OpenGoo

Desarrollo de una Oficina Web Open Source

Informe Final de Proyecto de Grado

Carrera de Ingeniería en Computación

Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay

Estudiantes: Ignacio de Soto, Marcos Saiz

Cliente: Conrado Viña

Tutores: Eduardo Fernández, Tomás Laurenzo

Resumen

OpenGoo es una aplicación de oficina a la que se accede mediante un navegador. Incluye módulos de edición de documentos, edición básica de hojas de cálculo, edición de presentaciones y gestión de proyectos. Es además, una herramienta para mantener la información organizada y centralizada: organizada por las funcionalidades de búsqueda y categorización por etiquetas; centralizada por tratarse de una aplicación web que puede ser accedida desde cualquier computadora conectada al servidor.

Se trata de un proyecto de código abierto basado en un proyecto existente de gestión de proyectos llamado activeCollab. El componente servidor utiliza PHP trabajando sobre un servidor web Apache y almacenando los datos en una base de datos MySQL. El cliente utiliza HTML, CSS, AJAX y Javascript corriendo en cualquiera de los navegadores modernos: Internet Explorer, Firefox, Safari u Opera.

Durante el desarrollo del proyecto, el trabajo se organizó como se describe a continuación: en primera instancia se trabajó en el relevamiento de los requerimientos del cliente[i], determinando el alcance de la implementación en el proyecto de grado así como la visión a largo plazo del proyecto OpenGoo. Luego se investigaron soluciones de código abierto que pudieran ser reutilizadas durante el proyecto de grado o en futuras versiones del producto. Por último, se implementó una primera versión de OpenGoo, tomando como base los requerimientos y conclusiones arribadas en la investigación.

El resultado fue un prototipo funcional, obtenido en relativamente poco tiempo. Esto se debió en gran medida a la reutilización de módulos y bibliotecas que fueron relevados en el estudio del estado del arte. En cuando a las expectativas del cliente, éste demostró estar muy satisfecho con el resultado alcanzado.

OpenGoo logró un cierto grado de notoriedad en Internet, lo que se evidenció tanto en diversas revisiones publicadas en *blogs* así como en correos alentadores de usuarios de la aplicación.

Palabras clave

Oficina Web, procesador de texto, hojas de cálculo, presentaciones, gestión de proyectos, web 2.0, código abierto.

[i] El cliente del proyecto fue el Ingeniero en computación Conrado Viña, director de Moove-IT, empresa dedicada a la capacitación y venta de servicios informáticos.

Tabla de contenido

1.Introducción

En los últimos años Internet ha sufrido una evolución que la ha llevado de ser una red de contenido semánticamente estático a una red de contenido semánticamente dinámico [5]. La mayor diferencia es que actualmente el contenido de las páginas es computado en oposición a almacenado. Esto permite que el contenido de los sitios sea definido por los usuarios [45]. Esta nueva "versión" de Internet es comúnmente llamada web 2.0.

Dentro de esta evolución, cabe destacar una tendencia que cobra fuerza: la modalidad de entrega de software como servicio (SaaS por sus siglas en inglés: *Software as a Service*). Esta implica que el vendedor provee el uso de un software como servicio [3] [34] [31]. En este marco han surgido cientos de pequeñas empresas ofreciendo todo tipo de servicios a través de Internet. Este proyecto de grado fue concebido en este marco temporal.

El proyecto de grado fue propuesto por el Ingeniero en Computación Conrado Viña, quién planteó que su objetivo en el marco del proyecto OpenGoo era lograr definir la arquitectura de base para una *web office*, que incluya funcionalidades de manejo de proyectos y edición de documentos. Además de esta definición, buscaba tener versiones básicas de los tres editores más comunes: de texto, de hojas de cálculo y de presentaciones.

El concepto de *web office* (traducido como oficina web) hace referencia a un conjunto de aplicaciones web, que corren en un servidor y permiten al usuario crear, editar y compartir información [20]. Incluye funcionalidades para productividad, colaboración y publicación, además de versiones adaptadas de las aplicaciones típicas de una oficina de escritorio (como Microsoft Office).

Las pautas de trabajo implicaron crear una solución *open source*[ii] principalmente debido a que se pretendía formar una comunidad que desarrollara y mantuviera el producto. Esta decisión permitió aprovechar otras aplicaciones de código abierto, e incorporarlas de forma gratuita (y sin necesidad de cambiar la forma de licenciamiento). En el estudio del estado del arte, por ende, se hizo especial hincapié en la búsqueda de proyectos *open source* para ser integrados a OpenGoo.

Se terminó por lograr una definición de qué es lo que el cliente buscaba a largo plazo, y se acordó qué subconjunto de esa extensa lista de módulos debía estar implementado al final del proyecto de grado.

Las principales tareas del proyecto fueron el estudio del estado del arte, la planificación del producto a largo plazo y la gestión y desarrollo del proyecto OpenGoo. Esta última incluyó tareas de diseño, implementación y verificación apuntando a obtener un producto que se ajustara a lo especificado.

Con respecto al desarrollo, se utilizó como base de la arquitectura un gestor de proyectos de código abierto[iii] llamado activeCollab. A éste se le agregó un módulo de gestión de documentos más avanzado, se le habilitó la posibilidad de compartir documentos por usuario, se le incluyó un procesador de texto, un editor de hojas de cálculo y uno de presentaciones.

[ii] Los términos *open source* y código abierto fueron usados indistintamente a lo largo del proyecto.

[iii] activeCollab se discontinuó como solución de *open source*, volviéndose una solución comercial en Octubre de 2007.

Con respecto a la verificación, se planificó que la versión final sería verificada por un pequeño número de *beta testers*[iv] para obtener respuesta directamente de los usuarios del sistema. Además, los foros fueron fuente de retroalimentación por parte de varios usuarios que descargaron y utilizaron OpenGoo.

1.1 Organización del documento

Este documento está organizado en seis capítulos:

1. Introducción
2. Requerimientos
3. Estado del Arte
4. Arquitectura
5. Implementación
6. Conclusiones

El capítulo 1 sitúa al proyecto de grado en su contexto y resume brevemente el trabajo realizado. Además incluye la organización del documento.

El capítulo 2 describe los requerimientos relevados al cliente. Es importante para entender algunas decisiones tomadas a lo largo del proyecto.

El capítulo 3 describe el estudio del estado del arte y las tecnologías y proyectos relevados para ser reutilizados en la solución a implementar. Este capítulo no es fundamental para entender el trabajo realizado.

El capítulo 4 describe la arquitectura del producto.

El capítulo 5 describe el proceso de implementación llevado a cabo.

El capítulo 6 detalla las conclusiones del trabajo realizado.

[iv] En las secciones 5.3 y 5.4 del capítulo 5 se puede leer más sobre la metodología utilizada con los *beta testers*.

2.Requerimientos

En este capítulo se establecen los requerimientos relevados al cliente a lo largo del proceso.

La propuesta inicial para el proyecto de grado era realizar un procesador de texto web como parte de un proyecto de oficina web más amplio. Sin embargo, posteriormente se decidió orientar el proyecto a la definición de la arquitectura e implementación de una oficina en línea completa, ya que se concluyó que era un camino más adecuado para lograr un mejor producto.

Se orientó el proceso, entonces, a relevar los requerimientos para desarrollar una web *office*. Este proceso fue extenso y se llevó a cabo paralelamente al estudio del estado del arte. Incluso durante la implementación se continuaron agregando requerimientos que se presentan como posibles trabajos a futuro en la sección 6.1 del capítulo 6.

2.1 Requerimientos funcionales

A continuación se listan a grandes rasgos las funcionalidades requeridas por el cliente en forma de módulos de la oficina web, distinguiendo entre las que están comprendidas dentro del alcance del proyecto y las que no. La lista detallada de requerimientos funcionales se encuentra en el anexo 2.

2.1.1 Requerimientos funcionales incluidos en el alcance[v]

- Edición de documentos
- Edición de presentaciones
- Edición de hojas de cálculo
- Gestión de documentos y archivos
    - Navegación de documentos creados y archivos subidos
    - Organización de los mismos
- Gestión de proyectos
    - Manejo de tareas
    - Manejo de hitos
- Permisos por usuario y no únicamente por proyecto
- Soporte para múltiples usuarios

2.1.2 Requerimientos funcionales excluidos del alcance

En esta sección se describen brevemente los módulos excluidos del alcance. Por más

[v] Los requerimientos que se encuentran en el listado fueron considerados e incluidos en el alcance del proyecto. Sin embargo, el módulo de hojas de cálculo no es más que una prueba de concepto. Los editores de documentos y presentaciones son utilizables, pero tampoco pueden ser considerados una versión final y completa.

detalles se puede consultar el anexo 12.

- *Dashboard* – Página inicial desde donde se puede monitorear y acceder rápidamente a la funcionalidad más importante de OpenGoo.
- Cliente de correo – Módulo para recibir y enviar correo electrónicos.
- Calendario – Módulo donde agendar eventos y visualizarlos por mes, semana y día.
- Fotos – Módulo para organización de fotografías e imágenes en álbumes.
- *Content Management System* (CMS) – Módulo de administración de contenido, que habilite a los usuarios a generar contenido en OpenGoo.
- Wiki – Módulo para la administración y publicación en *wikis*.
- Foro – Módulo para la administración y publicación en foros.
- *Blog* – Módulo de administración de *blogs* y publicación de artículos y comentarios.
- *Customer Relationship Manager* (CRM) – Módulo para seguimiento de clientes y prospectos.
- Contabilidad – Módulo de contabilidad.
- *Enterprise Resource Planning* (ERP) – Módulo para la planificación de los recursos de la empresa.
- Conferencias web – Módulo que permite conferencias web entre los usuarios del sistema.
- Redes sociales – Módulo que implemente estándares y sirva para integrar OpenGoo a distintas redes sociales existentes.

2.2 Requerimientos no funcionales

El alcance del proyecto OpenGoo sobrepasaba el alcance de un proyecto de grado de acuerdo a los tutores[vi]. Sin embargo, se pretendía que el trabajo continuara después de la culminación del proyecto. Esto implicó una serie de requerimientos no funcionales:

- La arquitectura debía ser modular y fácilmente extensible,
- Tanto la arquitectura como el código fuente debían ser elegantes y estar bien documentados.

Al tratarse de un proyecto de código abierto que se pretende sea desarrollado por la comunidad *open source*:

- debe estar implementado en un entorno de desarrollo ampliamente adoptado por dicha comunidad,
- deben liberarse versiones frecuentemente para generar visitas al sitio.

El cliente sugirió que al evaluar el lenguaje de *back-end* se tuviese presente el costo de servidor, además de la popularidad.

[vi] Tanto los tutores, como el cliente y los estudiantes estuvieron de acuerdo que el proyecto OpenGoo en su totalidad superaba el alcance de un proyecto de grado.

Además, al estar tratando potencialmente con programadores de todo el mundo y dado que el inglés es el idioma más usado en Internet [13], el producto debió estar disponible en inglés así como su código fuente.

Algunos requerimientos no funcionales se desprenden de la intención de vender OpenGoo en modalidad SaaS a una gran cantidad de usuarios:

- escalabilidad
- confiabilidad
- robustez
- *performance*

Otros requerimientos no funcionales deseables:

- amigabilidad
- mantenibilidad
- reusabilidad
- portabilidad
- interoperabilidad

En el anexo 13 se pueden ver más detalles respecto a los requerimientos funcionales.

3.Estado del arte

En el presente capítulo se introduce el contexto en el cual se llevó a cabo este proyecto de grado. Se comienza profundizando en el concepto de web 2.0 y luego se mencionan las características de las aplicaciones de código abierto. Más adelante se puede encontrar detalles del relevamiento realizado sobre las herramientas que se ajustaron de mejor manera a los intereses de OpenGoo. Se complementa este relevamiento con una breve introducción a las tecnologías web relevadas. Para finalizar, se justifican las tecnologías seleccionadas para el proyecto.

El relevamiento de soluciones es la sección más extensa del informe. Hace foco en la funcionalidad mínima para considerar una aplicación como integrable y también en la presentación de las principales características de las soluciones y formatos relevados.

3.1 Contexto

La revisión del estado del arte de este proyecto fue realizada entre los meses de abril y agosto de 2007. Después de este período se realizó un monitoreo de noticias relativas a las soluciones encontradas.

3.1.1 Web 2.0

De acuerdo a Tim O'Reilly[vii] el concepto de la web 2.0 está en pleno auge. Éste fue introducido por Dale Dougherty[viii] y su definición no está del todo clara ni estandarizada. Dentro del mundo web se define a las web 2.0 como "la representación de la evolución de las aplicaciones tradicionales hacia aplicaciones web enfocadas al usuario final. La web 2.0 es una actitud y no precisamente una tecnología" [43].

