VNIVERSIDAD NACIONAL AVTÓNOMA DE MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

" SISTEMA SOBRE PLONE PARA LA CAPTURA Y RECOLECCIÓN DE INFORMACIÓN CURRICULAR DEL INSTITUTO DE MATEMÁTICAS "

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERÍA (COMPUTACIÓN)

P R E S E N T A:

MARCO ANTONIO LÓPEZ RABADÁN

DIRECTOR DE TESIS:
"SERGIO RAJSBAUM GORODEZKY"

México, D. F. **2007.**

Agradecimientos

Primero que nada doy gracias a Dios, quien me permitió hacer la maestría y me ha dado todo lo que le he pedido. Agradezco a mi esposa Lizbeth por su paciencia, apoyo y comprensión conmigo durante toda la maestría. También agradezco a mi madre por apoyarme moral y económicamente cuando me faltaron los recursos.

El mancebo no puede ser sabio, porque prudencia
requiere experiencia. Necesaria es la experiencia
para saber cualquier cosa.

Aristóteles.

Índice general

Capítulo 1

Introducción

1.1. Antecedentes

Desde hace aproximadamente 17 años, el Instituto de Matemáticas (IMATE), realiza un informe anual de actividades el cual tiene como propósito reportar el trabajo que realizaron los investigadores y técnicos académicos de las diferentes sedes que tiene el Instituto. Cabe mencionar que el Instituto cuenta actualmente con cuatro sedes en el país, que se encuentran en: Cuernavaca, D. F., Morelia y Oaxaca.

Para realizar este informe cada uno de los investigadores debe proporcionar un reporte, el cual está basado en un formato preestablecido que cuenta con 33 rubros que denotan las actividades académicas del investigador durante el año. Por ejemplo, el rubro *Proyectos* tiene el siguiente esquema: nombre del proyecto, tipo de proyecto, líneas de investigación en las que se enmarca, objetivo de la investigación, patrocinador para realizar el proyecto y monto patrocinado. No es necesario que el investigador reporte los 33 rubros, sino únicamente aquellos en los que trabajó durante el año. Cabe aclarar que un rubro puede reportarse más de una ocasión (por ejemplo, un investigador puede realizar más de un artículo al año).

Este reporte inicialmente era entregado en papel y en los últimos años se ha hecho más común entregarlo por medio de un documento electrónico, lo cual evita la tarea de volver a capturar la información, aunque no organizarla.

Además del reporte que entregan los investigadores al IMATE cada año, para que se realice el informe anual de actividades; cada 3 o 5 años deben entregar un resumen de sus actividades realizadas durante los últimos años para participar en el *Programa de Primas al Desempeño del Personal Académico de Tiempo Completo (PRIDE)*. Cabe mencionar que este resumen se realiza utilizando el formato con los 33 rubros antes mencionados.

El *Programa de Apoyo a la Incorporación de Personal Académico de Tiempo Completo (PAIPA)* tiene la finalidad de apoyar y estimular la contratación de personal que se haya distinguido en la actividad académica o en la práctica profesional. Para que los investigadores puedan ingresar a este programa deben entregar un currículum vítae en el que muestren sus antecedentes académicos, escrito con un formato libre. Este currículum puede estar basado en los 33 rubros antes mencionados puesto que son las principales

actividades que realizan los investigadores.

Además, el currículum vítae es un documento que los académicos acostumbran presentar ante diferentes instituciones; en ocasiones varias veces al año. Ya sea para una contratación, un concurso, una promoción, una premiación, estímulos, etc.

Por parte de los investigadores, éstos deben realizar actividades redundantes al tener que capturar y organizar la misma información o parte de ésta para entregar su reporte de actividades para el IMATE, resumen de actividades para el PRIDE o para obtener un currículum vítae actualizado (ya sea para el PAIPA u otra institución). La Secretaría Académica, también realiza actividades redundantes al tener que recopilar, revisar y organizar la información que reportan los investigadores en diferentes ocasiones, siendo que parte de ésta ya se había revisado y organizado anteriormente.

El IMATE actualmente cuenta con 101 investigadores. En promedio cada investigador reporta 18 actividades dentro de los diferentes rubros durante un año [1], esto implica que se debe hacer un compendio de 1818 actividades aproximadamente. Esta tarea es complicada por la cantidad de información a organizar y propensa a errores, sin embargo, puede complicarse aún más si el investigador entrega el reporte en papel.

Debido a esta forma en que cada investigador entrega sus reportes, el Secretario Académico del IMATE, encargado de realizar el informe anual de actividades, se tarda en promedio dos meses y medio en realizar esta actividad, dedicando casi tiempo completo [2], sin contar que también tiene que procesar, de manera más breve otros que le lleguen a pedir durante el año.

1.2. Planteamiento del problema

Se analizó la forma en que se han venido realizando los diferentes reportes sobre las actividades de los investigadores del IMATE encontrando varios problemas que se muestran a continuación:

- **Propenso a errores humanos.** Debido a que los investigadores reportan sus actividades por medio de un documento en papel o de forma electrónica, como se mencionó anteriormente, no se pueden organizar las actividades de forma automática; es decir, es necesario que alguna persona las organice y las procese, ya sea total o parcialmente. Debido a esto se pueden cometer errores como confundir, omitir, duplicar, organizar mal la información o realizar cálculos incorrectos sobre ésta; sobre todo porque se trata de muchas actividades.

- **Se debe recordar o tener un registro de las actividades realizadas en años anteriores.** El reporte para el informe anual de actividades, puede incluir

[1] Este dato fue obtenido el 8 de Abril de 2007 del *Sistema de Información del Instituto de Matemáticas* (`https://info.matem.unam.mx/`) sobre las actividades reportadas por los investigadores, correspondientes al año 2006.

