# UNIVERSIDAD POLITÉCNICA SALESIANA
# SEDE QUITO

**CARRERA:**
**INGENIERÍA DE SISTEMAS**

**Trabajo de titulación previo a la obtención del título de:**
**Ingeniero de Sistemas**

**TEMA:**
**CONSTRUCCIÓN DE UNA DISTRIBUCIÓN MCLOUD PROTOTIPO PARA AGENTES MÓVILES OPTIMIZANDO RECURSOS**

**AUTOR:**
**CARLOS ANDRÉS HERRERA BALSECA**

**TUTOR:**
**WASHINGTON ARSENIO RAMÍREZ MONTALVAN**

**Quito, septiembre de 2020**

**CESIÓN DE DERECHOS DE AUTOR**

Yo Carlos Andrés Herrera Balseca, con documento de identificación N° 1726927831, manifiesto mi voluntad y cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del trabajo de grado/titulación intitulado: "CONSTRUCCIÓN DE UNA DISTRIBUCIÓN MCLOUD PROTOTIPO PARA AGENTES MÓVILES OPTIMIZANDO RECURSOS", mismo que ha sido desarrollado para optar por el título de: Ingeniero 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 mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia, suscribo este documento en el momento que hago entrega del trabajo final en digital a la Biblioteca de la Universidad Politécnica Salesiana.

.....

Herrera Balseca Carlos Andrés
CI: 1726927831
Quito, septiembre de 2020

**DECLARATORIA DE COAUTORÍA DEL DOCENTE TUTOR**

Yo declaro que bajo mi dirección y asesoría fue desarrollado el proyecto de titulación: CONSTRUCCIÓN DE UNA DISTRIBUCIÓN MCLOUD PROTOTIPO PARA AGENTES MÓVILES OPTIMIZANDO RECURSOS realizado por Carlos Andrés Herrera Balseca, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana, para ser considerados como trabajo final de titulación.

Quito, septiembre de 2020

.....

Washington Arsenio Ramírez Montalvan
CI: 1710804681

## DEDICATORIA

Dedico este trabajo a mi familia que tanto amo, Patricia, Segundo y mi hermano Santiago que me dieron con mucho esfuerzo y cariño todo el apoyo necesario para poder iniciar y así encaminar mi carrera universitaria, además de siempre estar al pendiente para ayudarme, a mi novia Karina que ha llegado a ser parte muy especial de mi vida y me ha dado todo su cariño y comprensión; finalmente a las personas que supieron compartir conmigo su experiencia y dieron paso a mi vida profesional en la rama de Auditoria Informática, más que mis compañeros de trabajo se convirtieron en grandes amigos. Agradecido por la oportunidad y por abrirme grandes puertas.

Carlos Andrés Herrera.

## AGRADECIMIENTO

Agradecido especialmente por todo el aprendizaje adquirido en la Universidad Politécnica Salesiana. A mi tutor Ing. Washington Ramírez por darme la mano y apoyarme en el desarrollo del proyecto y a todos los docentes que han sido parte de mi formación académica.

# ÍNDICE

# ÍNDICE DE TABLAS

## ÍNDICE DE FIGURAS

**Resumen**

El uso de dispositivos tecnológicos en el día a día de las personas se ha vuelto algo fundamental e incluso se han convertido en herramientas de trabajo, día a día y con el estilo de vida de las personas estos dispositivos se han tenido que adaptar y volverse portables para la mejor experiencia del usuario, así tuvimos el paso de computadores de escritorio a computadores portátiles, y de estos a dispositivos móviles tales como smartphones, tablets, etc.

La necesidad de ejecutar aplicativos en dispositivos móviles como se lo realiza en computadores llevó a los fabricantes de dispositivos móviles a mejorar e incluir altas características en sus nuevos modelos de smartphones y tablets, enfocándose en las mejoras de pantalla, batería, procesador y memoria. Esto con la finalidad de brindar una mejor experiencia al usuario final o consumidor.

Con la mejora de las características de estos nuevos dispositivos móviles se ha generado la obsolescencia de otros que por sus características no poseen la capacidad o rendimiento para ejecutar ciertas tareas o aplicativos, o en su defecto ejecutarlas de forma más lenta o con presencia de errores.

El presente proyecto tiene como finalidad construir una distribución de tal manera que dispositivos con bajas prestaciones tengan la posibilidad de presentar resultados de la ejecución con alta carga de procesamiento al usuario final.

**Abstract**

The use of technological devices in people's day to day life has become something fundamental and they have even become work tools, day by day and with people's lifestyle these devices have had to adapt and become portable. For the best user experience, we had the transition from desktop computers to laptops, and from these to mobile devices such as smartphones, tablets, etc.

The need to run applications on mobile devices as it is done on computers led mobile device manufacturers to improve and include high features in their new models of smartphones and tablets, focusing on screen, battery, processor and memory improvements. This in order to provide a better experience to the end user or consumer.

With the improvement of the characteristics of these new mobile devices, the obsolescence of others has been generated which, due to their characteristics, do not have the capacity or performance to execute certain tasks or applications, or failing to execute them more slowly or with the presence of errors.

The purpose of this project is to build a distribution in such a way that devices with low performance have the possibility of presenting results of execution with high processing load to the end user.

# Introducción

## Antecedentes

Desde la aparición del primer teléfono inteligente fabricado por IBM en el año de 1992, el mismo que fue considerado uno de los avances tecnológicos más prometedores puesto que además de realizar funciones básicas de un teléfono móvil como las de enviar un SMS, comunicarse con otras personas, etc., este dispositivo, el Simon Smartphone, era capaz de ejecutar actividades que se podían realizar en la computadora, incluyendo también una pantalla táctil para la mejor experiencia del usuario. El uso de dispositivos inteligentes actualmente ha revolucionado la forma de realizar actividades cotidianas como la comunicación, los negocios, la salud e incluso las áreas de ocio (Khazer & Rather, 2019).

En la actualidad los dispositivos móviles han evolucionado de manera increíble, convirtiéndose en plataformas informáticas sofisticadas. Características como la velocidad de su procesador, el tamaño de almacenamiento interno de datos y sus expansiones disponibles, las mejoras en los sistemas operativos con los que funcionan y la disponibilidad de banda ancha, han vuelto a los smartphones en dispositivos perfectos para la ejecución de tareas o aplicativos complejos (Shaukat, Shah, Saeed, & Azhar, 2015).

## Problema

Sin embargo y con el avance tecnológico tanto en hardware y software en dispositivos móviles, se genera la obsolescencia de dispositivos de generaciones anteriores, los mismos que por sus características no son capaces de ejecutar o tener un óptimo desempeño para la satisfacción de los usuarios, razón por la cual la industria encargada del ensamblaje y distribución de dispositivos móviles con más demanda a nivel global se han visto en la necesidad de exigir a sus fabricantes componentes con mayores capacidades en temas de hardware (memoria, procesador, batería,

pantalla, etc), con el fin de soportar la nueva tendencia de aplicativos y dar una mejor respuesta a quienes los utilizan en sus dispositivos (Qiwei & Cho, 2016).

**Justificación**

Este proyecto se encuentra enfocado en la construcción de una distribución tal que permita la ejecución de procesos con alta demanda de recursos en dispositivos móviles, generando optimización en el consumo de los mismos delegando las tareas con gran demanda de procesamiento a un dispositivo con mayores características o prestaciones.

**Objetivo General**

Analizar, desarrollar y construir una aplicación basada en agentes, para optimizar recursos computacionales de dispositivos móviles utilizando técnicas de *code offloading* para dispositivos Android.

**Objetivos específicos**

Desarrollar agentes móviles que permitan la ejecución remota de procesos.

Construir una distribución prototipo optimizada para dispositivos móviles.

Comprobar el nivel de ahorro de recursos utilizando métricas para evidenciar la optimización de recursos y tiempos de ejecución.

# Capítulo 1

# Marco Teórico

En el presente capítulo se dará a conocer conceptos y términos utilizados en la construcción de la distribución mCloud que serán necesarios para entender y comprender el proyecto propuesto.

## 1.1 Dispositivo móvil

Un dispositivo móvil se define como un artefacto electrónico que posee características limitadas, así como su memoria, el almacenamiento y la batería, que es utilizado para tareas determinadas como por ejemplo teléfonos celulares y tablets (Alonso, Ferreira, Álvarez, & García, 2011).

Para nuestro propósito nos enfocaremos en los teléfonos celulares que son dispositivos inalámbricos electrónicos basados en la tecnología de ondas de radio, que tiene la misma funcionalidad que cualquier teléfono de línea fija. Su principal característica es su movilidad, ya que la realización de llamadas no es dependiente de ningún terminal fijo y no requiere ningún tipo de cableado para llevar a cabo la conexión a la red telefónica. Aunque su principal función es la comunicación de voz, como el teléfono convencional, su rápido desarrollo ha incorporado funciones adicionales como mensajería instantánea (SMS), agenda, juegos, cámara fotográfica, agenda, acceso a Internet, reproducción de video e incluso GPS y reproductor mp3 (Alonso, Ferreira, Álvarez, & García, 2011).

**1.1.1 Tablets.** Considera como un computador portátil para uso personal, una tablet es un dispositivo tecnológico que pone a disposición del usuario capacidades tales como el uso herramientas de nominadas aplicativos tales como redes sociales, correo electrónico, geolocalización, etc. Una de sus características es la de sus capacidades en hardware que, a pesar de tener mejores prestaciones que un smartphone, no llega a ser comparable con las características de una computador portátil, sim embargo su portabilidad y la facilidad de contener información para poder ser visualizada en cualquier lugar lo que ha llevado a este dispositivo a tener gran acogida en el mercado de la computación móvil (Bedi, 2014).

**1.1.2 Smartphones.** Un smartphone, también conocido como teléfono inteligente es un término con el que se hace referencia a un teléfono celular que es capaz de tener mayor cantidad de funcionalidades que un teléfono celular común. Entre las principales características de un smartphone se encuentra la de permitir la implementación de programas que serán utilizados con el fin de satisfacer una necesidad del usuario, estos programas se conocen como aplicaciones. (De Lucas, 2013)

Una de las similitudes que comparten los dispositivos anteriormente mencionados, es la posibilidad de contener un sistema operativo que ayuda al usuario y brinda una interfaz más amigable para su interacción, a nivel global el sistema operativo para dispositivos móviles con mayor número de usuario es Android, según el trabajo realizado por Ahvanooey, Li, Rabbani, & Raza (2017) donde se puede ver el porcentaje de usuarios por cada sistema operativo móvil, en el periodo desde 2011 hasta 2017, el detalle a continuación en la figura 1.

Porcentaje de usuarios por Sistema Operativo

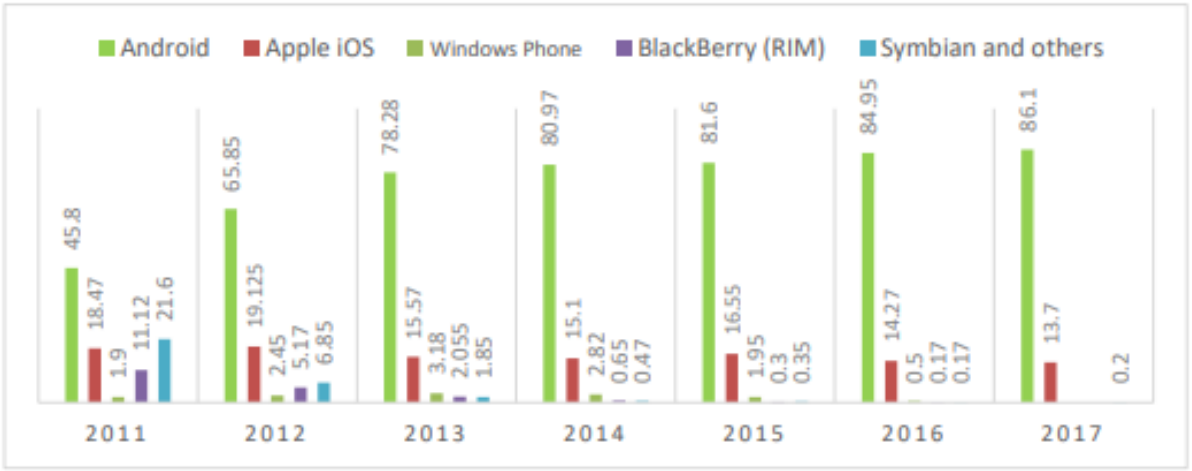



Figura 1. Porcentaje de usuarios por Sistema Operativo
Fuente: (Ahvanooey, Li, Rabbani, & Raza, 2017).

**1.1.3 Android.** Es un sistema operativo y una plataforma software, desarrollada por Android Inc. y que fue adquirida por Google en 2005, sin embargo, este sistema operativo se dio a conocer en el mercado tecnológico 3 años más tarde con la iniciativa de incentivar el uso de software libre. Esta plataforma se encuentra basada en Linux para teléfonos móviles. Este sistema operativo es utilizado por gran cantidad de dispositivos móviles, entre ellos las tablets, incluso computadores con pocos recursos o destinados para la ejecución de aplicativos propios de Android. Los diferentes aplicativos con compatibilidad para Android, pueden ser desarrollados en un entorno de Java. Una de sus características es estar sobre código libre lo que brinda a los programadores la facilidad de crear sus propios aplicativos y distribuirlos entre los usuarios de este sistema operativo. (Báez, y otros, 2012)**.**

## 1.2 Aplicaciones móviles

Las aplicaciones móviles son considerados programas adicionales para la ejecución en dispositivos portátiles o móviles tales como teléfonos celulares y tablets; estos aplicativos dan acceso al usuario a funcionalidades que no se encuentran disponibles en la versión de fabrica del dispositivo, sin

embargo pueden ser añadidas para tener acceso a distintos servicios como redes sociales, juegos, servicios de geolocalización, etc (Adolph, 2009).

**1.2.1 Tipos de aplicaciones móviles.** La expansión y sobre todo la evolución de dispositivos móviles en el mercado, además considerando que estos se han vuelto parte de la vida diaria de las personas, ha generado la necesidad de la creación de aplicaciones que puedan satisfacer una necesidad del usuario, estas aplicaciones han ido tomando mayor complejidad conforme incrementan las capacidades de los dispositivos en los que se ejecutan, de esta manera tenemos tres enfoques diferentes para el despliegue de un aplicativo móvil:

- **Aplicaciones Web:** son aplicativos que de forma similar a las páginas web tradicionales son desarrolladas mediante HTML, CSS y JavaScript. La característica más sobresaliente de este tipo de aplicaciones es la no necesidad de instalar componentes de terceros, su despliegue se lo realiza mediante el navegador nativo del dispositivo, una de sus desventajas es la dependencia de la conexión a una red.
- **Aplicaciones nativas:** la principal característica de estas aplicaciones es que son diseñadas para un sistema operativo específico, pudiendo ser iOS o Android, además de su semejanza a las tradicionales aplicaciones de escritorio para PCs, estas aplicaciones deben ser compiladas para obtener ejecutables que serán distribuidas mediante tiendas virtuales, tales como Play Store de Google o la App Store de Apple. Otra característica destacada de este tipo de aplicaciones la posibilidad de usar el hardware del dispositivo como la cámara, almacenamiento interno o externo, sensores, GPS, acelerómetro, etc.
- **Aplicaciones hibridas:** son aplicaciones que manejan las mejores características tanto de aplicaciones web como de aplicaciones nativas, es decir, son desarrolladas con lenguaje web, pero su interface se adapta a la vista de cualquier dispositivo. Su principal ventaja es

su adaptabilidad sin la necesidad de estar en un sistema operativo específico (Delía, Galdamez, Thomas, & Pesado, 2013).

## 1.3 Procesamiento de imágenes digitales

Existen varios programas para la edición y retoque de imágenes, esto con el fin de cubrir las necesidades de los usuarios de poder manipular las características a nivel de pixeles dentro de una fotografía o imagen definida, para esto existen varias técnicas con su respectivo resultado, por ejemplo, la segmentación, normalización y binarización (Magro, 2013). Esta última considerada para la ejecución del presente proyecto debido al uso de tareas repetitivas e intensivas que conlleva esta técnica, convirtiéndola un método con las características necesarias para probar la eficiencia de la distribución con agentes propuesta.

**1.3.1 Binarización de imágenes.** La binarización es un método en el que una imagen es sometida a un intercambio de sus valores numéricos originales (que se encuentran entre el rango de 0 a 255) a ceros (color negro) y 255 (color blanco) a nivel de su matriz digital donde se encuentran los valores correspondientes a los colores de los pixeles que la conforman, esta transformación de valores numéricos es efectuada en base a un umbral definido, de esta manera si el valor del pixel es mayor o menor a la del umbral se definirá el nuevo color, sea este blanco o negro (Magro, 2013).

$$P_{(x,y)} = \begin{cases} 0, & \left(\frac{Q_{red(x,y)} + Q_{green(x,y)} + Q_{blue(x,y)}}{3}\right) < umbral \\ 255, & \left(\frac{Q_{red(x,y)} + Q_{green(x,y)} + Q_{blue(x,y)}}{3}\right) \geq umbral \end{cases}$$

Donde:

$P_{(x,y)}$: nuevo valor del pixel

$Q_{abc(x,y)}$: valor del pixel actual en RGB

$(x, y)$: número de fila y columna en la matriz del mapa de bits de la imagen

Para la definición del umbral, Magro (2013) propone varias técnicas para esta actividad, entre ellos tenemos:

- **Histograma:** se basa en la definición mediante la consideración de los valores máximos y mínimos del histograma de la imagen.
- **Entropía:** método basado en el análisis de regiones frontales y su fondo.
- **Similitud:** basado en la concordancia entre tonos grises, bordes de imagen, etc.
- **Global:** establece un valor fijo como umbral para todos los pixeles que conforman la imagen.
- **Local:** asigna un valor de umbral a cada pixel según su segmentación.

La técnica de umbralización global simple es aplicable cuando se tienen como entrada imágenes definidas, esta característica relacionada con las imágenes digitales, sin embargo, para la elección del valor de umbral se debe considerar los puntos máximos y mínimos que posee la imagen.

> Así pues, si una determinada imagen tiene un valor máximo de blanco 200 y un mínimo de negro de 40 y aplicamos un valor medio de 1-255 (127), el resultado es muy diferente de si, por ejemplo, partimos de la misma imagen con un valor máximo de blanco de 255 y uno mínimo de negro de 0, es decir con mayor contraste y aplicamos el mismo umbral (Magro, 2013, pág. 445).

### 1.4 Mobile Cloud Computing

Para comprender de mejor manera lo que implica la computación en la nube móvil, se debe entender primero que la computación móvil se dedica a estudiar la aceptación de los dispositivos portátiles a los diferentes aplicativos incluyendo la característica de la movilidad, es decir la utilización de aplicativos en dispositivos móviles de uso cotidiano como smartphones, tablets, etc. Por otro lado, la computación en la nube se concentra principalmente en la administración de

recursos que pueden estar en un ambiente virtual y es compartido por varios usuarios, también conocidos como servidores o computadores remotos (Huang, 2011).

Partiendo de las dos definiciones anteriores que en la actualidad son utilizadas comúnmente, se puede definir a la computación en la nube móvil como servidores o servicios que se encuentran alojados en computadores remotos o dentro de una red local, que son utilizados por los dispositivos móviles para poder ejecutar tareas con altas demandas de procesamiento, que posiblemente en ejecuciones locales consumirían por completo su capacidad (Qi & Gani, 2012).

Niroshinie, Seng, & Wenny (2012) en su investigación menciona la existencia de aplicativos que pueden llegar a tener una gran demanda de procesamiento de información como juegos, procesamiento de imágenes, realidad aumentada, entre otros, que pueden no ser apropiados para su uso en dispositivos de características limitadas como los smartphones y es cuando la computación en la nube móvil gana una gran ventaja, al poder direccionar la carga de trabajo más fuerte hacia computadores con mayores prestaciones, esta técnica es conocida como *code offloading*.

