Universidad Nacional de Colombia
Facultad de Ingeniería

Arquitectura Orientada a Servicios para la distribución y presentación eficiente de imágenes de alta definición en dispositivos móviles

Germán Corredor Prada
Maestría en Ingeniería de Sistemas y Computación

Director:
Ph.D., Eduardo Romero Castro
Departamento de imágenes diagnósticas
Facultad de Medicina

Co-Directora:
Ph.D., Marcela Iregui Guerrero
Ingeniería Multimedia
Universidad Militar Nueva Granada

Bogotá D.C., 18 de septiembre de 2012

A mis padres y mi hermano que siempre me apoyaron y creyeron en mí

Agradecimientos

Inicialmente quiero agradecer a mi director de tesis, el profesor Eduardo Romero por su apoyo, colaboración y tiempo. A mi co-directora, la profesora Marcela Iregui por su confianza, paciencia, dedicación, enseñanzas y constante apoyo, pues todo ello ha contribuido significativamente en mi crecimiento profesional y personal.

También quiero manifestar un agradecimiento especial a Francisco Gómez, por su inmensa confianza, porque siempre estuvo pendiente de este proceso, y porque me ayudó a clarificar mi horizonte y muchas ideas de gran relevancia para el desarrollo de este proyecto.

A la Universidad Militar Nueva Granada, que financió el proyecto. Al grupo de investigación GIM y en especial al profesor Wilson Sarmiento, por su colaboración para resolver diferentes dudas e inconvenientes, a Diego Navarro con quien trabajé por algunos meses y estuvo dispuesto a colaborarme en todo momento, y a Sylvia Hernández con quien hemos trabajado de la mano en el desarrollo de la interfaz gráfica para el sistema.

Al grupo de investigación Bioingenium de la Universidad Nacional, por su calurosa acogida y por su valiosa compañía y apoyo. Quiero agradecer especialmente a Daniel Martínez, con quien trabajé de la mano para la implementación de los servicios web de la aplicación, y asimismo a Andrea Rueda, Alexánder Pinzón, Diana Marín y Ricardo Gutiérrez quienes me colaboraron a lo largo del proceso con sus comentarios, sugerencias y explicaciones.

A mis amigos y colegas David Eljach, Diana García y Camilo Jiménez, quienes dieron importantes aportes para enriquecer esta tesis.

A mis padres y hermano por sus enseñanzas de perseverancia y trabajo, y porque siempre me acompañaron a lo largo de mi carrera universitaria, brindándome su apoyo y voto de confianza.

Por último, agradezco a todas las personas que me acompañaron durante este proceso directa o indirectamente y que estuvieron dispuestas a apoyarme cuando lo necesité.

Contenido

1. Introducción **1**

2. Contextualización **6**
2.1. Introducción a JPEG2000 y JPIP 6
2.1.1. JPEG2000 6
2.1.2. JPIP 9
2.2. Arquitecturas Orientadas a Servicios 10
2.2.1. Servicios y servicios web 11
2.2.2. Arquitectura básica SOA 12
2.3. Estado del arte 13
2.3.1. Acceso a contenido multimedia 13
2.3.2. Arquitecturas orientadas a servicios en móviles 15

3. Sistema de visualización y distribución de imágenes **18**
3.1. Descripción general del sistema 18
3.2. Alcance 19
3.3. Cubrimiento de requerimientos 20
3.3.1. Casos de uso 20
3.3.2. Actores e interesados en el sistema 22
3.3.3. Diagrama de clases 22
3.4. Atributos de calidad 24
3.4.1. Flexibilidad 25
3.4.2. Eficacia 25
3.4.3. Usabilidad 25
3.4.4. Interoperabilidad 26
3.4.5. Desempeño 26
3.5. Diagrama de arquitectura 27
3.5.1. Capa de Almacenamiento 28
3.5.2. Capa de provisión de servicios 30

3.5.3. Capa cliente 33
3.6. Detalles de la implementación 36

4. Modelo para visualización de imágenes de alta definición en dispositivos móviles 43
4.1. Introduction 44
4.2. JPEG2000 Overview 46
4.3. System Architecture 46
4.3.1. Server 48
4.3.2. Client 49
4.4. System for streaming and visualization of images 50
4.4.1. Server Implementation 50
4.4.2. Client Implementation 50
4.5. Experimental results 51
4.5.1. Comparing JPEG and JPEG2000 52
4.5.2. Simulating a User Browsing Protocol 52
4.6. Conclusions 56

5. Arquitectura Orientada a Servicios para la distribución y visualización de imágenes de alta resolución en dispositivos móviles 57
5.1. Introducción 58
5.2. Arquitectura del sistema 61
5.2.1. Almacenamiento 64
5.2.2. Acceso a datos 64
5.2.3. Cliente móvil 65
5.2.4. Administrador del sistema 66
5.3. Sistema para la transmisión y visualización de imágenes de gran tamaño 66
5.3.1. Implementación del almacenamiento 66
5.3.2. Implementación del acceso a datos 66
5.3.3. Implementación del Cliente móvil 67
5.4. Resultados experimentales 68
5.4.1. Experimento 1. Eficiencia de decodificación y transmisión: JPEG vs.JPEG2000 69
5.4.2. Experimento 2. Tiempos de respuesta para regiones de diferente tamaño 70
5.4.3. Experimento 3. Visualización de una región variando el porcentaje de calidad 71

5.4.4. Experimento 4. Visualización de áreas contiguas y solapadas con mismo nivel de resolución 72
5.5. Conclusiones 75

6. Conclusiones y discusión **76**
6.1. Trabajo futuro 77

Bibliografía **79**

Lista de Figuras

2.1. Algoritmo de codificación J2K. 7
2.2. Organización del flujo de datos codificados J2K 8
2.3. Interacción Cliente-Servidor en JPIP 10
2.4. Cómo opera un servicio 12
2.5. Arquitectura básica SOA 13

3.1. Diagrama de casos de uso del sistema de visualización. 22
3.2. Diagrama de clases del sistema. 24
3.3. Diagrama de arquitectura para el sistema de visualización. . . 28
3.4. Estructura del archivo de índice. 30
3.5. Ingreso al sistema. 36
3.6. Selección de una imagen. 37
3.7. Solicitud de una región sin haber realizado solicitudes previas. 38
3.8. Solicitud de una región que necesita paquetes del servidor y previamente solicitados. 39
3.9. Solicitud de una región que sólo necesita paquetes previamente solicitados. 39
3.10. Solicitud de una región previamente solicitada almacenada en memoria gráfica. 40
3.11. Adición de nueva imagen al sistema. 40
3.12. Solicitud de una imagen con diferentes porcentajes de calidad a un mismo nivel de resolución. 41
3.13. Solicitud de una imagen satelital especificando diferentes niveles de resolución. 42

4.1. Top-level runtime view of the architecture. 48
4.2. Representation of the requested regions for a browsing protocol. 54
4.3. Variation of transmission and decoding times for a browsing protocol. 55

4.4. Evolution of memory usage of the application for a browsing protocol. 56

5.1. Variación del tiempo de transmisión y decodificación para solicitudes de áreas de diferente tamaño. 71
5.2. Variación del tiempo de transmisión y decodificación para solicitudes de un área con aumento progresivo de calidad. . . 72
5.3. Variación del tiempo de transmisión y decodificación para solicitudes de áreas contiguas y solapadas con un mismo nivel de resolución. 73
5.4. Variación del tiempo de transmisión y decodificación para solicitudes de áreas contiguas y solapadas con un mismo nivel de resolución. 74

Lista de Tablas

3.1. Descripción de los casos de uso del sistema. 21
3.2. Interesados en el sistema. 23
3.3. Servicios web provistos por la aplicación. 32

4.1. Bytes and decoding and computing times for a satellite image by using the JPEG standard. 53
4.2. Bytes and decoding and computing times for a satellite image by using the JPEG2000 standard. 53

5.1. Parámetros de codificación. 68
5.2. Resolución y tamaño para el JPEG piramidal. 69

Resumen

La utilización de dispositivos y servicios móviles ha aumentado significativamente en los últimos años gracias a las múltiples posibilidades que ofrecen. Actualmente, los dispositivos móviles se están utilizando para diferentes propósitos como la organización de datos, y acceso a información y contenido multimedia. Sin embargo, la visualización e interacción con contenido multimedia representan tareas complejas de realizar debido al tamaño de los datos y las limitaciones de tales dispositivos en términos de ancho de banda, espacio de almacenamiento, memoria, capacidad de procesamiento y tamaño de la pantalla. Por tanto, es necesario adaptar el contenido para permitir al usuario acceder e interactuar con éste de manera apropiada. Por otra parte, el escenario actual presenta una gran heterogeneidad a nivel de plataformas y sistemas operativos, por lo cual, es necesario definir mecanismos que permitan a las aplicaciones intercambiar datos con diferentes sistemas y soportar diferentes canales de acceso. En este trabajo se propone una arquitectura orientada a servicios para la visualización y distribución de imágenes remotas de alta definición en dispositivos móviles de manera flexible, eficaz, eficiente e interoperable, utilizando el estándar JPEG2000. Gracias a la granularidad de JPEG2000, a partir de un conjunto de peticiones de un usuario, es posible realizar una decodificación progresiva del contenido solicitado. Los resultados experimentales demuestran que la arquitectura presentada es eficaz para la transmisión y la visualización de grandes volúmenes de datos, a la vez que es eficiente en la utilización de recursos, presentando un buen rendimiento para los procesos de transmisión y decodificación, permitiendo una interacción simple y dinámica entre el usuario y el contenido.

Palabras clave: Arquitectura orientada a servicios, dispositivos móviles, decodificación, distribución, imágenes, JPEG2000, multimedia, visualización.

Abstract

In recent years, the use of mobile devices and services have largely increased because of the multiple possibilities they offer. Currently, mobile devices are being used for several purposes such as data organization, and information and multimedia access. However, data visualization and interaction with multimedia content are challenging tasks because of the content size and the limitations of such devices in terms of bandwidth, storage capacity, memory, CPU power and display size. Therefore, it is necessary to adapt the content in order to allow the user properly access and interact with that. Moreover, the current scenario is very heterogeneous at the level of platforms and operating systems, so it is necessary to define mechanisms that allow applications to exchange data with different systems and to support different access channels. In this work, it is proposed a service oriented architecture for streaming and visualization of remote high definition images on mobile devices in a flexible, effective, efficient and interoperable way, by using the JPEG2000 standard. By taking advantage of the JPEG2000 granularity, this model achieves to split data processing into different layers; in this way, from a set of requests from a client, it is possible to perform a progressive decoding of the requested content. Experimental results show that the introduced architecture is effective for transmission and visualization of large data volumes, while presents good performance, both for transmission and decoding processes, allowing a simple and dynamic interaction between user and images.

Keywords: Decoding, images, JPEG2000, Mobile devices, multimedia, Service Oriented Architecture, streaming, visualization.

Capítulo 1

Introducción

El acceso a la información en tiempo real ha cobrado gran importancia para el desarrollo de diversas actividades que realizan las personas de manera cotidiana. Esta es una de las razones por las cuales se ha masificado el uso de dispositivos y servicios móviles. Gracias a la portabilidad que éstos ofrecen, junto a los importantes avances tecnológicos que han experimentado, se ha posibilitado el desarrollo de aplicaciones más avanzadas, que permiten a las personas el manejo de información personal, acceso a información remota, interacción con servicios multimedia, entre otros. La importancia de los dispositivos móviles ha trascendido al punto que están siendo utilizados no sólo para aplicaciones de uso personal, sino también en áreas como la educación, ciencia, entretenimiento, comercio, industria, entre otras.

La creciente demanda de dispositivos móviles, ha incentivado al desarrollo de equipos cada vez más sofisticados, que cuentan con mayores capacidades de memoria y procesamiento, lo que ha extendido su alcance y posibilidades, tales como teléfonos inteligentes o "smartphones", tabletas o "tablet PCs", PDAs, cámaras digitales, consolas de videojuegos portátiles, entre otros. Dado que, una considerable parte de la información disponible se presenta como contenido visual, es imprescindible poder acceder a dicho contenido en una forma simple y rápida [8]. Sin embargo, a pesar de las capacidades con las que cuentan estos dispositivos, aún persisten algunas limitaciones externas, como las redes de comunicación y el acceso geográfico, y otras internas como la memoria, capacidad de procesamiento, espacio en disco, tamaño de la pantalla, y carga de batería del dispositivo [8, 18, 38]. Por esta razón, el acceso a contenido multimedia remoto de alta definición,

como imágenes y vídeos, representa una tarea compleja de realizar, pues este contenido, al poseer un gran volumen de datos, debe adaptarse para poder ser transmitido y procesado, de modo que pueda ser presentado al usuario de manera apropiada.

En respuesta a la necesidad de distribución y visualización de imágenes, se han desarrollado diferentes algoritmos de transmisión y compresión de contenido para facilitar el almacenamiento, distribución y presentación de contenido gráfico. El estándar JPEG ha sido utilizado por un largo tiempo para presentar imágenes codificadas, sin embargo, las necesidades actuales requieren la utilización de estándares de compresión que ofrezcan una mayor flexibilidad y rendimiento [40]. En el año 2000, el grupo JPEG (Joint Photographic Experts Group) publicó el estándar JPEG2000 (J2K), el cual está fundamentado en el concepto de regiones de interés y niveles de detalle; asimismo, está optimizado para la eficiencia, escalabilidad e interoperabilidad en redes y ambientes móviles [36]. Trabajos anteriores han demostrado que se pueden ahorrar muchos recursos mediante una adecuada comunicación de imágenes utilizando el estándar J2K [36], por lo cual, éste se presenta como una herramienta adecuada para abordar el problema de distribución y visualización de contenido en dispositivos móviles.

Diferentes trabajos han explorado las posibilidades de acceder a imágenes J2K mediante la optimización de la navegación de contenidos, proponiendo protocolos de distribución, manejo de plataformas cliente/servidor, estrategias de caché y técnicas de precarga, etc. [1,5,14,22–24,26,48]. Sin embargo, en dichos trabajos no se han considerado entornos móviles, donde existe una gran limitación de recursos computacionales. En el campo de los dispositivos móviles, los estudios se han enfocado en áreas como la tolerancia y recuperación de fallos en el envío sobre ambientes con ruido [15], la seguridad de la entrega de contenido [49] y la presentación de imágenes para pantallas con dimensiones reducidas [8, 9, 19, 38]. Sin embargo, es muy poco lo que se ha trabajado en cuanto al acceso y presentación eficiente de imágenes de alta definición (caracterizadas por poseer muy alta resolución, tamaño y calidad) en dispositivos móviles, teniendo en cuenta las restricciones de memoria y procesador del dispositivo, así como el ancho de banda del canal. Considerando que las tareas de transmisión, visualización e interacción con imágenes son bastante exigentes en términos computacionales, es necesario desarrollar herramientas adaptadas para satisfacer los requerimientos de los usuarios y superar las limitaciones que impone la navegación de imágenes de gran tamaño en dispositivos móviles. Para lograr esto, es necesario una

representación granular de la información, un uso óptimo ancho de banda del canal y un uso eficiente de los recursos de procesamiento.

Por otra parte, en el escenario actual, hay una clara tendencia hacia la estandarización y hacia el diseño de sistemas que estén en capacidad de comunicarse con otros sistemas, independientemente de la plataforma de desarrollo, sistema operativo o hardware. Esto cobra una gran importancia en el escenario de los dispositivos móviles, pues existe una gran heterogeneidad de especificaciones y de plataformas, y dado que, como se mencionó anteriormente, gracias a la portabilidad y sus capacidades, están siendo utilizados para el desarrollo de procesos críticos y de gran importancia en diversas áreas [28]. Por este motivo, adquiere relevancia la utilización de arquitecturas orientadas a servicios (SOA), las cuales promueven flexibilidad e interoperabilidad, y de servicios web, que proveen mecanismos estándar para la comunicación entre aplicaciones heterogéneas y operan sobre internet [33]. Sin embargo, la combinación de las características de SOA y las aplicaciones móviles presenta retos importantes, pues al favorecer la interoperabilidad, se ve comprometido el rendimiento del sistema, lo cual es crítico tratándose de dispositivos con capacidades limitadas [2]. En estos casos, es fundamental que se realice un buen diseño, donde optimice el envío y procesamiento de los datos, de modo que se pueda acceder e interactuar con el contenido con la calidad adecuada [28].

El presente proyecto propone una arquitectura orientada a servicios, de propósito general, que permite el fácil acceso y el envío óptimo de imágenes remotas de gran tamaño, permitiendo al usuario visualizar contenido gráfico de una manera simple y dinámica, con tiempos de respuesta apropiados. La arquitectura propuesta permite a un cliente móvil presentar regiones de la imagen con diferentes niveles de resolución y calidad, gracias a la granularidad que ofrece J2K. Esta arquitectura facilita el acceso rápido a las imágenes J2K y el envío óptimo de contenido hacia dispositivos móviles con diferentes capacidades y anchos de banda; adicionalmente, es modular, altamente granular e independiente del decodificador, lo cual ofrece flexibilidad para adaptarla a nuevos modelos y desarrollar aplicaciones especializadas.

El desarrollo se realizó de manera iterativa e incremental con el fin de abordar gradualmente la complejidad del proyecto. En primer lugar hubo una fase de recolección de información, en la cual se consultaron en diversas fuentes los trabajos realizados en el área y se definieron los requerimientos iniciales de la arquitectura a diseñar; en esta fase también se establecieron

las herramientas tecnológicas a emplear. Posteriormente, hubo una fase de desarrollo e implementación, la cual, estuvo compuesta de cuatro iteraciones, dentro de las cuales se realizaron actividades de exploración teórica, análisis, diseño e implementación y pruebas. En la primera iteración desarrolló un prototipo de la plataforma cliente/servidor para realizar el envío y visualización de imágenes. En la segunda iteración se introdujo al prototipo la visualización de imágenes con el estándar J2K, teniendo en cuenta las restricciones a nivel de ancho de banda del canal, memoria del dispositivo y calidad del contenido. En la tercera iteración se realizó el diseño, implementación e integración de los servicios web a la plataforma. En la cuarta y última iteración se realizó la integración de la plataforma cliente/servidor con una interfaz simple para dispositivos móviles, que permite la visualización del contenido de manera dinámica. Posteriormente se pasó a la fase de experimentación, en la cual, el modelo se sometió a pruebas de rendimiento a partir de un conjunto de solicitudes del usuario.

Como resultado de las contribuciones directas e indirectas de este trabajo, se realizaron algunos trabajos de investigación, los cuales se especifican a continuación:

- G. Corredor, M. Iregui. Distribución y navegación eficiente de contenido multimedia de alta definición en dispositivos móviles. Memorias del VI Encuentro de Investigación de la Universidad Militar Nueva Granada. ISBN: 978-958-8403-72-4. pp. 132-135, 2011.
- G. Corredor, D. Martínez, E. Romero, M. Iregui. Architectural model for visualization of high definition images on mobile devices. Aceptado para ponencia en "International Conference on Signal Processing and Multimedia Applications (SIGMAP 2012)", Roma, 2012
- D. Martínez, G. Corredor, M. Iregui, E. Romero. An Oriented Service Multilayer Architecture for Virtual Microscopy in Mobile Devices. Aceptado para ponencia en "11th European Congress on Telepathology and 5th International Congress on Virtual Microscopy", Venecia, 2012
- G. Corredor, D. Martínez, E. Romero, M. Iregui. Efficient streaming and visualization of high resolution images on mobile devices using a distributed architecture. Sometido a "IEEE Transactions on Multimedia, Special Issue: Cloud-Based Mobile Media: Infrastructure, Services and Applications."

- G. Corredor, D. Martínez, E. Romero, M. Iregui. Arquitectura Orientada a Servicios para la distribución y visualización de imágenes de alta resolución en dispositivos móviles. Sometido a "Revista Facultad de Ingeniería - Universidad de Antioquia"

Este documento se organiza de la siguiente manera. En el capítulo 2, se realiza una contextualización del problema, presentando una visión general de los estándares J2K y JPIP, y de las arquitecturas orientadas a servicios, así como un breve estado del arte en cuanto a acceso a contenido multimedia y la utilización de estos conceptos en dispositivos móviles. En el capítulo 3, se presentan las consideraciones para realizar el diseño y la implementación del modelo propuesto, incluyendo la especificación de requerimientos y los diagramas de arquitectura, clases, casos de uso y secuencia. El capítulo 4 introduce el problema de la transmisión y visualización de imágenes de alta definición en dispositivos móviles utilizando una implementación de sockets, sobre un canal TCP/IP. Este capítulo fue aceptado para presentarse en la "International Conference on Signal Processing and Multimedia Applications (SIGMAP 2012)". El 5 capítulo complementa el capítulo anterior presentando un modelo arquitectónico orientado a servicios. Este capítulo fue sometido como artículo de investigación a la Revista Facultad de Ingeniería de la Universidad de Antioquia. Finalmente, en el capítulo 6 se presentan las conclusiones y el trabajo futuro.

Capítulo 2

Contextualización

2.1. Introducción a JPEG2000 y JPIP

2.1.1. JPEG2000

JPEG2000 (J2K) es un estándar de compresión de imágenes basado en la transformada Wavelet y el codificador EBCOT, definido por el grupo JPEG, el cual fue diseñado para dar respuesta a la necesidad de un estándar más flexible [12, 34]. J2K tiene varias ventajas sobre el conocido y popularizado estándar JPEG, entre las cuales se encuentran [34]:

- Eficiencia en la compresión mejorada.
- Compresión con y sin pérdida.
- Representación multiresolución.
- Flujo de bits embebidos (Decodificación progresiva).
- Tolerancia a errores.
- Acceso y procesamiento aleatorio al flujo de datos codificados.
- Rendimiento mejorado para múltiples ciclos de compresión y descompresión.
- Formato de archivo más flexible.

