UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2 "TELECOMUNICACIONES"

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE

INGENIERO EN CIENCIAS INFORMÁTICAS

Título: Plataforma de Gestión de Contenidos para Dispositivos Móviles. Módulo Transcoding de contenidos.

Autor:

Enmanuel Echevarría González

Tutor:

Ing. Damián Ilizastegui Arriba.

Ciudad de La Habana, Junio de 2009

"Año del 50 Aniversario del Triunfo de la Revolución"

Declaración de Autoría

Se declara que Enmanuel Echevarría González es el único autor de este trabajo y autoriza a la Universidad de las Ciencias Informáticas (UCI) y a la Facultad 2 para que se haga el uso que sea pertinente con el mismo.

Para que así conste firmo la presente a los 22 días del mes de julio del 2009.

_____
Firma del Autor

_____
Firma del Tutor
Ing. Damián Ilizastegui Arriba.

El futuro de nuestra patria tiene que ser, necesariamente, un futuro de hombres de ciencia, de hombres de pensamiento.

Comandante en Jefe Fidel Castro Ruz

.

AGRADECIMIENTOS

Le agradezco en primer lugar a la Universidad de Ciencias Informáticas (UCI) que me ha convertido en una mejor persona profesionalmente, que me dio la oportunidad de conocer a compañeros de distintas provincias y municipios del país, que han sido durante estos cinco años un apoyo grande para mi y a los cuáles tengo que agradecerle muchísimo. Quisiera agradecerle además a los profesores que me brindaron todo su conocimiento. Un agradecimiento especial a la profesora Gilda Elena Stocker, que me ayudó mucho, sobre todo en los primeros años de la carrera. Agradecerle también a un profesor que ya no está con nosotros, Angel Alexis Céspedes Lescalle, que me brindó muy valiosos consejos que siempre llevaré conmigo. Agradecerle además a Darién Jesús Alvarez por confiar en mi varias veces en el proyecto. Agradecerle a la profesora Madelis Perez Gil que me ayudara mucho junto a mi tutor Damián Ilizastegui Arriba a la culminación del presente trabajo. A Damián en específico agradecerle por la vital ayuda brindada en la realización del módulo de transcoding, agradecerle además que me tuviera confianza y que me apoyara y sobre todas las cosas que fuera muy paciente conmigo. Quisiera agradecerle especialmente al profesor Ramón Alexander Anglada, porque fue el que me inició en el mundo del software libre y me hizo formar parte del grupo de soporte de software libre de la Factultad. Un agradecimiento, el más especial de todos a mi familia que durante cinco años me apoyó y ayudó a la culminación de los estudios, en especial a mi mamá: Dolores de la Caridad, a mi papá: Efraín Américo, a

mi hermana: Ludmila, a mis abuelas Cuca y Mima, a mis tías Mimí y Tere, a mi tío Fredy, a mi compañera de vida Marialys Gómez, a mi primer niño: Luis Manuel y al segundo que viene en camino.

DEDICATORIA

Le dedico este trabajo en primer lugar a toda mi familia, en especial a mis abuelos: Pipo y Lorenzo, que se que les hubiera gustado mucho verme graduar, a mi papá, a mi mamá, a mi compañera de vida y a mis dos niños, que han sido de vital importancia en mi presencia y permanencia en la Universidad, apoyándome siempre en cualquier problema que presentara. Y en segundo lugar le dedico este trabajo a todos mis compañeros de estudio, que sería injusto mencionarlos porque son muchos y se me quedarían.

Resumen

En estos momentos el país está incursionando en el mundo de la telefonía móvil. La encargada de brindar servicios de este tipo de tecnologías a nivel de país es ETECSA (Empresa de Telecomunicaciones de Cuba S.A), sin embargo actualmente la misma no cuenta con una plataforma automatizada que sea capaz de brindar servicios de venta de contenidos para móviles a sus clientes. Es por eso que se le propone a la facultad 2 la implementación de un producto que brinde servicios tales como: descarga de contenidos, y otros, incluidos en un portal WAP (Wireless Access Protocol). Como parte de este producto es necesario brindarle al cliente la posibilidad de descargar contenidos tales como: imagen, sonido o video. Existe un problema con la compatibilidad de los contenidos para diferentes tipos de teléfonos móviles y es que todos los teléfonos móviles no soportan los mismos tipos de formatos de ficheros, por lo que para la solución de este problema se ha hecho necesaria la inclusión de un módulo de *transcoding* que tiene como objetivo precisamente, transformar el contenido entre diferentes formatos y cambiar sus características de forma tal que se adecue a las necesidades específicas de un teléfono móvil dado. En el presente trabajo se presentan el diseño, la implementación, las pruebas realizadas, así como los artefactos generados correspondientes a la metodología de desarrollo de software RUP (Rational Unified Process).

Índice

Índice

Índice

Introducción.

La telefonía móvil se introdujo en Cuba por primera vez en el año 1991 con la creación de la Empresa Mixta CUBACEL S.A (http://www.cubacel.cu/), la cual ofreció servicios en la norma TDMA (Time Division Multiple Access) (800MHz) con cobertura nacional (11).

Posteriormente en el año 2001 se inició el servicio en la norma GSM (Global System for Mobile Communications) (900MHz) a través de la Empresa de Telecomunicaciones Celulares del Caribe (C_COM) con cobertura en la Ciudad de la Habana y Varadero (11).

En el año 2004 todos los servicios de telefonía móvil quedan a cargo de la Vicepresidencia de Servicios Móviles de ETECSA, a partir de la fusión de las empresas CUBACEL y C_COM a la Empresa de Telecomunicaciones de Cuba. A raíz de esta estructura se amplió la cobertura GSM (900 MHz) a nivel nacional y se abrieron nuevas oficinas comerciales en toda la isla (11).

En el año 2005 se realizaron 21 nuevos acuerdos de *roaming* (itinerancia: capacidad de hacer y recibir llamadas) internacional, alcanzando la cifra de 238, y se amplió la cobertura celular en la norma GSM (900MHz) con la ubicación de 51 radio bases a nivel nacional.

En el año 2006 se instalaron nuevas radio bases para prestar servicios de telefonía móvil en la norma GSM (850MHz) en la Ciudad de la Habana, Varadero, Cayo Coco y Cayo Guillermo, y se realizó el lanzamiento de nuevos servicios de voz para prepago, envío de SMS desde teléfonos con norma GSM a TDMA; envío y recepción de MMS (Multimedia Messaging Service) y la comercialización de la nueva tarjeta SIM (Subscriber Identify Module) de 64k. Se amplió la red de oficinas comerciales y de venta de tarjetas de recarga en todo el país, y en este año Cuba fue sede para la realización de la Reunión Plenaria #22 de la Asociación GSM de Latinoamérica (11).

Para enero del año 2008 se le pide a la UCI (Universidad de las Ciencias Informáticas) por parte de Dirección Central de Servicios Móviles (DCSM) crear un portal WAP (Wireless Access Protocol) que permitiera a los usuarios de CUBACEL obtener una serie de servicios.

Uno de los servicios que brinda este portal es la descarga de contenidos, ya sea de imagen, sonido o video desde el portal hacia el teléfono móvil del cliente. Aquí en este punto radica un problema y es que todos los teléfonos móviles no soportan los mismos formatos de estos tipos de archivos, por lo que se hace necesario poder transformar los formatos de los contenidos que deseen los clientes a formatos que permitan los celulares en dependencia de sus características, a este proceso se le conoce como proceso de *transcoding* por lo que el problema científico a resolver es: **¿Cómo automatizar el proceso de *transcoding* para teléfonos móviles?**

Como **objetivo general** del presente trabajo se tiene: Desarrollar el módulo de Transcoding de contenidos para la Plataforma de Gestión de Contenidos de CUBACEL.

La **función principal** que debe realizar la plataforma es la adaptación de contenido para teléfonos móviles a diferentes formatos en dependencia de las características de estos dispositivos.

A partir del **problema científico** se define como el **objeto de estudio** para la investigación todo lo referente a los procesos de *transcoding*; mientras que el **campo de acción** estaría enmarcado en la realización del *transcoding* para la telefonía móvil.

A partir de la necesidad de realizar un módulo que garantice el *transcoding* de contenidos, se plantea como **idea a defender** que, la implementación de un sistema de *transcoding* para la Plataforma de Gestión de Contenidos para dispositivos móviles, contribuirá a que los clientes puedan acceder a cualquier tipo de contenido que deseen, sin importarles las características de su móvil.

Para el cumplimiento del **objetivo principal** se propone la siguiente tarea de investigación:

I. Investigación sobre las herramientas de *transcoding* existentes en el mercado.

El posible resultado una vez concluido el estudio, es obtener un producto funcional y de calidad que sea capaz de realizar servicios de adaptación y manipulación de contenidos para cualquier modelo o tipo de móvil que necesite de los servicios del portal WAP de CUBACEL.

El trabajo constará de 5 capítulos:

I. **Fundamentación Teórica**, donde se analizarán las principales herramientas de *transcoding* existentes en el mercado, las metodologías más utilizadas en el proceso de desarrollo de software, así como las herramientas necesarias para la realización del software.

II. **OMA STI**, donde se analizarán las principales características del estándar internacional que regirá la construcción del módulo de *transcoding* de contenidos.

III. **Alembik**, donde se realizará una investigación de la herramienta que se utilizará como base para el desarrollo del módulo.

IV. **Propuesta del sistema**, donde se incluye todo lo concerniente a la ingeniería de software, dígase, modelo de Negocio, Levantamiento de Requisitos, Análisis y Diseño, Implementación y las Pruebas.

V. **Estudio de la factibilidad**, donde se hará la estimación del esfuerzo total del proyecto, así como su factibilidad a la hora de realizarlo.

Capítulo 1

Fundamentación Teórica

1.1 Introducción.

En el presente capítulo se analizarán las principales herramientas de *transcoding* existentes en el mercado para conocer si son utilizables o no en el desarrollo del software. También se definirá la metodología de desarrollo de software a aplicar para la elaboración del producto, así como las herramientas necesarias para su confección a través de un análisis de las principales características de las mismas.

1.2 Transcoding.

Transcoding es un servicio que se brinda para adaptar los archivos digitales de manera que los contenidos se puedan ver en diferentes dispositivos de reproducción. Trabajando como un intérprete, un *transcoder* traduce los archivos a un formato adecuado para el usuario final. Las traducciones se basan en complicados cálculos algorítmicos y requieren importantes recursos de procesamiento (1).

Los servidores de *transcoding* y de servicios reformatean los materiales y servicios que de otro modo tienen que ser desarrollados por separado para diferentes plataformas. Son comúnmente usados para adaptar contenidos para dispositivos móviles o servidores de video. Hay diferentes maneras en que puede llevarse a cabo el *transcoding* pero el proceso general sigue siendo el mismo: el formato de origen se traduce en un formato crudo intermedio para volver a traducirse en un formato que el usuario final del dispositivo reconoce (1).

En un ejemplo, el material original es analizado por un programa que crea una versión separada que contiene anotaciones. Las anotaciones incluyen información que dará instrucciones a la reformulación del proceso. Cuando una solicitud de archivo se envía al servidor, el servidor presenta la versión anotada de

una aplicación de autoría. El material es reformateado allí y enviado al servidor proxy. El servidor proxy accede a la información sobre las preferencias del dispositivo y adapta el material como sea necesario antes de la entrega del usuario final (1).

Hay dos opciones principales para los que deseen automatizar la reformulación de los contenidos: la utilización de un producto de servidor de *transcoding* o utilizando un servicio de *transcoding* (1).

En el país no existe una herramienta capaz de realizar el proceso de *transcoding* para servicios de telefonía móvil, que sea capaz de adaptarse a las necesidades correspondientes al módulo del proyecto que se está analizando, por lo que se hace necesario realizar una búsqueda de las principales herramientas que existen en el mundo, con la cual se pueda trabajar.

En la siguiente sección se presentarán algunas herramientas existentes en el mercado en la actualidad.

1.2.1 Herramientas de *Transcoding.*

A continuación se muestra una tabla comparativa de las principales herramientas de *transcoding* existentes en el mundo.

Nombre	Descripción	Características	Aplicaciones	Formatos Soportados
MediaCoder	*Transcoder* de contenidos libre, integra los formatos de audio y video más populares.	Conversión entre los formatos más populares de audio y video, posee control total sobre los parámetros de *transcoding*, posee una simplificada interfaz de usuario (UI) para dispositivos móviles populares.	Provee compresión / reducción de tamaño para archivos de audio y video, provee conversión de audio/video para dispositivos de reproducción de este tipo de contenido (ej. Reproductores de audio digitales, reproductores MP4, teléfonos móviles, entre otros.	MP3, AMR, WMA, RealAudioXviD, DivX, MPEG 1/2/4, RealVideo, Windows Media Video ,AVI, MPEG/VOB, Matroska, MP4, OGM,CD, DVD, VCD, SVCD, etc.

GX Transcoder 4.0.1.40	Herramienta gratuita para convertir entre numerosos formatos de audio, a pesar de ser gratuito es tan o más potente que otros conversores de pago.	Potente, flexible, permite ajustar en todas las operaciones de extracción y conversión la velocidad y calidad del resultado, sencillo de utilizar.	Además de las funciones de conversión de formatos de audio, también puede ser usado como reproductor de audio.	MP1, MP3, MP2, MPC, OGG Vorbis, PCM, G.721, G.723, G.726, G.729, ADPCM, GSM, VOX, DSP, VQF, RAW, QuickTime MOV, QT, etc.
VoDKA Batch Transcoder	Sistema distribuido de gestión de colas de transcoding para la adaptación de medios a diferentes perfiles de acceso.	Es un producto escalable y flexible. Escalable porque permite abordar proyectos de cualquier número de usuarios y hacerlo de manera progresiva a la medida que las necesidades de los clientes aumentan y flexible porque su arquitectura se ajusta a la red del cliente sin que éste tenga que hacer modificaciones en la misma fruto del nuevo servicio a montar.	Es posible la utilización de diferentes codecs de vídeo para, a partir de vídeos maestros (en alta calidad) generar otros perfiles de utilización para *streaming* (en otras calidades), como por ejemplo: conversión de video y audio a formatos para la realización de *streaming* a través de internet: Flash video, MPEG-4, H.264, entre otros.	AVI, MOV, MP4, MOV, 3GPP, MPEG-TS, MPEG-PS, MPEG-2, MPEG-4, H.264, Flash video, Windows Media, MP3, AAC, AMR
Alembik	Aplicación Java que provee servicios de *transcoding* para diferentes tipos de contenidos (imagen, audio, video, etc)	Es una herramienta de código abierto, libre, está basada en el estándar de *transcoding* que provee OMA (Open Mobile Alliance) STI (Standard Transcoding Interface)	Herramienta de *transcoding* muy aplicada en la telefonía móvil.	JPEG, PNG, GIF, TIFF, BMP, MPEG3, AMR, MIDI, WAV, MMF, MPG4, WMV, AVI

Una vez realizado la comparación entre las principales herramientas y tomando en cuenta de que: el lenguaje que se va a utilizar será Java (véase sección 1.3.2) y que el sistema que se propone debe ser capaz de realizar el proceso de *transcoding*, tanto a audio, video como a imágenes pues se propone la herramienta Alembik (véase capítulo 3), que será analizada en capítulos posteriores.

1.3 Metodologías y Lenguajes.

Las metodologías son el conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software. En el proceso de desarrollo de software la metodología define *quién* debe hacer *qué*, *cuándo* y *cómo* debe hacerlo para obtener los distintos productos parciales y finales. Es por eso que es necesaria su utilización en la confección del módulo de *transcoding,* pues guiará todo el proceso de desarrollo.

Actualmente se habla de metodologías tradicionales o robustas como Rational Unified Process (RUP) (http://www-01.ibm.com/software/rational/), Microsoft Solutions Framework (MSF) (http://msdn.microsoft.com/es-es/library/ms195024(VS.80).aspx) y Métrica o metodologías ágiles como Extreme Programming (http://www.extremeprogramming.org/) , Scrum (http://www.scrum.com/), Crystal Methods (http://www.thecrystalmethod.com/) y Feauture Driven Development (http://www.featuredrivendevelopment.com/) . La metodología a utilizar será la metodología Rational Unified Process (RUP) que es un proceso de desarrollo de software configurable que se adapta a través de los proyectos variados en tamaños y complejidad. Se basa en muchos años de experiencia en el uso de la tecnología orientada a objetos en el desarrollo de software, de misión crítica en una variedad de industrias por la compañía Rational.

1.3.1 Proceso Unificado de Modelado (RUP).

El Proceso Unificado guía a los equipos de proyecto en cómo administrar el desarrollo iterativo de un modo controlado mientras se balancean los requerimientos del negocio, el tiempo al mercado y los riesgos del proyecto. El proceso describe los diversos pasos involucrados en la captura de los requerimientos y en el establecimiento de una guía arquitectónica lo más pronto, para diseñar y probar el sistema hecho de acuerdo a los requerimientos y a la arquitectura. El proceso describe qué entregables producir, cómo

desarrollarlos y también provee patrones. El proceso unificado es soportado por herramientas que automatizan entre otras cosas, el modelado visual, la administración de cambios y las pruebas. Se caracteriza básicamente por ser vital la captura de requisitos, iteración actual condicionada por la anterior, se necesita de un buen líder de proyecto para garantizar el trabajo del equipo de desarrollo, se realiza un gran número de artefactos lo que puede provocar retrasos por mala preparación de los analistas, las responsabilidades están divididas y es aplicable a todo tipo de proyecto asumiendo sus extensiones.

