Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros de Telecomunicación

ARQUITECTURA E IMPLEMENTACIÓN DE PLATAFORMAS DE COLABORACIÓN Y RECOMENDACIÓN BASADAS EN CONTEXTO DE USUARIO

TRABAJO FIN DE MÁSTER

Daniel Gallego Vico

2010

Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros de Telecomunicación

Máster Universitario en
Ingeniería de Redes y Servicios Telemáticos

TRABAJO FIN DE MÁSTER

ARQUITECTURA E IMPLEMENTACIÓN DE PLATAFORMAS DE COLABORACIÓN Y RECOMENDACIÓN BASADAS EN CONTEXTO DE USUARIO

Autor
Daniel Gallego Vico

Director
Joaquín Salvachúa Rodríguez

Departamento de Ingeniería de Sistemas Telemáticos

2010

Resumen

Hoy en día, Internet se ha convertido en un ente social con vida propia donde diferentes tipos de redes han alcanzado una importancia sumamente destacable en lo que a proporcionar métodos de colaboración entre personas de todo el mundo se refiere. Para llevar esta colaboración a cabo, están disponible innumerables servicios que pueden ser integrados o combinados de cara a conseguir satisfacer completamente la demanda de los usuarios. Por lo tanto, y debido a este enorme crecimiento, estas redes sociales almacenan actualmente una cantidad increíble de datos sociales que pueden ser usados para generar un contexto colaborativo minucioso de todos esos usuarios.

Este documento comienza explorando un nuevo paradigma de construcción de sistemas CSCW para entornos corporativos siguiendo estas nuevas ideas puestas en escena por la Web Social para colaborar con el objetivo de generar dicho contexto colaborativo. Se presenta una descripción de los componentes implementados bajo estas ideas para colaborar en espacios de trabajo, destacando una nueva forma de ofrecer servicios de videoconferencia con filosofía cloud computing, de manera que aplicaciones de terceros puedan solicitar salas virtuales donde los usuarios pueden relacionarse mediante audio, vídeo, escritorio compartido o mensajería instantánea. Adicionalmente, se detallan los métodos que se han seguido para generar el contexto colaborativo mencionado anteriormente a partir de datos sociales complejos. Todo esto estará acompañado de figuras y resultados que nos permitirán afirmar al final del desarrollo que la arquitectura implementada está preparada para soportar generación de contexto colaborativo mediante la unión de datos del mundo empresarial y el mundo social, utilizando para ello todo tipo de datos alojados en la nube.

Por otro lado, y en relación con la reciente utilización de los sistemas de recomendación en Internet de acuerdo a la necesidad de ordenar la ingente cantidad de datos que existe ahora en la nube, en este documento se presentan los resultados de la investigación llevada a cabo en el área de los algoritmos de clustering y recomendación. De esta forma, y como resultado de este estudio, se describirá la arquitectura de una aplicación de recomendación que ha sido diseñada y desarrollada con motivo de este trabajo, cuyo principal objetivo es la recomendación de restaurantes utilizando como datos de entrada el contexto de los usuarios del sistema. Finalmente, se detalla el estudio que se ha llevado a cabo sobre métodos de autenticación SSO basados en redes sociales como Facebook y Twitter que se pretende integrar en la arquitectura general del sistema de recomendación construido.

Abstract

Nowadays, Internet is a place where social networks have reached an important impact in collaboration among people over the world in different ways using services that can be integrated or mashed up in order to fulfill user demands. Therefore, these social networks store an incredible set of social data that can be used to provide awareness about their users.

This document starts reviewing a new paradigm of building CSCW tools for business world following these new ideas provided by the social web to collaborate and generate awareness. An implementation of these concepts is described, including the components provided to collaborate in workspaces, such as a way to offer videoconference as a web cloud service over an interface which can be used by third parties to enrich their applications using virtual rooms where users can collaborate with audio, video, shared applications, IM, etc. Also, this document describes the way is generated awareness from these complex social data structures. We also present figures and validation results in the text to stress that this architecture has been defined to support awareness generation via joining current and future social data from business and social networks worlds, based on the idea of using social data stored in the cloud.

On the other hand, and related to the recent use of recommender systems in the Internet due to the existence of all the data stored in the cloud mentioned before, this document presents a research about clustering algorithms related to these recommender systems. In addition, an application for recommending restaurants based on users' awareness is described, including its architecture and implementation. Finally, a study of managing SSO authentication via social networks likes Facebook and Twitter is presented to complete the general architecture of this recommender system.

Índice general

Índice de figuras

Siglas

AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
APP	Atom Publishing Protocol
CAS	Central Authentication Service
COC	Convention Over Configuration
CRUD	Create Read Update Delete
CSCW	Computer Supported Cooperative Work
CSS	Cascading Style Sheet
CVS	Concurrent Versions System
DRY	Don't Repeat Yourself
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
J2EE	Java 2 Platform Enterprise Edition
JIT	Just In Time
JRE	Java Runtime Environment
JSON	JavaScript Object Notation
JSP	Java Server Page
LDAP	Lightweight Directory Access Protocol
MVC	Model View Controller (Modelo Vista Controlador)
MXML	Multimedia eXtensible Markup Language
PFC	Proyecto Fin de Carrera
REST	Representational State Transfer

RFC	Request For Comments
RIA	Rich Internet Applications
RoR	Ruby on Rails
RSS	Really Simple Syndication
SSO	Single Sign On
ST	Service Ticket
SWF	Shockwave Flash
TFM	Trabajo Fin de Máster
TGT	Ticket Granting Ticket
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VNC	Virtual Network Computing
XML	eXtensible Markup Language

1 Introducción

El presente documento tiene como principal objetivo ser una memoria o Trabajo Fin de Máster que abarque las labores de diseño, desarrollo e implementación de las diferentes líneas de investigación y estudio en las que el alumno ha trabajado durante la realización del Máster Universitario en Ingeniería de Redes y Servicios Telemáticos, impartido por el Departamento de Ingeniería de Sistemas Telemáticos de la ETS de Ingenieros de Telecomunicación de la Universidad Politécnica de Madrid.

Puesto que este máster tiene un carácter eminentemente investigador y adicionalmente, el alumno ha enfocado desde un primer momento su participación en él como un camino a recorrer de cara a realizar su Doctorado, la mayoría de los trabajos realizados en él se han adecuado en la manera de lo posible a intentar coincidir con las líneas de interés investigador en las que el alumno trabaja.

De esta forma, en la siguiente sección repasaremos brevemente el contexto en el que se sitúan los trabajos que se han realizado, comentando brevemente que temáticas son las que principalmente interesan al alumno.

A continuación, describiremos los principales objetivos que se tuvieron en mente a la hora de llevar a cabo estos trabajos, tanto desde el punto de vista de las plataformas construidas, como desde el punto de vista de las vías de investigación abiertas en las que se ha profundizado con resultados en forma de artículos publicados en congresos internacionales.

Finalmente, indicaremos brevemente la estructura que sigue el documento que comienza con estas líneas, de cara a comprender de primera mano cual es el orden que se ha seguido en su creación.

1.1 Contexto

Hoy en día, Internet se ha convertido en un eje central sobre el que millones de usuarios en todo el mundo realizan su actividad diaria tanto en facetas de trabajo, como en aquellas más relacionadas con el ocio o las relaciones sociales. En este último sentido, mencionar aplicaciones web como Facebook, Twitter, Flickr o Blogger equivale a hacer referencia a gran parte de la masa social que utiliza Internet, ya que muchas de ellas se han convertido en un medio para interconectar personas situadas en cualquier parte del globo, siendo por tanto responsables en gran medida del impulso socializador que la red de redes ha alcanzado en los últimos años. Por tanto, ahora mismo es una realidad afirmar que prácticamente todo usuario de Internet participa como usuario en alguna de ellas de un modo y otro.

Por otro lado, el mundo empresarial que históricamente siempre ha sido más reacio a incorporar este tipo de tecnologías disruptivas, lleva haciendo uso intensivo de ellas en los últimos años, incorporando poco a poco, aunque cada vez con mayor peso, capacidades de colaboración en las herramientas o plataformas de gestión de los empleados de las mismas, llegando a reproducir en algunos casos estructuras que se usan en las redes sociales comentadas anteriormente. Es más, con el surgimiento del movimiento del *Cloud Computing*, muchas empresas han comenzado a desplegar sus infraestructuras, plataformas y servicios en la nube para, por un lado abaratar costes, y por otro ofrecer servicios más completos al ser más sencilla su gestión, pudiendo abstraerse de los niveles de configuración, mantenimiento y balanceo de carga que siempre son problemas recurrentes en dichas arquitecturas.

Adicionalmente, y muy relacionado con los conceptos comentados, Internet se ha convertido como consecuencia en el mayor depósito de información que podemos encontrar en el mundo, ya que existen nubes de datos públicas y privadas que constantemente están aumentando de tamaño hasta niveles difícilmente gestionables. Por ello y de manera algo más reciente, se ha comenzado a aplicar técnicas de *clustering* y recomendación sobre estos datos existentes en la Web, consiguiendo ofrecer de una forma ordenada y personalizada esta ingente cantidad de datos a los usuarios de las distintas plataformas que hacen uso de este tipo de técnicas, consiguiendo por tanto hacer uso de manera unificada de datos que individualmente no ofrecen tanta información.

Por último, y acompañando a todos estos avances en el uso y la gestión de los recursos web, también se puede comprobar de primera mano la importancia que ha cobrado la experiencia de usuario en la creación de aplicaciones web tanto para entornos fijos, como para móviles. La evolución del paradigma de la Web 2.0 hacia las aplicaciones RIA (*Rich Internet Application*) [1] es un hecho constatado que ha provocado que actualmente se diseñen aplicaciones en muchos casos teniendo como principal objetivo ofrecer un interfaz de usuario llamativo con alta usabilidad que enmascare la complejidad tecnológica subyacente. O lo que es lo mismo, hoy en día se apuesta más por aplicaciones con alta experiencia de usuario que ofrezcan funcionalidades claramente reconocibles por el usuario, antes que por "vender tecnología" por si sola.

1.2 Objetivos

El objetivo principal de este Trabajo Fin de Máster (TFM) es reflejar las diferentes líneas de investigación y proyectos en los que el alumno ha trabajado en el último año durante la realización del presente máster, describiendo para ello de manera detallada las arquitecturas e implementaciones que se han desarrollado para probar las

investigaciones realizadas, estando algunas de ellas respaldadas por artículos en congresos internacionales y otras por artículos de investigación elaborados en las diferentes asignaturas del máster.

Así, y para que resulte clara la evolución de la labor investigadora que el alumno ha llevado a cabo durante todo este tiempo, se comenzará describiendo brevemente el Proyecto Fin de Carrera que el alumno desarrolló y que sirvió como plataforma para probar la integración de un servicio de videoconferencia desarrollado siguiendo arquitecturas *cloud computing*. También, relacionado con este proyecto, se presentarán los resultados de la creación de un sistema de generación de *awareness* o contexto colaborativo a partir de los datos sociales sobre los que se sustenta la mencionada plataforma colaborativa, y que se integró en la misma como prueba de concepto de las investigaciones realizadas.

A continuación, nos adentraremos en la segunda línea de estudio llevada a cabo por el alumno, relacionada con el campo de los sistemas de recomendación basados en perfiles de usuarios y contexto de los mismos, y que ha dado como resultado un sistemas completo con arquitectura Cliente-Servidor compuesto por un servidor de *clustering* de datos y gestión de recomendaciones, y un cliente web diseñado con orientación RIA para acceder a los datos.

Por tanto, se puede resumir diciendo que los objetivos que se propusieron para este TFM fueron los de crear sendos sistemas de colaboración y recomendación basados en contexto de usuario y sus relaciones sociales, para dar servicio a entornos corporativos o a aplicaciones de carácter más abierto.

1.3 Estructura del documento

El presente TFM que lleva como título "Arquitectura e Emplementación de Plataformas de Colaboración y Recomendación basadas en Contexto de Usuario" se divide en seis partes bien diferenciadas:

- En primer lugar la **Introducción** que ocupa estas líneas.
- Tras ésta, una sección dedicada al **Estado del Arte** que comprende por un lado un análisis de las diferentes aplicaciones colaborativas y sistemas de recomendación existentes en el mercado en el momento actual y que poseen características similares a las de este proyecto. Y por otro lado, se incluye un estudio de las tecnologías que se han usado de manera directa, o que están relacionadas con alguno de los componentes incluidos en el proyecto.
- En tercer lugar encontramos la sección denominada **Líneas de investigación y trabajo realizado**, en la que se describen dos plataformas desarrolladas por el autor. La primera de ellas orientada al trabajo colaborativo y la segunda a

proporcionar un sistema de recomendación, estando ambas orientadas a utilizar el contexto de sus usuarios para proporcionar un servicio mejorado.

- A continuación, encontraremos las **Conclusiones** que el autor ha extraído tanto a nivel tecnológico, como de diseño, resultados y experiencia personal adquirida durante la elaboración de los diferentes trabajos.
- En quinto lugar, tendremos una breve descripción de los **Trabajos futuros** que se pueden derivar de lo descrito en este documento, mostrando algunas de las posibles líneas de investigación más interesantes en las que el autor ha comenzado a trabajar actualmente.
- Para cerrar el trabajo, encontraremos una sección de **Bibliografía** en la que se podrán encontrar todas las referencias consultadas durante su elaboración.
- Finalmente, se adjunta un **Apéndice** que contiene los dos artículos que el autor elaboró como resultado del trabajo llevado a cabo y que fueron aceptados en congresos internacionales.

2 Estado del arte

Como paso previo al desarrollo de la memoria del trabajo que nos ocupa, convendría realizar un estudio meticuloso de las diferentes aplicaciones relacionadas con el área en la que se enmarca este proyecto, que no es otra que los servicios colaborativos y los sistemas de recomendación. Gracias a este análisis, podremos situarnos de una manera más correcta dentro del estado actual del sector en el momento en el que se escriben estas líneas, ayudando por tanto a comprender muchas de las decisiones tomadas a lo largo del trabajo para su realización.

2.1 Plataformas de colaboración

Puesto que uno de los objetivos principales de este Trabajo Fin de Máster ha estado relacionado con la ampliación de la plataforma colaborativa que el alumno desarrolló como Proyecto Fin de Carrera para entornos corporativos, añadiendo nuevos componentes como se verá más adelante, trataremos en primer lugar dos de las suites de software colaborativo o Groupware integrado más importantes que existen en el mercado para cubrir este conjunto de requisitos de colaboración en este tipo de entornos. Como es normal, suelen utilizarse cuando se quiere dar una solución unificada al trabajo existente entre un número elevado de usuarios concurrentes que se encuentran en diversas estaciones de trabajo conectadas a través de una red (Internet o intranet), y que por tanto aglutinan un conjunto de herramientas variado para dar respuesta a esas necesidades.

2.1.1 Lotus

Lotus es una herramienta integrada de Lotus Software (filial de IBM) que está compuesta por varias aplicaciones destinadas a la gestión de trabajo colaborativo. Su penetración en empresas es de carácter mundial, por lo que es una herramienta claramente asentada en el mercado.

El modo de funcionar se basa en el paradigma cliente-servidor, de manera que el lado cliente recibe el nombre de Lotus Notes, mientras que el lado servidor se denomina Lotus Dominio.

Figura 1. Logotipo de IBM Lotus Notes

IBM Lotus Notes [5] combina funciones de correo electrónico, calendario y planificación de agendas con una potente plataforma de escritorio para aplicaciones de colaboración, que entre otras cosas, permite compartir bases de datos de información, ya sean documentos, manuales o foros de discusión, además de permitir la coordinación de flujos de trabajo (workflows).

IBM Lotus Domino [6] ofrece funciones de colaboración que se pueden desplegar como una infraestructura central de planificación empresarial y de correo electrónico, como una plataforma de aplicaciones empresariales o como ambas cosas, aportando también un entorno fiable de mensajería y colaboración entre los empleados.

Además, las aplicaciones que ofrece Domino pueden ser accedidas mediante Notes o utilizando un navegador web, aspecto que aporta mayor flexibilidad a la hora de ser usadas desde diferentes localizaciones o plataformas.

Por otro lado, y de cara a realizar un proceso adaptativo de Lotus al nuevo contexto de las redes sociales, surge IBM Lotus Connections [7].

Figura 2. Logotipo de IBM Lotus Connections

Se caracteriza por ser un software social específicamente diseñado para los negocios, que, además de integrar todas las herramientas disponibles en el resto de programas de la suite de Lotus, añade un claro interés por la compartición y gestión de conocimientos en la organización, comunicando de manera inmediata a socios y clientes al establecer de forma dinámica nuevas conexiones entre las personas.

Para ello, se utilizan entre otras cosas perfiles de usuario para conocer el área de conocimiento y los intereses de cada trabajador, blogs gestionados por personas o grupos pertenecientes a la organización o sistemas de compartición de enlaces (bookmarks).

Adicionalmente, es posible también en este caso acceder a estos componentes desde un navegador web o desde otros programas como IBM Lotus Notes o Microsoft Office.

2.1.2 Microsoft Office Communicator 2007

Microsoft Office Communicator 2007 [8] es un cliente de comunicaciones en tiempo real (síncrono) unificado, que mejora la productividad al permitir a las personas comunicarse fácilmente con otras que se encuentran en zonas horarias o lugares

distintos mediante diversas opciones de comunicación, como mensajería instantánea, voz o vídeo.

Figura 3. Logotipo de Microsoft Office Communicator 2007

Al ser un producto desarrollado por Microsoft, la integración con los programas del sistema Microsoft Office 2007 (Word, Excel, PowerPoint, OneNote, Groove y SharePoint Server) es inmediata, proporcionando un entorno de usuario común para su uso.

Entre sus principales funcionalidades, destaca el acceso a la información de presencia y de directorio de usuarios, la capacidad de convertir las conversaciones desde mensajería instantánea en conversaciones telefónicas o de videoconferencia sobre la marcha, la posibilidad de controlar las comunicaciones entrantes con alertas, redirecciones automáticas de llamadas y configuración manual del indicador de estado.

2.2 Sistemas de recomendación y métodos de clustering

Con el paso de los años, Internet se ha convertido en el mayor depósito de información que la humanidad posee, creciendo día a día a un ritmo vertiginoso que pocos podrían haber imaginado tras su concepción. Esta información proviene de todo tipo de fuentes completamente heterogéneas, que van desde estudios científicos especializados en áreas de investigación minoritarias, a datos sobre películas o libros conocidos por la mayoría de los usuarios de Internet y que por este hecho, crece a mayor velocidad que en el primer caso, ya que no hay que olvidar que uno de los mayores logros que ha ofrecido Internet es que cualquiera puede añadir más conocimiento a la Web en la manera que desee.

Debido a esta pluralidad existente tanto en los tipos de datos como en las fuentes que los producen, infinidad de aplicaciones web han surgido en los últimos años con el objetivo de agrupar o reunir a usuarios interesados en dichos campos, y por tanto a la información o conocimiento que se crea alrededor de ellos. Este intento de ordenar o reunir semejante cantidad de información cristalizó en una primera aproximación en servicios como la Wikipedia, convirtiéndose en poco tiempo en una enciclopedia electrónica de dimensiones titánicas.

Por otro lado, y de manera más reciente han surgido diferentes portales web dedicados no sólo a reunir y ordenar información sobre diferentes temáticas, sino también a realizar recomendaciones sobre ellas en base a las preferencias o gustos de

los usuarios que se registran en ellas, basándose por tanto en potentes motores de recomendación que analizan cantidades de datos enormes. Estos sistemas de recomendación como GetGlue [9] han ganado adeptos rápidamente ya que permiten de manera transparente al usuario realizar un clustering dentro del conjunto total de usuarios que facilita la recomendación de productos (libros, películas, etc.) al agrupar a usuarios por sus preferencias y gustos, y por tanto, poder recomendar a un usuario productos que otros usuarios de su cluster han votado favorablemente. Es más, portales web que no son sistemas de recomendación en sí mismos, o al menos, no surgieron como tal, han adquirido recientemente estas capacidades, llegando a crear motores propios como es el caso de Amazon, que ha conseguido implementar su propio sistema con bastante éxito [10].

En la sección 3 de "Líneas de investigación y trabajos realizados" se describirá la arquitectura e implementación de un sistema de recomendación desarrollado por el alumno. Por ello, en primer lugar vamos a centrar nuestro foco de atención en los procesos de clustering que tanta importancia tienen como paso previo a poder realizar una recomendación adecuada en los sistemas anteriormente comentados.

2.2.1 ¿Qué entendemos por clustering?

Hablamos de clustering para referirnos a un proceso de análisis complejo que consiste en dividir un conjunto de datos en subconjuntos más pequeños (clusters), tal que los elementos agrupados en cada uno de dichos clusters sean similares en algún sentido que hemos definido a priori. En otras palabras, dichos elementos deben ser cercanos entre sí de acuerdo a una distancia métrica que elegiremos para realizar el proceso de clustering.

Por tanto, de manera intuitiva, cuando hablamos de clustering, nos referimos en cierto sentido a una clasificación de elementos, ya sean estos usuarios, películas, noticias o libros que serán agrupados atendiendo a una o varias de sus propiedades. Por ejemplo, un conjunto de usuarios podría segmentarse atendiendo a su edad, sexo… o a una combinación de varios. Como vemos, este tipo de técnicas no son ni mucho menos nuevas en nuestra sociedad, ya que cualquier estudio estadístico de carácter social o encuesta con la que nos encontremos tiende a producir como resultado un número determinado de clusters dentro del conjunto total de usuarios estudiados. Por ello, aunque en Internet su uso es relativamente reciente, en el mundo empresarial o en el de las ciencias sociales, las herramientas o algoritmos de clustering han cumplido desde hace mucho un papel fundamental.

Distancia métricas

Como hemos comentado anteriormente, para segmentar a un conjunto de usuarios en un proceso de clustering, debemos basarnos en una distancia métrica que nos

informe de la similitud entre los diferentes puntos estudiados, pudiendo por tanto agrupar aquellos que estén más cercanos dependiendo del nivel de precisión que busquemos a la hora de definir la cercanía entre dos elementos.

No es el objetivo de este artículo detenernos demasiado en este aspecto, aunque siempre resulta interesante destacar algunas de las distancias métricas más usuales que se utilizan actualmente para calcular la similitud entre los elementos analizados:

- **Distancia Euclídea**: es probablemente la más conocida, y se establece como la distancia "ordinaria", es decir, la línea recta entre dos puntos x(x1,…, xn) e y(y1,…,yn) de un espacio euclídeo.

$$d(x,y) = \|x - y\| = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

- **Distancia Manhattan**: basada en la geometría Taxi Cab [11] y considerada por Hermann Minkowski en el siglo 19, es una forma de geometría en la cual la métrica usual de la geometría euclídea es reemplazada por una nueva métrica en la cual la distancia entre dos puntos es la suma de las diferencias (absolutas) de sus coordenadas.

$$d_1(x,y) = \|x - y\|_1 = \sum_{i=1}^{n}|x_i - y_i|$$

Si observamos la comparación gráfica realizada en la siguiente figura, vemos que la Euclídea es la distancia más corta entre los puntos (0,0) y (1,1) con valor √2, ya que la Manhattan es 2. Sin embargo, ésta última es mucho más rápida de calcular debido a que las operaciones involucradas son más sencillas, lo que nos permitirá aumentar la velocidad de nuestro sistema a cambio de perder precisión en el resultado.

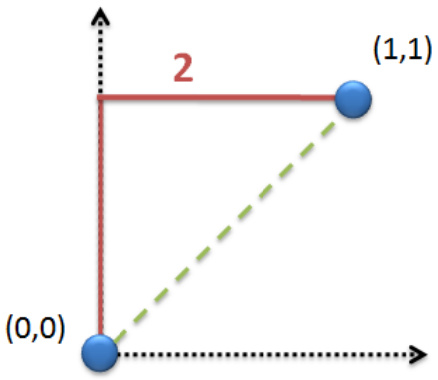

Figura 4. Representación gráfica de la distancia Euclídea (línea verde) y Manhattan (línea roja)

Clustering Jerárquico

El primero de los métodos de clustering clásicos es el Jerárquico. Partiendo de un conjunto de elementos (representado en la Figura 5 por la parte izquierda), podemos realizar un análisis de arriba-abajo, o de abajo-arriba.

En el primer caso, partimos de una situación en la que cada elemento es un cluster en sí mismo, y a partir del uso de la distancia métrica elegida, vamos generando clusters cada vez mayores dando por válidos valores de distancias entre elementos cada vez más altos, hasta que finalmente agrupamos a todos los elementos en el mismo cluster.

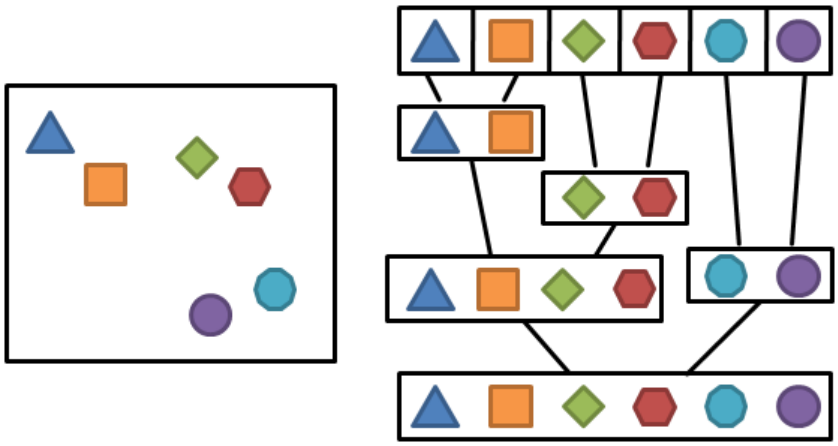

Figura 5. Ejemplo de clustering Jerárquico

En el segundo caso, partimos de un cluster único para todos los elementos, y lo segmentamos cada vez más al aumentar poco a poco la precisión que utilicemos en la distancia métrica, siendo el último paso posible el momento en el que cada elemento sea un único cluster, como podemos ver en la parte derecha de la Figura 5.

Clustering Particional

El Segundo de los métodos clásicos es el de tipo Particional, y es probablemente el más extendido e intuitivo. Consiste en generar todos los clusters de una sola vez teniendo en cuenta todos los elementos de manera simultánea, como podemos ver en el ejemplo que ilustra la siguiente figura. Este camino implica evidentemente un mayor coste computacional a cambio de una mayor precisión, aunque también existe la posibilidad de realizar el cálculo de dichos clusters mediante varias iteraciones, hasta conseguir el nivel de convergencia y precisión deseado.

