad CPU	2693 MHz
Memoria Total	5993992 KB
Swap Espacio Total	2097144 KB

Nota: Especificación del hardware del NameNode

Elaborado por: (Sánchez & Castro)

En la figura 61 podemos observar las métricas más destacables del servidor NameNode.

Métricas más destacadas NameNode.

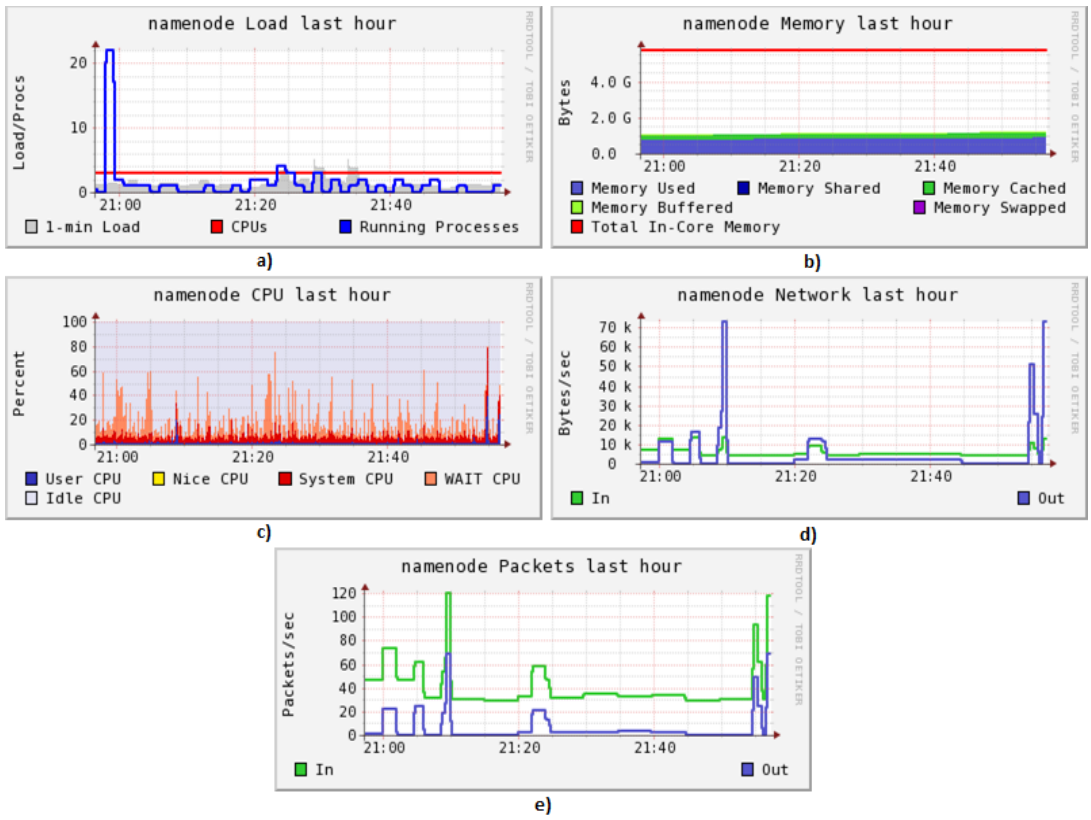


Figura 61. Representación gráfica de las métricas del NameNode: a) Carga, b) Memoria, c) CPU, d) Red y e) Paquetes

Elaborado por: (Sánchez & Castro)

El servidor DataNode posee la especificación de hardware de la tabla 20.

Tabla 20. Métricas DataNode.

Arquitectura	x86_64
Sistema operativo	Linux
Versión	2.6.32-431.el6.x86_64
Número CPUs	1 CPUs
Velocidad CPU	2693 MHz
Memoria Total	1923456 KB

Swap Espacio Total	2097144 KB

Nota: Especificación del hardware del DataNode

Elaborado por: (Sánchez & Castro)

En la figura 62 podemos observar las métricas más destacables del servidor DataNode.

Métricas más destacadas DataNode.

Figura 62. Representación gráfica de las métricas del DataNode: a) Carga, b) Memoria, c) CPU, d) Red y e) Paquetes

Elaborado por: (Sánchez & Castro)

El servidor JadeNode posee la especificación de hardware de la tabla 21.

Tabla 21. Métricas JadeNode.

Arquitectura	x86_64
Sistema operativo	Linux
Versión	2.6.32-431.el6.x86_64
Número CPUs	2 CPUs
Velocidad CPU	2693 MHz
Memoria Total	3925712 KB
Swap Espacio Total	1675256 KB

Nota: Especificación del hardware del JadeNode

Elaborado por: (Sánchez & Castro)

En la figura 63 podemos observar las métricas más destacables del servidor JadeNode.

Métricas más destacadas JadeNode.

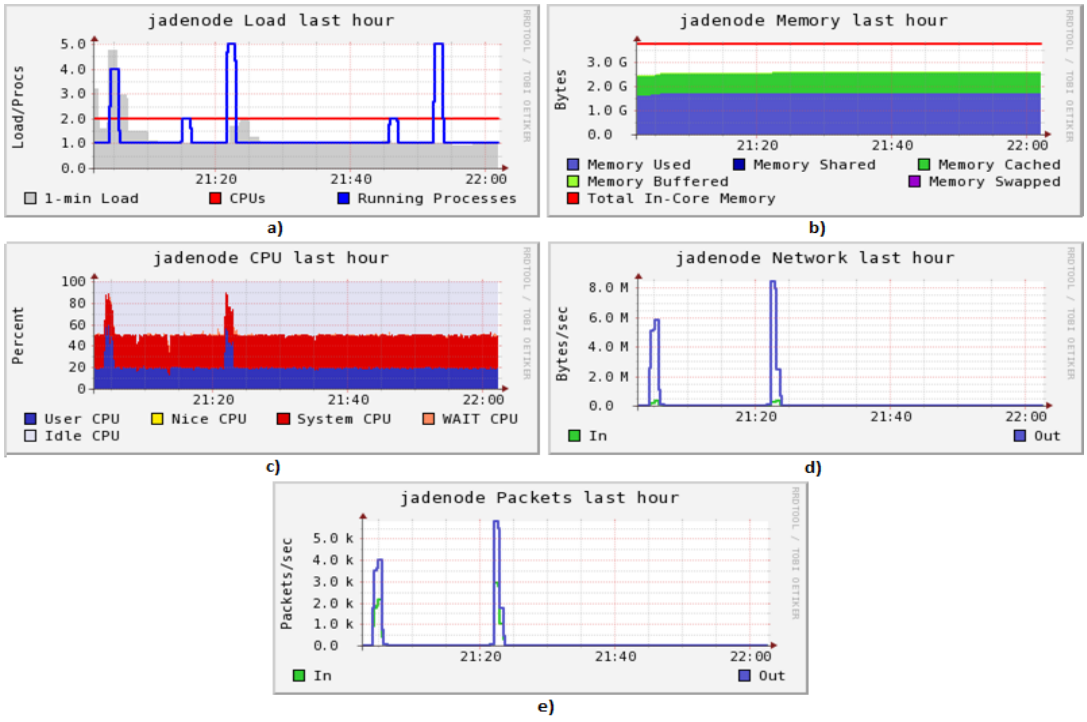


Figura 63. Representación gráfica de las métricas del JadeNode: a) Carga, b) Memoria, c) CPU, d) Red y e) Paquetes