1.3.2 Lenguaje de Programación.

La realización del software requiere de un lenguaje de programación. En la actualidad existen varios tipos de lenguajes entre los que se encuentran, los lenguajes de máquina, los lenguajes de bajo nivel, y los lenguajes de alto nivel. Para el caso específico de la aplicación que se está analizando es necesaria la realización de la misma con un lenguaje de alto nivel.

El lenguaje que se utilizará será **Java**. Plataforma de software desarrollada por Sun Microsystems. Esta plataforma ha sido desarrollada de tal manera que los programas desarrollados para ella puedan ejecutarse de la misma forma en diferentes tipos de arquitecturas y dispositivos computacionales (28).

La *plataforma Java* consta de tres partes principales:

- El lenguaje de programación.
- Máquina virtual de Java.
- API Java.

Java supone un significativo avance en el mundo de los entornos de software, y esto viene avalado por tres elementos claves que diferencian a este lenguaje desde un punto de vista tecnológico:

- Es un lenguaje de programación que ofrece la potencia del diseño orientado a objetos con una sintaxis fácilmente accesible y un entorno robusto y agradable.
- Proporciona un conjunto de clases potente y flexible.
- Incorpora sincronización y manejo de tareas en el lenguaje mismo e incorpora interfaces como un mecanismo alternativo a la herencia múltiple de C++ (28).

Java permite escribir programas de interfaz gráfica o textual. Además se pueden correr programas de manera incorporada o embebida en los navegadores web de Internet, aunque esto nunca llegó a popularizarse como se esperaba en un principio (28).

Los programas en Java generalmente son compilados a un lenguaje intermedio o *bytecode*, y luego interpretados por una maquina virtual de java: JVM (Java Virtual Machine). Esta última sirve como una plataforma de abstracción entre la máquina y el lenguaje permitiendo que se pueda "escribir el programa una vez, y correrlo en cualquier lado". También existen compiladores nativos de Java, tanto comercial como libre (28).

1.4 Herramientas y Tecnologías actuales.

1.4.1 SVN. Control de versiones.

SVN (http://subversion.tigris.org/) es un sistema de control de versiones usado para que varios desarrolladores puedan trabajar en un mismo proyecto en forma más o menos ordenada. Tiene una arquitectura cliente servidor con controles de concurrencia para cuando varios desarrolladores están trabajando en el mismo archivo y funciona más o menos así. En algún servidor se monta un repositorio SVN. En este lugar se van a registrar los cambios (revisiones) y los *logs* que se vayan generando. El cliente de SVN se baja una copia local de alguna revisión (generalmente la última), el desarrollador hace los cambios y los sube al servidor para que estén disponibles para los otros desarrolladores (además de generar un log con un comentario de qué cosa modificó, para qué, etc.) (2).

1.4.3 Visual Paradigm – UML.

La herramienta case que se utilizará será Visual Paradigm (http://www.visual-paradigm.com/), que es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Esta herramienta UML CASE ayuda a

una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas UML en Eclipse, realizar ingeniería inversa desde código Java a diagramas de clases, generar código Java y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML, posee SDE para Eclipse, que es una herramienta/extensión UML CASE ampliamente integrada con Eclipse. (3).

1.5 Plataforma de Desarrollo: JEE.

Java Platform, Enterprise Edition o Java EE (anteriormente conocido como Java 2 Platform, Enterprise Edition o J2EE hasta la versión 1.4), es una plataforma de programación—parte de la plataforma Java—para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java con arquitectura de niveles distribuida, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. La plataforma Java EE está definida por una *especificación*. Similar a otras especificaciones del Java Community Process, Java EE es también considerada informalmente como un estándar debido a que los suministradores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son *conformes a Java EE* (4).

Una de las características del desarrollo de aplicaciones con JEE es que gracias a una serie de librerías como son EasyMock (http://easymock.org/), JUnit (http://www.junit.org/), TestNG (http://testng.org/), etc..., los programadores crean las aplicaciones a partir de pruebas, es decir, se va construyendo un entorno que obliga al programador a implementar las entidades necesarias para que estas pruebas cumplan su cometido (4).

1.6 Herramientas de desarrollo.

1.6.1 Eclipse Ganymede.

Eclipse es una herramienta que ha alcanzado un grado de madurez enorme. Tiene herramientas para desarrollar aplicaciones de consola, aplicaciones Web y Web Services con diferentes servidores de aplicaciones. Hay componentes para desarrollar aplicaciones en lenguajes de scripting como Perl, Python y Ruby, y otros para C/C++ (5).

Con la salida de la versión Eclipse Ganymede (http://eclipse.org/ganymede) el manejo de repositorios SVN y varios relacionados con desarrollo de aplicaciones web están muy mejorados. Hay un editor de Java Script muy capaz y el explorador de SVN "Subversive" es mucho mejor que el plugin "Subclipse". Los elementos relacionados con la instalación de *plugins* han sido mejorados notablemente. Se mejoró el manejo de dependencias y ahora se puede "navegar" por los componentes que están disponibles para instalar en cada repositorio de componentes. La ayuda al instalar nuevos componentes fue mejorada. Si es instalado un nuevo componente como el Subversive, se agrega a la ayuda de una forma más destacada, para que se pueda aprender a usar las nuevas funciones que recién fueron instaladas (5).

1.6.2 JUnit.

Cuando se prueba un programa, el mismo se ejecuta con unos datos de entrada (casos de prueba) para verificar que el funcionamiento cumple los requisitos esperados. Se define como prueba *unitaria* a la prueba de uno de los módulos que componen un programa (6).

En los últimos años se han desarrollado un conjunto de herramientas que facilitan la elaboración de pruebas unitarias en diferentes lenguajes. Dicho conjunto se denomina *XUnit*. De entre dicho conjunto, **JUnit** es la herramienta utilizada para realizar pruebas unitarias en Java (6).

El concepto fundamental en estas herramientas es el caso de prueba (*test case*), y la suite de prueba (*test suite*). Los casos de prueba son clases o módulos que disponen de métodos para probar los métodos de una clase o módulo concreta/o. Así, para cada clase que se quiera probar, se definiría su correspondiente clase de caso de prueba. Mediante las suites se pueden organizar los casos de prueba, de forma que cada suite agrupa los casos de prueba de módulos que están funcionalmente relacionados (6).

Las pruebas que se van construyendo se estructuran así en forma de árbol, de modo que las hojas son los casos de prueba, y podemos ejecutar cualquier sub árbol (suite) (6).

De esta forma, se construyen programas que sirven para probar los módulos, y que podrán ejecutarse de forma automática. A medida que la aplicación vaya avanzando, se dispondrá de un conjunto importante de casos de prueba, que servirá para hacer pruebas de regresión. Eso es importante, puesto que cuando se

cambia un módulo que ya ha sido probado, el cambio puede haber afectado a otros módulos, y sería necesario volver a ejecutar las pruebas para verificar que todo sigue funcionando (6).

Aplicando lo anterior a Java, JUnit es un conjunto de clases *open source* (de código abierto) que permiten probar las aplicaciones Java (6).

1.7 Conclusiones parciales

En este capítulo se presentaron las principales herramientas de transcoding existentes en el mercado, donde se vieron varios ejemplos y dentro de todos ellos la que fue escogida para trabajar fue **Alembik**.

Se vieron también algunas de las principales metodologías que existen para el desarrollo del software, de las cuales se escogió la metodología de proceso unificado de modelado (RUP).

Finalmente se escogieron las principales herramientas tanto para el desarrollo del software como para el trabajo en general del mismo.

Capítulo 2

OMA STI

2.1 Introducción

El despliegue de aplicaciones multimedia requiere en muchos casos, de adaptación de contenidos debido a la diversidad de especificaciones de teléfonos móviles (tamaño de la pantalla, resolución, profundidad del color, así como los formatos de contenidos que soportan).

STI (http://www.openmobilealliance.org/Technical/release_program/sti_v10.aspx)(Standard Transcoding Interface) 1.0 es la primera especificación de una interfaz estándar entre Plataformas de Aplicación y una Plataforma de Transcodificación lo que significa la solución a problemas de integración y prueba cuando son desplegados los servicios multimedia a través de los servicios móviles.

Precisamente en el presente capítulo se abordará este estándar de *transcoding*, que como se verá después es el estándar por el que se rige la herramienta que se utilizará para la implementación del sistema: Alembik (véase capítulo 3).

2. 2 Introducción a la interfaz de *transcoding*

La figura 2 muestra de forma resumida la interacción entre una Plataforma de Aplicación y una Plataforma de Transcodificación.

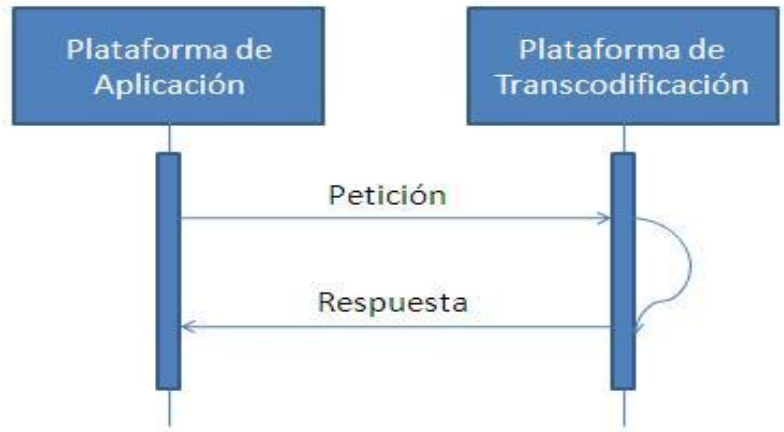


Figura 2 Operación de Transcodificación.

La comunicación entre la Plataforma de Aplicación y la Plataforma de Transcodificación está basada en la estructura petición – respuesta, donde una sesión de *transcoding* siempre es una petición seguida de una respuesta. La sesión solo se cerrará cuando ha sido recibida la respuesta. La Plataforma de Aplicación realizará una petición de *transcoding* a la Plataforma de Transcodificación. La Plataforma de Transcodificación recibe la petición, realiza las operaciones necesarias y genera una respuesta que devuelve hacia la Plataforma de Aplicación.

2.2.1 Elección del Protocolo.

La interfaz entre la Plataforma de Aplicación y la Plataforma de Transcodificación es realizada bajo la versión del protocolo SOAP (Simple Object Access Protocol) 1.1 sobre el protocolo HTTP(s).

La petición SOAP será enviada como parte del cuerpo de una petición POST de HTTP(s), mientras que la respuesta SOAP será enviada como parte del cuerpo de una respuesta HTTP(s).

2.2.2 Estructura de una petición de *transcoding*.

La figura 2.1 muestra un acercamiento a los componentes de una petición. Las peticiones pueden ser realizadas de dos maneras: como un trabajo de *transcoding* individual, la cual contiene solamente la

petición de *transcoding* de un solo contenido, o como un cuerpo de peticiones, que contiene varios trabajos de *transcoding*.

Una petición de *transcoding* contiene los contenidos que deben ser adjuntados dentro de un mensaje HTTP(S) POST.

La figura 2.1 muestra dos configuraciones para una petición de *transcoding* dentro de una petición HTTP POST. La primera es una petición HTTP con un cuerpo de múltiples partes MIME (*Extensiones Multipropósito de Correo Internet*) que contiene el mensaje SOAP como la primera parte del cuerpo y los contenidos adjuntos como las restantes partes del cuerpo. La segunda es una petición con un simple cuerpo XML (Extensible Markup Language), el cual es un mensaje SOAP que contiene el cuerpo de una petición STI**.** En la petición no hay contenidos adjuntos, pero hace referencias a contenidos externos.

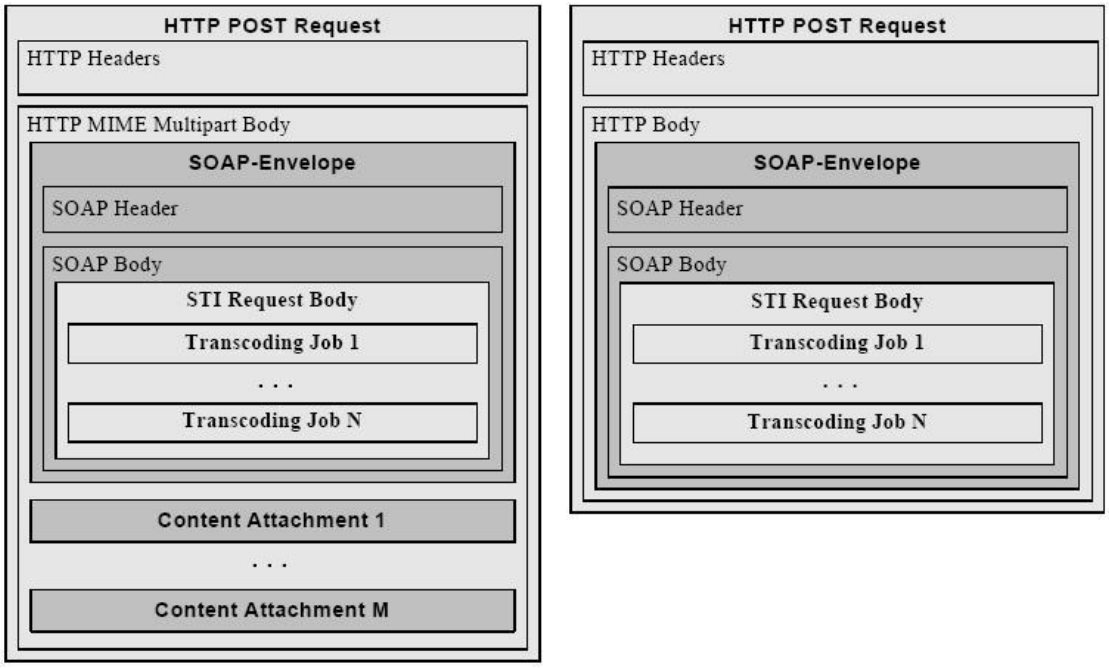


Figura 2.1 Estructura de dos configuraciones de una petición de transcoding dentro de una petición HTTP POST.

La cabecera SOAP (primera parte de la envoltura SOAP) es opcional. La versión 1.0 de STI no define ningún elemento para la cabecera SOAP.

SOAP provee un mecanismo flexible para extender un mensaje en una forma descentralizada y modular sin previo conocimiento entre las partes de comunicación. Ejemplos típicos de extensiones que pueden ser implementadas como entradas de cabeceras SOAP son: autenticación, administración de transacciones, pago, etc. SOAP define también algunos atributos que pueden ser usados para indicar quien debe tratar con una característica ya sea opcional u obligatoria.

Cada trabajo de *transcoding* en el cuerpo de la petición debe contener los parámetros fuentes y también debe contener los parámetros de lo que se quiere lograr como se muestra en la figura 2.2

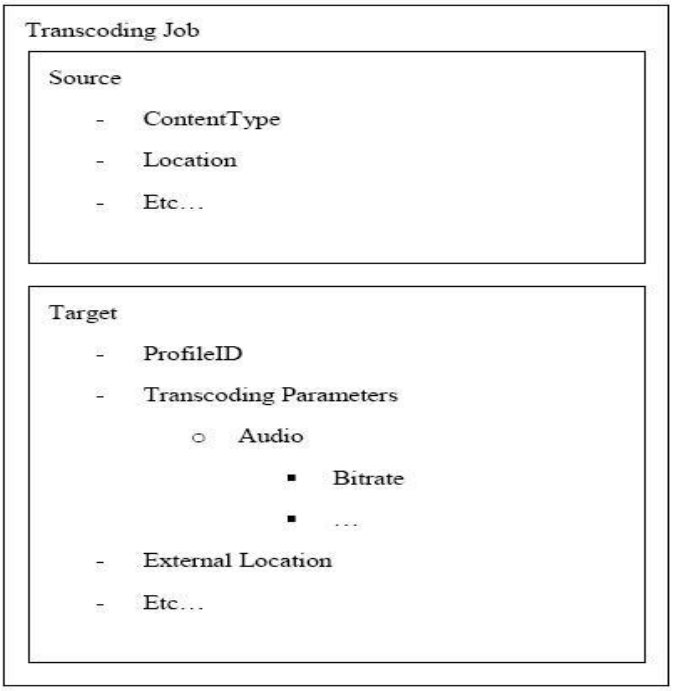


Figura 2.2 Estructura de un trabajo de transcoding.