[2] Este dato fue obtenido por medio de un cuestionario que se le realizó a Ángel Manuel Carrillo Hoyo, Secretario Académico del IMATE de 1993 a 2006, quien realizó 13 informes anuales. El cuestionario se encuentra en el anexo A.

actividades que duraron más de un año; el reporte para el PRIDE incluye actividades de los últimos 3 o 5 años; y un currículum vítae, ni se diga, contiene información académica y laboral a lo largo de la vida. Si no se guarda un registro de las actividades que se van realizando a lo largo de los años, se puede omitir información al momento de entregar un reporte o currículum vítae. Además, también el Instituto debe tener un registro sobre estas actividades para darles continuidad.

- **Largos tiempos de espera.** Los investigadores no pertenecientes a la sede del D. F. deben enviar por correspondencia su reporte de actividades, lo cual puede causar incertidumbre mientras no se reciba confirmación de recibido, puesto que no se tiene el 100 % de seguridad de que va a llegar bien a su destino. Además el proceso se hace más lento, puesto que realizar algún cambio o modificación puede implicar varios días.
- **Redundancia en las actividades.** Como ya se mencionó anteriormente, tanto los investigadores como el Instituto de Matemáticas realizan actividades redundantes al capturar, revisar u organizar las mismas actividades en varias ocasiones para obtener diferentes reportes.
- **El tiempo que toma realizar los reportes es grande.** Para generar el informe anual y demás reportes de actividades se siguen varios pasos.

  1. Se les solicita a los investigadores que envíen su reporte.
  2. Se envían y recolectan los reportes de los investigadores.
  3. Se clasifican y organiza las actividades según los diferentes rubros.
  4. Se realizan diversos cálculos por cada uno de los rubros.
  5. Se redacta el reporte o informe. Esta parte incluye cálculos ya realizados, redacción y un compendio de las actividades realizadas por los investigadores agrupados por diferentes parámetros como rubro y sede, entre otros.
  6. Se publican las actividades. En el caso del informe anual, se hace por medio de un libro que se distribuye en las diferentes sedes; en el caso del PRIDE se les entrega el reporte a un grupo de personas encargadas de evaluar el desempeño de los investigadores.

  En el caso del informe anual de actividades, desde la recolección de los reportes de los investigadores hasta el envío a impresión del libro se tarda el Secretario Académico aproximadamente dos mes y medio, y se ha llegado a tardar hasta tres meses y medio, lo cual es bastante tiempo que implica mucho trabajo. Todos los reportes podrían realizarse más rápido si se automatizaran varias de sus tareas y si se realizaran menos duplicadas.

1.3. Propuesta o solución

Se propone realizar un sistema sobre Plone, que es considerado uno de los mejores *Sistemas manejadores de contenido (SMC)* disponibles hoy en día [27], que además es

distribuido de forma gratuita bajo la licencia *GNU GPL*[3].

La razón por la cual se eligió Plone y no otra tecnología como Java es porque Plone permite el desarrollo de aplicaciones de forma muy rápida debido a que ya hay muchos productos disponibles. Además, Plone ya viene listo para usarse, sólo hay que personalizar algunos detalles; es decir, el manejor de usuarios, control de acceso, permisos y otras características subyacentes ya vienen implementadas y de forma eficiente, sólo habría que enfocarse en lo relacionado con el almacenamiento y recuperación de información curricular.

Este sistema debe permitir, mediante interfaces amigables, que los investigadores ingresen sus actividades conforme las vayan realizando a lo largo del año desde una máquina con conexión a Internet y un explorador Web. Estas actividades se van a ir almacenando a lo largo de los años, de tal forma que se tenga sólo un lugar donde el investigador guarde su información curricular pudiendo realizar consultas u obtener reportes sobre éstas.

El currículum vítae es un documento que los investigadores acostumbran presentar ante diversas instancias de evaluación, ya sea para recontratación, premiación, participar en un programa de estímulos, promoción, etc. Con este sistema se ofrece que el investigador tenga un currículum vítae actualizado.

Para su fácil localización, las actividades se van a organizar por rubros y se van a tomar los 33 que ya se venían manejando, tratando de modificarlos al mínimo puesto que la mayoría de investigadores ya se encuentran familiarizados con éstos. De esta forma se intenta que el cambio no sea tan difícil.

Se va a proporcionar una conexión segura cifrando la información que fluya entre el sistema propuesto y la máquina del usuario que lo utilice mediante un servidor Apache con SSL habilitado [4].

Hay varios tipos de usuarios que van a utilizar el sistema de forma remota (investigadores, personal interno del IMATE, administradores, desarrolladores y además está abierto al público en general), lo cuales deben ser considerados para implementar un adecuado control de acceso a la información, permitiendo que ésta sólo sea vista, creada, borrada o modificada por las personas autorizadas. Los usuarios que no estén registrados dentro del sistema sólo deben de poder acceder a la información que se haya elegido como pública.

Cabe mencionar que este proyecto se realizará en conjunto con otros dos compañeros del posgrado: Gildardo Bautista García Cano y Adriana Ramirez Vigueras; quienes se encargarán de proporcionar la plataforma donde se montará el sistema propuesto. Ellos se encargarán de realizar toda la instalación, desde el sistema operativo, plataforma Zope, implementación de un esquema de usuarios, diversas consultas; hasta tener un sitio que cubra las necesidades del IMATE.