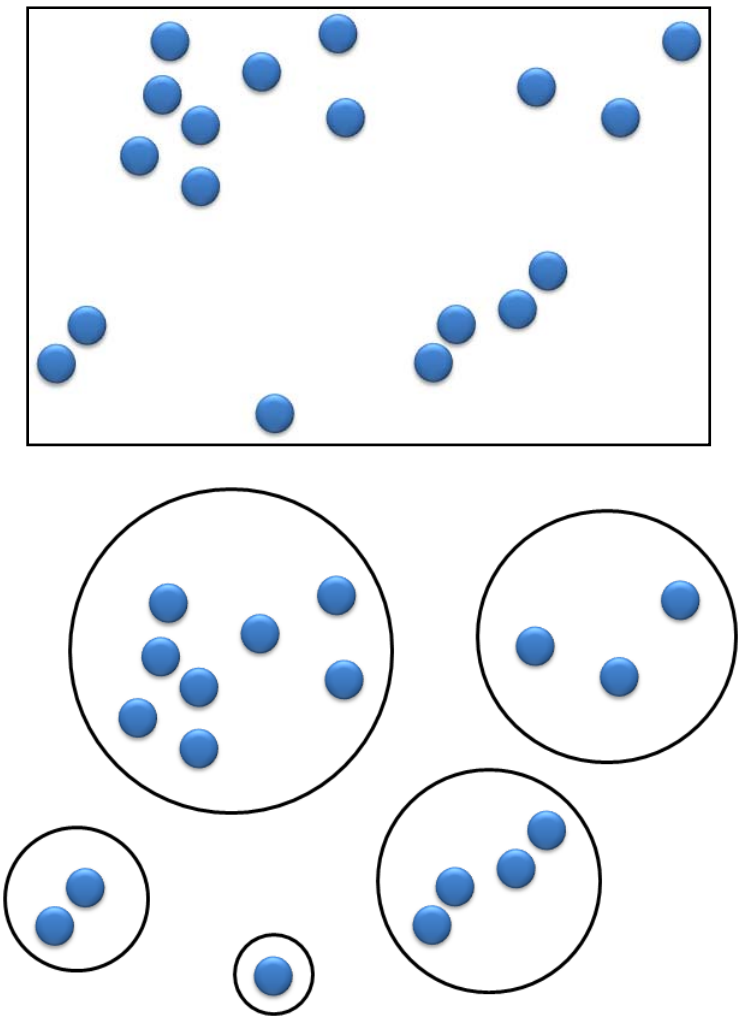

Figura 6. Ejemplo de clustering Particional

2.2.2 K-means clustering

Aunque el algoritmo estándar que se encuentra bajo el término "K-means" se conoce desde 1957, fue en 1967 cuando se acuñó el nombre actual en [12], por lo que no es ni mucho menos un algoritmo de reciente descubrimiento. K-means pertenece a la familia de los métodos de clustering particional y se conoce y utiliza en muchos ámbitos por ser bastante simple.

Los pasos que se siguen para llevarlo a cabo son los siguientes dado un conjunto de puntos que deseamos estudiar bajo un proceso de clustering (ilustrado en las figuras siguientes de manera resumida para un conjunto de 10 puntos y k=3):

1. Elegir k número de clusters.

2. Elegir k puntos adicionales dentro del espacio de datos al que se quiera aplicar el algoritmo para ser los centros de dichos clusters (Figura 7).

3. Calcular la distancia de todos los puntos a los k puntos designados como centros.

4. Asignar a cada punto el centro más cercano (Figura 8).

5. Una vez agrupados tras el paso anterior, calcular el punto central medio (centroide) de todos de los clusters en base a los puntos que lo conforman.

6. Reemplazar los k centros por los nuevos puntos medios calculados en el paso anterior.

7. Iterar los pasos 3 al 6 hasta que se consiga convergencia en forma de centros estáticos en todos los k clusters (Figura 9).

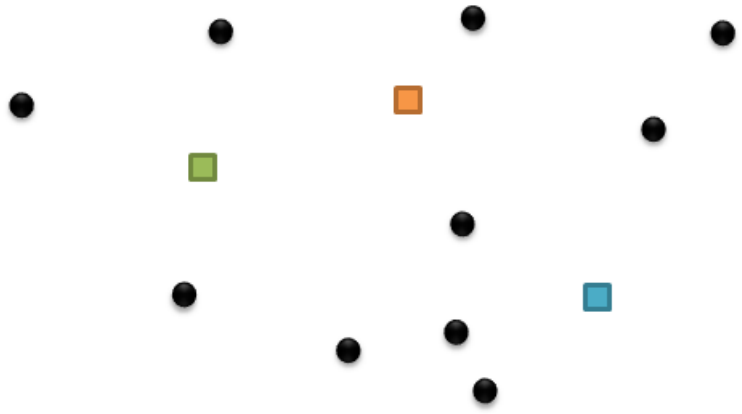

Figura 7. Selección inicial de centros en K-menas con k = 3

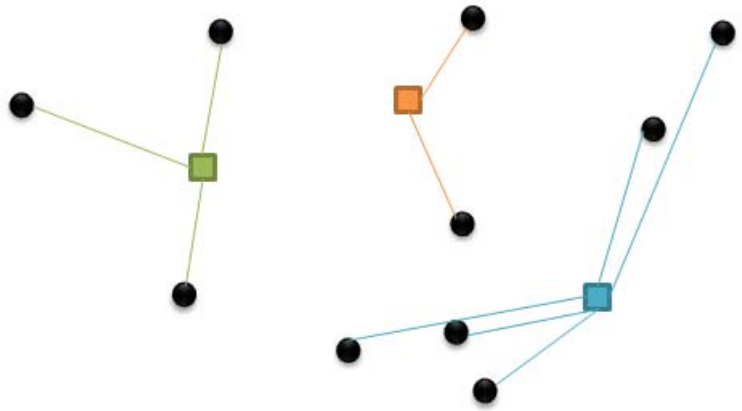

Figura 8. Primera asignación de puntos al centro más cercano

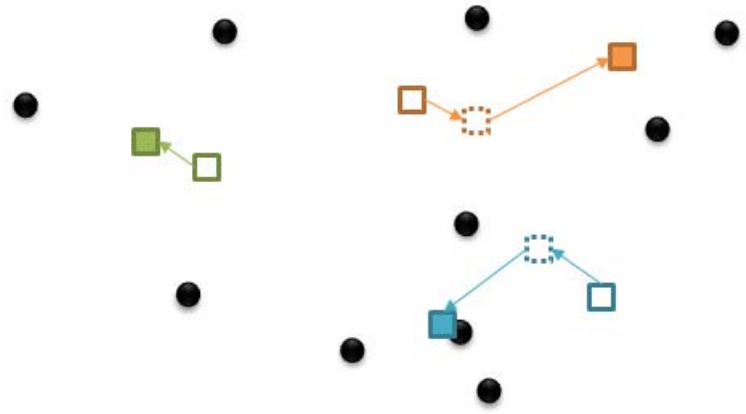

Figura 9. Iteraciones realizadas en K-menas hasta alcanzar los centros de los nuevos clusters. Los cuadrados con línea continua corresponden a la primera iteración, en discontinua a la segunda y los rellenos a la posición final de los centros tras converger el algoritmo

Matemáticamente podríamos expresarlo de la siguiente manera:

$$\arg min_S \sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2$$

Siendo (x1, x2, …, xn) un conjunto de observaciones donde cada una de ellas es un vector d-dimensional real. Luego, el objetivo del proceso de clustering mediante K-means consiste en particionar las n observaciones en k conjuntos (k < n) S = {S1, S2, …, Sk} que minimicen la diferencia de cuadrados en cada cluster, teniendo en cuenta que μi es la media de los puntos pertenecientes al conjunto Si.

Ahora bien, aunque el algoritmo como vemos es bastante sencillo, existen problemas relativamente graves en lo que a tiempo de computación se refiere.

Por un lado, la complejidad matemática asociada a este problema es NP-dura, ya que depende de:

$$k * n * O(distancia\ métrica) * num(iteraciones)$$

Esto sin duda alguna puede devolver valores muy altos, lo que implicaría un gran consumo de tiempo de procesador y memoria del sistema para poder realizar todos estos cálculos en un tiempo aceptable. Por tanto el riesgo de colapsar el sistema al completo es alto si trabajamos con conjuntos de puntos muy grandes, si la distancia métrica elegida es complicada (las dos que hemos comentado son relativamente sencillas, pero existen algunas mucho más complejas de calcular), o si el número de iteraciones para alcanzar la convergencia debido al número de clusters seleccionados es muy alto. Por ello, aunque seamos capaces de distribuir estos procesos de cálculo mediante técnicas de cloud computing como las que ofrece Mahout (el cual analizaremos en apartados posteriores) al disponer de distribución de tareas usando

Hadoop (lo veremos en detalle más adelante), la carga que apliquemos al sistema puede seguir siendo demasiado alta para la potencia de cálculo disponible.

Además, existe un riego adicional a la hora de elegir el valor de k y los centros iníciales (que se hace de manera aleatoria) con los que se comenzará la ejecución del algoritmo, ya que si son inapropiados o elegimos k pequeña para disminuir el nivel de complejidad, los resultados pueden ser muy pobres y por tanto ineficientes para nuestro sistema de recomendación.

Por ello, y principalmente debido a esta serie de problemas, numerosas líneas de investigación se han centrado durante muchos años en intentar mejorar, o al menos paliar, algunas de las debilidades del algoritmo K-menas. Veamos una de las más interesantes y que mejores resultados ha obtenido.

K-means++

Partiendo de los problemas ya comentados relacionados con la correcta elección del parámetro de entrada k y los centros iniciales, K-means++ es un algoritmo destinado a elegir correctamente dichos parámetros para conseguir unos resultados más precisos. De esta manera, este algoritmo propuesto en [15] en 2007 intenta evitar la generación de clusters pobres o ineficientes que en algunas ocasiones aparecen tras utilizar K-means estándar.

Antes de comentar los pasos que se llevan a cabo con este algoritmo, detengámonos un instante para evaluar cómo se podría mejorar la elección de los centros. Si se hace este ejercicio mental, la intuición nos dice rápidamente que la opción más interesante se basa en elegir los centros de los clusters iniciales de manera que estén repartidos lo más equitativamente en el espacio de puntos con el que vamos a trabajar, consiguiendo de esta manera que tras una primera iteración de K-means consigamos clusters lo más alejados posibles ente ellos, y por supuesto, con el menor solapamiento.

Luego, siguiendo este razonamiento vamos a describir los pasos que se siguen para conseguir este algoritmo a partir de un conjunto de puntos inicial al que llamaremos X:

1. Elegimos un centro c1 escogido mediante una distribución aleatoria uniforme.

2. Para cada punto restante x, calculamos D(x), la distancia entre x y el centro más cercano que ya haya sido seleccionado.

3. Añadimos un nuevo centro ci. Para ello lo tomamos teniendo en cuenta el mayor peso basado en una distribución de probabilidad llamada D2 dada por la siguiente fórmula:

$$\frac{D(x)^2}{\sum_{x \in X} D(x)^2}$$

4. Se repiten los pasos 2 y 3 hasta que se han calculado los k centros.

5. Ahora, una vez se tienen los centros iniciales elegidos, se procede a utilizar el algoritmo de K-means estándar explicado anteriormente.

Este método de elección de centros, en la línea de los procesos de "semillas" que se suelen aplicar a los algoritmos de aprendizaje, ofrece una mejora considerable del error que aparecía comúnmente en el algoritmo K-means.

Bien es verdad que este proceso implica un gasto de cómputo y tiempo inicial que antes no existía, pero a cambio se gana una velocidad de convergencia mucho mayor que de manera general mejora el proceso al completo. Es más, en [15] se demuestra mediante resultados empíricos aplicados a conjuntos de datos reales y sintéticos que, tras aplicar este algoritmo, los autores conseguían una disminución del error que iba desde el doble a mil veces el original.

Adicionalmente, también se demuestra que de manera aproximada, K-means++ es un algoritmo que posee complejidad $O(\log(k))$, donde k es el número de clusters elegidos, por lo que hemos conseguido disminuir de manera significativa la complejidad y por tanto los recursos necesarios en cuanto a tiempo y potencia de cálculo en comparación con el algoritmo estándar de K-means.

2.2.3 Canopy clustering

Debido a la importancia de K-means como algoritmo de clustering, nuevas propuestas han surgido de manera reciente no como un añadido a éste (como era el caso de K-means++), sino como algoritmos de clustering completos que pueden ser usados de manera individual o en conjunción con los anteriormente comentados siguiendo en parte su filosofía.

Este es el caso del algoritmo de clustering Canopy [16], surgido en el año 2000 y que fue ideado especialmente para trabajar con conjuntos de datos muy grandes de una manera mucho más rápida que con las propuestas anteriores, puesto que está indicado para trabajar de manera paralela en varias máquinas, consiguiendo de esta forma mayor eficiencia.

La idea principal se basa en dividir o segmentar el conjunto de datos inicial en clusters llamados canopies que pueden estar solapados en parte entre sí, usando en estos cálculos distancias métricas sencillas (como la Manhattan) que permitan acelerar estos cálculos. A continuación, se puede aplicar métodos de clustering más pesados

como K-means, pero sólo dentro de cada canopy, reduciendo por tanto el cálculo a conjuntos de datos mucho más pequeños.

De manera detallada, los pasos que sigue este algoritmo partiendo de que todos los objetos o datos a tratar son puntos en un espacio multidimensional son:

1. Se elige una distancia métrica de cálculo rápido.

2. Se definen dos umbrales de distancia T1 y T2 a partir de la elección anterior, teniendo en cuenta que T1 > T2.

3. Se elige aleatoriamente un punto del conjunto de puntos total y se extrae de él creando un canopy que lo contiene.

4. Se mide la distancia desde dicho punto al resto de puntos del conjunto.

5. Para cada distancia calculada, si es menor que T1, se añade este punto al canopy anterior. Si además, también es menor que T2, se elimina dicho punto del conjunto de puntos restantes. Así, si el punto está muy cerca de punto elegido inicialmente (su distancia es menor que T2), ya no se sigue analizando como un posible centro de un nuevo canopy.

6. Se iteran los pasos 3, 4 y 5 hasta que el conjunto de puntos inicial está vacío, obteniendo un grupo de canopies en el que cada uno de ellos contendrá uno o más puntos. Es decir, un punto podría estar en más de un canopie.

7. Ahora si se desea se puede utilizar un método de clustering más riguroso como K-means usando distancias métricas más complejas o precisas (por ejemplo, la Euclídea), pero sólo se aplica a cada uno de los canopies, asumiendo que aquellos puntos fuera del canopy están a distancia infinita, es decir, se ignoran. De esta forma, las distancias a calcular serán más cortas y por tanto se necesitará menos tiempo de cómputo.

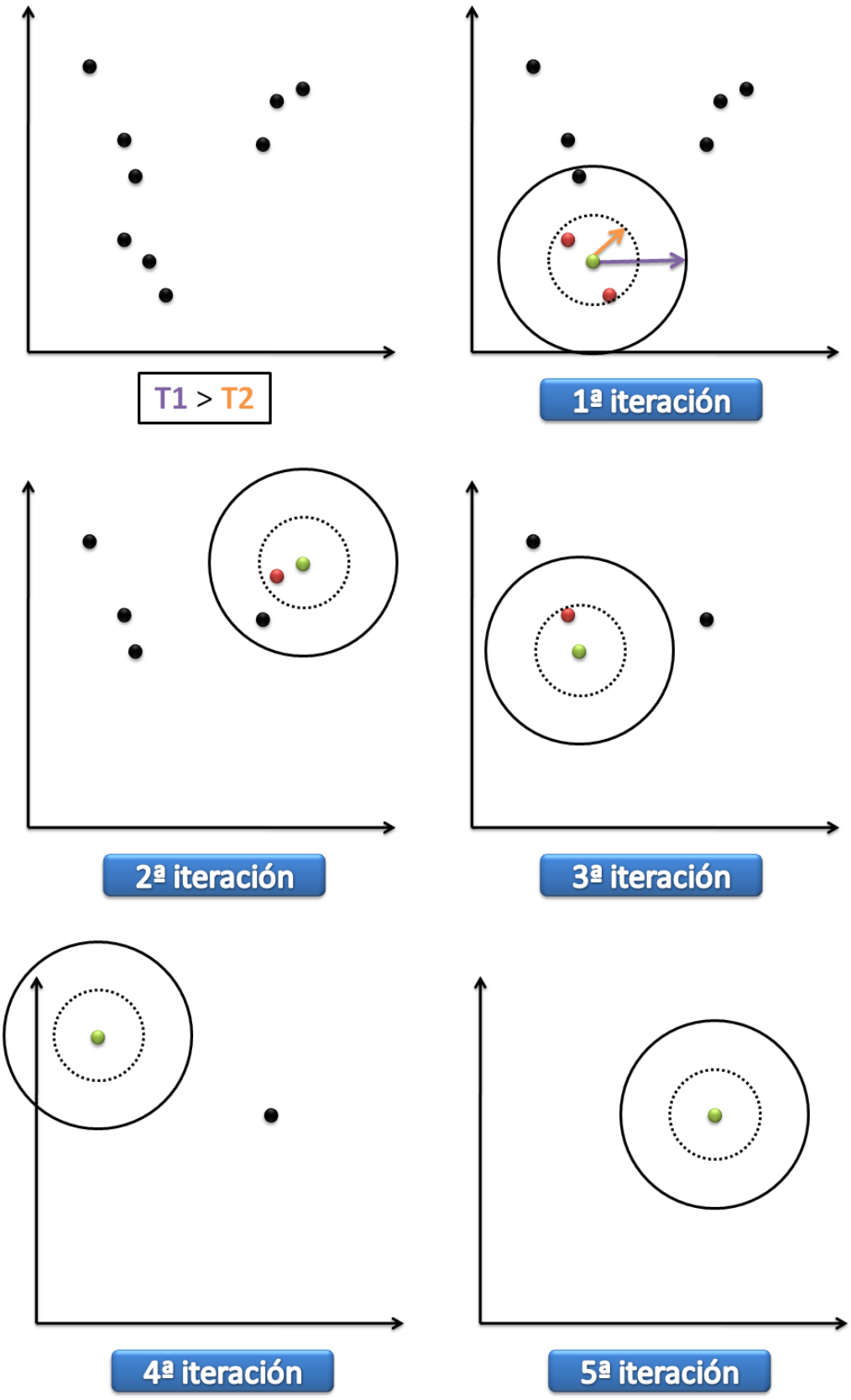

Figura 10. Conjunto de datos inicial, distancias umbrales T1 y T2 elegidas, e iteraciones de ejemplo de Canopy clustering con 9 puntos

Si analizamos el ejemplo de ejecución del algoritmo de clustering Canopy ilustrado por las figuras, vemos que tras aplicar los pasos descritos anteriormente, obtenemos un total de 5 canopies a partir del conjunto de puntos inicial, utilizando para formarlos dos distancias T1 (circunferencia continua) y T2 (circunferencia discontinua). Así, los puntos más claros son los centros de los diferentes canopies obtenidos, mientras que los puntos grises que se encuentran dentro de las circunferencias discontinuas serán aquellos que se eliminarán del conjunto de posibles centros de canopies por esta muy cerca de alguno ya existente.

Por lo tanto, si nos fijamos en el resultado final comprobaremos que algunos canopies están solapados, pues comparten uno o más de sus puntos como es por ejemplo el caso de los canopies de las iteraciones 1 y 3, o el caso de iteraciones 2 y 5.

Ahora, y como ya hemos comentado con anterioridad, se podría aplicar K-means sobre cada uno de los canopies obtenidos, siendo por tanto la complejidad mucho menor al pasar de trabajar con un conjunto de n = 9 puntos, a trabajar con canopies de cómo mucho n = 4 puntos en el ejemplo propuesto. Éstos además requieren un k menor, y por tanto menos iteraciones. Es decir, como resultado de aplicar Canopy antes de K-means, hemos disminuido tres de los cuatro factores que aumentaban la complejidad de éste último (k, n y el número de iteraciones), a cambio de aplicarlos sobre varios canopies, en los que muchas veces puede resultar trivial (como es el caso de las iteraciones 4 y 5 que sólo tienen un punto). Luego, podemos confirmar como ya expusimos antes que es posible elegir una distancia métrica más precisa al aplicar K-means en los canopies al haber reducido la complejidad en el resto de factores implicados, ganando de manera general en velocidad y tiempo de cómputo.

2.2.4 Aplicación en sistemas reales

Como ya hemos comentado anteriormente K-means es un algoritmo de clustering que se ha venido usando en diferentes áreas de conocimiento como las aplicaciones médicas, los estudios sociales o los estudios de mercado, desde hace muchos años.

Adicionalmente, en el campo de los sistemas de información y comunicaciones podemos encontrar ejemplos muy importantes que merece la pena mencionar.

Google MapReduce

En primer lugar, podemos hablar de Google [17] que ha hecho un esfuerzo importante a la hora de adquirir y perfeccionar este tipo de métodos de clustering debido al enorme conjunto de datos que analiza diariamente y que necesita ser segmentando. Especial importancia tiene su modelo de programación MapReduce [18] cuya implementación orientada a procesar y generar grandes conjuntos de datos está

usándose actualmente a diario en sus sistemas para gestionar las innumerables bases de datos que poseen.

Concretamente, MapReduce trabaja a partir de la función Map para procesar pares clave/valor y generar conjuntos intermedios de pares clave/valor que mediante la función Reduce combina todos los valores intermedios asociados con la misma clave, evitando de esta manera la redundancia de claves. Para realizar esa búsqueda de valores que poseen la misma clave se utiliza Canopy para detectar aquellos conjuntos de datos que poseen la misma clave y que por tanto son cercanos en cuanto a distancia métrica.

Además, este estilo funcional de implementación de MapReduce permite paralelizar automáticamente la ejecución de las diferentes tareas involucradas para el cálculo de canopies y las funciones Map y Reduce, aunque hablaremos de este enfoque distribuido en el apartado siguiente.

Mahout

En segundo lugar, tenemos el motor de recomendación con capacidad de filtrado colaborativo Mahout **¡Error! No se encuentra el origen de la referencia.** (anteriormente conocido como Taste), desarrollado en Java como proyecto de código libre bajo licencia Apache Lucene desde hace varios años.

Figura 11. Logotipo de Mahout

Mahout incorpora un amplio conjunto de algoritmos de clustering entre los que se encuentran K-means, Canopy, Fuzzy K-means **¡Error! No se encuentra el origen de la referencia.** (también conocido como Fuzzy C-means) o Dirichlet **¡Error! No se encuentra el origen de la referencia.**. Adicionalmente pone a disposición de los programadores implementaciones de varias distancias métricas muy utilizadas junto a estos algoritmos como son la Euclídea o la Manhattan, además de algoritmos de recomendación como la correlación de Pearson **¡Error! No se encuentra el origen de la referencia.**, la del Coseno (o Coseno ajustada) o la más reciente Slope One **¡Error! No se encuentra el origen de la referencia.**. De esta manera, Mahout permite trabajar con conjuntos de datos grandes para realizar sobre ellos procesos de clustering, clasificación, filtrado colaborativo y recomendaciones sobre los resultados obtenidos

Por otro lado, una característica importante que merece la pena destacar llegados a este punto, es que Mahout está desarrollado para permitir una arquitectura de despliegue y ejecución escalable. Esto es así porque se da la opción de construir nuestro sistema a partir de Mahout sobre el proyecto de código libre bajo licencia Apache conocido como Hadoop **¡Error! No se encuentra el origen de la referencia.**. Éste utiliza el paradigma MapReduce comentando con anterioridad para distribuir las tareas de clustering asociadas a los procesos de la función Reduce en diferentes máquinas, reduciendo por tanto los tiempos de ejecución globales. Es más, muchas de las implementaciones de los algoritmos de clustering proporcionadas por Mahout están preparadas para trabajar directamente con esta metodología de tareas paralelas distribuidas entre varias máquinas, lo que simplifica a programadores inexpertos en este campo hacer uso de ellas.

Figura 12. Logotipo de Hadoop

Específicamente, los pasos que se dan para distribuir la ejecución de algoritmos de clustering Canopy más K-means que ofrece Mahout son:

1) **Adecuación de los datos de entrada**: se parsea el conjunto de datos de entrada para transformarlo a alguno de los formatos específicos aceptados por la implementación elegida.
2) F**ase de generación de los canopies**: se ejecuta la función Map en paralelo sobre cada uno de los subconjuntos en los que se ha dividido el total de los datos, aplicando la distancia métrica junto a los valores umbrales T1 y T2 definidos para generar los canopies. En el mapeo, cada punto que se encuentra dentro de un canopy existente se manda a un mezclador asignándole el identificador del canopy al que corresponde. Tras ordenar todos los puntos por el identificador de canopy, el mezclador itera sobre todos los puntos que poseen el mismo identificador, reuniendo los puntos que compartan la misma clave y normalizando para obtener un centroide de canopy que es devuelto a la salida con la clave "centroid" y el valor que le corresponda. Finalmente, la función Reduce recibe todos estos centroides iniciales que se han calculado de manera distribuida y nuevamente aplica el cálculo de canopies usando la distancia y los umbrales anteriores, para de esta manera producir un conjunto final de centroides de canopies que se mandan a la salida.
3) **Fase de clustering**: en esta fase partimos del resultado final anterior para, de manera distribuida, aplicar un algoritmo de clustering más complejo como K-

means a cada uno de los canopies anteriores. Para ello tenemos en cuenta que cada mapeador, al tener la misma definición de canopy que el resto (distancia y umbrales), puede comprobar rápidamente que puntos de los que está tratando están asignados a uno o más canopies, teniendo por tanto esto en cuenta para el proceso de combinar finalmente los resultados de cada tarea que hemos ejecutado distribuidamente.

Luego, la aplicación de las funciones MapReduce al cálculo de clustering de manera distribuida para la obtención de canopies, además de los clusters finales posteriores obtenidos mediante K-means o con cualquier otro algoritmo similar, nos permite una mejora en los tiempos de ejecución. Esto es así porque podemos distribuir estas tareas de cálculo entre varias máquinas, balanceando por tanto la carga total del sistema, y sobre todo, acelerando el cálculo de clusters en sistemas con conjuntos iniciales de datos de entrada muy grandes, que cómo ya comentamos en la introducción, es a lo que cada vez se tiende más debido al continuo crecimiento de las bases de datos que subyacen en Internet.

Por tanto, y visto que la complejidad de los datos usados por sistemas de recomendación basados en perfiles de usuarios es cada vez más compleja, esta solución distribuida se antoja cada vez más necesaria para aquellas aplicaciones que quieran desplegar en la Web.

2.3 Tecnologías utilizadas para el desarrollo

En esta sección realizaremos un repaso exhaustivo por todas las tecnologías involucradas en el desarrollo del trabajo realizado durante este máster, tanto las que son utilizadas indirectamente por ser parte de componentes utilizados en la elaboración de las diferentes plataformas, como aquellas que han sido directamente responsables de la implementación de las mismas.

2.3.1 Ruby on Rails

Ruby on Rails, también conocido como "RoR" o "Rails" es un framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby, siguiendo el paradigma de la arquitectura Modelo Vista Controlador (MVC) y el uso de la Metodología Ágil como forma de desarrollo y programación.

El gran punto a favor de Rails es la gran capacidad y rapidez que ofrece para desarrollar aplicaciones escribiendo menos código que con otros frameworks, con un mínimo de configuración y con una serie de filosofías y convenciones bien definidas a la hora de llevar a cabo los proyectos web.

El lenguaje de programación Ruby

Ruby es un lenguaje de programación interpretado, orientado a objetos y distribuido bajo licencia de software libre. Combina sintaxis inspirada en Python y Perl con características de SmallTalk. Creado en Japón a mediados de los noventa, fue inicialmente diseñado y desarrollado por Yukihiro Matsumoto bajo la idea de dar lugar a un lenguaje de programación que se asemejase al natural.

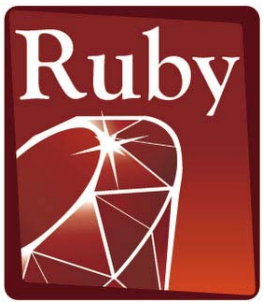

Figura 13. Logotipo de Ruby

Ruby es un lenguaje interpretado, es decir, no se compila, aspecto que permite una mayor flexibilidad a la hora de generar código de manera dinámica, cualidad que es de suma importancia para el desarrollo de aplicaciones web y que fue una de las principales razones para desarrollar el framework de Rails sobre él.

Al ser un lenguaje de programación completamente orientado a objetos, todos los tipos de datos son un objeto, incluidas las clases y tipos que otros lenguajes define como primitivas, (como podrían ser los tipos numéricos o las booleanos en Java). A su vez, toda función es un método, y las variable siempre son referencias a objetos, no los objetos en sí mismos.

Entre las grandes ventajas del lenguaje, destacan su flexibilidad para funcionar en múltiples paradigmas, ya que permite a sus usuarios alterarlo libremente, pues las partes esenciales de Ruby pueden redefinidas a placer.

Ruby utiliza un sistema propio para gestionar paquetes que provee un formato estándar para distribuir programas escritos en Ruby y bibliotecas. El formato en el que los programas son contenidos recibe el nombre de "gema" (gem). La herramienta que gestiona estas gemas es RubyGems. El framework Rails se instala como una gema, y es una de las razones principales que ha hecho que Ruby haya aumentando su penetración en el mercado en los últimos años al ir de la mano de Rails.