2.2.3 Respuesta al proceso de transcoding.

La respuesta al proceso de *transcoding* (la respuesta de la Plataforma de Transcodificación a la Plataforma de Aplicación) contiene los resultados de los trabajos. La estructura de una respuesta de *transcoding* está detallada en la figura 2.3

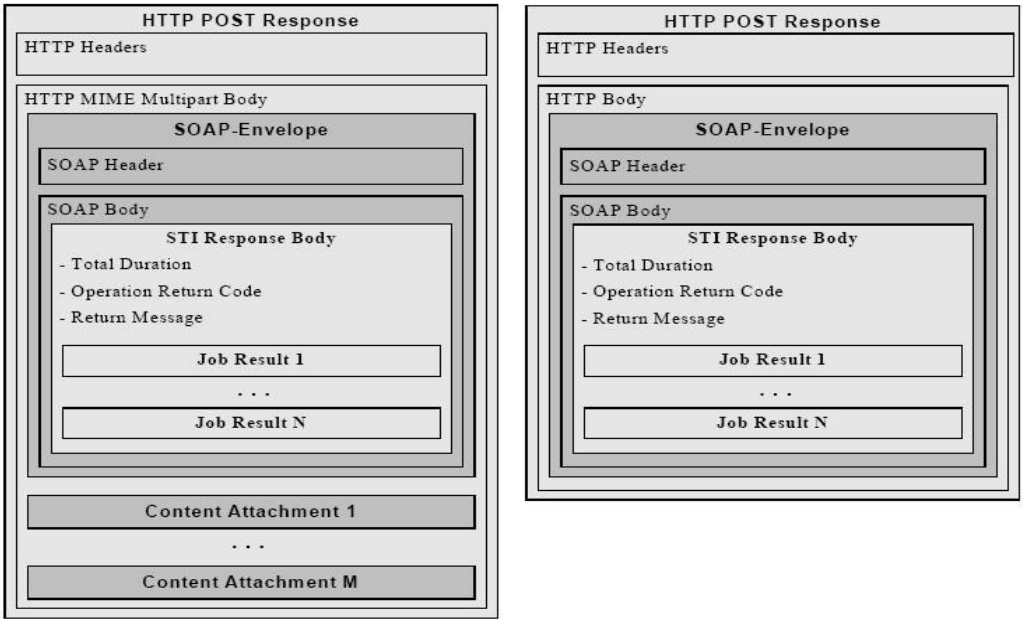


Figura 2.3 Estructura de una respuesta de transcoding.

En el cuerpo de la respuesta, los parámetros guardan relación a todo lo concerniente a la operación de transcoding o al resultado de un trabajo de *transcoding* individual. Por ejemplo, el parámetro de duración total corresponde a la duración completa de la operación. En el resultado del trabajo hay parámetros que describen en particular el proceso de *transcoding* que se llevó a cabo (ej. retorno del código del trabajo, retorno de una cadena de mensajes y la duración). Los resultados de los trabajos contenidos en la respuesta del transcoding se muestran en la figura 2.4

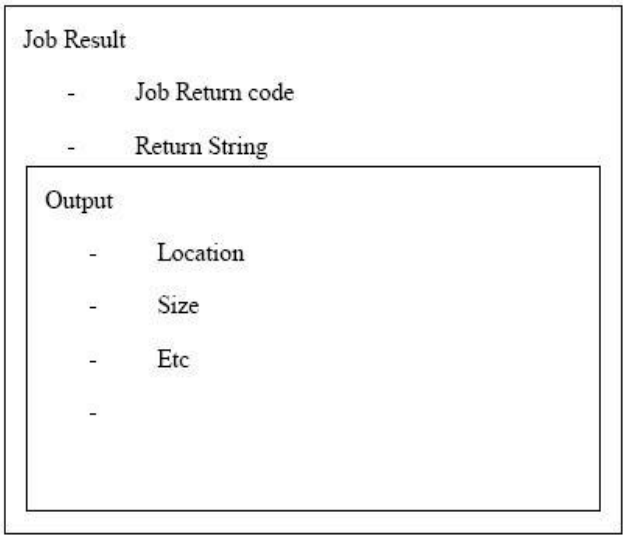


Figura 2.4 Estructura del resultado del trabajo.

2.2.4 Dependencias del bloque fuente y el bloque objetivo

El bloque de archivos fuentes de cada trabajo de transcoding deberá especificar el tipo de contenido y la ubicación del archivo. Debe además especificar parámetros adicionales que describan la fuente del contenido. Si presenta el bloque objetivo (*tarjet*) de cada trabajo de *transcoding* debe especificar explícitamente los parámetros de *transcoding* y/o la información del perfil y/o URI (Uniform Resource Identifier).

2.2.5 Contenidos adjuntos

Los datos de los contenidos deben ser referenciados dentro del cuerpo de petición de SOAP (y en el cuerpo de respuesta) y deben estar almacenados donde puedan ser accedidos por la Plataforma de Transcodificación, o adjuntados como una parte misma de la petición/respuesta de *transcoding*. La interfaz propuesta (STI 1.0) provee dos métodos que soportan la adjunción de contenidos:

1. Contenedor propio de petición/respuesta, en el cual los datos de los contenidos residen dentro de la petición/respuesta propiamente.

2. Referencias a elementos externos, en cuyo caso un trabajo de transcoding contiene un puntero a una ubicación remota desde donde el contenido puede ser halado por la plataforma de transcoding.

Las siguientes dos secciones describen cada uno de estos métodos. Nótese que las dos opciones pueden ser combinadas dentro de una simple operación: una petición de transcoding puede contener trabajo (s) teniendo referencias (URIs) externas a los elementos de los contenidos y trabajo (s) teniendo adjunto los elementos del contenido.

2.2.5.1 Referencias a los elementos de contenidos externos.

En el caso de que las referencias a elementos de contenidos externos sean usadas, el cuerpo de SOAP petición/respuesta debe contener la URIs apuntando a los ficheros relevantes.

La Plataforma de Transcodificación debe soportar la recuperación y la subida de contenidos externos vía HTTP (s). La Plataforma de Transcodificación debe también soportar otros protocolos, por ejemplo, FILE (se refiere a sistema de fichero) y FTP (File Transfer Protocol) (o cualquiera de sus variantes seguras). En el caso de que el protocolo de subida de contenidos requerido sea HTTP(s), la Plataforma de Transcodificación debe usar el método HTTP(s) PUT para poder realizar la subida de los mismos.

2.2.5.2 Contenedor propio de petición/respuesta.

En el caso de un contenedor propio de petición/respuesta, el cuerpo SOAP de petición/respuesta debe contener referencias a los adjuntos. Los adjuntos deben ser enviados delante con la petición de *transcoding* y la respuesta de *transcoding* como partes MIME. Cada cabecera de contenido adjunto (en ambos casos: petición de *transcoding* y respuesta de *transcoding*) debe contener un único ID.

2.2.6 Perfiles, parámetros, políticas y adaptación de clases de transcoding.

Para especificar los detalles de la petición de *transcoding*, la Plataforma de Aplicación debe usar la información de perfiles predefinidos y/o parámetros explícitos de *transcoding*, y además debe especificar una URI para la política de parámetros. Una vez recibida la petición de *transcoding*, la Plataforma de Transcodificación debe usar la información de los perfiles, los parámetros explícitos de *transcoding* y las políticas de parámetros en el orden que están definidos en esta especificación (STI).

2.2.6.1 Usando un perfil predefinido.

Los parámetros de *transcoding* deben ser indicados usando un parámetro identificador de perfil (profileID) para referenciar un perfil predeterminado.

El perfil representa información a ser considerada cuando se lleve a cabo el *transcoding*. A menudo, el perfil describirá las características de usuario del equipo.

El perfil puede ser referenciado usando la cabecera de un usuario-agente (*user-agent*), una cadena UAProf (User Agent Profile) URL (Uniform Resource Locator), UAProf, o una cadena propiedad. La cabecera del *user-agent* forma parte normalmente de las transacciones HTTP/WSP entre el usuario del equipo y la aplicación servidora. Esta cabecera es la más usada. La información UAProf es expresada como un enlace (URL) a la ubicación de la detallada información terminal. Las referencias de UAProf URL deben hacer referencia a una base de datos UAProf que las dos plataformas (Plataforma de Aplicación y Plataforma de Transcodificación) comparten.

La Plataforma de Transcodificación debe soportar el uso del *user-agent* o del UAProf URL como valores del parámetro profileID.

2.2.6.2 Usando parámetros explícitos de *transcoding*.

Los parámetros de *transcoding* deben ser también indicados usando una lista de parámetros explícitos, en la cual se reflejarán las características del dispositivo y/o especificados los requerimientos de la Plataforma de Aplicación.

Cuando un perfil es indicado y los parámetros explícitos son añadidos, los parámetros explícitos deben sobrescribir los parámetros correspondientes en el perfil referenciado.

2.2.6.3 Usando la política de parámetros.

La política de parámetros significa para la Plataforma de Aplicación especificar reglas para la petición de *transcoding*, las que definen el comportamiento de la Plataforma de Transcodificación (ej. orden entre los diferentes contenidos, definir, cuando o no usar un perfil por defecto, etc.). Solamente una referencia (URI) a la información de las políticas externas está estandarizada en la versión STI 1.0 (parámetro policyRef). Esto significa que una serie de políticas pueden ser referenciadas en una petición de *transcoding* provenientes de una Plataforma de Aplicación. La Plataforma de Transcodificación debe tomar estas referencias de políticas en consideración.

2.2.6.4 Usando la adaptación de clases

Hay situaciones donde la plataforma de aplicación necesita describir a la Plataforma de Transcodificación las clases de adaptaciones y especificar si son permitidas o no. En una aplicación MMS, el *mms Proxy/relay* puede necesitar describir a la plataforma de *transcoding* cuáles constituyen las “mayores” adaptaciones (en el contexto de MMS) y especificar cuáles no son permitidas. El *mms* Proxy/relay puede también querer aprender si las adaptaciones menores fueron realizadas. STI provee un mecanismo que permite a la Plataforma de Aplicación controlar cuáles adaptaciones de clases son permitidas y cuáles no, y averiguar cuáles adaptaciones de clases fueron realizadas.

La petición de *transcoding* de STI contiene parámetros que especifican cuáles adaptaciones de clases son de interés a la Plataforma de Aplicación y controlan cuales son permitidas (ej. adaptaciones menores o mayores de MMS son de interés y la mayor no es permitida).

La respuesta de *transcoding* contiene una lista de adaptaciones de clases, las cuales se asocian con la operación actual de *transcoding* realizada (ej. si menor, o mayor o ambas clases de adaptaciones han sido realizadas).

Una Plataforma de Transcodificación no debe llevar a cabo ninguna adaptación que no esté permitida.

2.2.7 Diagrama UML

El diagrama UML del anexo 1 brinda un acercamiento a todas las estructuras de datos usadas por el estándar STI con respecto a la petición de *transcoding* (representada como *TranscoderRequest*) y a la respuesta de *transcoding* (representada como *TranscoderResponse*).

2.3 Conclusiones parciales

En el capítulo finalizado se abordaron aspectos muy importantes para el trabajo con el estándar de *transcoding* (STI) brindado por OMA y que contiene como estructuras fundamentales la petición de *transcoding* y la respuesta de *transcoding*, viéndose de cada una de ellas sus partes fundamentales.

Capítulo 3

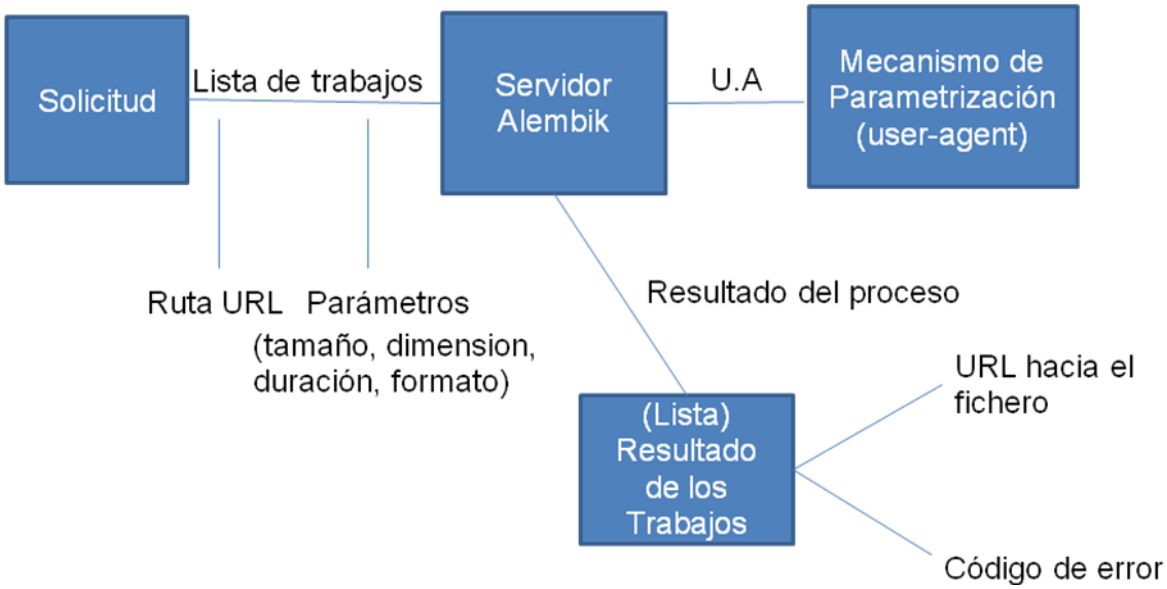
Alembik

3.1 Introducción

Como fue mencionado en el capítulo 1 en la sección 1.2.1, el servidor de *transcoding* de contenido: Alembik (http://alembik.sourceforge.net/), es una aplicación Java que provee servicios de *transcoding* para diferentes tipos de contenidos (imagen, audio, video, texto.). En la figura 3.1 se representa el proceso de *transco*ding llevado a cabo por el Alembik. En general, el servidor espera por las solicitudes, las cuales contienen una lista de trabajos de *transcoding*. Cada trabajo de *transcoding* contiene una ruta URL hacia un determinado contenido y cada contenido posee una serie de parámetros en dependencia del formato al que debe ser transformado.

Los parámetros de *transcoding* dependen del tipo de fichero entrado. Deben especificar, por ejemplo, el tamaño, la dimensión, la duración y el formato que se desea como salida. El servidor Alembik ofrece dos mecanismos de parametrización del lado del servidor. La primera opción es pasar un identificador *user-agent*, que resolverá las capacidades del dispositivo dentro de una serie de parámetros. La segunda opción es asociar un trabajo con el id de un parámetro predefinido, llamado perfil, cuya lista es internamente mantenida en el servidor.

Un vez que el servidor obtiene y valida una petición, inicia el proceso de *transcoding*. El resultado retornado al que le realiza la petición (aplicación cliente) es siempre un objeto respuesta que contiene una lista de resultados para todos los trabajos de *transcoding*. Generalmente, cada resultado contiene un puntero URL hacia la localización del fichero al que se le ha realizado el proceso de *transcoding* o un código de error en caso de que falle el proceso donde se plasma la razón por la cual el trabajo de *transcoding* falló.



3.1 – Proceso de *transcoding* del servidor Alembik.

El servidor realiza un proceso síncrono, o sea, espera por la realización de todos los procesos de *transcoding* antes de enviar una respuesta.

La arquitectura del Alembik se basa totalmente y es compatible con el estándar OMA STI versión 1.0 (véase capítulo 2).

El acceso a las funcionalidades de *transcoding* para clientes Java es posible a través de dos vías diferentes. La primera es a través del API (Application Programming Interface) *Java Transcoding* con la clase *org.alembik.TranscodingManager* que sirve como un punto de entrada. La segunda posibilidad es usar la librería JSP (Java Server Page) *Transcoding Tag* que provee soporte para el desarrollo de páginas WAP. Actualmente, los componentes de la librería también hacen uso del *TranscodingManager*.

Basado en su actual configuración el *TranscodingManager* se conecta al servidor a través de llamadas RMI (Java Remote Method Invocation) o peticiones SOAP. El modo de conexión está predeterminado

durante la generación de la librería *TranscodingClient*, la cual consiste en una serie de archivos JAR (Java Archives) construidos y empaquetados para un ambiente de cliente específico.

3.2 Proceso de construcción.

3.2.1 Estructura en módulos.

El servidor Alembik está construido básicamente con los siguientes módulos:

- Alembik-api.
- Alembik-core.
- Alembik-server.
- Alembik-soap.
- Alembik-build.
- Alembik-client.
- Alembik-taglib.
- Alembik-webapp.

Cada módulo está provisto de su propio script de construcción. Estos scripts dependen de dos ficheros que están dentro del directorio: /alembik-build/etc:

- local.properties.
- *%tipodeservidor%.local.properties*, donde el %tipo de servidor % pueden ser: *glassfish (*https://glassfish.dev.java.net/ *)*, *jboss (*www.jboss.org *)* o *tomcat (http://tomcat.apache.org/)*.

Algunos de las configuraciones descritas abajo forman parte de los valores iniciales de configuración que pueden ser cambiados.

3.2.2 Plataforma del servidor de aplicación.

Inicialmente ambos deben ser creados y archivados de acuerdo al contenido de sus homólogos: *simple.properties* y *%servertype%.sample.properties* respectivamente.

3.2.3 Almacenamiento.

Una ubicación en el disco debe proveerse para el almacenamiento, para poder archivar los contenidos fuentes con todos los resultados del proceso de *transcoding*. El directorio de almacenamiento *(transcoding.storage.root.dir)* tiene que ser elegido cuidadosamente, pues está directamente relacionado con un camino URL a los archivos a los que se les realizó el proceso de *transcoding* (*transcoding.storage.url.path*), el cual es enviado a los clientes. En este momento, el directorio root de almacenamiento debe ser igual al camino real, dentro del cual el apuntador URL es resuelto por el servidor web Alembik.