Esta definición es interesante porque resume en dos frases la esencia de la web 2.0. La primera parte habla de la evolución de las aplicaciones tradicionales a la web, marcando un contraste con la web 1.0. De acuerdo al artículo de O'Reilly, en la web 1.0 los sitios rara vez proveían servicios a los usuarios, sino que brindaban contenido principalmente estático. Por el contrario, la web 2.0 se caracteriza por aplicaciones que interactúan con el usuario y están "enfocadas en el usuario". Es muy frecuente confundir el concepto de web 2.0 con tecnologías o herramientas específicas, sin embargo, la segunda frase de esta definición deja en evidencia que no se trata de tecnología, sino de una forma de hacer las cosas.

El momento y las cifras que se manejan en las transacciones de venta de las empresas web 2.0 han llamado la atención de muchos emprendedores de todo el mundo. Es por esto que llama la atención la cantidad de organizaciones de agrupación de emprendedores que ha surgido en los últimos años[ix]. De todas formas, el epicentro de este movimiento mundial sigue siendo Silicon Valley en California, EEUU.

[vii] Tim O'Reilly es fundador y director ejecutivo de O'Reilly Media, una de las editoriales más importantes de textos de computación. Además es conferencista y activista a favor de estándares libres.

[viii] Dale Dougherty es co-fundador de O'Reilly Media junto con Tim O'Reilly. También junto con O'Reilly fue fundador de GNN en 1993, un portal que fue pionero en la venta de publicidad en la web.

3.1.2 Proyectos de código abierto

Los proyectos de código abierto se basan en el principio de que el software debe ser accesible para todos, y modificable por quien lo desee [8]. Éstos son gratuitos para descargar y utilizar, y sus licencias permiten su modificación o reutilización libremente siempre que se mantenga la licencia. Muchos de estos proyectos existen gracias al apoyo que la comunidad les brinda, pero detrás de otros está el sustento de empresas que los financian (como MySql [23] y JBoss [14] entre tantos otros).

El hecho de que OpenGoo sea un proyecto de código abierto permite aprovechar la vasta oferta de herramientas y bibliotecas *open source* disponibles para ser reutilizadas en el producto y de esta forma ahorrar horas de desarrollo. Debido al ambicioso alcance del proyecto OpenGoo, resultó muy redituable dedicar una buena parte del proyecto a estudiar herramientas de código abierto. Este relevamiento sería aprovechado cuando se continúe el desarrollo de OpenGoo más allá del proyecto de grado.

Las aplicaciones de código abierto tienen otra característica, y es que por su naturaleza resulta mucho más fácil armar una comunidad alrededor de ellas [4]. Aprovechando esta ventaja, se decidió que durante el proyecto de grado se dedicaría una parte de los recursos a abrir posibilidades de comunicación con esta comunidad, intentando lograr el apoyo de desarrolladores independientes de la comunidad *open source*. Esto implicaba desarrollar vías de comunicación que permitieran a los posibles contribuyentes sentirse cómodos, y respaldados desde el núcleo del equipo, así como aplicar metodologías de trabajo adecuadas para trabajo en grupos separados geográficamente. Se estimó que resultaría una buena inversión solicitar ayuda a la comunidad desde el principio, con el objetivo de materializar este apoyo en algo concreto para cuando el proyecto estuviese terminando.

3.2 Estudio de herramientas existentes

El primer paso antes de comenzar el relevamiento sobre herramientas existentes fue definir las características deseables de una herramienta que se integraría a OpenGoo. Fue necesario realizarlo en esta etapa del proceso debido a que se debía acotar el universo de herramientas a estudiar. De esta forma se pretendía tener pautas para priorizar el estudio de las herramientas que resultarían más útiles en el producto final.

La metodología implicó definir una serie de criterios a través de los cuales evaluar las aplicaciones halladas, y de acuerdo a ellos asignarles categorías. Las categorías abarcaron conceptos tan disímiles como las tecnologías utilizadas, el licenciamiento y el nivel de actividad de la comunidad.

Resumidamente, se busca una aplicación web, gratuita, de código abierto con una comunidad activa. Además, idealmente estaría desarrollada en PHP, HTML y AJAX[x] de forma de soportar formatos abiertos. En particular PHP fue seleccionado como lenguaje de *back-end* debido a su popularidad, estabilidad y la gran cantidad de desarrolladores que lo utilizan. La lista completa de criterios y los fundamentos detallados de las decisiones pueden encontrarse en el anexo 8.

[ix] Un claro ejemplo es el Open Coffee Club (http://www.opencoffeeclub.org) que fue fundado en Londres como una organización que vincule emprendedores del rubro informático con inversores y en menos de un año se expandió a lo largo y ancho de todos los continentes.

[x] Las siglas se encuentran definidas en el glosario

Los criterios fueron definidos como una guía, pero durante el relevamiento del estado del arte no se limitó el estudio exclusivamente a las soluciones que cumplían con todos ellos. Además de que no abundan las soluciones que cumplen con todas las condiciones antes mencionadas, resultaba importante considerar funcionalidades o interfaces de otro tipo de editores (como la nueva interfaz de barras de herramientas de Microsoft Word 2007 por ejemplo).

Igualmente los criterios fueron necesarios para definir las posibles aplicaciones a integrar a OpenGoo. Sin embargo, a la hora de evaluar soluciones se estudiaron otras que nunca se pensó en incluir en el proyecto. Éstas fueron incluidas en el estudio porque se consideró que tenían características destacables y se consideraban interesantes por distintos motivos.

3.2.1 Procesadores de Texto

El anexo 1 es el resultado del relevamiento de los procesadores de texto, con foco en los basados en Internet y que soportan el formato ODF. Este anexo contiene un extenso informe dónde se evalúan con distinto grado de profundidad más de quince procesadores de texto. El análisis es de abril de 2007 e incluye todos los procesadores encontrados a través del buscador Google, entre otras fuentes.

Las características básicas de un procesador de texto son la inserción, modificación y borrado de texto, junto con el estilizado del mismo. Los editores más sofisticados poseen, además, una gran cantidad de opciones de formateo avanzado. Al analizar cuales incluir, el grupo de desarrollo (integrado por el cliente y los estudiantes) elaboró una lista de funcionalidades mínimas deseables para el producto.

El criterio utilizado fue el de tomar Google Docs y Microsoft Word 2003 como base, seleccionando las funcionalidades imprescindibles. Estas fueron –a criterio de los estudiantes- las mínimas indispensables para que el editor sea utilizable y atractivo para los usuarios que no requieren funcionalidad avanzada.

Funcionalidad mínima

Las funcionalidades que se creyó necesario incluir en el procesador de texto son las básicas de edición y formateo de texto, así como el manejo de tablas y la navegación por teclado. El listado completo de funcionalidad se puede encontrar en el anexo 14.

El listado del anexo 14 pretende ser el mínimo indispensable de requerimientos funcionales. Sin embargo, en el relevamiento del estado del arte se observó que la gran mayoría de los editores de texto proporcionaban esta funcionalidad mínima y muchas otras que se suponen conocidas por el lector (ejemplos son los estilos, inserción de imágenes, corrector automático, inserción de encabezados y pies de página, etc.). La decisión de cuál editor utilizar no se iba a tomar basándose únicamente en un listado de funcionalidad. Se habían definido otros factores tanto o más importantes.

Soluciones relevadas

Del relevamiento del estado del arte surgieron varias soluciones que cumplían con los criterios definidos anteriormente. A partir de dicho informe se confeccionó la sección 15.1 del anexo 15, que contiene las aplicaciones que el equipo consideró más importantes,

junto con sus características más relevantes. Las más destacadas son Microsoft Word que es el procesador de escritorio más utilizado, OpenOffice Writer que es la alternativa *open source* mas utilizada, Google Docs que es una aplicación web nueva y popular, y FCKEditor que consta únicamente de un editor sin *back-end*.

Sobre el estudio completo se destaca que, a pesar de la gran cantidad de editores de código abierto que se encontraron, la mayoría están orientados a la creación de páginas HTML. Como consecuencia, no permiten compartir documentos entre distintos usuarios, a diferencia de editores de código propietario -como Google Docs [11]- que sí lo permiten. Por lo tanto, se está en condiciones de afirmar que a mayo de 2007 existía una carencia en la oferta de software de código abierto en cuanto a aplicaciones de oficina web.

A pesar de esto, y dada la semejanza entre la interfaz gráfica de un editor de páginas web con la de un editor de texto de una suite de oficina, resultaba posible reutilizar los editores de HTML. El camino a seguir implicaba tomar el cliente de un editor HTML y crear un editor de texto más completo, intentando imitar y mejorar los editores comerciales más avanzados.

Formatos relevados

Los formatos de documentos de texto relevados se pueden encontrar en el anexo 16. El mismo anexo incluye una breve descripción de formatos como DOC, ODT y HTML, entre otros.

3.2.2 Hojas de cálculo

El procedimiento seguido para investigar las hojas de cálculo fue el mismo utilizado para los editores de texto. Las características deseables del producto y los requerimientos no funcionales también son compartidos con los editores de texto.

Funcionalidad mínima

Para determinar la funcionalidad mínima se siguió la misma metodología que para los editores de texto, agregando a aquella lista las funcionalidades de inserción y borrado de celdas, así como funciones y operadores matemáticos. La lista completa con las características mínimas se puede encontrar en el anexo 14. Se buscó que esta lista contenga sólo funcionalidad simple como edición de texto, posibilidades mínimas de formateo y las fórmulas más básicas. No se incluyó funcionalidad avanzada como pueden ser fórmulas complejas, uso de macros y opciones de formateo avanzadas.

La cantidad de editores de hojas de cálculo resulto ser mucho menor que la de editores de texto. El fundamento encontrado para esto es que una gran cantidad de los editores de texto relevados son en realidad editores de HTML. Dichos editores son utilizados principalmente para gestionar el contenido de sitios dinámicamente. Sin embargo, no es tan frecuente encontrar sitios web que contengan hojas de cálculo. Este hecho, sumado a que el desarrollo de un módulo de hojas de cálculo es una tarea más exigente que el desarrollo de un editor HTML, hacen que resulten más escasas las herramientas para editar las primeras.

Soluciones relevadas

El anexo 15 contiene el relevamiento de los editores de hojas de cálculo. Entre ellos se destacan: Microsoft Excel el editor de hojas de cálculo de escritorio más utilizado; OpenOffice Calc la alternativa *open source* más utilizada; Google Spreadsheets que es de naturaleza web y goza de popularidad y TrimSpreadsheet que es pobre en interfaz, usabilidad y funcionalidad pero –al igual que FCKEditor- se trata de una aplicación cliente sin *back-end*.

Desde mayo de 2007 hasta febrero de 2008 se constató que la cantidad de editores estuvo cerca de duplicarse. En febrero de 2008 se congeló la lista y se dejaron de agregar las nuevas soluciones que surgieron. Este dato constituye una prueba de que nuestro proyecto de grado se enmarca en un momento de mucho dinamismo para las aplicaciones web 2.0 y más específicamente las web 2.0 office [19].

Formatos relevados

Los formatos de hoja de cálculo relevados fueron ODS, HTML y XLS. La descripción de los mismos se incluye en el anexo 16, habiendo sido seleccionados por considerarse los más utilizados [6] [46].

3.2.3 Editores de presentaciones

Las características deseables y los requerimientos no funcionales de los editores de presentaciones son compartidas con los otros dos editores.

Funcionalidad mínima

En lo que respecta a requerimientos funcionales, se siguió un procedimiento similar al aplicado para textos y hojas de cálculo. La funcionalidad requerida también resultó similar a la de los procesadores de texto, agregando cajas de texto e imágenes desplazables y la posibilidad de ver la presentación en pantalla completa. El listado completo se encuentra en el anexo 14. Nuevamente la funcionalidad mínima requerida incluyó únicamente comandos de edición básicos que permitieran utilizar el editor. Se descartó la funcionalidad compleja como transiciones con efectos, animaciones e incluso la inclusión de herramientas de dibujo.

La búsqueda de un editor de presentaciones resultó considerablemente más difícil que la de editores de texto y planillas. La cantidad de editores de presentaciones era notoriamente menor. Por este motivo al iniciar el estudio del estado del arte las principales soluciones que se usaron como referencia eran soluciones de escritorio. Las principales empresas del rubro aún no habían lanzado sus editores de presentaciones web, y algunas de ellas como Google lo lanzaron casi simultáneamente con OpenGoo [18].