### 1.5 Code offloading

Con la inclusión de la computación en la nube móvil, se deriva la utilización de técnicas específicas para optimizar y mejorar el desempeño de aplicativos en dispositivos móviles con capacidades limitadas, especialmente aplicativos o tareas en las que interviene el procesamiento intensivo. Una de estas técnicas es el *code offloading* que permite el envío de tareas intensivas a dispositivos, servidores o computadores que posean mayores prestaciones, con la finalidad de mejorar los resultados en cuanto a tiempos de respuesta, bajo uso de recursos locales y eficiencia en el procesado de información. Esta mejora en la experiencia del uso de estos aplicativos da al usuario una perspectiva de dispositivos móviles con mayor capacidad en sus recursos locales (Oliveira, da Silva, Machry, & Coelho, 2017).

## Agentes

A pesar de que no existe una definición específica, se denomina Agente al software desarrollado que se ejecuta dentro de un entorno para un fin específico y que tiene la característica de ser autónomo ya que trabaja sin la necesidad del manejo directo de un ser humano ya que tiene control de sus acciones, es proactivo porque tiene la capacidad de manejar varios comportamientos con el fin de llegar a su objetivo, inclusive un agente puede llegar a tener movilidad y así poder viajar a diferentes nodos de la red con la posibilidad de comunicarse con otros agentes existentes en el mismo ambiente y extraer información según sus necesidades. Este ambiente se denomina Sistemas Multi Agentes o MAS por sus siglas en ingles (Bellifemine, Caire, & Greenwood, 2007).

### 1.5.1 Tipos de agentes.

Entre los principales tipos de agentes se puede mencionar:

- **Agentes colaborativos:** estos agentes se encuentran diseñados para brindar soluciones a problemas de gran envergadura y que por su naturaleza se encuentran distribuidos. A pesar de que pueden aprender, esta característica no los define y tampoco sobresale en ellos.
- **Agentes de interfaz:** son agentes especializados para brindar ayuda a su usuario, su característica principal es el aprendizaje y la autonomía, estos usuarios pueden aprender mediante las acciones que realiza o decisiones que toma el usuario, además de tener a posibilidad de aprender de otros agentes.
- **Agentes de información:** son agentes desarrollados con el objetivo de consolidar, ordenar y en general administrar grandes cantidades de información de forma automática, un ejemplo claro son los motores de búsqueda en Internet.
- **Agentes reactivos: s**on agentes que responden ante una petición o evento generado dentro de su entorno, por esta razón carecen de autonomía y son utilizados como agentes estáticos en la simulación de escenarios artificiales como por ejemplo fenómenos naturales.

- **Agentes híbridos:** son agentes que poseen más de una característica antes mencionada, normalmente toman las mejores propiedades y se caracterizan por su rápida respuesta ante eventos en su entorno.
- **Agentes heterogéneos:** son agentes que se encuentran normalmente en sistemas robustos donde cada uno posee tareas o servicios específicos y se intercomunican para poder solventar y satisfacer problemas complejos pudiendo o no tener agentes híbridos en su entorno (Avancini, 2000).

**1.5.2 Sistemas Multi Agentes (MAS).** Son estructuras lógicas que se conforman por un sinnúmero de componentes informáticos conocidos como agentes y que a su vez interactúan con la finalidad de realizar las tareas para las que fueron diseñados. La finalidad de un sistema multi agente es la de operar sobre varios dominios de aplicación distintos y no permanecer en un solo ámbito de desarrollo, además de construir un sistema donde la información y el procesamiento se encuentra distribuido entre agentes y cada uno de ellos realiza sus tareas específicas o dotan de información determinada que permitan resolver un problema (Wooldridge, 2002).

**1.5.3 FIPA.** La Fundación para Agentes Físicos Inteligentes o FIPA por sus siglas en ingles es un organismo a nivel mundial que se dedica a fomentar el estudio y conocimiento de los agentes inteligentes y pasó a ser parte de los estándares de IEEE en 2005. FIPA se especializa en la creación y definición de especificaciones para la comunicación e interoperabilidad de agentes en un entorno determinado. Estas especificaciones y los eventos desarrollados por esta organización impulsaron el desarrollo y el avance de la tecnología basada en agentes al punto que varias ideas que surgieron en FIPA se están introduciendo en nuevas tecnologías como la Web (Foundation for Intelligent Physical Agents, 2002).

***1.5.3.1 Mensajes ACL FIPA.*** El lenguaje de comunicación de agentes o ACL es un lenguaje que permite la intercomunicación entre agentes dentro de un entorno determinado. Un mensaje ACL de FIPA se encuentra formado por parámetros necesarios para una comunicación de alto nivel, los parámetros más comunes son:

- **Performativo:** se trata del tipo de mensaje se va enviar o el tipo de comunicación que se llevará acabo.
- **Remitente:** es el ID del agente que va realizar el envío del mensaje.
- **Receptor:** se compone por el ID del agente que va recibir el mensaje.

- **Contenido:** es lo que se quiere transmitir desde el agente remitente hacia el agente receptor utilizando el performativo determinado en primera instancia (Foundation for Intelligent Physical Agents, 2002).

### 1.6 JADE

JADE es una plataforma que facilita la implementación de un sistema distribuido basado en agentes sin la necesidad de modificar el código principal de la aplicación, además una de sus principales ventajas es que se encuentra desarrollado sobre Java, un lenguaje orientado a objetos, utilizado frecuentemente y con una interfaz de programación de aplicaciones sencilla (Bellifemine, Caire, & Greenwood, 2007).

JADE también facilita el levantamiento de un sistema multi agentes con las especificaciones propuestos por FIPA, mediante su middleware. Un aplicativo distribuido sobre JADE puede ser implementado en varios host o nodos que pueden o no tener instalado el mismo Sistema Operativo. Adicionalmente la plataforma se convierte en un contenedor donde los agentes pueden comunicarse mediante mensajes ACL para delegar, ejecutar o consultar información que poseen otros agentes que se encuentran en los diferentes nodos del sistema (TILAB, 2020).

**1.6.1 Arquitectura de JADE.** Los principales componentes en la arquitectura de una plataforma JADE son los contenedores, quienes se pueden encontrar distribuidos en una red, y los mismos se convierten en el entorno o ambiente en que uno o más agentes pueden convivir. Como se puede visualizar en la figura 2 pueden existir uno o varios contenedores en una plataforma, sin embargo, existe un punto en común que es el contenedor principal que debe ser el primero en iniciarse para que posteriormente los otros contenedores puedan conectarse y todos los agentes que se encuentran en su interior puedan intercomunicarse. Se debe considerar que tanto contenedores como agentes deben tener un identificador o nombre único dentro de la plataforma caso contrario esto puede generar errores al momento de su vinculación al contenedor principal (Bellifemine, Caire, & Greenwood, 2007).

Arquitectura de una plataforma JADE

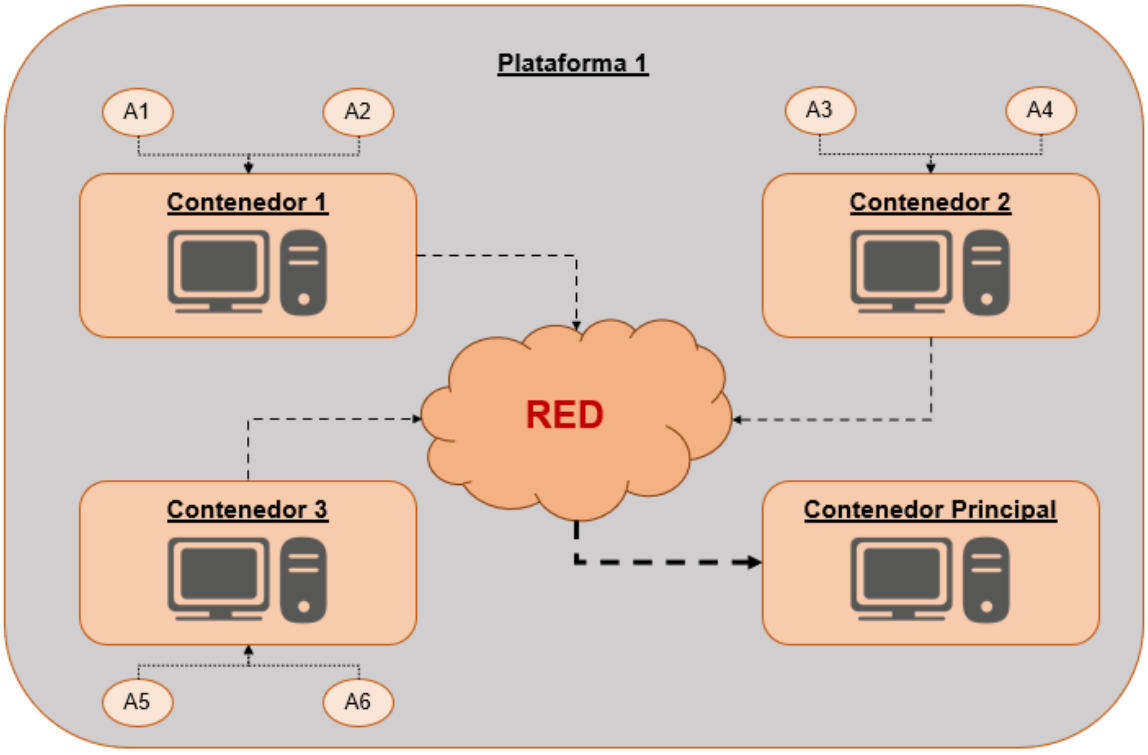



Figura 2. Arquitectura de una plataforma JADE
Fuente: (Bellifemine, Caire, & Greenwood, 2007)

Al iniciar el Contenedor Principal también se generan tres agentes predeterminados en su interior y son utilizados para la administración de la plataforma, estos agentes son:

- **RMA (Remote Management Agent):** también conocido como el agente de administración remota por sus siglas en inglés, este agente es el encargado de crear y eliminar agentes, así también como sus contenedores y gestiona la intercomunicación de los agentes mediante mensajes ACL.
- **AMS (Agent Management System):** es el agente que se encarga de la supervisión y es quien se inspecciona sobre la existencia de nombres de agentes únicos dentro de la plataforma, así como su adecuado registro para la intercomunicación.
- **DF (Directory Facilitator):** es el agente que se encarga de registrar los servicios que cada agente posee, a su vez también pone a disposición los servicios ya registrados en el caso de que un agente necesite usarlos (Garamendi, 2004).

**1.6.2 Agente Dummy.** El agente Dummy es considerado una herramienta que tiene la finalidad de probar los diferentes comportamientos que tendrán el o los varios agentes que se conectarán a la plataforma. Su principal y única característica es la de enviar mensajes personalizados desde la interfaz gráfica de JADE para corroborar la respuesta del resto de agentes y su funcionalidad; los parámetros para enviar un mensaje desde el Agente Dummy son el nombre del agente que recibirá el mensaje, el contenido del mensaje, así como el tipo de mensaje a enviar (Bellifemine, Caire, & Greenwood, 2007).

**1.6.3 Agente Sniffer.** El Agente Sniffer es el encargado de supervisar y mostrar gráficamente el intercambio de mensajes y la ocurrencia de eventos entre los diferentes agentes seleccionados dentro de la plataforma. Como se puede observar en la figura 3 al inicializar este agente en la parte izquierda se podrá visualizar los agentes que son parte de la plataforma, mientras que en la parte derecha se podrá observar los mensajes enviados y recibidos por los agentes (Bellifemine, Caire, & Greenwood, 2007).

Agente Sniffer

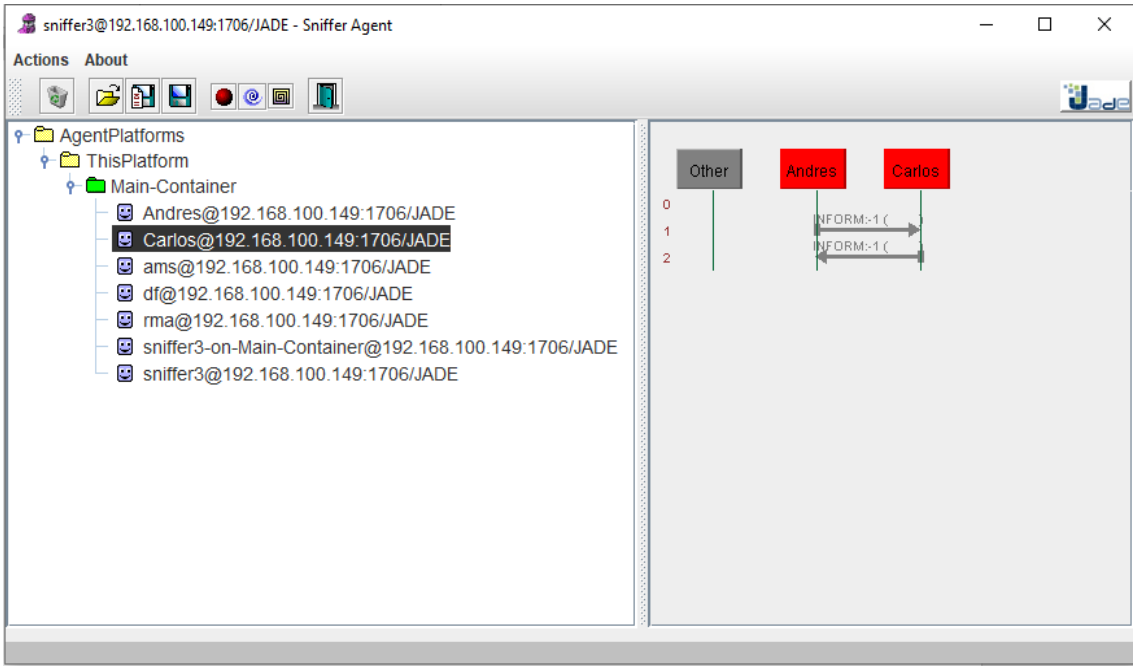



Figura 3. Agente Sniffer
Elaborado por: Carlos Andrés Herrera

**1.6.4 Creación de un agente JADE en Java.** Ya que JADE se encuentra creada sobre lenguaje de programación Java, como se puede observar en la figura 4, la creación de agentes se realiza heredando los métodos de su clase Agent, dentro de los mismos y para inicializar un agente se encuentra el método setup, que es el método obligatorio a implementarse y es donde situamos los parámetros de ejecución del agente que vamos a crear, así como sus comportamientos o acciones. Es importante recordar la importación de la librería de JADE en el IDE que se utiliza (Navarrete, 2008).

Creación de un agente JADE en Java

```
1 import jade.core.Agent;
2
3 public class MiPrimerAgente extends Agent{
4         protectd void setup(){
5                 System.out.println("El agente Carlos Andrés ha sido creado");
6         }
7 }
```

Figura 4. Creación de un agente JADE en Java
Elaborado por: Carlos Andrés Herrera

## 1.7 Metodologías orientadas a agentes

Partiendo de la característica principal de la distribución que se va construir basada en agentes, las metodologías y técnicas de desarrollo de software orientadas a objetos no se ajusta a las características de flexibilidad y autonomía de un agente, además de sus interacciones y demás beneficios dentro de una plataforma distribuida. Por esta razón nace la necesidad de involucrar metodologías adecuadas y que van acorde con los objetivos de las plataformas multi agente, como, por ejemplo, la metodología INGENIAS y GAIA (Wooldridge, Jennings, & Kinny, 2000). INGENIAS es un marco de referencia para la creación de Sistemas Multi Agente donde se resalta la identificación de tareas y actividades que va realizar cada agente y sus interacciones,

considerando la consecución de las metas globales del aplicativo o el sistema en desarrollo. Dentro de sus dos etapas previas al desarrollo del sistema, INGENIAS propone seis modelos, entre ellos: modelo de casos de uso, modelo de organización, modelo de tareas y metas, modelos de agentes, modelo de interacciones y su modelo de entorno.

GAIA es una metodología orientada al desarrollo de sistemas multi agentes, se considera una metodología completa puesto que se enfoca de forma general y específica a un sistema conformado por agentes, donde cuya principal característica es ver al aplicativo como una organización, con el fin de solventar un problema, donde cada agente cumple un rol y donde pueden intercomunicarse, haciendo de esta forma más sencilla la misión de alcance del objetivo de la plataforma o el aplicativo para el que fue construido (Wooldridge, Jennings, & Kinny, 2000).

Considerando la utilidad de cada una de las metodologías, GAIA es la seleccionada para el desarrollo de la plataforma que será estudiada en el presente documento, a continuación, se describen sus principales características.

**1.7.1 Metodología GAIA.** La metodología GAIA pone gran énfasis en los cinco modelos que propone dentro de sus etapas de análisis y diseño que direccionan el adecuado desarrollo de un sistema multi agente (Wooldridge, Jennings, & Kinny, 2000).

***1.7.1.1 Análisis.*** En la primera etapa se trata de dar sentido y conocer cómo funcionará el sistema o plataforma con una visión general, sin la necesidad o intención de una implementación de lo que se propone, adicionalmente, y para un mejor entendimiento esta etapa se puede interpretar mediante una estructura jerárquica, donde encabeza el sistema y hacia abajo en un segundo nivel se encuentran los roles y las interacciones que existirá entre ellos (Leal, Suarez, Guerrero, & Gutierrez, 2018).

Posteriormente y una vez que se hayan definido los roles que el sistema necesitará para su funcionamiento, en un tercer nivel y bajo estos se establecen responsabilidades para cada uno de ellos. Para que un rol sea reconocido como tal es necesario que posea cuatro características principales como son: responsabilidades, permisos, actividades y protocolos, como se puede ver en la figura 5 (Wooldridge, Jennings, & Kinny, 2000).

Estructura jerárquica GAIA

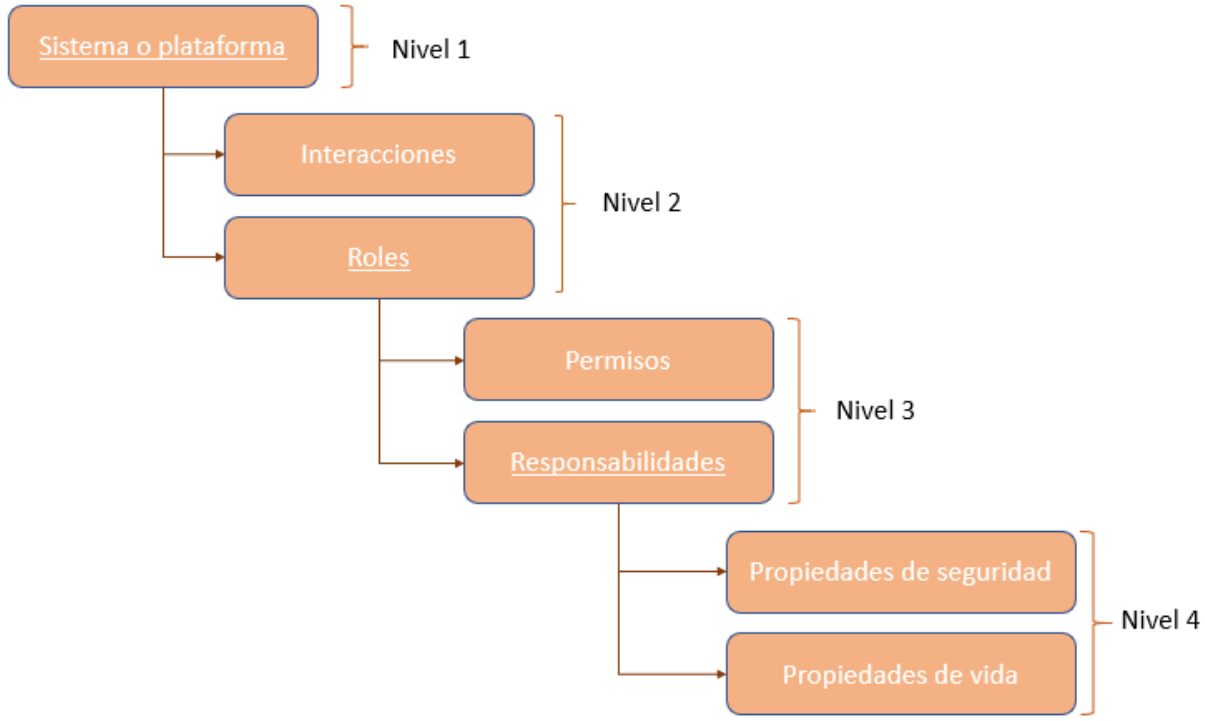



Figura 5. Estructura jerárquica GAIA
Elaborado por: Carlos Andrés Herrera

a) **Modelo de roles:** el modelo de roles se encarga de relevar y definir quienes son los roles indispensables del sistema, un rol tiene la misma definición que un cargo en una organización real y sus características destacan por tener atributos tales como:

- **Permisos:** nos permiten conocer los recursos que tiene a su disposición para ejecutar el rol designado, de esta forma pueden existir roles que necesiten solamente leer información, o también existirán roles que necesiten modificar la misma con el fin de cumplir con sus responsabilidades.
- **Responsabilidades:** que nos permite conocer la finalidad del rol y su funcionalidad dentro del sistema. Estas responsabilidades se dividen en dos: de vida y seguridad, en el primer caso se tratan de actividades constantes, mientras el rol o el agente se mantiene vivo,

mientras que el segundo caso hace referencia a situaciones que dentro del sistema serian no deseables.