Elaborado por: (Sánchez & Castro)

Como resultado de estas pruebas, la tabla 22 muestra la configuración mínima de hardware para que el sistema funcione correctamente.

Tabla 22. Configuración mínima de hardware sistema multiagente.

NameNode	
Número de CPUs	2 CPUs
Memoria RAM	4 GB
DataNode	
Número de CPUs	2 CPUs
Memoria RAM	2 GB
Jadenode	
Número de CPUs	2 CPUs
Memoria RAM	4 GB

Nota: Requisitos de hardware para el buen funcionamiento del sistema multiagente

Elaborado por: (Sánchez & Castro)

CONCLUSIONES

- Se propuso una arquitectura multiagente para cumplir con los requisitos de interoperabilidad entre sistemas médicos existentes y se demostró cómo las características de los sistemas multiagentes pueden mejorar en términos de robustez, fiabilidad y modularidad.
- El framework jade es ideal para realizar prototipos de sistemas multiagentes escritos en java, los mismos que pueden ser fácilmente adaptados con tecnología de terceros, además reduce drásticamente tanto la curva de aprendizaje como el tiempo de desarrollo, ya que abstrae la implementación de los sistemas multiagentes de forma transparente hacia el programador.
- Fue implementada una interfaz HL7 en el sistema multiagente, capaz de procesar mensajes que incluyen datos médicos. Lo que nos abre a la posibilidad de comunicarnos con otros sistemas médicos locales y remotos.
- El reto de la distribución y paralelización se manejó integrando el sistema multiagente con Hadoop para el almacenamiento distribuido de archivos grandes como las imágenes médicas DICOM, esto nos permitió centrarnos en el modelo de los agentes y su optimización dejando que Hadoop se encargue de cuestiones como tolerancia a fallos y escalabilidad.

RECOMENDACIONES

- Demostrada la factibilidad de la arquitectura multiagente, como siguiente paso se podría colocar ajusten en el sistema según las necesidades, con el fin de adaptar e implementar este proyecto a una o varias localidades médicas.
- Ampliar la funcionalidad del visualizador de la imagen médica DICOM, para el aplicativo solamente se implementó la función básica de la librería pixelmed.jar.
- Permitir subir más de una imagen médica DICOM al mismo tiempo y poder organizarlas por carpetas, en este momento solo se permite escoger y subir una a la vez y se encaminan a una misma ruta.
- El servicio web jade proporcionado en este proyecto podría ser fácilmente consumido en nuevos proyectos, como aplicativos móviles.
- Aprovechar el intercambio de información a través del estándar HL7, para implementar conexiones con distintos sistemas de información en el área de la salud remotos.
- Como parte de una nueva investigación, se podría realizar un ajuste o "tuning" en inglés, considerando implementar buenas prácticas en todo el ambiente del Sistema multiagente, lo que comprendería MySQL, Hadoop, Jboss, Apache Tomcat y Jade. Una de estas consideraciones podría ser migrar los datos relacionales de la base de datos MySQL a Hadoop y ajustar al sistema a este nuevo cambio.

REFERENCIAS

Aguilar, J., Rios Bolivar, A., Hidrobo, F., & Cerrada, M. (2012). *Sistemas MultiAgentes y sus aplicaciones al campo industrial.* Maracaibo: Universidad de los Andes.

Albert, M., Laengle, T., Woern, H., Capobianco, M., & Brighenti, A. (2003). Multi-agent Systems for Industrial Diagnostics. *Proceedings of 5th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, (págs. 483–488). Washington, DC.

Alegsa. (1998 - 2015). *Alegsa*. Obtenido de http://www.alegsa.com.ar/Dic/agente%20de%20software.php

Andrade, J. M. (s.f.). Modelo Multidimencional Base de datos Multidimensional: Uso para BIG DATA.

Apache JMeter Team. (1999). Obtenido de http://jmeter.apache.org/

ApacheHDFS. (29 de Junio de 2015). *Apache Hadoop*. Obtenido de http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

ApacheYarn. (29 de Junio de 2015). *Hadoop Apache*. Obtenido de http://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YARN.html

balsamiq. (2008). Obtenido de https://balsamiq.com/

Bellifemine, Caire, & Greenwood. (2007). *Developing Multi-Agent systems.* Wiley.

Bellifemine, F., Poggi, A., & Rimassa, G. (2001). Developing Multi Agent Systems with a FIPA-Compliant Agent Framework. *In Software–Practice and Experience, 31*, 103–128.

Betanco, Cruz, F. d., & Yamaira. (2013). SLD105 Propuesta de arquitectura de sistemas multiagente para el sector sanitario. *Ediciones pensando el futuro*, 89.

Botti, V., & Julián, V. (1995). Agentes Inteligentes: el siguiente paso en la Inteligencia Artificial: el siguiente paso en la Inteligencia Artificial. *Novática: Revista de la Asociación de Técnicos de Informática*.

Brooks, R. (1991). Intelligence without Representation. *Artificial Intelligence*(47), 139-159.

Bussmann, S., & Muller, J. (1992). A Negotiation Framework for Co-operating Agents. *Proceedings of CKBS-SIG* (págs. 1-17). Keele: In Deen, S.M.

Callejas, Parada, & Alarcón. (2012). Modelado e implementación de un sistema multiagente para el diagnósico de nefermedades de transmisión sexual. *Entramado*, 190-208.

Caparrini. (12 de Junio de 2014). *Slideshare.net.* Recuperado el 16 de Diciembre de 2015, de Slideshare.net: http:es.slideshare.net/FernandoCaparrini/sistemas multiagente

Capozzi, D., & Lanzola, G. (2010). An Agent-Based Architecture for Home Care Monitoring and Education of Chronic Patients. *2010 Complexity in Engineering*, (págs. 138-140).