Soluciones relevadas

Las soluciones relevadas se encuentran listadas en el anexo 15, e incluyen a Microsoft Powerpoint, OpenOffice Impress y Google Presentations. En su mayoría se trata de

aplicaciones de escritorio existentes desde antes de marzo de 2007. Las soluciones web comenzaron a aparecer a fines del año 2007. En abril de 2007 OpenOffice era el único editor de presentaciones de código abierto disponible en el mercado. Las otras aplicaciones relevadas eran software propietario y ninguna estaba preparada para ser usada como editor de presentaciones en un proyecto como OpenGoo. Por esta razón se empezó a manejar el hecho de que para cumplir con el requerimiento de incluir un editor de presentaciones, este debía ser desarrollado especialmente.

Formatos relevados

Resultaba importante definir un formato a soportar en el caso de que se decidiera desarrollar un nuevo editor de presentaciones web como parte de este proyecto de grado.
Los formatos candidatos fueron dos, luego de descartar los formatos propietarios y otros formatos complejos (como el de OpenOffice). Uno de ellos se llama S5 [21] (*a Simple Standards-Based Slide Show System*) y resultó muy adecuado para el proyecto porque contempla sus necesidades: es un formato abierto basado en HTML con un visualizador de diapositivas de código abierto ya implementado (hecho con Javascript y hojas de estilo).
Del proyecto S5 se desprenden varios otros proyectos [28] [29] [30] [37], evidencia de que S5 es apreciado y utilizado por la comunidad, convirtiéndose en una buena opción a integrar OpenGoo. Como ya fue mencionado, no existía un editor compatible de código abierto, razón por la cual de querer usar el formato, se debía implementar el editor.
Por otro lado, se encontró SMIL (Synchronized Multimedia Integration Language). Este lenguaje estandarizado tiene varias características similares a S5 en cuanto a su apertura, pero la gran ventaja es que ofrece más riqueza gráfica de la que S5 provee en su versión pura. Sin embargo, y a pesar de que SMIL es soportado por el navegador Internet Explorer, Mozilla Firefox ni siquiera contempla en su hoja de ruta añadirlo. Por este motivo el formato fue descartado, ya que Mozilla Firefox es el segundo navegador más utilizado en marzo de 2008.

3.2.4 Oficinas web

En las secciones anteriores se mencionaron las oficinas web más destacadas al mencionar sus editores. En el anexo 17 se puede encontrar un análisis más detallado de cada una de ellas como conjunto. De este análisis se destacan Zimbra y activeCollab como candidatos para ser utilizados en OpenGoo, ya que son las únicas soluciones de código abierto.

El estudio del estado del arte debía ser acotado en el tiempo, y fue debido a esta cota que el estudio de oficinas web no es más extenso. Básicamente la metodología utilizada para seleccionar los productos a investigar fue enumerar todas las suites que se encontraron y ordenarlas según importancia y relevancia al proyecto.

Luego se asignó un determinado período para este estudio y llegado el fin de este período se verificó una hipótesis que se venía manejando: hay muchas soluciones parciales al problema que OpenGoo pretende abordar, pero no hay soluciones completas, *open source* y que utilicen ampliamente estándares abiertos. Así se dio por confirmada la necesidad de implementar el producto.

3.3 Estudio sobre tecnologías web

De la sección 2.2 se desprenden algunos lineamientos que fueron tomados en cuenta a la hora de seleccionar las tecnologías web. Dentro de estos, lo más importante fue utilizar tecnologías y estándares abiertos, tomando decisiones que permitieran llegar a formar una comunidad que apoye el proyecto.

A la hora de evaluar tecnologías, el trabajo fue dividido en tres etapas: la primera consistiría en una búsqueda de candidatos, luego una preselección y finalmente un análisis minucioso para terminar con un orden de preferencia.

La búsqueda de candidatos se subdividió en dos partes, primero una lluvia de ideas de tecnologías conocidas y luego una búsqueda superficial en Internet. La búsqueda fue superficial debido a que se buscaba tecnologías conocidas y de uso masivo, con el objetivo de que la comunidad las conociera. Se consideró que si la tecnología no podía ser encontrada fácilmente (durante esta "búsqueda superficial"), ésta no era lo suficientemente popular. A pesar de ser un criterio subjetivo, tuvo como ventaja que ahorró mucho tiempo en este punto del estudio del estado del arte.

Al final de esta primera parte se definieron cuatro grandes rubros en los que se clasificaron las tecnologías encontradas: tecnologías de *front-end*, tecnologías de *back-end*, motores de base de datos y librerías accesorias.

Las herramientas que resultaron de esta búsqueda de candidatos fueron las siguientes:
Tecnologías de *front-end*: HTML/XHTML, Javascript, CSS, Java, Frames/iFrames, AJAX, Flash, Silverlight, SVG
Tecnologías de *back-end*: PHP, ASP, ASPx, JSP, Ruby on Rails
Motor de base de datos: MySql, Microsoft SQLServer, Posgre, Oracle
Bibliotecas: ExtJS, Openrico, Jquery, Prototype, Scriptaculous, qooxdoo

En el anexo 18 se puede encontrar los detalles de los relevamientos de tecnologías de *front-end*, *back-end* y motores de base de datos relacionales. Por otro lado las bibliotecas Javascript se analizan en el anexo 19. En la sección 3.4 se encuentran las conclusiones de estos relevamientos.

3.4 Tecnologías seleccionadas

A continuación se presentan las conclusiones y determinaciones que se desprenden del relevamiento del estado del arte y de los requerimientos planteados en el capítulo 2.

3.4.1Tecnologías de *front-end*

Las tecnologías cerradas y propietarias no son compatibles con la forma de comercialización que se planifica utilizar para OpenGoo. Además la necesidad de instalación de clientes complica la portabilidad a nuevas plataformas. Es por estos motivos que Flash, Silverlight y Java[xi] fueron descartados. Por otro lado, SVG fue descartado por no ser compatible con el navegador Internet Explorer.

Evaluando las tecnologías restantes, se definió que la mejor solución combinaría HTML,

[xi] Java no es una tecnología cerrada, pero necesita de la instalación de un cliente.

Javascript, AJAX y hojas de estilo CSS. Además, estas son las tecnologías que utilizan la mayoría de las aplicaciones web 2.0. Por otro lado, se dejó abierta la posibilidad de utilizar *frames* a pesar de sus desventajas [25].

3.4.2 Tecnologías de back-end

Al evaluar las tecnologías a utilizar en el *back-end* se tuvo especialmente en consideración que era necesario consolidar una comunidad de desarrolladores para OpenGoo. Por este motivo se debía elegir un lenguaje popular y estable.

Para medir la popularidad de los posibles lenguajes de *back-end* en la comunidad *open source* nos basamos en estadísticas sobre el uso de lenguajes en Sourceforge.net [xii] [xiii] [35]. El lenguaje web más usado es PHP con 13% [16] de los proyectos, aunque por encima de PHP se encuentra Java con 18% que también es usado como lenguaje de *back-end* para aplicaciones web. Dado que ese porcentaje abarca también las aplicaciones de escritorio su porcentaje para aplicaciones web es menor.

Los costos de mantenimiento tanto de Java como de PHP se verificaron con un rápido relevamiento de proveedores de hospedaje de sitios web remotos (*hosting*). Se comprobó que PHP es la más común de las tecnologías mencionadas. Además los proveedores que ofrecen esta tecnología lo hacen a un precio mucho menor que los que ofrecen Java o ASPX [7]. La popularidad y el costo de mantenimiento hicieron que PHP fuera el lenguaje más adecuado para el proyecto.

Existen plataformas para aplicaciones web que permiten desarrollar tanto el *front-end* como el *back-end* bajo un mismo lenguaje. OpenLaszlo [26] y Haxe [12] son dos ejemplos y cada uno define su propio lenguaje. OpenLaszlo ofrece un lenguaje de etiquetas basado en XML mientras que Haxe ofrece un lenguaje de scripting similar a Javascript pero fuertemente tipado. Ambos pueden compilar a Flash o DHTML (HTML Dinámico) con soporte para AJAX. El inconveniente de ambas plataformas es que, por tratarse de tecnologías muy nuevas, aún no cuentan con una comunidad de desarrolladores notoria.

3.4.3 Motor de Base de Datos

En cuanto al motor de base de datos, la decisión fue simple: se buscaba compatibilidad inicial con MySQL preferentemente o PostgreSQL en su defecto, que son los dos motores de base de datos gratuitos y de código abierto más utilizados. Particularmente en un relevamiento informal realizado por *Database Journal*, MySQL es el que más se ofrece en proveedores de *hosting [10]*. Con respecto a la base de datos se manejó el requerimiento que el sistema permitiera armar conectores para otros sistemas de bases de datos como SqlServer y Oracle.

[xii] SourceForge.net es el proveedor de *host* para proyectos open source más grande, con más de 173.000 proyectos open source y más de 1.800.000 usuarios

[xiii] Los datos son del 2006 ya que desde ese año SourceForge.net no sigue ofreciendo las estadísticas. Sin embargo se puede extrapolar los datos que se tiene desde el 2000 al 2006 hacia el 2007 obteniendo una aproximación aceptable.

3.4.4 Bibliotecas para el *front-end*

La selección de bibliotecas para el *front-end* se basó en dos criterios principalmente: funcionalidad y estética de los controles de interfaz gráfica. Además se consideraron positivamente la documentación, ejemplos y comprensibilidad del código de la biblioteca.

En total se relevaron más de 20 bibliotecas, quedando resumidas las características más importantes en el anexo 19. Del relevamiento surgió que los candidatos más importantes fueron ExtJS y Qooxdoo. Ambas se caracterizan por proveer las siguientes funcionalidades: animaciones, efectos, manejo prolijo de eventos, funcionalidad de interacción con el servidor por AJAX y variedad de controles de interfaz gráfica. Sin embargo, ExtJS fue considerada mejor candidata debido a la documentación abundante (que incluye la especificación de la API), las guías paso a paso y la gran cantidad de ejemplos completos.

Para aportar en la comparación de ambas bibliotecas se realizó un análisis de popularidad de ambas bibliotecas utilizando criterios que se usan frecuentemente para comparar popularidad en el ambiente web[xiv]. El resultado de este análisis se puede encontrar en el anexo 19, arribando a la conclusión de que ExtJS es más popular que Qooxdoo.

Siendo ExtJS la biblioteca de Javascript con más funcionalidad, más facilidades de desarrollo (en forma de manuales, guías y ejemplos) y un buen nivel de popularidad, se decidió que era la más indicada para incorporar proyecto OpenGoo.

[xiv] Se trata de criterios seleccionados arbitrariamente, pero que sirven para evaluar rápidamente la popularidad de determinados sitios Web.

4.Arquitectura

En este capítulo se describe la arquitectura seleccionada y se explican las decisiones tomadas respecto a las tecnologías utilizadas. Éstas últimas se basan fuertemente en el relevamiento del estado del arte que se puede encontrar en el capítulo 3 y los anexos complementarios. Además se incluye una descripción de los módulos ya implementados, así como las decisiones más importantes sobre Slimey, el módulo de edición de presentaciones.

4.1 Descripción de la arquitectura

La arquitectura de OpenGoo se puede dividir en dos partes, como se puede apreciar en la figura 4.1. Por un lado está el núcleo, que provee la mayor parte de la funcionalidad de integración, mientras que por otro lado, hay seis módulos que lo complementan:

1. Módulo de Administración de la oficina
2. Módulo de Administración de documentos
3. Módulo de Edición de documentos
4. Módulo de Edición de planillas
5. Módulo de Edición de presentaciones
6. Módulo de Gestión de proyectos

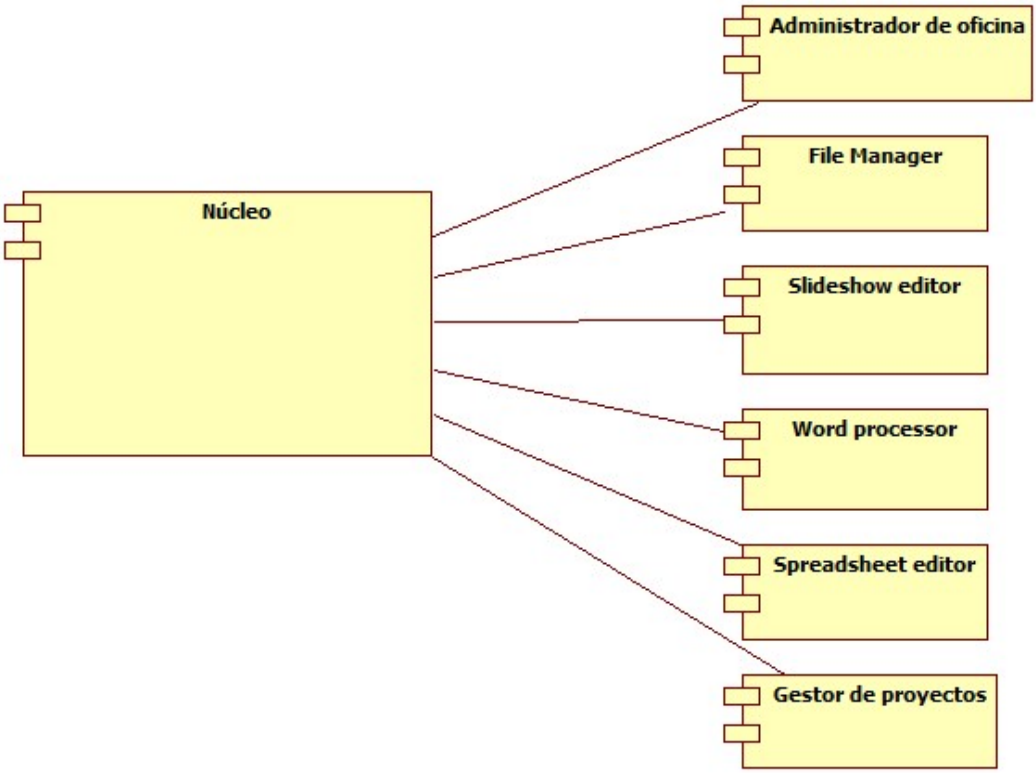

Figura 4. 1 - Componentes de OpenGoo

La arquitectura de OpenGoo se basa en el patrón de diseño MVC[xv].