[3]GNU GPL (General Public License o licencia pública general) es una licencia creada a mediados de los 80's, y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios. En `http://www.gnu.org/copyleft/gpl.html` se puede obtener más información.

[4]SSL o *Secure Socket Layer* es un protocolo para manejar el envío de mensajes seguros a través de la Internet utilizando cifrado de la llave pública y privada, además haciendo uso de los certificados digitales.

1.4. Objetivos

Al realizar este sistema se busca obtener los siguientes beneficios:

- Reducir el tiempo que toma crear el informe anual de actividades y demás reportes; además de la cantidad de trabajo necesario para realizarlo.
- Disminuir considerablemente la intervención humana en la generación de los reportes, para que de esta forma sean menos propensos a errores hummanos.
- Aprovechar la información que ya se tiene disponible en diferentes lugares con formato estándar de forma que el sistema pueda entender y clasificar esta información ahorrándole tiempo al usuario.
- Acceder al sistema desde cualquier estado o país, permitiendo de esta forma agilizar el intercambio de información, dándole certeza a los usuarios de que su información ha sido enviada y recibida de forma correcta y segura.
- Recordar a los investigadores sobre aquellas actividades aún vigentes que iniciaron uno o más años atrás, para que le puedan dar continuidad.
- Evitar capturas redundantes. Hay varios reportes que se entregan donde se incluyen las mismas actividades. Con este sistema ya no se tendrán que capturar las mismas actividades para cada reporte, sólo se hará una vez.
- Realizar consultas o cálculos sobre las actividades reportadas por los usuarios mediante interfaces amistosas sin tener que conocer el lugar o forma en que están almacenadas.
- Tener la información disponible en repositorios que permiten su manipulación.

1.5. Motivación

Actualmente se ha vuelto más importante estructurar la información como contenido Web para aquellas organizaciones que necesitan compartir información o distribuir conocimiento con otras personas u organizaciones que se encuentran geográficamente situadas en diferentes lugares. El IMATE no es la excepción, ya que cuenta con cuatro sedes en diferentes estados y personal que frecuentemente está viajando a otros países o estados de la República. Además, la mayoría de la información que el investigador reporta en sus actividades se desea que sea pública para cualquier otra persona que la quiera ver, sin importar que sean externos al IMATE.

La mayoría de las organizaciones que han implementado sitios Web sin la ayuda de un sistema manejador de contenido, se han visto envueltos en problemas asociados con el mantenimiento del contenido de sus sitios conforme los volúmenes de información se van incrementando.

Los SMC son una respuesta a este problema de organización y manejo de sitios con gran cantidad de información. Además permiten crear un sitio Web en considerablemente

poco tiempo. Aún más, existen SMC que ya vienen listos para utilizarse, como es el caso de Plone, sólo hay que personalizar algunos aspectos, como el visual y los permisos de acceso, para que se pueda utilizar.

Plone trabaja con un enfoque orientado a componentes permitiendo desarrollar aplicaciones de forma muy rápida, sin embargo tiene como desventaja una curva de aprendizaje bastante prologanda.

El desarrollo de un sistema basado en componentes puede parecer una tarea muy sencilla, sobre todo cuando se tienen tantos componentes disponibles, como es el caso de Plone. Sin embargo, implica varias tareas como son: establecer los requisitos del sistema que se va a construir, establecer un diseño arquitectónico, examinar requisitos para determinar los componentes disponibles que se van a utilizar, desarrollar componentes necesarios no disponibles, adaptar e integrar componentes.

Para el desarrollo del sistema propuesto se deben tener los siguientes conocimientos:

- Paradigma orientada a objetos. Pues se utiliza en gran medida Python que es orientado a objetos.
- Ingeniería de software basada en componentes. Por la magnitud del sistema se deben utilizar técnicas de ingeniería de software para obtener resultados eficientes, además debe ser enfocado a componentes por la forma en que trabaja Plone.
- Programar en Python. Gran parte de los componentes desarrollados en Plone se hace en Python.
- Desarrollo de páginas Web. Las interfaz entre el usuario y el sistema es por medio de páginas Web.
- Integración de páginas Web con el lenguaje ZPT. Para darle dinamismo a las páginas en Plone se utiliza el lenguaje ZPT, del cual se hablará más adelante.

1.6. Estructura de la tesis

A continuación se muestra una breve descripción de lo que se verá en cada capítulo:

Capítulo 2. Marco teórico

Este capítulo explica lo que es un sistema manejador de contenido y sus características. Se hace una comparación entre tres de los mejores sistemas manejadores, considerados hoy en día. Además se explican varias de las características de Plone, principalmente las que se utilizaron para desarrollar el sistema propuesto.

Capítulo 3. Análisis, diseño e implementación

En este capítulo se muestran los requerimientos del sistema y casos de uso, con lo cual se obtiene una buena perspectiva de lo que tiene que hacer el sistema. Después se muestran dos arquitecturas del sistema: física y de capas. Después se muestran los componentes

existentes que se utilizaron y los que se implementaron. Finalmente se muestra cómo se integraron los componentes para formar el sistema final.

Capítulo 4. Implementación del sistema SCRICI

Este capítulo inicia mostrando las herramientas utilizadas para el desarrollo del sistema, las cuales se recomiendan ampliamente para reducir el tiempo de desarrollo de un sistema sobre Plone. Después se muestra de forma más detallada que en el capítulo 3 la forma en que se fue desarrollando el sistema, aunque sin profundizar mucho en aspectos técnicos. Lo que se pretende con este capítulo es ampliar el panorama del desarrollo sobre Plone, servir como guía para poderle dar mantenimiento al sistema SCRICI y para que otras personas puedan realizar aplicaciones sobre Plone.