La Figura 2.1 presenta los bloques que componen el algoritmo de codificación de J2K. El primer paso consiste en el pre-procesamiento, en el cual la imagen es dividida en áreas rectangulares del mismo tamaño llamadas bloques o baldosas; asimismo, se realiza un salto de nivel DC, que centra los valores alrededor del cero, y una transformada de color, que transforma las imágenes del espacio RGB al YCnCr (irreversible) o al YUV (reversible). Posteriormente, se realiza la transformada Wavelet discreta (DWT, por sus siglas en inglés), que es utilizada para descomponer la imagen en sub-bandas, obteniendo así una representación de la imagen en diferentes resoluciones. En compresión con pérdida, los coeficientes Wavelet obtenidos pueden ser cuantificados, obteniendo una reducción independiente de cada sub-banda. En seguida, se realiza la codificación entrópica, en la cual se utiliza el algoritmo de bloque de código embebido con truncación óptima (EBCOT, por sus siglas en inglés). Este algoritmo permite una representación granular de la imagen a través de bloques de código. En una primera fase, se modelan los coeficientes con un compresor aritmético y posteriormente se abstraen los datos por capas de calidad utilizando una optimización tasa-distorsión [11]. Después de la codificación entrópica, se ejecuta un algoritmo de control de tasa, que reduce el tamaño del archivo truncando algunos datos mientras se mantiene baja la distorsión. Posteriormente, los datos deben ordenarse correctamente de acuerdo al estándar, para lo cual se pueden utilizar diferentes órdenes de progresión. Finalmente, los datos se escriben al flujo de datos codificados J2K.

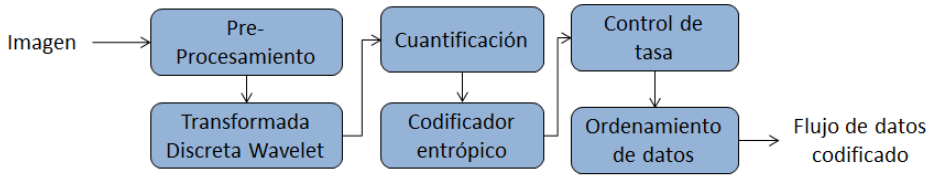


Figura 2.1: Algoritmo de codificación J2K.

El flujo de datos codificados de J2K

El flujo de datos codificados J2K (codestream) es un conjunto lineal de bits con una sintaxis definida que permite la organización de los datos de la imagen. Generalmente, este flujo de datos contiene diversos subconjuntos embebidos, los cuales pueden ser extraídos para recuperar una porción de la imagen original en alguna resolución, capa de calidad o región espacial.

Para lograr esto, J2K define colecciones de bloques codificados (code-blocks) espacialmente adyacentes, conocidas como precintos (precincts). Cada precinto es representado como una colección de paquetes J2K, con un paquete para cada capa de calidad, resolución espacial, componente y posición. De esta manera, cada vez que se requiere visualizar una región particular de interés (a un estado de magnificación y porcentaje de calidad determinado), no es necesario decodificar la totalidad de la imagen, sino extraer los paquetes asociados con el requerimiento, de acuerdo con las solicitudes realizadas por el usuario. Asimismo, es posible mejorar de manera progresiva la representación de la imagen al agregar información de las resoluciones y capas de calidad faltantes [45].

Este flujo de datos está compuesto principalmente por el encabezado principal (main header), los bloques (tiles) y los datos de la imagen (Figura 2.2).

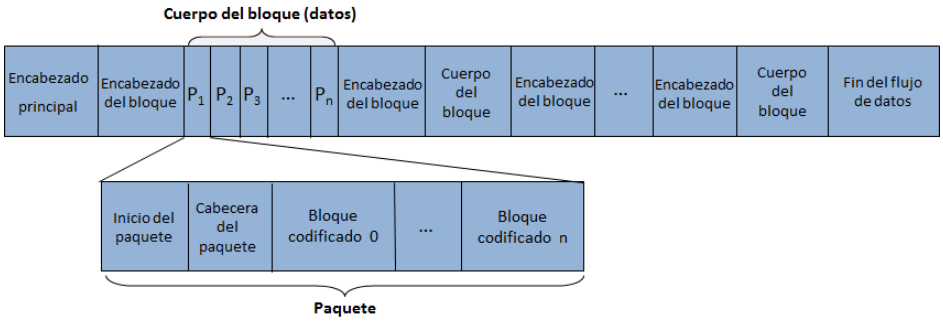


Figura 2.2: Organización del flujo de datos codificados J2K [11].

El encabezado principal contiene información de la imagen codificada, necesaria para realizar el proceso de descompresión: tamaño de la imagen, número de componentes, capas de calidad y niveles de descomposición, orden de progresión, filtro utilizado, tamaño de los precintos y del bloque codificado, entre otros.

Una imagen J2K puede ser dividida en regiones rectangulares codificadas independientemente, llamadas bloques. A su vez, un bloque puede dividirse en diferentes partes (tile-part). En el flujo de datos, el bloque se compone de un encabezado y un cuerpo. El primero contiene información de la codificación del bloque, el índice de la parte de bloque correspondiente y la longitud del cuerpo. El cuerpo corresponde a los datos de la imagen.

Como se mencionó anteriormente, los datos de la imagen pueden organizarse en diferentes paquetes. Un paquete se compone de un encabezado, que contiene, entre otras, información de los bloques codificados incluídos. Los bloques codificados se codifican utilizando árboles de etiquetas (tag trees), que permiten representar los datos de una manera jerárquica.

2.1.2. JPIP

El Protocolo Interactivo de JPEG2000 (JPIP, por sus siglas en inglés) es un protocolo de aplicación que permite la comunicación entre un cliente y un servidor para intercambiar contenido J2K. Este protocolo fue definido en la parte 9 del estándar J2K [13].

El objetivo de JPIP es explotar la escalabilidad de los flujos de datos J2K, permitiendo la entrega de los subconjuntos de datos más apropiados desde un servidor a un cliente interactivo, mejorando incrementalmente la calidad y/o la resolución de manera consistente con los intereses del usuario en un momento dado [45].

Mensajes JPIP

La comunicación entre el cliente y el servidor se realiza mediante el envío de mensajes de solicitud/respuesta. Las solicitudes identifican la ventana enfocada mediante sus atributos geométricos. JPIP define dos tipos de medios para el intercambio de mensajes: basados en precintos y basados en bloques. En el caso de los precintos, a cada encabezado de bloque se le asigna su propio contenedor y a cada precinto se le asigna un contenedor que contenga el flujo de datos del mismo. En el caso de los bloques, cada bloque se asigna a un contenedor que representa el flujo de datos formado al concatenar todas las partes del bloque, incluyendo todos sus encabezados. Si bien, la utilización de mensajes basados en bloques brinda simplicidad en el servidor, ofrece menor flexibilidad en el orden de envío [45].

Solicitudes JPIP

JPIP maneja dos tipos de solicitudes: con y sin estado. Las solicitudes con estado son empleadas en el contexto de una sesión, en la cual, el servidor

mantiene un registro del estado del cliente; es decir, contiene información de los datos que ya ha transmitido y al momento de procesar una solicitud solo envía la información que falta. Es posible que un cliente abra múltiples canales para una misma sesión, permitiéndole realizar diversas solicitudes de manera concurrente [45]. En una solicitud sin estado, el servidor no mantiene información del cliente, maneja cada petición como una transacción independiente, sin tener en cuenta las solicitudes previas [47].

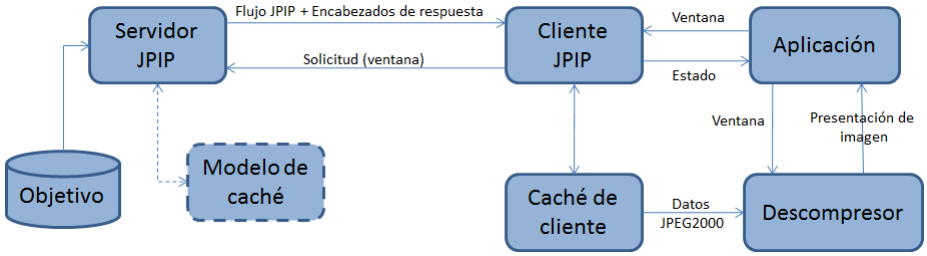


Figura 2.3: Interacción Cliente-Servidor en JPIP [45].

Archivos de índice

Para facilitar el acceso aleatorio a porciones específicas del flujo de datos de manera rápida y eficiente, JPIP propone la utilización de archivos de índices asociados a los archivos J2K. Los archivos de índice proveen una estructura organizada de los rangos de bytes asociados con información del encabezado principal, flujos de código, bloques, paquetes y metadatos, que permiten a una aplicación identificar y extraer bytes de los archivos J2K en una forma simple [45].

2.2. Arquitecturas Orientadas a Servicios

La Organización para el Avance de los Estándares Estructurados de Información (OASIS), define una arquitectura orientada a servicios (SOA, por sus siglas en inglés) como un paradigma para organizar y utilizar funcionalidades distribuidas que pueden encontrarse bajo el control de diferentes dominios; asimismo, provee medios uniformes para ofrecer, descubrir, interactuar y utilizar funcionalidades y, de este modo, obtener efectos deseados consistentes con precondiciones medibles [10]. De un modo más simple, se

puede definir como una aproximación arquitectural en la cual la lógica del negocio se organiza en unidades funcionales o módulos, llamados servicios, que poseen funcionalidades muy específicas y cuentan con interfaces que les permiten comunicarse con otros servicios.

SOA tiene dos objetivos primordiales. El primero de ellos es el desarrollar e implementar nuevas aplicaciones y capacidades a través de la agregación de nuevas unidades funcionales o servicios. En segundo lugar, permitir a las aplicaciones utilizar servicios de otras sobre diversos medios de comunicación [51]. SOA emplea tecnologías basadas en estándares para permitir que los servicios se puedan comunicar con otros servicios en un lenguaje común, lo que ataca el problema de la interoperabilidad, y asimismo, permite la adición y reemplazo de servicios sin que esto afecte el funcionamiento del sistema, brindando reusabilidad, flexibilidad y mantenibilidad [42].

Algunos de los beneficios que tiene la implementación de arquitecturas orientadas a servicios son [50]:

- Reducción de costos operacionales, gracias a la capacidad de SOA para correr sobre múltiples plataformas y sistemas operativos, incluyendo soluciones de código abierto.
- Reducción de los costos de mantenimiento, dada la posibilidad de aislar la capa de procesos del resto de la aplicación.
- Incremento de la agilidad, pues el desarrollo de unidades funcionales (servicios) facilita la labor de desarrollo.
- Conformidad, al tener mayor control sobre los procesos.
- Baja dependencia de sistemas legados.

2.2.1. Servicios y servicios web

Un servicio es una función muy específica de negocio que provee una interfaz que permite la comunicación con otros servicios. Una característica de gran importancia de los servicios es la abstracción, pues éstos tienen la misión de ocultar al programador o al integrador de qué manera se manipula información de bajo nivel; lo único que necesita saber la entidad que llama a determinado servicio es lo que éste hace, la información debe suministrarse y

lo que retorna, no cómo realiza su labor [51]. Otra consideración importante es su independencia, pues un servicio existe por sí mismo y no necesariamente conoce sobre otros servicios o de qué manera operan éstos [42]. Se puede considerar que gran parte del potencial que ofrece SOA se da gracias a los servicios y a la orquestación de los mismos. La Figura 2.4 presenta el modo en que opera un servicio.

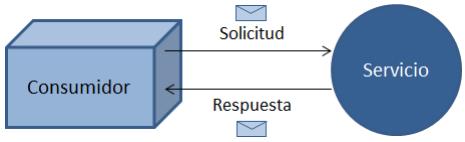


Figura 2.4: Cómo opera un servicio [51].

Dada la importancia que representa un servicio para lo que es SOA, es fundamental realizar un diseño adecuado del mismo. En [4] se describen algunas características que deben tener los servicios:

- Bajo acoplamiento con interfaces bien definidas.
- No tienen estado, es decir, no recuerdan entre invocaciones.
- Atomicidad y granularidad.
- Consideraciones de calidad.

Los servicios web son, en buena medida, responsables del resurgimiento de la popularidad de SOA pues son comúnmente utilizados para implementar este tipo de arquitecturas y ofrecen una solución basada en estándares para el problema de conectar proveedores y consumidores de información sobre internet [17].

2.2.2. Arquitectura básica SOA

La Figura 2.5 presenta los principales componentes de una arquitectura orientada a servicios básica y sus posibles interacciones.

Los clientes (consumidores de servicios) son los encargados de solicitar servicios; desde el punto de vista de arquitectura, éstos hacen referencia a las aplicaciones que buscan e invocan servicios. Los proveedores de servicios

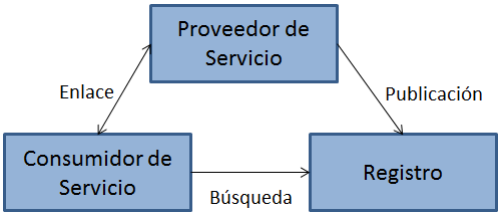


Figura 2.5: Arquitectura básica SOA [41].

publican servicios y los hacen disponibles a los clientes; éstos corresponden a las plataformas donde se da acceso y se ejecuta el servicio. Son aplicaciones ejecutándose en equipos accesibles desde una red. Por último, el registro junta a proveedores y clientes, ofreciendo una herramienta para publicar las interfaces de servicio.

A continuación se describen las interacciones entre los componentes descritos anteriormente:

- Publicación: Para que un servicio pueda ser utilizado, debe ser publicada la descripción del mismo.
- Búsqueda: El cliente busca una descripción de servicio directamente o mediante la búsqueda de servicios.
- Enlace: Cuando se ejecuta el cliente, éste invoca o inicia la interacción con el servicio con los detalles de la descripción del mismo.

2.3. Estado del arte

2.3.1. Acceso a contenido multimedia

En diferentes campos del conocimiento, por ejemplo en microscopía o en cartografía, es necesario acceder a imágenes de gran tamaño. Algunas de las características que ofrece J2K, como las regiones de interés y acceso aleatorio a los datos, presentan a este estándar como una herramienta propicia para trabajar con imágenes de gran tamaño, en forma dinámica y permitiendo escalabilidad e interoperabilidad en ambientes de red y con recursos limitados.

Diferentes estudios han trabajado en el acceso a contenido J2K mediante la optimización de la calidad en la navegación interactiva de contenido. Taubman, autor del algoritmo de codificación de J2K (EBCOT), publicó un estudio en el cual se presentan las consideraciones necesarias para implementar acceso aleatorio para la visualización dinámica de contenido J2K [48]. Por otra parte, Deshpande propuso un modelo de distribución de imágenes J2K utilizando el protocolo HTTP [5]. Estos trabajos sirvieron de base para otros estudios relacionados con la distribución de contenido J2K sobre plataformas cliente servidor [14, 22, 24, 26]. Otros trabajos se han centrado en la aplicación de estrategias de caché y precarga (prefetching) para mejorar la eficiencia en la navegación [1]. Este tema también ha sido objeto de estudio del grupo Bioingenium de la Universidad Nacional de Colombia, en el cual, se ha enfocado hacia el acceso flexible a la información y la visualización de grandes imágenes de patología, representando un aporte bastante significativo al campo de la microscopía [23].

Los trabajos mencionados anteriormente han contribuido de manera significativa en la temática de la distribución de contenido J2K, sin embargo, éstos no tienen en cuenta el problema de las capacidades mínimas disponibles de los dispositivos móviles. La explotación de este tipo de aplicaciones requiere de la calidad adecuada para que los contenidos puedan ser de utilidad para los usuarios, además de accesibilidad y seguridad [27]. Sin embargo, las redes móviles presentan dificultades debido a las limitaciones en ancho de banda, al ruido del canal, que introduce pérdidas de información, y a la limitación en los recursos de los equipos. Con el fin de proporcionar este tipo de servicios, es requerido un tratamiento especial a los contenidos multimedia para permitir a los usuarios acceder e interactuar de forma adecuada.

En el ámbito móvil, los estudios de distribución de contenido multimedia se han enfocado en áreas como confiabilidad y la tolerancia a fallos durante la transmisión en ambientes con ruido [15] y seguridad en la entrega de contenido [49]. Por otro lado, en [16] se propone un sistema de teleradiología para redes móviles, sin embargo, el desempeño de éste sigue siendo limitado para las aplicaciones médicas.

En [8] se presenta un modelo de atención visual para facilitar la navegación de imágenes en dispositivos con pantallas pequeñas sin pérdida de calidad. Sin embargo, en este trabajo se utilizaron imágenes de máximo 1600x1200 píxeles, por tanto, este modelo no aplicaría para imágenes muy grandes, como las satelitales, que pueden tener resoluciones de más de

15000x15000 pixeles. En [36] se propone un modelo para guiar la visualización de imágenes en dispositivos móviles aprovechando las características de J2K; a pesar que en este modelo solo se transmiten las porciones de código necesarias para procesar y presentar la región deseada al usuario, la solución propuesta requiere llenar las posiciones del flujo de código J2K omitidas con datos predefinidos para mantenerlo válido, y de esta manera pueda ser decodificado. Por lo anterior, esta solución no es viable para imágenes de gran tamaño, pues si el móvil debe procesar y decodificar áreas muy grandes, el tiempo de procesamiento y la memoria requerida pueden exceder las capacidades del dispositivo.

Por otra parte, Google Earth es una aplicación muy conocida para acceder a imágenes satelitales de gran tamaño, que corre sobre equipos de escritorio y dispositivos móviles. Esta aplicación utiliza un gran conjunto de bloques jerárquicos preprocesados de diferentes resoluciones espaciales, de modo que, cada vez que un usuario solicita aumentar el nivel de detalle de una imagen, se carga un nuevo conjunto de imágenes correspondientes a la región espacial solicitada [25]. Si bien, esta aplicación ofrece buenos tiempos de respuesta, requiere una organización jerárquica de archivos y directorios bastante compleja y por tanto una gran cantidad de espacio para almacenar la imagen completa. Adicionalmente, solo trabaja con las imágenes provistas y no se permite la utilización de algunas diferentes para uso privado.

2.3.2. Arquitecturas orientadas a servicios en móviles

Gracias a los avances tecnológicos, en la actualidad se está viendo una marcada tendencia hacia la realización de diversos procesos de manera independiente de la ubicación y de otras limitaciones contextuales. Por este motivo, ha surgido la necesidad de emplear múltiples canales de acceso a la información y el desarrollo de sistemas altamente flexibles y que permitan una fácil integración con otros, independientemente de la plataforma de desarrollo, sistema operativo o hardware donde se ejecute.

Con el fin de suplir las necesidades de consolidación e integración, se está tendiendo al desarrollo de modelos basados en arquitecturas orientadas a servicios, apuntando a mejorar la entrega de diversos servicios altamente modificables mientras que se reducen los costos tecnológicos [28]. SOA hace referencia a una arquitectura distribuida, interoperable, fácil de integrar, con alta escalabilidad y con características de seguridad [20]. Dos puntos

claves del proceso de SOA son el diseño de las interfaces de servicios y la división de los componentes del servicio y la cooperación entre ellos, pues va ligado a la calidad y escalabilidad de la arquitectura [20].

Gracias al rápido desarrollo de los dispositivos móviles y al rápido crecimiento de las redes de comunicación inalámbricas, las aplicaciones móviles han cobrado gran importancia para el desarrollo de procesos críticos a nivel organizacional [28]. Por esta razón, en [33] se propone el concepto de SOA móvil, que combina el diseño de sistemas y aplicaciones orientados a servicios, con el dominio de dispositivos móviles de capacidades limitadas. Las características de SOA son adecuadas en ambientes que dependen de conexiones poco robustas y donde se requiere acceder a servicios desde múltiples dispositivos, por tanto, las implementaciones orientadas a servicios se muestran adecuadas para la composición de aplicaciones móviles [33].

Sin embargo, la combinación de características asociadas con SOA y aplicaciones móviles ha presentado nuevos retos para arquitectos, diseñadores y desarrolladores, incluyendo la integración en ambientes heterogéneos y con conectividad inalámbrica [28]. La naturaleza pesada de la orientación a servicios, en términos de la sobrecarga de mensajes que es necesaria para lograr interoperabilidad y bajo acoplamiento, contrasta con la naturaleza liviana de los dispositivos móviles y con la necesidad de transmitir de manera inalámbrica tan pocos datos como sea posible [2]. Por este motivo, es necesario definir un modelo de gestión novedoso para brindar acceso a recursos computacionales de manera automática por demanda y entregar los servicios solicitados con calidad adecuada [28].

En el campo de SOA móvil se pueden identificar dos perspectivas. La primera, corresponde a los móviles como proveedores de servicios web, y la segunda, corresponde a los móviles como consumidores de servicios.

El campo de los móviles como proveedores de servicios web ha sido muy poco explorado, además, requiere superar diversos retos relacionados con el dispositivo como tal (comunicación, uso de memoria, rendimiento y consumo de energía) y otros relacionadas con la capacidad de permitir a los móviles mantener direcciones públicas y puntos que puedan ser alcanzables por consumidores de servicios, pues esta característica es rara vez provista por las compañías de comunicaciones [33].

Por otro lado, la implementación de móviles como consumidores de servicios web, puede ser relativamente fácil de lograr; sin embargo, hay una

serie de consideraciones a tener en cuenta relacionadas con las limitaciones de dichos dispositivos. Las restricciones de conexión favorecen el envío de la menor cantidad posible de datos y las restricciones del dispositivo favorecen el menor procesamiento posible. Respecto al consumo de energía, las transmisiones de radio frecuentes así como el procesamiento continuo de datos, consumen una considerable cantidad de energía, por lo cual, es deseable una implementación que reduzca ambos factores para prolongar el tiempo de vida de la batería [2, 28].

El consumo de servicios web en dispositivos móviles tiene muchos aspectos que pueden afectar la calidad del servicio. Se identifican cuatro componentes que componen la configuración técnica de un escenario de uso de servicios web móviles: el servicio, el dispositivo, la red de conexión y la aplicación que utiliza el servicio. Cada uno de esos tiene varios aspectos que determinan cuál aproximación puede ejecutarse mejor. En general, se concluye que al minimizar la cantidad de datos transferidos puede contribuir a mejorar los tiempos de respuesta [2].