La principal virtud de este patrón de diseño arquitectónico es que separa los datos de usuario, la interfaz gráfica y la lógica de control en tres componentes distintos. Éstos se interrelacionan como se puede apreciar en la figura 4.2.

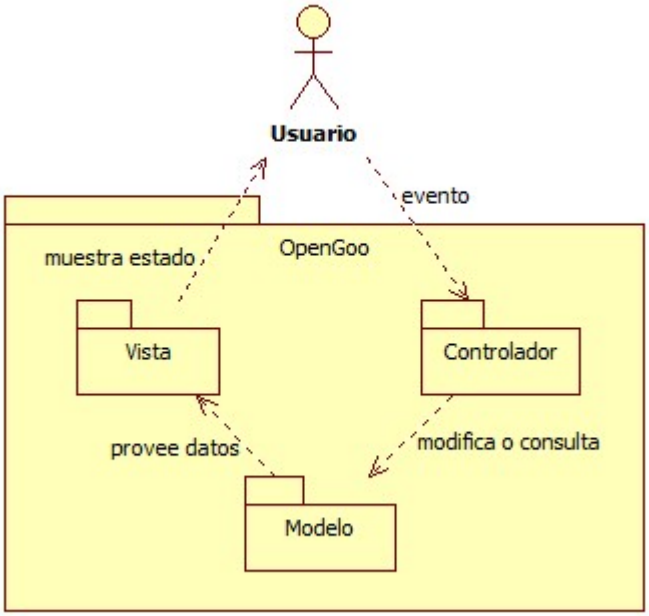

Figura 4.2 - Patrón MVC

Por tratarse de una aplicación web, el componente vista está compuesto principalmente por código HTML generado por *scripts* PHP. Básicamente cada página o sector de una página está relacionada directamente con una vista (que se encuentra dentro del componente vista). Las vistas están agrupadas por módulo y para el caso de los documentos, por ejemplo, existe un módulo llamado *files* que contiene las vistas: *add_file*, *list_files*, *file_details*, etc. Existen otros 19 módulos de vistas además del de de documentos.

Por otro lado se tiene el componente controlador. Este componente es el encargado de realizar llamados a la lógica del sistema para que, por ejemplo, se despliegue la lista correcta de documentos de determinado usuario. Las llamadas embebidas dentro de las vistas van dirigidas a los controladores. OpenGoo tiene dieciocho controladores, uno para manejar cada módulo definido para las vistas (con excepción de los módulos *menu* y *notifier*). El controlador atiende los eventos que provienen de la interfaz gráfica (por ejemplo una solicitud de listar archivos), los procesa accediendo al modelo y pudiendo realizar cambios en él para luego devolver el control a la vista para que despliegue una respuesta a los cambios.

Es así que la tarea de controladores y vistas se complementa con la del modelo. Es éste quien provee una representación en objetos de la información de la base de datos, para que los controladores y las vistas puedan hacer un uso amigable de los mismos. Cumple también un rol importante al realizar los chequeos de integridad de los datos y derivar datos que se desprender de otros. En la implementación de los modelos de OpenGoo, los objetos cachean en memoria los valores de sus atributos, evitando sucesivas llamadas

[xv] Del inglés Model View Controller, traducido como Modelo Vista Controlador

idénticas a la base de datos (ya que esas demoran los tiempos de respuesta innecesariamente). La cantidad de modelos que utiliza OpenGoo está directamente vinculada a la cantidad de tablas, ya que cada una de las 28 tablas de la base de datos se puede asociar con uno de los 36 modelos[xvi].

Si se desea profundizar en la arquitectura de OpenGoo, el anexo 9 contiene una descripción más detallada de los componentes. Además, en el mismo anexo, se pueden encontrar diagramas simplificados de la jerarquía de modelos.

4.2 Decisiones de diseño

En esta sección se describen brevemente las líneas de trabajo de implementación, detallando las decisiones tomadas y su justificación.

Se puede consultar el anexo 2 para encontrar un listado más exhaustivo de los requerimientos por módulo, y una explicación más detallada de los requerimientos no funcionales. El documento de casos de uso se encuentra en el anexo 5 y complementa el de requerimientos.

4.2.1 Núcleo de OpenGoo

Se plantearon dos posibilidades para el núcleo de OpenGoo. La primera consistía en desarrollarlo desde cero, mientras que la segunda contemplaba reutilizar algún proyecto de similares características. De los varios proyectos de código abierto que se estudiaron el que mejor se ajustó a las exigencias de OpenGoo fue el gestor de proyectos activeCollab. Se decidió tomarlo como base debido a la notoria reducción de tiempos de desarrollo que significaba reutilizarlo comparando con un desarrollo desde cero. Además fue un factor clave para cumplir con el requerimiento de contar con un prototipo a corto plazo.

La arquitectura de activeCollab fue la más elegante de todas las opciones consideradas, teniendo además la ventaja de incluir el módulo de gestión de proyectos que necesitaba OpenGoo, lo que reducía aún más el tiempo de desarrollo.

Entre la funcionalidad que venía incluida con dicho *framework [2]* y que no hubo que implementar se encuentra el sistema de autenticación, la administración de usuarios, la compatibilidad entre distintos navegadores y el soporte de varios lenguajes.

Otro factor relevante que confirmó la decisión fue la comunidad de usuarios formada alrededor de activeCollab. Por tratarse de una solución de código abierto exitosa existía una gran actividad en foros, donde se podía apreciar una cantidad de usuarios considerable[xvii]. Poco tiempo luego de adoptar activeCollab, el proyecto pasó a una licencia comercial, dejando a sus seguidores en busca de una alternativa. Lamentablemente, la mayoría de éstos se dirigió a otro proyecto *open source* basado en activeCollab llamado ProjectPier [27], ya que éste mantuvo el enfoque de gestión de proyectos, a diferencia de OpenGoo que apuntó a una oficina en línea.

[xvi] Existen 8 modelos que no tienen una tabla en la base datos asociada debido a que son entidades volátiles que no necesitan estar almacenadas.

[xvii] No se dispone de datos oficiales que respalden la afirmación, pero se considera la actividad en foros un reflejo de la cantidad de usuarios.

A pesar de todas las características positivas de activeCollab, se encontraron algunas limitaciones en su arquitectura, como el hecho de que contempla únicamente dos capas. Una capa adicional entre la capa de datos y la de presentación podría disminuir la cantidad de código repetido. Esta limitante, sin embargo, no dificulta en gran medida el trabajo con la arquitectura.

Otro punto negativo al respecto de la arquitectura fue la escasa documentación sobre la misma. Si bien todos los archivos de código fuente están extensamente documentados no se dispuso de una descripción más general de ella, lo que dificultó su entendimiento en las primeras instancias de implementación. Debido al abundante uso de patrones de diseño, sin embargo, se hizo sencillo familiarizarse con ella rápidamente.

La decisión de utilizar activeCollab trajo aparejada una decisión respecto a la utilización del sistema como SaaS. ActiveCollab fue diseñado para ser utilizado como una aplicación orientada a una empresa e instalable en el servidor de la misma. Desde su diseño fue concebida para almacenar la información de una sola empresa y sus clientes. Sin embargo el destino de OpenGoo es poder almacenar la información de varias empresas simultáneamente e integralmente, de forma de que puedan compartir información entre ellas.

Se debió decidir entre dos alternativas: mantener el diseño como fue concebido y al ofrecer el producto como SaaS ofrecer una nueva instancia del sistema para cada cliente; o modificar el sistema para que permita la interacción de varias empresas distintas bajo una misma base de datos. Las ventajas de cada alternativa se listan a continuación, llamando “monousuario” a la primera alternativa y “multiusuario” a la segunda:

1. Monousuario

    a. Posee mayor escalabilidad sobre el número de usuarios al venderse como servicio debido a que la instancia de cada usuario puede correr en un servidor distinto. Para distribuir la aplicación multiusuario en varios servidores hay que tener consideraciones desde la arquitectura misma del sistema.

    b. Es más adecuada para instalar en un servidor propio, como puede ser en una intranet de una empresa.

    c. La información se comparte por mecanismos externos a la base de datos, como puede ser *webservices*. Esto tiene como ventaja que se puede compartir la información entre usuarios que utilizan la modalidad SaaS y usuarios que utilizan una instalación en su propio servidor.

2. Multiusuario

    a. Es más apropiada para administrar si se va a vender como SaaS. Una aplicación monousuario contaría con varias instancias instaladas simultáneamente, por lo que las actualizaciones serían mucho más complejas.

    b. La interacción entre distintos usuarios es más fácil en una aplicación multiusuario, ya que todos los usuarios comparten la misma base de datos.

La opción que más se ajusta a las necesidades de OpenGoo es la de monousuario, porque

implica menos cambios de diseño y permite la comunicación entre sistemas instalados localmente y sistemas vendidos como SaaS, si bien esto no será implementado en el proyecto de grado.

4.2.2 Administrador de archivos

ActiveCollab incluía funcionalidades básicas para administrar archivos, sin embargo sus posibilidades eran muy reducidas y su usabilidad distaba de ser óptima. De acuerdo a los resultados del relevamiento de oficinas web, se definió que el administrador de archivos sería similar al de GoogleDocs. Se esperaba que permitiera listar y ordenar según atributos del archivo, por lo que se implementó esta funcionalidad en dos etapas. En una primera etapa se programó el *back-end* que proveía la funcionalidad junto con una interfaz en Javascript. Luego de incorporar la biblioteca ExtJS, se cambió la interfaz por una grilla de la biblioteca con rica funcionalidad de AJAX y estética mejorada.

4.2.3 Administrador de oficina

Una característica deseable para el núcleo era que no tuviese interfaz gráfica. Como consecuencia, algunas interfaces como las preferencias generales de los usuarios no tenían lugar en ningún otro módulo. Por este motivo se decidió que debía existir un módulo que reuniera toda la funcionalidad común. Este módulo fue bautizado administrador de oficina.
Al utilizar activeCollab como núcleo resultaba casi gratuita la tarea de integrarlo como administrador de la oficina. La principal razón es que en activeCollab no hay una clara separación entre el núcleo y la demás funcionalidad que ofrece.

4.2.4 Gestión de proyectos

Al igual que en el caso del administrador de oficina, el gestor de proyectos de activeCollab estaba integrado a la arquitectura y además era considerado muy bueno por la comunidad, tal como se puede verificar en los foros [1]. Por estos motivos se decidió conservarlo como parte de OpenGoo.

4.2.5 Procesador de texto

El procesador de texto más adecuado fue FCKEditor por todas las razones que se mencionan en el relevamiento del estado del arte. En particular fue seleccionado haciendo énfasis en su alto grado de compatibilidad, rica funcionalidad, gran cantidad de usuarios, fácil instalación y mantenimiento simple.

4.2.6 Editor de hojas de cálculo

Encontramos un único editor de código abierto en condiciones de ser reutilizado fácilmente: TrimSpreadsheets. Sin embargo, a diciembre de 2007 la funcionalidad de éste era básica y su interfaz gráfica muy rudimentaria. De todas formas se utilizó el editor TrimSpreadsheets sin modificaciones, dejando como trabajo a futuro la tarea de elaborar

un editor más completo y coherente con los criterios de usabilidad del proyecto.