Corchado, J. M., Alonso, R. S., & Tapia, D. I. (2011). Improving an Ambient Intelligence Based Multi-Agent System for Alzheimer Health Care.

Doukas, C., Pliakas, T., & Maglogiannis, I. (2010). Mobile healthcare information management utilizing cloud computing and android os. *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, (págs. 1037-1040).

Eclipse Foundation. (2004). Obtenido de https://eclipse.org/

Ecured. (18 de Julio de 2014). *Ecured.* Recuperado el 16 de Diciembre de 2015, de Ecured: http://www.ecured.cu/Sistemas_Multiagentes

Edureka. (2014). Obtenido de http://www.slideshare.net/EdurekaIN/hadoop-developer-36623915

Fic. (s.f.). DICOM (Digital Imaging and Communications in Medicine).

Ganglia Project. (2011). Obtenido de http://www.ganglia.info/

Georgeff, M., & Lansky, A. (1987). Reactive Reasoning and Planning: an Experiment with a Mobile Robot. *Proceedings of the 7th National Conference on Artificial Intelligence*, (págs. 677–682). Seattle, WA.

Greenwood, D., Vitaglione, G., Keller, L., & Calisti, M. (2006). Service Level Agreement Management with Adaptive Coordination. *Proceedings of the International Conference on Networking and Services (ICNS'06).* Silicon Valley, USA.

Hadoop, A. (2014). *hadoop.apache.org*. Obtenido de hadoop.apache.org

Hadoop, Apache. (29 de 06 de 2015). *Apache Hadoop Yarn*. Obtenido de https://hadoop.apache.org/docs/r2.7.1/hadoop-yarn/hadoop-yarn-site/YARN.html

Herranz, F. B. (2013). Desarrollo de aplicaciones DICOM para la gestión de imágenes biomédicas.

HL7. (2007 - 2016). *HL7*. Obtenido de http://www.hl7.org/implement/standards/product_brief.cfm?product_id=213

Hortonworks. (30 de Junio de 2011). *slideshare*. Obtenido de http://www.slideshare.net/hortonworks/nextgen-apache-hadoop-mapreduce

Hyacinth S, N., & Divine T, N. (1996). A Perspective on Software Agents Research.

IBM. (2012). *IBM*. Obtenido de https://www.ibm.com/developerworks/ssa/local/im/que-es-big-data/

Iglesias, C. A., Garijo, M., González, J. C., & Velasco, J. R. (1996). A methodological proposal for multiagent systems development extending CommonKADS. *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, (págs. 25-1/17). Banff, Canada.

Infochannel. (2012). *Infochannel*. Obtenido de http://www.infochannel.com.mx/infografia-big-data-big-business

Isern, D., Sánchez, D., & Moreno, A. (2009). Agents applied in health care: A review. *Elsevier*, 62.

Jennings, N. (1994). The Archon System and its Applications. *Proceedings of the 2nd International Working Conference on Cooperating Knowledge Based Systems (CKBS-94)* (págs. 13-29). University of Keele, UK: Dake Centre.

Jennings, N., & Wooldridge, M. (1998). Applications of Intelligent Agents. En *In Agent Technology: Foundations, Applications, and Markets* (págs. 3-28). Secaucus, NJ: Springer-Verlag,.

Klusch, M. (2001). Information Agent Technology for the Internet: a Survey. *Data Knowledge Engineering, 36*, 337–372.

Koufi, V., Malamateniou, F., & Vassilacopoulos, G. (2009). Building Interoperable Health Information Systems Using Agent and Workflow Technologies. *Medical Informatics in a United and Healthy Europe*, (págs. 180-184).

Labrou, Y., Finin, T., & Peng, Y. (1999). Agent Communication Languages: the Current Landscape. *IEEE Intelligent Systems, 14*, 45-52.

Latin.org, P. (18 de Enero de 2014). *Inteligencia artificial.* Recuperado el 16 de Diciembre de 2015, de Inteligencia artificial: http://escritura.proyectolatin.org/inteligencia-artificial/sistemas-multiagentes/

Lengua, R. A. (2012). *Diccionario de la real Academia de la Lengua.* Barcelona: Publicaciones españolas.

Ljungberg, M., & Lucas, A. (1992). The Oasis Air-traffic Management System. *Proceedings of the 2nd Pacific Rim International Conference on Artificial Intelligence*, (págs. 1183–1189). Seoul, Republic of Korea.

Lyell, M., & Liu, X. (2012). Software agent application to support the patient-at-home. *2012 International Conference on Collaboration Technologies and Systems (CTS)*, (págs. 97-103).

Meo, P. D., Quattrone, G., & Ursino, D. (2011). Integration of the HL7 Standard in a Multiagent.

Merchan, Á., Plaza, J., & Moreno, J. (2011). Implementación de un módulo de búsqueda de personas dentro de una base de datos de rostros en un ambiente distribuido usando hadoop y los servicios web de amazon (aws).

Montero, P. (8 de Abril de 2015). *pmonterom*. Obtenido de https://pmonterom.wordpress.com/2015/04/08/big-data/

Moreno, A., & Nealon, J. (2003). *Applications of Software Agent Technology*. Whitestein.

Neagu, N., Dorer, K., Greenwood, D., & Calisti, M. (2006). LS/ATN: Reporting on a Successful Agent-Based Solution for Transport Logistics Optimization. *Proceedings of the IEEE 2006 Workshop on Distributed Intelligent Systems (WDIS'06).* Prague.

Nwana, H., Lee, L., & Jennings, N. (1996). Coordination in Software Agent Systems. *BT Technology Journal, 14*, 79-88.

Orza, B., Cordos, A., Vlaicu, A., & Meza, S. (2010). Integrated medical system using DICOM and HL7 standards.

Palazzo, L., & Paolo, S. C. (2013). A Multi-Agent Architecture for Health Information Systems. *Architecture for Health Information Systems*, 52.

Parunak, H. (1987). Manufacturing Experience with the Contract Net. En M. Huhns, *Distributed Artificial Intelligence* (págs. 285–310). London, UK: M Huhns.

PixelMed Publishing. (2000-2015). Obtenido de http://www.pixelmed.com/

PostgreSQL Global Development Group. (1996). Obtenido de http://www.postgresql.org/

Prashant, S. (2011). *High Performance Multi-agent System based Simulations.* India: Center for Data Engineering.

Proctor, M. (2010). Obtenido de https://docs.jboss.org/drools/release/5.4.0.CR1/droolsjbpm-introduction-docs/html_single/

Rao, A., & Georgeff, M. (1995). BDI Agents: from Theory to Practice. *Proceedings of the 1st International*, (págs. 312-319). San Francisco, CA.