Capítulo 5. Resultados y conclusiones

En este capítulo se muestran los resultados obtenidos con el sistema realizado, conclusiones y trabajo futuro.

Capítulo 2

Marco teórico

En este capítulo se verán varios aspectos teóricos necesarios para poder entender la forma en que fue implementado el sistema de información curricular del IMATE.

Se inicia mostrando cómo han evolucionado los SMC, sus características y por qué son útiles. Después se muestra una comparación entre tres sistemas manejadores de los cuales Plone es el que se elige para implementar el sistema curricular. Finalmente se explican varias de las características principales de Plone que se utilizaron para el desarrollo del sistema.

2.1. Introducción

Internet ha mostrado ser un medio para brindar, de forma fácil y eficiente, una amplia variedad de servicios a millones de usuarios. Para el 12 de Enero de 2007 en [5] se estimó que el número de usuarios que navegan en Internet era de $1,093,529,692$, y este número sigue creciendo rápidamente debido a que la gente se ve atraída, no sólo por los servicios que se brindan, sino también por la facilidad de acceso a éstos. Ya que utilizando un explorador de Internet se puede acceder a una gran cantidad de servicios bajo diferentes situaciones de forma transparente para el usuario, como diferentes sistemas operativos, bases de datos, aplicaciones o hardware.

En [19] se menciona que el manejo de la información es fundamental para cualquier empresa, con ello se puede lograr un alto nivel competitivo dentro del mercado y obtener mayores niveles de capacidad de desarrollo. Debido a esto la mayoría de las organizaciones han implementado un sitio Web para compartir, distribuir o intercambiar información, ya sea con un sitio externo para comunicarse con sus clientes o con uno interno para uso de los empleados, pero en general, todos los sitios tienen un problema común, que es: *'cómo manejar el contenido que tienen'*.

En [28] se comentan los problemas relacionados con el manejo de contenido. Conforme la información del sitio Web comienza a crecer se vuelve más complicado darle mantenimiento. Debido a que es difícil seguir el rastro de todas las páginas o recursos disponibles en el sitio, se dejan recursos obsoletos sin borrar; se complica la tarea de actualizar información existente, sobre todo cuando la modificación afecta varias partes del sitio, el

encargado del sitio Web se vuelve un cuello de botella para la publicación de nuevo contenido y se llega a publicar contenido sin ser antes aprobado.

Debido a estos problemas se ha vuelto esencial el uso de algún tipo de sistema o proceso para el manejo de información para las organizaciones con una significante cantidad de contenido Web con tendencias a crecer y con contenido que se encuentra cambiando frecuentemente.

Los SMC son la respuesta a estos problemas, permitiendo establecer y organizar sitios Web de tal forma que éstos puedan crecer y cambiar rápidamente manteniendo al mismo tiempo la calidad del sitio.

2.2. Evolución de los Sistemas Manejadores de Contenido

En los primeros años de la WWW los sitios Web consistían en gran parte de páginas con texto estático, ligas y un número limitado de imágenes. Todo esto era hecho principalmente con código escrito en HTML. Las páginas del sitio eran editadas directamente en el servidor de producción o enviadas por ftp a este servidor. Los sistemas para el manejo del contenido Web consistían básicamente de herramientas que permitían la producción de código HTML [20].

Esto ocasionaba que ocurrieran problemas como los siguiente: páginas con HTML codificado de manera pobre, ligas rotas, dependencia entre el creador de la información y el diseñador de las páginas Web, provocando también retrasos en la publicación de contenido.

Poco a poco fueron surgiendo estándares para mejorar el desarrollo de sitios Web y se fue haciendo una práctica común usar un servidor diferente al de producción para el desarrollo de contenido Web. De esta forma las pruebas se hacían en otra máquina para que no estuvieran disponibles páginas que aún no deberían estarlo.

En 1995 se comenzó a utilizar el primer sistema 'Wiki' [1] desarrollado por Ward Cunningham, que es considerado el primer SMC. Los sistemas Wikis permiten la creación y edición de documentos Web de forma colectiva utilizando una notación sencilla.

Posteriormente en 1997 fue liberado PHP/FI 2.0 que fue la base de muchos SMC. De 1995 a 1999 aparecieron bastantes SMC de los cuales la mayoría eran creados por compañías que utilizaban software propietario. Mientras tanto el software de código abierto estaba madurando, como lo es MySQL, Linux y servidores Web como Apache, que se volvieron más estables y útiles. Para 2003 estas herramientas estaban tan bien integradas que cualquier persona podía crear sistemas equivalentes a otros que habían costado millones de dólares y miles de personas, ahora de forma gratuita.

Con el surgimiento de los sistemas manejadores de contenido, la responsabilidad de la creación de contenido Web pasó a manos del creador del contenido. Con esto ya no es necesaria la separación en dos servidores, uno productivo y otro para desarrollo, sino que el contenido puede ser creado directamente en el servidor productivo con un proceso

[1] Del hawaiano 'wiki wiki' significa rápido, haciendo referencia a la rapidez con que se puede crear el contenido. Es una forma de sitio Web en donde se permite que los usuarios creen, editen, borren o modifiquen el contenido de una página Web de forma interactiva, fácil y rápida.

definido de publicación.

2.3. Datos, información y contenido

Para poder hablar de los SMC, es importante primero entender bien los conceptos: datos, información y contenido; porque en ocasiones se pueden confundir estos tres, llevando a la utilización incorrecta de este tipo de sistemas y provocando resultados frustrantes. En [9] se definen de la siguiente forma:

- **Datos**. Unidad mínima de información que representa de forma simbólica (numérica, alfabética, etc.) un atributo o característica de una entidad, y no tiene valor semántico por sí mismo.