- **Actividades:** de forma similar a un método en la programación orientada a objetos, define un procedimiento a seguir (Wooldridge, Jennings, & Kinny, 2000).

En la figura 6 se puede observar la forma de representación.

Plantilla de un modelo de roles

| Rol | Nombre del rol |
|---|---|
| Descripción | Breve descripción del rol |
| Actividades | Actividades en la que el rol participa |
| Permisos | Derechos asociados con el rol |
| Responsabilidades | |
| De vida | Responsabilidades vitales |
| De seguridad | Responsabilidades de seguridad |

Figura 6. Plantilla de un modelo de roles
Elaborado por: Carlos Andrés Herrera

**b) Modelo de interacciones:** como en una organización donde existen varios roles o cargos, en algún momento va ser necesaria la intercomunicación entre ellos para el correcto funcionamiento del sistema con el fin de llegar al objetivo o resolución del problema, por esta razón GAIA propone el modelo de interacciones y bajo este modelo se establecen grupos de protocolos para cada una de las comunicaciones entre los roles existentes en el sistema multi agente. Es importante mencionar que los protocolos constan de las siguientes características:

- **Nombre del protocolo:** se describe la finalidad de la interacción
- **Iniciador:** corresponde a quien envía la interacción
- **Respondedor:** es el emisor de la interacción

- **Entradas:** información enviada por el iniciador
- **Salidas:** información correspondiente al resultado del respondedor
- **Descripción del protocolo:** descripción de alguna tarea o proceso que realiza el iniciador mientras la interacción se ejecuta (Wooldridge, Jennings, & Kinny, 2000).

En la figura 7 se puede observar la notación del modelo de internaciones.

Plantilla de un modelo de interacciones

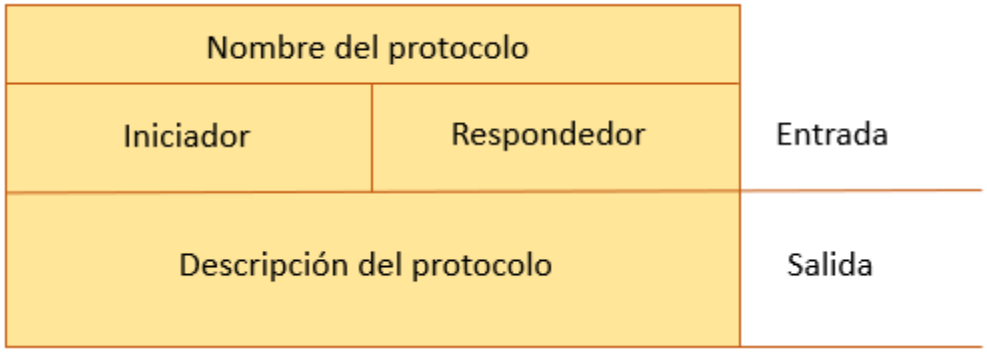



Figura 7. Plantilla de un modelo de interacciones
Elaborado por: Carlos Andrés Herrera

***1.7.1.2 Diseño.*** Para la etapa de diseño, a diferencia de las metodologías tradicionales orientadas a objetos que buscan llevar la etapa abstracta de análisis a un nivel de entendimiento mucho más sencillo con el fin de dar mayor facilidad al momento de la implementación, sin embargo, GAIA opta por dar mayor énfasis a la comunicación que existe entre agentes para poder llegar a construir una organización con objetivos en común. La etapa de diseño de GAIA propone su enfoque en el desarrollo de tres modelos principales: modelo de agentes, modelos de servicios y modelo de conocimiento (Leal, Suarez, Guerrero, & Gutierrez, 2018).

a) **Modelo de agentes:** el modelo de agentes tiene como finalidad la identificación de los diferentes agentes que serán parte del sistema, un agente puede contener uno o varios roles

dentro de sí mismo, de esta manera se consigue determinar cuál será la funcionalidad o el papel que jugará el agente dentro de la plataforma que se encuentra en construcción. La notación para el modelo de agentes se interpreta mediante un árbol simple como se representa en la figura 8, donde será también visible el número de instancias que tendrá el agente en el sistema de acuerdo a la notación definida en la figura 9 (Wooldridge, Jennings, & Kinny, 2000).

Notación en árbol del modelo de agente

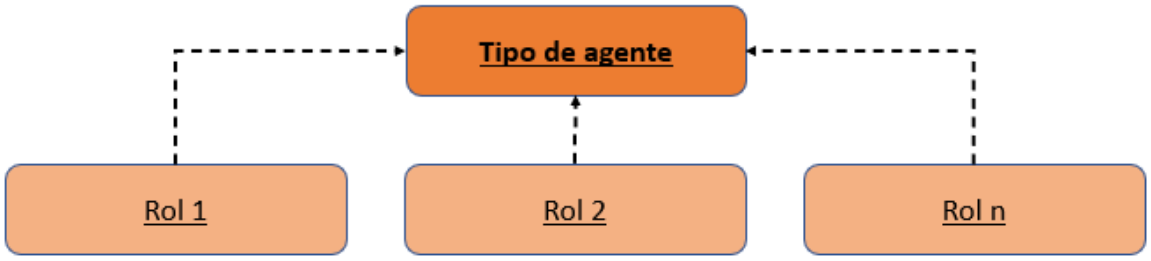



Figura 8. Notación en árbol del modelo de agente
Elaborado por: Carlos Andrés Herrera

Notación de instancias de un agente

| Notación | Descripción |
|---|---|
| n | n instancias del agente |
| m..n | existirán entre m y n instancias |
| * | cero o mas instancias |
| + | una o mas instancias del agente |

Figura 9. Notación de instancias de un agente
Elaborado por: Carlos Andrés Herrera

**b) Modelo de servicios:** este modelo se encargará de especificar los servicios que se encuentran asociados a cada uno de los roles que fueron relevados en la etapa de análisis, esto implica que cada actividad definida en la etapa mencionada se convertirá en un servicio en el presente

modelo. A pesar de tener similitud a un método en la programación orientada a objetos, un servicio es un conjunto de actividades en que formará parte un agente determinado. Los servicios deben tener definidos sus principales atributos: entradas, salidas, las condiciones previas y posteriores (Wooldridge, Jennings, & Kinny, 2000).

c) **Modelo de conocimiento:** finalmente, en la figura 10 se encuentra el modelo de conocimiento, que es el encargado de graficar las comunicaciones que existirán entre agentes, no necesariamente se detallan los mensajes que se enviarán, tampoco el momento en el que la comunicación ocurre, sin embargo, es parte fundamental para poder determinar la fluidez y funcionalidad de la plataforma a desarrollar. Ocasionalmente incluso en esta etapa surge la necesidad de regresar a ver y probablemente corregir errores en la etapa de análisis que de alguna forma puede tener impacto en el correcto desempeño del sistema multi agente visible en el modelo actual. Normalmente este modelo es interpretado mediante nodos y líneas que representan la comunicación entre ellos, para fines prácticos la flecha apunta hacia el destino sin necesidad de que exista una respuesta reciproca de vuelta (Wooldridge, Jennings, & Kinny, 2000).

Plantilla de modelo de conocimiento

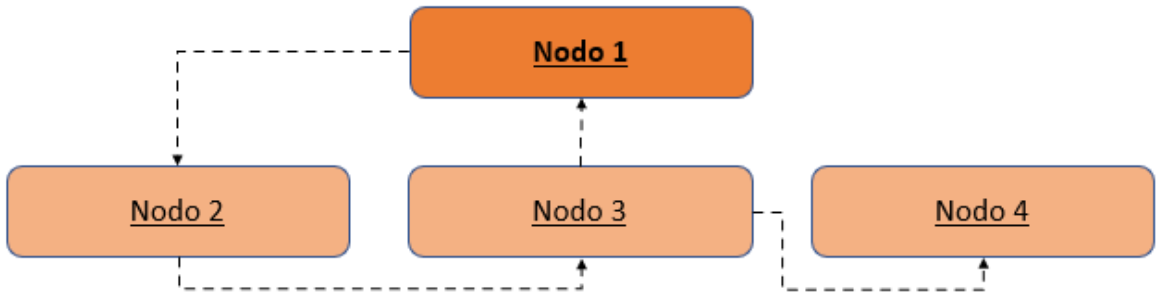



Figura 10. Plantilla de modelo de conocimiento
Elaborado por: Carlos Andrés Herrera

De manera simplificada, las actividades y etapas mencionadas anteriormente, así como su jerarquía se pueden ver expresadas en la figura 11.

Esquema de relación de GAIA

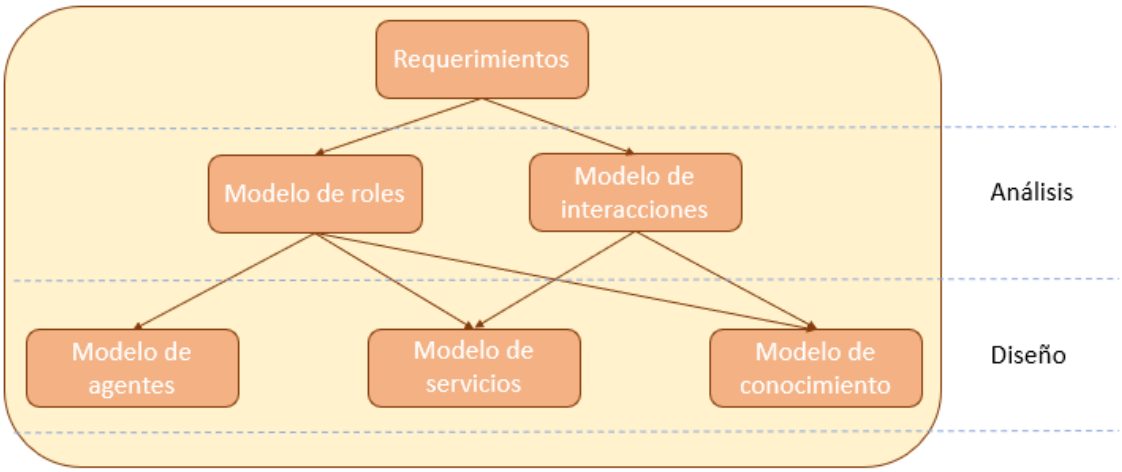



Figura 11. Esquema de relación de GAIA
Elaborado por: Carlos Andrés Herrera

### 1.8 Métricas

Una métrica en el contexto de software es considerada como uno de los principales elementos en el desarrollo de aplicaciones con el fin de medir y controlar su desempeño, incluso poder de cierta manera predecir su calidad. Para esto se consideran actividades tales como el registros y revisión de eventos durante la ejecución de la etapa de pruebas del software desarrollado. Por otra parte, también se puede considerar una métrica como la evaluación especifica de ciertas características del software (Lee & Cheung, 1987).

**1.8.1 Métrica de rendimiento.** Según Samadhiya, Wang, & Chen (2010) el modelo FURPS define a las métricas de rendimiento como la medición de la utilización de recursos, velocidad de procesamiento, tiempos de respuesta y eficiencia de un sistema. Considerando que el objetivo del Sistema Multi Agente es el de mejorar el tiempo de respuesta y optimizar el uso de recursos del dispositivo móvil a través de la baja utilización de los mismos, se analizarán las siguientes variables relacionadas a las métricas de rendimiento.

- **Tiempo:** mediante el método *nanoTime()* de Java obtenemos el valor del temporizador del sistema en nano segundos de ese instante, esta operación se la realiza en el momento en el que se va iniciar la ejecución del proceso que se desea medir, posteriormente necesitamos identificar un punto especifico donde el proceso deja de ejecutarse para poder obtener nuevamente el valor de *nanoTime()*; para luego obtener el tiempo transcurrido entre el inicio y fin del proceso que se desea medir.
- **Memoria:** para poder medir tanto el uso de memoria en tiempo de ejecución y los picos más altos de consumo, nos direccionamos a los archivos del sistema de Android, que al tener el mismo kernel que Linux, se puede aplicar los mismos conceptos y extraer la información necesaria desde sus ficheros.

  Así tenemos:

$$M_{uso} = ((m_r - m_s) * P)/Mb$$

  Donde:

  $m_r$: es la memoria residente o memoria dedicada al proceso en ejecución (valor en páginas).

  $m_s$: es la memoria compartida con otros procesos (valor en páginas).

  $P$: es el valor por página de memoria equivalente a 4096 expresada en KB

  $Mb$: es el valor de 1024 para tener el resultado de memoria en uso expresada en MB

- **CPU: p**ara la medición de esta variable se utilizó los conceptos y archivos del sistema del kernel de Linux, como datos de entrada necesitamos 3 tiempos de CPU: el tiempo de actividad del dispositivo, el tiempo de uso en modo kernel y el tiempo de uso en modo usuario.

  Posterior a la obtención de estos datos se aplica:

$$C_{uso} = \left[\frac{\left((t_{u_{actual}} - t_{u_{inicial}}) + (t_{k_{actual}} - t_{k_{inicial}})\right)}{(t_{a_{actual}} - t_{a_{inicial}})}\right] / N_p$$

  Donde:

  $t_{u_{actual}}$: es el tiempo en modo usuario del CPU capturado en ese instante.

  $t_{u_{iniciall}}$: es el tiempo inicial en modo usuario del CPU.

  $t_{k_{actual}}$: es el tiempo en modo kernel del CPU capturado en ese instante.

  $t_{k_{iniciall}}$: es el tiempo inicial en modo kernel del CPU.

  $t_{a_{actual}}$: es el tiempo que se ha mantenido activo el dispositivo capturado en ese instante.

  $t_{a_{iniciall}}$: es el tiempo que se ha mantenido activo el dispositivo captura al inicio del proceso.

  $N_p$: es el número de procesadores activos que tiene el dispositivo.

**1.8.2 Métrica de eficiencia de código.** Para la métrica de eficiencia de código se considerará los resultados obtenidos por la herramienta SonarQube, se trata de una herramienta de software libre que es utilizada para la revisión de código fuente, entre sus principales variables están las de detección de errores, detección de vulnerabilidades o seguridad y mantenibilidad; maneja varios lenguajes de programación entre ellos Java, Python, C#, entre otros. La revisión se lleva a cabo a partir de la dotación del código fuente del proyecto junto con sus clases compiladas, además del archivo de configuración Scanner donde se establecen características como el nombre del proyecto, directorio que será evaluado y clases compiladas del proyecto (SonarSource S.A, 2020).

**1.8.3 Métrica de transferencia de datos.** Para la medición de la métrica de transferencia de datos se usará la herramienta Live TCP UDP Watch, esta herramienta tiene licencia libre y es un aplicativo utilizado para extraer datos relevantes sobre una red tales como el tráfico de red, envío y recepción de paquetes, protocolos de transferencia, etc (Sofer, 2020).

## 1.9 Estado del arte

El uso de dispositivos móviles en la vida cotidiana ha crecido exponencialmente en la última década de acuerdo al informe publicado por el INEC (institución nacional encargada de levantar datos y presentar estadísticas del Ecuador para el uso del gobierno en la toma de decisiones) en 2017, donde se puede observar el incremento en el uso de dispositivos inteligentes de al menos cinco veces más en comparación al año 2012 donde el 6.2% de personas encuestadas declararon tener al menos un smartphone. En esta misma publicación se enfatiza cómo en el año 2017 esta cifra incrementa hasta el 37.2% a nivel nacional como se puede ver en la figura 12 (INEC, 2017).

Porcentaje de personas que tienen teléfono inteligente

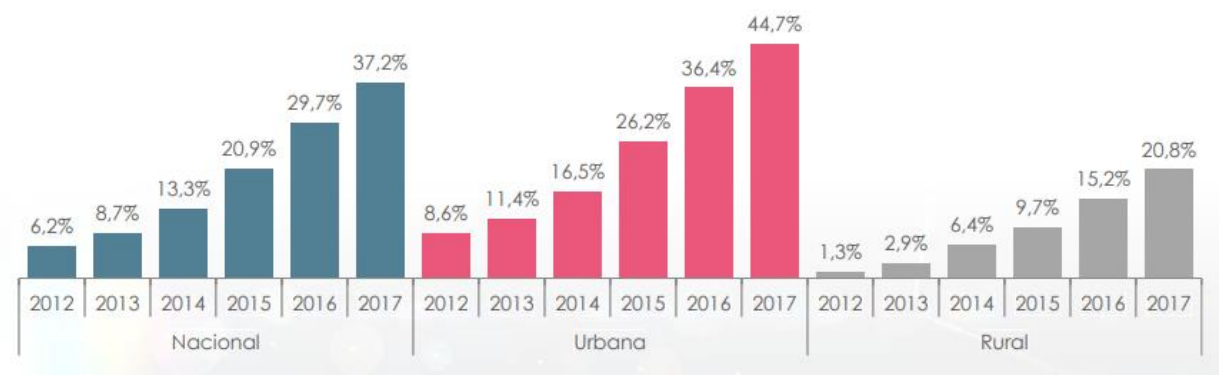



Figura 12. Porcentaje de personas que tienen teléfono inteligente
Fuente: (INEC, 2017).

Adicionalmente, y en este mismo periodo de tiempo se puede observar como el uso de computadores de escritorio no ha tenido variaciones significativas, al contrario, se ha mantenido en el rango del 26%, pero se puede observar en la figura 13 como en el año 2017 ha decaído un 0.8% en comparación a los datos del año 2016, dando a entender que la tendencia en el uso de tecnología es el incremento en dispositivos móviles por sus características inherentes como la portabilidad y la posibilidad de realizar tareas similares a las ejecutadas en un computador (INEC, 2017).