Red Hat projects. (2016). Obtenido de http://hibernate.org/

Rolim, C. O., Koch, F. L., Westphall, C. B., Werne, J., Fracalossi, A., & Salvador, G. S. (2010). Cloud Computing Solution for Patient's Data Collection in Health Care Institutions. *2010Second International Conference on eHealth, Telemedicine, and Social Medicine*, (págs. 95-99).

Russell, & Norvig. (2008). *Artificial Intelligence A Modern Approach.*

Sánchez, R., & Castro, A.

Sano, Y., & Fukuta, N. (2013). A GPU-based Framework for Large-scale Multi-Agent Traffic Simulations. *Second IIAI International Conference on Advanced Applied Informatics*, 262-267.

SAS. (2015). *SAS*. Obtenido de http://www.sas.com/en_us/insights/big-data/what-is-big-data.html

Searle, J. (1969). *Speech Acts.* Cambridge, MA: Cambridge University Press.

Sethia, P. (2011). High Performance Multi-agent System based Simulations.

Smith, R., & Davis, R. (1980). The Contract Net protocol: High Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers, 29,* 1104–1113.

Sparx Systems. (2015). Obtenido de http://www.sparxsystems.com/about.html

Ternero, M. d. (2009 - 2010). Sistemas multiagentes en redes de datos. *Departamento de Tecnología Electrónica - Universidad de Sevilla*.

TurnkeyConsulting. (2014). *turnkey consulting*. Obtenido de http://turnkey-consulting.blogspot.com/2014/09/componentes-de-una-plataforma-big-data.html

Weiss, G. (1999). Multi- agent System: a modern approach to distributed artificial intelligence.

Wikipedia. (3 de Febrero de 2015). *Wikipedia - sistema multiagente.* Recuperado el 16 de Diciembre de 2015, de Wikipedia - sistema multiagente: https://es.wikipedia.org/wiki/Sistema_multi-agente

ANEXOS

Anexo 1. Glosario

ACR. American College of Radiology, creador de los estándares para imágenes médicas.

ADMISIÓN DE PACIENTES. Actividad de recolección de datos de un paciente que solicita atención en los servicios de salud.

AGENTE. Entidad que funciona continua y autónomamente, en un entorno en el cual otros procesos ocurren para lograr objetivos específicos según las ramas de formación en las que se quiere implementar.

ANSI. American National Standards Institute, empresa encargada de supervisar el progreso de estándares para servicios, procesos y sistemas en los Estados Unidos.

APACHE TOMCAT. Sirve como contenedor de Servlets y JavaServer Pages, desarrollado con JAVA.

API. Interfaz de Programación de Aplicaciones, es un conjunto de llamadas a bibliotecas que brindan servicios desde los procesos, tiene la capacidad de comunicar entre elementos de software.

ASCII. Código Estándar Estadounidense para el Intercambio de Información, es el estándar para códigos de caracteres y numéricos, fundamentados en el alfabeto latino.

AUTOMATIZAR. Es el conjunto de métodos que permiten mejorar, simplificar procesos, obteniendo ahorro de tiempo y dinero a través de los sistemas de información.

BEHAVIOUR. Es una funcionalidad que incorpora un agente, se encargan de especificar tareas que serían los servicios que hace un agente para obtener sus objetivos.

BIG DATA. Se refiere a la acumulación de grandes cantidades de datos sean estructurados o no estructurados.

CASOS DE USO. Descripción de los pasos que se realizan para llevar a cabo algún proceso, es la representación de la interacción desarrollada entre un sistema y sus actores.

DATA SET. Representa una instancia de un objeto de información en el mundo real.

DATANODE. Parte de la arquitectura de HDFS de Hadoop, se encargan de mantener los datos y se encarga de la replicación de los mismos a través de la red.

.DCM. Extensión del formato de las imágenes médicas DICOM, las cuales pueden ser radiografías, tomografías, entre otras.

DICOM. Digital Imaging and Comunication in Medicine, es un estándar para imágenes médicas, son usadas alrededor de todo el mundo, puede almacenar imágenes 2D, posiciones 3D.

EDI. Intercambio Electrónico de Datos, es usado para la transferencia de documentos electrónicos de un sistema computacional a otro.

FORMULARIO HCU-001. Utilizado para la admisión de un paciente.

FORMULARIO HCU-002. Utilizado para examen físico regional de un paciente.

GANGLIA. Es un sistema de control para los sistemas de computación de alto rendimiento como son los clusters, el cual permite al usuario obtener remotamente estadísticas en tiempo real como pueden ser del funcionamiento de su CPU.

HADOOP. Es un framework creado por Apache de código abierto para elaborar aplicaciones sobre grandes clusters, se basa en JAVA puede soportar aplicaciones distribuidas.

HDFS. Hadoop Distributed File System, parte de la arquitectura de Hadoop, es el sistema de archivos distribuido que fue planteado para ejecutarse a nivel de hardware, es aplicado cuando se obtienen grandes agrupaciones de datos

HIS. Sistema de Información en Salud, son programas de cómputo ubicados en un hospital.

HL7. Health Level Seven Internacional, es un estándar que se encarga de definir mensajes y protocolos de intercambio de mensajes para apoyar la práctica clínica que mejoran la prestación de atención.

JADE. Java Agent DEvelopment Framework, Es un middleware desarrollado de sistemas multiagente, aquí los agentes interactúan con cualquier otro mediante el uso de un lenguaje de comunicación especial llamado lenguaje de comunicación de agentes, es adaptable para cualquier tipo de agente.

JAVA. Es un lenguaje de programación y una plataforma informática.

JAVA EE. Java Enterprise Edition, es el conjunto de especificaciones que facilitan el desarrollo y despliegue de aplicaciones empresariales multi-capa.

JAVA SERVER FACE. Es un framework para aplicaciones java basadas en la web, facilita el desarrollo de interfaces de usuario en aplicaciones Java EE.

JBOSS. Es un servidor de aplicaciones Java EE.

JMETER. Es una herramienta utilizada para pruebas de rendimiento, como pruebas funcionales para analizar, tantear el desempeño de aplicaciones web.

JOBTRACKER. Forma parte de la arquitectura base MapReduce de Hadoop, se encarga de gestionar los recursos del clúster.

LOG. Es un registro de actividad de un sistema, se guarda en un archivo de texto, al cual se va aumentando líneas a medida que se realizan operaciones sobre el sistema.

MAPEAR. Describir las diferentes fases de un proceso, se incluyen información sobre los agentes que intervienen.