El framework Rails

Rails fue el resultado del trabajo de David Heinemeier Hansson sobre una herramienta de gestión de proyectos que dio lugar a una aplicación web llamada 37signals. David lanzó Rails como proyecto de código abierto en Julio de 2004.

Figura 14. Logotipo Rails

Los principios fundamentales sobre los que se sustenta son:

- **"Don´t Repeat Yourself" o DRY (No te repitas)**. Intenta inculcar que las definiciones deberían hacerse una sola vez. Dado que Ruby on Rails es un framework de pila completa, los componentes están integrados de manera que no hace falta establecer puentes entre ellos.
- **"Convention Over Configuration" o COC (Convención sobre configuración).** Significa que el programador sólo necesita definir aquella configuración que no es convencional. Así, cuando se diseña una aplicación partiendo de cero sin una base de datos preexistente, el seguir las convenciones de Rails significa usar menos código (aunque el comportamiento puede ser configurado si el sistema debe ser compatible con un sistema heredado anterior).

De esta manera, y bajo las premisas anteriores, Rails se desarrolla bajo una arquitectura MVC (Modelo-Vista-Controlador), muy utilizada actualmente para organizar la programación de aplicaciones web orientadas a objetos sobre bases de datos. El Modelo consiste en las clases que representan a las tablas de la base de datos. La Vista es la lógica de visualización, o cómo se muestran los datos de las clases del Controlador. Con frecuencia, en las aplicaciones web la vista consiste en una cantidad mínima de código incluido en HTML. Por último, las clases del Controlador responden a la interacción del usuario e invocan a la lógica de la aplicación, que a su vez manipula los datos de las clases del Modelo y muestra los resultados usando las Vistas. En las aplicaciones web basadas en MVC, los métodos del controlador son invocados por el usuario usando el navegador web.

Adicionalmente, Rails provee varias herramientas con la intención de conseguir un desarrollo de aplicaciones de la forma más sencilla posible, entre la que se encuentra de manera señalada el scaffolding, que construye de forma automática tanto modelos, como vistas o controladores para un sitio web sencillo. El paquete básico también incluye un servidor Ruby muy simple llamado Webrick y un sistema make llamado

Rake. Con la inclusión de estas herramientas, se provee un entorno básico de desarrollo que viene con todas las versiones del software.

Actualmente, con la versión 2.3 de Rails se puede afirmar que poco a poco ha logrado asentarse como alternativa para el desarrollo de aplicaciones web a tecnologías como J2EE o PHP. Además, cuenta con una comunidad muy activa que actualiza el framework y aporta nuevas funcionalidades con gran frecuencia, lo que le da una mayor potencia de manera global.

2.3.2 Java

Java [23] es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para su ejecución por la máquina virtual, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

Figura 15. Logotipo de Java

Hasta la fecha, la plataforma Java ha atraído a más de 6,5 millones de desarrolladores de software y se utiliza en los principales sectores de la industria de todo el mundo, estando presente en un gran número de dispositivos, equipos y redes que tienen instalado JRE.

Su versatilidad y eficiencia, así como la portabilidad de su plataforma y la seguridad que aporta, la han convertido en la tecnología ideal para su aplicación en entornos distribuidos de todo tipo. Actualmente Java está disponible en más de 4.500 millones dispositivos (ordenadores, teléfonos móviles, tarjetas inteligentes, impresoras, juegos, etc.)

2.3.3 REST

El término REST surge por primera vez en el año 2000 en la tesis doctoral de Roy Fielding [24] (uno de los principales autores de la especificación del protocolo HTTP), describiendo un nuevo estilo de arquitectura para sistemas en red. REST es un acrónimo que quiere decir Transferencia de Estado Representacional (Representational State Transfer).

De esta forma, la idea principal de su tesis partió de la existencia de recursos que actualmente componen la Web. Un recurso es un objeto de interés para el usuario. Estos elementos de información pueden ser accedidos utilizando un identificador Uniforme de Recurso (URL). Por ello, REST promulga la necesidad de tener URLs claros que los identifiquen.

Por ejemplo, si tuviéramos un avión Boeing 747 y quisiéramos acceder a él, la compañía puede haber definido el recurso con la siguiente URL:

http://www.boeing.com/aircraft/747

Accediendo a este URL se nos devuelve una representación del recurso avión Boeing 747 (por ejemplo, la página Boeing747.html). El resultado de pulsar en algún enlace dentro de esta página es que vamos a acceder a otro recurso. Esta nueva representación sitúa a la aplicación en un nuevo estado. De esta forma, la aplicación cliente cambia de estado con cada representación de un recurso.

Características y filosofía de REST

Antes de nada se debe tener presente que REST no es un estándar, sino un estilo de arquitectura o una filosofía de cómo se pueden entender y diseñar servicios web con ese estilo, que evidentemente, no es una regla a seguir de manera fija, sino una recomendación. Por ello, la motivación que llevó a la creación de REST es capturar las características de la Web que la hacen exitosa. De esta forma, REST es de alguna manera una guía de estilo de cómo deberían ser las aplicaciones web "bien hechas". Por lo tanto, las características de REST se están usando para guiar la evolución de la Web, y ya han sido implantadas en entes tan importantes como la "blogosfera", eBay, Amazon, Yahoo!, o el mecanismo de enrutamiento de Ruby on Rails que soporta aplicaciones REST utilizando el patrón de diseño MVC.

Las características más importantes de REST son:

1. La clave para crear servicios web en una red REST (como la Web), es identificar todas las entidades conceptuales que se quieran exponer como servicios, es decir, los recursos.

2. Crear un URL para cada recurso. Los recursos deberían ser sustantivos, nunca verbos. Por ejemplo, no se debería usar:

http://www.parts-depot.com/parts/getPart?id=00345

Hay que notar el verbo, getPart. En lugar de eso se debe usar un nombre:

http://www.parts-depot.com/parts/00345

3. Es importante asignar categorías a los recursos de acuerdo a si los clientes quieren recibir una representación del recurso, o si deben poder modificar o añadir un recurso. Para el primer caso, los recursos deben ser accesibles por HTTP GET. Para el segundo caso, los recursos deben ser accesibles por HTTP POST, PUT o DELETE.

4. Las representaciones de los recursos no deberían ser objetos aislados del exterior. Es decir, hay que poner enlaces dentro de los recursos para acceder a otros recursos de forma que los clientes puedan acceder a más información o para obtener información relacionada con el recurso.

5. Todos los recursos accedidos por HTTP GET deberían evitar ser modificados en cualquier caso. Esto es, mediante HTTP GET sólo se debe acceder a una representación del recurso, para modificar un recurso tenemos que usar los otros métodos HTTP anteriormente comentados.

6. Se debe diseñar para revelar datos de forma gradual. No se debería entregar toda la información dentro de un mismo documento de respuesta. Es mejor proveer enlaces dentro del recurso para acceder a más detalles.

7. Se debe especificar el formato de los datos accedidos usando algún esquema. Para aquellos servicios que requieran un POST o un PUT, además hay que proveer un esquema para especificar el formato de la respuesta.

2.3.4 Atom

Atom surgió como un formato alternativo a RSS para la sindicación de contenidos en la web. Ben Trott fue uno de los creadores del formato que finalmente acabó siendo Atom. Él notó la incompatibilidad entre algunas versiones del protocolo RSS, ya que pensaba que los protocolos de publicación basados en XML-RPC no eran lo suficientemente interoperables.

Puesto que el desarrollo de RSS estaba congelado y en ningún caso habría compatibilidad retroactiva, había muchas ventajas en hacer un nuevo diseño desde cero. Se formó entonces el IETF Atom Publishing Format and Protocol Workgroup con los fundadores del nuevo formato. De esta forma, Atom hace referencia a dos estándares relacionados entre sí.

Por un lado, tenemos el Atom Syndication Format [25] (Formato de Sindicación Atom) que se basa en un lenguaje XML diseñado de manera específica para la difusión de feeds en la Web.

Por otro lado, está el Atom Publishing Protocol [26] (Protocolo de Publicación Atom), también llamado AtomPub o APP, un protocolo sencillo basado en HTTP para crear y actualizar recursos web.

2.3.5 JSON

Acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en AJAX. Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos en este contexto es que es mucho más sencillo escribir un analizador semántico de JSON. En JavaScript, un texto JSON se puede analizar fácilmente usando el procedimiento eval(), lo cual ha sido fundamental para que JSON haya sido aceptado por parte de la comunidad de desarrolladores AJAX, debido a la ubicuidad de JavaScript en casi cualquier navegador web.

Si bien es frecuente ver JSON posicionado contra XML, también es frecuente el uso de JSON y XML en la misma aplicación. Por ejemplo, una aplicación de cliente que integra datos de Google Maps con datos meteorológicos en SOAP hacen necesario soportar ambos formatos.

Así, cada vez hay más soporte de JSON mediante el uso de paquetes escritos por terceras partes. La lista de lenguajes soportados incluye ActionScript, C, C#, ColdFusion, Common Lisp, Delphi, E, Eiffel, Java, JavaScript, ML, Objective CAML, Perl, PHP, Python, Rebol, Ruby, y Lua.

2.3.6 Adobe Flash

Adobe Flash [27] es un conjunto de aplicaciones multimedia desarrollado y distribuido por Adobe Systems. Desde su creación en 1996, Flash se ha convertido en un medio muy popular de añadir efectos especiales e interactividad a las web tradicionales, ganando últimamente importancia en el mundo de las RIA donde estos aspectos cobran una importancia elevada.

Figura 16. Logotipo de Adobe Flash

Técnicamente, Adobe Flash es un software (no es el único) de creación de este tipo de aplicaciones, siendo realmente Adobe Flash Player el reproductor propiamente dicho, que consiste en una máquina virtual que permite, entre otras cosas, utilizar gráficos vectoriales, mapas de bits, flujos bidireccionales tanto de video como de audio, siendo posible controlar todo ello mediante un lenguaje denominado ActionScript (que se encuentra actualmente en su versión 3.0).

Historia

Flash nace en 1996 con el nombre de FutureSplash Animator de la mano de FutureWave Software (compañía ahora desaparecida), como un software de dibujo vectorial que permitía realizar animaciones, especialmente indicado para la web. La única manera de realizar ese tipo de animaciones en la web hasta el momento era mediante Java, pero la explosión del modelo del plug-ins de Netscape facilitó la penetración del producto impulsando su uso en la creación de páginas web importantes.

A finales de ese mismo año y como consecuencia de la aceptación conseguida, Macromedia adquiere FutureWave renombrando el producto estrella como Macromedia Flash 1.0, lo que catapultó el avance de la tecnología al añadirse paulatinamente multitud de características nuevas que iban desde la reproducción de contenido multimedia, hasta la introducción del lenguaje ActionScript, pasando por añadir capacidades de comunicación entre clientes, siempre a través de un servidor intermedio por razones de seguridad.

En 1999, Flash Player es incluido de serie con Microsoft Internet Explorer 5 produciéndose un gran salto en cuanto al número de instalaciones del producto, que superó rápidamente los 100 millones.

Unido a esto, al hablar sobre su expansión e incursión en la mayoría de los ordenadores mundiales hasta convertirse en estándar de facto, debemos tener en cuenta el impacto del "factor YouTube", uno de los principales motores de la Web 2.0 debido a su alto contenido social, que surgió a principios de 2005 y que basa la reproducción de sus vídeos en el uso de Adobe Flash Player.

La última versión de Macromedia Flash fue la 8, publicada en 2005. Ésta incluía entre otras cosas, la versión 2.0 del mencionado lenguaje ActionScript, que ofrecía un modelo de programación orientado a objetos.

Dicho año, Macromedia fue adquirido junto con su catálogo de productos por Adobe, que dio un nuevo giro al producto lanzando Adobe Flash Player 9 en 2007, siendo la primera ocasión en la que aparecía un nuevo reproductor no acompañado de la aplicación de creación correspondiente, separando definitivamente la plataforma del reproductor Flash de las posibles aplicaciones de desarrollo de contenido para la misma. Además, daba pleno soporte a ActionScrip 3.0 y una mejor integración con el resto de productos de Adobe.

Recientemente (finales de 2008), salió a la luz Adobe Flash Player 10 donde podemos encontrar efectos 3D, aceleración hardware, soporte de texto avanzado, generación dinámica de sonido, API de carga y descarga de archivos o la inclusión del nuevo códec de audio Speex, que ofrece una alternativa de baja latencia para la codificación de voz.

Desarrollo en Flash

Tradicionalmente, el desarrollo para la plataforma Flash se realizaba mediante la única aplicación existente al efecto, Adobe Flash.

La versión actual, Adobe Flash CS5 Professional, es una aplicación principalmente pensada para diseñadores debido a sus orígenes como programa de dibujo vectorial y a su uso especialmente extendido entre diseñadores web. Así, el interfaz gráfico está basado en metáforas indicadas para este grupo como las líneas de tiempo o los marcos, siendo la parte en la que se le da interactividad mediante ActionScript bastante inconexa y difícil de entender para los programadores tradicionales.

Como consecuencia de esta dificultad surgen alternativas de código abierto como FlashDevelop que, dando un enfoque más del estilo de los entornos de desarrollo tradicionales, permiten a los desarrolladores moverse en un ambiente más familiar y abordar proyectos más complejos. Como respuesta de Adobe a este problema surge Adobe Flex, del cual hablaremos en el siguiente punto.

En cualquier caso, el desarrollo de una aplicación Flash se ve explicado en la siguiente figura:

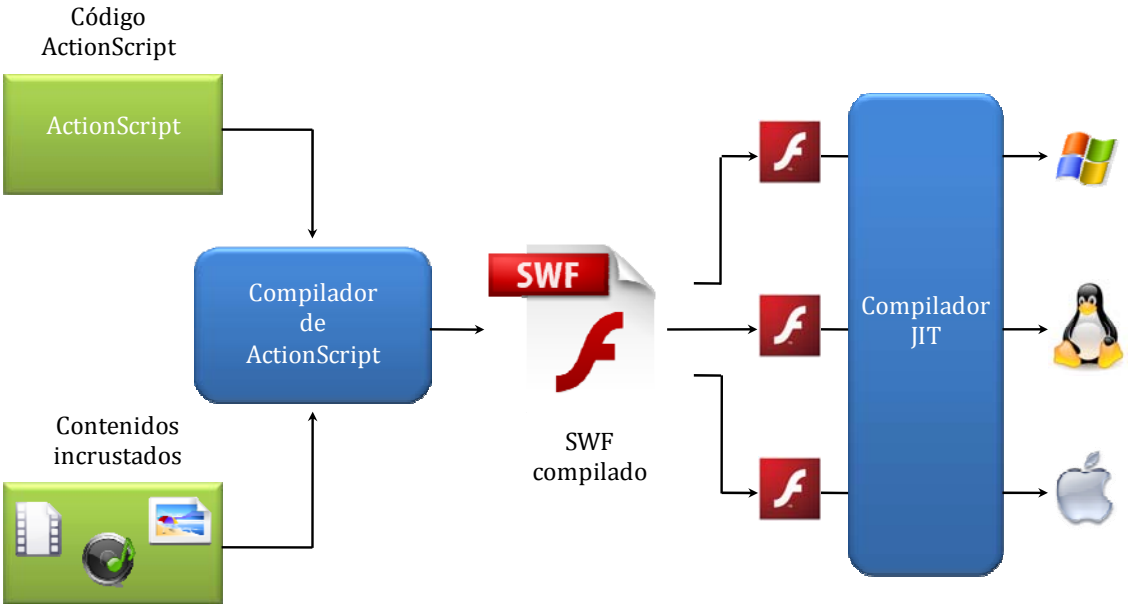

Figura 17. Desarrollo y compilación en Flash

Mediante el uso de uno de los entornos sugeridos, se genera el ActionScript necesario y se seleccionan los contenidos que irán incrustados, éstos, como se ha visto, pueden ser tanto video y audio, como animaciones o imágenes. Con todo esto se alimenta al compilador de ActionScript que generará un archivo en bytecode o código de bits idéntico para todos los sistemas operativos en los que se ejecuta la máquina virtual de Flash. Ya en la máquina final donde se ejecutará la aplicación, éste bytecode es traducido en tiempo de ejecución por un compilador JIT (Just In Time) que forma parte de Flash Player.

Seguridad: sandbox y archivos de políticas

A la hora de desarrollar aplicaciones cliente mediante tecnología Flash, debemos tener en cuenta que éstas se ejecutan localmente bajo la supervisión de un entorno de seguridad comúnmente denominado sandbox.

Éste entorno protegido es necesario principalmente por la naturaleza de las aplicaciones Flash, ya que en la mayoría de los casos, éstas corren en el navegador empotradas sin más consentimiento del usuario que la instalación inicial del plug-in en caso de que fuera necesario. Para proteger la transferencia de información a destinos no autorizados, Flash Player examina todas las peticiones de carga o acceso a recursos externos, además de las interacciones con otros archivos SWF o HTML. Por ello, cada vez que se intenta acceder a uno de estos recursos externos, esto es, no incrustado en el propio SWF, es rechazado o aprobado siguiendo unas normas y basándose en los siguientes factores:

- La operación de ActionScript usada para realizar la petición.

- La clasificación de seguridad del archivo .swf (basado principalmente en saber de dónde se ha obtenido el .swf).
- Los permisos de acceso expresamente especificados por el proveedor del recurso.
- Los permisos de acceso concedidos por el propio usuario (por ejemplo, acceso a la cámara y al micrófono).

En el caso de este trabajo, nos centramos en archivos SWF que se han obtenido de un servidor web, ya que es la manera en la que se distribuirá nuestra aplicación. En estas circunstancias se dice que el tipo de sandbox es remoto (remote security-sandbox-type).

El mecanismo que tiene el distribuidor (en la mayoría de los casos el que administra el servidor), de permitir el acceso a estos contenidos es el uso de archivos de políticas (policy files). Desde la versión 9.0.124.0 de Flash se han introducido ciertos cambios en este sentido. Los más importantes son la necesidad de tener un archivo distinto cuando se quiere dar acceso a sockets y la necesidad de correr un servidor de este tipo de archivos independiente a la aplicación.

En la siguiente tabla, podemos encontrar un resumen de los diferentes criterios de seguridad existentes en Flash:

Operación	Recursos del dominio de origen del SWF	Recursos fuera del dominio de origen del SWF
Carga de contenidos	Permitido	Permitido
Acceso a los datos de los contenidos	Permitido	Permitido con permiso del distribuidor
Carga de datos	Permitido	Permitido con permiso del distribuidor

2.3.7 Adobe Flex

Adobe Flex [28] es un conjunto de tecnologías que permite el desarrollo y despliegue de RIA basadas en la plataforma propietaria de Adobe Flash.

Actualmente, incluye un kit de desarrollo de software (Flex SDK) y un entorno de desarrollo o IDE llamado Flex Builder.

Figura 18. Logotipo de Adobe Flex

Los programadores tradicionales se encontraban con multitud de dificultades al trabajar con las herramientas de desarrollo Flash, al estar muy orientadas a diseñadores gráficos y animadores basando su interfaz en metáforas fácilmente entendibles por éstos como líneas de tiempos, cuadros… pero que no pertenecían al mundo de los desarrolladores de aplicaciones.

El objetivo de Flex es llenar este vacío para permitir a los desarrolladores de aplicaciones web construir rápida y fácilmente RIA basadas en un modelo multi-capa, donde las aplicaciones Flex son el nivel de presentación.

Para ello, se introduce como lenguaje principal de esta tecnología MXML (Multimedia eXtensible Markup Language), un lenguaje de marcado al estilo de XML que permite definir interfaces de usuario bastante similares a los de las aplicaciones de escritorio. La interactividad de estos interfaces viene dada por ActionScript, el lenguaje propio del núcleo de Flash Player que está basado en el estándar ECMAScript.

En el modelo multi-capa que hemos comentado anteriormente, el cliente sólo carga la aplicación una vez, no siendo necesaria la recarga al realizar cambios en ella, mejorando por tanto el flujo de datos frente a aplicaciones web basadas en HTML como PHP, ASP o JSP, donde era necesario recargar la página tras cada cambio.

Historia

La versión inicial de Adobe Flex, surgió como Macromedia Flex 1.0 en Marzo de 2004 (apareciendo poco después la versión 1.5). Comenzó siendo originalmente un producto orientado a empresas, con un servidor de aplicaciones J2EE que compilaba al vuelo obteniendo archivos SWF que eran servidos a los clientes.

Tras la adquisición de Macromedia por parte de Adobe, esta última decidió cambiar significativamente el modelo de licencia de Flex con el lanzamiento de Adobe Flex 2 en 2006. El núcleo de Flex 2 SDK, que contiene tanto el compilador en línea de comandos, como la biblioteca completa de clases y componentes de la interfaz de usuario, se ofreció como código abierto, siendo posible crear únicamente con este SDK aplicaciones Flex completas sin necesidad de pagar ninguna licencia.

Por otro lado, el entorno de desarrollo Flex Builder se rediseño para dar lugar a una nueva versión standalone basada en la plataforma Eclipse, o a la posibilidad de instalarlo como un plug-in de dicha plataforma, permitiendo diseñar interfaces de forma gráfica además de facilitar tareas como la depuración y el despliegue de aplicaciones.

Además, coincidiendo con el lanzamiento de Flex 2, Adobe introdujo la nueva versión de ActionScript bajo el nombre de ActionScript 3. Esta versión requería el uso de Flash Player 9 o superior, ya que su nueva máquina virtual estaba diseñada especialmente para ejecutar dicho código de manera más robusta.

En 2008, Adobe lanzó la versión estable de Flex 3 (que incluye el nuevo SDK y Flex Builder 3). Partiendo de la versión anterior, el cambio más significativo es que añade soporte para Adobe AIR 1.0 (surgido el mimo año), que proporciona una entorno de ejecución multiplataforma para la construcción de RIA en aplicaciones de escritorio, permitiendo por tanto portar aplicaciones Flex al escritorio sin realizar grandes cambios.

Actualmente, Adobe ha lanzado hace escasos meses la versión estable de Flex 4.0 (conocido bajo el nombre de Gumbo), que entre otros aspectos ha desacoplado completamente la creación de la capa de visualización, de la lógica de la aplicación, permitiendo desarrollar toda la capa de interfaz en Photoshop o Fireworks para después pasársela a la lógica de negocio como objetos MXML a los que añadirle la lógica de la aplicación. Adicionalmente, algunas de las funcionalidades que se han incorporado son:

- Cambios en el diseño del framework para que pueda existir colaboración continúa entre varios diseñadores y desarrolladores.
- Gestión de componentes 3D.
- Soporte completo para Adobe Flash Player 10.
- Cambios en el modelo de programación con nuevos namespaces MXML.
- Disponible un nuevo IDE bajo el nombre de Flash Builder 4.

Desarrollo en Flex

El proceso de desarrollo de una aplicación Flex debería seguir en mayor o menor medida los siguientes pasos:

- Definición de una interfaz de aplicación usando los elementos predefinidos incluidos en el SDK, o elementos nuevos creados por el desarrollador para el proyecto concreto que se pretende llevar a cabo.
- Posicionar dichos componentes en el diseño de la interfaz de usuario.
- Usar hojas de estilo (CSS) o temas para definir el diseño visual.

- Añadir el comportamiento dinámico de las diferentes partes de la aplicación mediante ActionScript.
- Definir y conectar con los servicios de datos según sea necesario.
- Compilar el código fuente para obtener un SWF que se ejecutará en Flash Player.

Comparado con el desarrollo de una aplicación Flash tradicional podríamos decir que la única diferencia es la aparición de lenguaje MXML, sin embargo, MXML es íntegramente traducido a ActionScript en tiempo de compilación obteniendo un archivo SWF totalmente compatible con los utilizados tradicionalmente.

De esta manera, se consigue obtener aplicaciones más potentes que serán desarrolladas de forma más acorde con lo habitual dentro del mundo del desarrollo software, consiguiendo una total transparencia respecto a Flash Player, que se ejecuta en la máquina del usuario.

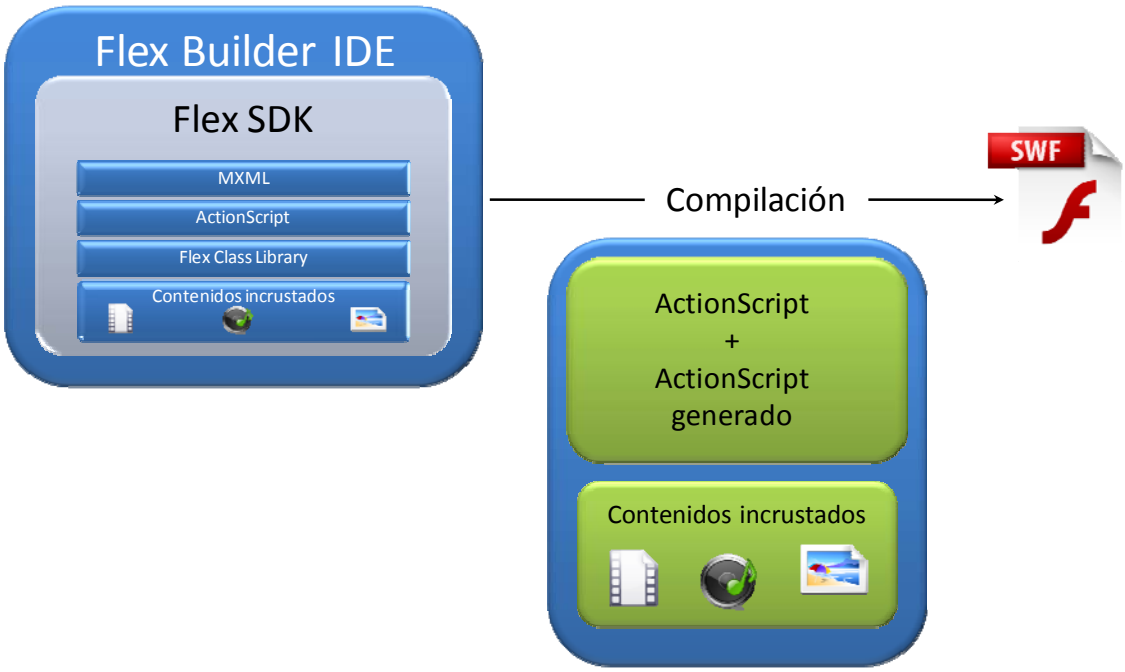

Figura 19. Desarrollo y compilación en Flex

2.3.8 Cairngorm

Cairngorm [29] es un framework estructural para el desarrollo de RIA sobre Adobe Flex desarrollado como código abierto, que fomenta el uso de patrones de diseño y potencia la escalabilidad y el crecimiento de las aplicaciones. Una de las ideas principales que intenta conseguir Cairngorm es la posibilidad de poder construir aplicaciones Flex con arquitectura MVC (Model View Controller). Esto facilita el diseño de aplicaciones con estados complejos y sincronización de datos entre cliente y

servidor, desacoplando las dificultades que pueda añadir la capa de datos frente al diseñador de la vista.

Surge como solución a proyectos donde participan múltiples desarrolladores que necesitan una coordinación correcta para tener una separación clara de las diferentes funcionalidades. Para ello, se hace imprescindible seguir unas pautas y estrategias comunes por parte de todos los involucrados en el proyecto.

Al ser un framework estructural y transversal a cualquier a cualquier aplicativo conseguimos múltiples ventajas:

- Un desarrollador que conozca Cairngorm puede incorporarse a cualquier proyecto basado en Cairngorm minimizando la curva de aprendizaje inicial.
- El know-how adquirido en otros proyectos basados en Cairngorm es aplicable de forma directa a cualquier otro proyecto.
- En un equipo de desarrolladores que conozca la metodología implícita a Cairngorm, aunque no hayan trabajado con anterioridad juntos, los conceptos son comunes a todos los integrantes, agilizando el planteamiento de las soluciones a los diferentes problemas que surjan.
- Reutilización real de código de forma transversal a los proyectos, con mínimos retoques de integración.
- Muy aceptado por la industria, por lo que existe una buena cantidad de documentación, información y ejemplos.
- Cairngorm induce a la creación de bastantes clases, normalmente sencillas y cortas. Esto facilita la modularidad a la hora de distribuir el trabajo y la integración.
- Al tratarse de un framework soportado oficialmente por Adobe su evolución es continua. Además, en caso de requerir alguna funcionalidad no incluida de base, podremos implementarla, ya que disponemos del código fuente y de su documentación asociada.