Porcentaje de personas que usan computadores

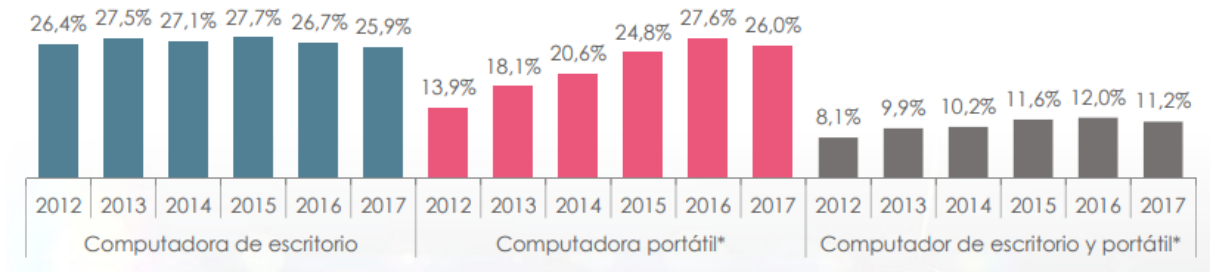



Figura 13. Porcentaje de personas que usan computadores
Fuente: (INEC, 2017).

Uno de los objetivos que más ha perseguido la computación móvil ha sido el tener la información disponible y en cualquier lugar para los usuarios, razón por la cual se ha impulsado la investigación de la tecnología inalámbrica con el uso de aplicativos que puedan ser implementados en estos dispositivos (Satyanarayanan, 2010).

A pesar de la portabilidad que tiene el uso de dispositivos móviles y la facilidad en el acceso a información para el usuario, existen dificultades en el uso de estos dispositivos para la ejecución de ciertos aplicativos, y es la limitación que estos poseen a nivel de recursos en el apartado de hardware, razón por la cual su explotación a su máximo nivel en muchos casos no es posible. Sin embargo y para mitigar este vacío se puede utilizar los servicios de recursos externos con mayores prestaciones para la ejecución de tareas que pueden ser muy demandantes en dispositivos móviles, conocida como computación en la nube móvil (Niroshinie, Seng, & Wenny, 2012).

La nube móvil ha dado paso a la creación de varios métodos o técnicas para el ahorro de recursos locales en dispositivos con características limitadas, una de ellas el *code offloading*, cuya mayor ventaja es reflejada en la disminución de tiempos de respuesta y la sensación al usuario de tener mayores prestaciones en sus dispositivos móviles. El concepto de *code offloading* viene de la distribución o delegación de los procesos intensivos hacia una instancia remota donde se puede ejecutar estas tareas con mayor facilidad y obtener de vuelta la respuesta en el dispositivo móvil (Flores, y otros, 2015).

Entre las diferentes propuestas de implementación de técnicas de *code offloading* a lo largo del tiempo, se pueden mencionar varias, por ejemplo:

- **MAUI:** en su trabajo de investigación, Cuervo y otros (2010), mencionan la capacidad de la batería y su tecnología como obstaculo principal para el desarrollo potencial de los dispositivos moviles, razón por la cual proponen el proyecto MAUI, enfocado en la utilización de *code offloading* para optimizar el uso de la batería, una de sus características

principales es la de gestionar el código y definición de los métodos que se van a enviar a procesar en un computador remoto a través de anotaciones manuales en el código, mediante la implementación de este sistema y probado en el aplicativo de reconocimiento de rostros, se pudo conseguir la disminución de consumo de energía del dispositivo móvil hasta 17 veces menos que con otros métodos de ejecución.

- **COMET:** este trabajo de investigación propone la carencia de memoria en dispositivos móviles como una característica limitadora al momento de ejecutar tareas demandantes, y enfatiza el uso de *code offloading* con memoria compartida en distintas maquinas que son parte de una misma red como solución a esta necesidad considerando la carga de trabajo que mantiene cada nodo para la asignación de las tareas a migrar. Los resultados aplicando esta técnica fueron de una aceleración en el tiempo de ejecución de hasta 15 veces más rápida a comparación de ejecuciones locales. (Gordon, Jamshidi, Mahlke, Mao, & Chen, 2012).
- **CloneCloud:** Chun, Ihm, Maniatis, & Naik (2010) en su trabajo resaltan la gran utilidad que tienen los dispositivos móviles para poner a disposición del usuario la información de manera ubicua y plantea la utilización de computadores de escritorio, portátiles e incluso la nube como una manera acertada de dar mayor agilidad y mejores resultados en la utilización de aplicativos en dispositivos móviles, de esta idea nace CloneCloud, con el objetivo de enviar las tareas que son más intensivas en ejecuciones locales, con la creación de dispositivos en la nube capaces de procesar estas tareas y devolver un resultado. De acuerdo a los resultados obtenidos la ejecución con CloneCloud tuvo una aceleración de hasta 21 veces mayor a las ejecuciones locales.

Se vuelve fundamental para la optimización de los recursos en dispositivos móviles el uso de técnicas de *code offloading* en procesos intensivos o demandantes tales como la binarización de imágenes digitales donde se evalúa la matriz de pixeles generada, considerando un valor determinado de umbral para el caso del uso de técnicas de umbral global, en el trabajo realizado por Magro (2013) se puede observar como el uso de este método tiene resultados positivos, en la figura 14 se puede ver los diferentes resultados aplicando un valor de umbral diferente, en el caso del recuadro 9 se trata de la imagen original en escala de grises, el recuadro 10 utiliza un valor de umbral de 110, mientras que el recuadro 11 y 12 es el resultado del uso de valor de umbral de 100 y 90 respectivamente.

Resultados con umbral global simple

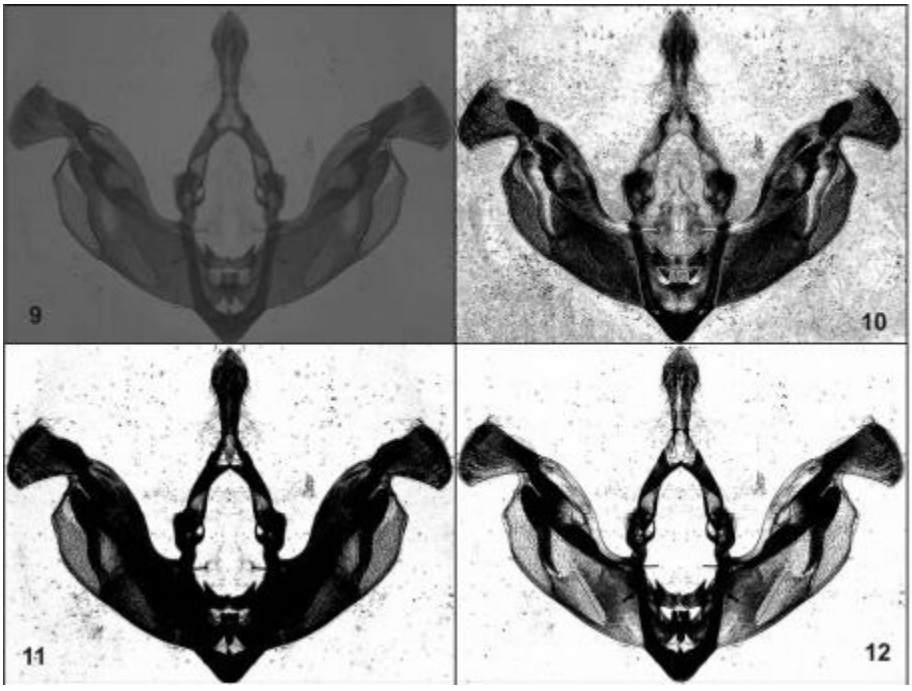



Figura 14. Resultados con umbral global simple
Fuente: (Magro, 2013)

Finalmente y conforme a las referencias obtenidas de las investigaciones anteriormente descritas, se pudo observar que el uso de técnicas de *code offloading*, designación de tareas y la implementación de computadores remotos es una de las soluciones a los problemas inherentes de los dispositivos móviles que poseen características limitadas en cuanto a hardware para el procesamiento de tareas intensivas como la binarización de imágenes, por esta razón y para la implementación de este proyecto se ha contemplado las siguientes características:

- Uso de técnicas de *code offloading*.
- Decisión automática en el envío de tareas hacia el computador remoto en base a las dimensiones de las imágenes seleccionadas.
- Utilización de recursos externos para la ejecución de tareas intensivas.
- Implementación de agentes colaborativos para la comunicación en la plataforma.
- Uso de la técnica de umbral global simple para la implementación de la binarización de imágenes

## Capítulo 2

## Metodología

Para el desarrollo de este proyecto se utilizó de la metodología GAIA conforme a las etapas descritas en el capítulo 1. Considerando que se basa en la implementación de un sistema multi agente para la binarización de imágenes, cuya arquitectura se puede observar en la figura 15, donde tanto los ClientAgent como el ServerAgent se conectan a un contenedor principal que se encuentra en un computador dentro de la red, para el funcionamiento un ClientAgent debe enviar solicitudes al ServerAgent mediante mensajes ACL para poder realizar el procesamiento de las imágenes recibidas. Posteriormente cuando el hilo creado por el ServerAgent haya terminado su ejecución, este creará un DeliveryAgent quién es el encargado de devolver la imagen binarizada al ClientAgent para después ser eliminado del sistema automáticamente, misma situación que sucede con el ClientAgent cuando se hayan terminado de atender sus solicitudes.

Arquitectura del Sistema Multi Agente propuesto

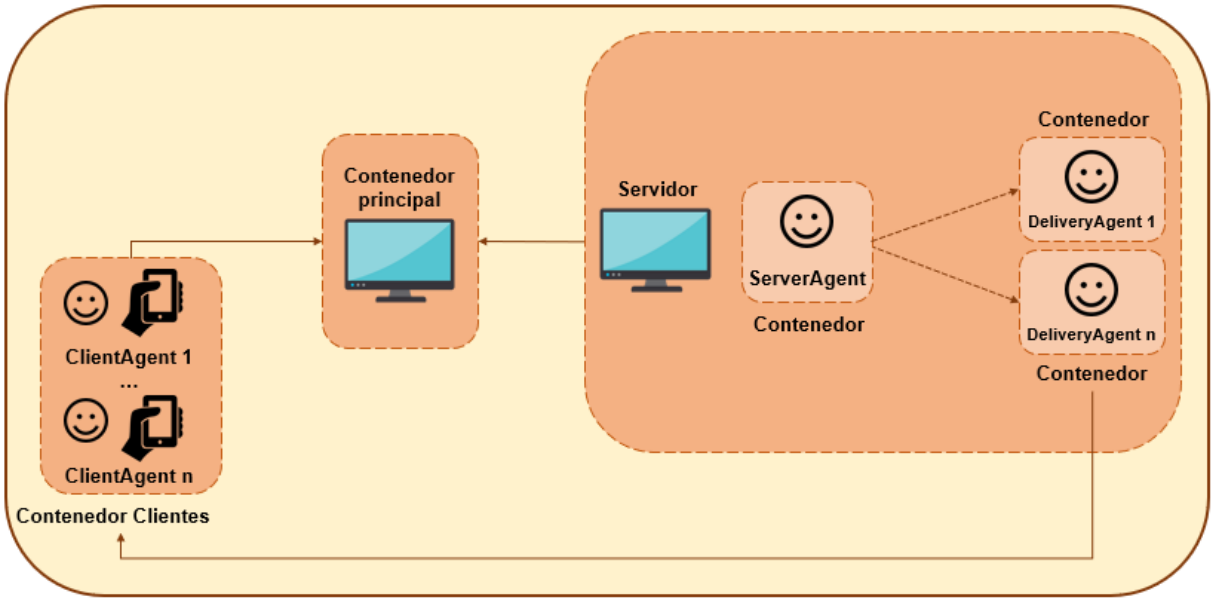



Figura 15. Arquitectura del Sistema Multi Agente propuesto
Elaborado por: Carlos Andrés Herrera

### 2.1. Fase 1: Esquematización de roles

Para el desarrollo del aplicativo propuesto primero se definieron los roles que van a ser parte de nuestro Sistema Multi Agente, así tenemos:

- **Cliente:** este rol será el encargado de inicializar y desencadenar el proceso. El Cliente enviará una solicitud con una o más imágenes seleccionadas por el usuario desde el almacenamiento interno del dispositivo o a su vez capturar una fotografía y enviarla para ser procesada, el objetivo es transformar las imágenes a escala de grises y volverla a guardar en un directorio específico en cuanto finalice el proceso. Para este rol se tiene la consideración de examinar una a una las imágenes y sus dimensiones al momento de ser seleccionadas para poder definir si su procesado será de forma local o se enviará el proceso a un agente remoto dentro de la plataforma, el esquema se puede ver en la figura 16.
- **Servidor:** cómo se puede ver en la figura 17, el rol de servidor se encontrará a la espera de recibir solicitudes para poder realizar el proceso de transformación de las imágenes receptadas, con la condición de bloquearse al no tener una cola de solicitudes en espera para poder disminuir el uso de recursos del computador en que se encuentra contenido.
- **Mensajero:** este rol se encarga principalmente de mantenerse a la espera de mensajes por parte del rol Servidor para re direccionarlos hacia el rol Cliente que haya sido el emisor en primera instancia de la solicitud a la plataforma. Una particularidad de este rol es que se crea como parte de las actividades del rol Servidor y su ciclo de vida termina al enviar la respuesta al rol Cliente, sus características se describen en la figura 18.

Esquema del rol Cliente

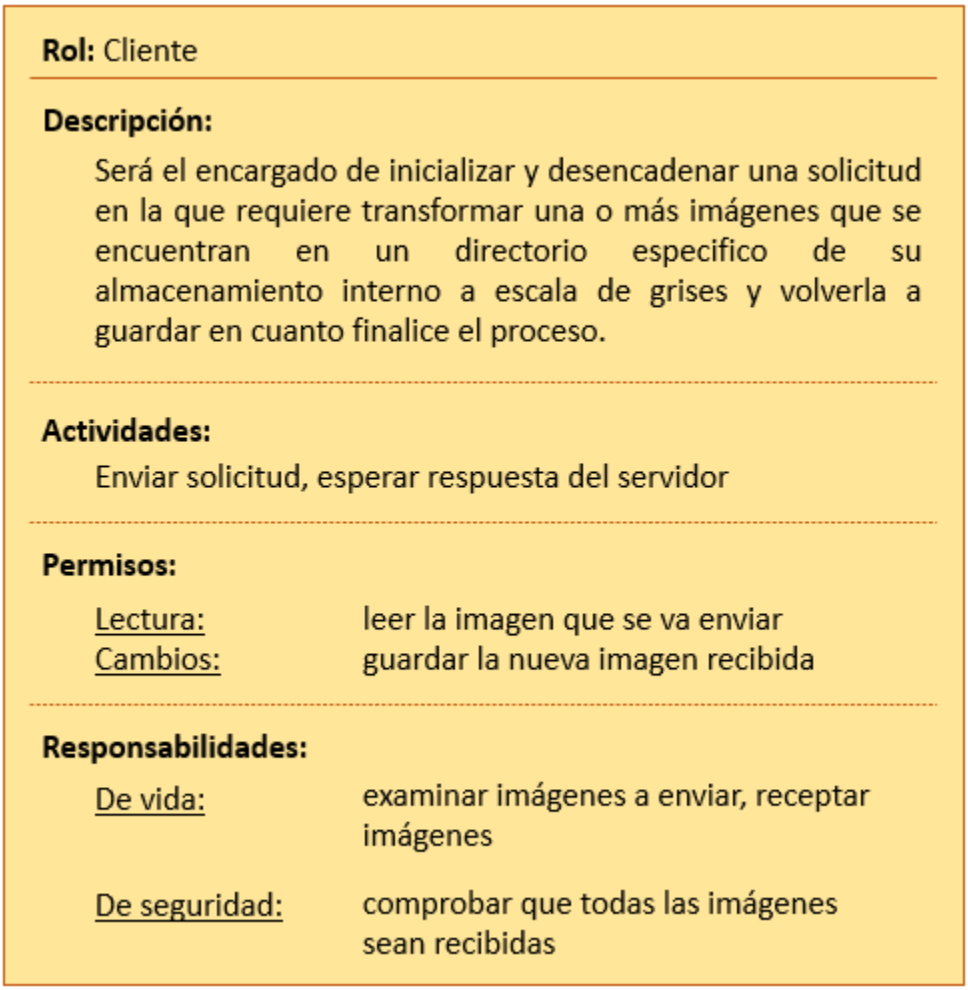



Figura 16. Esquema del rol Cliente
Elaborado por: Carlos Andrés Herrera

Esquema del rol servidor

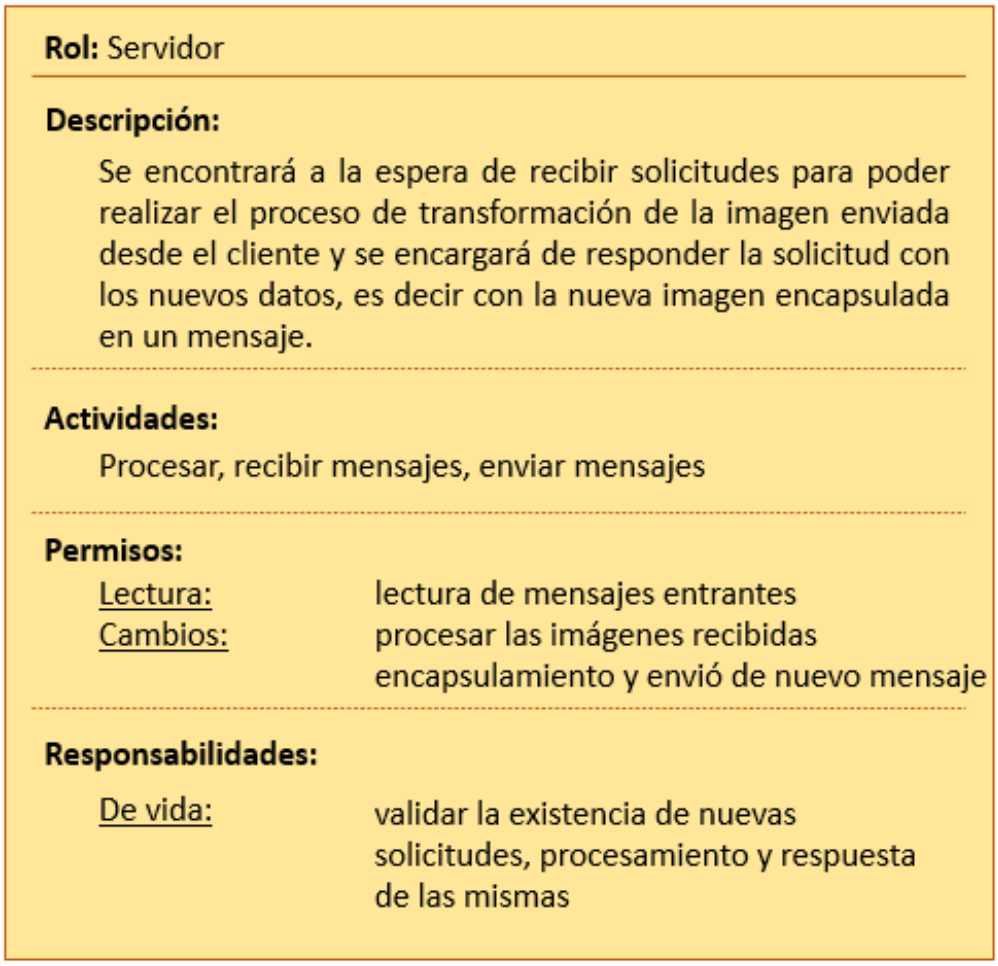



Figura 17. Esquema del rol servidor
Elaborado por: Carlos Andrés Herrera

Esquema del rol mensajero (temporal)