Capítulo 3

Sistema de visualización y distribución de imágenes

3.1. Descripción general del sistema

Como se mencionó anteriormente, diversas aplicaciones requieren acceder e interactuar con imágenes remotas de alta definición, caracterizadas por poseer alta resolución y calidad, así como gran tamaño. Ahora bien, gracias a la portabilidad y las capacidades que ofrecen, los dispositivos móviles se están utilizando para realizar labores de gran importancia, algunas de las cuales pueden involucrar la visualización de dichas imágenes de alta definición. Dado que estas imágenes representan un gran volumen de datos, para realizar la visualización sin pérdida de calidad o resolución en dispositivos móviles con capacidades limitadas, es indispensable diseñar un sistema que permita dar un tratamiento especial a los datos y permita al usuario acceder e interactuar con el contenido de manera simple y dinámica. Adicionalmente, las necesidades actuales están impulsando el desarrollo de aplicaciones altamente flexibles, escalables y que puedan comunicarse con otras aplicaciones de manera simple, sin importar el lenguaje en que se haya escrito, la plataforma donde se haya desarrollado o el sistema operativo en donde se ejecuten.

De este modo, este trabajo presenta un sistema que permite a una aplicación cliente, alojada en un dispositivo móvil, conectarse de forma remota a un conjunto de servicios, con el fin de acceder a imágenes de gran tamaño.

A pesar de las limitaciones del dispositivo sobre el que corre la aplicación cliente, se espera que el usuario tenga tiempos de respuesta adecuados y que el sistema sea amigable y simple de utilizar. El sistema debe realizar principalmente dos tareas:

- Presentar una lista de las imágenes disponibles, para que el usuario seleccione la que es de su interés.
- Permitir al usuario interactuar de manera dinámica con la imagen, solicitando diversas regiones con diferentes niveles de resolución y porcentaje de calidad.

3.2. Alcance

Esta investigación estudia el problema de la distribución y presentación de imágenes de gran tamaño en dispositivos móviles, con capacidades limitadas. El objetivo principal es lograr que el usuario pueda solicitar regiones de interés de una imagen de gran tamaño, con diferentes niveles de resolución y porcentaje de calidad, y pueda visualizarlas en un dispositivo móvil con tiempos de respuesta adecuados. Para determinar el rendimiento de la aplicación se realizaron mediciones de la memoria empleada y del tiempo de procesamiento de las solicitudes realizadas, asimismo, se hicieron comparaciones contra el estándar JPEG, las cuales se encuentran en los capítulos 4 y 5. Para el desarrollo de dichas pruebas se utilizó una imagen satelital que se considera representativa al poseer una resolución de 16384 x 16384 y un tamaño de archivo de 786 MB.

La arquitectura propuesta incluye una interfaz de usuario simple que permite a los usuarios solicitar regiones especificando las coordenadas de una región, un nivel de resolución y un porcentaje de calidad. Si bien, la aplicación es un prototipo, ofrece la posibilidad de trabajar con diferentes imágenes y es flexible para desarrollar sistemas más especializados.

Por otra parte, en la descripción de la arquitectura, se menciona una aplicación de administración, en la cual se agregarían imágenes al sistema de manera automática. Sin embargo, dado que no agrega valor al problema principal de esta investigación, no se incluye en el prototipo implementado. Actualmente, el proceso de adición de imágenes se está haciendo de forma manual, es decir, se codifica una imagen con alguna herramienta de software

para tal fin y posteriormente, se ejecuta la aplicación para crear el índice correspondiente. La creación de las miniaturas se realiza con un software de edición de imágenes.

Por último, el modelo de arquitectura también incluye un componente auxiliar a la interfaz de usuario encargado de administrar la memoria gráfica, el cual mantiene en memoria regiones decodificadas, previamente solicitadas; de este modo, en caso que un usuario que esté navegando por una imagen vuelva sobre una región ya visitada, ésta no necesita volver a cargarse. Ahora bien, la interfaz de usuario desarrollada está enfocada hacia la realización de pruebas de rendimiento de transmisión y decodificación, por tanto, sólo se utilizó para la solicitud y visualización de regiones, y no para navegación como tal. Por este motivo, el módulo de administración de memoria gráfica tampoco se desarrolló, sin embargo, el sistema ofrece flexibilidad para poder adicionarlo.

3.3. Cubrimiento de requerimientos

La captura y análisis de requerimientos es una etapa fundamental en el proceso del Software. Por este motivo, se considera pertinente la identificación de los requerimientos del proyecto, pues sirve como base para la comprensión del sistema y para identificar las necesidades del sistema. De este modo, es posible reconocer algunos de los requerimientos funcionales y no funcionales del sistema, y con ellos realizar una mejor estructuración de la arquitectura del mismo. A continuación, se presentarán los casos de uso, el análisis de los interesados en el sistema y los diagramas de secuencia de los escenarios más destacados.

3.3.1. Casos de uso

A partir del diagrama de casos de uso, es posible identificar los actores del sistema, esta información es de gran utilidad al momento de identificar algunos de los interesados a nivel arquitectural. La Figura 3.1 presenta el diagrama de casos de uso y la Tabla 3.1 la descripción general de cada caso de uso.

Tabla 3.1: Descripción de los casos de uso del sistema.

Caso de uso	**Descripción**
Conexión con el servidor	Permite a la aplicación cliente establecer conexión con el equipo remoto, en el cual residen las imágenes.
Visualización de lista de miniaturas	Presenta una lista con una versión miniatura de las imágenes que residen en el servidor para permitir al usuario la selección de alguna.
Solicitud de carga de imagen	Permite a la aplicación cliente indicar al servidor qué imagen desea visualizar, para que este último realice la precarga de la información de la misma.
Visualización de región de imagen	Permite a la aplicación cliente solicitar una región de la imagen, para su visualización.
Decodificación J2K	Decodifica un flujo codificado JPEG2000 y genera los pixeles para permitir su visualización en pantalla.
Administración de memoria gráfica	Se encarga de almacenar y recuperar los pixeles correspondientes a regiones previamente solicitadas y decodificadas. También se encarga de liberar memoria cuando esté al límite.
Administración caché de paquetes	Se encarga de almacenar paquetes codificados JPEG2000 recibidos desde el servidor y recuperarlos cuando necesiten ser reutilizados. También se encarcarga de liberar memoria cuando esté al límite.
Adición nueva imagen	Permite adicionar nuevas imágenes codificadas al servidor para que estén disponibles para los usuarios del sistema.
Creación archivos de índice	Crea archivos de índice para identificar porciones específicas del flujo codificado de las imágenes JPEG2000.
Creación de miniaturas	Crea versiones miniaturas de las imágenes para facilitar su consulta.
Codificación J2K	Codifica una imagen en formato JPEG2000.
Cerrar conexión	Termina la conexión entre el cliente y el servidor.

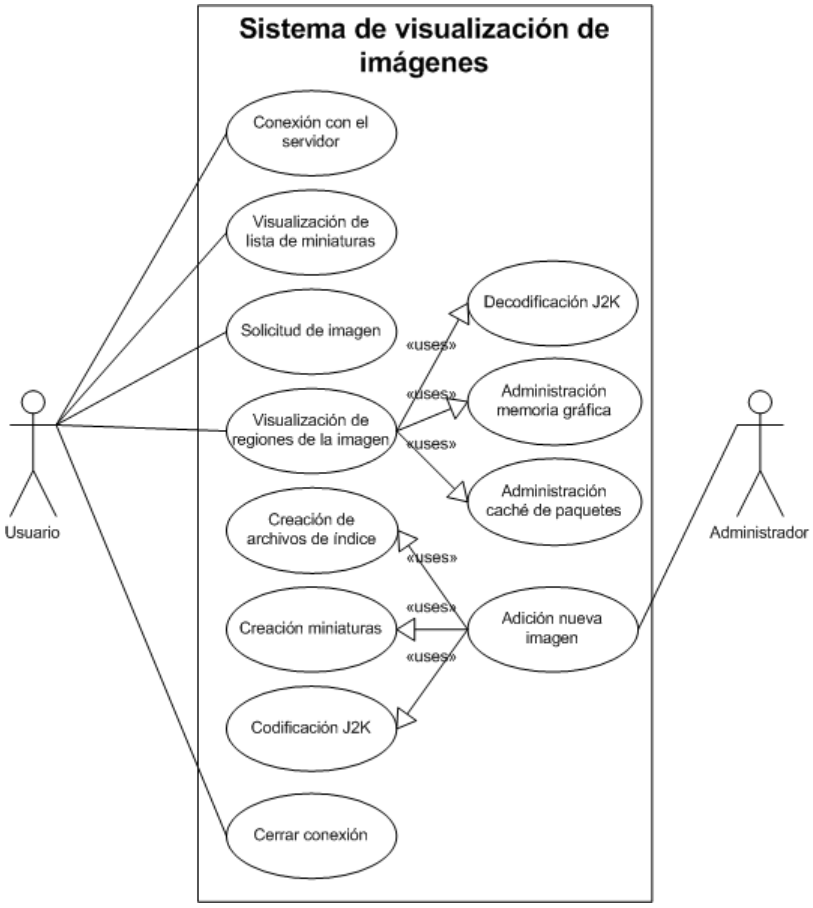


Figura 3.1: Diagrama de casos de uso del sistema de visualización.

3.3.2. Actores e interesados en el sistema

La Tabla 3.2 presenta los actores e interesados en el sistema con sus respectivas descripciones y preocupaciones.

3.3.3. Diagrama de clases

La Figura 3.2 presenta el diagrama de clases del sistema, el cual permite visualizar las relaciones de los elementos de información del sistema.

Tabla 3.2: Interesados en el sistema.

Interesado	**Descripción**	**Preocupaciones**
Usuarios particulares	Conjunto de usuarios que van a consultar e interactuar con las imágenes.	Usabilidad, Efectividad, Rendimiento, Interoperabilidad
Administrador	Personal encargado de adicionar nuevas imágenes al sistema para que puedan ser consultadas por los usuarios.	Mantenibilidad, Escalabilidad
Director de proyecto	Encargado general del proyecto, comprometido a dirigir las tareas que componen el desarrollo y puesta en funcionamiento.	Identificación de requerimientos funcionales y no funcionales, Diseño arquitectónico adecuado, Flexibilidad: Facilidad para adaptar el sistema a ambientes especializados, Interoperabilidad
Desarrolladores	Encargados de la codificación, despliegue y mantenimiento de los módulos del sistema.	Desarrollo sencillo, Mantenibilidad, Flexibilidad, Interoperabilidad
Personal de pruebas	Encargados de la realización de validación del sistema en términos de funcionalidad y atributos de calidad.	Diseño adecuado para para la realización de pruebas, de ser posible que se puedan realizar de manera automática.
Patrocinador	Persona o entidad que proporciona financiación para el desarrollo del sistema	Correcto Funcionamiento (Atributos funcionales adecuados).

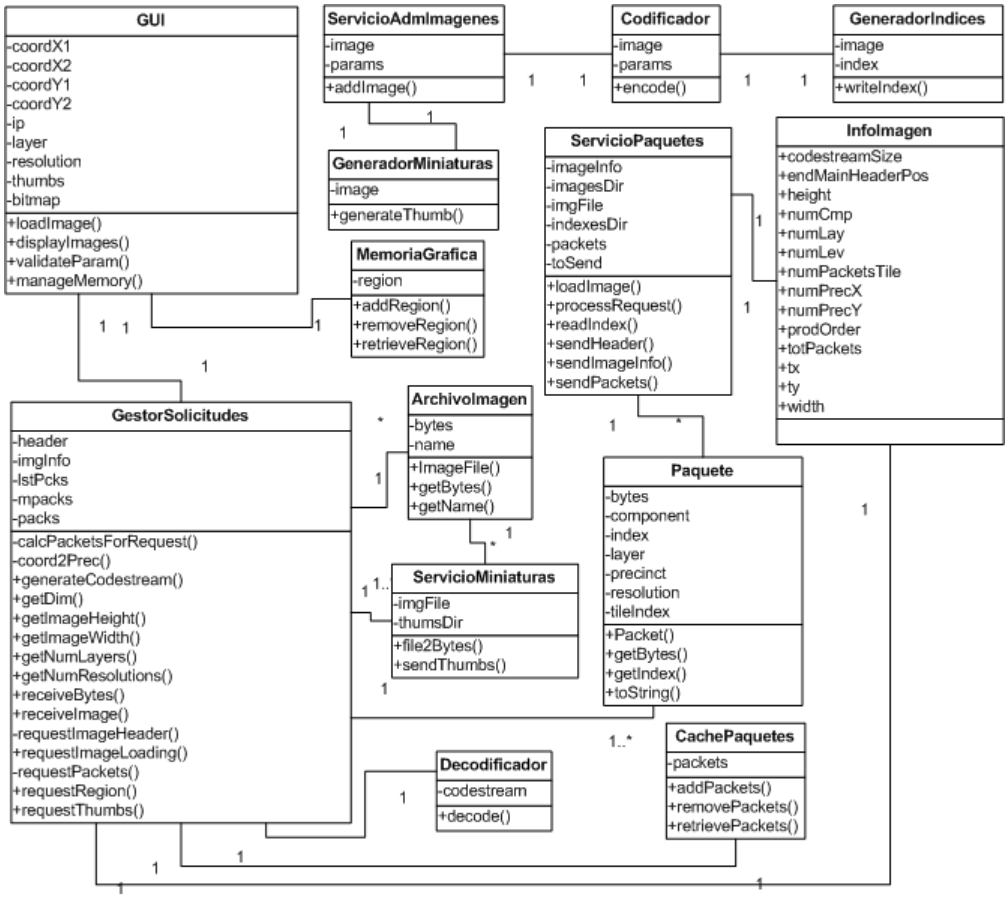


Figura 3.2: Diagrama de clases del sistema.

3.4. Atributos de calidad

Las decisiones arquitectónicas para el desarrollo del sistema de visualización están basadas en los requerimientos no funcionales, obtenidos a partir de la interpretación de los requerimientos del sistema. En este proceso de análisis se identificaron algunos atributos de calidad de mayor relevancia para los interesados en el sistema, como flexibilidad, efectividad, eficiencia, interoperabilidad y usabilidad, los cuales se exponen con más detalle a continuación.

3.4.1. Flexibilidad

Está relacionada con la simplicidad para modificar ciertas funcionalidades del sistema de modo que pueda ser utilizado en un contexto nuevo o similar sin necesidad de rediseñar.

En este escenario, el sistema debe ser modular, de modo que dé respuesta a los requerimientos funcionales actuales de una manera simple y que a su vez permita reemplazar o modificar módulos de manera rápida y simple.

Los procesos de modificación deben durar un máximo de 24 horas.

3.4.2. Eficacia

Hace referencia a la exactitud con que el sistema responde a las solicitudes del usuario.

Para lograr la eficacia deseada, es necesario que los algoritmos encargados de procesar las solicitudes, así como los que calculan los paquetes necesarios para recuperar una región solicitada, estén verificados y validados. Asimismo, es de gran importancia verificar que la comunicación entre los módulos o capas del sistema sea correcta, de modo que cada módulo esté recibiendo las solicitudes indicadas.

En un escenario de ejecución normal, la respuesta del sistema a este requerimiento es que debe presentar al usuario la información que solicitó de manera precisa en un 100 %.

3.4.3. Usabilidad

Está relacionada con la facilidad que tiene un usuario para aprender a utilizar la funcionalidad del sistema cuando se enfrentan a él por primera vez o cuando requiere la realización de una tarea nueva.

Para favorecer la usabilidad es recomendable utilizar mensajes detallados donde sea necesario, así como botones cuya descripción sea clara. También es importante la utilización de funciones, mecanismos de interacción e iconografía que los usuarios conozcan de otras aplicaciones para que la familiarización con el sistema sea rápida.

El usuario debe aprender el funcionamiento del sistema y poder empezar a navegar a través de la imagen en menos de 5 minutos.

3.4.4. Interoperabilidad

Es la capacidad que tiene un sistema para intercambiar información con otros, independiente de la plataforma sobre la que estén desarrollados.

Los servicios web definen un conjunto de tecnologías y especificaciones estándar que permiten a diversas aplicaciones comunicarse e intercambiar información, por lo cual, se presenta como un mecanismo adecuado para permitir a la aplicación responder al requerimiento de interoperabilidad.

3.4.5. Desempeño

Es el tiempo de respuesta del sistema ante la solicitud por parte del usuario. Este es el atributo de calidad más crítico. En el sistema de visualización existen diferentes tipos de solicitud: conexión al servidor, lista de imágenes disponibles, precarga de la imagen y regiones de la imagen para navegar. De dichas solicitudes, la que se considera más crítica es la última, pues se pretende que el usuario pueda navegar de un modo suave y que no tenga que esperar un largo tiempo para poder visualizar las regiones de la imagen.

Considerando que la aplicación cliente se ejecutará sobre un dispositivo móvil, este atributo de calidad cobra una gran importancia, pues es indispensable que el usuario tenga buenos tiempos de respuesta en un entorno de ejecución bastante limitado. Asimismo, la utilización de servicios web, tiene importantes repercusiones en el rendimiento de la aplicación, dato el tiempo y la carga necesaria para transmitir y procesar los datos XML, por tanto es indispensable la aplicación de diversas estrategias. Los tiempos de respuesta para este tipo de solicitudes están directamente relacionados con la transferencia de datos y del tiempo de procesamiento del dispositivo. Por tanto, existe la posibilidad de transmitir datos decodificados, los cuales tienen un volumen bastante alto, pero que agilizarían el proceso de presentación de la región en la pantalla o transmisión de datos codificados, que son mucho más livianos, pero que obligan al dispositivo a ejecutar los procesos de decodificación. En este caso, es preferible favorecer la transmisión de datos

codificados, pues si el área solicitada no excede en gran medida a la pantalla del dispositivo, los tiempos de procesamiento y decodificación no son tan altos.

En un escenario de ejecución normal, el sistema debe responder a una solicitud por parte del usuario en un tiempo no mayor a 5 segundos.

3.5. Diagrama de arquitectura

La arquitectura presentada responde a los requerimientos funcionales y atributos de calidad expresados en las primeras etapas del proceso. Se plantea la necesidad de una arquitectura que presente un bajo acoplamiento, que fomente la usabilidad, la capacidad de modificación y presente un buen desempeño, en respuesta a las necesidades primordiales del sistema.

Las arquitecturas orientadas a servicios se refieren a la utilización de servicios para dar soporte a los requerimientos de negocio. Este tipo de arquitectura trabaja de la mano con los servicios web, que hacen referencia a mecanismos de comunicación estándar entre las aplicaciones. Dada la necesidad de lograr una comunicación entre aplicaciones, se plantea la utilización de servicios web para la consecución de tal requerimiento, reconociendo la arquitectura que hay por debajo de estos procedimientos. Algunas de las fortalezas que ofrece una arquitectura orientada a servicios y que son de gran utilidad para el desarrollo del presente proyecto, es que brinda facilidades para trabajar con modelos de negocio basados en la colaboración con otros entes, con lo cual se facilitan aspectos como la escalabilidad de la aplicación, al brindar facilidad en la integración de tecnologías diferentes, así como la flexibilidad, al mejorar los tiempos de realización de cambios a nivel de procesos.

El punto de vista funcional de la arquitectura define los elementos funcionales del sistema y sus responsabilidades, interfaces e interrelaciones. Para el desarrollo de este punto de vista se definen los procesos que realiza el sistema y, a partir de allí, se identifican unidades funcionales que agrupan funcionalidad relacionada. Este punto de vista es importante a nivel de atributos de calidad como la modificabilidad y el desempeño, éste último de gran importancia para el proyecto actual. A continuación se describe el modelo de estructura funcional desarrollado. Para la realización de este modelo se utilizó un diagrama de componentes UML, en el cual se definen los elementos

funcionales del sistema y sus respectivas interfaces (Figura 3.3).

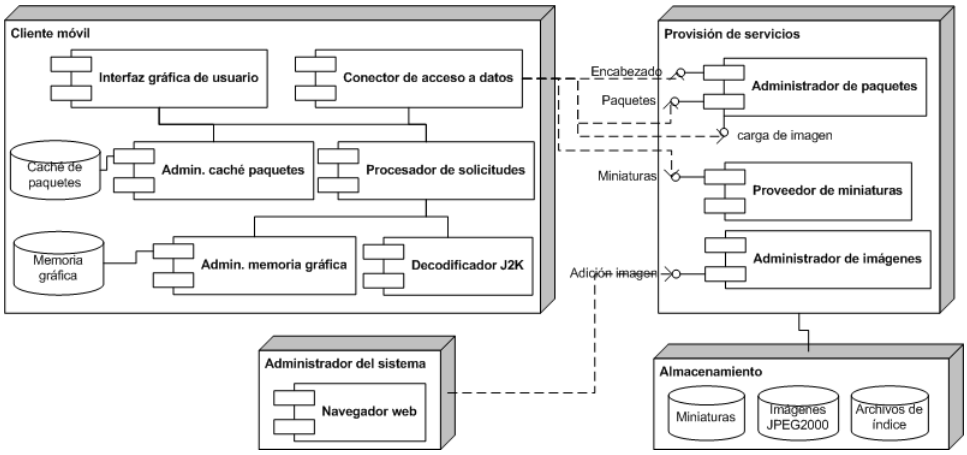


Figura 3.3: Diagrama de arquitectura para el sistema de visualización.

Teniendo en cuenta la limitada capacidad de procesamiento de los dispositivos móviles, la idea principal de esta arquitectura es permitir a una aplicación cliente, residente en un móvil, acceder solamente a regiones de interés de la imagen, de modo que exclusivamente se realiza el envío y procesamiento de los datos necesarios para satisfacer una solicitud específica. El sistema aprovecha la escalabilidad y granularidad de J2K para realizar el envío de los datos más apropiados a la aplicación, permitiendo mejorar, de manera incremental, la resolución y/o calidad de la imagen. El sistema está compuesto por tres capas y está distribuido en dos nodos: cliente y servidor. En el nodo servidor residen las capas de almacenamiento y de provisión de servicios.