  Algunos ejemplos de datos pueden ser: edad, altura, nombre, país o simplemente la letra *'x'*.

- **Información**. Es en lo que los humanos convierten su conocimiento cuando lo quieren comunicar a otras personas. Este conocimiento puede ser visible, audible o escrito.

  Algunos ejemplos de datos pueden ser: texto, sonidos, video e imágenes. Casi cualquier cosa puede ser información, inclusive un dato.

- **Contenido**. Información contenida dentro de un conjunto de metadatos[2] clasificados.

  Una de las definiciones del diccionario de la Real Academia Española sobre contenido es: *'Cosa que se contiene dentro de otra'*.

  La información es la que está contenida dentro de algo más, que son metadatos que permiten manipular la información.

La información tiende a fluir continuamente, como en las conversaciones, sin un inicio, final o atributos estándares. Mientras que los datos, que representan algo sobre una entidad, se pueden restringir para que esa representación sea de forma estándar. Por ejemplo, para que la edad sólo se represente con números enteros, la altura con números de punto flotante, etc.

Debido a esto, es más fácil manejar los datos que la información, ya que son más pequeños, simples y sus relaciones son bien conocidas, mientras que la información es extensa, compleja y con muchas relaciones importantes para su significado.

El contenido es información envuelta con datos que permiten definir y manipular la información. A estos datos se les conoce como *metadatos*, y prácticamente describen la información y le dan un contexto explícito de tal forma que la computadora la pueda manejar. La idea de esto es ponerle etiquetas a la información con datos, de tal forma que

[2]Los metadatos son datos que describen otros datos. En este caso los metadatos están describiendo la información.

la computadora, indirectamente a través de éstos, sepa qué hacer con la información (por ejemplo, organizarla, sistematizar su colección, manejo y publicación).

Actualmente se ha vuelto muy importante el manejo de la información, debido a la gran explosión que se ha venido dando en los últimos años. Por eso es importante distinguir la mejor forma en la que se va a estructurar ésta. Si se estructura un contenido con información muy variada, extensa y con mucho contexto puede ser difícil su automatización y manejo. En contraste, si sólo se estructura con datos se puede volver mecánica, poco interesante y difícil de entender para los que la van a utilizar.

2.4. Cuestiones a considerar en la creación de contenido

El incremento en la creación de contenido Web provocó nuevos problemas expuestos por Vidgen, Goodwin y Barnes [28], que deben de ser considerados en la creación de contenido:

- **Cuellos de botella.** El contenido llega en diferentes formas para ser editado y posteriormente publicado en una forma conveniente. El equipo de edición o personal correspondiente, toma el contenido para convertirlo en páginas Web con código HTML. Si las páginas tienen que ser modificadas, entonces se tiene que notificar al equipo de edición para que realice los cambios necesarios. Esta canalización del contenido puede ocasionar retrasos en la publicación del contenido.
- **Consistencia.** Si existen diferentes áreas o personas encargadas de la edición del contenido Web, pueden haber inconsistencias en el diseño y distribución del contenido dentro del sitio.
- **Navegación.** Si la estructura y el contenido no están cercanamente controlados, hay posibilidad de que las capacidades de navegación y búsquedas sufran inconsistencias haciendo más difícil para los usuarios encontrar información.
- **Duplicidad de información.** En muchas ocasiones el contenido de un sitio Web es una copia de información que se encuentra en otro lugar (ej. otros departamentos o instituciones), y los cambios que se hagan en un lugar se tienen que realizar manualmente en el otro. Idealmente la información no debe de estar almacenada redundantemente, pero si la información necesita ser copiada de un lugar a otro, entonces esto debe ser automatizado y controlado.
- **Control y audición de la información.** Puede haber contenido no autorizado dentro del sitio. El material publicado debe de ser sujeto a un proceso de revisión, aprobación y publicación para sólo publicar material aceptable, ya sea desde el punto de vista del mercado o legal.
- **Seguimiento de la información.** Para un manejo eficaz del contenido es necesario conocer varios aspectos sobre éste, como el creador del contenido, fecha de creación o fecha de última actualización.

La solución a estos problemas fue encontrada en los sistemas manejadores de contenido.

2.5. Sistemas Manejadores de Contenido (SMC)

En [9] se definen los SMC como sistemas que permiten la creación, manejo, distribución y búsqueda de contenido. Cubren todo el ciclo de vida de las páginas de un sitio Web, desde la creación y publicación del contenido hasta que es quitado y almacenado. Deben permitir que este contenido sea de una gran variedad de formatos y tipos, sin limitarse sólo a HTML, XML, videos, audio, imágenes y documentos.

Un SMC puede tener tanto páginas estáticas, escritas en HTML, como dinámicas, utilizando alguna tecnología como JSP, ASP o PHP. Pero sin importar si el contenido es una página estática o es mostrada utilizando páginas dinámicas, para el sistema sigue siendo contenido con diferencias menores, como la forma en que se va a mostrar. De hecho, la mayoría de los sitios Web grandes utilizan tanto páginas estáticas como dinámicas.

En [22] se indica que la creación del contenido debe estar estrictamente separada de la presentación que se le va a dar. Así el creador del contenido no tiene que tener conocimiento técnico para diseñar o crear una página, y el diseñador no tiene que estar creando cada página que se quiere publicar. De esta forma se reduce la complejidad de las tareas realizadas por los actores envueltos en el proceso de publicación de contenido, además de la fricción entre éstos.