Historia

La idea de Cairngorm se apoyó por primera vez en el libro Reality J2EE - Architecting for Flash MX, publicado por Macromedia Press en 2002.

Al igual que las tecnologías RIA propias de ActionScript 2.0 y estudiadas en Flash Remoting with Flash MX 2004, las ideas detrás de Cairngorm fueron revisadas en el ActionScript 2.0 Dictionary de Macromedia Press, teniendo también especial importancia el libro Developing Rich Clients with Macromedia Flex, primero en plantear las mejores prácticas para desarrollar RIA con Adobe Flex.

Debido al gran uso que se hizo de este framework en la comunidad de desarrolladores de aplicaciones RIA, Cairngorm pasó a ser considerado un estándar de facto, decidiéndose a posteriori convertirlo en un proyecto de código abierto.

La primera versión de código abierto fue Cairngorm 0.95 para Flex 1.5, anunciada en la conferencia Macromedia MAX2004 de Nueva Orleans en Noviembre de 2004, aunque poco después se actualizó a la 0.99.

Posteriormente, y ya para Flex 2 y Flex 3, han aparecido las versiones 2.0 (Febrero de 2006), 2.1 (Octubre de 2006) y 2.2 (Abril de 2007). En Junio de 2007, se lanzó la última versión hasta el momento, la 2.2.1.

Patrones de Cairngorm

Cairgorm utiliza un conjunto de patrones claramente definido sobre los que se sustenta toda la arquitectura MVC. Algunos de ellos, están basados en el catálogo de patrones de diseño de J2EE, lo cual aún potencia la documentación ya existente y el que muchos desarrolladores ya conozcan dichos patrones.

Value Objects (VO)

En el catálogo de patrones J2EE correspondería al tipo TransferObjects. Suelen ser objetos poco pesados y muy simples que no tienen ningún tipo de lógica de negocio, ya que actúan como simples contenedores de datos estructurales (mapeo orientado a objetos).

Al ser transversales a todas las capas, evitan tener que modificar la aplicación si varía la capa de consumo de datos, por lo que un cambio en cualquiera de las capas no afectará al resto (baja cohesión de código).

Dependiendo de la fuente de datos, estos objetos se construirán de varias maneras, por lo que un cambio en la capa generadora de estos objetos no implica un cambio en las capas consumidoras y viceversa, consiguiendo que dichos objetos actúen de forma ortogonal a la aplicación.

Paralelamente a la capa de consumo de datos, podemos hacer que las propiedades de nuestros Value Objects sean Bindables, consiguiendo que cualquier consumidor/suscriptor de una propiedad de nuestro Value Objetc será notificado si ésta cambia. De esta manera, permite relacionar el modelo de datos (Model) con la capa de presentación (View).

Commands

Surgen para aislar y reutilizar todas las funcionalidades que diseñemos. La idea sobre la que se cimentan es: "mejor tener muchos elementos especializados, que pocos

y no especializados". Por ello se basan en tener muchas clases cortas, una para cada funcionalidad que queramos conseguir. Por definición:

- Son la unidad mínima funcional de nuestra aplicación. Luego, en ellos reside la lógica de la misma.
- Son sin estado, por lo que entre distintas ejecuciones no guardan el estado, haciendo que las distintas capas de la aplicación sean transversales.
- Deberían poderse ejecutar desde cualquier lugar de nuestra aplicación, independientes al entorno y circunstancias de invocación.
- Un command debe encapsular su lógica interna y debe ser intercambiable por otro command sin ninguna repercusión a nivel de aplicación, consiguiendo de esta manera que actúen como cajas negras que aportan funcionalidad al que los ejecuta sin tener en cuenta su implementación.

Un command debe tener un método execute que es el que ejecuta la funcionalidad asociada, pudiendo ser llamado desde la vista en cualquier contexto. Sin embargo, como todos los commands se ejecutan de la misma forma, Cairngorm implementa una solución para la invocación basada en el patrón FrontController.

Por último, es importante saber que se pueden anidar commands para generar "MacroCommands", aunque esta opción no es buena, ya que interfiere en la capacidad de reutilización de los commands involucrados y hay técnicas mucho más adecuadas para anidar commands que comentaremos más adelante.

Front Controller

Está pensado para centralizar las invocaciones de la lógica de la aplicación en un único lugar, de manera que los commands existentes se "registran" en el FrontController mediante el método addCommand con la siguiente dupla como parámetros: <tipo de evento, command a ejecutar>.

Por tanto, al lanzar la aplicación un objeto CairngormEvent pasándole como parámetro uno de los tipos registrados en el FrontController, éste llevará asociado un command que se ejecutará mediante su método execute. Con esto, evitamos tener que llamar directamente a dicho método desde la vista, obteniendo una separación más clara entre vista, modelo y controlador.

La ejecución de estas funciones se mapea contra un objeto, que es almacenable, facilitando por tanto su posterior re-invocación, y por tanto ahorrando memoria.

Por otro lado, es importante tener en cuenta que la aplicación sólo tendrá un único FrontController que implementará un patrón Singleton, siendo por tanto un objeto único que almacenará todo el registro o mapeo de commands a eventos.

Finalmente, en el FrontController también se puede añadir funcionalidad transversal a los commands, como historiales de ejecución, etc.

Events

El concepto de EventDispatcher, listener y evento están tomados del patrón Observer, y tiene una fuerte implicación en la utilización de eventos dentro de Cairngorm.

Como es lógico, los eventos que lanzamos deben ser creados por el programador para adaptarlos o añadir datos o información concreta para una determinada familia de invocaciones o casos de uso, como podría ser todos los eventos relacionados con usuarios.

El evento a lanzar tiene que ser un CairngormEvent o cualquier otro evento que herede directa o indirectamente de él.

Al lanzar dicho evento, ejecutamos el método execute de un determinado command, al cual le llega como parámetro el evento lanzado. Ese evento será el encargado de llevar los datos necesarios (si los hay) para la ejecución del command. En el caso anterior de los usuarios, podría ser el nombre y contraseña del usuario en un proceso de identificación o login.

Aunque a priori podríamos concluir que lo único que hacemos, en el caso de Cairngorm, cuando lanzamos un evento es ejecutar el método execute de un determinado command, al cual le llega como parámetro el evento lanzado, realmente se tienen más implicaciones, y es que la invocación del método execute es reproducible a posteriori simplemente almacenando el evento que ha originado la ejecución.

Adicionalmente, garantiza que los commands sean totalmente ortogonales al resto del aplicativo y que no tengan dependencia con información contextual superflua o fácilmente accesible. Dicho de otra forma, los commands son independientes entre sí y no tienen grandes dependencias con el resto de la aplicación, lo cual permite su reutilización en contextos totalmente diferentes al pensado inicialmente.

Services y ServiceLocator

A la hora de tener comunicación con el servidor, aparece el concepto de Servicio como homólogo, en la parte cliente, de la lógica de negocio del servidor. Por tanto, el

lado cliente no realiza acciones directamente, sino que las delega al lado servidor mediante HTTPService, RemoteObject, WebService, etc.

Estos servicios realizan peticiones al servidor pasándole los parámetros que requiere, recogiendo la respuesta que el servidor genera.

Tienen que ser accesibles desde cualquier parte de la aplicación, además de ser únicos. Luego, si queremos invocar dos veces un servicio con parámetros diferentes y desde puntos distintos de nuestra aplicación no tendríamos que volver a crear el servicio, sino que utilizamos el que ya se ha creado en peticiones anteriores.

Además, para evitar la duplicidad de servicios, introducimos el patrón ServiceLocator que implementa un Singleton garantizando así una existencia única de todos los servicios definidos en él. Esto es, a través del ServiceLocator, podemos acceder a los servicios disponibles (declarados en él) para la aplicación sin tener que declararlos una y otra vez en cada uso.

Business Delegate

Teniendo presente lo que hemos explicado hasta el momento, un command podría utilizar un servicio que le permitiera establecer una conexión con el servidor sabiendo que previamente al uso del servicio, éste debería estar declarado en el ServiceLocator. Una vez hecho esto, se podría invocar la petición desde cualquier command.

Sin embargo, no hemos hablado de lo que se hará una vez se obtenga la respuesta de la invocación del servicio, ya que debe existir un mecanismo que la trate.

Para ello se introduce el concepto de Responder. Es una clase interface que define un método result y otro fault que se ejecutarán en base a que el resultado de la invocación del servicio sea satisfactorio o no.

De esta manera, aparece la idea de Bussiness Delegate, que no es otra cosa que una capa temporal situada entre el command y el servicio a usar, la cual permite gestionar las respuestas del servidor.

Luego, un command creará un Bussines Delegate que será el encargado de realizar la llamada al servicio y posteriormente tratará el resultado para devolver al command la información útil que espera. Por tanto, al tener el control sobre las invocaciones al servidor y sus respuestas, los delegates son el sitio ideal donde hacer el tratamiento de la información recibida para transformarla a Value Objects.

Finalmente, existen dos ventajas claras que vienen implícitas en el uso del patrón Bussines Delegate. Por un lado, conseguimos aislar la aplicación de la tecnología utilizada en el servidor.

Por otro lado, también se pueden utilizar los delegates para simular parte de la aplicación devolviendo Mock Objects (objetos generados de manera planeada para emular una posible respuesta) a los commands que los lanzan.

Model Locator

En el patrón MVC, el Modelo es la representación específica de la información con la cual el sistema opera, siendo la lógica de datos de la aplicación la que debe asegurar la integridad de éstos. Es decir, las variables, conjuntos de datos y constantes que la aplicación maneja se almacenan en este objeto que sigue el patrón de diseño Singleton, siendo accesibles todas ellas desde cualquier parte de la aplicación.

A partir de aquí, usando Bindings en dichos datos definidos en el Modelo, si un command actualiza un determinado conjunto de datos, las vistas que utilizan esos datos (gráficas, tablas, etc.) serán notificadas, actualizando así su representación de forma automática.

Luego, el Model Locator es la entidad encargada de gestionar el modelo consumidor/productor de datos en la aplicación, de forma que los commands serán productores de datos, mientras que las distintas vistas serán los consumidores.

Secuenciado de commands

Como ya se comentó al describir el objetivo de los commands, una de las principales ventajas que puede suscitar su uso es la posibilidad de encadenarlos secuencialmente para conseguir unir diferentes funciones.

Por ello, cuando necesitamos ejecutar una serie de commands de forma asíncrona siguiendo un orden determinado, podemos optar por dos vías claramente diferentes:

- **Secuenciado estático:**

Para una secuencia dada, creamos una serie de commands exclusiva para ésta, de manera que cada command fija el evento que viene después.

El problema es evidente: si queremos reutilizar un command que ya forma parte de dicha secuencia no podremos al estar diseñado para funcionar dentro de esa secuencia, por lo que nos veremos forzados a duplicar código al implementarlo de forma independiente.

Esto ocurre porque la clase SequenceCommand utiliza una propiedad nextEvent que es la que decide cuál es el siguiente evento (y por consiguiente, command) que se va a ejecutar.

La forma en la que los commands son lanzados hace que sea imposible introducir parámetros a esta propiedad y por lo tanto se mantiene estática.

- **Secuenciado dinámico:**

En este caso, la idea es conseguir crear de forma dinámica las secuencias de commands que deseamos ejecutar, de manera que no tengamos que crear commands específicos para cada secuencia, sino generales que puedan ser reutilizados en distintas secuencias o lanzados individualmente.

Para conseguir esto, utilizamos la clase ChainEvent, que es un recubrimiento de CairngormEvent que añade la propiedad nextChainedEvent, y la clase EventChainFactory que se ocupa de enlazar dinámicamente las diferentes instancias de eventos que queremos secuenciar. Para ello, se le pasa un Array de eventos a una instancia de ChainEvent y el dispatcher va asignando de menor a mayor índice cual es su nextEvent.

Con esto conseguimos un gran reutilización de código, pues todos los eventos y commands son independientes unos de otros a la hora de ejecutarse.

2.3.9 LDAP

LDAP (Lightweight Directory Access Protocol o Protocolo Ligero de Acceso a Directorios) es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. LDAP también es considerado una base de datos (aunque su sistema de almacenamiento puede ser diferente), a la que pueden realizarse consultas.

Habitualmente, almacena información de usuario (identificador y contraseña) y es utilizado para autenticarse, aunque es posible almacenar otra información como datos de interés del usuario, permisos, roles, idioma preferido, etc.

2.3.10 CAS

Es uno de los protocolos de SSO más utilizados en entornos web y por ello se ha convertido en una de las opciones más elegidas a la hora de conseguir este tipo de procesos de autenticación en los últimos años. Fue desarrollado por la universidad de Yale, aunque desde Diciembre de 2004 pertenece al proyecto Jasig [30].

En este protocolo intervienen al menos tres módulos: un cliente web, la aplicación o servicio web que requiere la autenticación del primero y el servidor de CAS. Sin

embargo, normalmente encontraremos un cuarto elemento que corresponderá a una base de datos donde se almacenarán las credenciales de los usuarios, que sólo será accesible por el servidor de CAS. Usualmente esta función la cumplirá un módulo LDAP.

De esta forma, y aunque el protocolo CAS tiene una versión estándar muy utilizada en entornos en red, vamos a centrarnos en una modificación reciente del mismo basada en el uso de un API RESTful que está descrito en [31], más adecuado para aportar SSO en aplicaciones que utilizan un cliente web RIA (como es el caso de los dos proyectos que se presentan en este Trabajo Fin de Máster).

Por tanto, los pasos que se siguen para completar el proceso de SSO son los siguientes:

1. La comunicación comienza cuando el usuario introduce en la pantalla inicial de log in del cliente web sus credenciales basadas en usuario y contraseña. Con estos datos se realiza una llamada HTTP de tipo POST a la URL https://CASserver.dit.upm.es/cas/v1/tickets introduciendo como parámetros en el cuerpo del mensaje un hash con claves username y password, y valores los proporcionados por el usuario.
2. A continuación, se produce una comunicación entre el servidor de CAS y el de LDAP, para comprobar la existencia del nombre de usuario y su contraseña en la base de datos LDAP.
3. El resultado de la confirmación es comunicado por parte del servidor de LDAP al servidor de CAS, (supondremos en este caso que la confirmación es satisfactoria).
4. Este mensaje lleva consigo un Ticket Granting Ticket (TGT) en una URL del tipo https://CASserver.dit.upm.es/cas/v1/TGT-21-3oIPvAQWl0fh37tRdec24CbykhelL6XhqmwDOkmfl5MPoSQB6M-CASserver.dit.upm.es, que es recibido por el cliente. Dicho TGT, creado de manera aleatoria, corresponde a una URI única que identifica unívocamente al cliente web frente al servidor CAS, creándose por tanto una sesión entre ellos.
5. El cliente usa el TGT anterior para realizar una llamada de tipo POST a la URL https://CASserver.dit.upm.es/cas/v1/tickets/TGT-21-3oIPvAQWl0fh37tRdec24CbykhelL6XhqmwDOkmfl5MPoSQB6M-CASserver.dit.upm.es, añadiendo como parámetro en el cuerpo del mensaje una hash con clave service y valor http://WebService.dit.upm.es/session. Por tanto, se está pidiendo al servidor de CAS que genere un Service Ticket (ST) que sirva para autenticar al servicio anterior en el servidor CAS.

6. El sexto mensaje corresponde a un 200 OK acompañado del Service Ticket (ST-18-yQKfbx95bv5CqyoChfwj-CASserver.dit.upm.es) correspondiente al servicio que queremos autenticar. Si los parámetros del POST anterior fueran incorrectos o incompletos, el servidor CAS enviaría un mensaje 400 Bad Request. Mientras que si se enviase algún tipo que no correspondiese a los que espera, mandaría un 415 Unsupported Media Type.
7. El cliente recoge el ST y lo utiliza para realizar una llamada GET a la URL http://WebService.dit.upm.es/session?ticket=ST-18-yQKfbx95bv5CqyoChfwj-CASserver.dit.upm.es. Con esto conseguimos que el servicio obtenga el ST que debe utilizar para autenticarse frente al servidor CAS.
8. El servicio envía el ST anterior al servidor de CAS para iniciar un proceso de validación.
9. Y por último, el servidor de CAS comunica que el usuario representado por el ST anterior es válido, permitiendo afirmar que se ha establecido una sesión CAS SSO.
10. Finalmente, se genera una cookie de sesión del servicio web que es entregada al cliente que se ejecuta en el navegador del usuario, permitiendo a partir de ese momento que use los recursos existentes en la aplicación o servicio web.

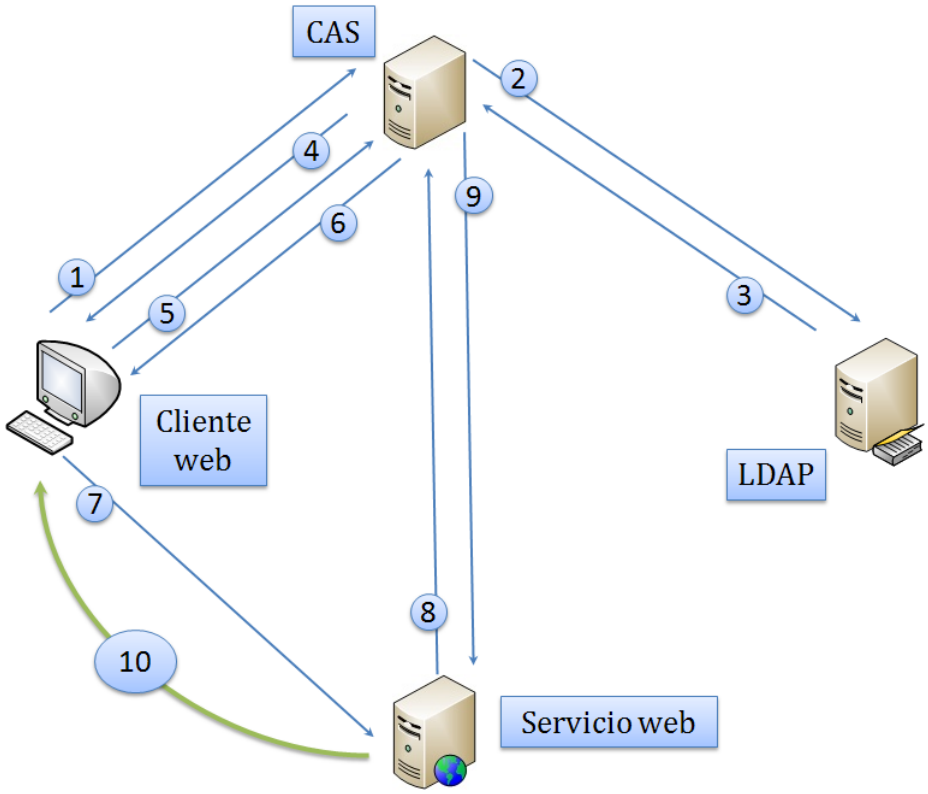

Figura 20. Intercambio de mensajes en el protocolo CAS RESTful

Nuevamente, si el usuario desease conectarse a un segundo servicio o aplicación web, puesto que ya posee el TGT que indica que está autenticado frente al servidor

CAS, y que por tanto tiene una sesión activa, simplemente sería necesario realizar los pasos 5 a 9 para obtener un nuevo ST que fuera válido para el nuevo servicio. Como es normal, esto igualmente se realizaría de manera transparente al usuario que no debería volver a introducir sus credenciales, cumpliéndose por tanto el proceso SSO que estábamos buscando.

2.3.11 Subversion

Subversion es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS (Concurrent Versions System). Es software libre bajo licencia de tipo Apache/BSD y se le conoce también como svn por ser ese el nombre de la herramienta en línea de comandos.

Una característica importante de Subversion es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto instante de tiempo.

Figura 21. Logotipo de Subversion

Así, Subversión es muy conocido en la comunidad de software libre, y se utiliza en muchos proyectos importantes, incluyendo la fundación del software de Apache, KDE, GNOME, Python o Ruby.

Además, en este proyecto, haremos uso concretamente de Subclipse, plugin bajo licencia EPL (Eclipse Public License 1.0) que integra Subversion en el entorno de desarrollo Eclipse, y que por tanto es instalable en el IDE Adobe Flex Builder al estar basado en dicha plataforma.

3 Líneas de investigación y trabajos realizados

Llegados a este punto, y una vez conocidas las tecnologías, algoritmos y protocolos involucrados en los desarrollos que el alumno ha llevado a cabo para este Trabajo Fin de Máster, es el momento de describir con detalle las diferentes líneas de investigación y resultados que se han obtenidos tras dichos trabajos.

En primer lugar hablaremos de la continuación que al alumno realizó de su Proyecto Fin de Carrera "Arquitectura e Implementación de Interfaz Flexible de Usuario para la Integración de Aplicaciones y Servicios Web" [33] conocido también como Itecsoft, y que le sirvió como base o plataforma de pruebas para demostrar el uso de la arquitectura de videoconferencia cloud llamada Nuve en la que participó tanto en su definición como en su desarrollo. Adicionalmente, y sobre la misma plataforma, analizaremos los resultados que se consiguieron tras aplicar una prueba de concepto resultado de una investigación relacionada con la generación de contexto colaborativo a partir de los datos sociales en los que se basa Itecsoft.

En segundo lugar, presentaremos un proyecto adicional en el que el alumno ha trabajado de manera paralela conocido bajo el nombre de Galactus, y en el que se consiguió construir un sistema de recomendación basado en un motor Mahout y un cliente RIA desarrollado en Flex.

3.1 Itecsoft

La plataforma colaborativa Itecsoft desarrollada por el alumno en su PFC se ideó con el objetivo de ofrecer un entorno de colaboración unificado para entornos corporativos, donde los usuarios pudiesen hacer uso de un conjunto de herramientas accesible mediante un interfaz único para trabajar de manera distribuida.

Luego, antes de adentrarnos específicamente lo desarrollado para este TFM, describiremos brevemente la arquitectura e implementación de esta plataforma, de manera que entendamos sobre que plataforma de desarrollaron los nuevos módulos que si son temas centrales de este documento.

Por tanto, y antes de nada, vamos a describir los requisitos que se marcaron para llevar a cabo la plataforma colaborativa Itecsoft:

- **Interacción**: en un entorno CSCW pueden existir dos tipos de colaboración: síncrona (audio conferencia, videoconferencia, mensajería instantánea o compartición de escritorio) y asíncrona (foros o correo electrónico). Como

hemos dicho antes, nuestra intención es seguir los movimientos sociales propios de la Web, por lo que ambas interacciones deberían estar disponibles para los usuarios de nuestro sistema.

- **Coordinación**: esta propiedad está basada en el tamaño de los grupos de usuarios y en cómo se comunican e interactúan entre ellos. Dependerá esencialmente del contexto en el que se muevan (conferencias, foros de debate, sesiones de brainstorming, etc.). Por lo tanto, es necesario establecer diferentes roles de usuario como administrador, revisor o invitado.
- **Distribución**: en base a las ideas planteadas por la Web Social, tener la posibilidad de trabajar y colaborar en un entorno distribuido es esencial para poder hacerlo desde cualquier parte del mundo. Por ello, hemos elegido la arquitectura Cliente-Servidor, siendo por tanto los usuarios capaces de conectarse al sistema desde cualquier terminal con Internet a través del cliente web.
- **Adaptable a diferentes tipos de usuarios**: el sistema debe ser consciente en todo momento del tipo de usuario que está conectado. Esto es, debería comportarse de una manera distinta en base al tipo de rol que posee dicho usuario, mostrando un interfaz adaptado a sus permisos.
- **Visualización**: nuevamente, de acuerdo al paradigma de experiencia de usuario que la Web 2.0 ha establecido, nosotros nos propusimos construir un sistema que siguiese las tendencias de las RIAs [1], donde la experiencia de usuario es muy superior a las aplicaciones tradicionales, hasta el punto de compararse con aplicaciones de escritorio por su alto nivel de usabilidad.
- **Datos públicos y privados**: todas las aplicaciones sociales (Facebook es un buen ejemplo), separan los datos públicos de los privados. En consecuencia, diseñamos la estructura de colaboración de Itecsoft en torno a espacios de trabajos o grupos de colaboración donde los usuarios con intereses similares puedan acceder y compartir datos propios que sólo serán accesibles para los miembros de ese espacio.

Adicionalmente, y ya como parte de los trabajos de fin de máster, el alumno decidió agregar una nueva propiedad muy importante enfocada a resolver los problemas de contexto colaborativo que se mencionarán en detalle en secciones posteriores donde se explicará cómo se llevó a cabo la implementación de esta nueva funcionalidad:

- **Consciencia social (awareness)**: tener la posibilidad de generar contexto social a partir de las estructuras sociales y temporales inherentes a la aplicación, podría ayudar a los usuarios a entender las actividades y proyectos en los que colaboran, así como ser conscientes del entorno social que los rodea y de la posición que ocupan en él. Por tanto, para conseguir esto analizaremos los

datos sociales que Itecsoft almacenará, generando contexto colaborativo a partir de ellos.

De esta manera y como acabamos de argumentar, la arquitectura elegida fue Cliente-Servidor, por lo que a continuación vamos a describir primero el lado servidor, hablando de manera diferenciada de cada uno de los módulos que lo conforman, para después hacerlo con el cliente explicando cómo realiza el recubrimiento de las estructuras de datos provistas por el servidor. Asimismo, se presenta la siguiente figura que resume de manera general la arquitectura del sistema al completo:

Figura 22. Arquitectura general de Itecsoft en la que el cliente web se comunica con los módulos del lado servidor mediante APIs REST

3.1.1 Servidor

SRI: Social Resource Infrastructure

Es la infraestructura responsable de proporcionar los datos sociales y las estructuras colaborativas necesarias para construir el sistema, actuando como pasarela entre la base de datos y el cliente web. El SRI es una aplicación web implementada en Ruby On Rails con arquitectura Modelo-Vista-Controlador (MVC). Además, posee un API REST (como se pueden ver en la figura) para soportar la comunicación entre servidor y cliente a través de llamadas HTTP con las que se podrán gestionar los recursos sociales almacenados en la base de datos. Los recursos REST disponibles son:

- **Usuarios**: todos los usuarios de la aplicación.

- **Espacios**: los espacios de trabajo que identifican a un proyecto o determinado grupo de usuarios.
- **Eventos**: usados para gestionar los eventos de la agenda y el calendario.
- **Artículos**: los mensajes y comentarios usados en la herramienta de foro.
- **Permisos**: cada usuario tendrá un fichero de permisos que especificará los diferentes roles que pueda tener en los espacios a los que pertenezca.

Adicionalmente, debemos destacar que el formato elegido para llevar a cabo la comunicación entre el SRI y el cliente web fue Atom. Esta decisión estuvo basada en el uso generalizado que este formato ha adquirido en los últimos años en Internet. Concretamente, el SRI usa namespaces Atom estandarizados (algunos de ellos usados por ejemplo por Google en Blogger), lo que permite a Itecsoft conectarse con otras aplicaciones sociales sin coste alguno. Asimismo, es importante poner énfasis en el hecho de que para realizar todas las operaciones CRUD (Create Read Update Destroy) en Atom son necesarios dos protocolos definidos en sendas RFCs: The Atom Syndication Format [25] para leer los recursos, y The Atom Publishing Protocol [26] para crear, editar y borrarlos.