3.5.1. Capa de Almacenamiento

En la capa de almacenamiento se encuentran alojadas las imágenes comprimidas con sus respectivos índices y una versión miniatura de cada una.

Como resultado de la codificación de las imágenes, se genera un flujo de datos codificados J2K, el cual se almacena en un archivo sin extensión. En este proyecto no se utiliza el formato JP2, ni las anotaciones de este, pues no son relevantes para los requerimientos de la aplicación. Las imágenes son codificadas con algunos parámetros predefinidos con el fin de facilitar las

labores de indexación y de permitir mayor flexibilidad al acceso a regiones de la misma. A continuación se especifican algunos de los parámetros de codificación.

- Orden de progresión: capa-resolución-componente-posición. Éste es útil en una aplicación de navegación de imágenes donde es deseable que se vaya refinando la calidad de la imagen progresivamente [34]
- Uso de marcador SOP. El marcador SOP (start of packet) indica cuándo inicia un paquete J2K. Para el presente trabajo, este marcador se utiliza para facilitar la creación de los archivos de índice, en los cuales se necesita identificar la posición de cada paquete en el flujo de código.
- Tipo de codificación: Sin pérdida. La codificación sin pérdida (Filtro W5X3) permite la reconstrucción de una imagen tal como se codificó, sin pérdida de información.
- Tamaño del bloque codificado (code-block): 64x64.
- Tamaño del precinto: 64x64. Este tamaño brinda un buen balance entre flexibilidad para acceder a regiones de la imagen y rendimiento en la codificación de la misma. Asimismo, se recomienda que el tamaño del precinto no sea menor que el tamaño del bloque codificado.

Las versiones miniaturas de las imágenes se emplean con el fin de permitir al usuario realizar una consulta rápida y simple de las imágenes que residen en el sistema, para que seleccione cuál desea consultar. En los capítulos 4 y 5 se muestra, mediante experimentación, que el estándar JPEG presenta un mejor rendimiento para la presentación de imágenes pequeñas, en las cuales no se requiere acceder a regiones específicas de la misma, por tanto, éste se empleó para codificar dichas miniaturas.

Como se mencionó en el capítulo anterior, el protocolo JPIP recomienda la utilización de archivos de índice para facilitar el acceso aleatorio a porciones específicas del flujo de datos J2K de manera rápida y eficiente. El índice de cada imagen corresponde a un archivo de texto simple. Para este trabajo se emplea una estructura de archivo de índice basada en la propuesta en [11]. El archivo de índice se compone de dos bloques de datos (Figura 3.4). El primero corresponde a la información de la imagen, en donde se

definen algunos de los parámetros de codificación de las imágenes e información del flujo de datos codificados. Esta información es utilizada por la aplicación cliente realizar labores de solicitud de contenido, y decodificación y presentación de las imágenes. El segundo bloque corresponde a la información de los paquetes de la imagen, donde se listan el índice del paquete, sus características (bloque, capa, resolución, componente, precinto) y posiciones inicial y final. Esta información es utilizada por la aplicación servidor para extraer y enviar los rangos de bytes correspondientes a los paquetes solicitadas por la aplicación cliente.

Actualmente el índice está diseñado e implementado para permitir que una persona pueda entenderlo de manera simple, lo cual hace que la lectura del mismo por parte de una aplicación no sea tan rápida. Sin embargo, es posible optimizarlo para obtener mejores de respuesta.

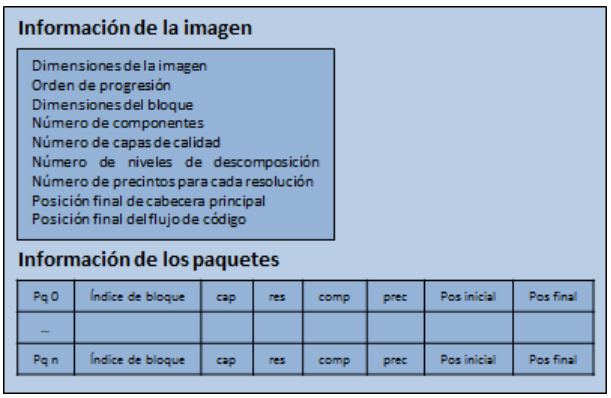


Figura 3.4: Estructura del archivo de índice.

3.5.2. Capa de provisión de servicios

Esta capa se encarga de interactuar con la capa de almacenamiento y de proporcionar las interfaces de servicios para permitir a la aplicación cliente interactuar con el contenido multimedia. Esta capa se compone de tres módulos, los cuales se definen a continuación. Asimismo, la Tabla 3.3 presenta los servicios web existentes, con sus respectivos métodos y respuestas.

Módulo Administrador de imágenes

Este módulo permite al administrador del sistema adicionar imágenes, de modo que éstas estén disponibles a los usuarios. El administrador especifica una imagen sin codificar y algunos de los parámetros de codificación para la misma (como número de capas de calidad y niveles de resolución), y el módulo se encarga de codificar dicha imagen, de generar el archivo de índice y la miniatura del mismo.

J2K define un conjunto de marcadores para delimitar y señalar las características de la imagen y del flujo de datos codificados. Todos los marcadores tienen un tamaño de dos bytes, el primero consiste de un byte con el valor 0xFF y el segundo denota el marcador específico, el cual puede tener un valor en el rango entre 0x01 y 0xFE [12]. Para generar el archivo de índices se desarrolló una aplicación que accede a los bytes de la imagen decodificada y lo recorre identificando algunos de los marcadores y extrayendo la información necesaria, así como las posiciones de algunos. La información extraída es copiada a un archivo de texto con la estructura descrita en la sección anterior e ilustrada en la Figura 3.4.

Módulo Proveedor de miniaturas

Como se mencionó anteriormente, cuando un usuario particular ingresa a la aplicación, puede visualizar un listado con las imágenes disponibles para que seleccione alguna. Este módulo se encarga de enviar el listado de los nombres de las imágenes disponibles, así como los bytes correspondientes a la versión miniatura de las mismas, para permitir a la aplicación cliente móvil que pueda decodificarlos y presentarlos al usuario, para ello, accede al directorio donde están alojadas las imágenes miniaturas, y convierte los archivos de imagen, codificados en formato JPEG, en flujos de bytes que pueda enviar hacia el cliente.

Módulo Administrador de paquetes

Este módulo es primordial para realizar la interacción del usuario con las imágenes. Cuando un usuario selecciona la imagen que desea consultar, el sistema debe hacer una carga de la información de la misma, con el fin de agilizar las consultas. Este proceso de carga incluye la lectura y procesamien-

to del archivo de índice de la imagen; posteriormente, se envía el encabezado principal e información de la imagen codificada (niveles de resolución, capas de calidad, dimensiones, etc.) a la aplicación cliente para que ésta pueda realizar el proceso de decodificación de la misma posteriormente.

Una vez que el usuario ha seleccionado la imagen, el sistema ha realizado la carga de ésta y la aplicación cliente tiene los datos que necesita para decodificar, se puede proceder a solicitar paquetes de la imagen, para presentar regiones a los usuarios. El sistema recibe el listado de los paquetes necesarios, utiliza la información del archivo de índice para acceder a los rangos de bytes donde se encuentran los paquetes solicitados y realiza el envío de los bytes de cada paquete.

Tabla 3.3: Servicios web provistos por la aplicación.

Servicio web	**Método**	**Parámetros**	**Respuesta**
Proveedor Miniaturas	solicitar Miniaturas	(ninguno)	Lista de los nombres y bytes de las miniaturas de las imágenes disponibles.
Admin Imagenes	agregar Imagen	imagen sin comprimir, parámetros de codificación.	Éxito o fallo en la adición de una nueva imagen, y la generación de su índice y miniatura.
Admin Paquetes	cargar Imagen	nombre de la imagen	Información de la imagen cargada.
	solicitar Encabezado	(ninguno)	Bytes correspondientes el encabezado principal de la imagen codificada, previamente cargada.
	solicitar Paquetes	lista de paquetes	Lista de los bytes codificados de los paquetes solicitados de la imagen previamente cargada.

3.5.3. Capa cliente

La última capa corresponde al cliente, que se comunica con la capa de provisión de servicios para poder interactuar con la información de la capa de almacenamiento. Existen dos tipos de aplicaciones cliente: el administrador y el móvil. El primero está encargado de administrar el sistema, adicionando nuevas imágenes codificadas; el segundo corresponde a una aplicación móvil que permite a usuarios particulares consultar las imágenes e interactuar con ellas.

Cliente móvil

Dadas las condiciones del problema presentado, el cliente móvil es uno de los puntos más críticos, pues debido a sus limitadas capacidades y al reducido tamaño de la pantalla, impone restricciones importantes que atacan directamente a los requerimientos no funcionales del sistema como la usabilidad y el rendimiento. El cliente móvil se compone de varios módulos, que permiten a un usuario especificar qué regiones de la imagen deben ser transmitidas y visualizar la representación de la región solicitada; a continuación se describen en detalle dichos módulos.

Conector de acceso a datos: Este módulo es responsable de comunicarse con el servidor (proveedor de servicios). Se encarga de construir y enviar mensajes de petición SOAP a partir de la información enviada por el Procesador de Solicitudes; asimismo, recibe las respuestas del servidor y las envía al Procesador de solicitudes.

Procesador de solicitudes: Este módulo funciona como una interfaz de programación de aplicaciones (API), pues permite a la interfaz gráfica hacer diferentes solicitudes de información sin que se tenga que preocupar por el modo en que se están procesando los datos. Este módulo también se comunica con los demás módulos del cliente móvil.

El procesador de solicitudes recibe las solicitudes de regiones que el usuario realiza mediante la interfaz gráfica y calcula los paquetes necesarios para responder a dicho requerimiento. Para ello utiliza una función encargada de calcular paquetes a partir de coordenadas, nivel de resolución y porcentaje de calidad, para ello emplea un mecanismo para mapear coordenadas a precintos y con esta información, teniendo en cuenta el orden de progresión de

la imagen codificada, determina el índice de los paquetes que se necesitan para representar la región solicitada.

Es posible que para representar cierta región se necesiten paquetes previamente solicitados al servidor. Como no es deseable una transmisión redundante de paquetes, cada vez que se reciben, éstos son almacenados en caché. De este modo, cuando se cuando se calculan los paquetes para una región, si algunos de éstos ya se encuentran en caché, no se vuelven a solicitar.

Cuando se tienen todos los paquetes necesarios para representar una región solicitada, bien sea que se hayan traído del servidor, de la caché o de ambos, se procede a generar un flujo de datos codificados válido para que pueda ser decodificado. Si bien, el algoritmo de compresión J2K y su estructura de paquetes proveen excelentes herramientas para el desarrollo de una aplicación interactiva, la sintaxis descrita en el estándar no incluye la construcción interactiva de un flujo de datos a partir de paquetes ordenados arbitrariamente [45] y dado que no necesariamente se cuenta con todos los datos de la imagen, es necesario hacer algunos ajustes para que el flujo codificado se pueda decodificar.

En [36] se trata este problema llenando las posiciones omitidas del flujo con datos predefinidos. Sin embargo, en este trabajo se propone una estrategia diferente: con base en la información de la región solicitada, se hacen algunas modificaciones al encabezado principal de la imagen, para que incluya solo los paquetes necesarios para reconstruir dicha región. Lo anterior, estableciendo el tamaño de la región, las capas de calidad y los niveles de resolución. De esta manera, el proceso de decodificación es más rápido y eficiente, pues el proceso de transcodificación del encabezado gasta muy pocos recursos y porque el decodificador no debe procesar información adicional vacía o innecesaria. Adicionalmente, esta estrategia brinda independencia del decodificador, pues al crear un flujo de datos válido, cualquier decodificador J2K que siga el estándar, puede procesarlo.

Una vez se construye el flujo de datos válido, se envía a la función de decodificación, la cual se encarga de calcular los pixeles que corresponden a la región solicitada. Posteriormente, estos píxeles son enviados a la interfaz gráfica, que se encarga de presentarlos al usuario.

Administrador de caché de Paquetes: Almacena todos los paquetes J2K recibidos y los recupera cuando son solicitados por el procesador de

solicitudes. El tamaño del caché es parametrizable y puede ser establecido dependiendo de la capacidad del dispositivo. Actualmente, este módulo utiliza una política menos usado recientemente (LRU, por sus siglas en inglés) para descartar paquetes y liberar memoria cuando esté al límite.

Decodificador J2K: Es responsable de la decodificación de la región de la imagen solicitada. Recibe como entrada un flujo codificado J2K válido y retorna el conjunto de los píxeles correspondientes a la representación de la imagen. Este módulo trabaja como interfaz de comunicación con un decodificador J2K estándar, permitiendo el envío de parámetros y la recepción de la respuesta.

Interfaz gráfica de usuario (GUI): El objetivo de esta interfaz es permitir a los usuarios la comprensión del contenido mediante mecanismos de representación e interacción adecuados. Esta interfaz permite al usuario conectarse al servidor, solicitar y visualizar las miniaturas de las imágenes disponibles, y especificar qué regiones de la imagen deben ser transmitidas, con qué nivel de detalle y calidad, para poder visualizarlas.

Administrador de memoria gráfica: Este módulo funciona como una extensión de la interfaz gráfica de usuario y se encarga de almacenar las regiones decodificadas, previamente solicitadas. Este módulo permite que cuando un usuario que esté navegando por una imagen vuelva sobre una región ya visitada, ésta no necesite volver a cargarse. Considerando que el uso de memoria es una restricción importante en los móviles, cuando la memoria llega a un límite parametrizable, se utiliza una política de descarte basada en la distancia, es decir, se van eliminando de manera gradual las regiones lejanas al punto de visualización del usuario.

Las Figuras 3.5-3.10 presentan los diagramas de secuencia de algunas de las interacciones del usuario con la interfaz gráfica.

Cliente de administración

El administrador del sistema es una aplicación cliente que permite a un usuario, encargado de administrar el sistema, adicionar nuevas imágenes para que estén disponibles para el cliente móvil. Para realizar este proceso, el administrador debe conectarse mediante una interfaz web al servicio de generación de índices y especificar la imagen a comprimir, así como algunos de los parámetros de codificación. La Figura 3.11 presenta el diagrama de

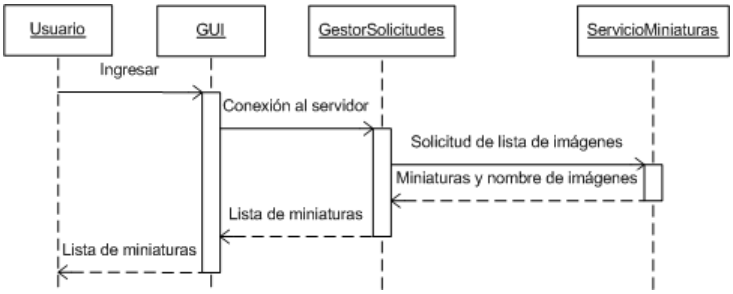


Figura 3.5: Ingreso al sistema.

secuencia correspondiente a la adición de una imagen al sistema.

3.6. Detalles de la implementación

Con el fin de validar la respuesta del sistema propuesto, teniendo en cuenta las limitaciones de los dispositivos móviles, relacionadas con el ancho de banda y el tamaño de la memoria y capacidad de procesamiento, se realizó una implementación de la arquitectura propuesta.

Para implementar el servidor, se utilizó la plataforma Java Enterprise Edition, en su versión 1.5, aprovechando el gran potencial de este lenguaje de programación, así como su interoperabilidad. Asimismo, se utilizó el servidor de aplicaciones GlassFish 3.1 para realizar el despliegue de la aplicación y la publicación de los servicios web. Las pruebas del servidor se ejecutaron sobre un equipo con sistema operativo Windows 7 Professional, 4 GB de memoria RAM y procesador de doble núcleo a 2.71 GHz.

La codificación de las imágenes se probó con diferentes codificadores J2K, entre los cuales se encuentran: JJ2000, es una implementación en Java [32]; Kakadu, es una implementación en C++ [43]; y Jasper, una implementación en C [3].

Para implementar el cliente, se utilizó la plataforma Android, específicamente la versión 2.1, considerando que es el sistema operativo móvil de más rápido crecimiento de los últimos años, que además está presente en un gran número de dispositivos de diferentes fabricantes. Las pruebas de ejecución se realizaron principalmente en dos dispositivos. El primero es el Samsung i9000 (Galaxy S), con sistema operativo Android 2.2, pantalla

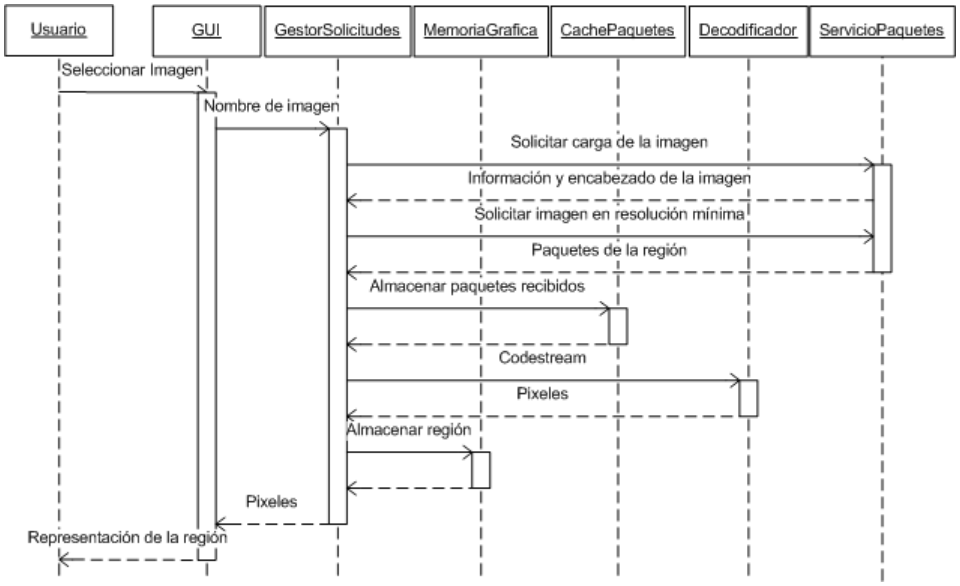


Figura 3.6: Selección de una imagen.

de 800x480, memoria de 512 MiB y procesador de 1GHz. El segundo es el Amazon Kindle Fire, con sistema operativo Android 2.3 (personalizado por Amazon), pantalla de 1024x600, memoria de 512 MB y procesador de doble núcleo a 1GHz.

Considerando las restricciones de memoria de los dispositivos, la decodificación se realizó utilizando la librería Jasper, que es mucho más eficiente que JJ2000 y no tiene restricciones de uso como Kakadu. Considerando que la plataforma Android opera sobre Java, se utilizó el Android NDK (Native Development Kit), una herramienta del SDK de Android que permite construir porciones de las aplicaciones en código nativo, para poder comunicar la aplicación con este decodificador, escrito en lenguaje de programación C.

Dado que la plataforma Android no cuenta con librerías para trabajar con servicios web, se utilizó una librería SOAP de código abierto llamada kSOAP, la cual está diseñada para operar en ambientes Java restringidos (como los móviles).

A continuación se presentan algunas capturas de pantalla con pruebas funcionales realizadas sobre la implementación. La Figura 3.12 presenta el resultado de solicitar una imagen con dimensiones de 1024 x 1024, codificada

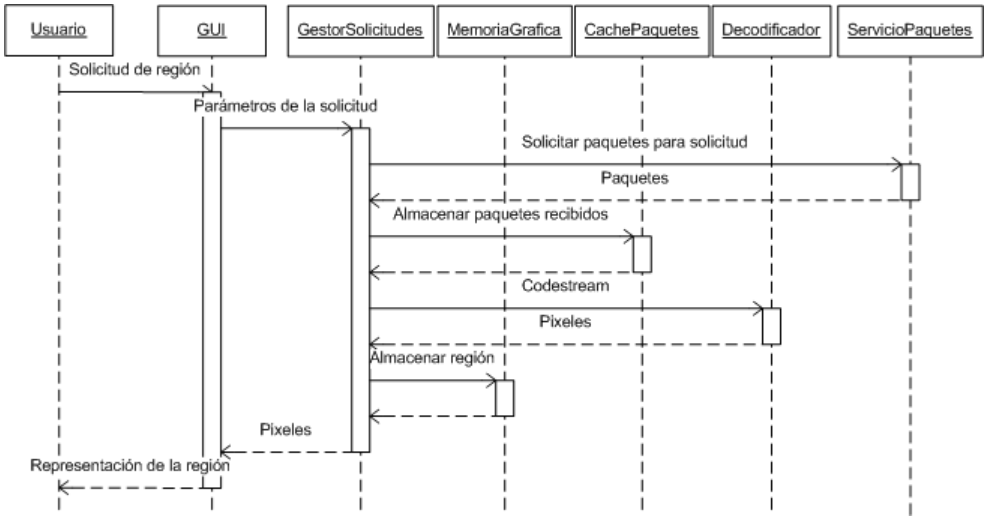


Figura 3.7: Solicitud de una región sin haber realizado solicitudes previas.

con 3 resoluciones y 10 capas de calidad, especificando diferentes capas de calidad a un mismo nivel de resolución. La Figura 3.13 presenta el resultado de solicitar una imagen satelital con dimensiones de 16384 x 16384, codificada con 5 resoluciones y 10 capas de calidad, especificando diferentes niveles de resolución, con calidad máxima.