3.2.4 Procesadores de multimedia.

Hay dos opciones para el procesamiento de imágenes: *ImageMagickMediaProcessor* (http://www.imagemagick.org/) y *GAIAMediaProcessor (http://gaia-git.sourceforge.net/roadmap.htm)*, mientras que hay solamente uno para audio y videos: *FFMpegMediaProcessor (http://ffmpeg.org/)*. La selección es realizada con los parámetros *transcoding. transcoding.%mediatype%.mediaprocessor*, donde *.%mediatype***%** puede ser imagen, audio y video.

Elegir ImageMagick y/o FFMpeg como herramientas de *transcoding* para el Alembik requiere además de una parametrización. En ese caso la línea de comando *full-path* debe ser especificada por cada herramienta seleccionada y en el caso de FFMpeg, la versión de sus librerías de codecs. Por último pero no menos importante debe ser determinado el directorio donde los ficheros de configuración WURFL (Wireless Universal Resource File) serán puestos.

3.2.5 Construcción del servidor.

Teniendo suministrados todos los parámetros mencionados anteriormente puede ser lanzado un proceso de construcción completo usando *assemble_dependencies*. Si la construcción fue satisfactoria debe haberse producido el archivo alembik.war dentro de la carpeta */alembik-build/assemble*, listo para el despliegue en un servidor de aplicación JEE.

3.3. API transcoding.

3.3.1 API de Java.

El componente *TranscodingManager* es el punto de entrada para ejecutar las solicitudes de transcoding en el ambiente Java. Esta clase ofrece dos diferentes maneras para la construcción de solicitudes: basadas en OMA y simplificadas.

La parte de la API basada en OMA sigue la especificación STI (véase capítulo 2) y usa las clases *TranscodingRequest* y *TranscodingResponse* como los objetos primarios de transferencia de datos (Primary Data Transfer Objects) para la comunicación con el servidor de *transcoding*. Véanse entonces algunos métodos del TranscodingManager:

- **public** TranscodingResponse processSync (TranscodingRequest request)
  throws TranscodingException {}

- **public void** processAsync (TranscodingRequest request)
  throws TranscodingException {}

- **public boolean** isSourceFileReady (TranscodingRequest request, String jobID)

- **public boolean** isTranscodedFileReady (TranscodingRequest request, String jobID) {}

- **public** JobResult getTranscodingInfo (TranscodingRequest request, String jobID) {}

Los primeros dos métodos: *processSync ()* y *processAsync()* son el núcleo de todo el *framework*. Envían una petición de *transcoding* al servidor en los modos sincrónicos y asincrónicos respectivamente. La versión sincrónica retorna una instancia del *TranscodingResponse* (Respuesta de *transcoding*), la cual contiene una serie de instancias *JobResult* (Resultado del trabajo). En el caso de que un cliente no desee

realizar ningún transcoding, solamente cargar algún archivo fuente multimedia almacenado en el servidor, puede apoyarse en el método *TranscodingUtils.setNoTranscoding ()* para realizar esta función.

Una vez que una solicitud es enviada, se encuentran dos métodos que obtienen el estado de cualquiera de los trabajos: *isSourceFileReady ()* y *isTranscodeFileReady ()*. Ambos dejan al cliente chequear la disponibilidad de una fuente y la salida de un fichero respectivamente.

Si se requiere de una información más detallada, el *framework* ofrece el método *getTranscodingInfo (),* por el cual el cliente puede obtener una descripción detallada del estado de un trabajo. El objeto retornado *JobResult* contiene una instancia de salida (*Output*) con la localización (URL) de un fichero resultado.

Para los clientes centrados en el ambiente de los móviles, el componente manager provee una utilidad de métodos para la construcción de solicitudes de *transcoding*. Véanse entonces algunos de estos métodos.

- **public** TranscodingRequest prepareUARequest (String sourceUrl,
  String userAgent, Class<? extends Media> mediaType,
  Long maxSize) {}

- **public** TranscodingRequest prepareUARequest (String sourceUrl,
  String userAgent, Class<? extends Media> mediaType,
  Integer width, Integer height) {}

- **public** TranscodingRequest prepareUARequestWithScaling (String sourceUrl,
  String userAgent, Class<? extends Media> mediaType,
  Integer relativeWidth, Integer relativeHeight) {}

- **public** TranscodingRequest prepareUARequestStreamingVideo (String sourceUrl,
  String userAgent, Long maxSize) {}

El primer método construye una solicitud de *transcoding* para una fuente URL del archivo multimedia, un identificador *user-agent* correspondiente al teléfono móvil, el tipo de contenido actual (imagen, audio,

video, o texto) y el tamaño límite deseado (puede ser null). El segundo aplica las dimensiones deseadas del fichero de salida (aplicado solamente a imágenes y videos). Alternativamente, puede ser usada la tercera opción, la cual especifica las dimensiones requeridas, como porcentajes de la pantalla del dispositivo. Con el último método, la transformación de *streaming* puede ser solicitada (videos solamente). En todos los casos el servidor toma la responsabilidad de traducir la cadena *user-agent* dentro de una serie de parámetros de *transcoding*.

3.4 Núcleo

El núcleo del servidor Alembik implementa la mayoría de las funcionalidades del lado del servidor y realiza el procesamiento de las peticiones clientes. Los pasos que se siguen para cada solicitud de *transcoding* son los siguientes:

- Carga de un fichero fuente.
- Resolución del perfil de *transcoding*.
- Selección del procesador de archivos media.
- Realización del proceso de *transcoding*.

La cadena de ejecución de todos estos pasos es controlada por la implementación de la interfaz *org.alembik.TranscodingService – TranscodingServiceImpl*, que recibe cada petición del lado del servidor. Primero el componente intenta obtener el archivo fuente media en un ambiente local usando el método *Storage.loadSourceFile ().* Una vez que el fichero está listo el control es pasado al método *ProfileResolver.evaluateProfile(),* que analiza las solicitudes y prepara todos los parámetros de transcoding (ej. Resolviendo cada *user-agent*). Teniendo todos los datos en su lugar el controlador selecciona un procesador apropiado para los contenidos media (*MediaProccessorFactory.getProccessor())* e inicia la operación de *transcoding* con el método *MediaProcessor.process ().*

Algunos procesadores de contenidos media como *ImageMagick* y *FFmpeg* están basados en librerías nativas y ejecutan las peticiones de *transcoding* externamente. Otros, incluyendo al *GAIAMediaProcessor*, hacen su trabajo usando librerías de java. Por ejemplo, el procesador *GAIA*, delega la mayoría de las

operaciones de *transcoding* a una cadena de componentes *org.alembik.processing.gaia.TransformationProcessor*, cada uno responsable de una sola solicitud de transformación por un cliente dentro del trabajo de *transcoding*.

3.5 Resolución del perfil.

Como se mencionó en la sección anterior, es el componente *org.alembik.resolver.ProfileResolver*, el que realiza el pre-procesamiento de los datos de los trabajos de *transcoding*. El pre-procesamiento de cada petición implica el análisis de los parámetros de *transcoding* enviados por el cliente. El cliente debe crear y pasar su propia instancia de *TranscodingParams*, estableciendo parámetros personalizados, por ejemplo, las dimensiones deseadas de una imagen, valores *bit rate* de audio/video o un número máximo de colores. Por otra parte, el cliente puede asignar un identificador de perfil (profileID) desde una lista de perfiles "pre-definidos", que están almacenados en el archivo profile-defs.xml. Otra opción es un identificador *user-agent* del dispositivo que realiza la solicitud, el cual puede ser pasado también a través de un profileID. En este caso el Alembik usa el componente apuntador *UserAgentResolver* para generar una serie de parámetros de *transcoding* que correspondan con las capacidades del dispositivo dado.

Un perfil definitivo a ser usado por cada trabajo en el proceso de *transcoding* está determinado en la siguiente manera. Si un trabajo de transcoding examinado tiene una instancia *TranscodingParams* asignada, Alembik trata solamente de averiguar si es el *user-agent* pasado como profileId. Si este es el caso, se combinan los parámetros obtenidos en la resolución del *user-agent* con los encontrados en la instancia *TranscodingParams.*

De otra manera, si no pueden ser determinados los parámetros de *transcoding*, pero todavía hay un valor del parámetro *profileId*, el Alembik trata de juntarlo con uno de los identificadores de perfiles predefinidos almacenados en el repositorio "profile-defs.xml". Cada perfil predefinido es distinguido por su único id y puede contener una serie de parámetros de *transcoding* relacionados a un simple tipo de media solamente. El proceso de unión se lleva a cabo por el componente *org.alembik.TranscodingProfileRepository*.

El ejemplo de cómo realizar la definición de un perfil predeterminado puede verse a continuación:

```
 <mts: transcodingProfile>
    <mts: profileID>TM_img_11</mts: profileID>
    <mts: transcodingParams>
       <image>
          <contentType>image/gif</contentType>
          <sizeLimit>3072</sizeLimit>
          <width>45</width>
          <height>33</height>
       </image>
    </mts: transcodingParams>
</mts:transcodingProfile>
```

3.5.1 Resolución del *User-Agent*

Alembik delega la resolución del *user-agent* a la implementación *org.alembik.resolver.UserAgentResolver* obtenida por la clase *UserAgentResolverFactory*. La implementación por defecto (*org.alembik.wurfl.WurflUAResolver*) hace uso de la librería WURFL y por cada tipo de contenido delega la evaluación de los parámetros a otras clases del paquete.

La relación entre las propiedades OMA generadas por Alembik y las capacidades obtenidas de WURFL están representadas y se pueden encontrar en el anexo 2.

3.6 Selección del procesador

Mientras que la clase *org.alembik.TranscodingServiceImpl* es responsable de enrutar las peticiones a las instancias apropiadas del *MediaProcessor*, el trabajo real es iniciado por *org.alembik.processing.MediaProcessorFactory*. El proceso es realizado de la siguiente forma: primeramente la clase *TranscodingServiceImpl* recibe una instancia de la clase *MediaProcessor* desde

MediaProcessorFactory y entonces invoca al método *process (TranscodingJob, JobResult).* La arquitectura está abierta para la introducción de las otras implementaciones *MediaProcessor*, es bastante fácil sustituir un procesador existente o la política de selección de procesadores sin cambiar una sola línea de código.

3.6.1 Procesamiento de archivos multimedia

El procesador de media es un componente estándar, cuyo rol principal es realizar la operación de transcoding. Cada uno tiene que implementar la interfaz *org.alembik.processing.MediaProcessor*.
Hasta el momento existen cinco procesadores de media implementados por el proyecto Alembik:

- *org.alembik.processing.DefaultMediaProcessor.*
- *org.alembik.processing.ImageMagickMediaProcessor.*
- *org.alembik.processing.gaia.GAIAMediaProcessor.*
- *org.alembik.processing.FFMpegMediaProcessor.*
- *org.alembik.processing.WebRenderingProcessor.*

DefaultMediaprocessor es un procesador controlador. Recibe todas las peticiones desde *TrasnscodingServiceImpl* y las despacha al pertinente *MediaProcessor*.

El procesador de media actual es seleccionado de acuerdo al tipo de contenido que se desee como resultado. *DefaultMediaProcessor* asume solamente una implementación para cada tipo de media: imagen, sonido, video, texto; el nombre actual de la clase es asignado por el módulo *Configurator.*

Fuera de los procesadores actuales hay dos capaces de realizar el procesamiento de imágenes: *ImageMagickMediaProcessor* y *GAIAMediaprocessor***.** Están basados respectivamente en el proyecto C++ ImageMagick y en el proyecto GAIA-GIT de Java. El procesador *FFMpegMediaProcessor* es usado para el transcoding de audio y video y está basado en las librerías *C/assembler FFmpeg*. El último,

WebRenderingProcessor, es también un módulo puro de Java, cuyo único propósito es el *transcoding* de textos, principalmente de contenido web HTML.

Los procesadores basados en librerías externas, llamados ImageMagick y FFmpeg, usan el patrón de ejecución de comandos *(java.lang.Runtime.getRuntime ().exec ("…"))* para el *transcoding* de ficheros. Ambos procesadores son de código abierto y son liberados para todos los sistemas operativos. Además a pesar de la gran cantidad de *codecs* que soportan, así como del alcance de sus funcionalidades se han introducido componentes adicionales. El procesador *ImageMagick* fue mejorado con la librería *Batik* que soporta el procesamiento de ficheros *SVG* (*org.alembik.processing.SVGUtility*), mientras que el procesador FFmpeg recibió del MP4Box utilidades para el procesamiento de *streaming* (*org.alembik.processing.MP4BoxUtility*)**.**

Como se mencionó anteriormente el procesador GAIA es una implementación para el *transcoding* de imágenes puro de Java. El módulo provee una serie de codificadores y decodificadores de Java para los formatos más populares de imágenes: **JPG**, **GIF**, **PNG**, **BMP** y **WBMP**. Una vez que se tiene el archivo decodificado, el control es pasado a la cadena de implementaciones de *TransformationProcessor*, servido inmediatamente por *org.alembik.processing.gaia.TransformationProcessorFactory*. *Factory* ofrece un simple contrato para enlazar los posibles tipos de transformaciones con los nombres de las implementaciones; el tipo de transformación (nombre) está adjunto en el camino del paquete, el cual consiste en un prefijo de *org.alembik.processing.gaia* y un sufijo del tipo de contenido que él sirve.

3.8 Almacenamiento

La capa de almacenamiento es la responsable de cargar contenido y escribir un fichero resultado para cada trabajo de *transcoding*. Los procesadores de media del Alembik suponen el acceso a los archivos media originales y salvan los resultados del proceso de *transcoding* a través de esta capa. Este enfoque permite una importante mejora en el rendimiento, siempre y cuando haya más solicitudes de transcoding para el mismo archivo de origen (y con la misma serie de parámetros de *transcoding*), particularmente si el archivo media se encuentra en un sitio remoto.

Alembik provee un poderoso sistema de almacenamiento usando el sistema local de ficheros; el directorio actual de almacenamiento está determinado por el módulo *Configurator*. La evaluación de los distintos nombres de ficheros está basada en un *algoritmo hashing* basado en la ubicación original de los ficheros y en los valores de los parámetros de la petición de transcoding. Este enfoque permite al servidor identificar claramente los contenidos almacenados en el Alembik y chequear si el archivo de origen o el archivo de salida perteneciente al *transcoding* no existen.

Cuando se pregunte por un acceso a un archivo origen, la clase *LocalStorage* evalúa su propio nombre de ficheros local y si ahí no hay archivo presente bajo la ubicación de evaluación, delega la carga de un fichero solicitado a una implementación de la interfaz *MediaLoader*.

La arquitectura de almacenamiento introduce la interfaz (*org.alembik.storage.Storage*), que podría ser implementada para proveer un módulo de almacenamiento personalizado. Una instancia de *Storage* siempre es recuperada a través de la clase *org.alembik.storage.Storage Factory*, la cual retorna una implementación pre-configurada Hasta el momento la única implementación es *org.alembik.storage.local.LocalStorage*.

3.8.1 Cargadores de archivos media.

Como fue mencionado anteriormente las implementaciones de la interfaz *MediaLoader* son usadas por *LocalStorage* para obtener una copia del fichero de origen en el sistema de fichero local. Al momento de escritura de la presente solamente hay dos implementaciones disponibles: *FileMediaLoader* y *HttpMediaLoader*, las cuales son usadas para cargar ficheros desde el sistema de fichero local y sobre la red con el protocolo HTTP respectivamente.

Es responsabilidad del cliente configurar el adecuado protocolo de carga para el acceso a cualquier fichero fuente.

A continuación se presentarán algunos métodos de la clase *org.alembik.storage.local.MediaLoader*:

```
Public interface MediaLoader
{
  public Name load (TranscodingJob job, JobResult result, long modeSecure)
    throws IOException, InterruptedException;
  public long lastModified (TranscodingJob job);
  public void evaluateContentType (TranscodingJob job);
}
```

El primer método *load ()* es responsable de buscar y poner un fichero fuente para un trabajo de *transcoding* dado dentro de un directorio de almacenamiento apropiado. El valor *true* de *ModeSecure* le dice a *MediaLoader* que podría haber otros procesos paralelos cargando el mismo archivo. El método *lastModfied ()* retorna la modificación *timestamp* del fichero original remoto, mientras que el método *evaluateContentType ()* provee el tipo *MIME* del fichero remoto.

3.8.3 Nombramiento de los ficheros almacenados.

Durante cada operación de transcoding el almacenamiento necesita computar (calcular) una carpeta adecuada, donde el fichero fuente y sus resultados estarán almacenados. El camino de la carpeta está determinado por la ubicación original del fichero fuente (llevado a cabo por el objeto *TranscodingJob.Source*).

3.9 Conclusiones parciales

En el capítulo que recién concluye, se abordaron aspectos esenciales que ayudan al trabajo con la herramienta Alembik, como fueron: su proceso de construcción, los parámetros principales para que se logre una correcta utilización, los métodos principales dentro del API que permiten acceder a las funcionalidades más importantes del servidor, la forma en que serán almacenados y accedidos los contenidos, entre otros aspectos que van a permitir la implementación del sistema de *transcoding* que será abordado en el próximo capítulo.

Capítulo 4

Propuesta del Sistema

4.1 Introducción

En el presente capítulo se realizará una descripción sobre el problema existente actual, las características que presentará el sistema, así como conceptos a tratar y las relaciones existentes a través de un Modelo de Dominio, esto debido a la poca definición de los procesos de negocio existentes. Se relacionarán los principales requerimientos que tendrá que cumplir el sistema, así como todo lo concerniente a los modelos de análisis, diseño, implementación y las pruebas.