Nuve

Como podemos ver en la figura, otro de los módulos pertenecientes al lado servidor es Nuve, que ofrece salas de videoconferencia como servicio mediante una arquitectura cloud. Sin embargo, este servicio no es tan simple como pueda parecer a priori, ya que permite a los usuarios acceder a un entorno colaborativo completo que incluye compartición de audio y vídeo, mensajería instantánea y compartición de escritorio. Sin embargo dejaremos la explicación de este componente para un sección posterior por ser parte del la labor realizada por el alumno durante este TFM.

Autenticación Single-Sign-On

El último módulo que nos queda por explicar es el encargado de la autenticación en Itecsoft. Dicho módulo está dividido en dos componentes. Por un lado tenemos un servidor de CAS que nos ofrece un servicio de Single-Sing-On válido para autenticar al cliente web en el SRI (y posibles módulos futuros). Así, el cliente se comunica con este componente a través de peticiones HTTP usando para ello el API RESTful [31] que posee esta implementación de CAS y que ya se comentó en la sección 2.3 de "Tecnologías utilizadas para el desarrollo". Por otro lado, tenemos una base de datos OpenLDAP [32] que almacena a los usuarios y sus performances. Por lo tanto, el servidor de CAS autenticará contra la base de datos LDAP las credenciales (usuario y contraseña) enviadas desde el cliente web, generando posteriormente los tickets correspondientes para iniciar la sesión de usuario tal y como se explicó en la sección anteriormente mencionada.

3.1.2 Cliente

El cliente web de Itecsoft ha sido desarrollado siguiendo el paradigma de las aplicaciones RIA. Para ello se ha usado Adobe Flex como tecnología de implementación, ya que facilita el diseño de este tipo de aplicaciones. Esto además tiene una ventaja añadida: los usuarios no necesitarán instalar nada en sus ordenadores debido a que la mayor parte de los navegadores tienen actualmente instalado el plug-in de Flash Player. Volviendo a la arquitectura del cliente, hay que mencionar que fue diseñada utilizando el framework de código abierto Cairngorm [29], ya que proporciona una arquitectura MVC a las aplicaciones Flex que hacen uso de él, facilitando de esta manera su implementación.

Por consiguiente, el cliente web se construye como recubrimiento de las estructuras de datos sociales existentes en el SRI y actúa como capa de presentación avanzada para permitir a los usuarios colaborar entre ellos utilizando el conjunto de herramientas que se obtienen de este recubrimiento y que están disponibles para ellos en los diferentes espacios de trabajo en los que estén registrados (como se ilustra en la figura). Luego, cada espacio tendrá los siguientes componentes que poseen integración horizontal entre ellos, lo que les permite actuar conjuntamente de manera simultánea:

- **Tablón**: es el sitio central de un espacio en el que se informa cuál es el objetivo del mismo, además de conocer en todo momento cuáles van a ser los próximos eventos o los mensajes con más actividad del foro.
- **Sala de Nuve**: proporciona una sala de videoconferencia con mensajería instantánea y compartición de escritorio.
- **Foro**: recubriendo los "artículos" del SRI, construimos un foro donde los usuarios del espacio pueden abrir hilos de conversación donde poder colaborar de manera asíncrona.
- **Agenda y Calendario**: de nuevo, usando la estructura de datos "eventos" del SRI, proporcionamos una herramienta de agenda y calendario donde poder fijar reuniones o eventos propios de la actividad del grupo de trabajo al que representa el espacio.
- **Presencia y Perfiles**: proporciona por un lado los perfiles de usuario públicos de las personas registradas en el espacio, así como un servicio de presencia que nos informa en todo momento de quién está conectado.
- **Contexto colaborativo**: este componente es el encargado de generar contexto colaborativo a partir de un grupo de usuarios en un espacio, de manera que ellos puedan ser conscientes de las interacciones y lazos sociales que mantienen dentro de la aplicación.

Una vez llegados a este punto, ya podemos entender completamente la figura que representaba la arquitectura general de Itecsoft. Además, teniendo en cuenta que el cliente es el encargado de integrar de manera unificada todos los módulos del lado servidor, podemos afirmar sin temor a equivocarnos que la orquestación entre todos los servicios existentes se consiguió lograr de manera satisfactoria, cumpliendo las propiedades que se marcaron como requisitos de diseño inicialmente.

3.2 Videoconferencia como Servicio: Nuve

Nuve es la evolución de Marte 3.0 [44], un servicio de videoconferencia Cliente-Servidor desarrollado con tecnología Java en el servidor de gestión de flujos de vídeo y audio conocido como Red5 [45] y tecnología Flex para la implementación del cliente web. La arquitectura de Nuve extiende la implementación anterior para convertirlo en un servicio escalable mediante técnicas de cloud computing, ofreciendo de esta manera salas virtuales a los usuarios de servicios de terceros, pudiendo por tanto integrar estas salas en diferentes aplicaciones como las representadas en la figura, o como es en este caso, en Itecsoft.

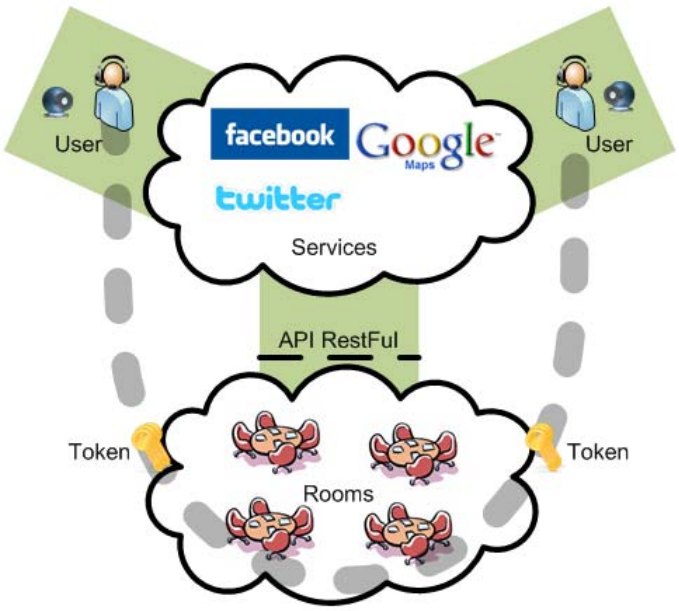

Figura 23. Arquitectura general de uso de Nuve

Para hacer esto Nuve posee, un API REST para permitir que la aplicación de terceros solicite y gestione dichas salas. En otras palabras, este interfaz pretende transformar un sistema tradicional de telecomunicaciones, como es la videoconferencia, en un recurso que puede ser usado por aplicaciones de terceros con una filosofía y arquitectura de cloud computing.

Los detalles del trabajo realizado por el alumno para la consecución de Nuve no se describen en esta sección ya que al final de este documento se presenta un apéndice

donde se puede encontrar el artículo [46] completo que fue publicado en 2009 en el congreso internacional CollaborateCom celebrado en Washington DC como resultado de la investigación llevada a cabo en este campo.

De todos modos, y en cuanto a su integración en la arquitectura de Itecsoft, se debe tener en cuenta cual es la conexión entre el SRI y Nuve, pues cuando creamos un nuevo espacio en Itecsoft, éste lleva asociado una sala de Nuve donde los usuarios del espacio son registrados. En consecuencia, tenemos una relación uno a uno entre espacios del SRI y salas de Nuve, y es el cliente web Flex el encargado de realizar las diferentes llamadas al API REST de Nuve para la gestión de las diferentes salas existentes en cada espacio, así como de cargar el módulo SWF que representa a la sala.

3.3 Generación de Contexto Colaborativo a partir de Itecsoft

En esta sección, describiremos la última herramienta existente en los espacios de trabajo de Itecsoft y que se puede ver en la figura de arquitectura general anterior con el nombre de "Contexto Colaborativo". Esta herramienta utiliza procesos y técnicas de generación de contexto colaborativo (concepto conocido como *awareness* en inglés) que detallaremos a continuación. Sin embargo, y como paso previo a entrar en el trabajo realizado, repasaremos las principales ideas que se tuvieron en cuenta para seguir esta línea de investigación.

3.3.1 Motivación

En los últimos años, el nombre "Web 2.0" ha sido el estandarte elegido para describir a todo un movimiento sustentando principalmente por todas las aplicaciones web sociales que han surgido en los últimos años, y que han conseguido revolucionar los cimientos y la forma de entender y usar Internet. Éste área, (con aplicaciones como Blogger, MySpace, Twitter, Facebook, Google Wave o Flickr), se ha convertido en una de las más prolíficas sobre todo porque las nuevas tecnologías son cada vez más importantes en ella, no sólo para ofrecer nuevas funcionalidades, sino para conseguir socializar la Web de una manera difícilmente imaginable hace años.

La idea principal subyacente a este tipo de aplicaciones es ofrecer un interfaz de usuario amigable, intuitivo y altamente usable, acompañado de una capacidad enorme para compartir información fácilmente entre sus usuarios. Dicho de otro modo, los usuarios pueden conectarse con el mundo entero de la manera que ellos prefieran simplemente eligiendo la o las aplicaciones sociales que mejor se adapten a su estilo de vida. Por lo tanto, este increíble conocimiento generado por todos estos usuarios se comparte en Internet y es accesible en la mayoría de los casos para todo el mundo, siendo por tanto una fuente de conocimiento inagotable.

Por otro lado, el mundo empresarial no ha asistido impertérrito a esa evolución, sino que ha entendido las ventajas de esta nueva forma social de colaboración que produce como hemos visto una cantidad ingente de información de todo tipo de temáticas. Es más, teniendo en cuenta que uno de los problemas principales que se les presenta es la pérdida de "know how" cuando alguno de sus empleados deja la compañía, conocer dónde ha podido permanecer o mejor dicho, en quién ha podido permanecer ese conocimiento es un objetivo sumamente importante. Por ello, sacar a la luz la actividad colectiva y el contexto colaborativo que se crea en una empresa es un tema esencial que se intenta conseguir desde hace años. Es por ello que las herramientas CSCW (Computer Supported Cooperative Work) han sido tan importantes para el mundo empresarial desde su aparición en la década de los 80 [34].

Sin embargo, dichas herramientas han tenido siempre unas estructuras de datos demasiado rígidas completamente alejadas de las teorías aplicadas en la Web 2.0. Como consecuencia, el problema de salvaguardar el conocimiento en una empresa u organización sigue sin resolverse completamente hoy en día.

Con este trabajo, se propone un nuevo punto de vista de las herramientas CSCW más acorde con las ideas de la Web Social para ofrecer los aspectos típicos de estas herramientas (colaboración entre usuarios), y en segundo lugar, la posibilidad de generar contexto colaborativo que nos muestre las uniones existentes entre los usuarios del sistema y una información más detallada de su entorno de trabajo desde un punto de vista más social. Este nuevo CSCW 2.0 que hemos desarrollado recibe el nombre de Itecsoft, y ya ha sido explicado en secciones anteriores, así que es por tanto el momento de adentrarnos en la generación del contexto colaborativo a partir de sus datos.

3.3.2 ¿Qué entendemos por Contexto Colaborativo?

Para responder a esta pregunta, es necesario introducir los conceptos que están detrás de él y que nos permiten su generación. En nuestro caso, seguiremos las ideas descritas por Fisher y Dourish en 2004 [35], que desarrollamos a continuación.

Redes Sociales

Son probablemente la pieza fundamental de la generación de contexto colaborativo. El análisis de redes sociales ha sido estudiado ampliamente desde hace mucho tiempo especialmente en las áreas de ciencias sociales [36], aunque más recientemente en las redes y movimientos sociales que se han formado en Internet. Describen las relaciones existentes entre conjuntos de personas mediante el análisis de los aspectos sociales que comparten y los lazos personales o colaborativos que los unen. Así, usando el análisis de redes sociales podremos encontrar las estructuras sociales y grupos de trabajo que aparecen cuando existe colaboración entre personas, entendiendo de esta manera los diferentes roles que interpretan cada una según su contexto actual.

Estructuras Temporales

Son complementarias a las redes sociales, pues describen cómo éstas evolucionan a lo largo del tiempo. Por tanto, nos muestran los ritmos de colaboración que una persona experimenta en los diferentes grupos o proyectos en los que trabaja o colabora, permitiéndonos comprender en mayor detalle cuál es su relación con el resto de usuarios en cada momento dentro de un grupo.

Luego, como consecuencia de la unión de estos dos conceptos o áreas de estudio y por supuesto, de su análisis, vamos a poder generar el contexto colaborativo que estábamos buscando y que nos mostrará cómo se comporta un grupo de usuarios que colabora y cuáles son sus relaciones durante todo el tiempo que permanecen en contacto, mostrando de esta manera los lazos que un usuario concreto ha establecido dentro de un grupo de colaboración, y pudiendo de esta manera relacionar todos estos datos para elaborar un estudio completo del entorno social que rodea a una persona.

3.3.3 Generación de Contexto Colaborativo en Itecsoft

Llegados a este punto, la pregunta es obvia: ¿cómo generamos contexto colaborativo en Itecsoft?

Como ya hemos visto, tenemos un conjunto muy rico en lo que a datos sociales se refiere, ya que hemos inspirado nuestras estructuras en las que se utilizan actualmente en las aplicaciones sociales propias del movimiento de la Web 2.0. Ahora bien, lo interesante es usar estos datos como entrada para nuestro sistema, de manera que podamos en primer lugar generar patrones colaborativos para después analizarlos y obtener el contexto colaborativo deseado. Por tanto, para generar esta información vamos a plantear las diferentes opciones complementarias que se nos presentan a partir de los recursos disponibles en el SRI: usuarios, espacios, eventos y artículos.

Generación basada en espacios y usuarios

Un espacio de trabajo, tal y como lo hemos definido anteriormente, es en sí mismo una estructura colaborativa ya que es un lugar donde personas que trabajan en el mismo proyecto o tarea, están registradas para colaborar unas con otras. Por lo tanto, la primera cosa que un usuario descubre al ingresar en un espacio es qué personas le acompañan y por tanto están registradas como él en dicho espacio.

Consecuentemente, la generación de patrones colaborativos basados en los recursos del SRI "espacios" y "usuarios" es relativamente sencilla, pues hay una relación directa entre ellos.

La siguiente figura muestra un ejemplo sencillo de cómo realizamos dicha generación de patrones, y por tanto, de cómo logramos obtener el contexto colaborativo. Como se puede ver, el usuario A pertenece a cuatros espacios en los que colabora con diferentes usuarios. De acuerdo al patrón de Red Completa, podemos comprobar rápidamente con quién colabora simplemente analizando qué usuarios comparten espacio de trabajo con él.

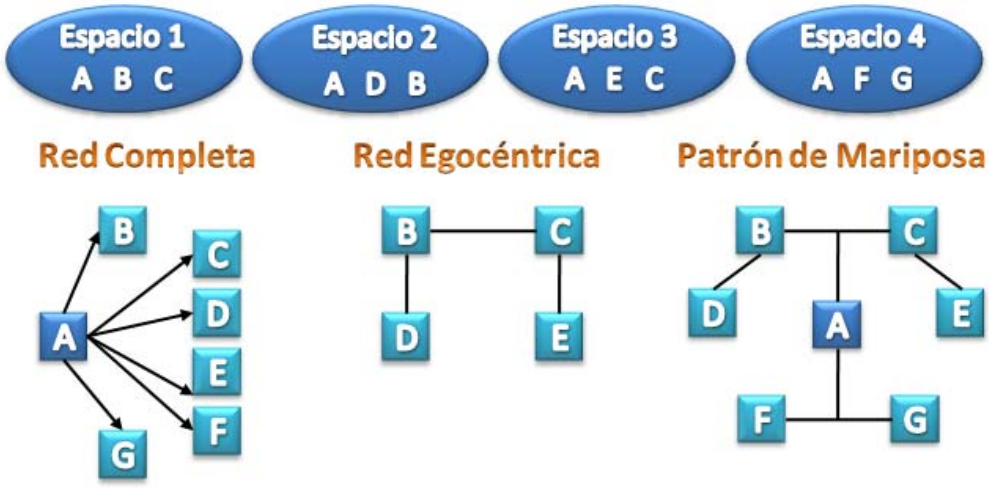

Figura 24. Generación de patrones colaborativos a partir de espacios y usuarios

Si por el contrario nos fijamos en el patrón de Red Egocéntrica [37], obtenemos información de contexto colaborativo relacionando a los usuarios anteriores entre sí. Esto lo logramos analizando los lazos que los unen a todos con A. Así, podemos ver cómo los equipos formados por (D, B) y (E, C) que a priori parecen disjuntos, están de hecho unidos por la colaboración que A mantiene con B y C en el espacio 1.

Finalmente, el patrón de Mariposa (también conocido como el de Roles Duales), nos informa de que A es el nexo de unión entre dos grupos de trabajo totalmente disjuntos, lo que nos permite identificar con este tipo de patrones los roles de A dentro de la empresa (por ejemplo, un directivo que lidera varios proyectos), consiguiendo situarlo en su contexto colaborativo de una manera mucho más certera.

Generación basada en artículos y usuarios

Ahora nos centraremos en el recurso "artículos" del SRI que utilizamos para generar el foro asociado a cada uno de los espacios. Luego, mediante el análisis de estos hilos de conversación que incluirán mensajes iníciales y comentarios realizados como respuesta, podemos extraer información social. Es más, podemos conectar a usuarios entre sí en base a la estructura padre-hijo (mensaje-comentario) que poseerá este tipo de colaboración asíncrona.

Y lo que es más importante si cabe, podemos analizar temporalmente estas estructuras de redes sociales que hemos generado con el estudio anterior utilizando las fechas que van asociadas a la dupla (usuario, mensaje), dándole por tanto un peso a esta información que puede depender del nivel de profundidad temporal que estemos dispuestos a analizar.

En la siguiente sección presentaremos un caso de uso donde veremos en mayor detalle una aplicación de este tipo de técnicas, así como los resultados obtenidos de la ejecución de estas herramientas de generación de contexto en Itecsoft.

Generación basada en eventos y usuarios

Otra manera de generar patrones de colaboración temporales y por lo tanto, obtener un contexto colaborativo más detallado, consiste en el estudio de los recursos "eventos" y "usuarios" de manera conjunta. Específicamente, si prestamos atención a los eventos que se utilizan en el componente de Agenda y Calendario que relacionan usuarios, tipos de reunión y temáticas dentro de un espacio, la información que nos brindan es tremendamente rica y obtener patrones similares a los de los casos anteriores no sería complicado.

Macro patrones

Por último en esta sección dónde analizamos las diferentes vías de generar contexto colaborativo en Itecsoft, podemos dar un paso más allá basándonos en el conjunto de patrones colaborativos que hemos obtenido de los recursos sociales anteriores. Esto es, tenemos la posibilidad de analizar a un nivel superior los datos sociales de los que disponemos realizando análisis conjunto de los resultados previos. Si tenemos en cuenta que normalmente los patrones colaborativos no suelen aparecer de manera aislada, y lo que es todavía más común, que suelen aparecer solapados, podemos combinarlos para obtener Macro patrones. Este nuevo tipo de patrones nos dará una información mucho más veraz y completa sobre el contexto colaborativo que rodea al grupo de usuarios que estamos analizando, en comparación con los patrones generados tras un primer análisis.

Por ejemplo, el patrón de Mariposa ilustrado por la figura anterior tiene implícitamente en su interior el patrón de Red Egocéntrica, por lo que se trata de un macro patrón que nos da una información más detallada sobre el usuario A y su entorno.

De esta manera, cuando deseemos generar contexto colaborativo, podemos analizar los datos sociales de los que disponemos a varios niveles de complejidad, dependiendo del nivel de detalle e interrelación entre usuarios que estemos buscando.

3.3.4 Caso de uso de Generación de Contexto

Finalmente, y para demostrar que la investigación realizada en este campo ha tenido el éxito que buscábamos, se implementaron los métodos de generación de contexto colaborativo en el cliente web de Itecsoft basados en el análisis de patrones previamente descritos. Así, conseguimos dar una solución al siguiente caso de uso que obedece a un tipo de problema muy común en la colaboración diaria a nivel corporativo como es el siguiente:

"Bruce es un diseñador software que acaba de ser trasladado a un nuevo departamento en su compañía en el que se trabaja en un proyecto relacionado con el área de cloud computing. Bruce no conoce demasiado la materia por lo que necesita hablar con sus nuevos compañeros para ponerse al día. Sin embargo, desconoce quiénes son los expertos en su nuevo grupo ya que está formado por personas de varias sedes repartidas por diferentes países. ¿Qué puede hacer?"

Lo primero que hicimos fue suponer que la empresa u organización donde Bruce trabaja ha desplegado Itecsoft hace un tiempo para proporcionar una plataforma de colaboración completa que puedan utilizar todos sus trabajadores, indiferentemente desde dónde lo hagan físicamente. En ese caso, "*Bruce usa el Motor de Búsqueda de Expertos existente en la aplicación para encontrar a expertos en temas de cloud computing en su nuevo grupo de trabajo que posee un espacio llamado DIT en la aplicación*".

Figura 25. Captura de pantalla de Itecsoft en la que se muestra la herramienta de búsqueda de expertos que hace uso del generador de contexto colaborativo

La figura ilustra un patrón social de tipo Top 5 en el que informa de los expertos en cloud computing en dicho espacio durante los últimos cuatro meses. Nuestro módulo de generación de contexto colaborativo internamente ha analizado la evolución temporal de los usuarios que han escrito mensajes en el foro o han organizado eventos sobre este tema. Tras eso, los ha ordenado por su importancia, teniendo en cuenta los diferentes lazos sociales que puedan existir entre ellos.

Como resultado, la herramienta destaca al experto más importante en base a su mayor número de mensajes y eventos (es decir, a su mayor aportación colaborativa al proyecto), además de indicar su evolución durante los meses analizados.

"Ahora Bruce sabe quiénes son los expertos que estaba buscando, y más concretamente, ha descubierto que Ben es el experto más destacado en este campo dentro de su nuevo departamento. Por tanto, una vez este contexto colaborativo ha sido generado para Bruce, podría hablar con Ben en tiempo real (mediante videoconferencia o mensajería instantánea), seguir las conversaciones que ha mantenido en el foro o unirse a los próximos eventos que Ben u otro experto organice en el componente de Agenda y Calendario para tratar temas relacionados con el cloud computing".

3.4 Galactus

El segundo proyecto en el que el alumno ha estado trabajando durante este año recibe el nombre de Galactus, y tiene como principales objetivos los siguientes:

- Construir un sistema de recomendación de restaurantes de toda España basado en datos bancarios sobre transacciones de sus usuarios en dichos comercios proporcionados por la entidad.
- El sistema deberá tener como datos de entrada para la recomendación información de contexto del usuario que pueden ser la hora a la que se desea comer, el día de la semana y la dirección de lugar donde se desea encontrar restaurantes que cumplan los parámetros anteriores.
- La aplicación debe poder ser accesible desde la web para facilitar el que sea consultada en cualquier momento y desde cualquier lugar.
- Se debe conseguir un interfaz de usuario sencillo, fácil de usar y agradable al usuario, de manera que el cliente pueda ser incluido como *widget* en aplicaciones de terceros.
- Los resultados de la recomendación deben mostrarse utilizando mapas para su correcta localización física.

De esta forma, el modelo elegido para llevar a cabo esta aplicación fue el de Cliente-Servidor, pudiendo realizar una consulta a la misma desde cualquier navegador web y permitiendo por tanto a los usuarios poder obtener una recomendación desde cualquier dispositivo que tenga instalado Flash Player (ya que como se verá más adelante, el cliente web está implementado en Flex).

Por tanto, para comprender correctamente el desarrollo que se llevó a cabo, vamos a presentar en primer lugar el lado servidor, para describir a continuación el lado cliente. Adicionalmente, se presenta la siguiente figura con la arquitectura global del sistema donde se pueden ver todos los módulos, tanto los desarrollados por el alumno, como los de aplicaciones de terceros que se han utilizado para proporcionar algunas de las características requeridas:

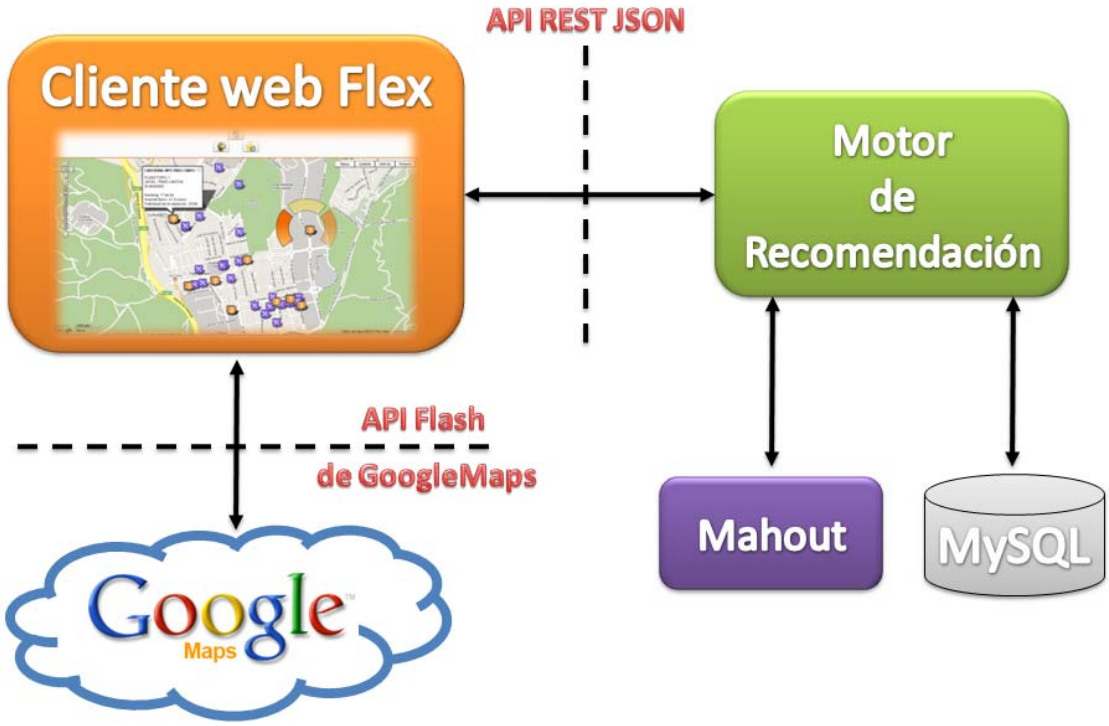

Figura 26. Arquitectura general de Galactus

3.4.1 Servidor: clustering de datos y recomendación

Tal y como se puede ver en la figura anterior, el lado servidor situado a la derecha está compuesto por tres módulos que vamos a describir a continuación.

Base de datos MySQL

Es la encargada de mantener los datos sobre restaurantes y transacciones bancarias efectuadas en ellos. Luego, dispone de una base de datos con las siguientes tablas:

transactions table

Nombre de la columna	Tipo de variable	Significado
transactionId	INT NOT NULL AUTO_INCREMENT PRIMARY KEY	Identificador único de transacción
amount	INT NOT NULL	Pago realizado con tarjeta en el restaurante
date	VARCHAR (250)	Fecha del pago
time	VARCHAR (250)	Hora del pago
codCom	INT NOT NULL	Código de comercio que identifica al restaurante

restaurants table

Nombre de la columna	Tipo de variable	Significado
codCom	INT NOT NULL AUTO_INCREMENT PRIMARY KEY	Identificador único de comercio
name	VARCHAR (250) NOT NULL	Nombre del restaurante
address	VARCHAR (250) NOT NULL	Dirección del restaurante
postalCode	INT NOT NULL	Código postal
townName	VARCHAR (250) NOT NULL	Nombre de la población
phone	INT NOT NULL	Teléfono del restaurante
latitude	DOUBLE NOT NULL	Latitud del restaurante
longitude	DOUBLE NOT NULL	Longitud del restaurante
accuracy	INT NOT NULL	Precisión de la localización por coordenadas

Como vemos el JOIN entre ambas tablas se hace a través de la columna **codCom**, que en la tabla **restaurants** es *primary key*, y en la tabla **transactions** es *foreign key*.