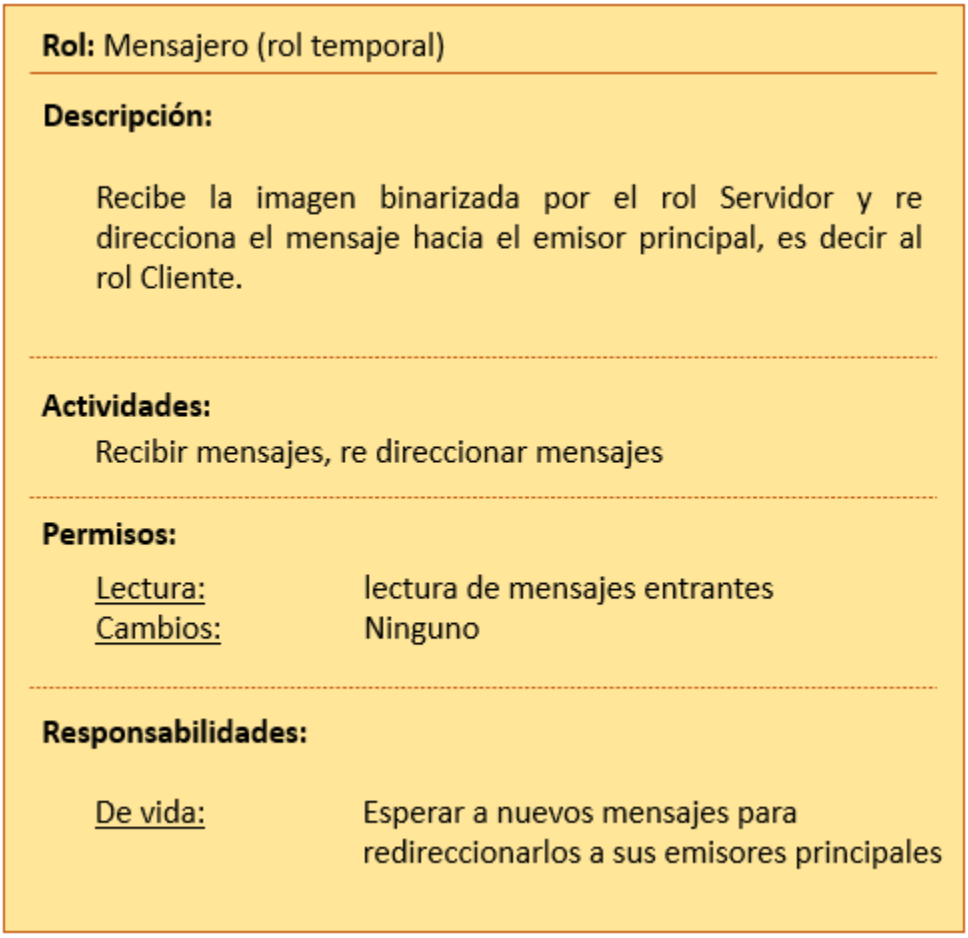



Figura 18. Esquema del rol mensajero (temporal)
Elaborado por: Carlos Andrés Herrera

### 2.2. Fase 2: Esquematización de interacciones

Para el Sistema Multi Agente propuesto se consideró la interacción entre los roles creados en el punto anterior, de tal manera que cuando el rol cliente empiece a requerir la binarización de las imágenes seleccionadas como se puede observar en la figura 19, se procederá con la validación de la imagen y resolución que esta tenga para determinar si se ejecuta de forma local o se envía a procesarla en el computador remoto mediante el envío de solicitudes con la transformación de la imagen a mapas de bits hacia la nube donde el rol Servidor realizará la transformación. Una vez el rol servidor recibe el mensaje con los datos necesarios (ID del emisor y mapa de bits) crea un nuevo

hilo para poder procesar solicitudes de forma paralela. En la figura 20 se puede ver que una vez la imagen termina de ser procesada, se crea un rol temporal denominado Mensajero, quien mantiene contacto tanto con el rol Servidor y el rol Cliente para poder re direccionar el mensaje de respuesta hacia el desencadenador de la solicitud, esta comunicación se puede ver en la figura 21.

Protocolo de envió de imágenes

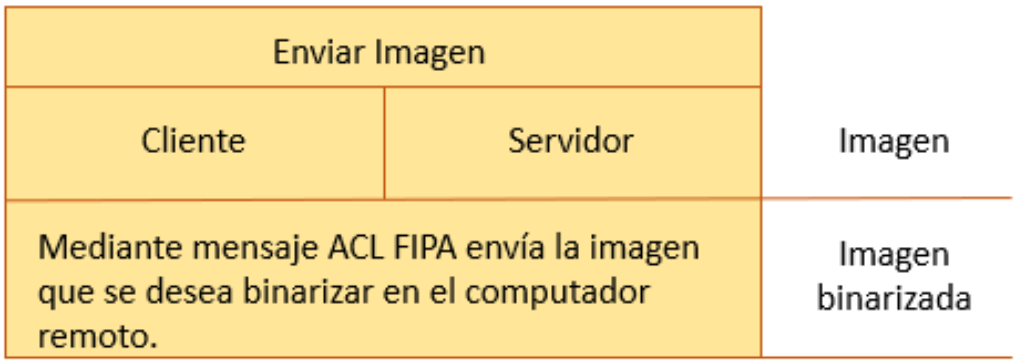



Figura 19. Protocolo de envió de imágenes
Elaborado por: Carlos Andrés Herrera

Protocolo de imagen procesada

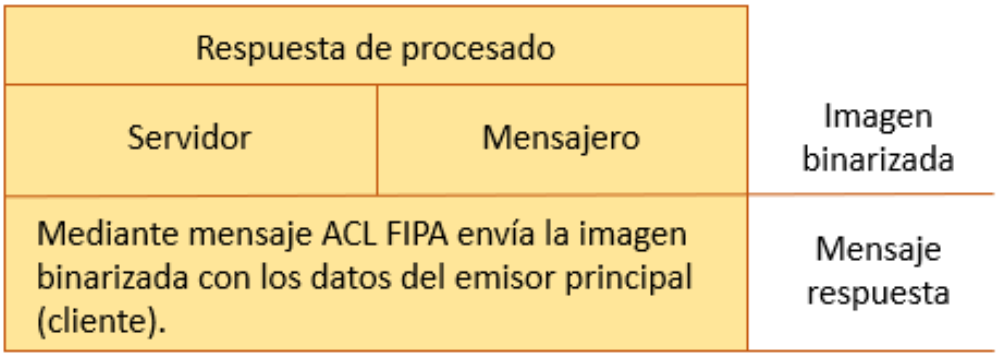



Figura 20. Protocolo de imagen procesada
Elaborado por: Carlos Andrés Herrera

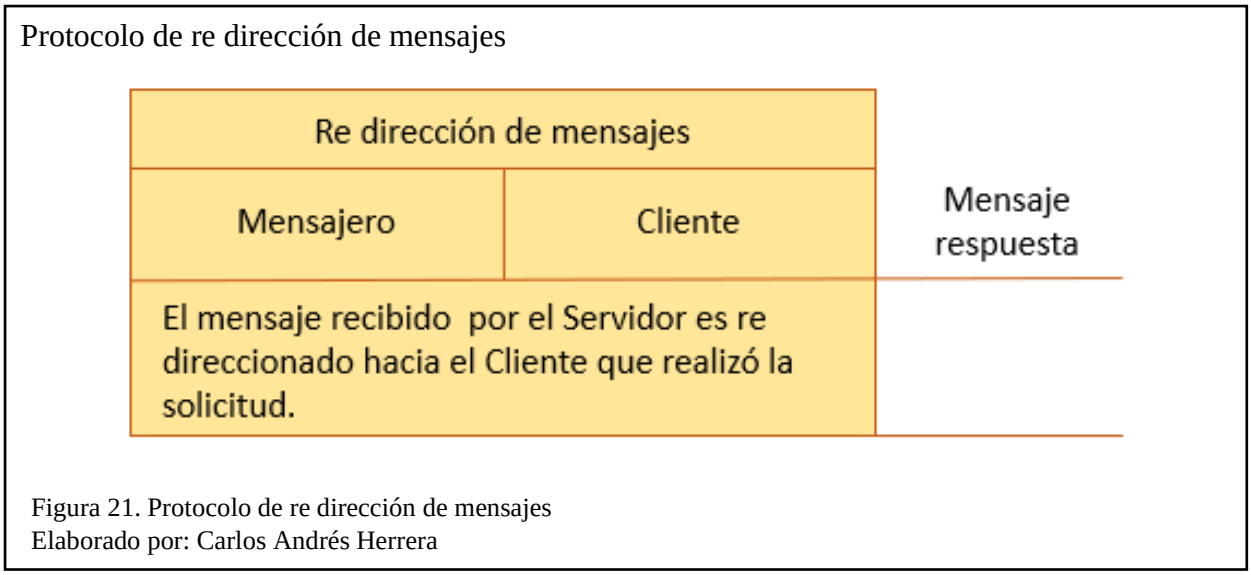



Figura 21. Protocolo de re dirección de mensajes
Elaborado por: Carlos Andrés Herrera

### 2.3. Fase 3: Esquematización de agentes

Para el modelado de agentes de la plataforma en desarrollo se consideró un computador portátil como computador remoto donde se instanciará un agente Servidor del tipo de rol Servidor, en el caso de los clientes dependerá de la demanda de dispositivos conectándose a la plataforma, y se creará uno por dispositivo, es decir existirán cero o más agentes de tipo Cliente, dado que la plataforma puede tener el caso donde no se encuentre conectado ningún dispositivo generando solicitudes. Por último y de la misma manera los agentes de tipo Mensajero serán creados por el hilo generado por el Servidor dependiendo la cantidad de solicitudes a responder o número de mensajes con imágenes procesadas por el Servidor que hayan sido atendidas, esto puede ser cero o más agentes de este tipo, estas correspondencias se encuentran diagramadas en la figura 22.

Modelo de agentes

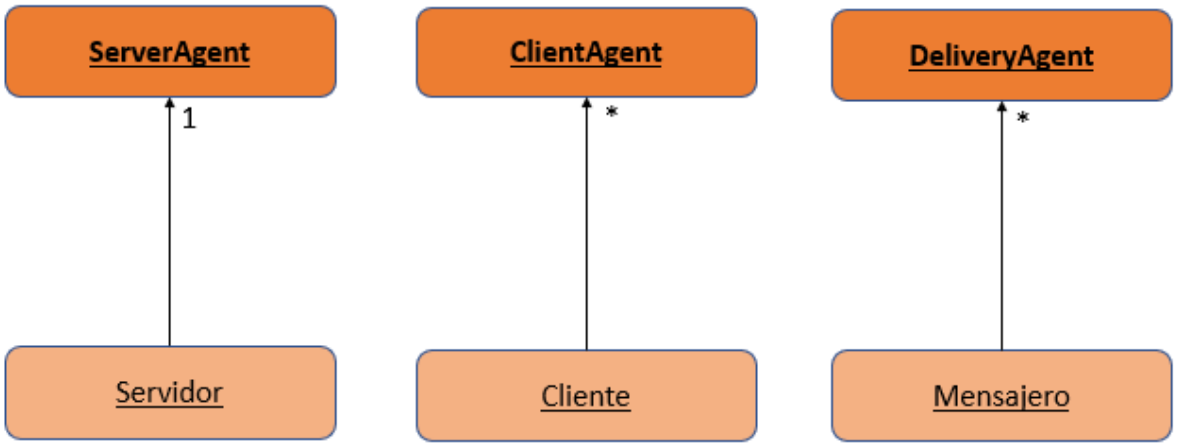



Figura 22. Modelo de agentes
Elaborado por: Carlos Andrés Herrera

### 2.4. Fase 4: Esquematización de servicios

En este apartado se describen los servicios que dispondrán los agentes dentro del Sistema Multi Agente, en la tabla 1 se puede ver el detalle de los servicios que contiene el sistema.

Tabla 1. Modelo de servicios

| Servicio | Entradas | Salidas | Pre condiciones |
|---|---|---|---|
| Examinar imágenes | Imagen | Mapa de bits | Imágenes con formato JPG con valor profundidad de bits igual a 24 |
| Procesamiento local | Mapa de bits | Mapa de bits (binarizado) | Producto de las dimensiones de la imagen menores a 20000000, valor correspondiente a matrices de 5000 pixeles x 4000 pixeles o equivalente |
| Envío de imágenes | Mapa de bits | Mensaje ACL | Producto de las dimensiones de la imagen menores a 20000000, valor correspondiente a matrices de 5000 pixeles x 4000 pixeles o equivalente |
| Recepción de imágenes | Mensaje ACL | No aplica | No aplica |

| | | | |
|---|---|---|---|
| Procesamiento de imágenes | Mapa de bits | Mapa de bits (binarizado) | No aplica |
| Envío de mensaje | Mapa de bits | Mensaje ACL | No aplica |
| Respuesta de solicitud | Mensaje ACL | Mensaje ACL | No aplica |

Nota: esta tabla contiene el modelo de servicios

### 2.5. Fase 5: Esquematización de conocimiento

El modelo de conocimiento del Sistema Multi Agente coloca al ClientAgent en la parte superior del gráfico para poder distinguir quien desencadena la solicitud para la plataforma, seguido de las comunicaciones que existen entre el agente mensajero y el servidor, el ciclo de vida del proceso se encuentra en la figura 23.

Modelo de conocimiento

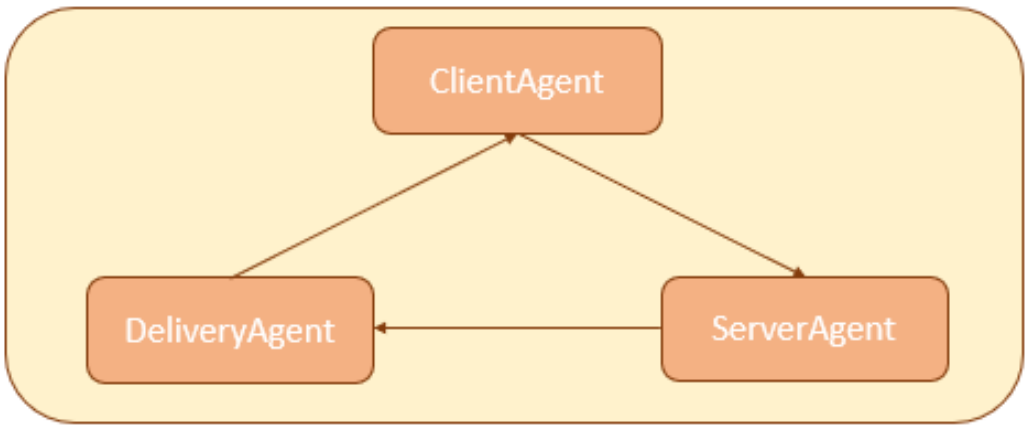



Figura 23. Modelo de conocimiento
Elaborado por: Carlos Andrés Herrera

### 2.6. Fase 6: Desarrollo de la aplicación

En esta etapa se repasa y explica tanto las herramientas que son utilizadas en la etapa de desarrollo del Sistema Multi Agente propuesto como la implementación del sistema.

**2.6.1. Requerimientos técnicos.** Los requerimientos tanto de hardware como de software que se utilizó en la implementación del proyecto se describen a continuación en la tabla 2:

Tabla 2. Requerimientos de hardware y software

| Dispositivo | Uso | Características |
|---|---|---|
| Computador portátil | Servidor / Desarrollo | **Hardware:**<br>Procesador Intel Core i7<br>Memoria 12 GB<br>Disco duro 1TB<br>**Software:**<br>Windows 10<br>Android Studio 4.0<br>JADE 4.5.0<br>Eclipse 2019-06 |
| Computador de escritorio | Contenedor Principal | **Hardware:**<br>Procesador AMD Athlon II<br>Memoria 6 GB<br>Disco duro 500 GB<br>**Software:**<br>Windows 10<br>JADE 4.5.0 |
| Smartphone | Cliente | **Hardware:**<br>Procesador Exynos 7884<br>Memoria 3 GB<br>Almacenamiento 32 GB<br>**Software:**<br>Android 10 |

Nota: esta tabla contiene las especificaciones de los dispositivos utilizados en el Sistema Multi Agente

**2.6.2. Ejecución de la plataforma.** Para la puesta en marcha del Sistema Multi Agente se debe considerar el orden en el que se ejecuta cada uno de los componentes, iniciando con la interfaz de JADE que será la instancia a la que los agentes se vincularán para poder interactuar, este componente se ejecuta desde un archivo .bat en que se incluye el siguiente script donde se especifica el puerto por el que se realizará la comunicación, en este caso el puerto 1706:

*java -cp jade.jar jade.Boot -gui -port 1706*

Una vez se ejecutó el archivo .bat se podrá visualizar la interfaz de JADE, que a la vez alberga al contenedor principal, como se ve en la figura 24.

Ejecución de la interfaz de JADE

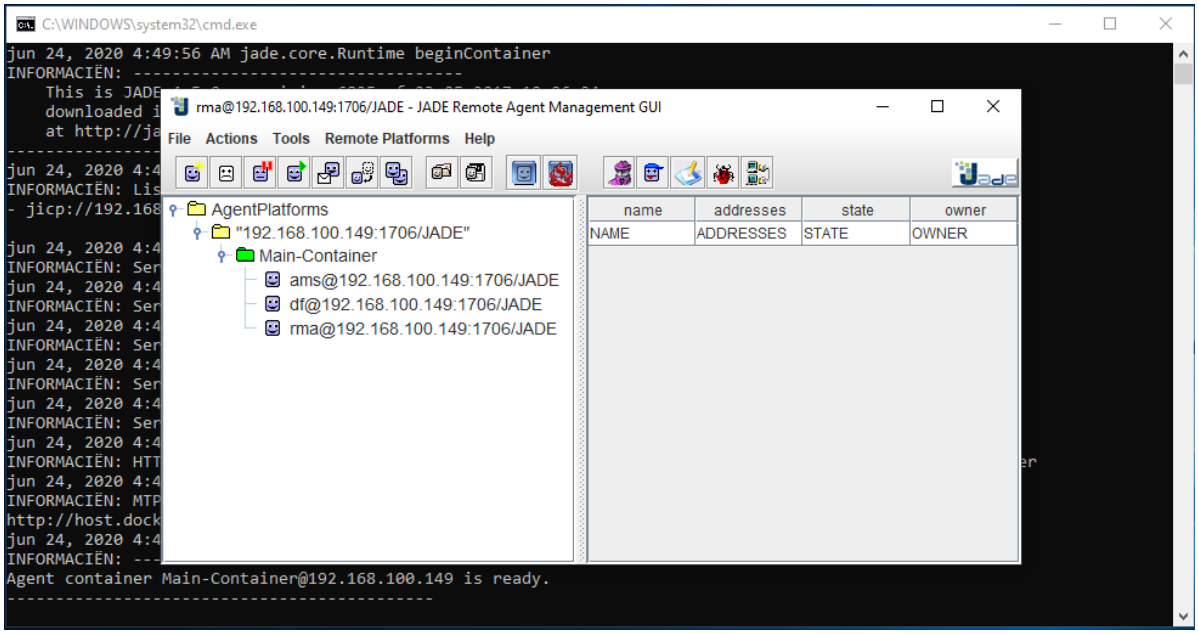



Figura 24. Ejecución de la interfaz de JADE
Elaborado por: Carlos Andrés Herrera

**2.6.3. Selección y captura de imágenes.** El aplicativo móvil al tener necesidad de acceder al almacenamiento interno y a la cámara, requiere de la habilitación de estos permisos en tiempo de ejecución, en la figura 25 se puede observar el código generado para esta actividad.

Método para permitir accesos en tiempo de ejecución

```
//BRINDA PERMISOS EN TIEMPO DE EJECUCIÓN PARA EL ALMACENAMIENRO INETRNO Y LA CAMARA
if(ContextCompat.checkSelfPermission( context: MainActivity.this, Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions( activity: MainActivity.this, new String[]{
            Manifest.permission.CAMERA, Manifest.permission.WRITE_EXTERNAL_STORAGE
    }, requestCode: 0);
}
```

Figura 25. Método para permitir accesos en tiempo de ejecución
Elaborado por: Carlos Andrés Herrera

El aplicativo instalado en el lado del cliente fue desarrollado en Android Studio, en la figura 26 se puede ver la interfaz del aplicativo que permite al usuario seleccionar las imágenes que desea binarizar desde dos orígenes, que son el almacenamiento interno (opción “Seleccionar Imagen”) o desde la captura de una fotografía mediante la cámara del dispositivo móvil (opción “Tomar Foto”).