4.2 Objeto de estudio.

4.2.1 Definición del problema

La empresa de servicios móviles de Cuba, CUBACEL, le ha dado la responsabilidad a la Universidad de Ciencias Informáticas y en especial a la facultad 2 de la misma, de confeccionar una plataforma capaz de realizar algunas funciones para teléfonos móviles como la descarga de contenido, a través de un portal WAP. Esta descarga de contenido la puede realizar cualquier tipo de teléfono móvil. Cada móvil posee características diferentes y junto con ellas no todos soportan los mismos tipos de formato de contenidos (imagen, sonido, video, texto), por lo que es necesario contar con una aplicación que gestione la transformación a distintos formatos de contenido en dependencia de las características del celular que pide el servicio de descarga.

Con lo planteado anteriormente surge una problemática y es precisamente: ¿Cómo realizar el proceso de *transcoding* (proceso de transformación de contenidos) dentro de la plataforma de Gestión de Contenidos?

4.2.2 Objeto de automatización.

Se realizará un módulo de *transcoding* que será capaz de transformar el contenido entre diferentes formatos y cambiar sus características de forma tal que se adecue a las necesidades específicas de un dispositivo móvil dado.

4.2.3 Información que se maneja.

La información a manejar está relacionada con la petición que realiza la plataforma de Gestión de Contenido a la plataforma de *transcoding* en el momento que sea necesaria. La misma estará compuesta por:

- Identificador del contenido almacenado.
- *User-Agent* del teléfono cliente (por donde se puede conocer el tipo de móvil y sus características).
- Tipo de contenido (Content/Type) del contenido al que se le realizará el proceso de *transcoding*.
- Tipo de contenido (Content/Type) del contenido que se quiere lograr como resultado del proceso.

4.3 Propuesta del sistema.

4.3.1 Modelo de Dominio.

Tanto las reglas como los procesos de negocio no están bien definidos, por lo que se pasará a la realización de un modelo de dominio, que sea capaz de capturar los tipos de objetos más importantes en el contexto del negocio, así como brindar un framework conceptual de las cosas relacionadas con el contexto del problema.

Conceptos:

Cliente: Usuario del dispositivo móvil que solicita la descarga.

Cache: Sistema de archivos en el cual estarán almacenados los contenidos a los que se les ha realizado el transcoding.

Contenido: Se refiere al contenido con el que se va a trabajar.

Módulo de Contenido: Es el módulo encargado de gestionar el contenido.

Transcoder: Sistema que realizará el proceso de transformación de los contenidos.

Repositorio Móvil: Contenedor de los tipos de contenido que soportan los distintos tipos de móviles de acuerdo con su *user-agent*.

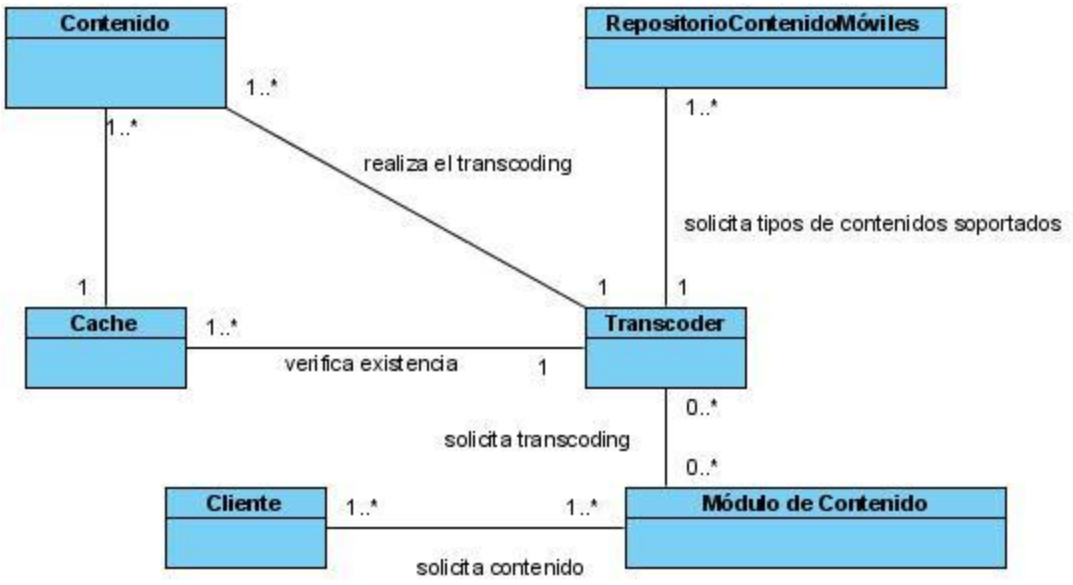


Figura 4.1 Modelo de dominio.

4.4 Especificación de los requisitos de software.

Con la especificación de los requerimientos del software se definirá el ámbito del sistema, se establecerá un acuerdo sobre lo que el sistema debe realizar, proveerá una base para la estimación (se realizará en el próximo capítulo) de recursos y tiempo de desarrollo del sistema.

4.4.1 Requerimientos funcionales

Los requerimientos funcionales darán la medida de las funcionalidades que el sistema debe cumplir una vez construido.

RF1 Transformar el contenido de acuerdo al móvil que está realizando la petición.

RF2 Crear un mecanismo de cache de forma tal que cuando se realice un proceso de *transcoding* a un dispositivo móvil y después se realice otra petición que contenga el mismo tipo de contenido y el mismo *user-agent* no sea necesario volver a realizarle el proceso.

4.4.2 Requerimientos no funcionales.

Tener presente en el sistema los requerimientos no funcionales dará la medida de qué características harán al producto atractivo, usable, rápido o confiable, entre otras cosas necesarias para el agrado del cliente.

4.4.2.1 Software.

- Es imprescindible tener instalado la máquina virtual de java.
- Para transformar videos y audio es necesario tener instalado el software FFMpeg.
- Para transformar imágenes se recomienda la instalación del software ImageMagick.

4.4.2.2 Hardware.

- Es necesario poseer una computadora personal con 512 de memoria RAM, procesador Pentium IV o superior.

4.4.2.3 Restricciones en el diseño y la implementación.

- El lenguaje de programación a ser usado para la implementación será el lenguaje de programación orientado a objetos: JAVA.
- La herramienta Case a utilizar será: Visual Paradigm con soporte para UML 6.0.
- El sistema debe cumplir con estándares internacionales de transcoding de contenidos para móviles como el STI 1.0 de OMA (véase Capítulo 2).

4.4.2.4 Rendimiento.

- El sistema debe ser capaz de realizar las tareas de respuesta en un tiempo relativamente corto.
- El sistema debe ser escalable y comportarse de forma estable ante grandes cantidades de demandas.

4.4.2.5 Soporte.

- Una vez entregado el sistema se le dará mantenimiento.

4.4.2.6 Portabilidad.

- El sistema será multiplataforma.
- El sistema debe ser desarrollado en forma modular.

4.5 Modelo del sistema

4.5.1 Actores del sistema

Actores	Justificación
Módulo de Contenido	Es el encargado de realizar la petición de *transcoding*.

4.5.2. Casos de Uso del sistema.

CU-1	Realizar *transcoding*
Actor	Módulo de Contenido.
Descripción	El Módulo de Contenido solicita al módulo de transcoding que lleve a cabo el proceso de *transcoding* de un contenido, dado un identificador de contenido y el *user-agent* del móvil.
Referencia	*RF1, RF2*

4.5.3 Diagrama de Casos de Uso del sistema.

Figura 4.2 Diagrama de Casos de Uso del Sistema.

4.5.4 Descripción de los casos de uso del sistema.

Caso de uso		
CU-1	Realizar el transcoding	
Propósito	Realizar el proceso de transformación de contenido para que pueda ser accedido por cualquier tipo de dispositivo móvil.	
Actores: Módulo de contenido(inicializa)		
Resumen: El caso de uso se inicia cuando el módulo de contenido solicita que sea realizado un proceso de *transcoding* para un contenido dado, brindándole el identificador de contenido y el *user-agent.*		
Referencias	RF1, RF2	
Acción del actor		**Respuesta del sistema**
1) Módulo de contenido solicita el *transcoding* brindando el user-agent del cliente y el identificador del contenido guardado en el sistema de archivos.		2) Verifica que no exista un contenido con ese mismo user-agent en la cache Si existe ir a 2.1
		3) Busca Contenido
		4) Realiza el proceso de *transcoding* con el alembik.
		5) Una vez obtenido el archivo al que se le realiza el proceso de *transcoding* lo copia para la caché.

	6) Devuelve el contenido al Módulo de Contenido.
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
	2.1) Devolver contenido al módulo de contenido.

4.6 Análisis y Diseño.

Como resultado del flujo de trabajo de requisitos se obtiene una vista externa del sistema, se describe lo que se espera de él a través del Diagrama de casos de uso. A partir de aquí se profundiza en el caso de uso detallándolo de manera que permita reflejar una vista interna del sistema descrita con el lenguaje de desarrollo. En esta vista interna se especifican mejor el caso de uso y se determinan las clases necesarias para llevar a cabo las funcionalidades en él contenidas (7).

Este proceso se desarrolla fundamentalmente dentro de la fase de elaboración y se corresponde principalmente con el flujo de trabajo de análisis y diseño (7).

4.6.1 Definición de análisis.

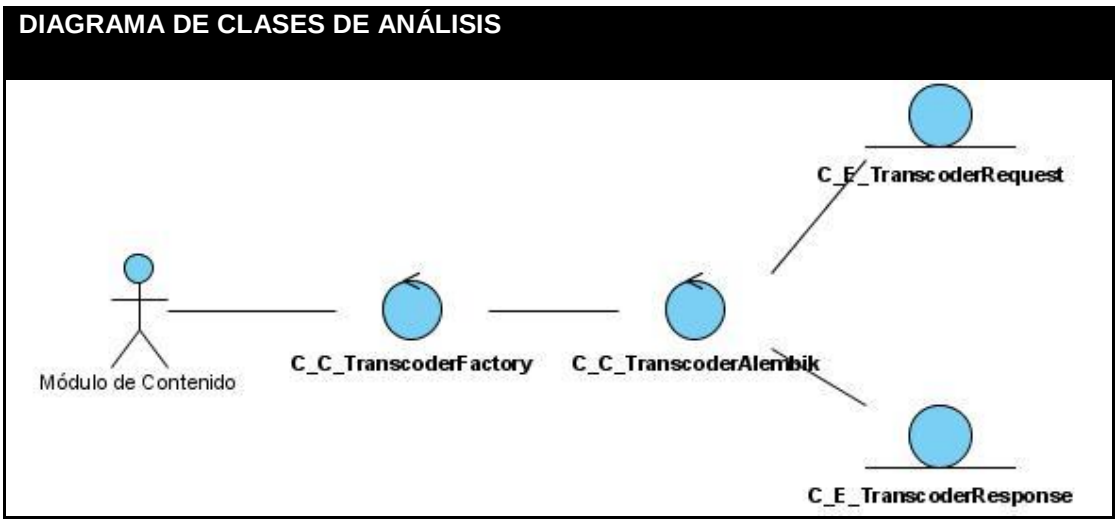
Durante el análisis, son analizados los requisitos que fueron descritos en la captura de requisitos, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema entero, incluyendo su arquitectura (7).

4.6.1.2 Diagrama de clases del análisis.

En el diagrama de clases correspondiente al modelo de análisis se pueden encontrar dos tipos principales de clases:

- Entidad: De este tipo de clases se tienen a: C_E_TranscoderResponse y a C_E_TranscoderRequest, que a pesar de ser clases entidad porque guardan en un momento determinado información no son clases persistente.
- Control: De este tipo de clases se tienen a: C_C_TranscoderFactory y a C_C_TranscoderAlembik, donde la primera es la encargada de crear una instancia de la clase TranscoderAlembik, que a su vez es la encargada de realizar el proceso de *transcoding* en correspondencia con las clases entidad anteriormente mencionadas.

4.6.1.1 Diagrama de Clases del análisis.



4.6.2 Diseño.

El diseño es el centro de atención al final de la fase de elaboración que propone **RUP** y el comienzo de las iteraciones de construcción. Esto contribuye a una arquitectura estable y sólida, y crear un plano del modelo de implementación. Durante la fase de construcción, cuando la arquitectura es estable y los requisitos están bien entendidos, el centro de atención se desplaza a la implementación (8).

En el diseño se modela el sistema y se encuentra la forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Una entrada esencial en el diseño es el resultado del análisis, o sea el modelo de análisis, que proporciona una comprensión

detallada de los requisitos. Además, impone una estructura del sistema que debemos esforzarnos por conservar lo más fielmente posible cuando demos forma al sistema (8).

4.6.2.1 Patrones de Diseño.

Un patrón es una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio. Un sistema bien estructurado está lleno de patrones. Surgieron en el año 1977, inventados por Christopher Alexander (8).

Cada patrón describe un problema que ocurre una y otra vez en nuestro ambiente, y luego describe el núcleo de la solución a ese problema, de tal manera que se pueda reutilizar esa solución cada vez que aparezca en un problema determinado (8).

A continuación se presenta una relación de los principales patrones por los que se regirá la arquitectura y el diseño del módulo de transcoding.

Patrones Creacionales.

- **Factory Method** (Método de fabricación): Centraliza en una clase constructora la creación de objetos de un subtipo de un tipo determinado, ocultando al usuario la casuística para elegir el subtipo que crear (8).

- **Singleton** (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia (8).

Patrones para asignar responsabilidades. GRASP.

- **Patrón Experto:** Su principal uso es el de asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad (8).

- **Patrón Creador:** El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento (8).

- **Patrón Bajo Acoplamiento:** El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Acoplamiento bajo significa que una clase no depende de muchas clases. Acoplamiento alto significa que una clase recurre a muchas otras clases, lo que presenta varios problemas: Los cambios de las clases afines ocasionan cambios locales, difíciles de entender cuando están aisladas, difíciles de reutilizar puesto que dependen de otras clases. El uso de este patrón conlleva a la asignación de una responsabilidad para mantener bajo acoplamiento (8).

- **Patrón Alta Cohesión:** La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Su uso principal es el de asignar una responsabilidad de modo que la cohesión sea alta (8).

Una vez definidos los patrones de diseño se pasará a la confección del diagrama de clases del diseño.

4.6.2.2 Diagrama de clases del diseño.

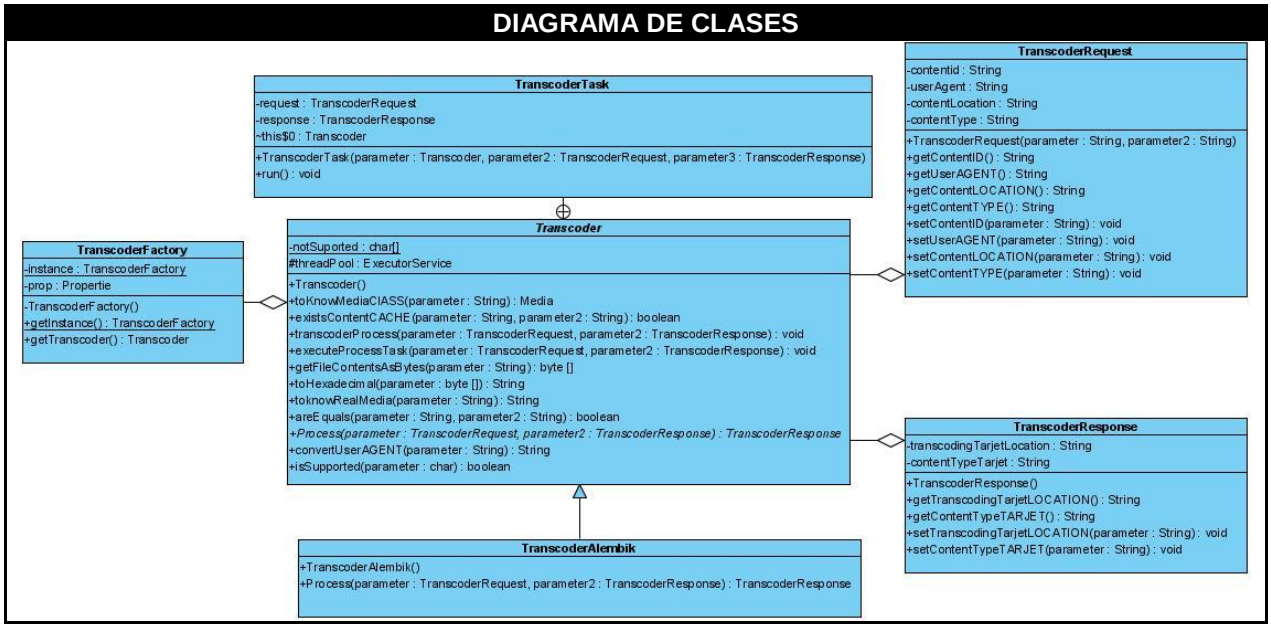
4.6.2.2.1 Clases del diseño.