Mahout

Como ya se comentó en la sección correspondiente de "Estado del Arte" sobre sistemas de recomendación y métodos de clustering, Mahout nos provee de un conjunto de algoritmos de clustering, distancias métricas y algoritmos de

recomendación implementados en Java que podemos utilizar como librerías a las que llamar desde módulos de nivel superior.

En el caso de este proyecto, y aunque no se utiliza por el momento, ya se ha integrado este módulo para en un futuro poder llevar a cabo recomendaciones basándonos en datos personales de los usuarios como sexo o edad, y que como veremos en la sección "3.4.3 Gestión de usuarios y autenticación SSO basada en redes sociales", serán datos de los que dispondremos al conectarnos con dichas redes sociales y poder acceder a los perfiles de sus usuarios.

Motor de Recomendación

Este módulo desarrollado por el alumno en Java es el encargado de recibir los datos de entrada proporcionados por el usuario para la recomendación y aplicar el proceso de clustering anterior con el fin de devolver una recomendación adecuada.

La comunicación con el cliente web se hace a través de un API JSON que sigue la filosofía REST, de manera que los parámetros de consulta entregados por el usuario se mandan en formato JSON y la recomendación de restaurantes también se hace siguiendo el mismo formato.

Para conseguir esta arquitectura en Java se ha utilizado la librería Jersey [42], perteneciente a al proyecto GlassFish de código libre bajo licencia CDDL+GPL que sigue la referencia JAX-RS (JSR 311) para construir servicios Web con filosofía RESTful. La ventaja de utilizar esta implementación de referencia es que Jersey provee un API a los desarrolladores que facilita el desarrollo siguiente dicha filosofía.

De esta manera, el proceso que lleva a cabo el motor de recomendación está formado por los siguientes pasos que se detallan a continuación:

1. El usuario, a través del cliente web envía mediante el API REST un documento JSON que contiene la siguiente tupla de datos <hora, tipo de día, cuadrante de coordenadas>. Este cuadrante de coordenadas no es más que una representación de la vista de mapa que el usuario está observando y que está centrada en la dirección que proporcionó en el formulario de entrada para elaborar su recomendación.
2. El motor de recomendación realiza una consulta a la base de datos para obtener los comercios que están dentro de ese cuadrante.
3. Una vez hecho esto, se consulta nuevamente a la base de datos para obtener las transacciones que pertenecen a esos comercios y que están dentro del rango de hora y tipo de día especificado por el usuario.
4. Tras obtener las transacciones que cumplan todos los requisitos, se contará el número de ellas por comercio, de manera que podamos construir un ranking

de restaurantes para la recomendación basado en el número de pagos que se han hecho con tarjeta en ellos. Es decir, los restaurantes con más pagos serán los que estén en posiciones más altas del ranking.

5. Adicionalmente, y aprovechando este último análisis de las transacciones por comercio, obtendremos un valor medio basado en los importes realizados para generar una variable de "importe típico" que será devuelto en la recomendación.
6. Finalmente, se devuelve al cliente web un archivo JSON con los restaurantes recomendados ordenados según el ranking calculado y que llevará los siguientes datos para cada uno de ellos tal y como se puede ver en el resultado que muestra el cliente web en la siguiente figura:
    - Nombre
    - Dirección
    - Código postal
    - Población
    - Teléfono
    - Posición en el ranking
    - Importe típico
    - Fiabilidad de localización

Figura 27. Captura de pantalla de una recomendación ofrecida por el cliente web

De esta manera, y como hemos podido ver, el motor realiza la lógica de recomendación y trata la información de la base de datos teniendo en cuenta los parámetros propuestos por el usuario que están relacionados con su contexto físico y temporal actual. Evidentemente y como hemos comentado anteriormente, puesto que el módulo de Mahout ya está integrado en la arquitectura del sistema, si el usuario introdujese una información más personalizada de su contexto personal (edad, sexo, etc.), la obtuviésemos de aplicaciones sociales como comentaremos más adelante, o por

otro lado, si tuviésemos un sistema de puntuación de restaurantes o características similares en la aplicación, utilizar las capacidades de recomendación avanzadas que ofrece Mahout sería algo inmediato y que nos daría recomendaciones más certeras partiendo de un contexto de usuario mucho más rico, como por ejemplo, donde comen sus contactos sociales.

3.4.2 Cliente web: gestión de consultas y resultados en mapa

El cliente web ha sido desarrollado bajo una filosofía RIA a la que ha contribuido el uso de Adobe Flex para su implementación. Además, se ha elegido la arquitectura MVC aportada por el uso del *framework* Cairngorm [29] que sin duda alguna facilita el desarrollo de aplicaciones basadas en el paradigma Cliente-Servidor, ya que desacoplamos perfectamente la gestión de datos, de la lógica de la aplicación y el interfaz de usuario.

Adicionalmente, y puesto que uno de los requisitos de diseño era el uso de mapas para la localizar los restaurantes, se eligió Google Maps como solución por su amplia gama de posibilidades y su actualización continúa. Por ello, y puesto que el cliente que se estaba desarrollando era bajo tecnología Flex, se optó por usar el API Flash de Google Maps [43] que provee un amplio conjunto de métodos para gestionar varios tipos de mapas, marcadores de localizaciones, rutas, etc. Todo ello reunido en la biblioteca *map_flex_1_18.swc* apta para ser utilizada desde Flex Builder.

De esta forma, lo primero que se hizo fue obtener una clave de uso de Google Maps para poder utilizar dicho API, teniendo en cuenta que no existían limitaciones de uso hasta las 500.000 visitas diarias, aspecto que para esta primera versión del sistema sin duda alguna no era un problema.

Una vez integrada la biblioteca y conectada la aplicación con Google Maps para tener el mapa cargado en nuestro cliente web Flex, la gestión de localización se hizo de la siguiente manera:

1. Cuando un usuario introduce los datos de la recomendación a través del formulario inicial, se recoge la dirección proporcionada y se crea un objeto *ClientGeocoder* al que se le pasa el String que identifica a dicha dirección.

Figura 28. Captura de pantalla del formulario de entrada para realizar la recomendación

2. Tras esto se ejecuta el método *geocode()* que inicia una comunicación con Google Maps para que nos devuelva el resultado de traducir la dirección anterior a un punto del globo mediante su latitud y longitud.
3. Suponiendo que la consulta haya tenido éxito (se activa un evento de tipo *GeocodingEvent.GEOCODING_SUCCESS*), Google Maps nos devuelve el resultado de haber posicionado dicha dirección que es recogido por un *listener* en el que nos encargamos en primer lugar de centrar la vista del mapa en esas coordenadas, y en segundo lugar de aumentar el zoom para que se vea correctamente la dirección que se ha buscado.
4. Una vez hecho esto, tenemos el mapa centrado y adaptado a la dirección que se ha buscado, y de esta manera generamos en el cliente un cuadrante de coordenadas que se consigue obteniendo los valores máximos y mínimos de latitud y longitud que se pueden ver en el mapa. Es decir, tomamos cuatro parámetros que son: latitud y longitud de la esquina derecha superior y la esquina izquierda inferior, tal y como muestra la figura:

Figura 29. Creación del cuadrante de coordenadas para la recomendación

5. Tras todos estos cálculos, y a través del API JSON ya explicado anteriormente, se envía la tupla de datos <hora, tipo de día, cuadrante de coordenadas> para que el Motor lleve a cabo la recomendación de restaurantes.
6. Finalmente, cuando el motor devuelve el resultado de la recomendación al cliente, lo que se tiene es un documento JSON en el que de manera ordenada (por ranking) se indican los datos y localización de cada uno de los restaurantes que se encuentran dentro del cuadrante anterior y que cumplen los parámetros de entrada. De esta forma, y tras parsearlo adecuadamente se obtiene cada uno de los restaurantes que hay que posicionar en el mapa. Para ello se utiliza la gestión de marcadores de Google Maps, aunque en el caso de esta aplicación, el alumno desarrolló su propio recubrimiento de la clase *Marker* para conseguir integrar marcadores personalizados. Además, y para gestionar el problema de localizaciones con múltiples resultados, se implementó una solución mediante menús radiales que se crean de manera dinámica en tiempo de ejecución en base al número de resultados que comparten las mismas coordenadas, y en lo que se tiene un degradado de color en el menú en base a la posición en el ranking de cada uno de los restaurantes representados por él. En la siguiente figura, podemos ver un ejemplo de los resultados generados en la aplicación tras realizar la consulta de la figura 27.

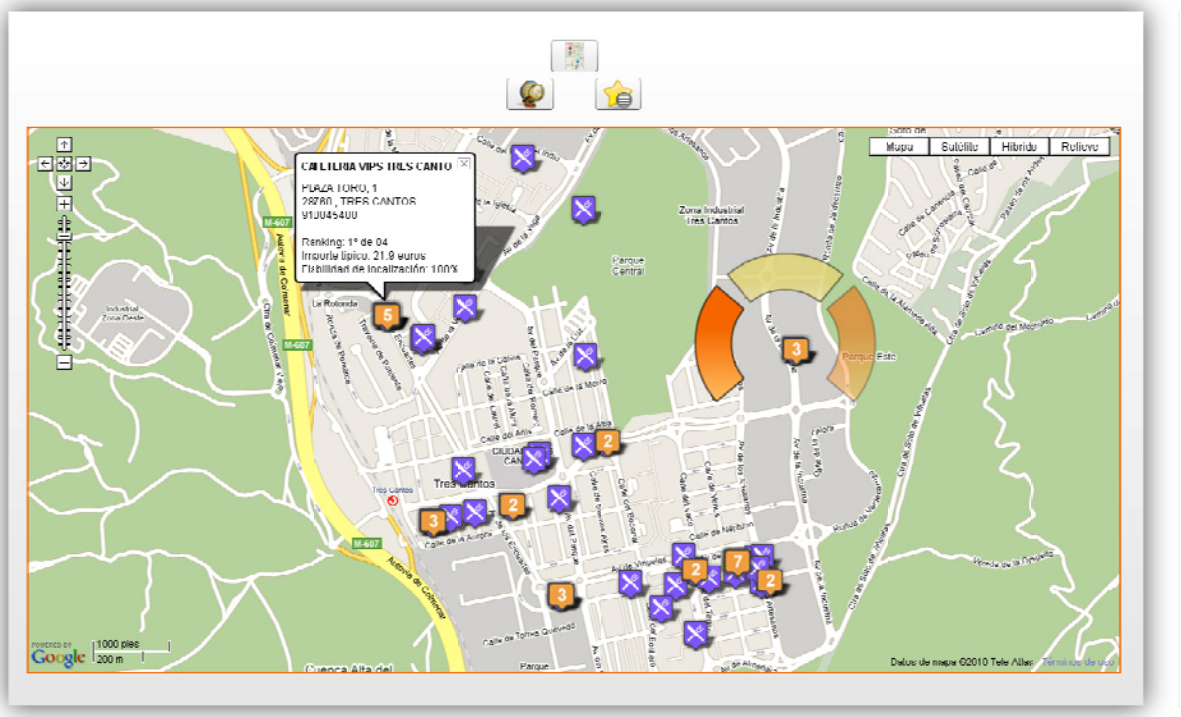

Figura 30. Captura de pantalla con resultados posicionados en el mapa y ejemplo de menú radial para localizaciones geográficas con resultados múltiples

En consecuencia, y como se puede ver en la figura, los restaurantes se identifican con un marcador simbólico o numérico dependiendo del número de resultados que existan en las mismas coordenadas geográficas. Además, y para obtener la información de un restaurante basta con pinchar sobre el marcador para, en el caso de un único resultado obtener directamente los datos del restaurante. Por el contrario, en el caso de un marcador múltiple, al pinchar se despliega un menú radial que tiene tantos botones como restaurantes haya en esa localización, de manera que seleccionando cualquiera de ellos se puede ver su información, teniendo en cuenta que de izquierda a derecha se produce un degradado de color para informar que cuanto más degradado esté, su posición es peor en el ranking global de la recomendación.

Adicionalmente a la pantalla anterior, se ofrece una segunda forma de visualizar los datos en formato lista, donde pulsando en cualquiera de ellos se lleva al usuario automáticamente al mapa destacando el restaurante elegido con su información. Esto nos sirve para comprobar de una manera mucho más rápida cual es el orden de los restaurantes ofrecidos en el ranking global de la recomendación para esa zona, como podemos ver en la figura que aparece a continuación.

Figura 31. Captura de pantalla con los 10 primeros resultados ordenados según su posición en el ranking de la recomendación ofrecida por el sistema

Evidentemente, y ante cualquier movimiento del mapa realizado por parte del usuario, ya sea por acciones de *drag and drop*, de zoom o de uso de las flechas de desplazamiento, la recomendación de restaurantes en el mapa se actualizaría para mostrar los nuevos resultados adecuados a la nueva zona que se visualice (al haberse actualizado el cuadrante de coordenadas), y que en todo momento deben cumplir los parámetros de búsqueda iníciales hasta que el usuario no decida cambiarlos.

3.4.3 Gestión de usuarios y autenticación SSO basada en redes sociales

Por último, y aunque por el momento no se ha integrado un módulo de autenticación al sistema, si se ha estudiado ya cual será el modelo de gestión de sesiones de usuarios elegido. De esta forma, y puesto que el sistema tiene una clara orientación a ser utilizada como *widget* en aplicaciones de terceros, o más interesante aún, convertirse en una aplicación integrada en una red social como Facebook, la solución más interesante para gestionar los usuarios que puedan hacer uso de ella sería el de aprovechar dichas redes sociales como gestores de identidad y autenticación.

Así, eligiendo esta opción simplificaremos por un lado la gestión de los usuarios, ya que la delegamos en dichas redes sociales, y por otro y de cara al futuro, nos

reservamos la posibilidad de poder acceder a los datos personales que dichos usuarios nos den permiso para usar con el objetivo de poder mejorar las recomendaciones que el sistema les ofrezca, ya que completaremos los actuales datos de entrada con parámetros como la edad, el género, su localización actual, etc.

Por tanto, hemos elegido integrar en un futuro cercano dos esquemas de gestión de autenticación de tipo Single Sign-On (SSO): Facebook y Twitter, ya que entendemos que debido a la amplia penetración de ambas redes sociales (probablemente las de mayor número de usuarios en el mundo), la mayoría de los usuarios actuales de Internet tiene al menos cuenta sino en ambas, al menos en una de ellas. Es más, de cara a la posible creación de módulos futuros que debiesen estar integrados bajo un mismo proceso de autenticación, la elección de un sistema de SSO es sin duda alguna una buena manera de asegurarnos que no tendremos que gestionar en el futuro más que una instancia de autenticación para acceder a todo nuestro sistema, siendo adicionalmente interesante de cara al usuario si ya participa en otras aplicaciones que utilicen este sistema, ya que automáticamente iniciaría sesión en todos ellos con una única entrega de credenciales.

Pasemos por tanto a detallar cómo funcionan las dos opciones que hemos elegido para Galactus.

Facebook SSO

Facebook permite a los desarrolladores de aplicaciones web modificar los procesos de inicio de sesión de sus sistemas para que se basen en realizar una acción de log in en dichas aplicaciones a partir del uso de las cuentas de Facebook de sus usuarios. Además, esto facilitaría que la aplicación obtuviese datos de dicha cuenta (nombre, foto, información de contacto, etc.) mientras el usuario tuviese sesión iniciada en Facebook, aunque por el momento y como hemos dicho antes, sólo nos ceñiremos al proceso de autenticación.

La plataforma de autenticación y autorización de Facebook se basa en el protocolo OAuth 2.0 (también conocido como OAuth WRAP) [38] para ofrecer un servicio de SSO que utiliza un SDK JavaScript [39] para proporcionar un API que hace transparente al desarrollador los detalles del protocolo anterior. De esta manera y para entender correctamente el proceso de autenticación nos apoyaremos en la siguiente para explicar los mensajes que intervienen, teniendo en cuenta que como paso previo a todo este protocolo se debe haber registrado la aplicación web en Facebook para obtener un identificador *APP_ID* reconocible por él.

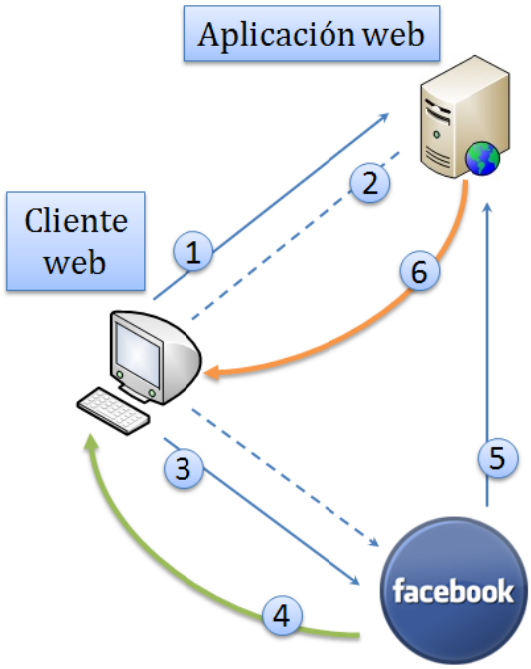

Figura 32. Intercambio de mensajes en el protocolo de autenticación SSO de Facebook

1. La comunicación comienza cuando el cliente solicita iniciar sesión en la aplicación web pulsando el botón de log in.
2. Si el usuario no tenía sesión iniciada en Facebook, se le redirige a un pantalla de la aplicación social en la que introducir sus credenciales, informando a Facebook de que aplicación de terceros está iniciando un proceso de autenticación SSO mediante el envío del identificador que previamente se posee de haber registrado la aplicación web.
3. El usuario introduce su email y contraseña e inicia sesión en Facebook
4. Tras el paso anterior, se genera una cookie de sesión de Facebook con nombre fbs_APP_ID que es entregada al cliente, y en la que el API JavaScript ha guardo los detalles del usuario autenticado.
5. Facebook informa a la aplicación web, representada por el identificador recibido previamente, de que el proceso de autenticación ha concluido con éxito.
6. Finalmente, se genera una cookie de sesión de la aplicación web que es entregada al cliente para que se complete la autenticación del mismo.

Adicionalmente, si el usuario en el paso 2 ya hubiese tenido una sesión con Facebook iniciada, y por tanto sólo poseyese en ese momento la cookie de Facebook, pero no la de la aplicación web, el SDK automáticamente detectaría esta condición y de manera transparente al usuario realizaría los pasos restantes para completar la autenticación sin necesidad de requerir las credenciales, ni de pulsar nuevamente el botón de log in. Por tanto, el *callback* denomidado *auth.sessionChange* del SDK es

llamado cada vez que el usuario comienza un proceso de log in o de log out en la aplicación web.

Evidentemente, una vez el usuario tiene sesión en Facebook, el proceso de SSO anterior se repetiría para cualquier otra aplicación web que estuviese registrada en la red social y a la que el usuario desease entrar.

Twitter SSO

Twitter, al igual que en el caso de Facebook, ofrece una gestión de autenticación y autorización centralizada que puede ser incluida en aplicaciones de terceros con el objetivo de evitar la gestión de procesos de inicio de sesión. Nuevamente, este servicio se basa en el protocolo OAuth, aunque con ciertas modificaciones que comentaremos a continuación. El en el protocolo OAuth Core 1.0 [40] se obliga a que las aplicaciones de terceros realicen las peticiones al método *oauth/authorize*, mientras que en el caso de Twitter [41] los tokens de autenticación deben ser enviados mediante el parámetro *oauth_token* al método *oauth/authenticate*.

Este método realiza las siguientes acciones que son ilustradas por el flujo de decisión representado por la siguiente figura y explicado en detalle en los puntos que vemos a continuación:

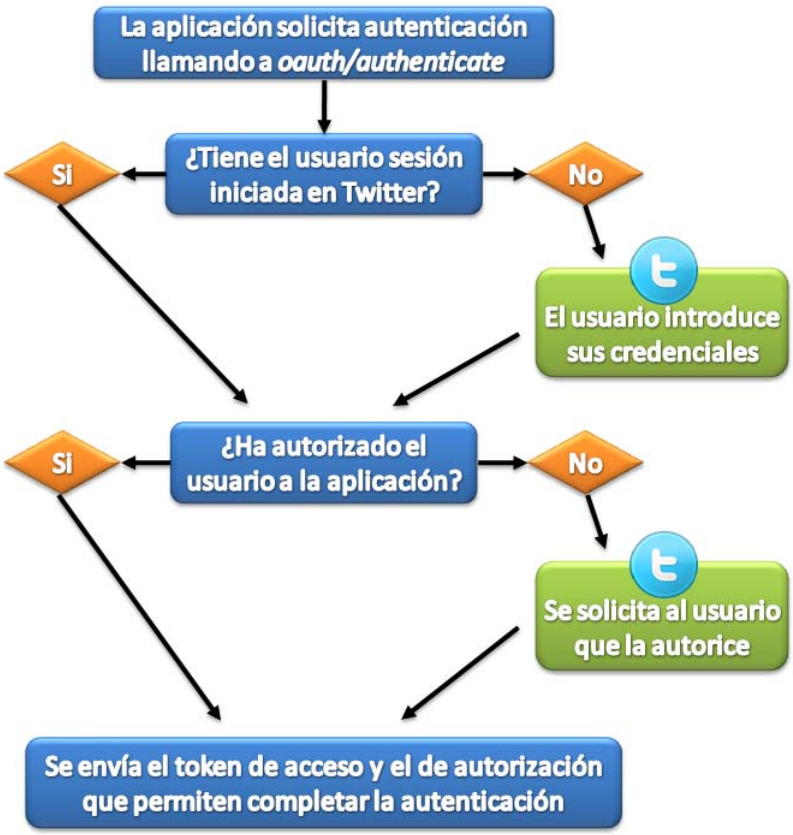

Figura 33. Flujo de decisión presente en el proceso de autenticación SSO mediante Twitter

- Si el usuario tiene sesión iniciada en Twitter y ha autorizado la conexión con la aplicación web, entonces el usuario se autentica de manera inmediata y es dirigido a la misma.
- Si el usuario no se ha iniciado sesión en Twitter, pero si ha autorizado a la aplicación web, entonces es redirigido a Twitter para iniciar sesión, y una vez hecho esto se autentica para ser devuelto a la aplicación original.
- Si el usuario ha iniciado sesión en Twitter, pero todavía no ha autorizado la aplicación, entonces se procede a solicitar la autorización OAuth pertinente para conectar Twitter con dicha aplicación de terceros, y una vez hecho esto, se devuelve al usuario a la aplicación web autenticado.
- Si el usuario no tiene sesión iniciada en Twitter y tampoco ha autorizado la aplicación para su autenticación mediante Twitter, en primer lugar debe iniciar sesión en la red social, y posteriormente, debe autorizar mediante OAuth a la misma para después ser redirigido a ella.

Por tanto, aunque el modelo de Twitter a nivel de arquitectura es similar al expuesto anteriormente en la figura 25 correspondiente a Facebook, tiene una forma sustancialmente diferente de autorizar a las aplicaciones que utilizarán el servicio de SSO de la red social. Mientras que en el primero las aplicaciones se autorizaban frente a Facebook mediante un proceso de registro en la fase de desarrollo que implicaba la posesión de un identificador (*APP_ID*) único que posteriormente se intercambia entre la aplicación de terceros y Facebook, en el caso de Twitter dicho registro no existe, y es por eso que se utiliza un proceso de autorización en tiempo de ejecución en el que el usuario debe aprobar la aplicación frente a Twitter antes de poder completar el proceso de autenticación.

Luego, en el caso de Facebook SSO es la aplicación web de terceros (es decir, nosotros como desarrolladores) la responsable de identificarse frente a Facebook con un registro previo que le proporcione un identificador único para usar en las llamadas del API JavaScript. Sin embargo, en el caso de Twitter es el usuario el que personalmente autoriza cada una de las aplicaciones sobre las que hacer log in con su cuenta, guardándose esta autorización expresa en la información asociada a dicho usuario en las bases de datos de Twitter.

4 Conclusiones

A lo largo de este documento hemos podido comprobar cómo los diferentes objetivos que se marcaron en primera instancia en las diferentes líneas de investigación que han llevado asociadas los trabajos se llevaron con éxito a buen puerto.

Por un lado, la elección de la plataforma colaborativa Itecsoft elaborada como Proyecto Fin de Carrera por el alumno, y usada como banco de pruebas durante este Trabajo Fin de Máster en el que integrar tanto Nuve, como el nuevo módulo de generación de contexto colaborativo, no hace más que confirmar que el trabajo realizado ha sido desarrollado con un enfoque investigador permanente muy centrado en el área de la colaboración de usuarios y las redes sociales, por lo que sin duda no se comete un error al afirmar que el campo de estudio en el que se sitúa el alumno es claro desde antes de comenzar el máster, y por tanto, no se ha hecho más que profundizar con un carácter si cabe más investigador que desarrollador durante el último año.

Por otro lado, la construcción de un sistema de colaboración desde cero como es el caso de Galactus, que a priori parece algo desligado de la línea de estudio anterior, rápidamente se ve redirigido al comprobar que se han establecido los cimientos necesarios para que en versiones futuras del sistema la plataforma sea capaz de producir recomendaciones basadas en contextos de usuario sociales mucho más ricos que los utilizados actualmente. Como prueba de esto, se puede ver que el estudio y la integración de Mahout en el sistema y la intención de utilizar redes sociales como Facebook o Twitter para gestionar a los usuarios de la aplicación y a sus procesos de autenticación bajo el modelo SSO, no hace más que confirmar la orientación clara que el alumno tiene por las aplicaciones sociales y las plataformas colaborativas basadas en contexto de usuario.

Sin duda alguna, y aunque no se ha explicitado en el presente documento, la integración de Galactus como componente adicional dentro de la plataforma Itecsoft sería inmediata pues se ha cuidado constantemente el uso de arquitecturas distribuidas sin acoplamiento de componentes, de manera que el cliente web de Galactus ha sido desarrollado como *widget* para ser integrado en aplicaciones de terceros. Además, si tenemos en cuenta que ambos proyectos utilizan tecnología Flex en los clientes web y servidores con APIs REST, la posible integración se antoja relativamente sencilla de cara a una posible evolución futura de ambas plataformas, la colaborativa y la de recomendación.

Desde un punto de vista más personal, si hablamos de los conocimientos adquiridos durante la elaboración de este Trabajo Fin de Máster, sin duda alguna han sido muchos y muy variados. En primer lugar, se ha profundizado en el uso de tecnologías como Flex o el diseño y la implementación de arquitecturas REST, que ya eran conocidos por el alumno. En segundo lugar, se ha trabajado intensamente en el estudio del mundo del cloud computing, dando como resultado la creación de Nuve y por ende, del artículo que se logró publicar en un congreso internacional. En tercer lugar, y también con resultados en congresos internacionales, se puede hablar del campo del *awareness* o contexto colaborativo, descubierto por el alumno durante una asignatura del máster y que tan buenos resultados ha logrado conseguir tras profundizar en él. Por último, podemos hablar del área de los sistemas de recomendación y los procesos de clustering, donde el alumno se ha interesado especialmente por los algoritmos matemáticos subyacentes y a como se gestionan desde Mahout, hasta el punto de estar iniciando en estos momentos la preparación de un artículo acorde a las evoluciones futuras que se quieren aplicar en Galactus.

Finalmente, y teniendo en cuenta que el alumno ha cursado este máster como paso previo para realizar sus estudios de doctorado, y por tanto su tesis, resulta gratificante descubrir que gracias a los trabajos realizados durante este máster, y especialmente a los desarrollos llevados a cabo para este Trabajo Fin de Máster, se ha conseguido publicar en entornos internacionales los resultados de una labor investigadora que el alumno se ha tomado muy en serio en el último año, y que sin duda alguna espera poder aumentar en los próximos años partiendo de la situación actual que se ha descrito en este documento.

5 Trabajos futuros

Para terminar, vamos a describir de manera muy resumida las posibles líneas de investigación y trabajos futuros que se desprenden de lo detallado en este documento.