Selección de imágenes


Figura 26. Selección de imágenes
Elaborado por: Carlos Andrés Herrera

Una vez seleccionadas las imágenes o capturadas las fotografías necesarias, se guardan en un arreglo de tipo mapa de bits.

**2.6.4. Selección de método de ejecución.** Para fines prácticos en el aplicativo móvil se implementó la selección del tipo de ejecución que desea realizar el usuario, de esta manera tenemos lo siguiente:

- **Local:** para la ejecución local, las n imágenes que seleccione el usuario se procesaran utilizando recursos netamente del dispositivo móvil, en este caso el tiempo de ejecución dependerá de la capacidad del mismo puesto que la binarización recorre el mapa de bits de cada una de las imágenes solicitadas mediante la creación en segundo plano de hilos de la clase Binarizacion.java para una ejecución en paralelo, y posteriormente guardarlas en un directorio específico de su almacenamiento interno.
- **Nube:** para el caso de la ejecución en la nube o remota, el dispositivo únicamente detecta las imágenes o capturas las fotografías según sea el caso, los datos de la IP de la plataforma a la que se va conectar son proporcionadas por el usuario mediante la interfaz del aplicativo; posteriormente crea el agente cliente a partir de la clase ClientAgent.java que hereda de la clase Agent contenida en la librería Jade.java proporcionada por Jade Tilab quienes son sus desarrolladores. El agente creado se conecta a la plataforma que debe ser ejecutada con anterioridad, una vez reconocido en el entorno de JADE, genera hilos asíncronos enviando los mapas de bits serializados por medio de un mensaje ACL con el protocolo FIPA específico para Sistemas Multi Agente hacia el agente Servidor (computador remoto) quien será el encargado de realizar el procesamiento de las imágenes.
- **Dividido:** al igual que el caso del envió a la nube, el procesamiento se realiza en el lado del servidor con la particularidad que en esta ocasión el agente Cliente realiza una validación previa donde toma cada mapa de bits de las imágenes o fotografías seleccionadas, calcula el producto de las dimensiones de la imagen y dependiendo de este resultado (que a su vez

determina las veces que se ejecutará el bucle for para la binarización de cada uno de los pixeles de la imagen) toma la decisión de ejecutar la binarización de forma local o en su efecto empieza a enviar solicitudes con las imágenes a través de mensajes hacia el Servidor, este umbral puede ser determinado por la variable limite, para el caso en desarrollo se ha considerado que la imágenes que se procesan en el Servidor son todas cuyo producto de las dimensiones es mayor a veinte millones. Cabe recalcar que mientras más pequeño es este límite mayor número de imágenes serán procesadas por el computador remoto y por ende se disminuye el uso de recursos locales del dispositivo móvil.

$$f(x) = \begin{cases} Local, & (x * y) > limite \\ Servidor, & (x * y) \leq limite \end{cases}$$

Donde:

$(x * y)$: es el producto de las dimensiones del mapa de bits de la imagen seleccionada.

límite: es la variable que define el máximo número de veces que se evaluará el bucle for de forma local, en este caso particular veinte millones.

Para los tres casos descritos anteriormente se utilizó el mismo algoritmo para la binarización de las imágenes, descrito en la sección 1.3.1, el bucle for utilizado se puede ver en la figura 27.

Algoritmo de binarización

```
finalImage = Bitmap.createBitmap(bitmap.getWidth(), bitmap.getHeight(), bitmap.getConfig());
int colorPixel;
int width = bitmap.getWidth();
int heigth = bitmap.getHeight();

for (int x = 0; x < width; x++) {
    for (int y = 0; y < heigth; y++) {
        colorPixel = bitmap.getPixel(x, y);
        int promedio = (Color.red(colorPixel) + Color.green(colorPixel) + Color.blue(colorPixel)) / 3;
        if (promedio < umbral)
            finalImage.setPixel(x, y, Color.rgb( red: 0,  green: 0,  blue: 0));
        else
            finalImage.setPixel(x, y, Color.rgb( red: 255,  green: 255,  blue: 255));
    }
}
```

Figura 27. Algoritmo de binarización
Elaborado por: Carlos Andrés Herrera

Cabe recalcar que al momento de generar el mensaje ACL desde el dispositivo móvil para el envió de cada una de las imágenes se genera su correspondiente mapa de bits para después generar una matriz de bytes que es el formato que puede ser enviado por mensajes entre agentes. Para esta transformación se utilizó una instancia de la clase *ByteArrayOutputStream* donde se almacenó el resultado de la compresión del mapa de bits de las imágenes mediante el método *compress* de la clase *Bitmap*, una de las características de este método es la de poder comprimir el mapa de bits, sin embargo se optó por colocar una calidad de 95 para no tener perdida de la información de la imagen y tampoco generar una desborde de memoria en la lectura del mapa de bits realizada por el dispositivo móvil, considerando que el margen de calidad puede ir desde 0 a 100, siendo 100 la mejor calidad para imágenes grandes y 0 la proporción utilizada para imágenes con menor gama de colores o imágenes pequeñas.

**2.6.5. Procesado en el Servidor.** El servidor fue desarrollado en Java Estándar en el IDE de Eclipse, para el cual se creó las clases:

- **ServerAgent:** esta clase hereda de Agent contenida en la librería Jade.java proporcionada por Jade Tilab quienes son sus desarrolladores, esta clase crea el agente Servidor y lo conecta con el contenedor principal, posterior a su registro en el entorno de JADE, se

mantiene a la espera de la recepción de solicitudes por parte de los agentes Cliente, mientras no exista una petición este agente pasa a un estado de suspensión para reducir el consumo de recursos del computador que lo contiene mediante el uso del método *doSuspend()*. Una vez detecta una solicitud (mediante mensajes ACL de FIPA) el ServerAgent se activa generando hilos asíncronos de la clase Binarización.java, esta clase contiene el código para procesar las imágenes que son receptadas con la particularidad del reuso de estos hilos con el fin de optimizar recursos en el servidor y no generar referencias innecesarias. La matriz de bytes que llega al servidor es leída mediante una instancia de la clase *BufferedImage* con el fin de poder manipular la información de sus pixeles y generar la nueva imagen binarizada, una vez ejecutado el proceso se encapsula nuevamente la imagen a una matriz de bytes para poder ser enviada mediante mensaje ACL.

- **DeliveryAgent:** este agente recepta los mensajes que vienen desde el ServerAgent y que contienen los datos del ClientAgent y la imagen binarizada para poder generar agentes individuales temporales y re direccionar las imágenes hacia el ClientAgent, su participación es de importancia ya que se encarga de recuperar las imágenes resultantes sin interrumpir la ejecución asíncrona de los hilos generados por el ServerAgent, brindando una mayor fluidez en el procesado de imágenes que ingresan a la plataforma sin generar cuellos de botella a la hora de responder las peticiones.

Para la ejecución del ServerAgent se debe considerar el ingreso de la IP del contener principal de la plataforma, así como el puerto de comunicación y el nombre del agente que se va crear, estos datos se incluyen como argumentos al momento de la ejecución.

*-host 192.168.100.149 -port 1706 -local-port 1706 -container -container-name Server*

Adicionalmente y para el mejor desempeño del servidor puesto a que va recibir n solicitudes, se habilita el uso de 10000 MB de memoria RAM como argumento en el apartado de Máquina Virtual (VM) con la siguiente línea:

*-server -Xmx10000m*

Otra de las líneas que se configura al momento de la ejecución es el tiempo de espera para la entrega de mensajes entre agentes, esto se ve reflejado en la siguiente línea ingresada como argumento en Eclipse:

*-jade_core_messaging_MessageManager_deliverytimethreshold 1000000000*

La parametrización mencionada anteriormente se puede observar en la figura 28.

Ejecución del agente Servidor (Eclipse)

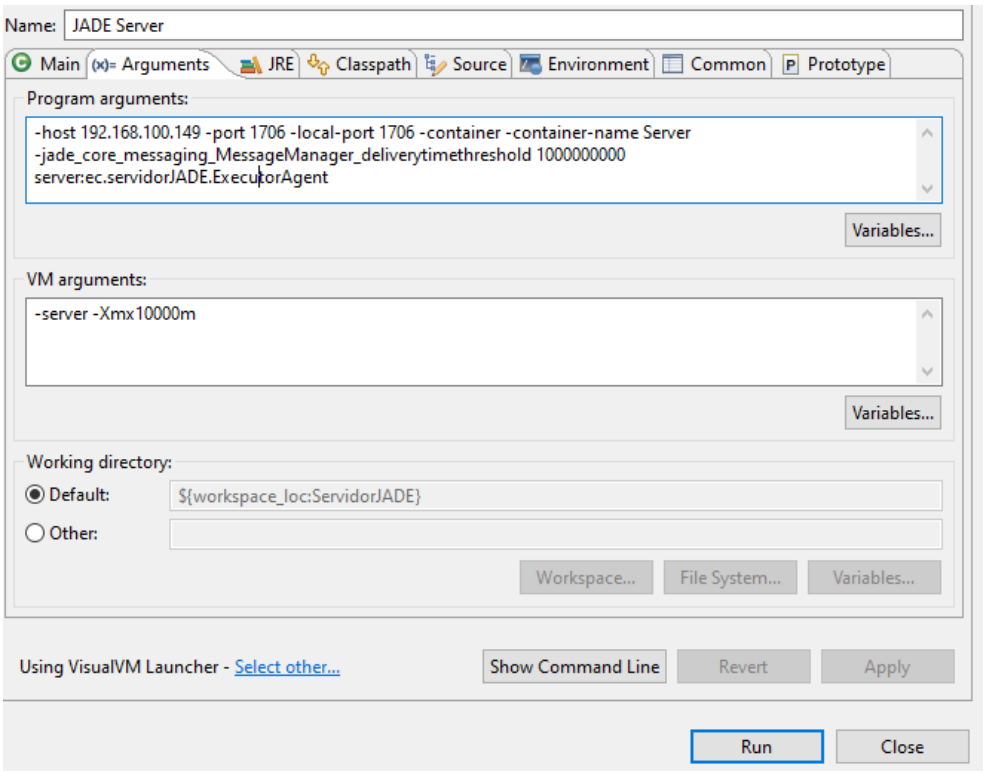



Figura 28. Ejecución del agente Servidor (Eclipse)
Elaborado por: Carlos Andrés Herrera

Mientras en la Interfaz de JADE se puede visualizar la integración del agente Server, en la consola se puede observar los mensajes que maneja la plataforma indicando la correcta inclusión del nuevo agente en la plataforma como se ve en la figura 29.

Ejecución del agente Servidor

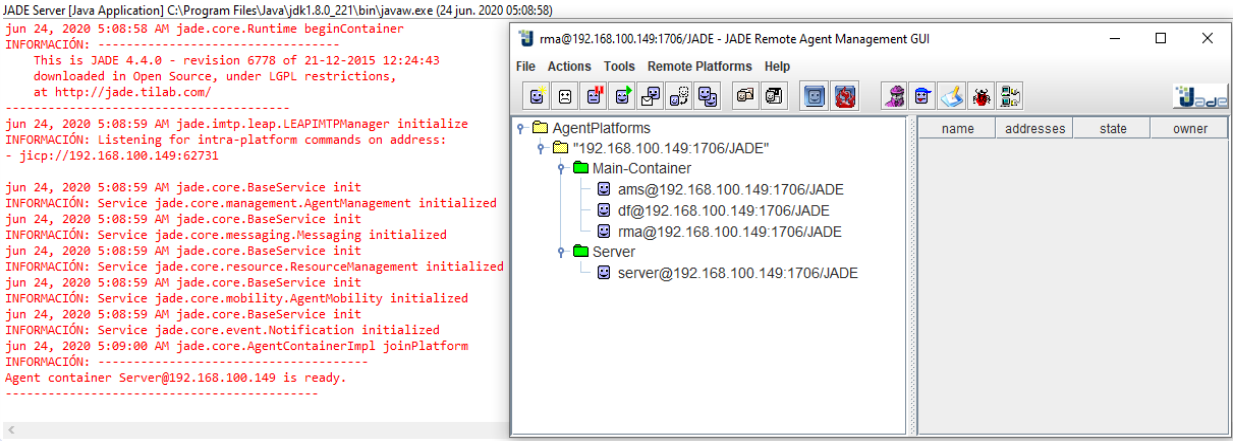



Figura 29. Ejecución del agente Servidor
Elaborado por: Carlos Andrés Herrera

**2.6.6. Envío de respuestas.** Posterior al procesado de las imágenes seleccionadas que fueron enviadas hacia el Servidor, se realiza la encapsulación de la imagen resultante en un mensaje ACL creando una instancia del tipo ACLMessage e instanciando los parámetros necesarios para poder generar tanto un nuevo agente del tipo Delivery (temporal) como su contenedor correspondiente, como se puede ver a continuación en la figura 30. Cuando el mensaje es enviado al Cliente, tanto el contenedor como el agente Delivery actual se eliminan del contenedor principal mediante el método *kill()*.

Generación de Agentes Delivery

```
ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
ImageIO.write(salida, "jpg", byteArrayOutputStream);
byteArray = byteArrayOutputStream.toByteArray();
ACLMessage envio = new ACLMessage(ACLMessage.INFORM);
envio.addReceiver(msg.getSender());
envio.setContentObject(byteArray);

jade.core.Runtime runtime = jade.core.Runtime.instance();
Profile profile = new ProfileImpl();
profile.setParameter(Profile.MAIN_HOST, "192.168.100.149");
profile.setParameter(Profile.MAIN_PORT, "1706");
ContainerController container = runtime.createAgentContainer(profile);
runtime.setCloseVM(true);
try {
    AgentController ag =    container.createNewAgent("Delivery-"+Thread.currentThread().getId(),
                            "ec.servidorJADE.DeliveryAgent",
                            new ACLMessage[] {envio});
} catch (StaleProxyException e) {
    e.printStackTrace();
}
```

Figura 30. Generación de Agentes Delivery
Elaborado por: Carlos Andrés Herrera

**2.6.7. Recepción de respuestas.** En el momento en que el ClientAgent termina de enviar las imágenes que serán procesadas en el Servidor, pasa a un estado de bloqueo, y se mantiene al pendiente de la recepción de las respuestas generadas por el Servidor a través de los Agentes Delivery. En cuanto un mensaje de respuesta llega al ClientAgent, este genera el mapa de bits correspondiente y almacena la imagen binarizada en el almacenamiento interno del dispositivo, esto lo realiza hasta que el número de imágenes guardadas por el agente sea igual al número de imágenes enviadas, posterior a esto el agente cliente se elimina del contenedor principal. El resultado final, serán imágenes binarizadas, como se puede ver en la figura 31, este resultado se lo puede ver en la carpeta denominada "salida" que es el path por defecto donde se almacenarán las imágenes procesadas.

Resultado de la imagen procesada

Figura 31. Resultado de la imagen procesada
Elaborado por: Carlos Andrés Herrera

La interacción que mantienen los agentes dentro de la plataforma se puede observar mediante la inclusión de un agente Sniffer, el resultado se puede observar a continuación en la figura 32.

Envío de mensajes en la plataforma

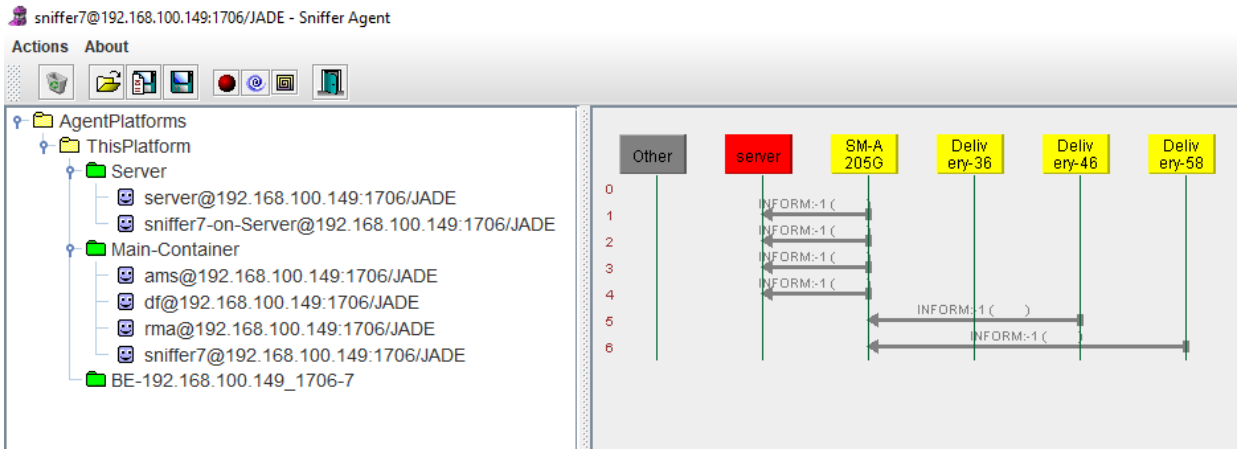



Figura 32. Envío de mensajes en la plataforma
Elaborado por: Carlos Andrés Herrera

# Capítulo 3

## Análisis de resultados

En el presente capítulo se mostrarán los resultados de la ejecución tanto de forma local en el dispositivo móvil, de forma remota en un computador que será parte de la plataforma y cuyo rol será la de servidor, así como la ejecución de forma dividida donde de acuerdo a un límite establecido por el programador (en este caso el límite son las matrices de 5000 pixeles x 4000 pixeles o equivalente) se definirá si el procesamiento se realiza con recursos del dispositivo móvil o si se envía al computador remoto de la plataforma.

En la tabla 3, se puede observar las imágenes utilizadas para las pruebas y sus características:

Tabla 3. Imágenes usadas en las pruebas

| Imagen | Tamaño (MB) | Dimensiones | Formato | Profundidad de bits |
|---|---|---|---|---|
| Image1 | 11.9 | 7680x4320 | JPG | 24 |
| Image2 | 10.9 | 10929x5553 | JPG | 24 |
| Image3 | 13.4 | 7680x4320 | JPG | 24 |
| Image4 | 12 | 7680x4320 | JPG | 24 |
| Image5 | 11.4 | 7680x4320 | JPG | 24 |
| Image6 | 1.17 | 1920x1080 | JPG | 24 |
| Image7 | 1.48 | 3840x2160 | JPG | 24 |
| Image8 | 0.25 | 1920x1200 | JPG | 24 |
| Image9 | 1.08 | 1920x1080 | JPG | 24 |
| Image10 | 1.65 | 3840x2160 | JPG | 24 |

Nota: esta tabla contiene las características de las imágenes usadas para las pruebas realizadas

### 3.1. Resultados

Para la medición de tiempo de ejecución, uso de memoria RAM y uso de CPU se ejecutó la totalidad de las imágenes descritas en la tabla 3; se realizó la ejecución de forma remota (barras color azul) usando los recursos netamente del dispositivo móvil; por otro, las barras ClientAgent

(barra color naranja) y ServerAgent (barra color plomo) representan la ejecución usando la técnica de *code offloading* en el Sistema Multi Agente, la medición se realizó de forma independiente para cada agente con el fin de identificar el uso de recursos en cada caso. Hay que considerar que el ClientAgent se encuentra en el dispositivo móvil y el ServerAgent en un computador remoto.

Por otro lado, para la medición de transferencia de datos se consideró agrupar las imágenes listadas en la Tabla 3 en 3 escenarios:

- **Escenario 1 (4 imágenes):** Image1, Image2, Image6, Image7.
- **Escenario 2 (8 imágenes):** Image1, Image2, Image3, Image4, Image6, Image7, Image8, Image9.
- **Escenario 3 (10 imágenes):** todas las imágenes.