4.2.7 Editor de presentaciones

En julio de 2007 no existían editores de presentaciones bajo licencia de código abierto. Lo más cercano era un visualizador de presentaciones S5. Ante la urgencia de tomar una determinación al respecto, el camino a seguir fue aprovechar el visualizador provisto por el creador del formato S5 e implementar un editor de diapositivas que utilizara dicho formato para almacenar las presentaciones.

En consecuencia se desarrolló Slimey, el primer editor de código abierto para presentaciones compatible con el formato S5. En ese momento el rubro de editores de presentaciones web era casi inexistente. Tal es así que ni siquiera Google había lanzado al mercado el propio. Al llegar al final del proyecto de grado existen varios editores con funcionalidades similares, pero ninguno de ellos de código abierto.

Debido a su gran tamaño, Slimey se describe en una sección independiente.

4.3 Slimey

El editor de presentaciones *Slimey* se mantuvo separado de OpenGoo, creando un proyecto *open source* independiente. De esta forma se aseguró que fuera fácilmente integrable a cualquier otro proyecto (además de *OpenGoo*). Se hizo de esta forma debido al requerimiento de que la arquitectura fuese modular. El modelo de integración del FCKEditor se utilizó como base, ya que fue concebido con el mismo fin.

Otro requerimiento fue desarrollar un marco en el cual Slimey se convierta en un proyecto autosuficiente y que crezca con la colaboración voluntaria de desarrolladores. Para esto se creó un sitio para intercambio de ideas y comentarios sobre el nuevo proyecto, además de que se usó SourceForge.net [35] para alojar el código fuente.

Slimey fue desarrollado para ser portable entre distintos navegadores. En particular se verificó su funcionamiento en las últimas versiones de Internet Explorer, Firefox, Safari y Opera[xviii]. Además, se agregó el requerimiento de que funcione correctamente en Internet Explorer 6, ya que a pesar de no ser la última versión, a julio de 2007 seguía siendo más usado que Internet Explorer 7 [39].

En cuanto a funcionalidad se definió que Slimey debía imitar a los editores más conocidos como Microsoft PowerPoint u OpenOffice Impress, de modo que a usuarios de dichos productos les resultara familiar. Esto implicó que el editor debía permitir ciertas funcionalidades típicas de aplicaciones de escritorio pero poco comunes en ambientes web, como la de arrastrar y soltar (*drag and drop*) elementos dentro de la diapositiva. La interfaz también se asemejaría a los productos antes mencionados en cuanto a la diagramación. A la izquierda tiene un panel con una lista de las diapositivas de la presentación en miniatura, dónde se elige la diapositiva a editar, y a la derecha un panel central donde se puede modificar la diapositiva seleccionada.

[xviii] Al 20 de marzo de 2008 las últimas versiones son: Internet Explorer 7, Mozilla Firefox 2, Apple Safari 3 y Opera 9.26.

4.3.1 Arquitectura de Slimey

Como Slimey es un proyecto de código abierto que pretende ser utilizado e incluso desarrollado por la comunidad, se puso especial énfasis en desarrollar una arquitectura prolija y escalable, siguiendo los principios de alta cohesión y bajo acoplamiento.

Al tratarse de una aplicación web con un alto componente de interacción en tiempo real con el usuario, fue necesario implementar la mayoría de Slimey en Javascript.

Resumidamente, la arquitectura está dividida en cuatro grandes componentes:

- Slimey: maneja la inclusión de Slimey a una aplicación web, así como el precargado de imágenes y otras tareas administrativas.

- SlimeyNavigation: maneja la navegación entre las diapositivas. Se encarga de mantener la información de cada diapositiva y cuál es la actual en edición. También se encarga de generar el contenido final en formato SLIM[xix] a partir de la información de todas las diapositivas.

- SlimeyEditor: maneja la edición de cada diapositiva. Ofrece una interfaz para ejecutar acciones sobe el contenido de la diapositiva, que permite deshacer acciones. Para esto utiliza un *stack* de deshacer y uno de rehacer, implementados por la clase SlimeyStack. También permite a otros componentes registrarse como oyentes de eventos, de forma tal que sean notificados cuando estos ocurren. Algunos ejemplos son cuando un nuevo elemento es seleccionado o cuando una acción es realizada sobre la diapositiva.

- SlimeyToolbar: administra las herramientas de edición. Se encarga de crear y desplegar al usuario las posibles herramientas que puede utilizar para modificar la diapositiva actual. Estas herramientas son implementadas por la clase SlimeyTool, que a su vez crea una SlimeyAction para afectar el contenido del editor.

Las relaciones entre estos componentes se pueden ver en la figura 4.3.

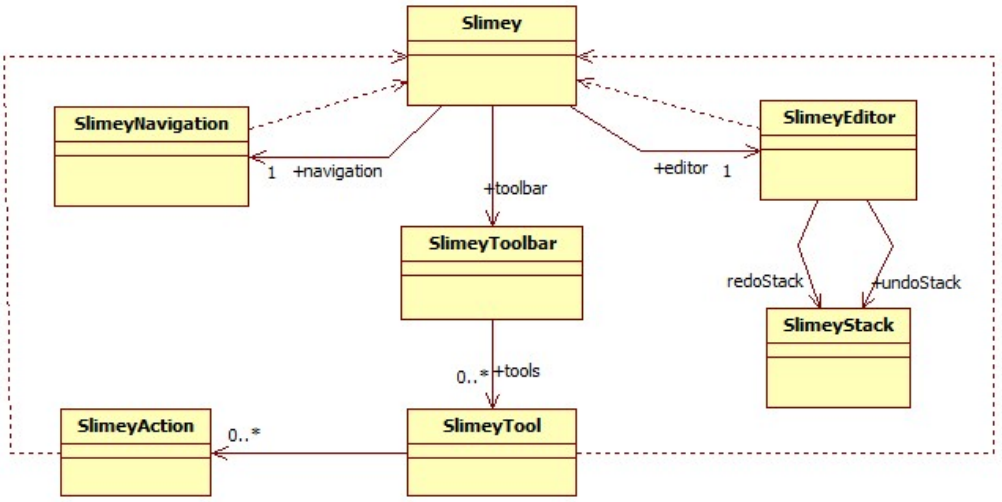

Figura 4.3 - Arquitectura global de Slimey

[xix] El formato SLIM se describe en la sección 4.3.2.

En el anexo 10 se puede encontrar información complementaria sobre la arquitectura y configuración de Slimey. Allí también se pueden encontrar otros diagramas de clases y una descripción de los archivos que componen este editor.

4.3.2 Formato de presentaciones

Inicialmente Slimey iba a ser un editor de presentaciones en formato S5. Sin embargo, el formato S5 no tiene una organización responsable de mantenerlo, actualizarlo y corregirlo, sino que fue puesto bajo dominio público por su creador, Eric Meyer [21] y desde el año 2006 no ha sido actualizado.

El formato S5 es muy limitado en sus características, mientras que Slimey apunta a ser un editor de presentaciones con funcionalidad avanzada que compita con las alternativas de escritorio. Para que el formato de las presentaciones no sea una limitante se decidió definir un formato abierto y extensible propio, pero que mantendría como premisa la compatibilidad con S5. La definición de este formato debería estar a cargo del proyecto Slimey. Este formato se nombró SLIM, que es un acrónimo para *Slideshow Microformat* (Microformato de Presentaciones), haciendo referencia al concepto de microformato definido por el sitio microformats.org[xx].

También está a cargo del proyecto Slimey desarrollar el motor visualizador del formato SLIM, que fue llamado Slime (acrónimo de SLIM *Engine*).

En la primera versión de Slimey, tanto el formato SLIM como el motor Slime tienen sólo pequeñas modificaciones respecto a S5 y su motor, por lo que su definición y desarrollo no significó un aumento considerable de esfuerzo. SLIM sí aportó la independencia y flexibilidad para que Slimey pueda evolucionar libremente. Idealmente, SLIM se convertirá en un estándar de formato de presentaciones web, avalado por una organización para la estandarización.

[xx] Microformats.org define a los microformatos como formatos simples, abiertos y basados en estándares ampliamente aceptados. Agrega que los microformatos deben ser diseñados para humanos (en oposición a diseñados para máquinas).

5.Implementación

Este capítulo describe las decisiones más relevantes en cuanto a la implementación de OpenGoo. Se comienza describiendo las grandes áreas en las que se debieron realizar tareas de implementación. Luego se explica el proceso de verificación y se enumeran los principales hallazgos del mismo.

5.1 Descripción de la implementación

En esta sección se describen las principales tareas de implementación realizadas sobre activeCollab, el proyecto en el cual está basado OpenGoo. En este proyecto se planificó integrar 4 nuevos módulos: el de gestión de documentos, el procesador de texto, editor de hojas de cálculo y editor de presentaciones. Además se buscó mejorar el *look and feel* de la interfaz y reformular el sistema de permisos.

El plazo manejado para la inclusión de esta funcionalidad fue de cuatro meses de desarrollo. Las tareas comenzaron intensivamente en agosto y terminaron en enero, con un mes de atraso por contratiempos no planificados.

Todas las funcionalidades planificadas fueron ponderadas de igual forma, y todas las partes consideraron que el alcance era más que razonable y que no debería haber razones para quitar funcionalidad del alcance. Esto es así debido a que la primera planificación completa dio por fecha de fin del proyecto a febrero de 2008, dejando un margen de al menos cuatro meses para retrasos.

A continuación se describe con más detalles los cambios realizados. Éstos se agrupan en las tres categorías que dan nombre a las próximas secciones.

5.1.1 Mejoras a la funcionalidad existente

Dentro de esta categoría se encuentran las tareas de implementación que no introducen nuevas funcionalidades al sistema. Se pasa a describirlas brevemente:

1. Se mejoró la interfaz de etiquetas. ActiveCollab tenía una interfaz de etiquetas en la que cada vez que se deseara etiquetar un elemento era necesario escribir una por una todas las etiquetas. Esto tenía dos desventajas de usabilidad: la primera es que hacía que el usuario pierda tiempo, y la otra es que era muy propenso a errores dactilográficos. Se modificó la interfaz de selección de etiquetas para que el usuario pudiese elegir cuales incluir entre las etiquetas existentes. La nueva interfaz no resultó suficientemente amigable, razón por la cuál se introdujo posteriormente capacidades de auto completado al escribir las etiquetas.

2. Se implementó la posibilidad de compartir archivos por fuera de los proyectos. Debido a que activeCollab fue desarrollado con foco en un administrador de proyectos, los documentos que podían ser accedidos por determinado usuario tenían que pertenecer a un proyecto y sólo podían ser accedidos por miembros del mismo. Además activeCollab tampoco permitía dar permisos diferenciales a distintos miembros del proyecto. Estas limitantes eran demasiado estrictas para el

concepto de *web office* que manejaba OpenGoo. Por este motivo se reestructuró el esquema de permisos, pasando de un sistema basado en proyectos a uno basado en usuarios. Ahora existe la posibilidad de poder asignar permisos a cada usuario independientemente del proyecto al que pertenezca. Complementariamente se comenzaron a manejar dos niveles de permisos para los documentos: un nivel básico que permite únicamente la lectura, mientras que el avanzado permite también la escritura.

Se realizaron modificaciones en la estructura de directorios, en el *back-end* y en la interfaz de activeCollab para que los editores funcionen correctamente.

5.1.2 Rediseño de la interfaz gráfica

La interfaz heredada de activeCollab era excesivamente simple, rudimentaria, poco atractiva y no estaba optimizada para garantizar una buena experiencia del usuario de acuerdo a los atributos de usabilidad definidos por Schneiderman [32]. En particular las principales carencias detectadas fueron que no se proveen suficientes guías a los usuarios, no hay formas rápidas de realizar tareas comunes (para usuarios frecuentes), ni existen opciones para deshacer.

La tarea de rediseño se dividió en tres tareas:

Rediseño del look and feel y diagramación globales

Esta tarea implicó cambiar la plantilla de todas las páginas de OpenGoo. Dentro de los cambios más relevantes se destaca que se trató de maximizar la superficie de trabajo, minimizando el área de la pantalla "desperdiciada" en elementos no productivos como espacios blancos.

Para visualizar los cambios, se puede observar la secuencia de figuras 5.1, 5.2 y 5.3. Los más notorios fueron:

- La ampliación de la superficie de trabajo.
- La modificación de la posición y forma del menú.
- El cambio en estilo de la interfaz.

Además se realizaron una gran cantidad de cambios menores en la interfaz para darle al proyecto una cara propia y distinguirlo de activeCollab. Dentro de estos se incluyen cambios de fuente, eliminación de ciertos íconos, cambios de determinadas proporciones, etc.