MAPREDUCE. Es el motor de hadoop, funciona en nodos que realizan las opciones de mapeo donde se recoge los datos, reduce que se encarga de procesar y generar resultados.

MYSQL. Es un sistema de gestión de base de datos relacional.

NAMENODE. Parte de la arquitectura de HDFS de Hadoop, mantiene la información (metadatos) de los ficheros que residen en el HDFS

NODEMANAGER. Forma parte de la arquitectura Yarn de hadoop, es el agente por equipo que se encarga de los contenedores, el seguimiento de su uso de los recursos (CPU, memoria, disco, red).

ONTOLOGÍA. Se usa para las aplicaciones desarrolladas en base al contexto de los sistemas multiagente.

PARALELIZAR. Proceso mediante el cual la computadora ejecuta las instrucciones de un programa dividiendo entre varios procesadores con el objetivo de ejecutar el programa en menos tiempo.

REQUERIMIENTOS FUNCIONALES. Se define como tareas observables o funciones que deben llevarse a cabo, sus alcances y limitaciones.

REQUERIMIENTOS NO FUNCIONALES. Define la interacción con el usuario, la apariencia, el mantenimiento del sistema y la operatividad.

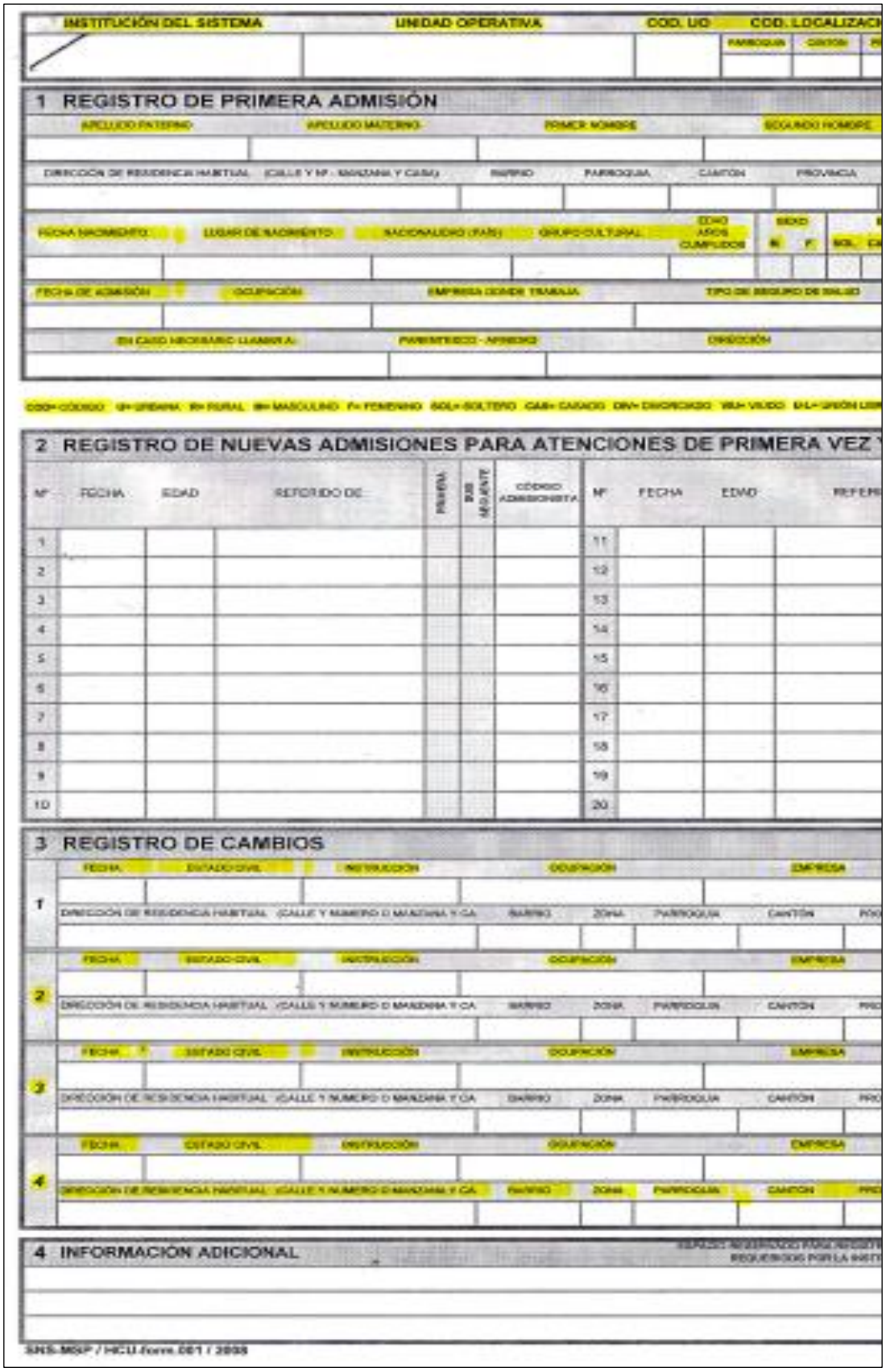
RESOURCEMANAGER. Forma parte de la arquitectura Yarn de hadoop, es la máxima autoridad que intercede los recursos entre todas las aplicaciones en el sistema.