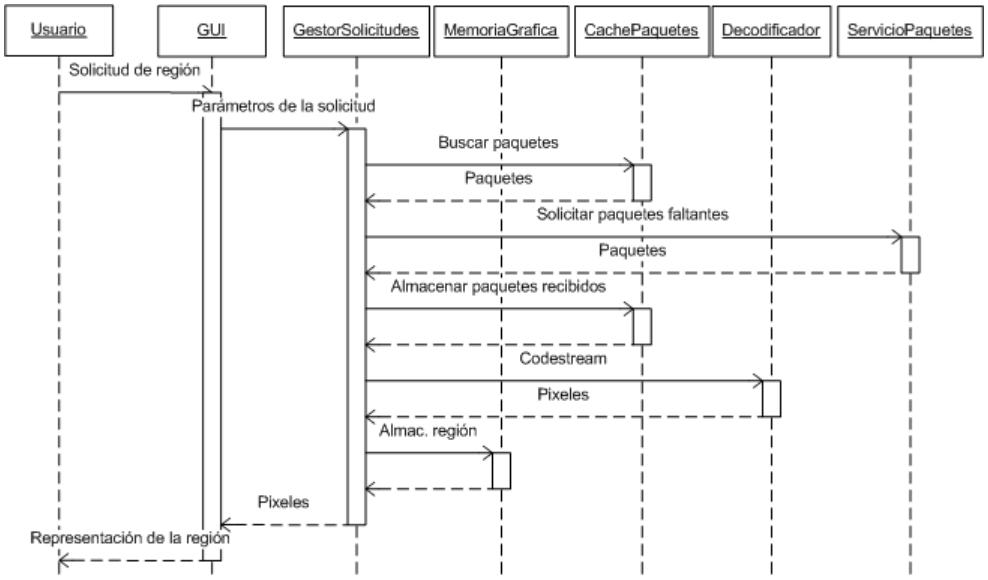


Figura 3.8: Solicitud de una región que necesita paquetes del servidor y previamente solicitados.

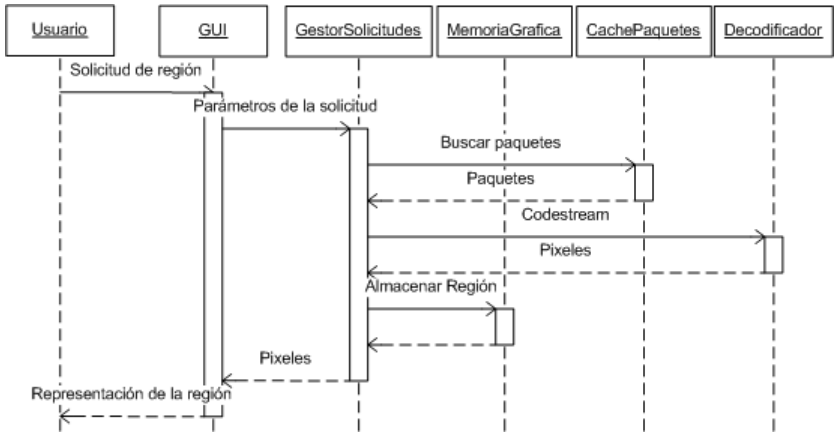


Figura 3.9: Solicitud de una región que sólo necesita paquetes previamente solicitados.

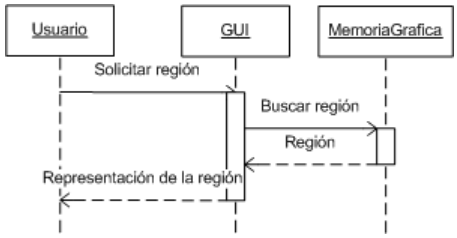


Figura 3.10: Solicitud de una región previamente solicitada almacenada en memoria gráfica.

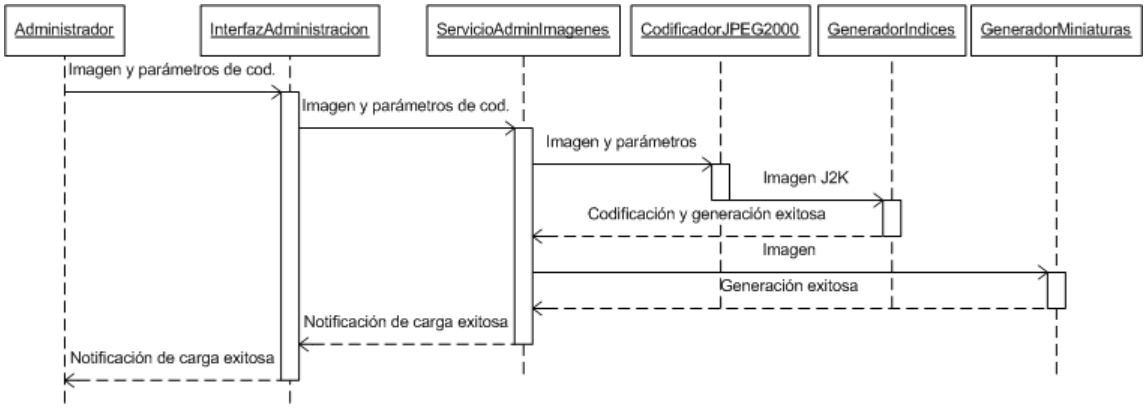
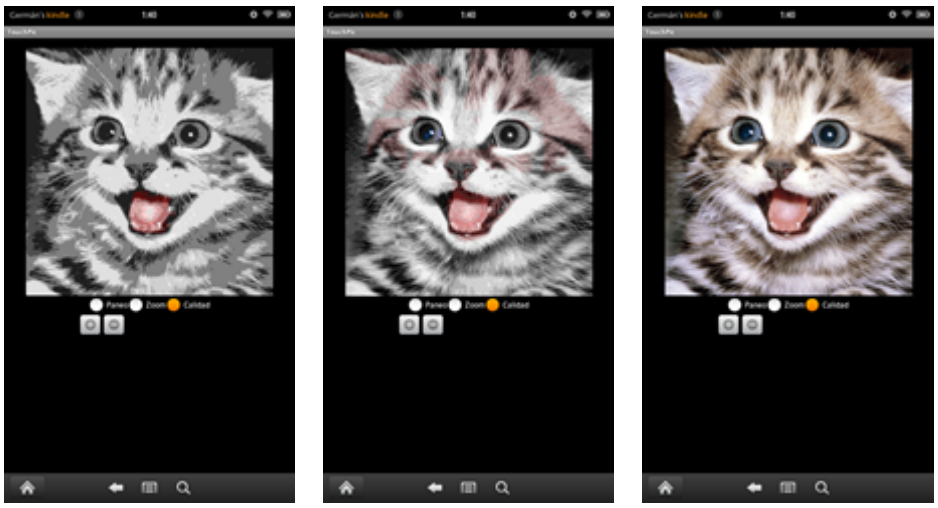
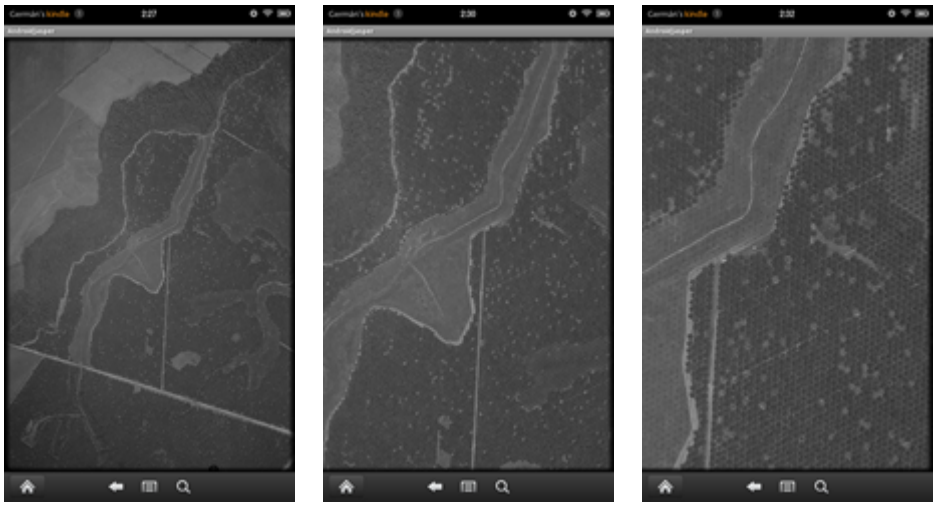


Figura 3.11: Adición de nueva imagen al sistema.

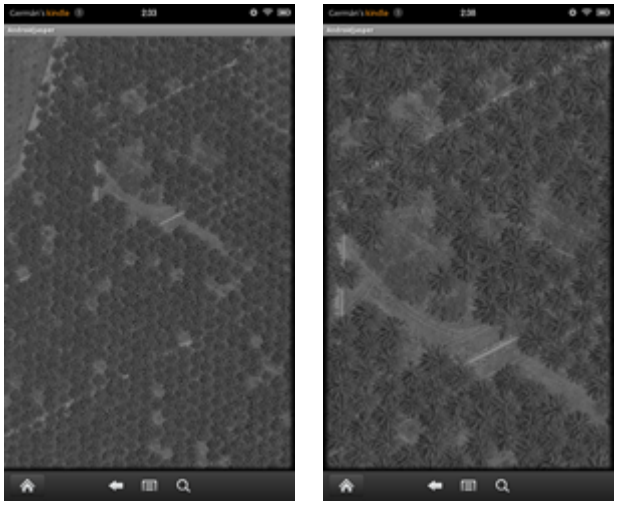


(a) (b) (c)

Figura 3.12: Solicitud de una imagen con diferentes porcentajes de calidad a un mismo nivel de resolución. La figura (a) presenta la imagen con 20 % de calidad, la figura (b) presenta la imagen con 50 % de calidad y la figura (c) presenta la imagen con calidad máxima.



(a) (b) (c)



(d) (e)

Figura 3.13: Solicitud de una imagen satelital codificada con cinco resoluciones y diez capas de calidad, especificando diferentes niveles de resolución.

Capítulo 4

Modelo para visualización de imágenes de alta definición en dispositivos móviles

G. Corredor, D. Martínez, E. Romero, M. Iregui. Architectural model for visualization of high definition images on mobile devices. Aceptado para ponencia en "International Conference on Signal Processing and Multimedia Applications (SIGMAP 2012)", Roma, 2012

Abstract: *In recent years, the mobile device demand has largely increased because of the accessibility, ubiquity and portability of such devices, which are being used not only for personal purposes but also in several applications like education, science, entertainment, commerce and industry, among others. Visualization and interaction with high definition multimedia content, like large images and videos, using mobile devices, represents a challenge because of their very limited machine resources and bandwidth. For such application, this content requires special treatment so that users can properly access and interact. In this article, it is proposed an architectural model for efficient streaming and visualization of very large images on mobile devices using the JPEG2000 standard and an adapted image transfer protocol. Results show that the introduced architecture is effective for visualizing regions of large images and presents good performance, both for transmission and decoding processes, allowing a simple and dynamic interaction between user and images.*

Keywords: *Architecture, decoding, images, interaction, JPEG2000, Mobile devices, protocol, multimedia, streaming, visualization.*

4.1. Introduction

Real-time access to information has become an important task in different daily activities; this is why the use of portable devices and mobile services has increased in recent years. Portability and last hardware advances of mobile devices have induced people to use several mobile applications for managing personal information, remote information access, multimedia services, etc. [37]. Furthermore, currently, the importance of mobile devices is beyond the personal use, because they are starting to be extensively used in fields like education, science and research, entertainment, commerce, industry, among others [18].

The increasing demand of mobile devices has promoted development of more sophisticated devices, with larger capabilities, which have broaden the scope and possibilities. In despite of the latter technological advances in such devices, they still have several external limitations such as communication networks, geographic access and internal limitations such as memory, CPU power and display size [6]. All those limitations make difficult to load, decode and visualize multimedia content such as large images and videos. In some fields, like microscopy or cartography, it is necessary to work with high definition images, characterized by its large size, high resolution and quality. These images are represented by a very large volume of data, so, for visualizing them on mobile devices it is necessary to give them a special treatment to allow the users to properly access and interact.

To ease large image streaming and visualization, different transmission and coding algorithms have been developed, which facilitate storage, transmission and display of graphic content. In particular, the JPEG standard has been used for a long for coding images; however, current necessities require using more flexible and efficient standards [40]. In 2000, the Joint Photographic Expert Group (JPEG) published the JPEG2000 (J2K) standard, which, based on the concept of regions of interest and scalability, represents advances for the image compression technology, for which the image coding system has been optimized not only for being efficient, but also for being scalable and interoperable in network and mobile environments [36, 40]. Le-

vels of detail, regions of interest and progressive transmission are popular concepts for handling graphical data in resource-limited environments [35]. Thererfore, the J2K standard is presented as a suitable tool to address a proper streaming and visualization on mobile devices.

Several works have explored the possibilities to access to J2K content by optimizing the interactive navigation, by proposing distribution protocols, management of client/server platforms, cache and prefetching strategies, among others [1, 5, 14, 22–24, 26, 48]. However these works do not consider the problem of minimal available capabilities of mobile devices.

In the mobile field, some works have focused on areas such as reliability and error resilience over noisy channels [15] and content delivery security [49]. Liu et al [8] presented a model for browsing of images on small displays, however they work with images of range of 1600x1200 pixels; this model is not applicable to very large images, as satellite imaging, which may have sizes of more than 15000x15000 pixels. Rosenbaum and Schumann [36] proposed a model for viewer guidance for mobile devices by exploiting the J2K features; their solution requires to filling the omitted code-stream positions with predefined data to keep it compliant and can be decoded, however, this solution is not applicable to very large images because if the area to be processed and decoded is very large, the required processing time and memory may exceed the capacities of the device. Otherwise, Google Earth is a well-known application for accessing to large satellite images that runs on desktop computers and mobile devices. It uses a very large set of hierarchy prerendered tiles of different spatial resolutions, then, each time a user increases the level detail of the image, a new set of images is loaded [25]. In this way, the application requires a complex organization and hierarchy of files and directories and a big amount of space to store the whole image. Furthermore, this application only works with the provided images and does not allow using different ones.

The present investigation introduces an architectural model to allow optimal interaction with very large images of general purpose from mobile devices. The proposed model facilitates a mobile client to browse very large remote images by displaying regions in a flexible and scalable way, offering a nearly seamless navigation, adapted to the restricted capabilities of the devices and the channel bandwidth. This represents an advantage over available applications that do not allow accessing and navigation in very large images or regions in full quality and resolution. Moreover, the proposed architecture

is modular and decoder independent, permitting thereby easy adaptation to new models and specialized applications.

This paper is organized as follows. In section 4.2, a brief overview of JPEG2000 standard is presented. Section 4.3 introduces the proposed model for streaming and visualization of high definition images on mobile devices. In Section 4.4, an implementation of the proposed model is presented. In section 4.5, experimental results are reported, providing evidence of the performance and efficacy of the model. Finally, section 4.6 presents brief conclusions.

4.2. JPEG2000 Overview

JPEG2000 is an image compression standard designed by the Joint Photographic Expert Group, based on the Discrete Wavelet Transform and the EBCOT encoder [12]. This standard provides several advantages such as improved compression efficiency, lossy and lossless compression, multiple resolution representation, random code-stream access and processing and quality refinement [34].

A single J2K data stream typically contains numerous embedded subsets, which may be extracted to recover a portion of the original image at any of a large number of different spatial resolutions, image quality layers, or in selected spatial regions. A J2K image is split into rectangular regions that are encoded independently, called tiles, but also defines collections of spatially adjacent code-blocks, known as precincts. Each precinct is represented as a collection of packets, with one packet for each quality layer, resolution level and component. These embedded compressed data subsets allow a low quality or low resolution image, or one whose details cover only a small spatial region, to be incrementally improved by adding the missing elements from the compressed data stream [45].

4.3. System Architecture

The proposed application aims to be as simple as possible in order to reach acceptable response times, provided the highly limited memory, processing power and low bandwidth of mobile devices, among other critical

constraints. The main idea is to allow the user to navigate in a seamless way by requesting partial content of an image, according to each region of interest, instead of processing the whole image; in such scenario, only data contributing to the requested representation are transmitted and decoded.

Through the exploitation of the granularity and scalability of J2K data streams, the most appropriate subsets are delivered from a server to an interactive client, incrementally improving the image quality and/or resolution in a consistent manner with the client's interests at any given time [45]. As mentioned in the previous section, J2K codestream is conformed by packets, one for each quality layer, resolution level, component and position. In this way, only those packets belonging to a region at a specific resolution level and quality percent of the image could be retrieved. Because tiles compress less efficiently, introduce unpleasant boundary artifacts at low bit-rates and provide poor resolution scaling properties [45] and because of packets flexibility, in this work, packets are adopted as the fundamental unit of data exchange between server and client.

For this purpose, the server, which stores the set of still encoded images, must be capable of extracting packets from images according to a given request. To enable random access to specific portions of the codestream in a fast way an index file for each image has been off-line generated based on the JPIP specification [13], facilitating a minimal memory use and reduced disk access. The index is a simple text file containing the information organized into two boxes: image information box and packet information box. The former contains information such as width and height, progression order id, number of components, number of quality layers, number of decomposition levels, etc. The latter contains the index, the features (i.e. tile, layer, resolution, component and precinct), and the starting and ending positions of each image packet. The server is also responsible for generating a sequence of J2K packets and delivering it to the client, which is located on a mobile device.

The client has a Graphic User Interface (GUI) that allows the user to facilitate the comprehension of the content through an appropriate representation and means for interaction, by specifying which regions of a particular image must be displayed with a certain degree of detail. The GUI enables to set up the Window of Interest (WoI) coordinates, the resolution level and the quality percent. For this purpose, a method that calculates the necessary packets to meet the user requirement was designed. Since most of the data

may be reused and a non-redundant data transmission is desired, a cache management is performed by storing the received packets; in this way, only missing data are requested to the server.

Although the J2K compression algorithm and its packet structure provide excellent tools on which to base an interactive image application, the data syntax described in the standard excludes the interactive construction of a valid data stream from arbitrarily ordered packets [45]. As not every available image data might be needed, it is necessary to do some modifications to data to keep compliance; previous works have addressed this issue by filling the omitted code-stream positions with predefined data [36]. However, in this work it is proposed a different strategy: based on the information of the requested region, the image main header is modified as to include specifically the packets required to reconstruct the queried ROI. In other words, the image size, quality layers and resolution levels are set to the ROI. In this way, the decoding process is faster and efficient because the transcoding process requires only little computing power and memory, and because the decoder do not have to process trash or empty data. This strategy allows using the decoder with no change, to decompress a set of packets; so that any J2K decoder can be used.

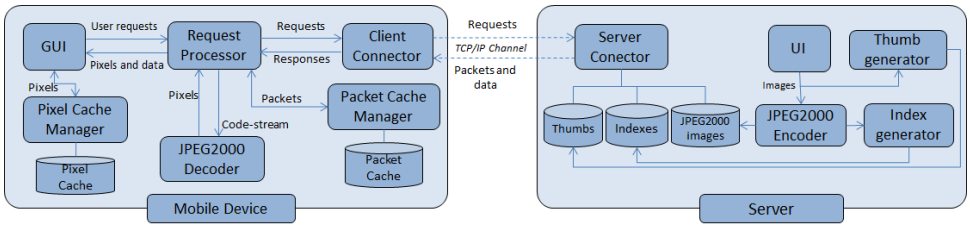


Figura 4.1: Top-level runtime view of the architecture.

4.3.1. Server

The server stores the J2K images and their respective index files and thumbnails. It sends packets and information of the images to the client through a TCP/IP channel.

- User interface (UI): It is an interface through which the system's administrator can manage the server. It naturally allows the addition of new images to make them available to the client.

- J2K Encoder: It is responsible for encoding the images in the J2K standard. It uses a predefined set of parameters to perform encoding. Once the image is encoded, it is stored as a file at a specific location on the server.
- Index generator: It generates an index file for each codified image. The index file is used to get image information and to ease random access to specific portions of the codestream.
- Thumb generator: It generates a thumbnail file for each image. The thumbnails are used to rapidly allow the client to access and view the available images.
- Server connector: It is responsible for communication with the client. It receives and processes the client requests and sends packets and image data.

4.3.2. Client

The client is located in a mobile device that allows the user to interact through a GUI by requesting and viewing regions of interest at certain level of resolution and quality.

Client Connector: It is responsible for communication with the server. It sends request messages to the server and sends the received response data to the RP. Request Processor (RP): The RP is an interface between the GUI and the Client Connector. It is responsible for processing the requests from the GUI. It sends the corresponding request to the Client Connector, sends and retrieves packets from the Packet Cache Manager, generates a compliant J2K codestream to send it to the J2K decoder and sends the generated pixels to the GUI. Packet Cache manager: It stores all the received J2K packets and recover them when are requested by the RP. The cache size is parameterizable and may be set depending of the capacity of the device. The packet cache manager currently uses a Least Recently Used policy to discard packets and free memory when it is almost full. J2K Decoder: The J2K decoder is responsible for decoding the requested image region. It receives as input a transcoded compliant bitstream and returns the set of pixels corresponding to the image representation. Pixel Cache manager: It stores the decoded pixels. It currently uses a policy to discard packets based on the distance; it stores the adjacent areas of a user request and deletes them

when the user is centred in a non-adjacent (distant) region when memory is almost full. The size of the pixel cache is also parameterizable. Graphic User Interface (GUI): It allows the user to specify which regions of an image should be transmitted, at which degree of detail.

4.4. System for streaming and visualization of images

A simple prototype was developed in order to validate the proposed architecture. Hereafter, there is a brief description of the features of the developed application. Since the architecture consists of a client and a server, we developed a stand-alone application for each node.

4.4.1. Server Implementation

As mentioned in the previous section, the server stores the J2K images and their respective index files and thumbnails, and allows communication with the client. The server was developed in the Java platform, Standard Edition, version 1.6.

Uncompressed images and coding parameters are sent to the J2K encoder, which generates a coded image file; encoding was performed with the Jasper J2K implementation [3]. The start of packet (SOP) marker was used to facilitate the index file generation. The index generator receives as input an encoded image; it reads the markers of the image and extracts the necessary information to generate the index file. Finally, the Server Connector is provided with a socket that receives and processes the client requests and sends the corresponding responses.