Esta agrupación se la realizó con el fin de visualizar de manera más clara la transferencia que existe en cada escenario.

**3.1.1. Tiempo de ejecución total.** En la figura 33; se puede observar como el tiempo de ejecución es visiblemente mayor utilizando los recursos del dispositivo móvil sin ninguna técnica a diferencia en cuanto se lo realiza con la técnica de *code offloading* sobre el Sistema Multi Agente, esto debido a que las imágenes más grandes en relación a sus dimensiones son enviadas al ServerAgent que se encuentra en un computador remoto para su procesamiento, razón por la cual el tiempo de ejecución se divide entre el ClientAgent y el ServerAgent.

Comparativa de tiempo de ejecución

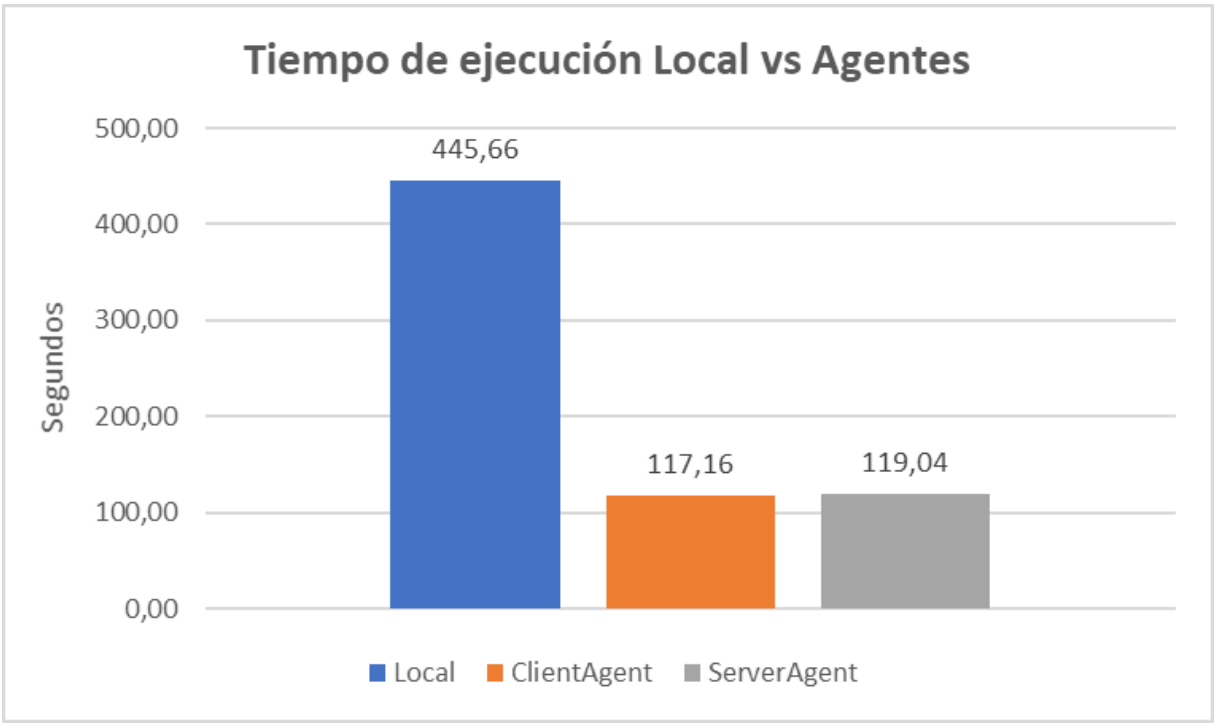



Figura 33. Comparativa de tiempo de ejecución
Elaborado por: Carlos Andrés Herrera

**3.1.2. Uso de memoria RAM.** En la figura 34; se puede observar que el uso de memoria RAM del dispositivo móvil en la ejecución local es extensa llegando incluso hasta dos tercios de su total, sin embargo con la ejecución utilizando técnicas de *code offloading*, la carga de procesamiento en el dispositivo móvil (ClientAgent) es muy baja puesto que las imágenes más grande en relación a sus dimensiones son enviadas al computador remoto (ServerAgent) que cuenta con mayor capacidad de hardware y es la razón por la que la carga de trabajo se encuentra en el lado remoto e incluso llegando a utilizar hasta un 16% de la memoria total del computador.

Comparativa de uso de memoria

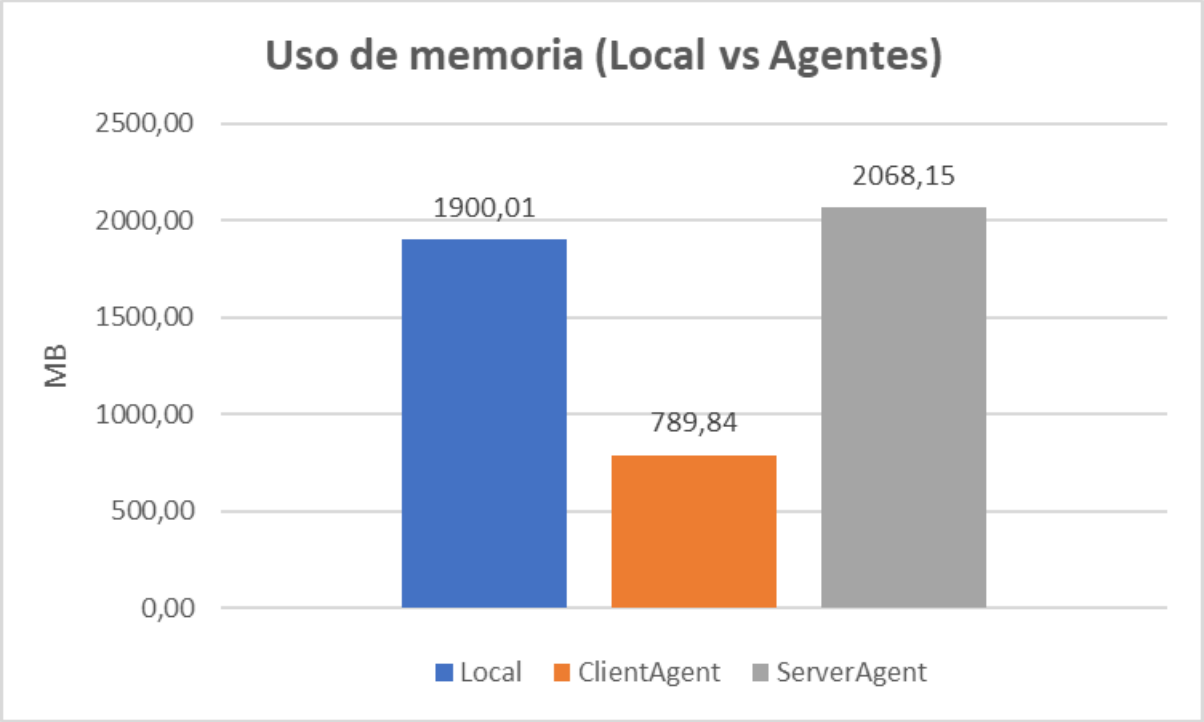



Figura 34. Comparativa de uso de memoria
Elaborado por: Carlos Andrés Herrera

**3.1.3. Uso de CPU.** En la figura 35; se puede observar que el uso de la CPU en el caso de la ejecución local con los recursos del dispositivo móvil sin utilizar ninguna técnica en particular se mantiene en 13%, sin embargo y con la ejecución utilizando técnicas de *code offloading* se puede observar como la carga de trabajo y por ende el porcentaje de uso de la CPU va equilibrándose entre el ClientAgent y el ServerAgent, considerando que las imágenes más grandes conforme a sus dimensiones se envían hacia el computador remoto para ser ejecutadas.

Uso de CPU

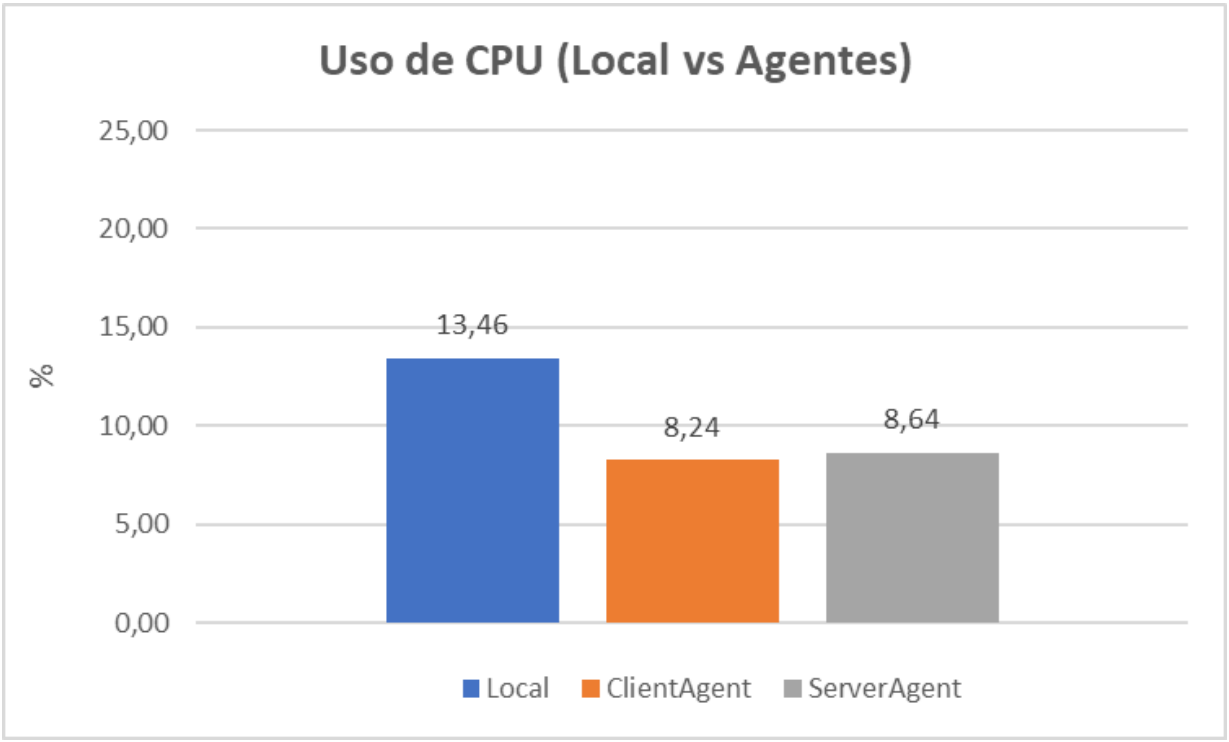



Figura 35. Uso de CPU
Elaborado por: Carlos Andrés Herrera

**3.1.4. Eficiencia del código.** En base a los resultados obtenidos de la herramienta SonarQube al revisar el código fuente del proyecto desarrollado en Android Studio (Cliente) detallados en la figura 36, se puede observar que cumple con las características de fiabilidad, seguridad y mantenibilidad razón por la cual el código se considera eficiente y sin vulnerabilidades.

Resultado de SonarQube (Proyecto Android Studio)

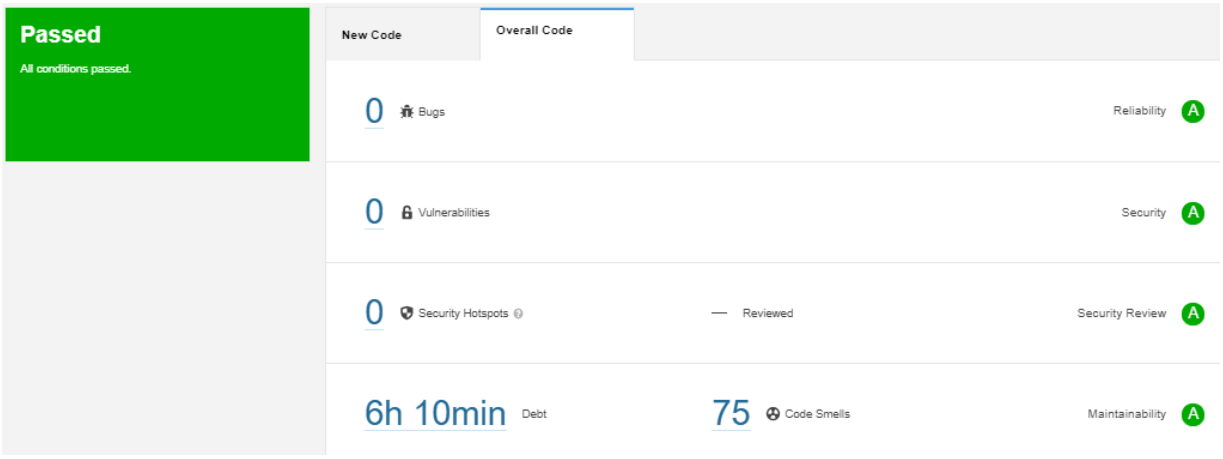



Figura 36. Resultado de SonarQube (Proyecto Android Studio)
Elaborado por: Carlos Andrés Herrera

De igual forma se realizó la revisión del código fuente del proyecto desarrollado en Eclipse (Servidor) para determinar la eficiencia del código; como se puede observar en la figura 37 el proyecto cumple con las condiciones necesarias para ser considerado eficiente y libre de vulnerabilidades.

Resultado de SonarQube (Proyecto Eclipse)

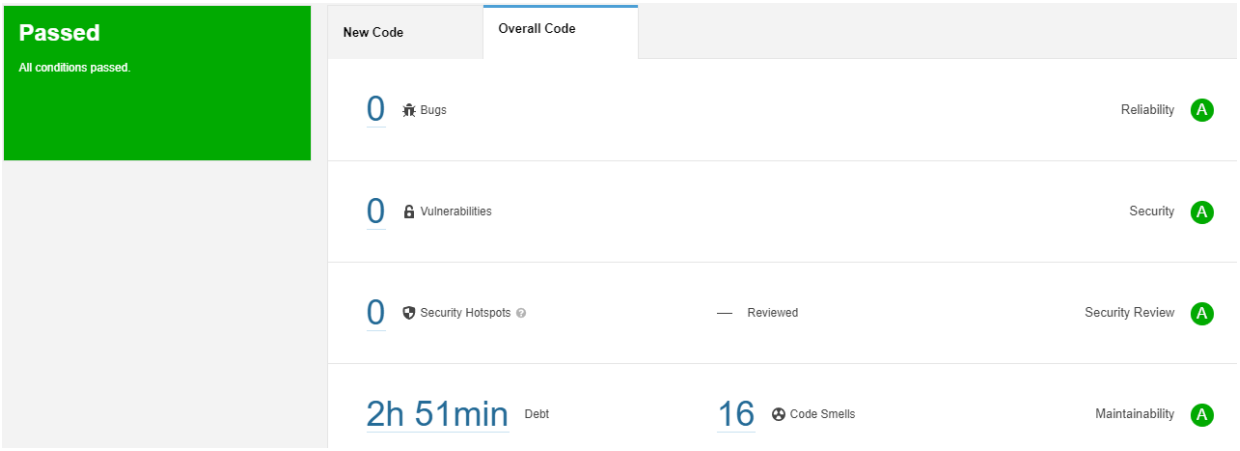
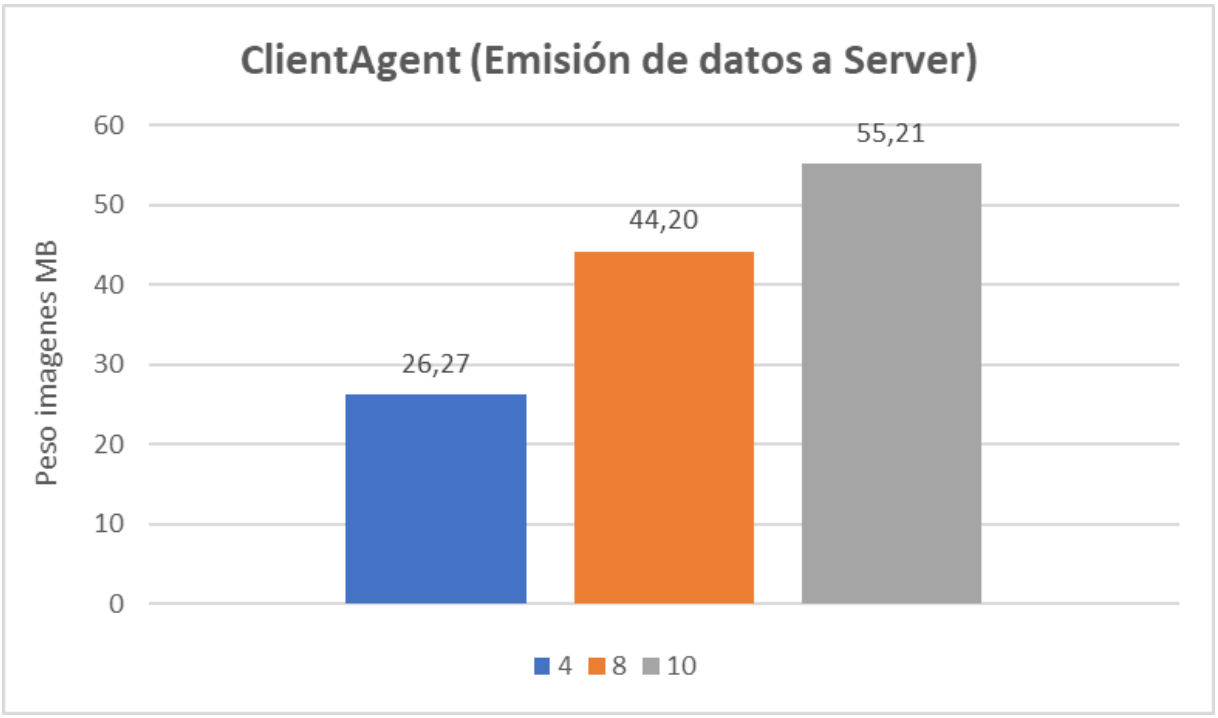


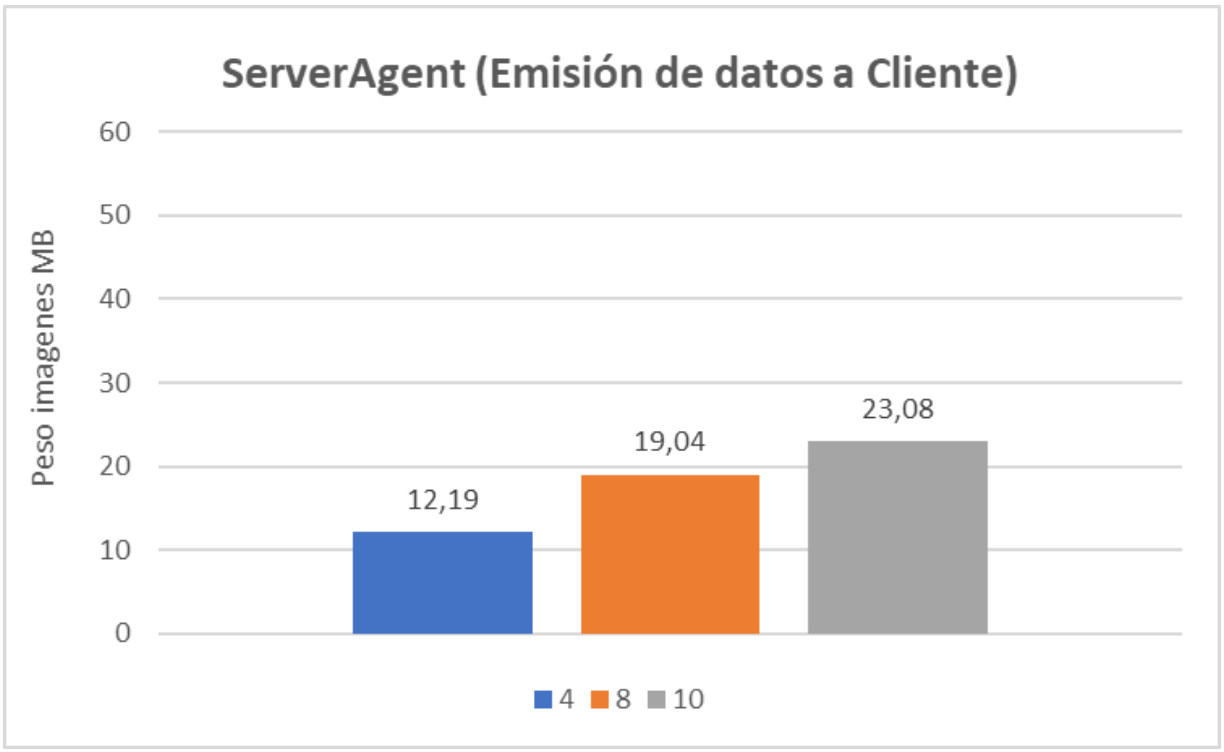

Figura 37. Resultado de SonarQube (Proyecto Eclipse)
Elaborado por: Carlos Andrés Herrera

**3.1.5. Transferencia de datos.** En la figura 38 se puede ver que en base a los datos obtenidos en la transferencia de información a través de la red en los tres escenarios planteados el tamaño de información que es enviada desde el ServerAgent es menor a la que se envía desde el ClientAgent, esto debido a la binarización de las imágenes ya que las imágenes enviadas desde el ClientAgent son imágenes a color, sin embargo las imágenes que se envían desde el ServerAgent son procesadas por lo tanto son imágenes en blanco y negro.