La creación del contenido incluye la agregación de metadatos a la información. Como se mencionó anteriormente, el contenido es información y metadatos. Estos metadatos permiten que el SMC pueda manejar el contenido. Ejemplos de metadatos pueden ser: creador del contenido, fecha de última modificación y tipo de contenido; además pueden definir la forma en la que se visualizará el contenido o su comportamiento.

Usualmente el almacenamiento del contenido es realizado por medio de un repositorio centralizado y soportado por varias herramientas que permiten su manejo y manipulación. Estas herramientas pueden incluir: control de versiones, respaldos, recuperación, workflow, seguridad en la integridad del contenido, reportes, etc.

Los SMC incluyen una forma de definir la lógica de negocios llamada *workflow*, que es responsable de coordinar, programar y hacer cumplir ciertas tareas. El workflow se define en [22] como una serie de pasos que ocurren periódicamente en un SMC con el fin de coordinar las partes del sistema que se encuentran en movimiento, es decir, aquellas partes que se encuentran en estado cambiante. Por ejemplo, un proceso de publicación de contenido puede implicar varias etapas: la creación del contenido, revisión, aprobación y publicación. Cada una de estas etapas es un estado y el contenido en movimiento es aquel que cambia de un estado a otro.

Cada paso del Workflow puede ser activado de diferentes formas, ya sea manualmente por el usuario (puede ser con un botón), al crear o borrar contenido, en cierta hora y fecha, al ocurrir un evento, etc. Además, cada paso puede tener tareas que permiten cambiar el estado de algún objeto. Siguiendo el ejemplo anterior, en el estado de revisión de un contenido se puede querer que nadie más pueda acceder a éste sino sólo la persona que lo va a revisar y en el de publicación, que todos puedan acceder a éste.

Los SMC permiten la creación, edición, manejo y publicación de contenido a equipos variados de técnicos y no técnicos de forma tanto centralizada como descentralizada. Estas funciones son dirigidas mediante un conjunto de reglas, procesos y workflows centralizados,

que aseguran un sitio Web válido y coherente [19].

Los Sistemas Manejadores de Contenido Web *(SMCW)*, así como los SMC son parte del gran concepto de *Manejo de Contenido*. Un SMC es considerado SMCW si tienen su interfaz de control sobre Web o si la forma en que muestran y envían la información es sobre Web por medio de Internet o una Intranet. Pero frecuentemente se encuentra en la literatura que se describen ambos como si fueran lo mismo, como en [26], [27], [9] y [14]. Además la mayor parte de la literatura que se encontró se enfoca principalmente a SMCW. Debido a esto y por simplicidad, en el resto de la tesis se utilizará el concepto de SMC sin importar que se trate de un SMCW, puesto que en SMCW es un subtipo de un SMC.

2.6. Comparación entre diferentes SMC

Actualmente existen muchos SMC muy buenos, gratuitos y de código abierto. En el 2006, *Packt Publishing Ltd* [6] hizo un concurso en el que participaron jueces de *The Open Source Collective*, *MySQL*, *Eclipse Foundation* y 16,000 usuarios que votaron por el SMC que consideraron mejor. Los resultados se publicaron el 14 de noviembre de 2006, y los tres primeros lugares quedaron en el siguiente orden: *Joomla!*, *Drupal* y *Plone*.

Debido a que existen tantos SMC disponibles actualmente, sólo se mostrará una comparación entre Joomla!, Drupal y Plone, que fueron de los más populares, aunque en realidad se analizaron más.

A continuación se muestran varias tablas comparativas entre estos tres SMC, obtenidos del sitio *CMS Matrix* [2] que provee información sobre más de 750 SMC, en las que se muestran varias de las características analizadas.

A cada característica de estas tablas se le dio un peso ponderado de acuerdo a los requerimientos del sistema, o posibles ventajas que se pueden obtener de éstas. Este peso va de uno a cinco, utilizando uno para las características menos importantes, que puedan brindar muy pocos o ningún beneficio; y cinco para las características más importantes para el IMATE.

2.6.1. Requerimientos del sistema

En la siguiente tabla se muestra una comparación de los requerimientos del sistema entre Joomla!, Plone y Drupal. En este punto lo que se busca principalmente es utilizar sólo software libre, por lo tanto, los tres son aceptables.

Característica	**Drupal 5.1**	**Joomla! 1.0.7**	**Plone 2.5**
Servidor de aplicación	PHP 4.3.3+	Recomendado Apache, cualquier servidor que soporte PHP y MySQL	Zope
Costo aproximado	Gratis	Gratis	Gratis
Base de datos utilizada	MySQL, Postgres	MySQL	Zope
Licencia	GNU GPL	GNU GPL	GNU GPL

Característica	**Drupal 5.1**	**Joomla! 1.0.7**	**Plone 2.5**
Sistema operativo	Cualquiera	Cualquiera	Cualquiera
Lenguaje de programación	PHP	PHP	Python
Acceso como root requerido para instalar la aplicación	No	No	No
Acceso a un shell requerido para instalar la aplicación	No	No	Si
Servidor Web con el que es compatible	Apache, IIS	Apache	Apache, IIS, Zope

Cuadro 2.1: Comparación de los requerimientos del sistema

Plone guarda por default la información en la base de datos de Zope, llamada ZODB, pero puede interactuar con la mayoría de bases de datos relacionales de código abierto y comerciales. Sin embargo la ZODB brindar varias ventajas que se mencionarán más adelante.

Plone viene con un servidor Web propio de Zope, llamado *ZServer*, el cual pueden consultar directamente los usuarios. Este servidor Web funciona bien, sin embargo no fue diseñado para ser un servidor robusto y no está siendo frecuentemente actualizado. En su lugar se pueden utilizar otros servidores Web, como Apache, que es más seguro y se actualiza más frecuentemente.