4.4.2. Client Implementation

The client was developed in the Android platform, version 2.2. The client runs a Graphic User Interface, a Request Processor, a J2K decoder, a Pixel Cache Manager, a Packet Cache Manager and a Client Connector. The communication with the server is done through a socket that the Client Connector manages.

The GUI allows the client to connect to the server by typing the corresponding IP address. Once the connection is established, it is requested the list of available images with their respective thumbnail, which are shown to the user through the GUI. When the user selects one image, it is requested the load of that one and the server sends its information (size, number of resolutions, quality layers, etc.) and its main header. Next, the user selects a region of interest by setting the coordinates, a level of resolution and quality percent. If the requested region is not in the pixel cache, the request is sent to the RP, which calculates the necessary packets to meet that requirement and looks at which packets are stored in packet cache memory, in this way, it only requests to the server the missing data. Once the client has the necessary packets, either from the server or from the cache, they are transcoded into a compliant codestream. As mentioned in the previous section, to let the packets to be decoded, it is necessary to modify the image main header. For the decoding process it was used the Jasper Software, so it was developed an interface function that allows to send to Jasper the codestream and to receive the corresponding pixels. Finally, the pixels are sent to the graphic interface to be presented to the user, that pixels are stored on the pixel cache.

4.5. Experimental results

Two nodes were used to perform the experiments: A server and a mobile device. The former was a desktop computer with operating system Windows 7, 4 GB RAM memory and 2.71 GHz dual-core processor. The latter was the Amazon Kindle Fire, a tablet with operating system Android 2.3, 1024x600 display size, 512 MB RAM memory and 1 GHz dual core processor. The mobile device was connected to a Wi-Fi network with a bandwidth of 25 Kbps.

The experiments were performed with an uncompressed satellite image of 786433 KB and resolution 16384x16384 pixels. The image was J2K compressed by using the following coding parameters. LRCP progression order, 3 components, 10 quality layers, 4 resolution levels (5 resolutions), lossless codification, precinct size of 64x64 for each resolution and codeblock size of 64x64. The final size of that image was 125544 KB, reaching a compression rate of 84.04

4.5.1. Comparing JPEG and JPEG2000

In the first experiment, the performance of streaming and visualization by using JPEG and JPEG2000 standards was compared. As the JPEG standard does not provide multiresolution representation, a JPEG pyramid was constructed, i.e., the original image was coded with 5 different image sizes, each corresponding to a different resolution level. The total size of the 5 JPEG images was 170529 KB, reaching a compression of 78.32

Transmission and processing time were compared for a set of user requests. In this case, the user starts at a certain region of interest and zoom in over it, from the first resolution to the fourth, at full quality. Results are shown in Tables 4.1 y 4.2.

Results show that the number of transmitted bytes for a JPEG image is greater than the JPEG2000 image; due the JPEG standard does not allow a straight access to regions of the image, it is necessary to request and to decode the whole image in order to present it or a region to the user.

This represents an issue when the transmitted image is too large, because it is required a long time for downloading it and because the device may not have enough memory to store and process it. Furthermore, processing time for JPEG is less than for JPEG2000 for small size images; however, for large images in the JPEG format, due the limitation in memory and processing power, the application generated an error and could not decode and show the image representation.

4.5.2. Simulating a User Browsing Protocol

For this experiment, memory usage and transmission and computing times were measured for a set of user requests by simulating a browsing protocol for which a user scrolls over adjacent and overlapping regions. The user starts viewing an image in the lowest resolution level and 50

While the original uncompressed image is 768 MB in size, at the end of the browsing protocol, only 952 KB of compressed data are transmitted. Transmission of the packets took only 4s and the image processing took only 14s for the whole navigation.

Tabla 4.1: Bytes and decoding and computing times for a satellite image by using the JPEG standard. N/D (No data) means that the request was not processed because of memory overflow.

Res. level	**Transm. bytes**	**Transm. time (ms)**	**Computing time (ms)**
1	576288	710,4±134,42	147,1±36,82
2	2334063	2332±353,16	469,7±11,21
3	9310361	8355,3±1582,5	N/D
4	35151251	N/D	N/D

Tabla 4.2: Bytes and decoding and computing times for a satellite image by using the JPEG2000 standard.

Res. level	**Transm. bytes**	**Transm. time (ms)**	**Computing time (ms)**
1	283472	2102,6±622,36	1660±158,73
2	210359	1466,8±141,65	1602±15,82
3	216217	1808,5±764,87	1747,3±285,53
4	199148	1446,1±21,87	1582,7±15,83

Measurement of Transmission and Computing Times

In this experiment, the transmission and computing times were measured for the browsing protocol. Figure 4.3 presents the results of this experiment and includes the accumulated time of transmission and computing.

Requests (a) to (e) represent the scrolling over adjacent and overlapping areas. In this case, it was used a packet cache policy because some regions intersect with previously requested regions. Once the graphic area representation is obtained, it is added to the screen to compose the whole region. Response times of the first request are relatively high because there are not data in cache. As the area of the second region does not overlap with the first, every packet has to be transmitted and decoded. For next requests, transmission and decoding times are lower because of the intersected regions.

Requests (f) to (h) represent a zoom in operation over a specific area. The requested regions correspond to areas of 512x512 pixels. Results show that the transmission time slightly increases because the use of packet cache.

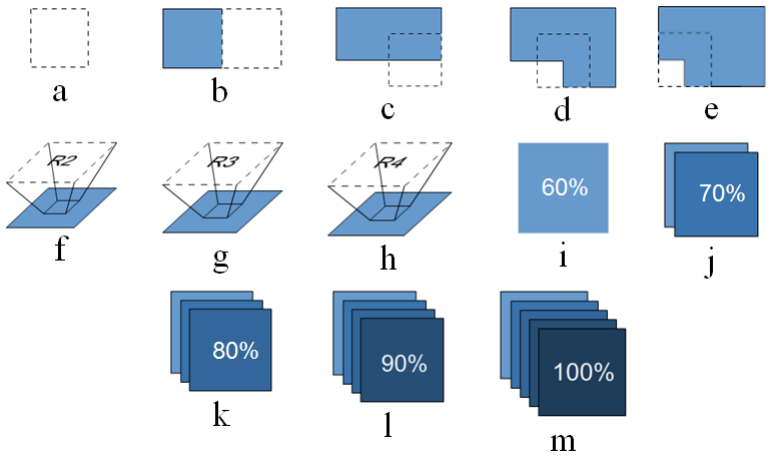


Figura 4.2: Representation of the requested regions. Dashed areas correspond to regions to be requested, while shaded áreas correspond to previously requested areas, stored in cache. Figure (a) shows the first request in which there are not data stored in cache. Figure (b) shows the request of an adjacent region to the first one. Figures (c) to (e) shows requests of areas that overlap previously requested areas. Figures (f) to (h) shows requests of areas simulating a zoom in, since the second resolution level to the fourth one. Figures (i) to (m) shows requests for a quality refinement.

Regarding the decoding process, given that the requested area is lower than the device's display size, it is responding in proper times. Other experiments have shown that requesting areas larger than the display may have repercussions on the system performance. It is highly recommended that requested areas do not double the screen size.

Request (i) to (m) stand for a quality refinement operation over a specific area. Here, the user was starting at a specific region with 50

Memory Usage Evolution

In this experiment was measured the memory usage of the application for the browsing protocol previously depicted. The memory was measured each second during the execution of the browsing session, for a total of 20 seconds. Results are shown in Figure 4.4.

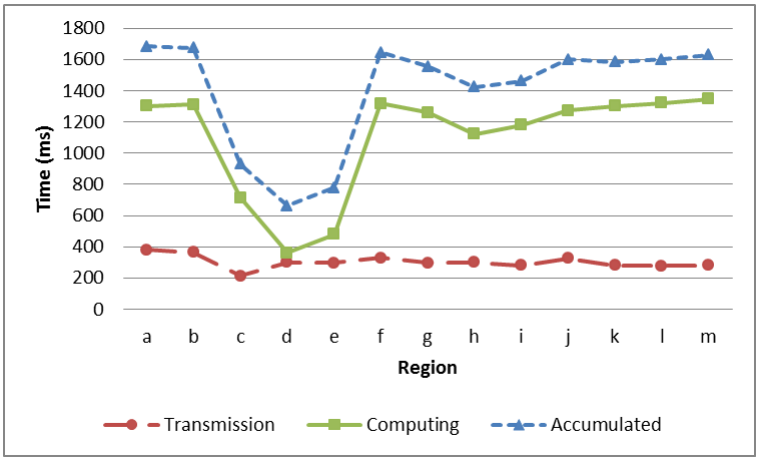


Figura 4.3: Variation of transmission and decoding times for a browsing protocol. Requests (c) to (e) represents regions that overlap with previously requested data, so, less amount of data must be transmitted and processed. Requests (f) to (h) represents a zoom in over a region. Requests (i) to (m) represents a quality refinement process.

Results show that the application memory usage keeps between about 15 and 25 MB, which represents only 4.88

General results show that the introduced architecture not only allows visualizing regions of very large images but also presents a good performance, both for transmission and decoding processes. The procedure to determine every packet contributing to a desired ROI requires low computing power and decoding time is reduced because the area of the requested image is not larger than the display size. It was defined a maximum response time to visualize a requested area of 5 seconds and the results shown the user do not have to wait more than 2 seconds. Additionally, a software-based decoder was used, however, if a hardware-based one is used, response times may improve in a meaningful way. The visualization is adaptable to user necessities; because it can display the best quality and the best resolution level adapted to the possibilities of the device (memory, computing power and screen size). Also, it was demonstrated that use of J2K is most appropriated for large images on mobile devices than JPEG. The use of J2K has important benefits; the standard inherently provides a high compression performance, but also a structure of the encoded data which allows accessing the stream only in those packets contributing to the required ROI and thereby much

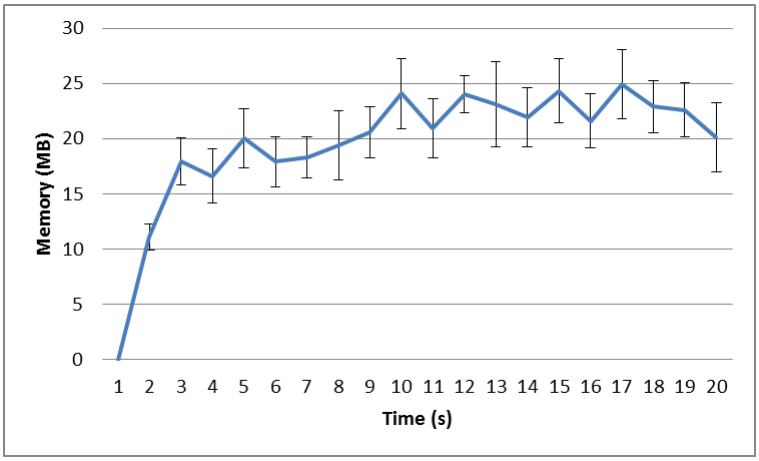


Figura 4.4: Evolution of memory usage of the application for a browsing protocol.

bandwidth can be saved because it is not necessary to transmit and process the whole image. In the server side, the use of J2K standard gives an improved compression efficiency, which allows storing several large images and their respective indexes without requiring a lot of disk space.

4.6. Conclusions

In this paper, an architecture model for streaming and visualization of high definition images, characterized by its high resolution, quality and size, on mobile devices was presented. The proposed model allows a user to request regions of the image at certain level of resolution and quality. The proposed solution takes advantage of the J2K granularity to allow accessing to high definition images to make an optimal content delivery to mobile devices with different capabilities and bandwidth. The client only decodes the required data to generate the representation of the image region, so the memory and processing usage is minimized and the system presents proper response times. Due to the modular design and the loosely coupled decoder, it is easy to adapt the architecture to new models and to develop specialized applications. Future work includes index generation optimization and implementing cache strategies based on user navigation.

Capítulo 5

Arquitectura Orientada a Servicios para la distribución y visualización de imágenes de alta resolución en dispositivos móviles

G. Corredor, D. Martínez, E. Romero, M. Iregui. Arquitectura Orientada a Servicios para la distribución y visualización de imágenes de alta resolución en dispositivos móviles. Sometido a "Revista Facultad de Ingeniería - Universidad de Antioquia"

Abstract: *Data visualization and interaction of multimedia content are challenging tasks on any device, above all if they are mobile, because of the resource limitation, namely low-bandwidth, low storage capacities, small memory, CPU power and display size. In this article, we propose a flexible and service oriented architecture, based upon the principle of cloud processing, for efficient streaming and visualization of high definition images on mobile devices using the JPEG2000 standard and an image transfer protocol, designed to streaming coded JPEG2000 packets. By taking advantage of the JPEG2000 granularity, we achieved to split data processing into different layers. Our results suggest that the proposed architecture is effective for transmission and visualization of large information volumes, while is efficient in*

the use of resources, allowing to build different focused and sophisticated applications.

Keywords: *Cloud processing, images, interaction, JPEG2000, mobile devices, multimedia, protocol, streaming, service oriented architecture, visualization*

Resumen: *El acceso y visualización de contenido multimedia son tareas complejas de realizar sobre cualquier dispositivo, especialmente cuando éstos son móviles, debido al tamaño del contenido y a las limitaciones de recursos como ancho de banda, capacidad de almacenamiento, memoria, capacidad de procesamiento y tamaño de la pantalla. En este artículo, se propone una arquitectura flexible y orientada a servicios, basada en principio de procesamiento en la nube, para la transmisión y visualización eficiente de imágenes de alta definición en dispositivos móviles, utilizando el estándar JPEG2000 y un protocolo de transferencia de imágenes, diseñado para la transmisión de paquetes de la imagen. Gracias a la granularidad JPEG2000, se ha logrado dividir el procesamiento de datos en diferentes capas. Los resultados demuestran que la arquitectura propuesta es eficaz para la transmisión y la visualización de grandes volúmenes de información, a la vez que es eficiente en la utilización de recursos, permitiendo así, usarla de base para desarrollar diferentes aplicaciones especializadas.*

Palabras clave: *Arquitectura orientada a servicios, dispositivos móviles, distribución, imágenes, interacción, JPEG2000, multimedia, procesamiento en la nube, protocolo, visualización*

5.1. Introducción

En los últimos años, la demanda de servicios móviles ha aumentado significativamente, gracias a la ubicuidad y portabilidad que ofrecen [8, 18]. Este incremento ha incentivado el desarrollo de dispositivos cada vez más potentes y sofisticados, tales como teléfonos inteligentes ó "smartphones", tabletas o "tablet PCs", PDAs, cámaras digitales, consolas de video juegos portátiles, entre otros. La última generación de dispositivos móviles ha mostrado importantes avances en hardware, así como nuevas capacidades de software, por lo cual, el número de aplicaciones móviles disponibles también ha aumentado notoriamente. Hoy en día, se están utilizando los dispositivos para diferentes tareas como la gestión de información personal, acceso

inmediato a información y utilización de servicios multimedia [37]. Adicionalmente, hoy en día, la importancia de los dispositivos móviles va más allá del uso personal, pues han ingresado en los dominios de la educación, la ciencia, los negocios, las aplicaciones militares, etc [18].

Dado que, gran parte de la información disponible se presenta como contenido visual, es imprescindible poder acceder a dicho contenido en una forma simple y rápida [8]. Sin embargo, no es una tarea fácil, pues en algunos casos específicos, es necesario manipular grandes volúmenes de datos (por ejemplo, imágenes de alta resolución). A pesar de los últimos avances tecnológicos, todavía existen varios retos que se deben superar para hacer frente a las limitaciones en memoria, capacidad de procesamiento, espacio en disco, tamaño de pantalla y carga de la batería de los dispositivos móviles; ya que, dichas limitaciones hacen que el proceso de transmitir, decodificar y visualizar imágenes de muy alta resolución sea complicado de realizar.

Para el manejo de datos en entornos con recursos limitados son populares algunos conceptos como niveles de detalle, regiones de interés y transmisión progresiva [35]. El estándar JPEG2000 (J2K), basado en el concepto de regiones de interés y acceso aleatorio a los datos, presenta importantes avances en la tecnología de compresión de imágenes, pues el sistema de codificación ha sido optimizado no sólo para ser eficiente, sino también para que sea escalable e interoperable en entornos de red y móviles [36, 40]. Algunos trabajos han demostrado que se pueden ahorrar muchos recursos mediante una adecuada comunicación de imágenes utilizando el estándar J2K [36].

Varios trabajos han explorado las posibilidades de acceder a las imágenes J2K mediante la optimización de la navegación en los contenidos. En [48], se describe una serie de consideraciones importantes para la implementación de acceso aleatorio para visualización dinámica. Deshpande propone en [5] un protocolo de distribución de imágenes J2K utilizando el protocolo HTTP. Otros autores han trabajado en la distribución de contenidos sobre plataformas cliente/servidor [14, 22, 24, 26] y en la utilización de estrategias para mejorar la eficiencia en la navegación, tales como la administración memoria caché y técnicas de precarga [1]. La distribución de la carga computacional también se ha reportado en el plano de procesamiento en aplicaciones médicas [23].

Si bien los trabajos mencionados anteriormente han considerado entornos con recursos computacionales limitados, no han abordado el problema de las muy reducidas capacidades de los dispositivos móviles, que exigen un

tratamiento especial del contenido multimedia para permitir a los usuarios acceder e interactuar de forma adecuada. En el ámbito móvil se han realizado estudios orientados a la confiabilidad de la comunicación y la tolerancia a errores de transmisión [21, 29, 30]. Por otro lado, En [16] se propone un sistema de tele-radiología para redes móviles, pero el desempeño de éste sigue siendo limitado para las aplicaciones médicas.

Las tareas de visualización e interacción con imágenes son bastante exigentes en términos computacionales, por lo cual, es necesario desarrollar herramientas adaptadas para satisfacer las necesidades de los usuarios y superar las limitaciones que impone la navegación de imágenes de gran tamaño en dispositivos móviles. Para lograr esto, es necesario una representación granular de la información, un uso óptimo ancho de banda del canal y un uso eficiente de los recursos de procesamiento. Este trabajo de investigación presenta una arquitectura orientada a servicios, basada en el procesamiento en la nube, que permite la interacción con grandes volúmenes de datos desde dispositivos móviles de una manera óptima. La arquitectura propuesta permite a un cliente móvil presentar regiones de la imagen con diferentes niveles de resolución y calidad, gracias a la granularidad que ofrece J2K. Esta arquitectura es escalable y soporta un número cada vez mayor de usuarios sin afectar el rendimiento de la plataforma.

El flujo de trabajo se divide en tres capas: almacenamiento, acceso a datos y el cliente. El almacenamiento es una capa virtual, que contiene las imágenes comprimidas, así como los archivos de índice y las miniaturas correspondientes para cada una. Esta capa se distribuye a lo largo de una red de nodos, permitiendo así, almacenar más datos y facilitar el acceso a múltiples usuarios. La capa de acceso a los datos se encarga de proporcionar los servicios necesarios para permitir el acceso a las imágenes y a los paquetes que conforman una región de interés solicitada. Por último, la capa cliente se comunica con la capa de acceso a datos para interactuar con las imágenes e información existente en la capa de almacenamiento. Esta arquitectura facilita el acceso rápido a las imágenes y el envío óptimo de contenido hacia dispositivos móviles con diferentes capacidades y anchos de banda; además, es modular, altamente granular e independiente del decodificador, lo cual ofrece flexibilidad para adaptarla a nuevos modelos y desarrollar aplicaciones especializadas.

Este artículo se organiza de la siguiente manera. En la sección 5.2, se detalla el modelo propuesto para la transmisión y la visualización de imáge-

nes de alta definición en dispositivos móviles, incluyendo una visión general del estándar JPEG2000, en donde se presentan las propiedades estructurales que permiten el acceso a la información relevante para determinadas regiones espaciales, niveles de resolución y capas de calidad de la imagen. En la sección 5.3, se presenta una implementación del modelo propuesto. En la sección 5.4, se presentan los resultados experimentales que demuestran el rendimiento y la eficacia del modelo. Finalmente, en la sección 5.5 se presentan algunas conclusiones y el trabajo futuro.

5.2. Arquitectura del sistema

Teniendo en cuenta la limitada capacidad de procesamiento de los dispositivos móviles, la idea principal de esta arquitectura es permitir a un cliente con capacidades mínimas acceder solamente a regiones de interés de la imagen, de modo que solamente se realiza el envío y procesamiento de los datos necesarios para satisfacer solicitud específica. El sistema aprovecha la escalabilidad y granularidad de J2K para realizar el envío de los datos más apropiados a un cliente, permitiendo mejorar, de manera incremental, la resolución y/o calidad de la imagen.

Un flujo de datos J2K normalmente contiene numerosos subconjuntos embebidos que pueden ser decodificados de manera independiente y que representan resoluciones, capas de calidad y regiones espaciales. Este acceso multidimensional a los datos se define en el estándar J2K como bloques codificados ("codeblocks") de regiones espacialmente adyacentes, y se conocen precintos o "precincts". Cada precinto está representado como un conjunto de paquetes, con un paquete por cada capa de la calidad, nivel de resolución, componente y posición espacial.

Estos subconjuntos de datos pueden representar o una imagen completa con baja calidad y/o resolución, o alguna cuyos detalles corresponden solamente a una pequeña región espacial. La representación anterior permite mejorar progresivamente la visualización de la imagen, agregando los elementos que hacen falta [45]. Asimismo, una imagen J2K puede dividirse en regiones rectangulares codificadas de manera independiente, llamadas bloques, baldosas o "tiles"; una imagen debe tener al menos un bloque. La Figura 2.2 muestra una típica representación del flujo de datos codificados J2K, conocido como "codestream".