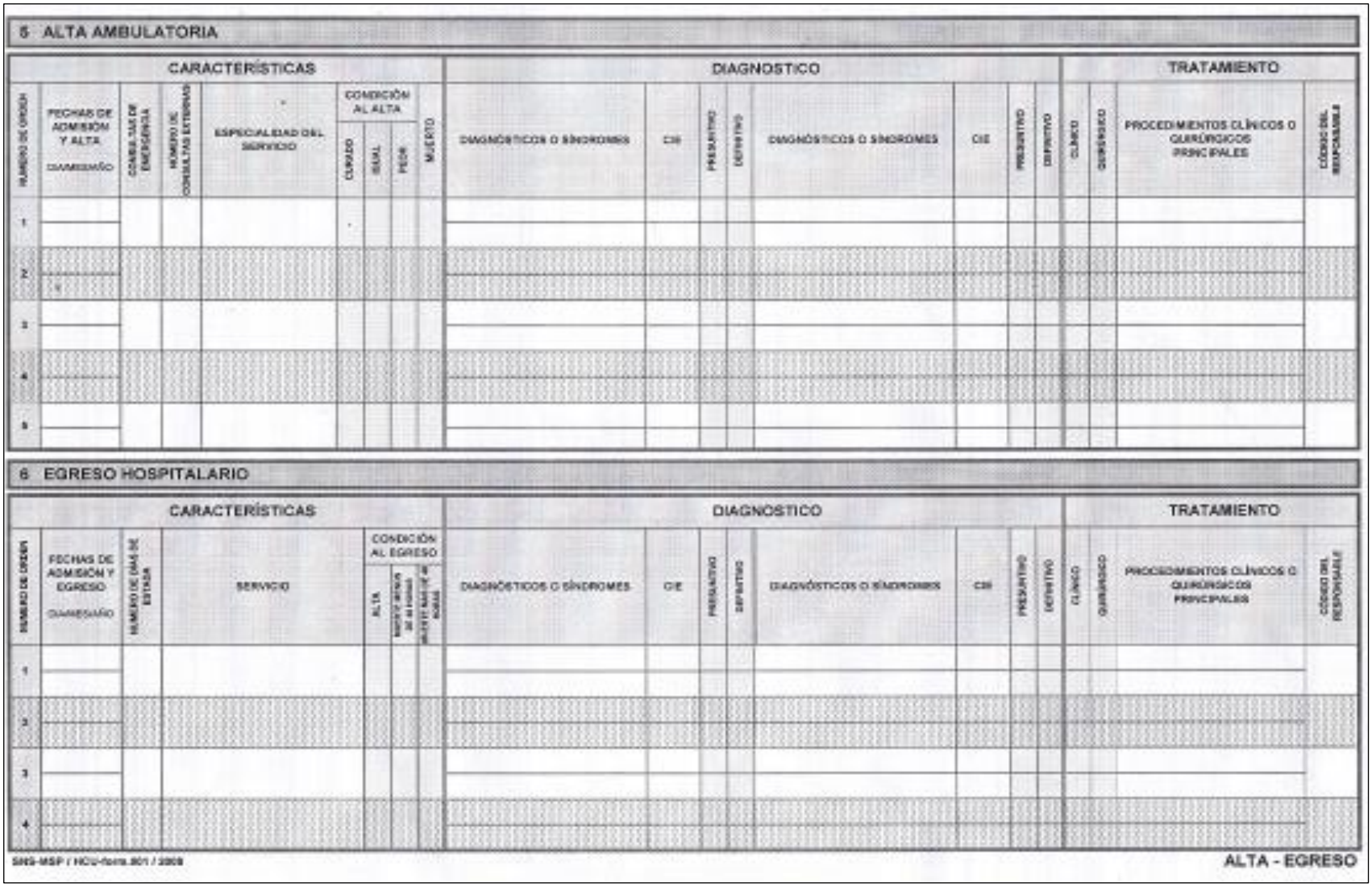
SMA. Sistema Multiagente, es una ciencia ya que se fundamenta en postulados epistemológicos tanto de la lógica formal de la teoría de sistemas, como del constructivismo como corriente que explica la realidad desde varios puntos de vista, también es una técnica ya que un sistema multiagente es un sistema distribuido entre nodos en este contexto.

WSIG. Web Services Integration Gateway, es una aplicación web cuyo objetivo es exponer los servicios proporcionados por los agentes JADE.

YELLOW PAGES. Permiten a los agentes publicar los servicios que proporcionan, para así otros agentes puedan acceder a ellos.

Anexo 2. Formulario





5 ALTA AMBULATORIA

CARACTERÍSTICAS

NÚMERO DE ORDEN	FECHAS DE ADMISIÓN Y ALTA DIA/MES/AÑO	CONSULTA DE EMERGENCIA	NÚMERO DE CONSULTAS EXTERNAS	ESPECIALIDAD DEL SERVICIO	CONDICIÓN AL ALTA: CURADO	CONDICIÓN AL ALTA: MEJORADO	CONDICIÓN AL ALTA: FUGA	MUERTO
1								
IV								
II								
III								
III								

DIAGNOSTICO

DIAGNÓSTICOS O SÍNDROMES	CIE	PRESUNTIVO	DEFINITIVO	DIAGNÓSTICOS O SÍNDROMES	CIE	PRESUNTIVO	DEFINITIVO

TRATAMIENTO

CLÍNICO	QUIRÚRGICO	PROCEDIMIENTOS CLÍNICOS O QUIRÚRGICOS PRINCIPALES	CÓDIGO DEL RESPONSABLE

6 EGRESO HOSPITALARIO

CARACTERÍSTICAS

NÚMERO DE ORDEN	FECHAS DE ADMISIÓN Y EGRESO DIA/MES/AÑO	NÚMERO DE DÍAS DE ESTADA	SERVICIO	CONDICIÓN AL EGRESO: ALTA	CONDICIÓN AL EGRESO: MEJORADO	CONDICIÓN AL EGRESO: FUGA	CONDICIÓN AL EGRESO: MUERTE
1							
II							
3							
4							