El servidor Web elegido para que se encuentre enfrente debe recibir las peticiones y redireccionarlas hacia un objeto llamado *Virtual Host Monster* (VHM), el cual se encarga de cambiar las URL generadas por Zope. La figura 2.2 muestra cómo se realiza esta configuración.

2.6.2. Seguridad

En la siguiente tabla se muestra una comparación de la seguridad entre Joomla!, Plone y Drupal.

En la siguiente tabla se utilizará el acrónimo ASC para indicar *Agregable sin costo*.

Característica	**Drupal 5.1**	**Joomla! 1.0.7**	**Plone 2.5**	**Peso**
Permite auditar mediante el registro de los usuarios que agregaron, actualizaron o borraron contenido	Si	No	Si	5
Captcha (prueba para determinar si un usuario es humano o no, consiste en que el usuario ingrese una serie de caracteres que se muestran en una imagen distorsionada)	ASC	Si	ASC	1

Característica	**Drupal 5.1**	**Joomla! 1.0.7**	**Plone 2.5**	**Peso**
Aprobación de contenido a publicar	Si	Si	Si	5
Verificación de email (mediante el envío de una llave por email para asegurarse que han ingresado uno válido)	Si	Si	Si	3
Permite definir diferentes privilegios por contenido	Si	No	Si	4
Soporta autenticación con Kerberos	No	No	ASC	1
Permite autenticación basada en LDAP	ASC	ASC	ASC	1
Histórico de autenticaciones al sistema (usuario y fecha)	Si	Si	ASC	4
Soporta autenticación vía NIS	No	No	ASC	1
Soporta autenticación vía NTLM	ASC	No	Si	1
Se le pueden agregar esquemas adicionales de autenticación más haya de los esquemas propietarios y LDAP	Si	No	Si	1
Provee mecanismos para notificar al administrador cuando se detecta un problema	No	No	ASC	4
Provee un lugar seguro para realizar pruebas sin afectar el resto del sitio	No	No	Si	4
Facilidad de ver qué usuario se encuentra en el sitio, qué está haciendo y cerrar su sesión si es necesario	Si	Si	ASC	3
Soporta autenticación vía SMB	No	No	ASC	1
Compatible con SSL	Si	No	Si	1
Autenticación con SSL (y después volverse a modo normal HTTP)	Si	Si	ASC	3
Puede ser configurado para que algunas páginas utilice SSL y otras no	No	No	No	1
Manejo de versiones	Si	Si	Si	3

Cuadro 2.2: Comparación de seguridad entre SMC

Cabe mencionar que Plone es el único de los tres que provee: autenticación con Kerberos, autenticación vía NIS, autenticación vía SMB, mecanismos para notificar al administrador cuando se detecta un problema y un lugar seguro para realizar pruebas.

La única característica de la tabla 2.2 que no provee Plone, al igual que los otros dos, es la posibilidad de mezclar páginas que utilicen SSL con páginas que no lo utilicen, y además ésta tiene un peso bajo. Las demás características Plone las provee, ya sea que las trae integradas o se le pueden agregar, mientras que los otros dos nos proveen algunas de ellas. Más aún, la mayoría de las características con mayor peso ponderado ya las trae integradas por default Plone.

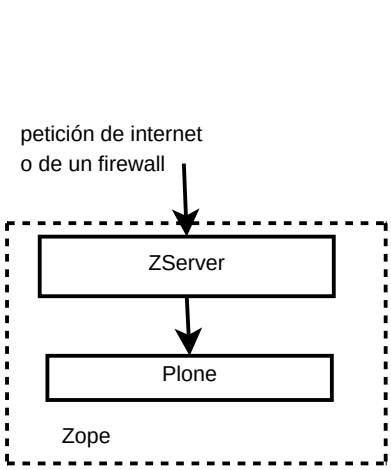

Figura 2.1: Configuración con el ZServer

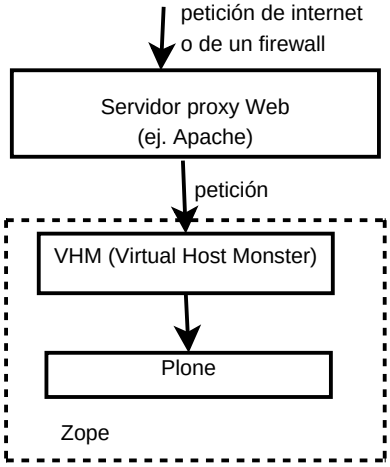

Figura 2.2: Servidor Web alterno

2.6.3. Soporte

En la siguiente tabla se muestra una comparación del soporte entre Joomla!, Plone y Drupal.

En la siguiente tabla se utilizará el acrónimo ASC para indicar *Agregable sin costo*.