Como se mencionó anteriormente, el flujo de datos codificados J2K está conformado por paquetes, uno por cada capa de la calidad, nivel de resolución, componentes y posición, por tanto, la solicitud de una región espacial, con cierto nivel de resolución o calidad, puede obtenerse de paquetes específicos en el flujo de bits. La arquitectura propuesta en este artículo utiliza paquetes como unidad fundamental de codificación y de intercambio de datos entre las diferentes capas, no solo por la flexibilidad que ofrecen, sino también porque al utilizar bloques (tiles) se comprime de manera menos eficiente, se introducen artefactos poco deseables a bajas tasas de bits y éstos presentan propiedades de escala en resolución muy limitadas [45].

La arquitectura se compone de tres capas. La primera, la de almacenamiento, es la encargada de almacenar las imágenes comprimidas con sus respectivos índices y una versión miniatura de cada una, ésta última se utiliza para que un cliente pueda consultar y visualizar un listado de imágenes disponibles de manera rápida. El índice es un archivo de texto que organiza la información de una imagen J2K en dos grupos: información de la imagen e información de los paquetes (Figura 3.4). El primer grupo contiene información como ancho y alto de la imagen, orden de progresión, número de componentes, número de capas de calidad, número de niveles de descomposición, etc. El segundo contiene el índice, las características (bloque, capa, resolución, componente y precinto) y las posiciones inicial y final de cada paquete de la imagen. Los archivos de índice proporcionan una manera sencilla para acceder y extraer porciones específicas del código de la imagen, minimizando el uso de la memoria y recursos de acceso a disco. Dado el gran tamaño de las imágenes y sus respectivos índices, el almacenamiento resulta ser costoso, razón por la cual se ha adoptado un esquema distribuido, basado en el modelo de programación y marco de trabajo (framework) de Hadoop MapReduce (Hadoop). Esta aplicación permite gestionar grandes volúmenes de datos de manera rápida y simple, pues se encarga de almacenar la información en diferentes nodos y de resolver las referencias al nodo que contiene la información solicitada. Este esquema brinda alta disponibilidad de la información y escalabilidad, proporcionada por la adición de nodos de almacenamiento independientes, sin aumentar demasiado el tiempo de acceso a la misma.

La segunda capa es la capa de acceso de datos. Esta capa proporciona interfaces orientadas a servicios para interactuar con la información presente en la capa de almacenamiento; la utilización de servicios web permite que aplicaciones de diversas plataformas puedan consumirlas. Esta capa se en-

carga de procesar las diversas solicitudes que pueda realizar un cliente, por ejemplo, adicionar imágenes, para consultar las miniaturas, o para solicitar la carga de una imagen, el envío del encabezado y paquetes de la misma. Para facilitar el acceso aleatorio y rápido a los paquetes de las imágenes codificadas, se ha diseñado una aplicación que interactua con los índices.

La última capa corresponde al cliente, que se comunica con la capa de acceso a datos para poder interactuar con la información presente en la capa de almacenamiento. Pueden haber dos tipos de cliente: el administrador y el móvil. El primero está encargado de administrar el sistema, adicionando nuevas imágenes codificadas; el segundo corresponde a una aplicación móvil que puede consultar e interactuar con las imágenes.

El cliente móvil posee una interfaz gráfica que permite al usuario especificar qué regiones de la imagen deben ser transmitidas y en qué nivel de detalle, especificando las coordenadas, el nivel de resolución y el porcentaje de calidad. Este cliente también posee un módulo encargado de calcular los paquetes necesarios para satisfacer la solicitud del usuario. Una vez que el cliente ha realizado el cálculo de los paquetes necesarios para determinada solicitud, éstos son solicitados a la capa de acceso a datos. Ahora bien, dado que algunos datos se pueden volver a utilizar y no es deseable transmisiones redundantes, se realiza un caché, almacenando los paquetes recibidos previamente; de este modo, solo se solicita la información faltante.

Aunque el algoritmo de compresión J2K y su estructura de paquetes ofrecen buenas herramientas para el desarrollo de aplicaciones interactivas de imágenes, la sintaxis de los datos en el estándar no permite la construcción interactiva de un flujo de datos válido con un conjunto de paquetes ordenados arbitrariamente [45]. Puesto que solo se están solicitando algunas regiones de la imagen, no se tienen disponibles todos los datos de la misma, por lo cual es necesario hacer algunas modificaciones a éstos para que el flujo de datos sea válido y pueda ser procesado. Trabajos anteriores han abordado este problema llenando la información del flujo de código que hace falta con datos predefinidos [36]. Sin embargo, en este trabajo se diseñó una estrategia diferente: con base en la información de la región solicitada, se realizan algunas modificaciones sobre el encabezado principal de la imagen. Esta estrategia es mucho más rápida y eficiente, porque el proceso de modificar la cabecera no requiere de muchos recursos computacionales y además el decodificador responde más rápido al no tener que procesar datos vacíos o “basura”. Con esta estrategia no es necesario hacer modificaciones

importantes al decodificador para que pueda descomprimir un conjunto de paquetes, lo cual, brinda la posibilidad de utilizar prácticamente cualquier decodificador J2K.

La Figura 3.3 presenta el diagrama de componentes de la arquitectura propuesta. A continuación se realiza una breve descripción de los módulos de la arquitectura.

5.2.1. Almacenamiento

La capa de almacenamiento se compone de un conjunto de equipos de cómputo distribuidos que almacenan las imágenes J2K y sus respectivos índices y miniaturas. Esta capa utiliza un sistema de archivos distribuido, que permite manejar volúmenes de datos en aumento y a la vez brinda una manera simple para acceder a la información.

5.2.2. Acceso a datos

La capa de acceso a datos corresponde a un equipo que proporciona las interfaces necesarias para solicitar miniaturas, paquetes e información de las imágenes. También proporciona las interfaces para adicionar imágenes al sistema y generar su respectivo índice. Las interfaces de esta capa son orientadas a servicios.

- Módulo Administrador de imágenes: Es responsable de la adición de imágenes y generación de sus respectivos índices. Para acceder a este módulo se puede utilizar la interfaz de adición de imágenes.
- Módulo Proveedor de miniaturas: Se encarga de acceder y enviar las miniaturas de las imágenes comprimidas. El cliente puede acceder a este módulo mediante la interfaz de envío de miniaturas.
- Módulo Administrador de paquetes: Es responsable de cargar la información de una imagen y enviar datos como atributos, encabezado y paquetes de la misma. Para acceder a este módulo, se puede utilizar alguna de las interfaces ofrecidas, dependiendo de los datos requeridos: encabezado, paquetes y carga de la imagen. El esquema orientado a servicios permite manejar solicitudes de manera simultánea.

5.2.3. Cliente móvil

Este cliente corresponde a un dispositivo móvil que permite al usuario interactuar a través de una interfaz gráfica de usuario solicitando y visualizando regiones de interés con cierto nivel de resolución y calidad.

- Conector de acceso a datos: Es responsable de comunicarse con los servicios de la capa de acceso a datos. Construye y envía mensajes de petición SOAP y recibe y reenvía las respuestas respectivas al Procesador de solicitudes.
- Procesador de solicitudes: Es una interfaz entre la GUI y el resto del sistema. Es responsable de procesar las peticiones que un usuario realice mediante la interfaz gráfica, de calcular los paquetes necesarios para responder a determinada solicitud y pedirlos al conector de acceso o al administrador de caché, y de generar un flujo de código válido para enviarlo al decodificador. Asimismo, se encarga de enviar los datos decodificados (píxeles) a la interfaz de usuario para que puedan ser presentados.
- Decodificador J2K: Es responsable de la decodificación de la región de la imagen solicitada. Recibe como entrada un flujo codificado J2K válido y retorna el conjunto de los píxeles correspondientes a la representación de la imagen.
- Administrador de Caché: Es responsable de interactuar con la memoria caché. Almacena y devuelve paquetes recibidos desde la capa de acceso a datos y también se encarga de borrar algunos paquetes cuando la memoria esté llena.
- Interfaz gráfica de usuario (GUI): El reducido tamaño de pantalla de los dispositivos móviles impone un problema de usabilidad importante para la visualizar imágenes de gran tamaño, por tanto, la interfaz gráfica permite a los usuarios la comprensión del contenido mediante unos mecanismos de representación e interacción adecuados. Esta interfaz permite al usuario especificar qué regiones de la imagen deben ser transmitidas y presenta las respuestas de manera visual.

5.2.4. Administrador del sistema

El administrador del sistema es una aplicación cliente que permite a un usuario, encargado de administrar el sistema, adicionar nuevas imágenes para que estén disponibles para el cliente móvil. Para realizar este proceso, el administrador debe conectarse mediante una interfaz web al servicio de generación de índices y especificar la imagen a comprimir, así como algunos de los parámetros de codificación.

5.3. Sistema para la transmisión y visualización de imágenes de gran tamaño

Con el fin de validar la arquitectura propuesta, se desarrolló un prototipo sencillo. A continuación, se realiza una breve descripción de la aplicación desarrollada y en la siguiente sección se presentan los experimentos realizados sobre este prototipo.

5.3.1. Implementación del almacenamiento

Para realizar el almacenamiento, se utilizó el sistema de archivos distribuidos Hadoop (HDSF), el cual permite configurar un número de nodos para almacenar y replicar los datos. Este esquema otorga tolerancia a fallos y disponibilidad de la información almacenada en los diferentes nodos. Las imágenes J2K codificadas, las miniaturas y los índices se almacenan y replican entre los nodos, los cuales son equipos de cómputo con capacidades promedio.

5.3.2. Implementación del acceso a datos

Como se mencionó en la sección anterior, la capa de acceso a datos proporciona interfaces para interactuar con las imágenes codificadas e información de las mismas. Esta aplicación se desarrolló sobre la plataforma Java Enterprise Edition, versión 5. Asimismo, se utilizó un servidor de aplicaciones Glassfish versión 3.1 para desplegar la aplicación.

Las imágenes sin comprimir y los parámetros de compresión se envían a

un codificador J2K arbitrario; en este caso, se utilizó Jasper [3]. Para el proceso de codificación se forzó la utilización del marcador de inicio de paquetes (SOP) para facilitar la creación de los archivos de índice. El generador de índices recibe como entrada una imagen codificada, lee los marcadores de imagen y extrae la información necesaria para generar el archivo de índice correspondiente (ver Figura 3.4). Por último, las interfaces web reciben y procesan las solicitudes del cliente, y envían las respuestas correspondientes. La Tabla 3.3 muestra los servicios web disponibles, sus respectivos métodos y las respuestas que envían.

5.3.3. Implementación del Cliente móvil

El cliente se ejecuta una interfaz gráfica de usuario, un procesador de solicitudes, un decodificador J2K y un conector de acceso a datos. La interfaz gráfica de usuario permite al cliente conectarse a los servicios estableciendo la dirección IP correspondiente. Una vez establecida la conexión, se solicita la lista de miniaturas disponibles, las cuales se muestran al usuario a través de la interfaz. Cuando el usuario selecciona una imagen, se solicita la carga de ésta y posteriormente el envío de su encabezado principal. A continuación, el usuario selecciona una región de interés estableciendo las coordenadas, un nivel de resolución y el porcentaje de calidad. Esta solicitud se envía al procesador de solicitudes, que se encarga de calcular los paquetes necesarios para satisfacer dicha solicitud. Como se mencionó previamente, el cliente cuenta con memoria caché, la cual almacena los paquetes que recibe de la capa de acceso a datos; de este modo, el procesador de solicitudes revisa cuáles de los paquetes necesarios se encuentran en caché y solicita al conector de acceso a datos solamente los paquetes faltantes.

Una vez se tienen los paquetes necesarios, se crea una secuencia de código J2K válida. Como se mencionó en la sección anterior, para lograr la decodificación de los paquetes, se realizan algunas modificaciones sobre la cabecera principal de la imagen. En este caso, también se utilizó Jasper para realizar el proceso de decodificación. Se desarrolló una interfaz que permitió el envío del flujo codificado J2K hacia Jasper y la recepción de los píxeles correspondientes. Finalmente, dichos píxeles se envían a la interfaz gráfica para así presentar al usuario la representación de la región solicitada.

Tabla 5.1: Parámetros de codificación.

Parámetro	**Valor**
Orden de progresión	capa-resolución-componente-posición
Componentes	3
Capas de calidad	10
Niveles de resolución	4
Codificación	Sin pérdida (W5X3)
Usar SOP	sí
Usar EPH	no
Tamaño del precinto	64x64 para cada resolución
Tamaño del bloque codificado (codeblock)	64x64

5.4. Resultados experimentales

La ejecución de la capa de almacenamiento se realizó desplegando la aplicación HDFS en 5 equipos con sistema operativo Linux. Las máquinas que conforman esta red, tienen memoria de 1 GB, procesador de 2.2 GHz y discos con velocidad de 7200 rpm. Para ejecutar la capa de acceso a datos, se seleccionó a un servidor con sistema operativo SUSE 11.4, con un procesador de 4 núcleos de 2.8 GHz y 5 GB de memoria RAM.

El cliente móvil fue desarrollado sobre la plataforma Android. Se utilizó Android por ser una plataforma gratuita y libre para el desarrollo de aplicaciones y porque actualmente un gran número de dispositivos utilizan este sistema operativo. Particularmente, el cliente se ejecutó sobre el dispositivo Samsung i9000 (Galaxy S), con sistema operativo Android 2.2 (Froyo), 800x480 tamaño de la pantalla, memoria de 512 MiB de RAM y un procesador de 1 GHz.

Los experimentos fueron realizados con una imagen satelital sin comprimir de 786433 KB y resolución 16384x16384 píxeles, en formato PPM. La imagen resultante después del proceso de codificación J2K, utilizando los parámetros mostrados en la Tabla 5.1, quedó con un tamaño de 125544 KB, con lo cual, se logró una compresión aproximada de 84.04

Tabla 5.2: Resolución y tamaño para el JPEG piramidal.

Resolución (pixeles)	**Tamaño (KB)**
16384 x 16384	124266
8192 x 8192	34328
4096 x 4096	9093
2048 x 2048	2280
1024 x 1024	562

5.4.1. Experimento 1. Eficiencia de decodificación y transmisión: JPEG vs.JPEG2000

Para este experimento se compararon los rendimientos de transmisión y decodificación de los estándares JPEG y J2K. Dado que el estándar JPEG no proporciona una representación multiresolución, se construyó un JPEG piramidal, es decir, la imagen original fue codificada con 5 tamaños de imagen diferentes, cada uno correspondiente a un nivel de resolución. La Tabla 5.2 muestra la resolución y tamaño de cada imagen del JPEG piramidal. El tamaño total de las imágenes JPEG fue 170529 KB, con lo cual, se obtuvo una compresión aproximada de 78.32

En este experimento, se compararon los tiempos de transmisión y decodificación para un conjunto de solicitudes en las cuales se simuló una situación en la que un usuario está visualizando una región en el primer nivel de resolución y calidad máxima, y posteriormente empieza a realizar un acercamiento hasta el cuarto nivel de resolución. Los resultados se muestran en la Tablas 4.1 y 4.2.

Los resultados muestran que, a pesar que se logró realizar la transmisión de la imagen JPEG en tercer nivel de resolución, la aplicación no pudo realizar la decodificación de ésta. Por otro lado, la aplicación no pudo realizar la transmisión del cuarto nivel de calidad de la imagen JPEG, pues se llegó al límite permitido de memoria y procesamiento del dispositivo.

Los resultados muestran que el número de bytes de transmisión y, por ende, el tiempo de transmisión para una imagen JPEG es mucho mayor que para una imagen J2K; ésto se presenta debido a que el estándar JPEG no permite el acceso aleatorio a regiones de la imagen y es necesario solicitar y decodificar toda la imagen para que pueda ser presentada al usuario.

Si bien, para una imagen pequeña se tienen tiempos de decodificación mucho mejores al utilizar el estándar JPEG, este estándar no es el más adecuado para la transmisión y visualización de imágenes de gran tamaño. En primer lugar, la descarga de una imagen completa puede tomar mucho tiempo, dependiendo del tamaño de la imagen y las condiciones de la red; además, es posible que el dispositivo, dadas sus limitaciones computacionales, no sea capaz de almacenarla en memoria o en disco para procesarla. Esto se pudo evidenciar en los experimentos realizados; al solicitar la tercera resolución, aunque la aplicación fue capaz de descargar la imagen, no pudo realizar la decodificación de la misma, por lo cual generó un error. Adicionalmente, al realizar la solicitud de la cuarta resolución, la aplicación ni siquiera pudo realizar la descarga de la imagen, pues llegó al límite de memoria disponible, por lo cual, también generó un error y se cerró.

Cabe anotar que, para la realización de estos experimentos no se utilizó una estrategia de caché para JPEG2000, lo que significa que si se realiza administración de caché, tanto para transmisión como para decodificación, se obtendrían mejores tiempos de respuesta.

5.4.2. Experimento 2. Tiempos de respuesta para regiones de diferente tamaño

En esta prueba, se midieron los tiempos de respuesta de solicitudes de áreas con diferente tamaño.

La Figura 5.1 presenta los resultados de este experimento de manera gráfica e incluye el tiempo acumulado de los procesos de transmisión y decodificación. Los resultados muestran que para áreas pequeñas se obtiene tiempos de respuesta buenos, no mayores a 4 segundos; sin embargo, para áreas más grandes, por ejemplo, 1024x1024 píxeles, los tiempos de respuesta no son tan buenos, superando los 10 segundos. Incluso, al solicitar un área de 2048x2048 píxeles, el dispositivo no logra procesarla y la aplicación genera un error. Dado que el tamaño de la pantalla del dispositivo no supera los 1024x1024 píxeles, cuando un usuario esté navegando e interactuando con las imágenes, no será necesario que se soliciten regiones de tal magnitud, por lo cual, los tiempos de respuesta se mantendrán dentro de un límite aceptable.

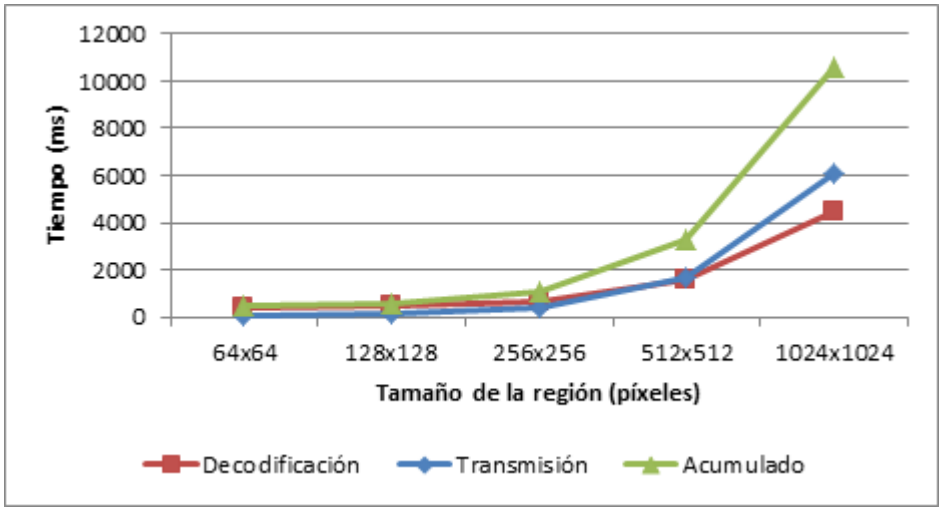


Figura 5.1: Variación del tiempo de transmisión y decodificación para solicitudes de áreas de diferente tamaño. Se puede ver un aumento bastante rápido en los tiempos de respuesta cuando el área aumenta. Para un área de 1024x1024 píxeles el tiempo es bastante alto, sin embargo, dadas las características del dispositivo, no es necesario realizar solicitudes de esta magnitud. Para un área de 512x512 píxeles, más acorde a la pantalla del dispositivo, se presentan buenos tiempos de respuesta.

5.4.3. Experimento 3. Visualización de una región variando el porcentaje de calidad

En este experimento, se simuló el proceso de refinamiento de calidad para una región de 512x512 píxeles. Para ello, se solicitó un área con el porcentaje de calidad más bajo y progresivamente se solicitaron y decodificaron las capas correspondientes a los porcentajes de calidad superiores, mejorando así la presentación de la imagen que el usuario visualiza. Este proceso aprovecha la administración de memoria caché, pues utiliza los paquetes solicitados previamente, reduciendo los tiempos de transmisión para procesar la petición.

Los resultados de la Figura 5.2 muestran que el tiempo de decodificación aumenta para una capa respecto a la anterior; esto es lógico pues para realizar la decodificación de una nueva capa, se necesitan los datos de la anterior y no es posible hacer un caché de decodificación para capas de calidad de

manera directa. Ahora bien, en cuanto a transmisión, se puede notar que no representa un gran tiempo, gracias a la utilización de memoria caché. Por último, es notorio un aumento en los tiempos de respuesta cuando se aumenta la calidad, sin embargo, gracias a la flexibilidad del sistema, el usuario puede determinar cuál es el porcentaje máximo de calidad deseado para cada solicitud, de acuerdo a sus necesidades, y de esta manera, obtener tiempos de respuesta adecuados.

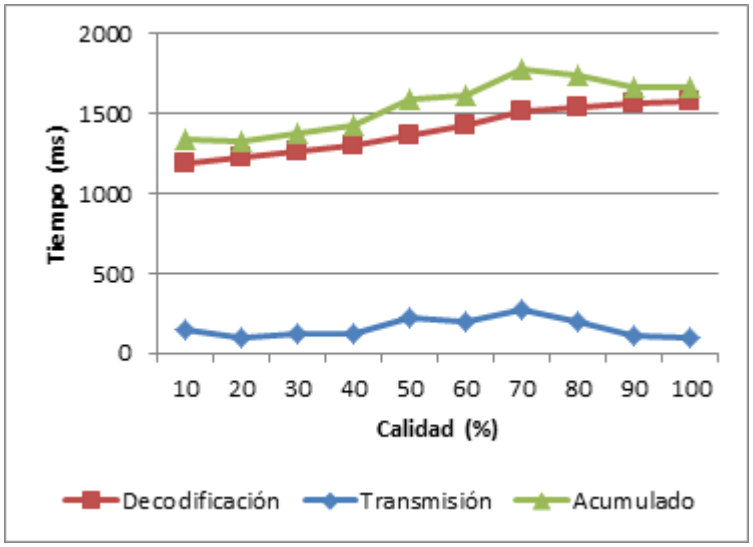


Figura 5.2: Variación del tiempo de transmisión y decodificación para solicitudes de un área con aumento progresivo de calidad. Se puede evidenciar que los tiempos de transmisión no son muy altos ni presentan gran variación gracias a la utilización de caché de paquetes, mientras que hay un aumento considerable en los tiempos de decodificación cuando se solicita un porcentaje de calidad altos; sin embargo, la aplicación es flexible y permite al usuario seleccionar el porcentaje de calidad que desee de acuerdo a sus necesidades y a las capacidades del dispositivo, lo cual puede ayudar a que tenga buenos tiempos de respuesta.