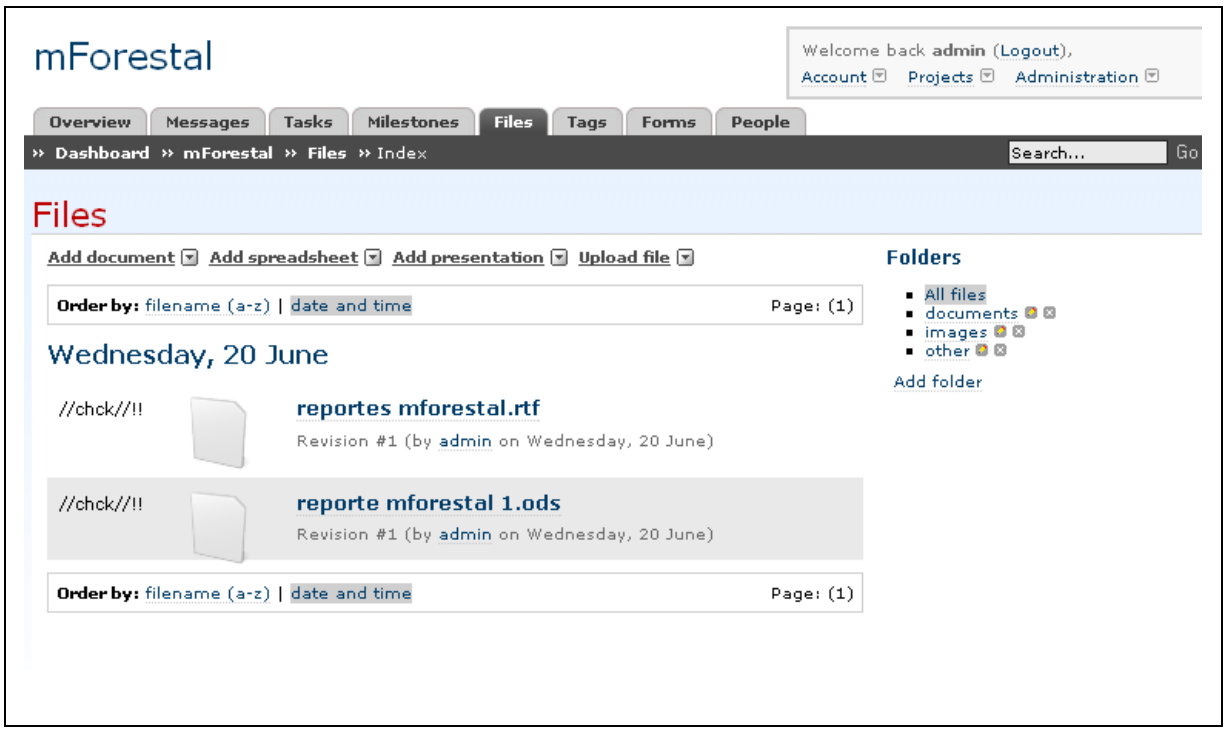

Figura 5.1 - Look and feel inicial

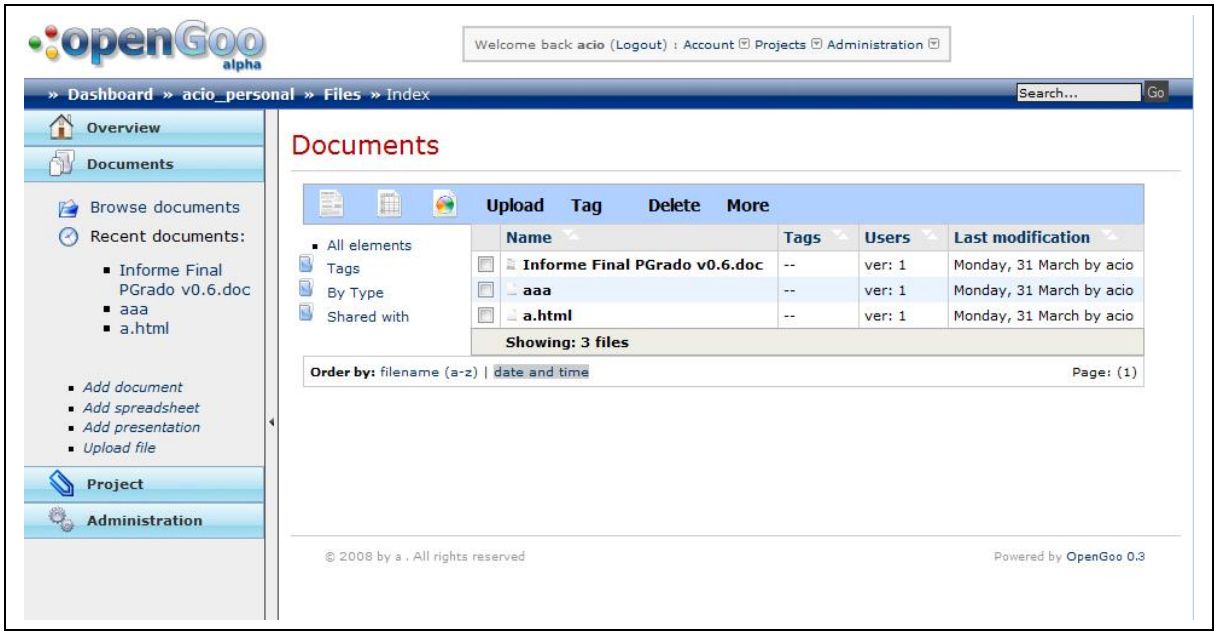

Figura 5.2 - Primera iteración sobre el Look and feel

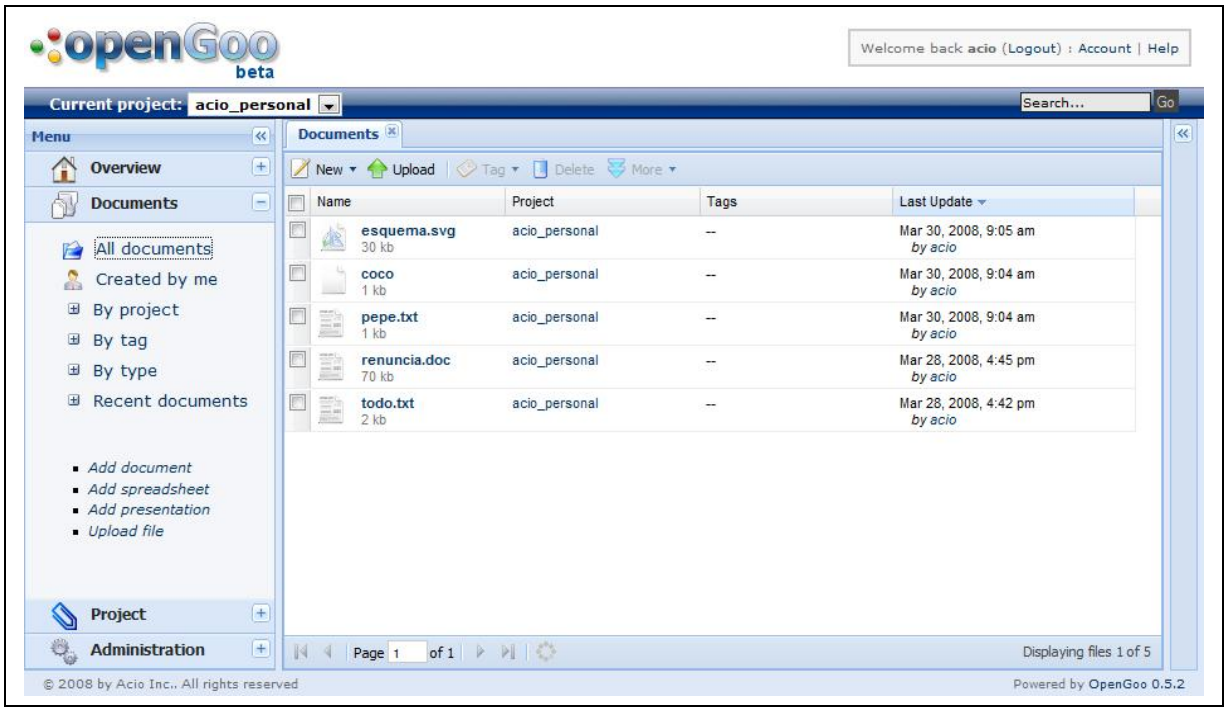

Figura 5.3 - Look and feel final, luego de integrar ExtJS

Rediseño del administrador de archivos

El administrador de archivos incluido en activeCollab contaba con escasa funcionalidad y no cumplía con varias pautas de usabilidad de las anteriormente mencionadas. Incluía demasiado texto informativo para cada archivo y tenía íconos muy grandes, lo que hacían que en un monitor de 17 pulgadas no se pudieran ver más de cuatro archivos del listado simultáneamente. Se notaba claramente un desaprovechamiento del espacio. Además, las funcionalidades provistas desde la interfaz del administrador de archivos eran limitadas, por esta razón la vista parecía un listado de archivos más que un administrador de archivos.
Tomando como modelo GoogleDocs fue que se decidió dividir la pantalla en tres partes:

1. Un árbol de navegación que se encontraría a la izquierda. Éste daría la posibilidad de acceder rápidamente a los documentos de acuerdo a su autor, el tipo de archivo, el proyecto al que pertenece y los usuarios con quien se comparte.

2. Una barra de herramientas en la parte superior, que desde el administrador de archivos diera acceso a toda la funcionalidad asociada a los archivos. Los botones permiten realizar las siguientes acciones:

    a. Crear nuevos archivos de texto, planillas o presentaciones utilizando los editores

    b. Subir archivos desde la PC

    c. Etiquetar archivos con acceso directo a los etiquetas usados más recientemente

    d. Crear etiquetas y asignarlos a un documento sin abandonar el administrador de archivos

e. Eliminar archivos

f. Ver propiedades

g. Ver opciones de compartir

h. Reproducir presentación (únicamente para presentaciones hechas con Slimey)

i. Descargar

j. Abrir el contenido del archivo con un editor compatible

3. Un panel principal donde se listan los archivos y se pueden ordenar por nombre o por fecha de modificación.

Además se realizaron cambios para que el administrador de archivos fuera más agradable visualmente, cambiándose colores y sustituyéndose íconos. Luego de la incorporación de la biblioteca ExtJS se iteró una vez más sobre el administrador de archivos modificando el aspecto gráfico.

La evolución del administrador de archivos en el tiempo puede verse en las figuras 5.4, 5.5 y 5.6.

Figura 5.4 - Administrador de archivos original

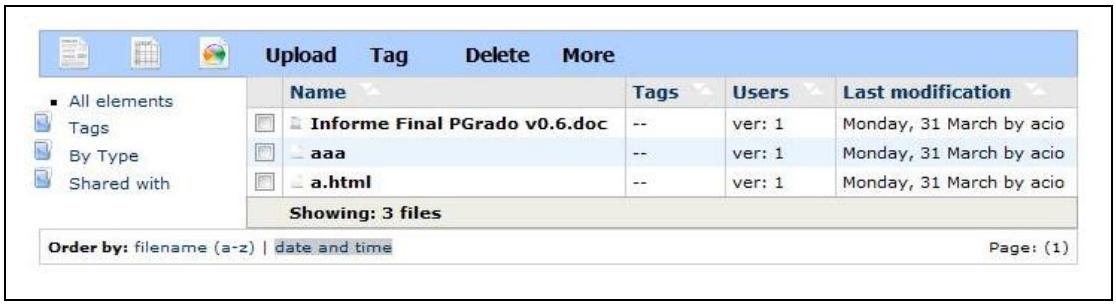

Figura 5.5 - Administrador de archivos más funcional

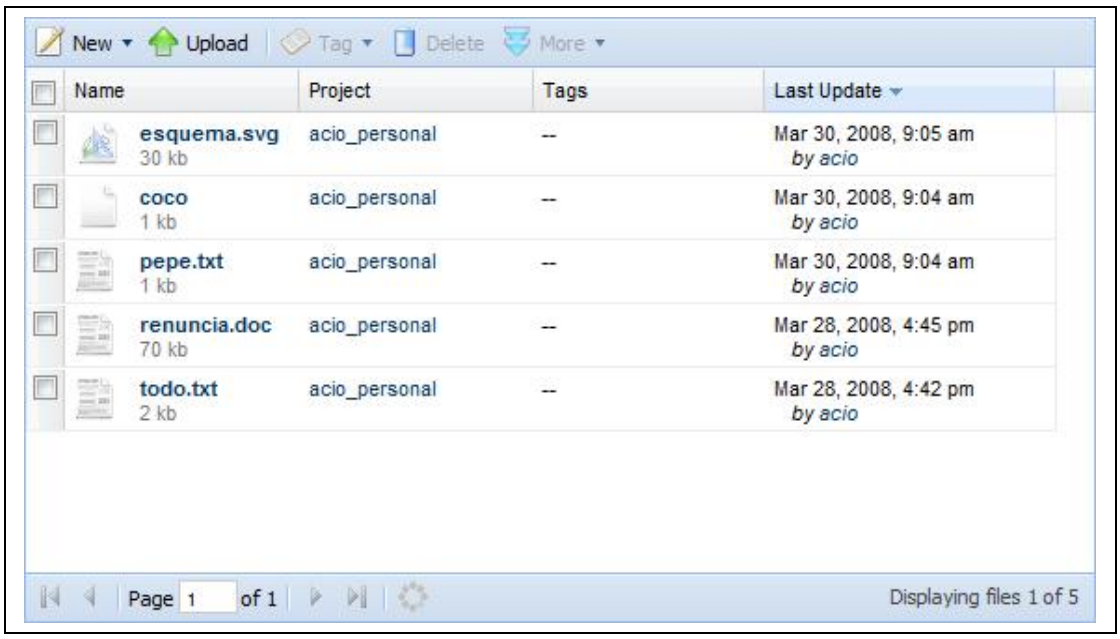

Figura 5.6 - Administrador de archivos mejorado estéticamente

5.1.3 Implementación de Slimey

Para la implementación de Slimey se manejaron diversas ideas[xxi] pero por motivos de tiempo no se logró implementar todas. Las que sí se implementaron son:

1. Inserción de cuadros de texto desplazables con el ratón
2. Inserción de imágenes, de forma que el origen de las imágenes pudiera ser configurado
3. Redimensionado de imágenes con el ratón
4. Inserción de listas numeradas y no numeradas
5. Edición del contenido en su lugar
6. Borrado de elementos
7. Inserción y borrado de diapositivas

[xxi] Los anexos 14 y 15 contienen esta información más detallada.