Comenzaremos con el área de la generación de contexto colaborativo utilizando las estructuras de datos de Itecsoft, ya que aunque este proyecto ha sido desplegado dentro de un entorno empresarial real, y sin duda alguna la realimentación obtenida de sus usuarios siempre será un importante parámetro y fuente de información a tener en cuenta a la hora de seguir evolucionando y probando el sistema en el futuro, no debemos dejar de mirar más allá, pues existen otras líneas de investigación muy prometedoras que se han podido descubrir durante el transcurso de este desarrollo que son igualmente interesantes.

En primer lugar, el área de la generación de contexto colaborativo y sobre todo, el análisis de estructuras de datos sociales para obtener patrones de colaboración a partir de redes sociales y estructuras temporales son áreas que difícilmente se agotarán en el futuro. Más si cabe si tenemos en cuenta que cada día las relaciones entre usuarios en Internet son cada vez más complejas de acuerdo al alto grado de lazos sociales que se forman en los diferentes contextos en los que una persona puede colaborar socialmente hoy en día. Es por esto que estudiar nuevos métodos y algoritmos que nos permitan analizar más profundamente estos datos será siempre un vía investigadora en la que deberíamos trabajar para incluir los resultados que obtengamos en nuestro sistema, haciéndolo por tanto mucho más potente.

En segundo lugar, y basándonos en las ideas que se están proponiendo actualmente en el Social Web Incubator Group del W3C, y sobre todo en el interesante concepto del Socially Aware Cloud Storage propuesto por Tim Berners-Lee [47], se abre un campo de investigación sumamente atractivo, ya que intenta promover una reestructuración de las aplicaciones de redes sociales (Facebook, LinkedIn o Twitter) para que los datos de sus usuarios puedan ser usados por aplicaciones de terceros de manera estandarizada. Si pensamos por un momento en el modelo actual de almacenamiento de los datos de estas redes sociales, que en la mayoría de los casos actúan más como silos que como almacenes ordenados, y lo que es peor, que no permiten que dichos datos sean accedidos si no es a través de sus APIs específicas (normalmente completamente diferentes entre sí), podemos comprender rápidamente que es necesaria una forma de conseguir una arquitectura común de compartición de dichos datos, logrando que aplicaciones de terceros pueda hacer uso de ellos con una filosofía cloud que haga independiente su gestión mediante un API unificado.

Luego, parece claro que alcanzar este tipo de arquitecturas en la Web nos permitiría en un futuro unir los datos sociales que un usuario tuviese en dichas redes sociales con los datos que estuviesen almacenados en Itecsoft, consiguiendo de esta manera una integración muy interesante que nos permitiría generar un contexto colaborativo de ese usuario mucho más complejo y detallado, pudiendo ofrecerle una información sobre su entorno social y colaborativo mucho más útil y eficiente si cabe que la que proveemos actualmente.

En cuanto a la línea abierta con Galactus en el campo de los sistemas de recomendación, la evolución es clara y ya se ha mencionado someramente en este documento. Conseguir datos personales de los usuarios lo más fieles posibles a la realidad es sin duda alguna una manera excepcional de lograr que las recomendaciones que se ofrezcan sean lo más certeras posibles. De esta forma, intentar integrar nuestra aplicación con redes sociales que puedan proveer dichos datos personales, o mejor aún, acceder a los datos de los usuarios a través de entidades más serias como bancos o administraciones públicas que poseen una información mucho más detallada, nos daría una potencia de recomendación que actualmente ningún sistema de estas características tiene en el mundo. Evidentemente, acompañar la adquisición de estos datos con técnicas de clustering avanzadas y sobre todo, distribuidas sobre una arquitectura cloud con bases de dato NoSQL, nos permitiría procesar una cantidad inmensa de datos en un tiempo aceptable, incrementando más si cabe la potencia del sistema de recomendación.

Internet es el mayor depósito de información del mundo, y los usuarios cada vez están más dispuestos a alojar su información en la nube, por lo que colaborar con ellos para tener acceso a los mismos es una vía en la que se debe profundizar para que el futuro de los sistemas de recomendación se transforme en presente.

Bibliografía

[1] F. Moritz, "Rich Internet Applications (RIA): A Convergence of User Interface Paradigms of Web and Desktop - Exemplified by JavaFX". PhD thesis, University of Applied Science Kaiserslautern, Germany, January 2008.

[2] S. Ruby, D. Thomas, D. Heinemeier et al, *Agile Web Development with Rails*, Third Edition, The Pragmatic Bookshelf, 2009, 784 pages.

[3] Ruby, "Ruby 1.9.1 Release", 2010. [En línea]. Disponible en: http://www.ruby-lang.org/en/. [Accedido el 6 de Junio de 2010].

[4] Ruby on Rails, "Ruby on Rails 2.3.8 Release", 25 de Mayo de 2010. [En línea]. Disponible en: http://rubyonrails.org/. [Accedido el 6 de Junio de 2010].

[5] Lotus Notes, "IBM Lotus Notes", 2009. [En línea]. http://www-142.ibm.com/software/dre/ecatalog/detail.wss?locale=es_ES&synkey=I105893X30130U84. [Accedido el 7 de Junio de 2010].

[6] Lotus Domino, "IBM Lotus Domino", 2009. [En línea] http://www-142.ibm.com/software/dre/ecatalog/detail.wss?locale=es_ES&synkey=P105893S13390H40. [Accedido el 7 de Junio de 2010].

[7] Lotus Connections, "IBM Lotus Connections", 2009. [En línea] http://www-142.ibm.com/software/dre/ecatalog/detail.wss?locale=es_ES&synkey=O035990J93692T45. [Accedido el 7 de Junio de 2010].

[8] Microsoft Office Communicator 2007, "Microsoft Office Communicator 2007: Información general", 2009. [En línea] http://office.microsoft.com/es-es/help/HA102037153082.aspx. [Accedido el 7 de Junio de 2010].

[9] GetGlue, "The Network That Sticks To You", 2010. [En línea]. Disponible en: http://getglue.com/ [Accedido el 9 de Junio de 2010].

[10] G. Linden, B. Smith y J. York, "Amazon.com Recommendations: Item-to-Item Collaborative Filtering", IEEE Internet Computing, vol. 7, no 1, pp. 76-80, Enero 2003.

[11] C. Reinhardt, "Taxi Cab Geometry: History and Applications", TMME, vol. 2, no. 1, pp. 38-64, Julio 2004.

[12] J. B. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations", in Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, 1967, pp. 281-297.

[13] Apache Mahout [En línea]. Disponible en: http://lucene.apache.org/mahout/ [Accedido el 9 de Junio de 2010].

[14] Apache Hadoop [En línea]. Disponible en: http://hadoop.apache.org/ [Accedido el 9 de Junio de 2010].

[15] D. Arthur y S. Vassilvitskii, "k-means++: the advantages of careful seeding", in Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, 2007, pp. 1027-1035.

[16] A. McCallum, K. Nigam y L. H. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching", in Proceedings of the International Conference on Knowledge Discovery and Data Mining, Boston, Massachusetts, United States, 2000, pp. 169-178.

[17] Google Code University, "Google: Cluster Computing and MapReduce", 2007. [En línea]. Disponible en: http://code.google.com/intl/es-ES/edu/submissions/mapreduce-minilecture/listing.html [Accedido el 9 de Junio de 2010].

[18] J. Dean y S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", in Proceedings of the OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004.

[19] U. Kaymak y M. Setnes, "Extended Fuzzy Clustering Algorithms", in ERIM Report Series Research in Management, Rotterdam, The Netherlands, Noviembre 2000.

[20] P. McCullagh y J. Yang," How many clusters?", International Society for Bayesian Analysis, vol. 3, no. 1, pp. 101-120, 2008.

[21] J. L. Rodgers y A. W. Nicewander, "Thirteen Ways to Look at the Correlation Coefficient", The American Statistician, vol. 42, no. 1, pp. 59-66, 1988.

[22] D. Lemire y A. Maclachlan, "Slope One Predictors for Online Rating-Based Collaborative Filtering", in Proceedings of the SIAM Data Mining (SDM'05), Newport Beach, California, Abril 2005, pp. 21-23.

[23] Java [En línea]. Disponible en: http://java.com/ [Accedido el 12 de Junio de 2010].

[24] R. Fielding, "Architectural Styles and the Design of Network-based Software Architectures". Doctoral dissertation, University of California, Irvine, USA, 2000.

[25] RFC 4287, "The Atom Syndication Format", The Internet Engineering Task Force (IETF), 2005. [En línea]. Disponible en: http://www.ietf.org/rfc/rfc4287.txt [Accedido el 12 de Junio de 2010].

[26] RFC 5023, "The Atom Publishing Protocol", The Internet Engineering Task Force (IETF), 2007. [En línea]. Disponible en: http://www.ietf.org/rfc/rfc5023.txt [Accedido el 12 de Junio de 2010].

[27] Plataforma Adobe Flash [En línea]. Disponible en: http://www.adobe.com/es/flashplatform/ [Accedido el 12 de Junio de 2010].

[28] Adobe Flex [En línea]. Disponible en: http://www.adobe.com/es/products/flex/[Accedido el 12 de Junio de 2010].

[29] Adobe Open Source, "Cairngorm", 2008. [En línea]. Disponible en: http://opensource.adobe.com/wiki/display/cairngorm/Cairngorm [Accedido el 12 de Junio de 2010].

[30] Jasig Community, "CAS Protocol", 2009. [En línea]. Disponible en: http://www.jasig.org/cas/protocol [Accedido el 12 de Junio de 2010].

[31] Jasig Wiki, "RESTful API - CAS User Manual", 2009. [En línea]. Disponible en: http://www.ja-sig.org/wiki/display/CASUM/RESTful+API [Accedido el 12 de Junio de 2010].

[32] OpenLDAP, "A open source implementation of the Lightweight Directory Access Protocol", 2009. [En línea]. Available: http://www.openldap.org/ [Accedido el 12 de Junio de 2010].

[33] D. Gallego, "Arquitectura e Implementación de Interfaz Flexible de Usuario para la Integración de Aplicaciones y Servicios Web". Proyecto Fin de Carrera, ETSIT UPM, Madrid, España, Mayo 2009.

[34] J. Grudin, "Computer-supported cooperative work: history and focus". Computer, vol. 27, no. 5, pp. 19-26, 1994.

[35] D. Fisher and P. Dourish, "Social and temporal structures in everyday collaboration", in Proceedings of the Conference on Human Factors in Computing Systems. Vienna, Austria, 2004, pp. 551-558.

[36] S. Wasserman and K. Faust, Social network analysis: methods and applications, Cambridge, UK : Cambridge University Press, 1994.

[37] D. Fisher, "Using Egocentric Networks to Understand Communication", IEEE Internet Computing, vol. 9, no. 5, pp. 20-28, September 2005.

[38] OAuth, "OAuth WRAP", 2010. [En línea]. Disponible en: http://wiki.oauth.net/OAuth-WRAP [Accedido el 13 de Juni de 2010].

[39] Facebook developers, "Single Sign-On Authentication with the JavaScript SDK", 2010. [En línea]. Disponible en: http://developers.facebook.com/docs/authentication/ [Accedido el 13 de Junio de 2010].

[40] OAuth, “OAuth Core 1.0”, 2010. [En línea]. Disponible en: http://oauth.net/ [Accedido el 13 de Junio de 2010].

[41] Twitter API Wiki, “Sign in with Twitter”, 2010 [En línea]. Disponible en: http://apiwiki.twitter.com/Sign-in-with-Twitter [Accedido el 13 de Junio de 2010].

[42] Jersey, “Java RESTful Web services”, 2009. [En línea]. Disponible en: https://jersey.dev.java.net/ [Accedido el 13 de Junio de 2010].

[43] Google Code, “API de Google Maps para Flash”, 2010. [En línea]. Disponible en: http://code.google.com/intl/es-ES/apis/maps/documentation/flash/[Accedido el 13 de Junio de 2010].

[44] J. Cerviño, P. Rodríguez, J. Salvachúa, G. Huecas and F. Escribano, “Marte 3.0: una videoconferencia 2.0”, JITEL 2008, Madrid, España, Septiembre 2008, pp. 209-216.

[45] Red5, “Open source Flash server in Java”, 2005-2010. [En línea]. Disponible en: http://red5.org/ [Accedido el 13 de Junio de 2010].

[46] P. Rodríguez, D. Gallego, J. Cerviño, F. Escribano, J. Quemada and J. Salvachúa, "VaaS: Videoconference as a Service", 5th International Conference on Collaborative Computing: Networking, Applications and Worksharing, (CollaborateCom 2009), pp. 1 - 11, Noviembre 2009, Washington, DC.

[47] T. Berners-Lee, “Socially Aware Cloud Storage. First draft”, 2009. [En línea]. Disponible en: http://www.w3.org/DesignIssues/CloudStorage.html [Accedido el 13 de Junio de 2010].

[48] IEEE Xplore, “Digital Library”, 2010. [En línea]. Disponible en: http://ieeexplore.ieee.org/. [Accedido el 7 de Junio de 2010].

Apéndice

En esta sección se presentan los artículos aceptados en congresos internacionales que el autor elaboró de manera paralela al trabajo descrito en este documento como resultado de las investigaciones llevadas a cabo, lo que viene a reforzar el carácter innovador del trabajo realizado y su repercusión en el ámbito de la investigación.

El primero de ellos se encuentra publicado en actas del congreso y disponible para su consulta en IEEE Xplore [48], por lo que se adjunta en su formato original. Mientras que el segundo artículo, pese a haber sido aceptado recientemente, no se adjunta por no estar publicado en actas del congreso en estos momentos al no haberse celebrado todavía el mismo mientras se escriben estas líneas.

P. Rodríguez, D. Gallego, J. Cerviño, F. Escribano, J. Quemada and J. Salvachúa, "VaaS: Videoconference as a Service", 5th International Conference on Collaborative Computing: Networking, Applications and Worksharing, (CollaborateCom 2009), pp. 1 - 11, November 2009, Washington, DC.

D. Gallego, I. Martínez and J. Salvachúa, "Generating Awareness from Collaborative Working Environment using Social Data", Collaborative Technologies 2010 (IADIS Multi Conference on Computer Science and Information Systems 2010), 26 – 31 July, Freiburg, Germany.

VaaS: Videoconference as a Service

Pedro Rodríguez, Daniel Gallego, Javier Cerviño, Fernando Escribano, Juan Quemada, Joaquín Salvachúa
Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid,
Avda. Complutense 30, Ciudad Universitaria,
28040 Madrid, Spain
{prodriguez, dgallego, jcervino, fec, jquemada, jsalvachua}@dit.upm.es

***Abstract* — Internet is a place nowadays where interoperating services are offered which can be integrated or mashed up in order to fulfill user demands. This paper proposes a way to offer videoconference as a web service over an interface which can be used by third parties to enrich their applications. This interface includes a security mechanism supporting delegated authorization to allow integration into third party's environments. Via this interface virtual rooms are provided where users can collaborate with audio, video, shared applications, IM, etc. An implementation of these concepts is described, including performance figures and validation results. We would finally like to stress that this architecture has been defined to support a scalable cloud computing service over the Internet.**

Cloud computing; Videoconferencing; Web; Real-Time Collaboration; SOA; ROA.

I. INTRODUCTION

A wide variety of services is accessed nowadays via open interfaces, such that their functionalities can be combined offering an extra value to final users. Today we can share our photos in Flickr, with Twitter using widgets merged into iGoogle. Why not integrating collaborative rooms where we can work together with others. Internet services are becoming more and more usual, and mash-ups and combinations which generate added value are becoming a must in the Internet.

In this paper, we propose a new interface for integrating collaborative videoconferencing as another service in those mash-ups. It would provide to any web application a videoconferencing service, executable from the browser. This way, users can communicate among themselves easily.

Our proposal meets the following requirements:

- User management is taken care of by third parties' applications while it leaves the authorization to them.
- It focuses on conference rooms, where users meet to collaborate.
- The security requirements have been designed to allow integration into third party's applications.
- The interface provides some Quality of Service (QoS) inside each room to every application (or consumer services) used.

In other words, this interface aims to transform a traditional telecommunication service, such as videoconference, into a resource that will be used by third parties. Furthermore, this approach enables the transformation of a standard client-server service into a Cloud Computing service which provides to users access to a collaborative environment including audio, video and shared applications. This architecture has been named "Nuve".

The structure of the paper is as follows: the next section gives a summary of work closely related to this topic. Section 3 aims to place this work in context while the following two sections introduce the conceptual model and define the operations of this interface. Section 6 describes the designed security mechanism to authenticate requests to the interface. Finally, the last two sections present the results obtained and detail the conclusions drawn from the work.

II. RELATED WORK

A variety of videoconferencing applications exist in Internet such as [1], [2], [3], [4], etc. They all enable collaborative videoconferencing with more or less functionality. However, none of them provides a way for third parties' applications to take advantage of those resources. We feel that videoconference and real-time collaboration tools in general can be a very powerful complement for many existent applications.

Our idea is based on telecommunication operator's Data Centers. While typical software and Web companies, such as Google and Microsoft, are focusing their attention on terms like Cloud Computing when in fact, ISPs have Data Centers more sophisticated than they have. They can use these to house third parties' services, or, in the other hand, to restructure their own services.

In our proposal, Nuve, the evolution began from the Marte 3.0 [5] client-server web-conferencing application developed by our work group. It is based on rooms, and it provides users with tools for performing web collaboration such as: video, audio, instant messaging and desktop sharing.

III. VIDEOCONFERENCE AND CLOUD COMPUTING

There exist many videoconferencing applications but users do not use them very frequently and do not extract all the potential they have. This is probably because they cannot be easily integrated in existing user environments. This is why we think that by offering collaborative videoconferencing as a Cloud Computing service, users will integrate it more easily into their environments.

In [7], it is said that Cloud Computing distinguishes itself from other computing paradigms in the following aspects:

- *User-centric interfaces:* using Web browsers. Nuve allows users to connect videoconference rooms through a Flex application executing in the web browser. This application offers a common user interface for all of them.
- *On-demand service provisioning:* this paradigm provides resources and services for users on demand. Nuve provides rooms to the services so users can use it.
- *QoS guaranteed offer:* computing clouds guarantees bandwidth, latency and so on. Nuve guarantees a correct bandwidth to the services because it analyzes the connection state and it modifies the communication parameters to adapt it.
- *Autonomous System:* the cloud is autonomous and transparent to users. Our objective is for Nuve to become a videoconference room provider to higher level services.
- *Scalability and flexibility:* the architecture is flexible, so it can adapt and scale itself depending on the number of users. In the present version of Nuve, this property is not yet implemented, but in the future, Nuve pretends to be scalable and flexible using clusters.

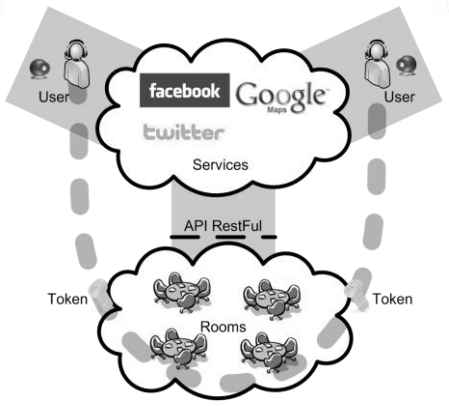

Fig. 1 Design of Videoconference as a Service

In addition, Nuve has some technological properties that also play an important role in Cloud Computing. These properties are described in [7]:

- *Virtualization:* in the present version there is only one virtual machine. However, in the future many virtual instances of Nuve could be started on demand, using similar services like Amazon EC2 [8].
- *Orchestration:* using APIs, an application can be the result of orchestrating a set of services. One of these services could be Nuve.
- *Web Service and Service Oriented Architecture:* Nuve offers an interface to control their resources, in other words, the videoconference rooms.
- *Web 2.0:* the Nuve user interface follows the philosophy of the Rich Internet Applications (RIA), using Flex applications to increase the usability.

In words of the NIST [6] "*Cloud computing is a pay-per-use model for enabling available, convenient, on-demand network access to a shared pool of configurable computing resources*". Thus, using REST interfaces to offer our Cloud Computing services seems to be natural.

Regarding the API architecture in this context, the objectives of our work were:

- Following the REST principles designing a generic, simple, flexible, reusable and stateless interface, useful to a multiconference service.
- Building a Cloud Computing oriented service that follows the principles we are going to explain later.
- Implementing a proof of concept.

The API developed will allow us to do all this things in the future.

IV. Conceptual Model

The main objective of Nuve architecture is to offer a videoconference service for third parties. As such, others will manage and control the communication while the system is the responsible of maintaining the communication alive and guarantee a Quality of Service. To understand this correctly, first we need to define a model based on resources and actors that will use those resources. In this section, we are going to describe the resources and their functionality inside the service.

TABLE 1
RESOURCES FROM THE CONCEPTUAL MODEL

Resource	Definition
Users	It represents the third parties' users who will communicate with others in each room.
Tokens	It is the ticket used to delegate the authorization of users to third parties' applications.
Services	They are the third party's applications that manage the rooms and give access to users in order to communicate in each of these rooms.
Rooms	They are the virtual spaces where the users will collaborate among themselves with video, audio and desktop sharing.

The conceptual model this architecture follows is that of a videoconference room provider (Fig. 1). We can define these rooms as meeting points where users will be able to establish conversations using audio, video and IM. Besides, they will be able to share applications executed in their own desktops. The management and control of these rooms is taken care of by other services. For this reason, they will be responsible for making these rooms accessible and giving users and other

services the needed authorizations (via tokens). In accordance with this, we can define four basic resources (TABLE 1). Each of those types of resources will be managed over the REST interface of Nuve with the standard HTTP operations: GET, POST, PUT and DELETE.

A. Users

A Nuve user is a person who has accessed a room and is communicating with other users from Nuve in the same room.

Every user can share his audio from his microphone or send the video obtained from his webcam. Besides, he can show or give control over his applications that are being executed in his computer. Also, he can chat to other users using the IM client incorporated in the room. All of these things are done using an application that is executing in the web browser through the user's Flash Player.

Using Flash Player as base for a videoconference application allows us to assert (based on [9]) that, in the great majority of desktop computers it is not necessary to install any kind of software, apart from the software that is already installed. To understand better this statement, we could say, for instance that any computer which has already accessed a video from Youtube, can access a Nuve videoconference without any problem.

Additionally, if a user wants to share applications executed in his computer, he must install the Java Virtual Machine. However, as we can see in [9], the Java Virtual Machine is usually installed in the great majority of computers.

In the real world, every participant in a meeting plays a different role so, in Nuve, there are three roles for users:

1) Observer

The observer user is present in the room and can see and listen to everything that is happening, but he cannot take part.

2) Participant

This role represents those users that can speak with the rest sending their video, audio, IM and desktop applications.

3) Administrator

Administrator role allows changing other users control over their videos, audios, IM and applications. Also, administrators can change the mode with which the users show themselves to everyone in the room.

B. Rooms

Rooms are the main resource in which the Nuve service model is based on. As we defined previously, a room is a virtual space where the users can communicate among themselves in meetings. Depending on the character of the meeting, a Nuve room may be equivalent in the real world to a company meeting room, an auditorium, a round table, a university class or the living room of a house where friends gather to speak about daily events. The objective of the meeting is defined by the service and his users. The Nuve aim is to guarantee bandwidth and Quality of Service in all the rooms.

The possible communication channels among users are voice, video, instant messaging (IM) and desktop sharing.

C. Services

This resource represents those consumer services that want to use the rooms provided by Nuve. To understand it well, we can use Facebook as an example of a service as it could be arranged for its users to participate in a Nuve conference. When we add a new service in Nuve, in fact we create a shared key between them. This key is used by the service to make calls to REST methods from the API. From now on, the service can create, edit and delete rooms, giving access to its users or denying it.

D. Tokens

Although we will see them in detail in the API security section, we can define the tokens here as a resource necessary for the whole user's authentication between the service provider (Nuve) and the consumer. A token represents the entrance key for a user in an existing session in Nuve. The service is the one who requests the token through the REST interface. Finally, it provides the token to the user so he or she can connect to the session.

V. NUVE API

In this section we will explain the REST architecture done in this work (illustrated in Fig. 2), which, as we said before, is based on four resources: users, rooms, services and tokens. We will see which HTTP operations are enabled for each one and what is expected to send and received in each call.

Consumer services will send HTTP requests to this interface, asking for the creation/deletion of rooms, giving access to users and retrieving information about conferences. End users of these services could take the role of administrators who will create, delete and modify rooms, or it could be done by the service automatically. Even third parties could create a wrapper in order to offer different kinds of services that would include videoconferencing rooms, and would manage them through protocols like SOAP, XMPP, etc.

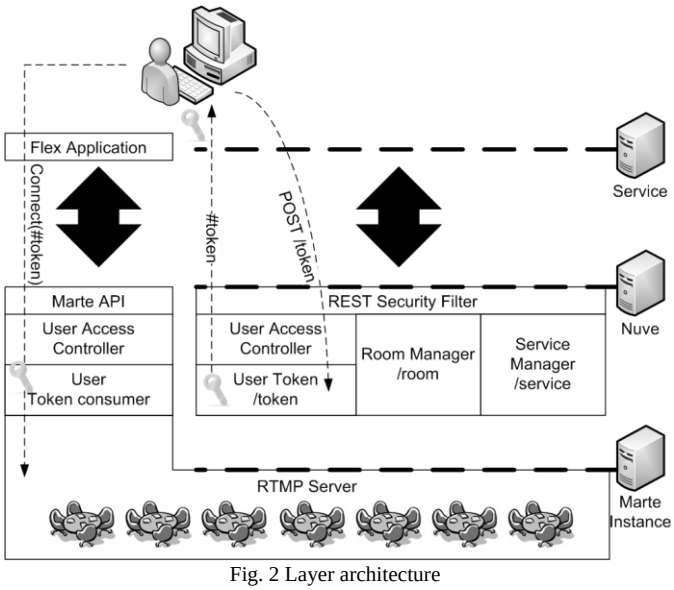

Fig. 2 Layer architecture

In order to support the usage of rooms by the services, the information about resources should be submitted in different ways. In our case, we have chosen the next three data structures

that are the most used nowadays in the World Wide Web: JSON [10] (which is the JavaScript serialization of objects, very important for objects implemented with this architecture), XML (that is a markup language with data structures compatible with JSON and is implemented in almost all application servers), and at last, HTML which is used mainly for making data representation easier to other services.

The API has been designed according to the recommendations from [11] for the purpose of making a good use of the REST philosophy. As it is usually recommended when designing this kind of APIs, our work proposes different commands using HTTP methods (GET, POST, PUT and DELETE) on the next four resources:

A. User

It represents users, providing different services for obtaining a list of them as well as information about one in particular.