5.4.4. Experimento 4. Visualización de áreas contiguas y solapadas con mismo nivel de resolución

En este experimento, se simuló un protocolo de navegación en el cual un usuario se desplaza a través de regiones contiguas y solapadas, utilizando

un mismo nivel de resolución. La Figura 5.3 muestra la representación de las regiones consideradas para esta prueba. En este caso, también se utilizó administración de caché. Dado que algunas de las regiones solicitadas se superponen con regiones previamente solicitados, solamente se solicitan los paquetes correspondientes al área restante; asimismo, solamente se decodifica el área restante y esta se agrega a la pantalla de visualización del usuario para componer la región completa.

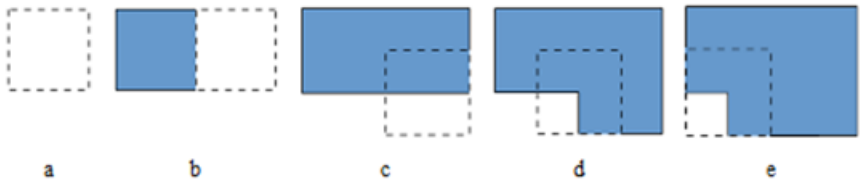


Figura 5.3: Variación del tiempo de transmisión y decodificación para solicitudes de áreas contiguas y solapadas con un mismo nivel de resolución. En este caso se puede ver que al solicitar una región completamente nueva, los tiempos de respuesta aumentan notoriamente, sin embargo, cuando se solicitan regiones que se solapan, los tiempos de transmisión y decodificación disminuyen gracias a la administración caché de paquetes y de píxeles que se realiza.

La Figura 5.4 presenta los resultados de esta prueba. Los tiempos de respuesta de la primera solicitud son altos en proporción, considerando que no se ha realizado una petición previa y por tanto no hay datos en caché. Asimismo, como el área de la segunda solicitud no se solapa con la de la primera, todos los paquetes correspondientes a ésta deben ser transmitidos y decodificados. En las siguientes solicitudes, los tiempos de transmisión y decodificación son mucho menores, gracias a que las regiones solicitadas se solapan con regiones de peticiones anteriores y por ello se necesita enviar y procesar una menor cantidad de datos.

En términos generales, los resultados muestran que la arquitectura introducida en este artículo presenta un buen desempeño, tanto para los procesos de transmisión y decodificación. El procedimiento para determinar los paquetes que contribuyen a una región de interés utiliza pocos recursos computacionales. Por otra parte, el tiempo de decodificación se mantiene en límites aceptables siempre que el área solicitada no exceda en gran medida el tamaño de la pantalla de visualización. En cuanto a la transmisión de imágenes, el uso de J2K tiene ventajas importantes, pues este estándar

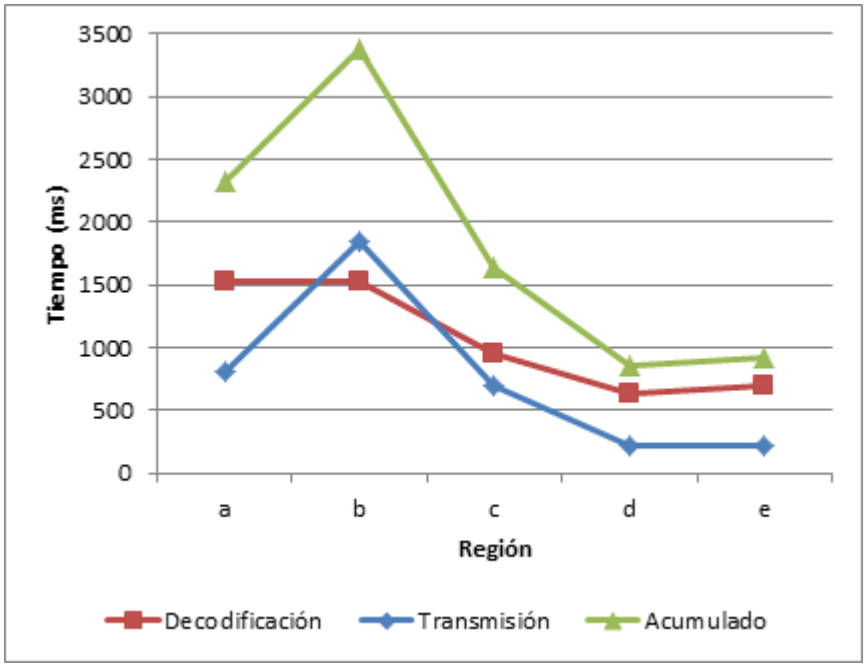


Figura 5.4: Variación del tiempo de transmisión y decodificación para solicitudes de áreas contiguas y solapadas con un mismo nivel de resolución. En este caso se puede ver que al solicitar una región completamente nueva, los tiempos de respuesta aumentan notoriamente, sin embargo, cuando se solicitan regiones que se solapan, los tiempos de transmisión y decodificación disminuyen gracias a la administración caché de paquetes y de píxeles que se realiza.

no solo proporciona un alto rendimiento en la compresión, sino que también presenta una organización de los datos codificados bastante flexible, que permite acceder solamente a los paquetes que contribuyen a la región solicitada, de este modo, se puede ahorrar mucho tiempo de transmisión al no tener que transportar la imagen completa.

Como nota final, la razón por la cual se realizan procesos como la decodificación y el cálculo de paquetes en el cliente, es que esta arquitectura pretende favorecer más los tiempos de transmisión que los de decodificación. Esto es debido a que los tiempos de transmisión presentan una mayor variación, pues no solo dependen de las capacidades del dispositivo, sino también de la velocidad y utilización (congestión) de la red en la cual se encuentre,

algo que escapa al control del usuario.

5.5. Conclusiones

En este artículo, se presentó una arquitectura orientada a servicios para la transmisión y la visualización de imágenes remotas de gran tamaño en dispositivos móviles, en los cuales se presenta un ancho de banda reducido y limitados recursos computacionales, de una manera eficaz y eficiente. El modelo aprovecha la flexibilidad y granularidad de J2K, permitiendo a un usuario solicitar regiones de la imagen con cierto nivel de resolución y calidad. El acceso directo a los paquetes de la imagen comprimida permite transmitir sólo la parte necesaria de la imagen, reduciendo el tiempo de transmisión. Asimismo, dado que el cliente sólo debe decodificar los paquetes necesarios para generar la representación de la región solicitada, se reduce notoriamente el gasto de memoria y el tiempo de procesamiento del dispositivo móvil. La implementación de mecanismos de caché, también presenta una mejora significativa, pues contribuye a reducir los tiempos de transmisión y decodificación en algunos casos. Los resultados presentados demuestran que el sistema responde adecuadamente a las peticiones de usuario en un tiempo corto. Además, gracias al diseño modular y a la independencia del decodificador, esta arquitectura puede adaptarse de manera simple a nuevos modelos y al desarrollo de aplicaciones especializadas.

Capítulo 6

Conclusiones y discusión

En este trabajo de investigación se presenta una arquitectura orientada a servicios para la transmisión y visualización de imágenes remotas de alta calidad, caracterizadas por poseer alta resolución y calidad, así como gran tamaño, en dispositivos móviles. La solución propuesta aprovecha la granularidad del estándar J2K para permitir acceder a las imágenes y realizar una entrega de contenido óptima a dispositivos móviles con diferentes capacidades y ancho de banda.

Las aplicaciones orientadas a servicios poseen una naturaleza pesada debido a la transmisión y procesamiento de datos XML que, si bien es adecuado para organizar información, no está diseñado para el mejor rendimiento. Dadas estas condiciones, junto a las limitaciones computacionales de los dispositivos móviles, fue necesario diseñar un sistema bastante simplificado, que minimice la cantidad de datos transmitidos y el procesamiento de los mismos. Los resultados experimentales mostraron que el tiempo de transmisión no es muy alto, pues, en primer lugar, se están transmitiendo datos codificados, que gracias a la eficiencia de la compresión de J2K, tienen un tamaño mucho menor a los datos descomprimidos, además, solo se están recuperando los paquetes que contribuyen a la reconstrucción de una región particular, por lo cual, no se están transmitiendo grandes volúmenes de datos. Adicionalmente, gracias a la utilización de caché de paquetes, se elimina la solicitud y transmisión de datos redundantes. Por otra parte, el tiempo de decodificación también es relativamente es bajo, pues se están procesando regiones que no exceden por mucho el tamaño de la pantalla de visualización del dispositivo y se están utilizando librerías nativas que son

mucho más rápidas para el procesamiento de los datos.

Otra gran ventaja del sistema, es que se diseñó para ser independiente del decodificador, lo cual trae consigo ventajas importantes, pues no es necesario entrar a modificar un decodificador para poder integrarlo al modelo, sino que basta con crear una interfaz para que se puedan comunicar.

La revisión del estado del arte demostró que el área de la transmisión y visualización de contenido multimedia de alta definición en móviles ha sido muy poco explorada y actualmente hay pocas aplicaciones que realicen tareas similares. El modelo propuesto exhibe algunas características que lo distinguen de otros trabajos en al área: la arquitectura es orientada a servicios, provee acceso a las imágenes de una manera estándar mediante internet, permite la utilización de cualquier tipo de imágenes y permite la recuperación de contenido de una forma efectiva y dinámica en dispositivos con capacidades limitadas. Los resultados muestran que el modelo propuesto no solamente permite la visualización de regiones de imágenes de gran tamaño, sino que también presenta un buen rendimiento, tanto para el proceso de transmisión, como para el de decodificación. El sistema propuesto es flexible, pues cuenta con un diseño altamente modular, lo cual permite que la arquitectura pueda adaptarse a diferentes problemas y aplicaciones especializadas.

6.1. Trabajo futuro

Muchas de las consideraciones que se tuvieron en cuenta en el diseño del sistema no se implementaron teniendo en cuenta algunas restricciones de tiempo y el poco valor que agregaban al trabajo de investigación. Sin embargo, se han determinado algunos aspectos que permitirían pasar de un prototipo a una aplicación estable, y asimismo, mejorar el rendimiento del sistema, los cuales se mencionan a continuación.

Antes de poder realizar solicitudes, el sistema debe realizar una carga de la información de la imagen, que extrae del archivo de índice respectivo. Este proceso de carga se toma un tiempo considerable para terminar, pues el archivo de índice está diseñado para que sea posible leerlo y entenderlo por una persona, por lo cual no está optimizada la lectura del mismo. Sin embargo, es posible modificar la estructura del mismo, consiguiendo que su lectura y procesamiento sea mucho más rápido.

El proceso para determinar los paquetes necesarios para una región solicitada se realiza mediante la utilización de algunos ciclos anidados. Si bien, este proceso se ejecuta de manera rápida y sin consumir demasiados recursos, es posible hacer algunas mejoras a dicha función que permitan mejorar el tiempo de cálculo.

Actualmente, se está manejando una política de limpieza de caché de paquetes basada en el menos recientemente utilizado, sin embargo, existe la posibilidad de proponer y poner a prueba mecanismos más óptimos, que estén directamente relacionados con el modo en que un usuario realiza la navegación sobre la imagen. También se plantea el desarrollo y puesta en funcionamiento del administrador de memoria gráfica, auxiliar a la interfaz de usuario, el cual permitiría mejorar notablemente la experiencia del usuario cuando éste navegue sobre la imagen, asimismo, está vigente la posibilidad de investigar mecanismos óptimos para la limpieza de esta memoria.

Existe la posibilidad de implementar un mecanismo de soft-caché, en el cual, se almacenan los coeficientes wavelet generados a partir de la decodificación de una región. De este modo, si el usuario realiza un acercamiento sobre la imagen, no es necesario realizar la decodificación completa de la nueva región de visualización, sino que pueden reutilizarse dichos coeficientes, así solo sería necesario aplicar las transformadas wavelet y de color, procesos que son mucho más livianos que realizar el proceso completo de decodificación. Esta estrategia permitiría mejorar los tiempos de respuesta en las operaciones de acercamiento (zoom in) y alejamiento (zoom out).

En la implementación actual se está utilizando un decodificador basado en software. Ahora bien, dada la flexibilidad que ofrece el sistema, es posible utilizar un decodificador basado en hardware; de esta manera se favorecería significativamente el rendimiento del sistema, obteniendo mejores tiempos de respuesta.

Bibliografía

[1] M. Iregui B. Macq A. Descampe, C. De Vleeschouwer and F. Marques. Pre-fetching and caching strategies for remote and interactive browsing of jpeg2000 images. *IEEE Transactions on Image Processing*, 16(5):1339–1354, 2007.

[2] A. Miede J. Eckert R. Steinmetz A. Papageorgiou, J. Blendin. Study and comparison of adaptation mechanisms for performance enhancements of mobile web service consumption. In *IEEE 6th World Congress on Services*, 2010.

[3] M.D. Adams and F. Kossentini. Jasper: A software-based jpeg-2000 codec implementation.

[4] V.; Wrobel T.; Lin C Chen, R.; Shen. Applying soa and web 2.0 to telecom: Legacy and ims next-generation architectures. In *IEEE International Conference on e-Business Engineering*, 2008.

[5] S. Deshpande and W. Zeng. Scalable streaming of jpeg2000 images using hypertext transfer protocol. In *Proceedings ACM Multimedia*, 2001.

[6] S. Nilekar O. Rekutin E. Agu, K. Banerjee and D. Kramer. A middleware architecture for mobile 3d graphics. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops*, 2005.

[7] Apache Foundation. Apache hadoop project: Scalable distributed computing.

[8] W. Ma H. Zhang H. Liu, X. Xie. Automatic browsing of large pictures on mobile devices. In *Proceedings of the 11th ACM international conference on Multimedia*, 2003.

[9] Muhammad Abul Hasan and Changick Kim. An automatic image browsing technique for small display users. In *11th International Conference on Advanced Communication Technology*, 2009.

[10] J. Howerton. Service-oriented architecture and web 2.0. *IT Professional*, 9:62–64, 2007.

[11] M. Iregui. Efficient strategies for navigation through very large jpeg2000 images. Master's thesis, Universite catholique de Louvain, 2008.

[12] 15444-1 ISO/IEC. Information technology - jpeg2000 image coding system - part 1: Core coding system, 2000.

[13] 15444-9 ISO/IEC. Information technology - jpeg2000 image coding system - part 9: Interactivity tools, apis and protocols, 2004.

[14] Suenaga M. Iregui C. DE VLEESCHOUWER B. Macq. J. Meessen, T. Suenaga. Layered architecture for navigation in jpeg2000 mega-images. *WIAMIS 2003 - 4th European Workshop in Image Analysis for Multimedia Interactive Services*, pages 92–95, 2003.

[15] J.M. Kahn Keang-Po Hoa. Image transmission over noisy channels using multicarrier modulation. *Signal Processing: Image Communication*, 9:159–169, 1997.

[16] Yang KH Lee CK Yoo SK Kim DK, Kim EY. A mobile tele-radiology imaging system with jpeg2000 for an emergency care. *Journal of Digital Imaging*, Volume 24, Number 4:709–718, 2011.

[17] D.; Nuzzi E. & Verdecchia K. Kurkovsky, S.; Strimple. Convergence of web 2.0 and soa: Taking advantage of web services to implement a multimodal social networking system. In *The 11th IEEE International Conference on Computational Science and Engineering*, 2008.

[18] L. Feng L. Qiao and L. Zhou. Information presentation on mobile devices: Techniques and practices. *Lecture Notes in Computer Science*, 4976/2008:395–406, 2008.

[19] Xin Fan Wei-Ying Ma Hong-Jiang Zhang He-Qin Zhou Li-Qun Chen, Xing Xie. A visual attention model for adapting images on small displays. *Multimedia Systems*, 2003.

[20] Z. Liu and X. Gao. Soa based mobile device test. In *Second International Conference on Intelligent Computation Technology and Automation*, 2009.

[21] Zhuo L. Liu Z., Zhang J. A distributed jpeg2000 coding algorithm based on mobile agent. In *International Conference on Wireless Communications & Signal Processing*, 2009.

[22] B. Macq M. Iregui, P. Chevalier. Optimal caching mechanisms for jpeg2000 communications. *EUSIPCO - European Signal Processing Conference*, 3:201–204, 2002.

[23] E. Romero M. Iregui, F. Gomez. Strategies for efficient virtual microscopy in pathological samples using jpeg2000. *Micron*, 38:700–713, 2007.

[24] P. Chevalier B. Macq M. Iregui, J. Meessen. Flexible access to jpeg2000 codestreams. In *23rd Symposium on Information Theory in the Benelux*, 2002.

[25] Inc. MicroImages. Google earth structure, 2010.

[26] M. Moshfeghi and J. Ta. Eficient image browsing with jpeg2000 internet protocol. In *Medical Imaging 2004: PACS and Imaging Informatics*, 2004.

[27] F. Le Mouel and F. Andre. Distribution over mobile environments. *Proceedings of 2000 ACM Symposium on Applied Computing*, 2:568–569, 2000.

[28] L. Auer. N. Kryvinska, C. Strauss. Next generation applications mobility management with soa. In *Proceedings of the 2010 International Conference on Complex, Intelligent and Software Intensive Systems*, 2010.

[29] M.G. Strintzis N. Thomos, N.V. Boulgouris. Optimized transmission of jpeg2000 streams over wireless channels. In *IEEE transactions on Image processing*, 2006.

[30] Fresian Natu and Lavagetto. Transmission of jpeg2000 codestreams over movile radio channels. In . *Internatioanl Conference on Image processing*, 2005.

[31] P. Kopylov S. Grohn P. Franti, E. Ageenko and F. Berger. Compression of map images for real-time applications. *Image and Vision Computing*, 22:1105–1115, 2004.

[32] J. Askelof B. Berthelot D. Bouchard-F. Henry G. Mozelle P. Onno R. Grosbois, D. Santa Cruz. jj2000 - a pure java jpeg 2000 image codec.

[33] J.Leppanen S. Toivonen R. Tergujeff, J. Haajanen. Mobile soa: Service orientation on lightweight mobile devices. In *IEEE International Conference on Web Services*, 2007.

[34] M. Rabbani and J. Rajan. An overview of the jpeg2000 still image compression standard. *Signal Processing: Image communication*, 17:3–48, 2002.

[35] U. Rauschenbach and H. Schumann. Demand-driven image transmission with levels of detail and regions of interest. *Computers & Graphics*, 23:857–866, 1999.

[36] R. Rosenbaum and H. Schumann. Jpeg2000 based viewer guidance for mobile image browsing. In *12th International Multi-Media Modelling Conference Proceedings*, 2006.

[37] S. Brandt S. Buchinger, S. Kriglstein and H. Hlavacs. A survey on user studies and technical aspects of mobile multimedia applications. *Entertainment Computing*, Volume 2, Issue 3:175–190, 2011.

[38] L. Chittaro S. Burigat and S. Gabrielli. Navigation techniques for small-screen devices: An evaluation on maps and web pages. *International Journal of Human-Computer Studies*, Vol. 66, No. 2:78–97, 2008.

[39] S. Senda N. Nakajima T. Hiroaki K. Yamada S. Wang, A. Mituhasi. Adaptive data transmission on browsing of scanned documents using jpeg2000. In *Proceedings of the IEEE Workshop on Knowledge Media Networking*, 2002.

[40] C.M. Sarraf and R.M. Wakim. Improving jpeg 2000 images delivery over gprs mobile networks. In *Proceedings of the 6th WSEAS Int. Conf. on Electronics, Hardware, Wireless and Optical Communications*, 2007.

[41] C. Schroth. Web 2.0 versus soa: Converging concepts enabling seamless cross-organizational collaboration. In *The 9th IEEE International Conference on E-Commerce Technology*, 2007.

[42] J. Senor. Service-oriented architecture: A new alternative to traditional technology integration. *iWay Software Withepaper*, 2005.

[43] D. Taubman. Kakadu software.

[44] D. Taubman. High performance scalable image compression with ebcot. *IEEE Transactions on Image Processing*, 9(7):1151–1170, 2000.

[45] D. Taubman. Remote browsing of jpeg2000 images. In *Proceedings IEEE ICIP*, 2002.

[46] D. Taubman and M.W. Marcellin, editors. *JPEG2000 Image Compression, Fundamentals, Standards and Practice*. Kluwer Academic Publishers, 2002.

[47] D. Taubman and R. Prandolinib. Architecture, philosophy and performance of jpip: Internet protocol standard for jpeg2000. *Proceedings of the SPIE*, 5150:791–805, 2003.

[48] D. Taubman and R. Rosenbaum. Rate distortion optimized interactive browsing of jpeg2000 images. *IEEE International Conference on Image Processing (ICIP2003)*, 3:765–768, 2003.

[49] E. de la Hoz I. Marsa U. Diaz, B. Alarcos. Jpeg2000 images protection sent to mobile devices. In *Proceedings of the I International Conference on Ubiquitous Computing: Applications, Technology and Social Issues*, 2006.

[50] T. Venema. The oracle it modernization series: Modernization - the path to soa. *Oracle White paper*, 2008.

[51] R Wilding. The business of soa. evolving an agile enterprise with metastorm enterprise. *Metastorm White paper*, 2007.