Transferencia de datos




(a)



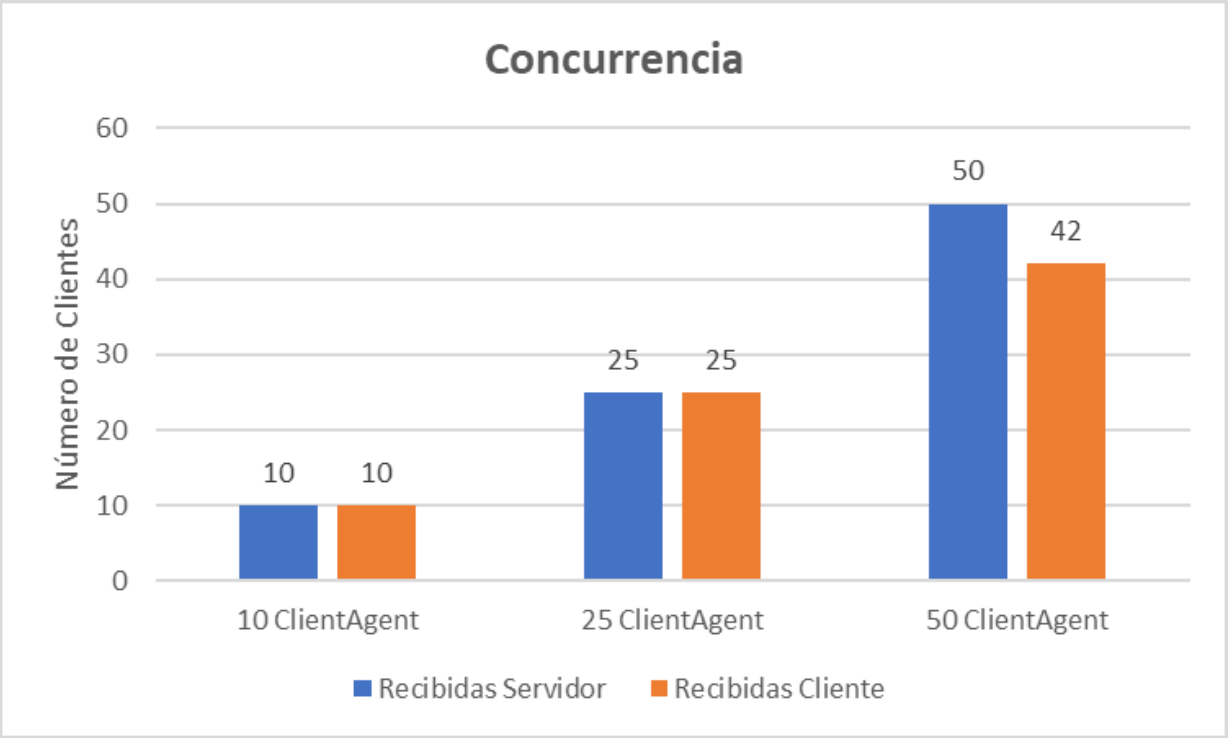

(b)

Figura 38. Transferencia de datos
Elaborado por: Carlos Andrés Herrera

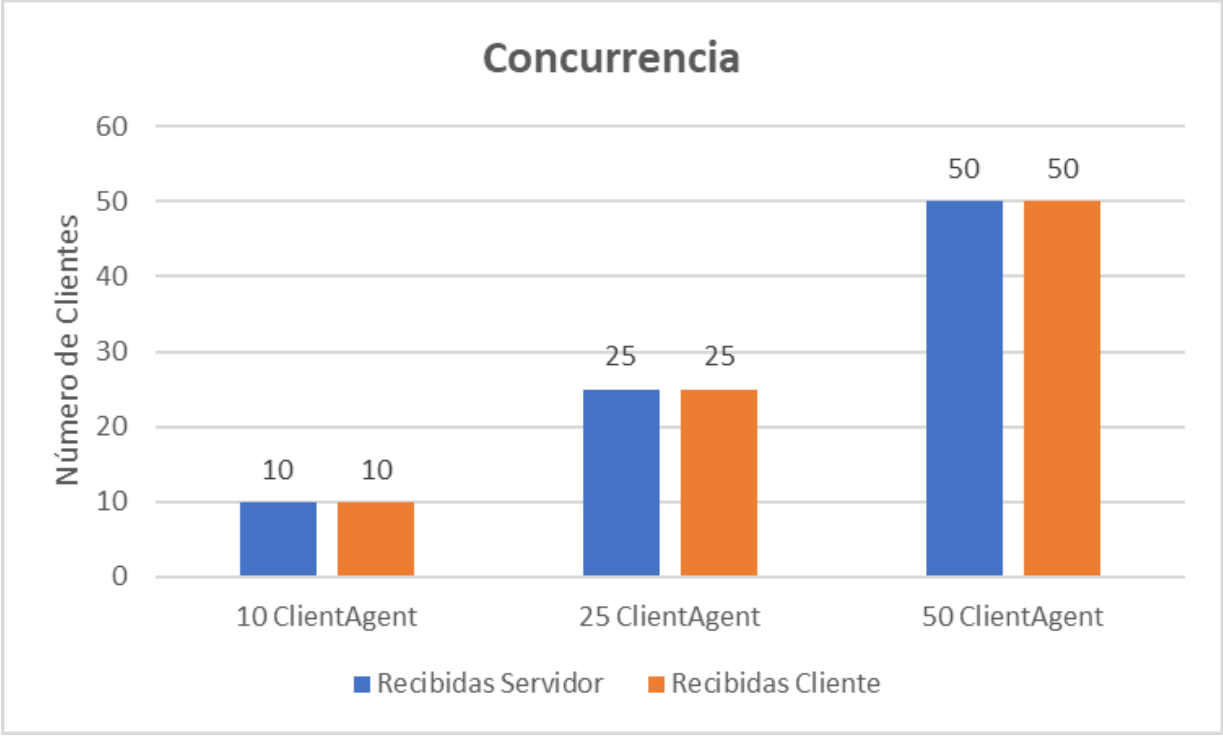
**3.1.6. Concurrencia.** Para medir la concurrencia que el ServerAgent recibe y su comportamiento en base a diferentes escenarios, se generó un ClientAgent en Eclipse quien es el encargado de crear varios Clientes, cada uno de ellos envió una imagen. En la figura 39a se puede observar como en los dos primeros escenarios se recibe el mismo número tanto en peticiones como en respuestas, sin embargo, en el tercer escenario donde se probó con 50 peticiones (ClientAgent) se puede observar cómo hasta el ServerAgent llega el mismo número, pero las respuestas al ClientAgent solamente llegan 42, esto debido a que la cola de mensajes crece por enviar crece y tiende a generar un desbordamiento del mecanismo de la entrega de mensajes.

Para poder solucionar el inconveniente del alto número de imágenes enviadas desde el ServerAgent, se aumentó la espera de 3 segundos antes de matar al contener con el agente que estaba realizando el envío. Con esta implementación se puede observar que en la figura 39b ya en los tres escenarios están llegando el mismo número de respuestas que las peticiones enviadas.

Concurrencia




(a)




(b)

Figura 39. Concurrencia
Elaborado por: Carlos Andrés Herrera

## CONCLUSIONES

- Construida y probada la distribución de la plataforma para agentes, se puede concluir que el uso de los mismos en un sistema conformado por varios hosts donde se encuentra contenidos, puede ser de mucha ayuda y beneficio para la ejecución de tareas o procesos que requieren una alta demanda de recursos en hardware en escenarios donde el usuario posee dispositivos con limitaciones de características en temas de CPU, memoria y batería, como son los smartphones, tablets, etc.; además de proporcionar mejores tiempos de respuesta que pueden reducirse hasta un 75%.
- El uso de una metodología especializada, como GAIA, para el desarrollo de Sistemas Multi Agente es fundamental y de gran soporte para la correcta identificación de los actores (agentes) y los roles (actividades) que desempeñará cada uno de ellos en la plataforma desarrollada, esto constituye una optimización de tiempo en las primeras etapas de desarrollo de software y como consiguiente la disminución de errores y correcciones en etapas futuras.
- JADE si bien es un marco de referencia para la creación de Sistemas Multi Agente, también se convierte en una herramienta que facilita de gran manera la inducción al manejo de la Programación Orientada Agentes y consigo la mejora en la implementación de sistemas colaborativos, considerando que una de las principales características de un agente es la de ejecutar tareas específicas dentro de una red o a su vez comunicarse con los diferentes host en busca de un agente que pueda ejecutar las tareas necesarias para cumplir con un objetivo general.
- Mediante el uso de un Sistema Multi Agente el usuario no solo puede tener acceso a la comunicación con los diferentes contenedores y agentes que son parte de la plataforma, sino que, al poder delegar tareas a uno o varios agentes, el esfuerzo en el procesamiento de datos se

distribuye entre los nodos que componen la plataforma y por ende se produce un ahorro en cuanto a utilización de recursos cuando los dispositivos utilizados no cuentan con las capacidades necesarias para poder realizar un procesamiento que demande características altas de hardware.

- El uso de técnicas de *code offloading* es uno de los fundamentos utilizados en este trabajo, ya que con la correcta implementación del Sistema Multi Agente logra distribuir la carga de procesamiento intensivo a diferentes agentes o dispositivos, en este caso el Cliente y el Servidor, proporcionando una mejor respuesta y optimización de recursos al no solo enviar todas las solicitudes al computador remoto, sino también apoyarse del Cliente y procesar imágenes localmente siempre y cuando sean de menor resolución.
- Con la distribución de las tareas de un sistema dentro de una red de computadores con mejores prestaciones que un dispositivo móvil es notable la mejora en cuanto a tiempos de ejecución y optimización de recursos en el dispositivo móvil bajando en promedio un 50% en el consumo de recursos móviles, puesto que su tarea con mayor demanda es la de definir qué elementos se ejecutan de forma remota y que se ejecuta de forma local, sin embargo también se puede observar que existen elementos que se suman a los recursos utilizados como por ejemplo la red, que para el caso del sistema desarrollado no tiene mayor efecto ya que el tiempo de ejecución en el servidor baja totalmente a comparación de una ejecución local y esto compensa la transferencia de datos hacia el servidor que ocurre en una ejecución remota.

## RECOMENDACIONES

- Para tener mejores resultados a la hora de implementar un Sistema Multi Agente es importante considerar las diferentes metodologías específicas para estos sistemas, especialmente tener gran énfasis en las etapas de análisis y diseño, puesto que el desarrollo e implementación serán mucho más llevaderas y con menor errores si se tiene una definición clara tanto de roles, actividades y protocolos.
- Al momento de recopilar información para presentar resultados es importante remitirse a la documentación oficial de Android, ya que existen métodos de cálculo recomendados en distintos foros para la obtención del uso de recursos, sin embargo, no tienen una fuente veraz como para considerarlos exactos.
- En relación a la optimización de recursos, se debe considerar las buenas prácticas de programación, como establecer como valores nulos las referencias a objetos que no se van a volver a utilizar en el aplicativo, dado que estas instancias pueden implicar un desborde del uso de memoria y por lo tanto la no respuesta del aplicativo y su posterior cierre.

## LISTA DE REFERENCIAS

**Artículos académicos**


Adolph, M. (07 de 2009). ITU International Telecommunication Union. Obtenido de Mobile Applications: https://www.itu.int/dms_pub/itu-t/oth/23/01/T230100000C0004PDFE.pdf

Ahvanooey, M., Li, Q., Rabbani, M., & Raza, A. (2017). A Survey on Smartphones Security: Software Vulnerabilities, Malware, and Attacks. Nankín: (IJACSA) International Journal of Advanced Computer Science and Applications.

Alonso, A., Ferreira, I., Álvarez, M., & García, R. (2011). Dispositivos móviles. Oviedo: E.P.S.I.G : Ingeniería de Telecomunicación Universidad de Oviedo. Obtenido de Dispositivos móviles.

Báez, M., Borrego, Á., Cordero, J., Crus, L., Gonzalez, M., Hernandez, F., . . . Rodríguez, J. S. (2012). academia. Madrid: E.M.E. Obtenido de Introducción a Android.

Chun, B G., Ihm, S., Maniatis, P., & Naik, M. (2010). CloneCloud: Boosting Mobile Device Applications Through Cloud Clone Execution. California.

Cuervo, E., Balasubramanian, A., Cho, D k., Wolman, A., Saroiu, S., Chandra, R., & Bahl, P. (2010). MAUI: Making Smartphones Last Longer with Code Offload. Massachusetts: 8th international conference on Mobile systems, applications, and services.

Khazer, M., & Rather, S. (2019). Impact of Smartphones on Young Generation. Obtenido de https://www.researchgate.net/profile/Mudasir_Rather2/publication/333104023_Impact_of_Smartphones_on_Young_Generation/links/5cfe8a6c92851c874c5d7e0d/Impact-of-Smartphones-on-Young-Generation.pdf

Lee, H., & Cheung, W. (1987). An empirical study of software metrics. IEEE Transactions on Software Engineering.

Magro, R. (2013). Binarización de imágenes digitales y su algoritmia como herramienta aplicada a la ilustración entomológica. Boletín de la Sociedad Entomológica Aragonesa.

Niroshinie, F., Seng, L., & Wenny, R. (2012). Mobile cloud computing: A survey. Elsevier, 1-23.

Wooldridge, M. (2002). An introduction MultiAgent Systems. Chichester: John Wiley & Sons.

Wooldridge, M., Jennings, N., & Kinny, D. (2000). The Gaia Methodology for Agent-Oriented Analysis and Design. Kluwer Academic Publisher.

Huang, D. (2011). Mobile Cloud Computing. Arizona.

Oliveira, R., da Silva, N., Machry, M., & Coelho, T. (2017). A transparent code offloading technique for Android devices. Porto Alegre: Pontifical Catholic University of Rio Grande do Sul (PUCRS).

Shaukat, A., Shah, K., Saeed, M., & Azhar, R. (Feberro de 2015). Sensors and Mobile Phones: Evolution and State-of-the-Art. Peshawar: Pakistan journal of science. Obtenido de Sensors and Mobile Phones: Evolution and State-of-the-Art.

**Bibliografía**

Bellifemine, F., Caire, G., & Greenwood, D. (2007). Developing Multi-Agent Systems with JADE. Chichester: John Wiley & Sons.

Flores, H., Hui, P., Tarkoma, S., Li, Y., Srirama, S., & Buyya, R. (2015). Mobile Code Offloading: From Concept to Practice and Beyond. IEEE Communications Magazine, 1-9.

**Conferencias**

Bedi, K. (2014). Tablet PC & smartphone uses in education. 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO).

Delía, L., Galdamez, N., Thomas, P., & Pesado, P. (2013). Un Análisis Experimental de Tipo de Aplicaciones para Dispositivos Móviles. XVIII Congreso Argentino de Ciencias de la Computación.

Gordon, M., Jamshidi, A., Mahlke, S., Mao, M., & Chen, X. (2012). COMET: Code Offload by Migrating Execution Transparently. Michigan: 10th USENIX Symposium on Operating Systems Design and Implementation.

Qi, H., & Gani, A. (2012). Research on mobile cloud computing: review, trend and perspectives. Bangkok: Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP).

Qiwei, H., & Cho, D. (2016). Characterizing the Technological Evolution of Smartphones: Insights from Performance Benchmarks. Pittsburgh: 18th Annual International Conference on Electronic Commerce: e-Commerce in Smart connected World. Obtenido

de Characterizing the Technological Evolution of Smartphones: Insights from Performance Benchmarks.

**Documentos de sitios Web**

De Lucas, G. (2013). Camára de Comercio de España. Obtenido de Evolución de las aplicaciones para móviles: https://empresarias.camara.es/estaticos/upload/0/007/7438.pdf

Foundation for Intelligent Physical Agents. (03 de 12 de 2002). FIPA ORG. Obtenido de FIPA ACL Message Structure Specification: http://www.fipa.org/specs/fipa00061/SC00061G.pdf

Garamendi, J. F. (2004). ADIMEN. Obtenido de Introducción a JADE Java Agent DEvelopment: https://adimen.si.ehu.es/~rigau/teaching/EHU/TAIA/Curs2013-2014/JADE/JADE.pdf

INEC. (12 de 2017). Tecnologías de la Información y Comunicación. Obtenido de https://www.ecuadorencifras.gob.ec/documentos/web-inec/Estadisticas_Sociales/TIC/2017/Tics%202017_270718.pdf

Leal, I., Suarez, J., Guerrero, C., & Gutierrez, L. (2018). Caracterización de las metodologías INGENIAS y GAIA en las etapas de análisis y diseño. Espacios, 1-10.

Samadhiya, D., Wang, S H., & Chen, D. (2010). Quality Models: Role and Value in Software Engineering. 2nd International Conference on Software Technology and Engineering(ICSTE).

Satyanarayanan, M. (2010). Mobile Computing: the Next Decade. Pittsburgh: School of Computer Science.

Sofer, N. (2020). NirSoft. Obtenido de LiveTcpUdpWatch: https://www.nirsoft.net/utils/live_tcp_udp_watch.html

SonarSource S.A. (2020). SonarQube Docs. Obtenido de SonarQube Docs: https://docs.sonarqube.org/latest/

TILAB. (2020). JADE TILAB. Obtenido de https://jade.tilab.com/

**Tesis/trabajos de titulación**

Avancini, H. (2000). FraMaS: Un Framework para Sistemas Multi-agente basado en Composición. UNIVERSIDAD NACIONAL DEL CENTRO DE LA PROVINCIA DE BUENOS AIRES. Tandil: Universidad Nacional del Centro de la Provincia de Buenos Aires. Obtenido de http://www.nmis.isti.cnr.it/avancini/FraMaS_Tesis.pdf

Navarrete, E. (2008). Desarrollo de una agenda multiagente orientada a la web para la gestión de reuniones utilizando la plataforma JADE - Java en la Escuela Politécnica del Ejército sede Latacunga. Latacunga: Escuela Politécnica del Ejército.