DIAGNOSTICO

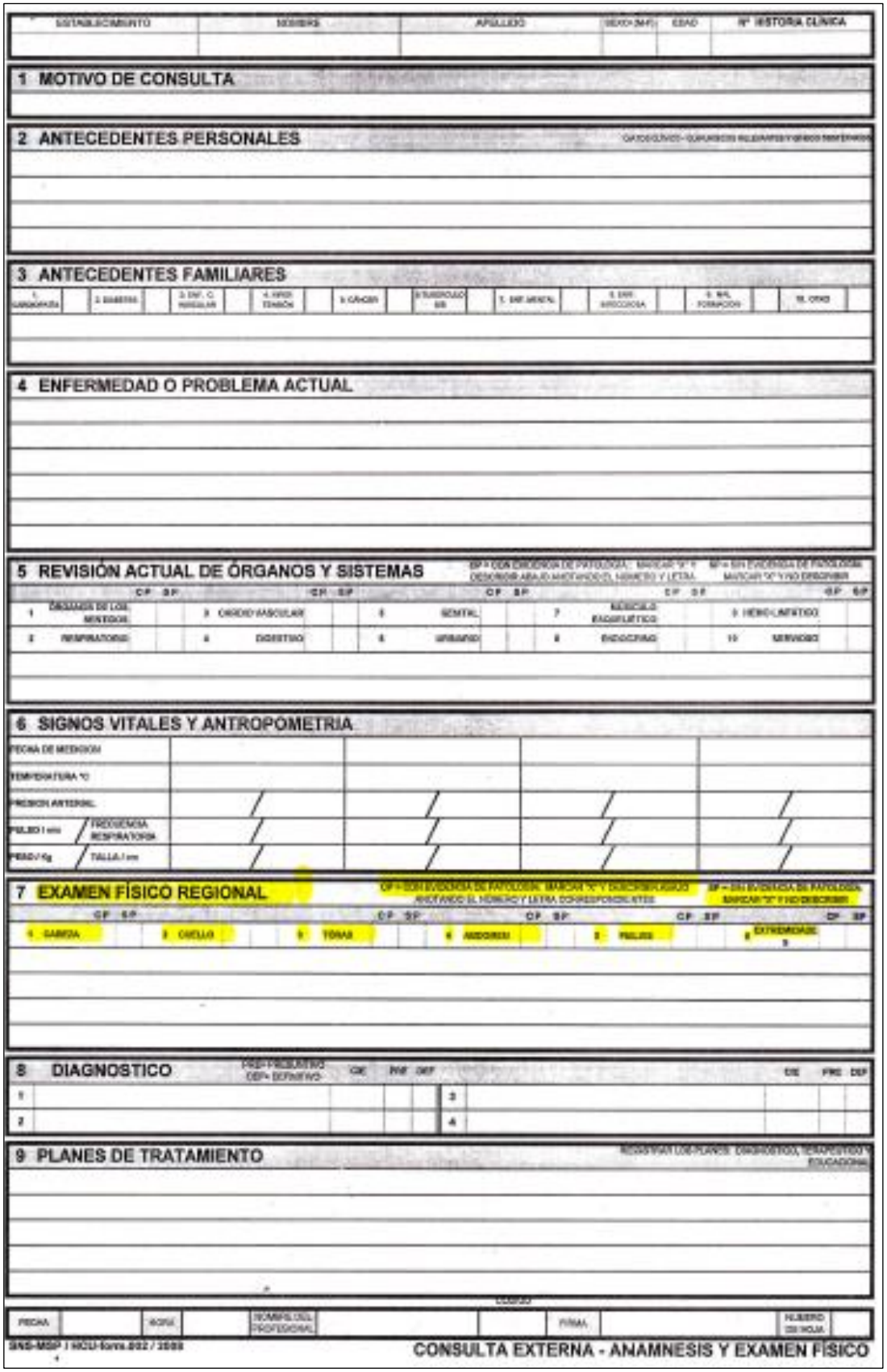
DIAGNÓSTICOS O SÍNDROMES	CIE	PRESUNTIVO	DEFINITIVO	DIAGNÓSTICOS O SÍNDROMES	CIE	PRESUNTIVO	DEFINITIVO

TRATAMIENTO

CLÍNICO	QUIRÚRGICO	PROCEDIMIENTOS CLÍNICOS O QUIRÚRGICOS PRINCIPALES	CÓDIGO DEL RESPONSABLE

SNS-MSP / HCU-form.001 / 2008

ALTA - EGRESO



ESTABLECIMIENTO	NOMBRES	APELLIDO	SEXO (M-F)	EDAD	Nº HISTORIA CLÍNICA

1 MOTIVO DE CONSULTA

2 ANTECEDENTES PERSONALES

3 ANTECEDENTES FAMILIARES

1. GARGANTA		2. DIABETES		3. ENF. C. VASCULAR		4. HTAS TENSIÓN		5. CÁNCER		6. TUBERCULOSIS		7. ENF. MENTAL		8. ENF. ARTICULAR		9. MAL FORMACIÓN		10. OTRO	

4 ENFERMEDAD O PROBLEMA ACTUAL

5 REVISIÓN ACTUAL DE ÓRGANOS Y SISTEMAS

CP = CON EVIDENCIA DE PATOLOGÍA: MARCAR "P" Y DESCRIBIR ABAJO ANOTANDO EL NÚMERO Y LETRA. SP = SIN EVIDENCIA DE PATOLOGÍA: MARCAR "S" Y NO DESCRIBIR

	CP	SP		CP	SP		CP	SP		CP	SP		CP	SP
1 ÓRGANOS DE LOS SENTIDOS			3 OSEO-MUSCULAR			5 GENITAL			7 NERVIOSO PSIQUIÁTRICO			9 HEMO-LINFÁTICO		
2 RESPIRATORIO			4 DIGESTIVO			6 URINARIO			8 ENDOCRINO			10 NERVIOSO		

6 SIGNOS VITALES Y ANTROPOMETRÍA

FECHA DE MEDICIÓN				
TEMPERATURA °C				
PRESIÓN ARTERIAL	/	/	/	/
PULSO / min / FRECUENCIA RESPIRATORIA	/	/	/	/
PESO / Kg / TALLA / cm	/	/	/	/

7 EXAMEN FÍSICO REGIONAL

CP = CON EVIDENCIA DE PATOLOGÍA: MARCAR "P" Y DESCRIBIR ABAJO ANOTANDO EL NÚMERO Y LETRA CORRESPONDIENTE. SP = SIN EVIDENCIA DE PATOLOGÍA: MARCAR "S" Y NO DESCRIBIR

	CP	SP		CP	SP		CP	SP		CP	SP		CP	SP
1 CABEZA			3 CUELLO			5 TÓRAX			6 ABDOMEN			8 PIELES		
													9 EXTREMIDADES	

8 DIAGNOSTICO

PRE= PRESUNTIVO DEF= DEFINITIVO

	CIE	PRE	DEF		CIE	PRE	DEF
1				3			
2				4			

9 PLANES DE TRATAMIENTO

REGISTRAR LOS PLANES: DIAGNÓSTICOS, TERAPÉUTICOS Y EDUCACIONALES

FECHA		HORA		NOMBRES DEL PROFESIONAL			FIRMA		NÚMERO DE HOJA	

SNS-MSP / HCU-form.002 / 2008

CONSULTA EXTERNA - ANAMNESIS Y EXAMEN FÍSICO

10 EVOLUCIÓN

FIRMAR AL PIE DE CADA NOTA

FECHA (DIA/MES/AÑO)	HORA	NOTAS DE EVOLUCIÓN

11 PRESCRIPCIONES

FIRMAR AL PIE DE CADA PRESCRIPCIÓN

REGISTRAR EN ROJO LA ADMINISTRACIÓN DE FÁRMACOS Y OTROS PRODUCTOS (ENFERMERÍA)

FARMACOTERAPIA E INDICACIONES (PARA ENFERMERÍA Y OTRO PERSONAL)	ADMINISTR. FÁRMACOS Y OTROS

SNS-MSP / HCU-form.002 / 2008

CONSULTA EXTERNA - EVOLUCIÓN Y PRESCRIPCIONES

ESTABLECIMIENTO	NOMBRE	APELLIDO	SEXO (M-F)	Nº HOJA	Nº HISTORIA CLÍNICA

REGISTRAR EN ROJO LA ADMINISTRACIÓN DE FÁRMACOS Y OTROS PRODUCTOS (ENFERMERÍA)