We could get information about a user sending a GET HTTP request to a URL with the next structure: /room/{id}/user/{username}, being {id} the room identifier to which the user is supposed to be connected and {username} the name that the user has in this room. The response can be sent back in any of these formats: XML, JSON and HTTP. An example for this kind of communication would start with:

```
GET /room/321/user/Bob HTTP/1.1
Accept: application/xml
[Security info]
[CRLF]
```

The response would have the next structure:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 60
[CRLF]
<user><username>Bob</username><role>participant</r
ole><status>online</status></user>
```

Furthermore, we could also ask for the entire list of connected users by sending a GET command to the URL /room/{id}/user/. We could even throw a user out from the room by sending a DELETE message to /room/{id}/user/{username}.

On this resource we cannot create users, although it could be useful if we would want to invite users to one room. However, since a user is going to connect directly to Nuve, we decided to let the implementation of this functionality to consumer services. A use case could be a scenario in which a user would send an email to other person with the link to the room. If the other user would click the link it would take him to the room.

In short, and referring to tables 1 and 2, it is possible to request information in XML, JSON or HTML format of the resource which represents the user who is connected to the Nuve's room.

B. Room

It represents the space in which users can collaborate and share their video, audio and data. We can perform different operations such as create, modify, delete, list rooms and even give access to users. As we said before, this is the main resource of Nuve and, as such, it is important for us to have perfect control and get detailed information in various formats.

This resource designed to be directly managed by consumer services or by users that could interact with it through their respective services.

Regarding the available operations, we will start with those of them that are read-only, which are the same that in the case of the resource "user".

Any service could request information about any of the rooms which belong to it. For instance, by sending a method GET to /room/ if it wants to get a list of all of its rooms, or to /room/{id} if it wants information about only one of them. The information returned by this way can be represented in XML, JSON or HTML format, and an example of the HTML one is below:

```
<noscript>
<object  id="MarteRoom">
<param name="movie" value="Marte.swf"/>
<param name="quality" value="high"/>
...
<embed src="Marte.swf"
     quality="high"
     type="application/x-shockwave-flash"
     ...
     play="true"/>
</object>
</noscript>
```

This example shows part of the code of a web page which would be useful to include in the Nuve client that, as we mentioned before, is a Flash application.

Regarding the "write" requests, the first one would be the creation of a conferencing room that, as it usually occurs in REST architectures, is accessible through a POST method to the collection URL. In this case an example for this would be:

```
POST /room HTTP/1.1
Content-Type: application/json
Content-Length: 26
[Security info]
[CRLF]
{"room":{"name":"MyRoom"}}
```

The information sent to the server could be either in XML format or in JSON (that is the case of this example):

The response of the server would be the next:

```
HTTP/1.1 301 Moved Permanently
Content-Type: application/json
Content-Length: 26
Location: /room/MyRoom
[CRLF]
{"room":{"name":"MyRoom"}}
```

As we see in this example, the response in REST models to the request for the creation of a resource by means of a POST method is a message of the type "Moved Permanently", and the

response includes among its headers one that shows the URL where the information of the room is located.

The other writing operations are used to update and delete rooms. The first one is a PUT request sent to the URL of the room (in the example it would be /room/MyRoom) with new information about the resource. The response for this request would be a "200 OK".

The last option would be to delete the room. This can be done through the DELETE operation, as we can see at example:

```
DELETE /room/MyRoom HTTP/1.1
 [Security info]
[CRLF]
```

The response in the case of Nuve successfully removing the room would be a "200 OK".

C. Service

Nuve's API allows adding and removing services that are authorized to use its Rooms. Furthermore, we can ask for the entire list of services through a "root" service with special permissions. As in previous cases, to create a service we will use a POST operation to the URL of the resource collection (/service), and to delete an existent service we will do the same with the DELETE operation to the URL of the resource which we want to remove (/service/{service_id}). It is important to keep in mind that a service can only be removed by the service that owns it or by the "root" service.

Information about a resource can be represented in JSON, XML or HTML format, but we can only create a service by sending information with the first two ones.

Below is an example of a service described in XML format:

```
<service>
     <name>Facebook</name>
     <id>123okopwqeop21893</id>
     <key>u94832er893wjhr893j98</key>
</service>
```

We will see all this parameters in greater detail in a later section, when we talk about the security that we have implemented in this API.

As we saw before, we can retrieve information through a service with special permissions. In fact, this service is the only one allowed to manage the ownerships of the rest of services, controlling the creation and deletion of their rooms. Therefore, it is a integral part of the entire architecture. Its aim is to facilitate the work of administrators and that is why it is possible to provide information of each service in HTML format.

D. Token

The last resource of this API is the one that allow us to give access to end users. In the next section we will explain in detail its functionality, but now we are going to define the operation that can be used to create them. This operation is a little different because this is a special resource and, as we have already said thought essentially to authenticate users of other services. Besides it can be used to check that these users who are going to take part in conferences come from services that are the owners of such conferences.

The typical use case, for instance, could be one in which one of these resources takes part is the creation of one token. Tokens being nothing more than unique identifiers randomly created that have a limited time to live. In other words, some minutes later (usually three minutes), these token cannot be used and they are removed from the system. The way to create them is shown in the next example:

```
POST /room/MyRoom/token HTTP/1.1
Content-Type: application/xml
Content-Length: 63
[Security info]
[CRLF]
<token>
     <username>Bob</username>
     <role>participant</role>
</token>
```

When the server receives this request, Nuve creates a new unique identifier and relate it through a database to a username, a role and a room. Thanks to this, when a user wants to connect to one session, Nuve can retrieve this data from the database in order to know the room where the user wants to participate, with which username, and what role the user will play in the session. The response of Nuve would be the next:

```
HTTP/1.1 301 Moved Permanently
Content-Type: application/xml
Content-Length: 26
Location: /room/MyRoom/token/123p2j13io21
[CRLF]
<token>
     <username>Bob</username>
     <role>participant</role>
     <id>123p2j13io21</id>
</token>
```

It is necessary to know that each operation has the scope of the service that requests it and Nuve is aware of it thanks to the security information that the service includes in each of the requests. Also, as stated above, there is a special service called "root" that has enough permissions to operate over everything else.

We can see a summary of all operations that each service can use on each resource. The first table explains the operations that we can do with information in XML and JSON format, and the second one is for information represented with HTML.

TABLE 2
HTTP METHODS USED FOR XML/JSON CONTENTS

	GET	POST	PUT	DELETE
/user	X			X
/room	X	X	X	X
/token		X		
/service	X	X		X

TABLE 3
HTTP METHODS USED FOR HTML CONTENTS

	GET	POST	PUT	DELETE
/user	X			X
/room	X			X
/token				
/service	X			X

We can deduce the importance of the service authentication from the description that we have made through this section. Due to this, in the next one we will comment further details about it.

VI. SECURITY

This section describes the architecture of the solution implemented in order to provide the system with a security layer. The adopted mechanism is based on the combination of an extension to the HTTP header and the use of tokens for final user access.

A. Description of the problem and previous experiences

The next subsections explain the different approaches considered when designing the security solution for Nuve. Firstly, a general definition of what is considered as security in this kind of applications is given. After that, a brief analysis of the existing implementations is presented in order to provide some background on the existent works.

1) Security requirements in collaborative software

Before focusing on the problem, it is important to define the general security concerns in collaborative software (CSCW: Computer Supported Collaborative Software from now on).

A wide variety of software applications fall under the definition of CSCW, ranging from relatively simple document repositories to full blown real time multimedia systems that allow for much more complex interactions among users. The collaborative software matrix [12] perfectly illustrates that fact.

Due to that environment variety, security in this context is a little hard to define. A number of studies have been published trying to unify the definition of security requirements as well as the notation used to describe them properly ([13] and [14]). As a result of the analysis of those publications it is possible to extract a subset of rules that try to cover the most critical security problems:

- Participants follow the previously established workflow.
- The existent roles are consistent as well as the constraints imposed to them.
- Only authorized users can access the system.
- Users enter the system with the corresponding roles.
- Any temporal or conditional constraints can be applied to the resources.
- Each user's private data cannot be accessed by anyone else.

When it comes to videoconferencing (same place, different time in the matrix) the number of roles usually is very limited, simplifying interaction compared to other schemes with more complex workflows.

According to this, we reach the basic conclusion that in a context of videoconference where the usual workflow is quite simple, the main security concerns come from the authentication of the users and their authorization so they have access to the right resources depending on their roles.

2) REST Authentication

Having analyzed the security needs in a general context, it is time to study the different mechanisms that have already been implemented and published but focusing on REST which is the interface used in our system.

The most important alternatives researched and, as such, shown in this paper are: Amazon S3 [15] and OAuth [16]. Both were chosen because they are used broadly today and are proven to work.

First of all, it has to be noted that both are based, above all, on modifications of the standard HTTP authorization header defined in [17].

To sum the process up, it consists on the server responding to non authorized requests with a 401 code including a WWW-Authenticate field which specifies a challenge for the client. The next call coming from the client should include an Authorization field which is completed with the data implied by the challenge. Finally, the server should check the reply and process the request if it meets the requirements.

No specific type of authentication is enforced although in [18] two are proposed: Basic and Digest. The first one is really simple including only two fields (name and password) which are transmitted in plain text. Digest authentication is a little more complex as it proposes the use of MD5 [19] cryptographic hashing and nonce values to avoid replay attacks.

After this brief digression we switch our focus back to the studied systems. Amazon S3 is an online storage service offered by Amazon Web Serviced designed to make life easier for web applications developers with important needs in terms of data space but lacking the required infrastructure. It offers its users both SOAP and REST interfaces allowing them to perform various actions like adding or removing data. It is a paid service and, as such, it is extremely important for its users to be authenticated.

The modification of the HTTP header developed by Amazon is called "AWS". In this type of authentication, the reply to the initial 401 code is to introduce in the said header the following line: Authorization: AWS AWSAccessKeyId:Signature where AWSAccessKeyId is

given by Amazon to the client after registering as well as another key which will be used (together with a string containing several values) to calculate the signature by means of the HMAC_SHA1 algorithm.

This signature can be easily reproduced at server-side because the client's id and key are already known.

The other studied case we will explain here is the OAuth protocol which, among other security mechanisms, includes one similar to ones explained above. OAuth allows a user to grant access to their information on one site (the Service Provider), to another site (called Consumer), without sharing all of their identity. The communication between the provider and the consumer is not very different in concept to the one we see in Nuve.

The use of the HTTP header is quite similar to Amazon's but the information provided in it is not always the same and changes depending on the action requested. As a consequence, the data included in the header is not only used for authentication but also for application level purposes.

Furthermore, and without getting into too much detail, OAuth authentication uses a token which is given to the consumer to access the provider's resources. In the process, the client gives his or her credentials to the provider but never to the consumer.

To conclude, both alternatives aim to authenticate and authorize the agent that is requesting the operations. We conclude that, in the case of Nuve, this type of mechanism satisfies the security requirements obtained from the first part of this section. However, it is necessary to develop a more specific solution for our system.

B. Security in Nuve: MAuth

1) HTTP Authorization

Regarding HTTP Authorization, the path chosen is similar to the ones described above, that is, an extension to the HTTP standard header. However, like OAuth, the header is also used to carry parameters used by the application.

The full header containing all possible parameters used is as follows:

```
Authorization: MAuth
     realm="http://marte3.dit.upm.es",
     mauth_signature_method="HMAC_SHA1",
     mauth_serviceid="ServiceName",
     mauth_signature="jlsa731232=",
     mauth_timestamp="1231321321",
     mauth_cnonce="123123aadf",
     mauth_version="3.1",
     mauth_username="user",
     mauth_role="participant"
```

Below we describe individually each one of them:

- mauth_signature_method: It indicates the signature method used to sign the request. HMAC_SHA1 is the only one supported. The key used in the process is simetric and is exchanged off channel.
- mauth_serviceid: A unique identifier for the service. It is used by Nuve to obtain the key and for several other purposes at application level.
- mauth_signature: The signature generated by the method explained above.
- mauth_timestamp: Unless otherwise specified, the number of seconds since January 1, 1970 00:00:00 GMT. The value must be a positive integer and must be equal or greater than the timestamp used in previous requests.
- mauth_cnonce: A positive integer that must be different for each request with the same timestamp. Used to prevent replay attacks.
- mauth_version: Current version number.

All the parameters mentioned above are obligatory. The next two are only used for requesting access for a user to a room.

- mauth_username: Name of the user trying to access the conference.
- mauth_role: Role of the user in the conference. The possible roles a user can take in a room are: "participant", "administrator", and "observer". Each role defines limits to what a user can do while the conference is taking place.

The string used to calculate the signature varies depending on the parameters included in the header.

The format of the string is:

```
(mauth_serviceid, mauth_timestamp, mauth_cnonce,
[mauth_username, mauth_role])
```

The parameters between square brackets are only present when needed.

To better understand the flow of the authentication, let's study a particular case. A service wants to obtain the list of the existent conference rooms.

Initially, the service issues a request to Nuve:

```
GET /rooms HTTP/1.1
Host: marte3.dit.upm.es
```

The request did not include authorization information so the Nuve server replies with a 401 code indicating that the request was not authorized and providing information about the authentication type that should be used.

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: MAuth,
  realm="marte3.dit.upm.es"
```

Now, the service knows that Nuve uses MAuth to authenticate requests and must fill in every parameter to have its request approved and processed:

```
GET /rooms HTTP/1.1
WWW-Authenticate: MAuth,
  realm="marte3.dit.upm.es",
```

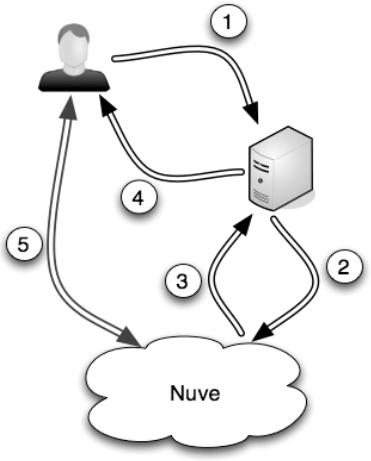

Fig. 3 Authentication messages

```
mauth_signature_method="HMAC_SHA1",
mauth_serviceid="global",
mauth_signature="dasawaraj212312",
mauth_timestamp="32132131",
mauth_cnonce="654sa5d6asdads",
mauth_version="3.1"
```

In this particular case, we are not using mauth_username and mauth_role as they are not needed for this request.

Once this point is reached, the server has all the needed data to verify the authenticity of the request, if everything is right it replies the service with a message including the list.

2) User authentication: tokens

This subsection gives a detailed explanation of the process of authenticating users in Nuve without the need of directly exchanging information between the final users' pc and the Nuve server.

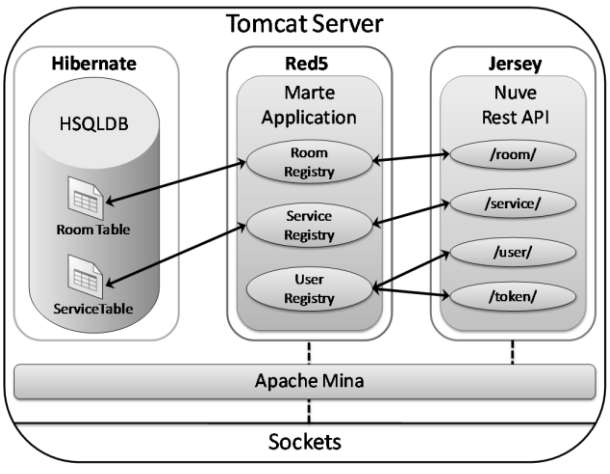

Fig. 4 Implemented code modules

This is achieved via the previously mentioned token. Only users that own a valid token have access to a Nuve conference. As explained above, tokens are generated dynamically every time a service asks for one, besides, information concerning the user (the service it belongs to and the role he or she will play) is stored in the server. Furthermore, tokens have an expiration date rendering them useless after a given period of time.

The usual workflow followed after the creation of a conference is illustrated in Fig.3.

1. The user requests access to a videoconference room. In order to do that, he or she probably has previously identified himself in the context of the service.

2. The service issues a request to Nuve asking for access to the conference for that specific user. That request includes all the authorization data mentioned before.

3. If the authentication is successful, Nuve sends a valid token back to the service.

4. The client is redirected to a dynamically generated web page which contains the Flash client needed to participate in the conference. The token contains data about the conference room and the user's role so he or she does not have to introduce any information and can only access the conference that was requested by the service.

5. The user can start interacting with others.

VII. RESULTS

This section describes the implementation of a proof of concept that we used to test the new architecture and draw conclusions about the results. Firstly, we will give a brief explanation about the test environment for this implementation detailing each of the components used and finally we will present all the tests performed and the results obtained from them.

A. Implementation

The API implementation has its roots in the Marte 3.0 application which, as we have explained before, was developed in a previous work. This application used an Apache Tomcat server on top of which a Red5 server was installed. Red5 allows for voice/video communications to be established among Adobe Flex/Flash clients. As Tomcat was an already set piece of the architecture we decided to use a Java library and we chose Jersey [20], developed by Sun and quite useful for creating RestFul APIs easily.

The working API had to embody the theories exposed throughout this article, so we started by defining the four mentioned resources (rooms, services, users and tokens). The logic behind the creation, modification, removal and reading of each of the resources was in its core a simple call to already implemented Marte classes so we did not have to rework all the parts concerning the management of the application. The most important parts reused from Marte are the following:

The RoomRegistry component takes care of all the functionality regarding creation and removal of rooms. Besides, it does so safely avoiding all the possible problems that might come up when performing those actions. For instance, when a room is deleted, RoomRegistry disconnects all the users present in those rooms providing the needed

explanation to the clients. Furthermore, it checks whether a service is authorized to delete a room.

The ServiceRegistry module is quite similar but it deals with services. When a service is created, it performs serveral actions like analyzing the existence of any conflicts with other subscribed services and when a service is deleted it deletes all its owned rooms by using the RoomRegistry component.

The last component used from Marte 3.0 is the UserRegistry which allows us to add or remove users from a room. For instance, we will use this component when a room is deleted via RoomRegistry to remove all the users from it.

Finally, to take care of all the persistence needs of the application, a HSQLDB database is used through Hibernate. It only stores data about subscribed services and rooms so two related tables were created on for each.

In Fig.5 the logic behind the removal of a service is shown as an example.

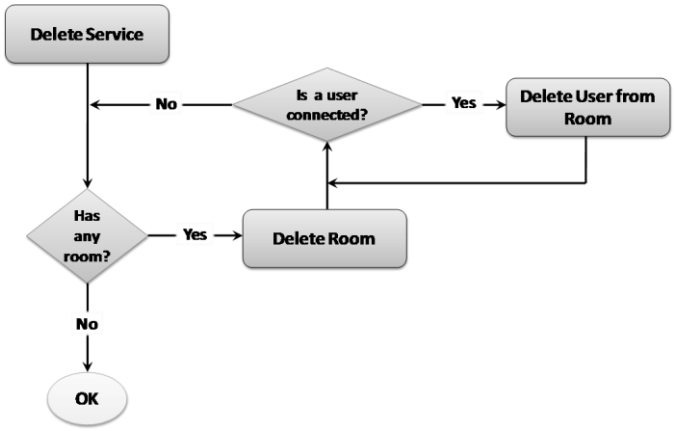

Fig. 5 Service deletion programming logic

B. Test environment

The objective of these tests was to measure the capacity of the Nuve server in terms of the number of users that was able to support, the bandwidth of a common session, monitoring the CPU usage of the machine in which the server is hosted.

The test system was a VMWare virtual machine running on top of an Intel Core 2 Duo with 2 GB of RAM. The virtual machine is limited to one CPU and 512 MB of RAM. The operating system used is an Ubuntu Linux 8.10.

In order to get this data we deployed the previous implementation in a machine to which different users from the same subnet were connected, all of this through an Ethernet connection and a Switch that supports a bandwidth of 1 Gbps. All the users and the server had network interfaces of 1 Gbps.

Regarding the bandwidth consumption we show different figures that we explain below.

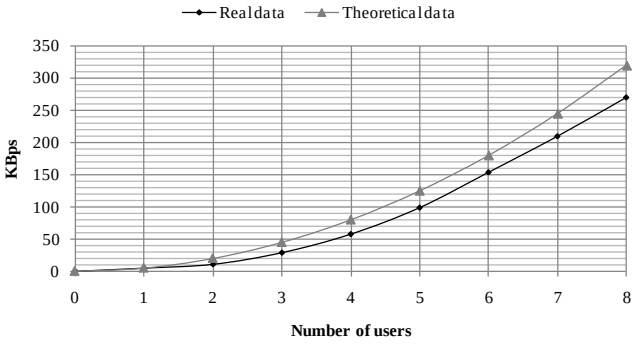

Fig. 6 Bandwidth consumed by the audio

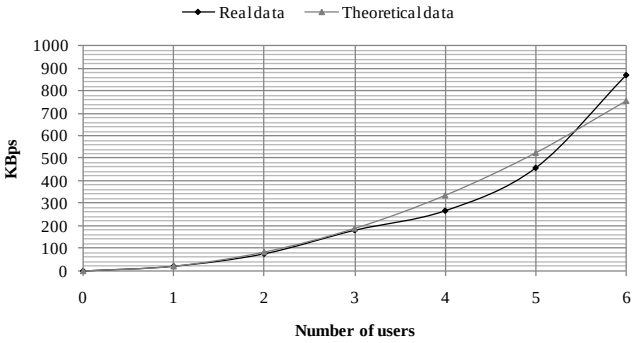

Fig. 7 Bandwidth consumed by both audio and video

At Fig. 6 we show the measured data on bandwidth that was used during a conference in which users only communicated using their voice and they did not use webcam or screen sharing. During these tests all users sent audio through their microphones at all times and simultaneously. We can also see an empirical approximation to these results that represents the next equation:

$$BW_{total} = {N_{users}}^2 \cdot BW_{audio}$$

Being BW_{total} the bandwidth consumed by the centralized server (which contains the Marte application), N_{users} is the number of users that are connected to the videoconference and BW_{audio} is the bandwidth consumed by the audio of each user (in the testing all users consume the same bandwidth).

Based on the bandwidth measured during the tests and applying the equation we get an approximated value for the bandwidth used by each user, that is 5 KBps.

Fig. 7 shows the bandwidth consumed by a session like the previous one, but in this case users are also sharing the video streams produced by their webcams. This test helps us to calculate the average bandwidth consumed by each user. We assume that the worst case is that in which all users are continuously in movement (that is the instant in which more bandwidth is going to be used). In this case the measured values should be represented by the next equation:

$$BW_{total} = {N_{users}}^2 \cdot (BW_{audio} + BW_{video})$$

We can get from this equation that the bandwidth consumed by each user that is sharing its video is 16KBps.

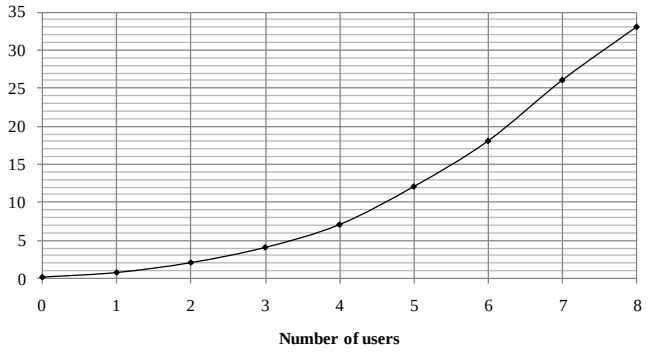

Fig. 8 CPU percentage consumed by the audio

At Fig. 8 we see how the percentage of CPU consumed varies in an audio conference, while at Fig. 9 we see the same data but in a conference that includes video and audio. As a result, we can infer that there are not many differences between videoconferences and audio conferences in terms of CPU usage in the server.

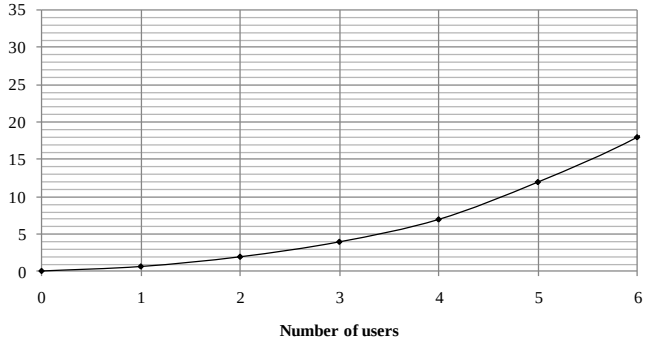

Fig. 9 CPU percentage consumed by both audio and video

It is important to notice that these tests were made in scenarios in which all users were connected to the same conference room. In a test with many rooms, the maximum bandwidth capacity of the server could be calculated by the next equation:

$$BW_{total} = \sum_i BW_{Room\ i} = \sum_i N_{users\ from\ i}{}^2 \cdot (BW_{audio} + BW_{video})$$

VIII. Conclusions and Future Work

Throughout this paper we have shown the main features of the proposed architecture in the context of other Cloud Computing systems. We have detailed the resources oriented architecture of Nuve and as well as provided a description of the service interface. A prototype of a Nuve system has been described validated and performance measures have been provided, showing that this architecture can be easily implemented in a cost-effective way. The extension of this implementation to scalable Cloud Computing services which could provide tons of virtual rooms to users seems to be straight forward by adding a virtual room allocator among the Cloud of virtual room servers.

The most critical part of this work has focused on the development of a security mechanism which enabling the integration of the service into third parties' applications and mash-ups.

The same core system has been reused in various projects we are working on. As part of these projects we have successfully installed the core in Linux, Windows, Mac OS X and Solaris. That is, we only have to maintain one group of physical CPUs which, in turn, have virtual machines running on top. The different projects are represented as services and they can all share the same, unmodified, core. Besides, while currently new virtual machines have to be set up by hand, we are working on the automation of the process. As a showcase we have integrated Nuve into Google Wave [21].

The Flex/Flash Rich Internet Applications (RIA) development framework from Adobe has proved to be very effective for implementing the prototype. The old Marte 3.0 client-server system was transformed into the Nuve architecture in three months by two persons working 60% of their time on it which seems to be very reasonable resource expenditure for such a development.

Using the Flex/Flash RIA based videoconferencing has the additional benefit of avoiding the user to have to install any application in most cases because most browsers today have the Flash Player plug-in installed.

Finally, even though the tests are focused on measuring the limits of the system and do not represent a real scenario where usually one user sends more information than the others, they serve us to estimate the maximum video and audio bandwidth consumption by a normal user in this first implementation. Regarding server CPU usage, the results show that in future works we have to design a low-level architecture that can be scaled through several server machines without overloading any of them. In order to achieve this scalability and guarantee some QoS, we will need to instantiate virtual machines and turn then on and off. This motivates us to follow the Cloud Computing principles.

References

[1] Google Video Chat. [URL] http://mail.google.com/videochat

[2] Skype. [URL] http://www.skype.com/

[3] Ribbit. [URL] http://www.ribbit.com/

[4] ooVoo. [URL] http://www.oovoo.com

[5] J. Cerviño, P. Rodríguez, J. Salvachúa, G. Huecas y F. Escribano, "Marte 3.0: una videoconferencia 2.0" JITEL 2008, pps: 209-216 16-18, September 2008.

[6] P. Mell and T. Grance, "Draft NIST Working Definition of Cloud Computing", January 2009.

[7] L. Wang, G. Von Laszewski, M. Kunze and J. Tao, "Cloud Computing: a Perspective Study", Dec. 2008.

[8] Amazon Elastic Compute Cloud (Amazon EC2) [URL] http://aws.amazon.com/ec2/, access on June 2009.

[9] Flash Player Penetration [URL] http://www.adobe.com/products/player_census/flashplayer/, access on June 2009.

[10] JSON [URL] http://www.json.org/, access on June 2009

[11] Cesare Pautasso and Erik Wilde, "From SOA to REST - Designing and Implementing RESTful Services". Tutorial at 18th Int. World Wide Web

Conference, Madrid 2009. [URL] http://www2009.org/tutorials/T9-F.html, access on June 2009.

[12] R. Johansen, “Groupware: Computer support for business teams”. New York: The Free Press 1988.

[13] Tanvir Ahmed, Anand R. Tripathi, “Static Verification of Security Requirements in Role Based CSCW Systems”, Symposium on Access Control Models and Technologies, 196-203 Como, 2003. ISBN: 1-58113-681-1.

[14] A. Tripathi, T. Ahmed, and R. Kumar. “Specification of Secure Distributed Collaboration Systems. IEEE International Symposium on Autonomous Distributed Systems (ISADS), April 2003.

[15] Amazon S3 [URL] http://aws.amazon.com/s3/, access on June 2009

[16] OAuth [URL] http://oauth.net/, access on June 2009.

[17] R. Fielding, J. Gettys, J. Mogul, H. Frysyk, L. Masinter, P. Leach and T. Berners-Lee, "Hypertext Transfer Protocol - HTTP/1.1", RFC 2616, June 1999.

[18] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, E. Sink and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[19] R. Rivest, “The MD5 Message-Digest Algorithm”, RFC 1321, April 1992.

[20] JSR 311: JAX-RS: The Java API for RESTful Web Services [URL] http://jcp.org/en/jsr/detail?id=311, access on June 2009.

[21] Google Wave. [URL] http://wave.google.com