Una clase de diseño es una construcción similar en la implementación del sistema:

- El lenguaje utilizado para especificar una clase del diseño es lo mismo que el lenguaje de programación (JAVA). Las operaciones, atributos, tipos, visibilidad (public, protected, private), etc se pueden especificar con la sintaxis del lenguaje elegido.
- Las relaciones entre clases de diseño se traducen de manera directa al lenguaje:
    - generalización: herencia
    - asociaciones, agregaciones: atributos

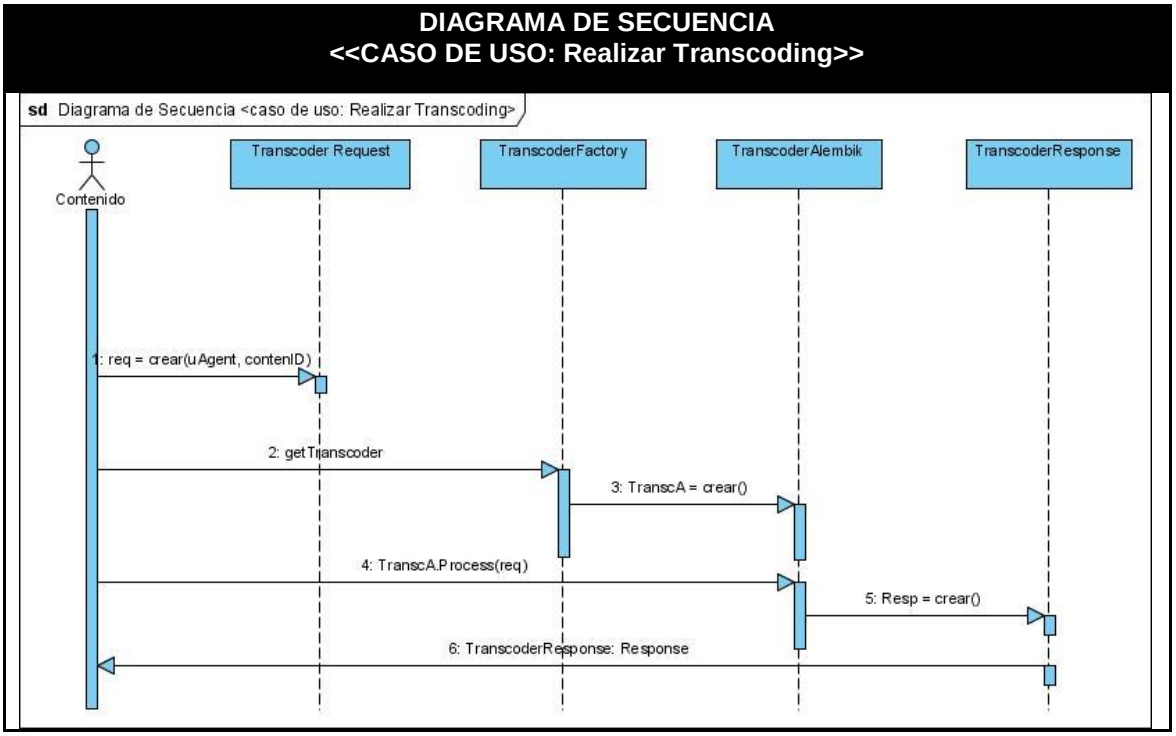
- Los métodos de una clase del diseño tienen correspondencia directa con el correspondiente método en la implementación de las clases.
- Se pueden postergar algunos requisitos a implementación (por ejemplo: manera de nombrar los atributos, operaciones,...).
- Una clase de diseño puede proporcionar interfaces si tiene sentido hacerlo en el lenguaje de programación.

4.6.2.2.2 Diagrama de clases



En el anexo 3 se puede encontrar una descripción detallada de todas las clases del Diseño.

4.6.2.2.3 Diagrama de interacción.



4.7 Implementación.

El flujo de trabajo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue (9).

Los **diagramas de despliegue** y **componentes,** que son artefactos generados en este flujo de trabajo conforman lo que se conoce como un modelo de implementación al describir los componentes a construir y su organización y dependencia entre nodos físicos en los que funcionará la aplicación (9).

4.7.1 Diagramas de despliegue y componentes.

El sistema de *transcoding* será desplegado en cualquier computadora que tenga instalado la máquina virtual de java y que cumpla con los requisitos de hardware, mencionados en la sección **4.4.2.2**.

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación. En la figura **4.3** se muestra el diagrama de componentes correspondiente al sistema de *transcoding*.

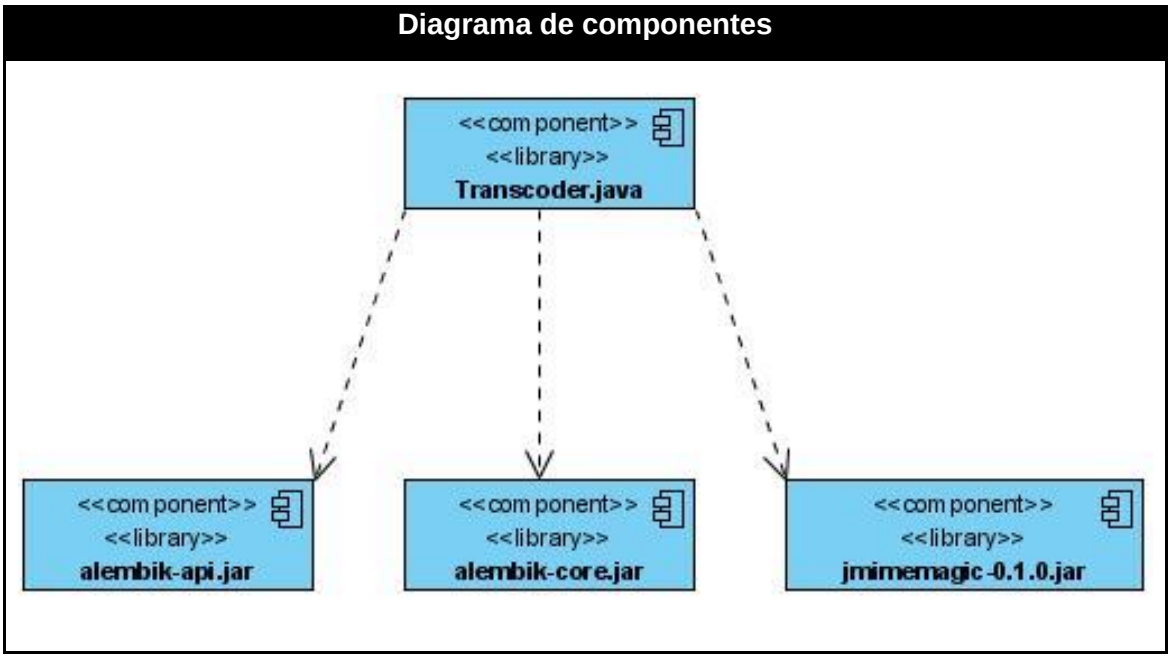


Figura 4.3 – Diagrama de componentes.

4.8 Realizando pruebas.

4.8.1 Prueba de unidad.

La prueba de unidad es la primera fase de las pruebas dinámicas y se realiza sobre el módulo del software de manera independiente. El objetivo es comprobar que el módulo de transcoding, está

correctamente codificado (10).

Las pruebas se realizarán con la ayuda de la herramienta JUnit (véase capítulo 1, sección 1.6.2).

4.8.1.1 Resultados obtenidos de las pruebas aplicadas a los componentes.

4.8.1.1.1 Verificando que se lleve a cabo el proceso de *transcoding.*

El objetivo del siguiente caso de prueba es que se realice la correspondiente transformación al contenido con id = luisma para el movil de user-agent = Mozz

```
public void testTranscodingProces()
{
    TranscoderResponse response = new TranscoderResponse();
    TranscoderRequest request = new TranscoderRequest("luisma", "Mozz");
    Transcoder transcoder = TranscoderFactory.getInstance().getTranscoder();
    transcoder.executeProcessTask(request, response);
}
```

4.8.1.1.2 Verificando el formato del contenido al que se le realiza el transcoding.

El objetivo del caso de prueba es que al realizarse el proceso de *transcoding* al contenido con id = luisma, para el móvil con user-agent = Nokia 20, el formato de la imagen sea “gif”, que es el único formato de imagen que soporta ese tipo de celular.

```
public void testVerifyFormat()
{
    File file = CreateFile();
    String type = "";
    try
    {
        type = Magic.getMagicMatch(file, false).getMimeType();
```

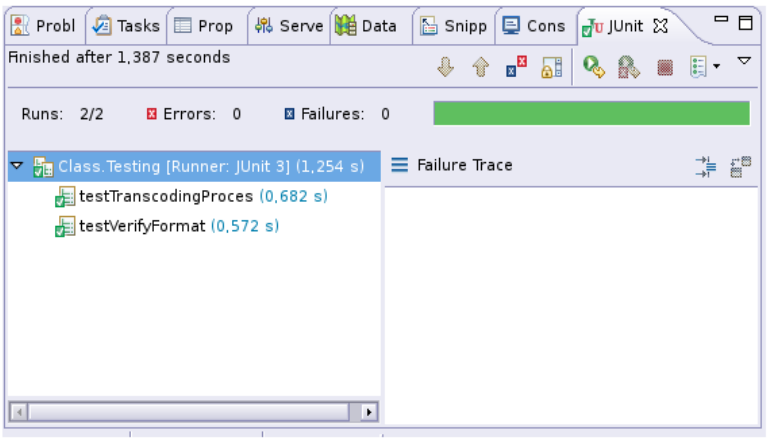
```
        }
        catch(MagicParseException magicPe)
        {
            magicPe.printStackTrace();
        }
        catch(MagicMatchNotFoundException magicmathe)
        {
            magicmathe.printStackTrace();
        }
        catch(MagicException magice)
        {
            magice.printStackTrace();
        }
        assertTrue(type.equals("image/gif"));
    }
```

, donde CreateFile() es el siguiente método

```
public File CreateFile()
    {
        TranscoderResponse response = new TranscoderResponse();
        TranscoderRequest request = new TranscoderRequest("luisma", "Nokia 20");
        Transcoder transcoder = TranscoderFactory.getInstance().getTranscoder();
        response = transcoder.Process(request, response);
        File file = new File(response.getTranscodingTarjetLOCATION());
        return file;
    }
```

4.8.2 Probando veracidad de los resultados.

Después de analizar los casos de pruebas mencionados anteriormente, el resultado mostrado por la herramienta es el siguiente:



, donde se puede observar que todos los casos de prueba fueron totalmente satisfactorios.

4.9 Conclusiones parciales

Al concluir este capítulo se cuenta con un módulo de *transcoding*, que realiza las transformaciones de contenidos. A este sistema se le aplicaron pruebas (véase Capítulo 4, sección 8), dando todas ellas satisfactorias, lo que da la medida de la correcta funcionalidad del módulo.

Capítulo 5

Estudio de factibilidad

5.1 Introducción.

En el presente capítulo, haciendo uso del método de estimación: Análisis de Puntos por Caso de Uso, se realizará la estimación del costo, el esfuerzo y el tiempo necesarios para obtener el software.

5.2 Planificación.

5.2.1 Puntos de Casos de Uso.

La estimación mediante el análisis de Puntos de Caso de Uso, se trata de estimación del tiempo de desarrollo de un proyecto mediante la asignación de "pesos" a un cierto número de factores que lo afectan, para finalmente, contabilizar el tiempo total estimado para el proyecto a partir de esos factores. A continuación, se detallan los pasos a seguir para la aplicación de éste método.

5.2.1.1 Cálculo de Puntos de Casos de Usos sin ajustar.

El primer paso a realizar será el cálculo de los Puntos de Caso de Uso sin ajustar, calculándose de la siguiente manera:

UUCP = UAW + UUCW

Donde:

- **UUCP**: Puntos de Casos de Uso sin ajustar.
- **UAW**: Factor de Peso de los Actores sin ajustar.
- **UUCW**: Factor de Peso de los Casos de Uso sin ajustar.

Para calcular el Factor de Peso de los Actores sin ajustar se debe realizar un análisis de la cantidad de actores presentes en el sistema, así como la complejidad de cada uno de ellos.

Los criterios para el análisis se encuentran en la tabla siguiente:

Tipo de Actor	Descripción	Peso	Cantidad * Peso
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (*API*, *Application Programming Interface*).	1	0*1
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto.	2	1*2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica.	3	0*3
∑(actores*Peso)			2

El Factor de Peso de los Casos de Uso sin ajustar se calcula mediante un análisis de la cantidad de Casos de Uso presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los Casos de Uso se establece teniendo en cuenta la cantidad de transacciones efectuadas en el mismo, donde una transacción se entiende como una secuencia de actividades atómica, es decir, se efectúa la secuencia de actividades completa, o no se efectúa ninguna de las actividades de la secuencia y está representada por uno o más pasos del flujo de eventos principal del Caso de Uso, pudiendo existir más de una transacción dentro del mismo Caso de Uso. Los criterios se muestran en la siguiente tabla:

Tipo de Caso de Uso	Descripción	Peso	Cantidad * Peso
Simple	El Caso de Uso contiene de 1 a 3 transacciones.	5	1*5
Medio	El Caso de Uso contiene de 4 a 7 transacciones.	10	0*10
Complejo	El Caso de Uso contiene más de 8 transacciones.	15	0*15
Σ CU*Peso			5

Una vez calculado el Factor de Peso de los Actores sin ajustar (**UAW**) y Factor de Peso de los Casos de Uso sin ajustar (**UUCW**) se procede al cálculo de los Puntos de Casos de Uso sin ajustar (**UUCP**)

UUCP= UAW + UUCW = 2+5 = **7**

5.2.1.2 Cálculo de Puntos de Casos de Uso ajustados

Una vez que se tienen los Puntos de Casos de Uso sin ajustar, se debe ajustar éste valor mediante la siguiente ecuación:

UCP = UUCP * TCF * EF

Donde:

- **UCP**: Puntos de Casos de Uso ajustados.
- **UUCP**: Puntos de Casos de Uso sin ajustar.
- **TCF**: Factor de complejidad técnica.
- **EF**: Factor de ambiente.

Factor de complejidad técnica (**TCF**): Este coeficiente se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante y 5 un aporte muy importante. En la siguiente tabla se muestra el significado y el peso de cada uno de éstos factores:

Factor	Descripción	Peso	Peso * Valor

T1	Sistema distribuido.	2	2 * 1
T2	Objetivos de performance o tiempo de respuesta.	1	1 * 5
T3	Eficiencia del usuario final.	1	1 * 5
T4	Procesamiento interno complejo.	1	1 * 2
T5	El código debe ser reutilizable.	1	1 * 5
T6	Facilidad de instalación.	0.5	0.5 * 5
T7	Facilidad de uso.	0.5	0.5 * 3
T8	Portabilidad.	2	2 * 5
T9	Facilidad de cambio.	1	1 * 4
T10	Concurrencia.	1	1 * 5
T11	Incluye objetivos especiales de seguridad.	1	1 * 2
T12	Provee acceso directo a terceras partes.	1	1 * 5
T13	Se requieren facilidades especiales de entrenamiento a los usuarios.	1	1 * 1
Total			50

La ecuación para su cálculo es:

TCF = 0.6 + 0.01 * Σ (Peso i * Valor Asignado i)

Entonces, **TCF** = 0.6 + 0.01 * 50 = **1.1**

Factor de ambiente: Contempla las habilidades y el entrenamiento del grupo de desarrollo por su importancia en las estimaciones de tiempo. Al igual que en el factor de complejidad técnica se cuantifican las habilidades y el entrenamiento del grupo de desarrollo con valores de 0 a 5. La ecuación para su cálculo es:

EF = 1.4 - 0.03 * Σ (Peso i * Valor Asignado i).

Factor	Descripción	Peso	Peso * Valor
E1	Familiaridad con el modelo de proyecto utilizado.	1.5	1.5 * 3
E2	Experiencia en la aplicación.	0.5	0.5 * 1
E3	Experiencia en orientación a objetos.	1	1 * 4
E4	Capacidad del analista líder.	0.5	0.5 * 5
E5	Motivación.	1	1 * 5
E6	Estabilidad de los requerimientos.	2	2 * 5
E7	Personal part - time.	-1	-1 * 3
E8	Dificultad del lenguaje de programación.	-1	-1 * 3
Total			20.5

Entonces, **EF** = 1.4 - 0.03 * 20.5 = **0,785**

Finalmente, **UCP (Puntos de Caso de Uso ajustados) =** UUCP * TCF * EF = 7 * 1.1 * 0.785 = **6.0445**

5.2.1.3 Estimación del esfuerzo.

El esfuerzo en horas - hombre viene dado por:

E = UCP * CF

Donde:

- **E**: Esfuerzo estimado en horas-hombre.
- **UCP**: Puntos de Casos de Uso ajustados.
- **CF**: Factor de conversión.