8. Deshacer y rehacer
9. Formateo de cuadros de texto (fuente, color, tamaño de fuente, negrita, cursiva y subrayado)
10. "Enviar al fondo" y "traer al frente" para poder ordenar los elementos en altura
11. Ver código fuente de lo generado
12. Previsualizar la presentación

5.2 Proceso de desarrollo

La metodología de implementación propuesta en los requerimientos implicaba iteraciones cortas y rápidas de forma de liberar versiones alfa frecuentemente. La ventaja principal era mantener el sitio del proyecto activo y así demostrar que no se trataba de un proyecto abandonado. Sin embargo, la desventaja era que exigía un mayor esfuerzo al equipo de desarrollo para cada liberación (porque implicaba iteraciones de verificación y control de calidad frecuentes).

Se realizaron liberaciones cada dos semanas en las épocas donde el trabajo de desarrollo fue más intenso. Esta frecuencia descendió al final del proyecto cuando el equipo pasó a dedicarse principalmente a tareas de documentación.

En total se realizaron cinco grandes liberaciones de OpenGoo previas a la entrega del proyecto de grado, y dos revisiones sobre la última versión para corrección de errores. El anexo 21 contiene un detalle de las características que se incorporaron a OpenGoo en cada versión. Básicamente, se trata de las funcionalidades que se describieron en la sección 5.1, con un mayor nivel de detalle y agrupadas de acuerdo a la versión en que se incorporaron.

5.3 Verificación

Para cada liberación alfa de OpenGoo se realizó una pequeña etapa de pruebas. Por tratarse de versiones alfa, no se buscaba invertir mucho tiempo en verificación, sino que la idea era demostrar movimiento en el proyecto frecuentemente e ir corrigiendo los errores que fuesen surgiendo a medida que sean descubiertos o reportados.

Sin embargo, luego de la liberación de la versión beta, se utilizó una metodología distinta. En lugar de realizar verificación exhaustiva utilizando metodología tradicional, se convocó a un grupo reducido de usuarios a que utilicen el sistema de forma guiada. Lo que se pretendió fue conocer las reacciones de usuarios que se enfrentaban por primera vez al sistema. Fue así debido a que se consideraba que esta información era mucho más valiosa que el hecho de conocer un listado de errores funcionales del sistema. Resultaba más importante conocer los problemas de usabilidad y amigabilidad, que los problemas de funcionamiento[xxii].

El equipo de desarrollo definió una lista compuesta por diez tareas para que cinco usuarios diferentes las completen. A cada uno de estos usuarios se le crearía un usuario

[xxii] Los tutores, el cliente y los estudiantes evaluaron que –dada la etapa de desarrollo en que se encontraba el proyecto- el tipo de verificación seleccionada era el más indicado.

del sistema y se le entregarían las instrucciones a seguir[xxiii]. Resumidamente, las instrucciones son una secuencia de uso en la que el usuario prueba el sistema editando documentos, etiquetándolos y creando tareas entre otras actividades.

Luego de entregadas las instrucciones, se iniciaba un cronómetro para controlar el tiempo total, y el estudiante observaba el comportamiento del usuario, sacando apuntes sobre lo que considerase más relevante. Se buscaba dejar registradas[xxiv] las siguientes actitudes del usuario para cada tarea: pedidos de ayuda, pérdida de foco y comportamiento errático en el lado negativo, y finalización rápida de la tarea por el lado positivo. También resultaba importante registrar los comentarios de los usuarios respecto al sistema.

Los resultados de la verificación[xxv] fueron procesados, llegando a una lista de problemas específicos de usabilidad que no vale la pena resaltar aquí, pero que se puede apreciar en el anexo 7. El contenido de este anexo deja en evidencia las dificultades con que se encontraron los usuarios al utilizar determinadas partes del sistema. En definitiva se trata de un listado de problemas de la aplicación donde quedan de manifiesto dificultades de usabilidad de distinta índole, desde opciones confusas hasta inconsistencias en el flujo de control.

La satisfacción de cliente fue evaluada informalmente y se destaca que su grado de conformidad con el producto final obtenido fue bueno. A pesar de los esfuerzos de desarrollo e implementación, el mayor valor de este proyecto de grado lo encuentra en el hecho de haber recorrido un proceso de maduración y materialización de su idea. Sin desmedro de esto, es claro que -a pesar de la evolución positiva- aún queda camino por recorrer antes de lograr un producto comercializable.

5.4 Retroalimentación de la comunidad

ActiveCollab de por sí ya era una aplicación funcional, que incluía un instalador y manejo de actualizaciones. OpenGoo mantuvo esta funcionalidad, y en el foro [40] se pueden encontrar testimonios de usuarios que lo instalaron y probaron. Además, las estadísticas de SourceForge [36] muestran más de dos mil descargas[xxvi] al mes de abril de 2008. Todo esto a pesar de que –por tratarse de una versión beta- su uso está desaconsejado en ambientes de producción. El único lugar dónde se sabe que se utiliza (en ambiente de producción) es en la *start-up* Archivo3.com[xxvii]. Por otra parte, se recibió un correo electrónico manifestando interés en utilizar OpenGoo en la gestión interna del sitio NBA.com[xxviii].

[xxiii] El anexo 5 contiene las instrucciones exactas que se les dieron a los usuarios.

[xxiv] El anexo 4 es un ejemplo del formulario vacío que el estudiante debería llenar mientras el usuario realiza las tareas asignadas.

[xxv] El anexo 6 contiene los reportes completos para los cinco usuarios en formato bruto.

[xxvi] En abril de 2008 la cantidad de descargas de OpenGoo vía Sourceforge superó las dos mil.

[xxvii] Archivo3.com está conformada por Ignacio de Soto, Marcos Saiz, Conrado Viña y otros socios.

[xxviii] El comentario de que OpenGoo es usado en la gestión interna de la NBA (Nacional Basketball Association) llegó por mail al cliente del proyecto (ver anexo 20).

El foro tuvo niveles de actividad motivadores (160 *posts a abril de 2008*) y no faltaron los comentarios de aliento en el *blog* del producto. También se recogieron sugerencias, y se supieron aprovechar todas las instancias de retroalimentación para generar oportunidades de mejora.

En el marco del proyecto se contactó un desarrollador uruguayo en la universidad de Rochester, EE.UU. interesado en colaborar en el desarrollo de las hojas de cálculo. Aparte, a lo largo del proyecto surgieron algunos contactos de usuarios no técnicos que ofrecieron ayuda. Además se estableció un vínculo con el grupo de desarrolladores de Clipperz (un proyecto de código abierto que hace especial hincapié en la privacidad de los datos) y otros proyectos similares. A pesar de que ninguno de estos contactos se materializó en trabajo a favor del proyecto, las experiencias fueron más que enriquecedoras.

Un dato alentador para el proyecto OpenGoo es que fue seleccionado como expositor para el Open Coffee Show. Este evento fue organizado por el Open Coffee Club[xxix] Montevideo [24] y en él se expusieron varios emprendimientos nacionales.

Con respecto al cliente, se pudo verificar que su satisfacción fue alcanzada con éxito. OpenGoo continúa siendo desarrollado en su emprendimiento Archivo3.com, del cual los estudiantes forman parte. El objetivo final es vender OpenGoo como SaaS. Por otro lado, dada la experiencia positiva del cliente con el proyecto, éste fue presentado como nuevo proyecto de grado para el año 2008.

[xxix] El Open Coffee Club es una organización sin fines de lucro fundada por Saul Kline, un emprendedor de Gran Bretaña, que pretende reunir a emprendedores e inversores informalmente una vez por semana.

6.Conclusiones

El desarrollo de OpenGoo para el proyecto de grado se dio por finalizado al liberar la versión 0.5.2. Se logró un prototipo de *web office* que integra edición de documentos de texto, hojas de cálculo y presentaciones, además de herramientas para la gestión de proyectos. Además se llegó a dar forma a un núcleo al que resulta fácil integrar nuevos módulos. La versión 0.5.2 incluyó los 28 casos de uso planificados dentro del alcance. De todas formas, el equipo de desarrollo continuó su trabajo en OpenGoo mientras elaboraba el presente informe.

Analizando los resultados de la verificación, se destaca que los usuarios encontraron la usabilidad de los nuevos módulos (editores y administrador de archivos) mejor que la del módulo de gestión de proyectos. A pesar de los esfuerzos realizados para mejorarla, la interfaz de gestión de proyectos todavía no resulta tan amigable como para que no se generen dudas respecto a su funcionamiento[xxx].

Se logró mejorar activeCollab en todos los aspectos planificados, dándole un *look and feel* que fue aprobado por la comunidad de usuarios, y agregándole una buena cantidad de funcionalidades nuevas. El único punto de los objetivos iniciales que no obtuvo los resultados esperados –y fue incluido como mejora a futuro- fue el rediseño del *look and feel del gestor* de proyectos, ya que se mantuvo el original.

A pesar de lo alcanzado en la versión 0.5.2 de OpenGoo, aún se está lejos de una versión completa, principalmente debido a la extensa y ambiciosa lista de funcionalidades pretendidas. De los 22 módulos candidatos se implementaron 7, como fue negociado en el alcance. Además, para los módulos implementados se podría agregar más funcionalidad, como el soporte de formatos de documentos. Éstas son algunas de las funcionalidades que están siendo implementadas luego de culminado el proyecto de grado.

Se puede afirmar que se marcó el rumbo para el posicionamiento de OpenGoo como una oficina web de código abierto[xxxi] y se desarrolló Slimey, el primer editor de presentaciones de código abierto compatible con el formato S5. Por tratarse de un producto novedoso realizado con herramientas de vanguardia, se podría considerar que se alcanzó un pequeño éxito en lo que a la comunidad de código abierto se refiere.

Se detectó que durante el año 2008 surgieron nuevos sitios Web 2.0 cuyos objetivos son similares a los que OpenGoo persigue. Muchos de estos sitios resuelven necesidades específicas y otros –como Microsoft Office Workspaces [22]– están dirigidos a otro tipo de usuarios. Sin embargo, no ha surgido ningún proyecto *open source* que haya hecho pública una hoja de ruta tan ambiciosa como la de OpenGoo.

[xxx] Estas afirmaciones se basan en la verificación con usuarios desarrollada en el capítulo 5.

[xxxi] El proyecto OpenGoo fue mencionado en diversos *blogs*. En uno de ellos se realizó una crítica muy completa sobre la primera versión. [41]

6.1 Trabajos futuros

Desarrollar una oficina web completa era una tarea que superaba ampliamente el objetivo de este proyecto. Esto se debió a que el objetivo final de OpenGoo era demasiado ambicioso para un único proyecto de grado. Igualmente, se puso especial énfasis en documentar todas las ideas y mejoras que fueron surgiendo en el proceso de desarrollo del proyecto.

Este listado de trabajo a futuro se dividió en dos grandes áreas: limitaciones de la funcionalidad existente y nuevos módulos a integrar al sistema.