1 EVOLUCIÓN

FECHA (DÍA / MES / AÑO)	HORA	NOTAS DE EVOLUCIÓN

2 PRESCRIPCIONES FIRMAR AL PIE DE CADA PRESCRIPCIÓN

FARMACOTERAPIA E INDICACIONES (PARA ENFERMERÍA Y OTRO PERSONAL)	ADMINISTR. FÁRMACOS INSUMOS

SNS-MSP / HCU-form.005 / 2008

EVOLUCIÓN Y PRESCRIPCIONES (1)

ESTABLECIMIENTO	NOMBRE	APELLIDO	SEXO (M-F)	Nº HOJA	Nº HISTORIA CLÍNICA

1 EVOLUCIÓN

FECHA (DÍA / MES / AÑO)	HORA	NOTAS DE EVOLUCIÓN

2 PRESCRIPCIONES

FARMACOTERAPIA E INDICACIONES	

ESTABLECIMIENTO	NOMBRE	APELLIDO	SEXO (M-F)	EDAD	Nº HISTORIA CLÍNICA

MENOR DE 1 AÑO		1 - 4 AÑOS		5 - 9 AÑOS PROGRAMADO		5-14 AÑOS NO PROGRAMADO		10-14 AÑOS PROGRAMADO		15-19 AÑOS		MAYOR DE 20 AÑOS		EMBARAZADA	

1 MOTIVO DE CONSULTA

2 ENFERMEDAD O PROBLEMA ACTUAL

3 ANTECEDENTES PERSONALES Y FAMILIARES

1. ALERGIA ANTIBIÓTICO		2. ALERGIA ANESTESIA		3. HEMO RRAGIAS		4. VIH/SIDA		5. TUBER CULOSIS		6. ASMA		7. DIABETES		8. HIPER TENSIÓN		9. ENF. CARDIACA		10. OTRO	

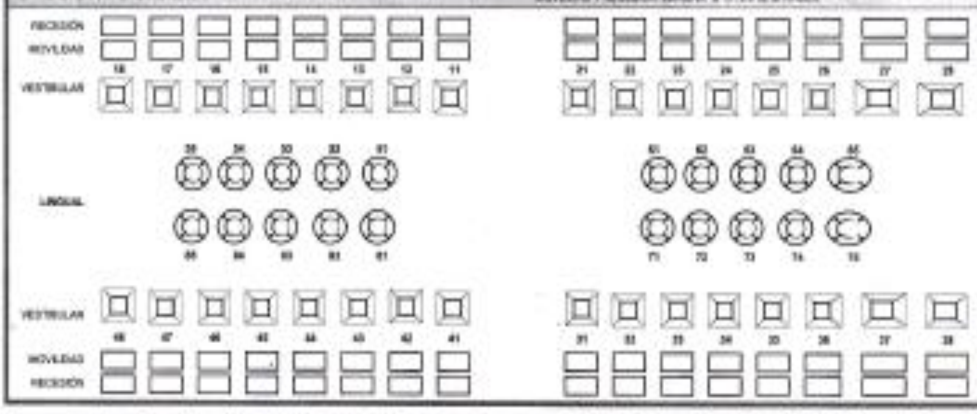
4 SIGNOS VITALES

PRESIÓN ARTERIAL		FRECUENCIA CARDIACA min		TEMPERATURA A °C		F. RESPIRAT. min	

5 EXAMEN DEL SISTEMA ESTOMATOGNÁTICO

1. LABIOS		2. MEJILLAS		3. MAXILAR SUPERIOR		4. MAXILAR INFERIOR		5. LENGUA		6. PALADAR		7. PISO		8. CARRILLOS	
9. GLÁNDULAS SALIVALES		10. ORO FARINGE		11. A. T. M.		12. GANGLIOS									

6 ODONTOGRAMA



7 INDICADORES DE SALUD BUCAL

HIGIENE ORAL SIMPLIFICADA									ENFERMEDAD PERIODONTAL		MAL OCLUSIÓN		FLUOROSIS	
PIEZAS DENTALES						PLACA 0-1-2-3	CÁLCULO 0-1-2-3	GINGIVITIS 0-1	LEVE		ANGLE I		LEVE	
16		17		55					MODERADA		ANGLE II		MODERADA	
11		21		51					SEVERA		ANGLE III		SEVERA	
26		27		65										
36		37		75										
31		41		71										
46		47		85										
TOTALES														

8 ÍNDICES CPO-ceo

	C	P	O	TOTAL
D				
	c	e	o	TOTAL
d				

9 SIMBOLOGÍA DEL ODONTOGRAMA

✱ rojo	SELLANTE NECESARIO	⊗	PÉRDIDA (OTRA CAUSA)	▭	PRÓTESIS TOTAL
✱ azul	SELLANTE REALIZADO	△	ENDODONCIA	⊡	CORONA
X rojo	EXTRACCIÓN INDICADA	□—□	PRÓTESIS FIJA	O azul	OBTURADO
X azul	PÉRDIDA POR CARIES	(—)	PRÓTESIS REMOVIBLE	O rojo	CARIES

SNS-MSP / HCU-form.033/ 2008

ODONTOLOGÍA (1)

10 PLANES DE DIAGNÓSTICO, TERAPÉUTICO Y EDUCACIONAL

ODONTOLOGÍA		QUÍMICA SANGUÍNEA		RAYOS - X		OTROS		

11 DIAGNÓSTICO

	PRE= PRESUNTIVO DEF= DEFINITIVO		CIE	PRE	DEF			CIE	PRE	DEF
1						3				
2						4				

CÓDIGO

FECHA DE APERTURA		FECHA DE CONTROL		PROFESIONAL		FIRMA		NÚMERO DE HOJA	

12 TRATAMIENTO

SESIÓN Y FECHA	DIAGNOSTICOS Y COMPLICACIONES	PROCEDIMIENTOS	PRESCRIPCIONES	CÓDIGO Y FIRMA
SESIÓN 1				CÓDIGO
FECHA				FIRMA
SESIÓN 2				CÓDIGO
FECHA				FIRMA
SESIÓN 3				CÓDIGO
FECHA				FIRMA
SESIÓN 4				CÓDIGO
FECHA				FIRMA
SESIÓN 5				CÓDIGO
FECHA				FIRMA
SESIÓN 6				CÓDIGO
FECHA				FIRMA
SESIÓN 7				CÓDIGO
FECHA				FIRMA
SESIÓN 8				CÓDIGO
FECHA				FIRMA
SESIÓN 9				CÓDIGO
FECHA				FIRMA