CF = 20 horas-hombre (si Total EF ≤ 2)

CF = 28 horas-hombre (si Total EF = 3 ó Total EF = 4)

CF = abandonar o cambiar proyecto (si Total EF ≥ 5)

Total EF = Cant EF < 3 (entre E1 –E6) + Cant EF > 3 (entre E7, E8)

Como **Total EF** = 2 + 0

Total EF = 2

CF = 20 horas-hombre (porque Total EF ≤ 2)

Luego **E** = 6.0445 * 20 horas-hombre

E = 120.89 horas-hombre

5.2.1.4 Distribución del Esfuerzo entre las diferentes actividades.

Para llevar a cabo una estimación más completa de la duración total del módulo, se agrega el esfuerzo de las demás actividades. Para ello se plantea la distribución del esfuerzo entre las diferentes actividades:

Actividad	Porcentaje	Horas-Hombre

Investigación	40%	241.6
Análisis	10%	60.445
Diseño	10%	60.445
Implementación	20%	120.89
Pruebas	10%	60.445
Otras actividades	10%	60.445
Total	100%	604.45

El Esfuerzo Total sería 604.45 horas-hombre, si estimamos teniendo en cuenta que un mes tiene 176 horas laborables, pues se trabajan 8 horas diarias 22 días al mes, entonces el Esfuerzo Total en mes-hombre sería 3.434 mes-hombre, aproximadamente 4 meses.

5.2.1.5 Calcular el costo de todo el proyecto.

Costo = CHM * ET / CH

Donde:

- **CH**: Cantidad de hombres.
- **CHM**: Costo Hombre – Mes.
- **ET**: Esfuerzo Total.

Si la Cantidad de hombres es 1 y se tiene un Salario Promedio mensual igual a $100.00.

Entonces **CHM** = **CH** * Salario Promedio

CHM = 1 * 100

CHM = 100.00

Luego **Costo** = **CHM * ET / CH**

Costo = 100.00 * 4

Costo = $ 400

5.2.1.6 Calcular el tiempo de desarrollo de todo el proyecto.

A partir de los datos conocidos anteriormente se podría asegurar que una persona puede realizar el trabajo en un tiempo aproximado de 4 meses, por lo que el módulo de transcoding debe ser realizado en un período de 4 meses.

5.3 Conclusiones parciales

Teniendo en cuenta que: para la implementación del sistema se emplearon tecnologías totalmente libres, con las que el pago de licencias no afectará el costo de desarrollo, los desarrolladores del sistema son estudiantes de la facultad dos de la Universidad de las Ciencias Informáticas, por lo que no hay gastos en salario de profesionales y que la aplicación va dirigida a la Plataforma de Gestión de Contenidos de Cubacel, permitiendo una mejora de los servicios de la misma, se llega a la conclusión de que el desarrollo del sistema es completamente factible.

CONCLUSIONES

Con la realización del presente trabajo se arriba a las siguientes conclusiones:

- Debido a la necesidad de crear un sistema de *transcoding* para la Plataforma de Gestión de Contenidos de CUBACEL se realizó una comparación sobre las principales herramientas de *transcoding* en el mundo y que nos trajo como resultado la elección del servidor de *transcoding* Alembik.

- Se realizó un estudio de un estándar internacional de *transcoding*: OMA STI 1.0 (véase Capitulo 2), usado en la implementación de la herramienta anteriormente dicha.

- Para poder trabajar con el Alembik se realizó un estudio de su funcionamiento interno, o sea, los principales métodos y configuraciones que sirvieran para ser usadas en el modulo.

- Para el desarrollo del módulo se definió la estrategia de diseño basada en el uso de patrones de arquitectura y diseño previamente abordados, siendo de vital importancia la herramienta CASE utilizada que jugó un papel decisivo en la automatización de todos los procesos llevados a cabo para obtener las metas propuestas para el diseño del sistema.

- Una vez diseñado e implementado el sistema se le realizaron pruebas de unidad (véase Capitulo 4 sección 8), dando todas ellas satisfactorias, lo que da la medida de que el módulo puede ser desplegado y utilizado por la Plataforma de Gestión de Contenidos de CUBACEL.

RECOMENDACIONES

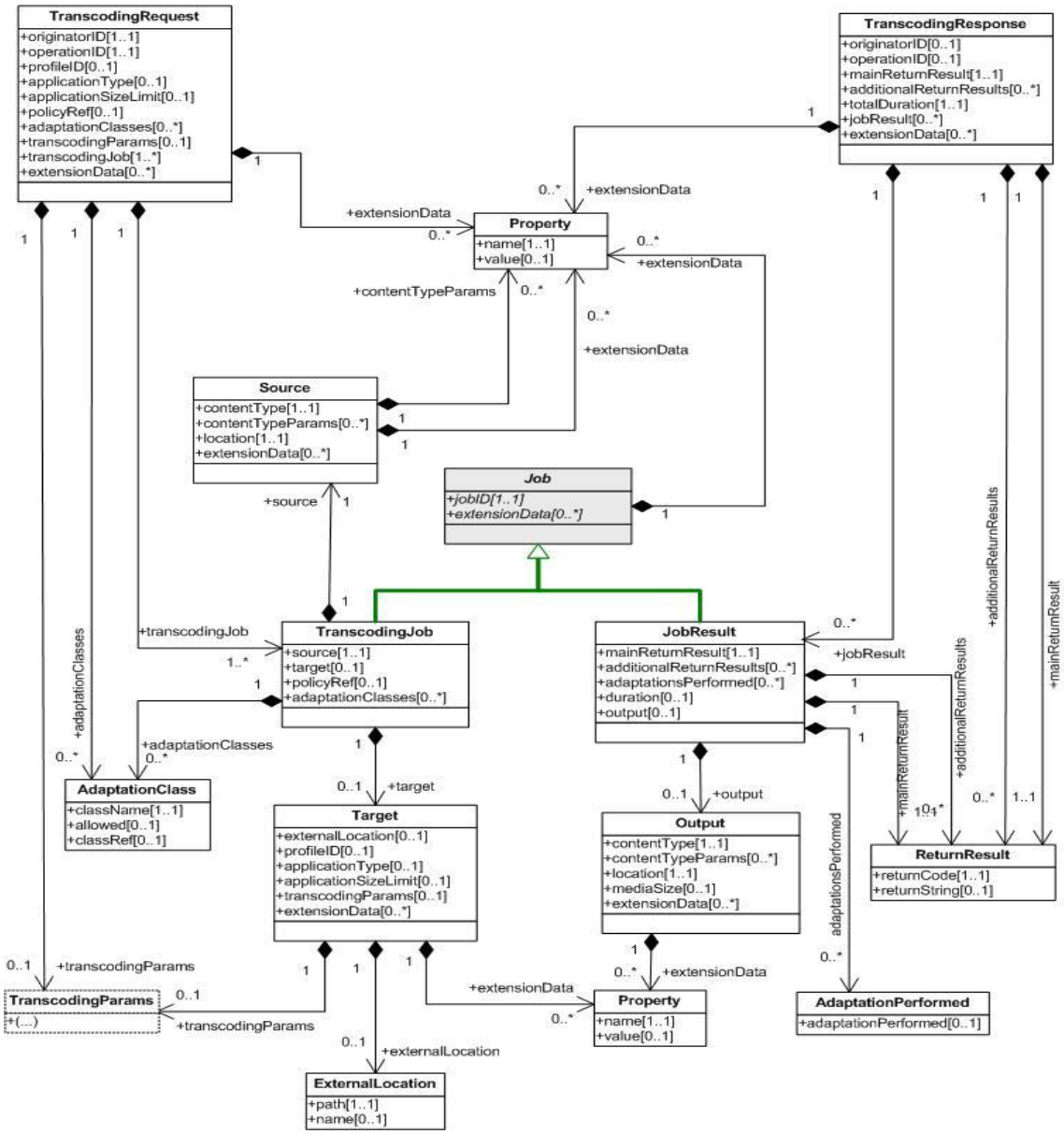
Se recomienda:

- Realizar una investigación más profunda sobre la herramienta Alembik, de forma tal que permita otra forma de obtención de los contenidos que no sea a través de los protocolos http, ftp o a través del sistema de fichero, por ejemplo, que obtenga contenidos que estén previamente guardados en una base de datos, para que permita mejorar su manejo.

- Integrar el modulo de *transcoding* a la Plataforma de Gestión de Contenidos, para corroborar su buen funcionamiento.

Anexos:

Anexo 1: Diagrama UML para los procesos de petición y respuesta del estándar STI 1.0.



Anexo 2: Relación entre las propiedades OMA generadas por Alembik y las capacidades obtenidas desde WURFL.

2.1 Relación entre las propiedades OMA generadas por Alembik y las capacidades obtenidas desde WURFL para imágenes.

OMA Image property name	Alembik property value	wurfl group	wurfl capability
width	wurfl capability value	image_format	max_image_width
height	wurfl capability value	image_format	max_image_height
content-type	image/jpeg	image_format	jpg
	image/png	image_format	png
	image/gif	image_format	gif
	image/tiff	image_format	tiff
	image/bmp	image_format	bmp
	image/gif	image_format	gif_animated
	image/x-epoc-mbm	image_format	epoc_bmp
	image/vnd.wap.wbmp	image_format	wbmp
size-limit	wurfl capability value	object_download	picture_directdownload_size_limit
colorScheme.scheme	PaletteGrey or PaletteColor	image_format	greyscale
colorScheme.depth	wurfl capability value	image_format	colors

2.2 Relación entre las propiedades OMA generadas por Alembik y las capacidades obtenidas desde WURFL para audio.

OMA Audio property name	Alembik property value	wurfl group	wurfl capability
bit-rate	96000	sound_format	mp3
	96000	sound_format	aac

	18250	sound_format	awb
	10200	sound_format	amr
	8000	sound_format	au
	64000	sound_format	wav
	8000	sound_format	mmf, voices (2)
	46000	sound_format	mmf, voices (3)
	12000	sound_format	mmf, voices (5)
	24000	sound_format	mmf, voices (7)
channels	Stereo	sound_format	mp3
	Stereo	sound_format	aac
	Mono	sound_format	awb
	Mono	sound_format	amr
	Mono	sound_format	au
	Stereo	sound_format	wav
	Mono	sound_format	mmf, voices (2)
	Mono	sound_format	mmf, voices (3)
	Stereo	sound_format	mmf, voices (5)
	Stereo	sound_format	mmf, voices (7)
	Mono	sound_format	evrc
	Mono	sound_format	qcelp
	Mono	sound_format	nokia_ringtone
	Mono	sound_format	imelody
	Mono	sound_format	digiplug
	Mono	sound_format	compactmidi
	Mono	sound_format	xmf
	Mono	sound_format	rmf
	Mono	sound_format	sp_midi
	Mono	sound_format	midi_polyphonic
	Mono	sound_format	midi_monophonic
	Mono	sound_format	mld
	Mono	sound_format	smf
content-type	audio/mpeg3	sound_format	mp3
	audio/x-aac	sound_format	aac
	audio/amr-wb	sound_format	awb
	audio/amr	sound_format	amr

	audio/au	sound_format	au
	audio/wav	sound_format	wav
	audio/mmf	sound_format	mmf, voices
	audio/mmf	sound_format	mmf, voices
	audio/mmf	sound_format	mmf, voices
	audio/mmf	sound_format	mmf, voices
	audio/evrc	sound_format	evrc
	audio/qcelp	sound_format	qcelp
	audio/midi	sound_format	nokia_ringtone
	audio/midi	sound_format	imelody
	audio/midi	sound_format	digiplug
	audio/midi	sound_format	compactmidi
	audio/midi	sound_format	xmf
	audio/midi	sound_format	rmf
	audio/midi	sound_format	sp_midi
	audio/midi	sound_format	midi_polyphonic
	audio/midi	sound_format	midi_monophonic
	audio/midi	sound_format	mld
	audio/midi	sound_format	smf
sampling-rate	48000	sound_format	mp3
	96000	sound_format	aac
	16000	sound_format	awb
	8000	sound_format	amr
	8000	sound_format	au
	44100	sound_format	wav
	8000	sound_format	mmf, voices (2)
	48000	sound_format	mmf, voices (3)
	12000	sound_format	mmf, voices (5)
	24000	sound_format	mmf, voices (7)
	96000	sound_format	qcelp
sampling-resolution	16	sound_format	wav
	8	sound_format	au
size-limit	wurfl capability value	object_download	ringtone_directdownload_size_limit

2.3 Relación entre las propiedades OMA generadas por Alembik y las capacidades obtenidas desde WURFL para video.

OMA Video property name	Alembik property value	wurfl group	wurfl capability
content-type	video/vnd.rn-realmedia	object_download streaming	video_real_media_8 streaming_real_media_8
	video/vnd.rn-realmedia	object_download streaming	video_real_media_9 streaming_real_media_9
	video/vnd.rn-realmedia	object_download streaming	video_real_media_10 streaming_real_media_10
	video/3gpp	object_download streaming	video_3gpp streaming_3gpp
	video/3gpp2	object_download streaming	video_3gpp2
	video/mp4	object_download streaming	video_mp4 streaming_mp4
	video/x-ms-wmv	object_download streaming	video_wmv streaming_wmv
	video/quicktime	object_download streaming	video_mov streaming_mov
size-limit	wurfl capability value	object_download streaming	video_directdownload_size_limit streaming_video_size_limit
videoVisual.codec	video/rv30	object_download streaming	video_real_media_8 streaming_real_media_8
	video/rv40	object_download streaming	video_real_media_9 streaming_real_media_9
	video/rv50	object_download streaming	video_real_media_10 streaming_real_media_10
	video/mpg4	object_download streaming	video_3gpp streaming_3gpp video_vcodec_mpeg4 streaming_video_vcodec_mpeg4

	video/h263_0	object_download streaming	video_3gpp streaming_3gpp video_vcodec_h263_0 streaming_video_vcodec_h263_0
	video/mpg4	object_download streaming	video_3gpp2
	video/mpg4	object_download streaming	video_mp4 streaming_mp4
	video/wmv	object_download streaming	video_wmv streaming_wmv
	video/mpg4	object_download streaming	video_mov streaming_mov video_vcodec_mpeg4 streaming_video_vcodec_mpeg4
	video/h263_0	object_download streaming	video_mov streaming_mov video_vcodec_h263_0 streaming_video_vcodec_h263_0
	video/h263_3	object_download streaming	video_mov streaming_mov video_vcodec_h263_3 streaming_video_vcodec_h263_3
	video/h264	object_download streaming	video_mov streaming_mov video_vcodec_h264
videoVisual.width	176	object_download streaming	video_qcif streaming_video_qcif
	128	object_download streaming	video_sqcif streaming_video_sqcif
	wurfl capability value	object_download streaming	video_max_width
	wurfl capability value	object_download streaming	max_image_width
videoVisual.height	144	object_download streaming	video_qcif streaming_video_qcif
	96	object_download streaming	video_sqcif streaming_video_sqcif

	wurfl capability value	object_download streaming	video_max_height
	wurfl capability value	object_download streaming	max_image_height
videoVisual.bitrate	wurfl capability value	object_download streaming	streaming_video_max_bit_rate (only if streaming is requested)
	videoVisual.width * videoVisual.height * 7	object_download streaming	if exist capablities for videoVisual.witdh & .height
	10547	object_download streaming	video_3gpp streaming_3gpp video_vcodec_h263_0 streaming_video_vcodec_h263_0
	10547	object_download streaming	video_mov streaming_mov video_vcodec_h263_0 streaming_video_vcodec_h263_0
	10547	object_download streaming	video_mov streaming_mov video_vcodec_h263_3 streaming_video_vcodec_h263_3
	10547	object_download streaming	video_mov streaming_mov video_vcodec_h264
	10547	object_download streaming	video_3gpp streaming_3gpp video_vcodec_mpeg4 streaming_video_vcodec_mpeg4
	10547	object_download streaming	video_3gpp2
	10547	object_download streaming	video_mp4 streaming_mp4
	10547	object_download streaming	video_mov streaming_mov video_vcodec_mpeg4 streaming_video_vcodec_mpeg4
videoVisual.framerate	wurfl capability value	object_download streaming	video_max_frame_rate streaming_video_max_frame_rate

	10	object_download streaming	video_3gpp streaming_3gpp video_vcodec_h263_0 streaming_video_vcodec_h263_0
	10	object_download streaming	video_mov streaming_mov video_vcodec_h263_0 streaming_video_vcodec_h263_0
	10	object_download streaming	video_mov streaming_mov video_vcodec_h263_3 streaming_video_vcodec_h263_3
	10	object_download streaming	video_mov streaming_mov video_vcodec_h264
	10	object_download streaming	video_3gpp streaming_3gpp video_vcodec_mpeg4 streaming_video_vcodec_mpeg4
	10	object_download streaming	video_3gpp2
	10	object_download streaming	video_mp4 streaming_mp4
	10	object_download streaming	video_mov streaming_mov video_vcodec_mpeg4 streaming_video_vcodec_mpeg4
videoAudio.codec	audio/aac	object_download streaming	video_real_media_8 streaming_real_media_8
	audio/aac	object_download streaming	video_real_media_9 streaming_real_media_9
	audio/aac	object_download streaming	video_real_media_10 streaming_real_media_10
	audio/amr	object_download streaming	video_3gpp streaming_3gpp video_acodec_amr streaming_video_acodec_amr

	audio/aac	object_download streaming	video_3gpp streaming_3gpp video_acodec_aac streaming_video_acodec_aac
	audio/amr	object_download streaming	video_3gpp2 video_acodec_amr
	audio/aac	object_download streaming	video_3gpp2 video_acodec_aac
	audio/aac	object_download streaming	video_mp4 streaming_mp4
	audio/wmav1	object_download streaming	video_wmv streaming_wmv
	audio/amr	object_download streaming	video_mov streaming_mov video_acodec_amr streaming_video_acodec_amr
	audio/aac	object_download streaming	video_mov streaming_mov video_acodec_aac streaming_video_acodec_aac
videoAudio.channels	Stereo	object_download streaming	video_real_media_8 streaming_real_media_8
	Stereo	object_download streaming	video_real_media_9 streaming_real_media_9
	Stereo	object_download streaming	video_real_media_10 streaming_real_media_10
	Mono	object_download streaming	video_3gpp streaming_3gpp video_acodec_amr streaming_video_acodec_amr
	Stereo	object_download streaming	video_3gpp streaming_3gpp video_acodec_aac streaming_video_acodec_aac
	Mono	object_download streaming	video_3gpp2 video_acodec_amr