Con respecto a las limitaciones de la funcionalidad existente, el anexo 11 lista veinte áreas en las que existen oportunidades de mejora. Éstas varían desde requerimientos simples, como la posibilidad de habilitar el almacenamiento automático al editar documentos, o la posibilidad de subir más de un archivo a la vez, hasta requerimientos de complejidad muy superior. Dentro de estas últimas, se incluyen requerimientos que implican realizar tareas de análisis de código y verificación exhaustiva. Ejemplos de estas tareas más complejas son: las pruebas de escalabilidad, concurrencia y seguridad, la adaptación de la interfaz para dispositivos móviles, y la definición de una interfaz para *plugins*. En este listado de limitaciones del sistema sólo se incluyen mejoras que no implican agregar nuevos módulos.

El anexo 12 contiene un listado de módulos complementarios a los existentes. Hay módulos planificados para el corto plazo, e incluso algunos que ya se comenzaron a desarrollar, como ser el módulo de manejo de contactos y el de manejo de eventos de calendario. También hay módulos que se ven más lejanos y aún no se encuentran planificados, como la integración de un sistema de *Enterprise Resource Planning* o la integración con redes sociales. El listado del anexo 12 no pretende ser una especificación detallada, sino que incluye una pequeña descripción de los primeros módulos que se planifica incluir en el producto.

De acuerdo a la planificación, el proceso a seguir para cada uno de los nuevos módulos debería ser similar al que se siguió para los módulos que actualmente se encuentran integrados. La primera etapa consistiría en un relevamiento del estado del arte enfocado en el nuevo módulo. La segunda se centraría en la definición de si existe un módulo que justifique ser integrado, o si se desarrollará desde cero. La tercera etapa comprendería la incorporación del nuevo módulo, mientras que la última etapa consistiría en un proceso de verificación.

En definitiva, el objetivo final del proyecto OpenGoo es servir de núcleo conector de distintos proyectos de código abierto, manteniendo la visión de ofrecer una solución de *web office* completa, actualizada y que pueda competir con las soluciones propietarias más populares.

Glosario

- **Accounting**: Gestión de la actividad financiera.
- **activeCollab**: Gestor de proyectos de código abierto utilizado como núcleo de OpenGoo.
- **AJAX**: Asynchronous Javascript And Xml.
- **ASP**: ActiveX Server Pages .
- **Back-end**: Porción de una aplicación que corre en el servidor.
- **Beta tester**: Usuarios responsables de verificar un software mediante el uso pretendido.
- **Blog**: Aplicación web que permite a un usuario publicar artículos y a otros usuarios comentar los artículos.
- **Cliente de correo**: Software que accede a un servidor de correo electrónico.
- **CMS**: *Content Management System*. Software que gestiona contenido.
- **CRM**: *Customer Relationship Management*. Software que gestiona la interacción de una empresa con sus clients,
- **CSS**: Cascade StyleSheet.
- **Dashboard**: Componente que representa una vista general de un software.
- **DHTML**: Sitios web en HTML con contenido dinámico provisto por scripts.
- **DOC**: Formato de documentos de texto utilizado por Microsoft Word desde 1997 hasta 2006.
- **ERP**: *Enterprise Resource Planning*. Software que gestiona diversos aspectos de funcionamiento interno de una empresa.
- **ExtJS**: Una biblioteca de Javascript que brinda funcionalidad AJAX y controles de interfaz gráfica web.
- **Foro**: Aplicación web donde los usuarios intercambian opiniones en distintas categorías mediante la publicación de mensajes.
- **Front-end**: Porción de una aplicación que corre en la computadora cliente.
- **Google Docs**: Aplicaciones de Google que permiten la edición de documentos, presentaciones y hojas de cálculo.
- **Host, hosting**: Proveedor de espacio de almacenamiento de sitios web.
- **HTML**: HyperText Markup Language.
- **IP**: *Internet Protocol*. El protocolo usado para Internet.
- **Java**: Lenguaje de programación.
- **Javascript**: Lenguaje de scripting para páginas web.
- **Jsp**: Java server pages .
- **MVC**: Model View Controller.
- **ODF**: *Open Document Format*. Conjunto de formatos abiertos para aplicaciones de oficina.

- **ODS**: *Open Document Spreadsheets*. Formato abierto para hojas de cálculo.
- **ODT**: *Open Document Text*. Formato abierto para documentos de texto.
- **Oficina en línea**: *Ver Web Office*.
- **Oficina virtual**: Ver *Web Office*.
- **Oficina web**: Ver *Web Office*.
- **Open Source**: Código libre o código abierto.
- **Plugin**: Extensión a un programa.
- **Post**: Mensaje publicado en un foro o blog.
- **PHP**: PHP Hypertext Preprocessor. Lenguaje de programación web de servidor.
- **S5**: Simple Standards-Based Slide Show System. Formato de presentaciones web.
- **SaaS**: Software as a Service .
- **Script**: Código que ejecuta sin necesidad de ser compilado.
- **SLIM**: Formato de presentaciones web compatible con S5 definido para el proyecto Slimey.
- **Slime**: Motor que corre presentaciones en formato SLIM.
- **Slimey**: Editor de presentaciones web en formato SLIM.
- **Social Networking**: Una aplicación que soporta Social Networking permite que sus usuarios intercambian perfiles y actividades.
- **Software**: Término para referirse a los programas de computadora.
- **Start-up**: Empresa que recién comienza.
- **Template**: Plantilla.
- **Web**: Término utilizado para referirse a Internet.
- **Web 2.0**: Concepto que define la situación actual de las aplicaciones web, de contenido interactivo, comparado con las aplicaciones anteriores de contenido más estático.
- **Web Conferencing**: Software que permite la comunicación en vivo entre varias personas, cada una desde un computador distinto en cualquier lugar del mundo.
- **Web Office**: Software que se accede por Internet y que reúne un conjunto de herramientas de oficina.
- **Wiki**: Aplicación web cuyo contenido es definido por usuarios como artículos.
- **XHTML**: Estándar similar a HTML pero compatible con XML.

Referencias

[1] ACTIVECOLLAB FORUMS[en línea] <http://www.activeCollab.com/forums/> [20 de marzo de 2008]

[2] ACTIVECOLLAB FORUMS. Framework [en línea] <http://www.activeCollab.com/forums/topic/13/> [20 de marzo de 2008]

[3] BERG O. The Content Economy. [en línea] <http://www.thecontenteconomy.com/2007/09/what-characterizes-saas-software.html> [21 de marzo de 2008]

[4] CHEUNG, B. Open Source Software: The power of community. [en línea] <http://www.linuxinsider.com/story/59127.html> [21 de marzo de 2008]

[5] CHRYSTOFER FRY ET AL. Static and Dynamic Semantics of the Web. [en línea] <http://web.media.mit.edu/~lieber/Lieberary/Dynamic-Semantics/Dynamic-Semantics.html> [20 de marzo de 2008]

[6] CLEVELAND, G. Selecting Electronic Document Formats. [en línea] <http://www.ifla.org/VI/5/op/udtop11/udtop11.htm> [16 de marzo de 2008]

[7] COMPARE HOSTING COMPANIES [en línea] <http://www.comparehostingcompanies.com/> [21 de marzo de 2008]

[8] FREE SOFTWARE FOUNDATION. [en línea] <http://www.fsf.org/licensing/essays/free-sw.html > [21 de marzo de 2008]

[9] FREASHMEAT.NET. [en línea] <http://freshmeat.net/ [16 de marzo de 2008] >

[10] GILFILLAND, I. Database Journal Marzo 8 de 2005 [en línea] <http://www.databasejournal.com/features/mysql/article.php/3486596> [21 de marzo de 2008]

[11] GOOGLE DOCS. [en línea] <http:// docs.google.com> [21 de marzo de 2008]

[12] HAXE [en línea] <http://haxe.org/> [21 de marzo de 2008]

[13] INTERNETWORLDSTATS. [en línea] <http://www.internetworldstats.com/stats7.htm> [29 de marzo de 2008]

[14] JBOSS [en línea] <http://www.jboss.com>

[15] KOPFLER, E. Obsesivecollaborator.com [en línea] <http://obsessivecollaborator.com/2007/09/OpenGoo.html> [30 de marzo de 2008]

[16] LABELLE, F. PROGRAMMIGN LANGUAGE USAGE GRAPH [en línea] http://www.cs.berkeley.edu/~flab/languages.html [20 de marzo de 2008]

[17] LEADBEATER, C. LINUX.com [en línea] <http://www.linux.com/articles/47307> [31 de marzo de 2008]

[18] MACMANUS, R. Google Presentations Launched[en línea] <http://www.readwriteweb.com/archives/google_presentations_launched.php> [21 de marzo de 2008]

[19] MACMANUS, R. Web 2.0 Office. [en línea] <http://www.readwriteweb.com/archives/web_20_office.php> [21 de marzo de 2008]

[20] MACMANUS, R. Web Office Defined [en línea] http://www.readwriteweb.com/archives/web_office_defined.php> [12 de abril de 2008]

[21] MEYER, E. S5 Home Page [en línea] <http://meyerweb.com/eric/tools/s5/> [16 de marzo de 2008]

[22] Microsoft Live Workspace. [en línea] <http://office.live.com [20 de marzo de 2008] >

[23] MYSQL [en línea] <http://www.mysql.com>

[24] OPEN COFFEE CLUB MONTEVIDEO. [en línea] <http://www.opencoffeemvd.org/index.php?option=com_content&task=view&id=1&Itemid=2> [20 de marzo de 2008]

[25] OPENJS [en línea] <http://www.openjs.com/tutorials/advanced_tutorial/frames.php> [3 de abril de 2008]

[26] OPENLASZLO [en línea] <http://www.openlaszlo.org/> [21 de marzo de 2008]

[27] PROJECT PIER [en línea] <http://www.projectpier.org> [30 de marzo de 2008]

[28] S5 PROJECT. [en línea] <http://s5project.org/>

[29] S5 RELOADED [en línea] <http://www.netzgesta.de/S5/> [16 de marzo de 2008]

[30] S6 [en línea] <http://tzi.org:2000/s6.html> [16 de marzo de 2008]

[31] SCHULLER, S. SaaSBlogs [en línea] <http://www.saasblogs.com/2006/12/01/what-is-saas-the-answer-is-rooted-in-the-end-user/> [21 de marzo de 2008]

[32] SHNEIDERMAN,B. Designing the user interface [en línea] < http://www.cs.umd.edu/~ben/ >[21 de marzo de 2008],

[33] SKOS EN W3.org. [en línea] <http://www.w3.org/TR/2008/WD-skos-primer-20080221/> [20 de marzo de 2008]

[34] SOFTWARE & INFORMATION INDUSTRY ASSOCIATION . Software as a Service: Strategic Backgrounder. [en línea] <*http://www.siia.net/estore/ssb-01.pdf*> págs. 4 - 5, Febrero 2001.

[35] SOURCEFORGE [en línea] <http://sourceforge.net> [20 de marzo de 2008]

[36] SOURCEFORGE. [en línea] < http://sourceforge.net/projects/opengoo> [2 de abril de 2008]

[37] SSG. A S5, Slidy Generator [en línea] <http://blog.amicoimmaginario.it/category/projects/rem/ssg/> [16 de marzo de 2008]

[38] TECHSOUP. [en línea] <http://www.techsoup.org/learningcenter/software/page4765.cfm?forum=2009&forumtopicid=63155&mid=228492&show=comment&> [20 de marzo de 2008]

[39] THE COUNTER [en línea] <http://www.thecounter.com/stats/2007/July/browser.php> [29 de julio de 2007]

[40] THE GOO FORUM [en línea] <http://forums.opengoo.org> [30 de marzo de 2008]

[41] THE OBSESSIVE COLLABORATOR. Opengoo [en línea] <http://obsessivecollaborator.com/2007/09/opengoo.html> [15 de noviembre de 2007]

[42] UNIVERSITY OF ILLINOIS. [en línea] <http://wiki.cs.uiuc.edu/ProjectIdeas/Web+Based+Presentation+Editor+for+ODF+documents> [20 de marzo de 2008]

[43] VAN DER HENST, C. ¿Qué es la Web 2.0? [en línea] <http://www.maestrosdelweb.com/editorial/web2/> [21 de marzo de 2008]

[44] WAKE FOREST UNIVERSITY- [en línea] <http://www.ecn.wfu.edu/~cottrell/wp/wp-sp.htk> [20 de marzo de 2008]

[45] YAHOO INC. Yahoo!Buzz. [en línea] <http://buzz.yahoo.com/> [21 de marzo de 2008]

[46] ZAMZAR [en línea] <http://www.zamzar.com/> .[16 de marzo de 2008]