	Stereo	object_download streaming	video_3gpp2 video_acodec_aac
	Stereo	object_download streaming	video_mp4 streaming_mp4
	Stereo	object_download streaming	video_wmv streaming_wmv
	Mono	object_download streaming	video_mov streaming_mov video_acodec_amr streaming_video_acodec_amr
	Stereo	object_download streaming	video_mov streaming_mov video_acodec_aac streaming_video_acodec_aac
videoAudio.sampling-rate	96000	object_download streaming	video_real_media_8 streaming_real_media_8
	96000	object_download streaming	video_real_media_9 streaming_real_media_9
	96000	object_download streaming	video_real_media_10 streaming_real_media_10
	8000	object_download streaming	video_3gpp streaming_3gpp video_acodec_amr streaming_video_acodec_amr
	96000	object_download streaming	video_3gpp streaming_3gpp video_acodec_aac streaming_video_acodec_aac
	8000	object_download streaming	video_3gpp2 video_acodec_amr
	96000	object_download streaming	video_3gpp2 video_acodec_aac
	96000	object_download streaming	video_mp4 streaming_mp4

	8000	object_download streaming	video_mov streaming_mov video_acodec_amr streaming_video_acodec_amr
	96000	object_download streaming	video_mov streaming_mov video_acodec_aac streaming_video_acodec_aac
videoAudio.bit-rate	wurfl capability value	object_download streaming	streaming_video_max_audio_bit_rate (only if streaming is requested)
	96000	object_download streaming	video_real_media_8 streaming_real_media_8
	96000	object_download streaming	video_real_media_9 streaming_real_media_9
	96000	object_download streaming	video_real_media_10 streaming_real_media_10
	10200	object_download streaming	video_3gpp streaming_3gpp video_acodec_amr streaming_video_acodec_amr
	96000	object_download streaming	video_3gpp streaming_3gpp video_acodec_aac streaming_video_acodec_aac
	10200	object_download streaming	video_3gpp2 video_acodec_amr
	96000	object_download streaming	video_3gpp2 video_acodec_aac
	96000	object_download streaming	video_mp4 streaming_mp4
	10200	object_download streaming	video_mov streaming_mov video_acodec_amr streaming_video_acodec_amr

	96000	object_download streaming	video_mov streaming_mov video_acodec_aac streaming_video_acodec_aac

Anexo 3: Descripción de las clases del Diseño.

3.1 Clase: Transcoder.

Nombre: Transcoder	
Tipo de clase (controladora)	
Atributo	**Tipo**
threadPool	ExecutorService
notSuported	char[]
TranscoderTask	class
Para cada responsabilidad:	
Nombre:	**Transcoder**
Descripción:	Constructor de la clase.
Nombre:	**toknowMediaCLASS**
Descripción	Encargada de devolver, dada la ubicación de un contenido, la clase de Media que es, o sea, si es Image, Audio, o Video.
Nombre:	**existsContentCACHE**
Descripción	Encargada de devolver verdadero si existe un contenido determinado en la cache y falso en caso contrario.
Nombre:	**executeProcessTask**
Descripción	Encargada de ejecutar un hilo de la clase atributo del Transcoder, TranscoderTask.
Nombre:	**Process**
Descripción:	Es una operación abstracta que será redefinida en la clase hija y constituye la operación fundamental del sistema.
Nombre:	**convertUserAGENT**
Descripción:	Encargada de devolver un userAgent que esté en correspondencia con los caracteres soportados para nombrar ficheros.
Nombre:	**transcoderProcess**
Descripción:	Encargada de ejecutar la funcionalidad **Process** y una vez terminada su ejecución notificar que ya terminó.
Nombre:	**getFileContentsAsBytes**
Descripción:	Función encargada de devolver los bytes de un archivo dada su ubicación.
Nombre:	**toHexadecimal**
Descripción:	Convierte los bytes de un archivo a hexadecimal.
Nombre:	**toknowRealMedia**
Descripción:	Encargada de devolver una cadena con el tipo de contenido de un

	archivo dada su ubicación.
Nombre:	**areEquals**
Descripción:	Devuelve verdadero si dos cadenas son iguales, si no devuelve falso.
Nombre:	**isSupported**
Descripción:	Devuelve si un carácter es soportado o no por el sistema de archivos.

3.2 Clase: Transcoder Alembik.

Nombre: TranscoderAlembik	
Tipo de clase (controladora)	
Atributo	**Tipo**
-	-
Para cada responsabilidad:	
Nombre:	**TranscoderAlembik**
Descripción:	Constructor de la clase.
Nombre:	**Process**
Descripción:	Se redefine la operación abstracta Process de la clase padre Transcoder. Operación encargada de realizar el proceso de transcoding llamando a las clases del servidor de transcoding Alembik.

3.3 Clase: TranscoderRequest.

Nombre: TranscoderRequest	
Tipo de clase (entidad)	
Atributo	**Tipo**
contentid	String
userAgent	String
contentLocation	String
contentType	String
Para cada responsabilidad:	
Nombre:	**TranscoderRequest**
Descripción:	Constructor de la clase.
Nombre:	**getContentID**
Descripción:	Operación que devuelve el atributo contentID.
Nombre:	**getUserAGENT**
Descripción:	Operación que devuelve el atributo userAgent.
Nombre:	**getContentLOCATION**
Descripción:	Operación que devuelve el atributo contentLocation.
Nombre:	**getContentTYPE**
Descripción:	Operación que devuelve el atributo contentType.
Nombre:	**setContentID**
Descripción:	Operación que cambia el valor del atributo contentID por uno que se le pase por parámetro.

Nombre:	**setUserAGENT**
Descripción:	Operación que cambia el valor del atributo userAgent por uno que se le pase por parámetro.
Nombre:	**setContentLOCATION**
Descripción:	Operación que cambia el valor del atributo contentLocation por uno que se le pase por parámetro.
Nombre:	**setContentTYPE**
Descripción:	Operación que cambia el valor del atributo contentType por uno que se le pase por parámetro.

3.4 Clase: TranscoderResponse.

Nombre: TranscoderResponse	
Tipo de clase (entidad)	
Atributo	**Tipo**
transcodingTarjetLocation	String
contentTypeTarjet	String
Para cada responsabilidad:	
Nombre:	**TranscoderResponse**
Descripción:	Constructor de la clase.
Nombre:	**getTranscodingTarjetLOCATION**
Descripción:	Operación que devuelve el atributo transcodingTarjetLocation.
Nombre:	**getContentTypeTARJET**
Descripción:	Operación que devuelve el atributo contentTypeTarjet.
Nombre:	**setTranscodingTarjetLOCATION**
Descripción:	Operación que cambia el valor del atributo transcodingTarjetLocation por uno que se le pase por parámetro.
Nombre:	**setContentTypeTARJET**
Descripción:	Operación que cambia el valor del atributo contentTypeTarjet por uno que se le pase por parámetro.

3.5 Clase: TranscoderFactory.

Nombre: TranscoderFactory	
Tipo de clase (controladora)	
Atributo	**Tipo**
instance	TranscoderFactory
prop	Propertie
Para cada responsabilidad:	
Nombre:	**TranscoderFactory**
Descripción:	Constructor de la clase.
Nombre:	**getTranscoder**

Descripción:	Operación que devuelve una instancia del transcoder que esté configurado en la propertie **prop**
Nombre:	**getInstance**
Descripción:	Operación que devuelve una instancia de la clase Factory, si esta no ha sido inicializada.

3.6 Clase: TranscoderTask.

Nombre: TranscoderTask	
Tipo de clase (controladora)	
Atributo	Tipo
request	TranscoderRequest
response	TranscoderResponse
Para cada responsabilidad:	
Nombre:	**TranscoderTask**
Descripción:	Constructor de la clase.
Nombre:	**run**
Descripción:	Operación encargada de ejecutar el método **transcoderProcess** de forma tal que pueda ser ejecutado en un hilo.

Bibliografía Consultada:

1. **Transcoding**, 2008. [Disponible en: http://searchsoa.techtarget.com/sDefinition/0,,sid26_gci515589,00.html].
2. **SVN**, 2009. [Disponible en: http://lihuen.info.unlp.edu.ar/index.php/C%C3%B3mo_usar_SVN].
3. **Visual_Paradigm**, 2009 [Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/].
4. **JEE**, 2009 [Disponible en: http://cea-fidelmazo.blogspot.com/].
5. **Denis.** manzana mecánica, 2008 [Disponible en: http://www.manzanamecanica.org/2008/07/eclipse_ganymede.html].
6. **JUnit**, 2004 [Disponible en : http://www.jtech.ua.es/tutoriales/apuntes/sesion-junit-apuntes.htm].
7. **Ingeniería de Software 1**, Fase de Elaboración. Flujo de trabajo de Análisis y Diseño, 2009.
8. **Ingeniería de Software 2**, Continuación del FT Análisis y Diseño. Modelo de Diseño, 2009.
9. **Ingeniería de Software 2**, Flujo de Implementación, 2009.
10. **Ingeniería de Software 2**, Técnicas de Evaluación de Software, 2006.
11. **Cubacel**, 2009 [Disponible en: http://www.cubacel.cu/].
12. **MediaCoder**, 20009 [Disponible en: http://mediacoder.sourceforge.net/].
13. **VoDKA Batch Transcoder**, 2009 [Disponible en: http://www.lambdastream.com/index.php?page=vodka-batch-transcoder&hl=es_ES]
14. **GXTranscoder**, 2009 [Disponible en: http://gxtranscoder.softonic.com/].
15. **Alembik**, 2009 [Disponible en: http://alembik.sourceforge.net/].
16. **Microsoft Solutions Framework**, 2009 [Disponible en: http://msdn.microsoft.com/es-es/library/ms195024(VS.80).aspx].
17. **Extreme Programming**, 2009 [Disponible en: http://www.extremeprogramming.org/
18. **Scrum**, 2009 [Disponible en: http://www.scrum.com/scrum/rugby/site/index.html].
19. **ChrystalMethod**, 2009 [Disponible en: http://www.thecrystalmethod.com/].

20. **Feature Driven Development**, 2009 [Disponible en: http://www.featuredrivendevelopment.com/]

21. **EasyMock**, 2009 [Disponible en: http://easymock.org/].

22. **JUnit**, 2009 [Disponible en: http://www.junit.org/]

23. **TestNG**, 2009 [Disponible en: http://testng.org/doc/documentation-main.html].

24. **Standard Transcoding Interface Specification**, 2007 [Disponible en: http://www.openmobilealliance.org/Technical/release_program/docs/STI/V1_0-20070515-A/OMA-TS-STI-V1_0-20070515-A.pdf].

25. **Curso de Java**, 2008 [Disponible en: http://www.monografias.com/trabajos/java/java.shtml].

26. **Thread**, 2009 [Disponible en: http://java.sun.com/developer/Books/javaprogramming/threads/chap13.pdf].

27. **ExecutorService (Java 2 Platafform SE 5.0)**, [Disponible en: http://java.sun.com/j2se/1.5.0/docs/api/java/util/concurrent/ExecutorService.html].

28. **Introducción a JAVA**, 2008 [Disponible en: http://tikal.cifn.unam.mx/%7Ejsegura/LCGII/java2.htm].

Glosario de Términos.

A

API: Application Programming Interface, o interfaz de programación de aplicaciones, es el conjunto de funciones y procedimientos (o métodos, si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

B

Bit rate: En telecomunicación e informática, el término bit rate (en español velocidad binaria, cadencia, tasa o flujo de bits) define el número de bits que se transmiten por unidad de tiempo a través de un sistema de transmisión digital o entre dos dispositivos digitales. Así pues, el bit rate es la velocidad de transferencia de datos.

C

Códec: Es una abreviatura de Compresor-Decompresor. Describe una especificación desarrollada en software, hardware o una combinación de ambos, capaz de transformar un archivo con un flujo de datos (*stream*) o una señal.

F

FFMpeg: (http://www.ffmpeg.org/) Completa herramienta para el procesamiento de audio y video. FFmpeg es desarrollado bajo Linux, pero puede ser usado en la mayoría de los sistemas operativos, incluyendo Windows.

FTP: File Transfer Protocol, o Protocolo de Transferencia de Archivos, es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP, basado en la arquitectura cliente-servidor.

G

GSM: Global System for Mobile Communications: Sistema Global para Comunicaciones Móviles. Sistema telefónico digital muy usado en Europa.

H

HTTP: Conjunto de reglas que se usa en Internet para pedir y ofrecer páginas de la red y demás información.

I

ImageMagick: Aplicación que sirve para crear, editar y componer imágenes. Puede leer, convertir y guardar imágenes en una gran variedad de formatos.

J

JAR: Un archivo JAR (por sus siglas en inglés, (Java archive) es un tipo de archivo que permite ejecutar aplicaciones escritas en lenguaje Java.

JSP: Es un acrónimo de Java Server Pages, que en castellano vendría a decir algo como Páginas de Servidor Java. Es, pues, una tecnología orientada a crear páginas web con programación en Java.

JEE: Es una tecnología que apunta simplificar el diseño y la puesta en práctica de los usos de la empresa.

JVM: Una Máquina virtual Java (en inglés *Java Virtual Machine*, JVM) es un programa nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el Java bytecode), el cual es generado por el compilador del lenguaje Java.

L

Logs: Registro oficial de eventos durante un periodo de tiempo en particular. Para los profesionales en seguridad informática es usado para registrar datos o información sobre quién, qué, cuándo, dónde y por qué un evento ocurre para un dispositivo en particular o aplicación.

M

MIME: (*Extensiones Multipropósito de Correo Internet*) es un estándar propuesto en 1991 por Bell Communications para expandir las capacidades limitadas del correo electrónico y en particular para permitir la inserción de documentos (como imágenes, sonido y texto) en un mensaje.

MMS: Multimedia Messaging System, o **Sistema de Mensajería Multimedia** es un estándar de mensajería que le permite a los teléfonos móviles enviar y recibir contenidos multimedia, incorporando sonido, video, fotos o cualquier otro contenido disponible en el futuro.

MP4Box: Conversor MPEG-4. Puede importar vídeo MPEG-4 (DivX/ Xvid/ 3ivx/ ffmpeg (http://www.ffmpeg.org/)) y flujos de audio hacia un contenedor *.mp4.

O

OMA: Open Mobile Alliance**,** es una organización de estándares que desarrolla estándares abiertos para la industria de telefonía móvil.

Open source: El Open Source o Código Abierto es una revolucionaria forma de desarrollar y distribuir el software, el código abierto permite que varios programadores puedan leer, modificar y redistribuir el código fuente de un programa.

POST: Es un método de peticiones HTTP.

R

RMI: Java Remote Method Invocation, es un mecanismo ofrecido por Java para invocar un método de manera remota. Forma parte del entorno estándar de ejecución de Java y provee de un mecanismo simple para la comunicación de servidores en aplicaciones distribuidas basadas exclusivamente en Java.

S

SIM: Una tarjeta SIM (acrónimo de *Subscriber Identify Module*, 'Módulo de Identificación del Suscriptor' **MIS**) es una tarjeta inteligente desmontable usada en teléfonos móviles que almacena de forma segura la

clave de servicio del suscriptor usada para identificarse ante la red, de forma que sea posible cambiar la línea de un terminal a otro simplemente cambiando la tarjeta.

SOAP: Simple Object Access Protocol, es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

STI: Standard Transcoding Interface, es un estándar de *transcoding*, perteneciente a OMA.

Streaming: Término que se refiere a ver u oír un archivo directamente en una página web sin necesidad de descargarlo antes al ordenador.

T

TDMA: Time Division Multiple Access: Acceso Múltiple de División de Tiempo. Norma de transmisión de datos a través de teléfonos inalámbricos.

U

UML: Unified Modeling Language, o Lenguaje Unificado de Modelado, es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad.

URI: Uniform Resource Identifier, es una cadena corta de caracteres que identifica inequívocamente un recurso (servicio, página, documento, dirección de correo electrónico, enciclopedia, etc.).

URL: Uniform Resource Locator, es decir, localizador uniforme de recurso y se refiere a la dirección única que identifica a una página web en Internet.

W

WAP: Son las siglas de Wireless Aplication Protocol (Protocolo de Aplicaciones Inalámbricas), un estándar seguro que permite que los usuarios accedan a información de forma instantánea a través de dispositivos inalámbricos como PDAs, teléfonos móviles, buscas, walkie-talkies y teléfonos inteligentes (smartphones).

WAR: Es un tipo de archivo que permite empaquetar una aplicación web para que sea desplegada en un contenedor web JEE, ó en un servidor de aplicaciones JEE.

WURFL: Wireless Universal Resource File, es un fichero de configuración XML que contiene información sobre las capacidades y características de cualquier dispositivo móvil.

X

XML: Extensible Markup Language («lenguaje de marcas), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).