Característica	**Drupal 5.1**	**Joomla! 1.0.7**	**Plone 2.5**	**Peso**
Tiene algún programa de certificación	No	No	No	1
Provee plantillas de código que facilitan a los nuevos desarrolladores escribir plugins	Si	No	Si	3
Hay libros o manuales comerciales disponibles	Si	Si	Si	3
Soporte comercial disponible	Si	Si	Si	1
Capacitación comercial disponible	Si	Si	Si	1
Tiene una comunidad de desarrolladores en línea especialmente para este producto de forma gratuita	Si	Si	Si	2
Tiene algún tipo de ayuda sensitiva al contexto integrada	Si	Si	Limitada	4
Puede ser extendido por medio de una API documentada y abierta	Si	Si	Si	4
Existen organizaciones de servicios profesionales disponibles que proveen servicios para este SMC	Si	Si	Si	1
Existen foros o pizarrones de mensajes públicos disponibles para este sistema	Si	Si	Si	5
Lista de mail público	Si	No	Si	4

Característica	**Drupal 5.1**	**Joomla! 1.0.7**	**Plone 2.5**	**Peso**
Tiene alguna forma de realizar pruebas de código para verificar que funcione correctamente	ASC	No	Si	4
Existen desarrolladores externos que proveen plugins para el sistema	Si	Si	Si	4
Hay conferencias sobre este sistema donde se discuten ideas, se obtiene capacitación, etc.	Si	Si	Si	3

Cuadro 2.3: Comparación de soporte entre SMC

En cuanto a soporte, Plone no cuenta con un programa de certificación, al igual que los otros dos. Además la ayuda sensitiva al contexto que provee Plone es inferior a la proporcionada por Joomla! y Drupal, ya que Plone la provee sólo de forma limitada.

En las demás características de la tabla 2.3 Plone es igual o superior a Drupal y Joomla!.

2.6.4. Facilidad de uso

En la siguiente tabla se muestra una comparación de la facilidad de uso entre Joomla!, Plone y Drupal.

En la siguiente tabla se utilizará el acrónimo ASC para indicar *Agregable sin costo.*

Característica	**Drupal 5.1**	**Joomla! 1.0.7**	**Plone 2.5**	**Peso**
Utiliza URLs amigables y fáciles de leer	Si	Si	Si	4
Se puede ajustar el tamaño de las imágenes que se suben al sistema	ASC	Si	Si	4
Tiene un lenguaje de macros que permite generar sistemas de navegación sin haber programado	ASC	Si	Si	4
Tiene alguna forma de subir información en masa	ASC	No	Si	4
Tiene revisor de ortografía	ASC	No	ASC	4
Tiene algún asistente para generar plantillas/temas/estilos o lo relacionado con la distribución visual sin conocer HTML/CSS	No	No	ASC	3
Los usuarios se pueden subscribir a varias secciones del sitio y recibir algún tipo de notificación	ASC	No	ASC	3
Tiene algún lenguaje de plantillas para controlar el aspecto visual integrando capacidades de llamadas a funciones programadas propias del sitio	Limitado	Si	Si	5

Característica	**Drupal 5.1**	**Joomla! 1.0.7**	**Plone 2.5**	**Peso**
Permite deshacer acciones	Limitado	No	Si	4
Tiene algún editor WYSIWYG	ASC	Si	Si	5

Cuadro 2.4: Comparación de facilidad de uso

Nótese que en todas las características mostradas en la tabla 2.4 Plone es igual o superior a Drupal y Joomla!.

2.6.5. Rendimiento

En la siguiente tabla se muestra una comparación del rendimiento entre Joomla!, Plone y Drupal.

En la siguiente tabla se utilizará el acrónimo ASC para indicar *Agregable sin costo.*

Característica	**Drupal 5.1**	**Joomla! 1.0.7**	**Plone 2.5**	**Peso**
Tiene mecanismos de cache avanzado	Si	Si	Si	5
Réplica de la base de datos (lee de base de datos esclava y escribe en una base de datos maestra)	No	No	Si	3
Permite balancear la carga entre diferentes servidores	Si	No	Si	4
Páginas en cache	Si	Si	Si	4
Puede exportar el contenido como páginas estáticas (en HTML) para que servidores de cache o servidores Web lo puedan utilizar	No	No	ASC	3

Cuadro 2.5: Comparación de rendimiento

Nótese que Plone es el único que soporta replica de la base de datos y exportar su contenido como páginas estáticas. Además de que en las otras características de la tabla 2.5 es igual o superior a Drupal y Joomla!.

2.6.6. Manejo

En la siguiente tabla se muestra una comparación del manejo entre Joomla!, Plone y Drupal.

En la siguiente tabla se utilizará el acrónimo ASC para indicar *Agregable sin costo.*

Característica	**Drupal 5.1**	**Joomla! 1.0.7**	**Plone 2.5**	**Peso**
Manejo de publicidad	ASC	Si	ASC	3
Calendarización de contenido para ser agregado o removido	ASC	Si	Si	3
El contenido puede ser editado directamente en el lugar donde será publicado	Si	Si	Si	4
El sistema puede ser completamente administrado desde un explorador Web	Si	Si	Si	5
Permite tener sub-sitios dentro del sitio con su propia navegación y jerarquía de contenidos	Si	Si	Si	4
Tiene mecanismos para transportar estilos, plantillas, etc., entre sitios de forma que se pueda crear un tema en un sitio y reusarlo en otros	Si	Si	Si	4
Tiene un sistema de basura que permite recobrar contenido borrado	No	Si	ASC	2
Tiene un sistema de estadísticas que permite ver cosas como contenido visto o número de usuarios por periodo de tiempo, etc.	Si	Si	ASC	4
Tiene un sistema de workflow integrado que puede ser usado para el manejo del negocio (más allá de sólo aprobar contenido)	Limitado	No	Si	5

Cuadro 2.6: Comparación de manejo

Nótese que Plone es el único que trae un sistema integrado de workflow completo para el manejo de la lógica del negocio, ya que Drupal lo implementa de forma limitada y Joomla no lo implementa. Además en [9] se menciona que el manejo de workflow es una característica que todo sistema manejador de contenido debería implementar y es un punto al que se le dio bastante peso por ser importante.

2.6.7. Interacción con otros sistemas

En la siguiente tabla se muestra una comparación de la interacción con otros sistemas entre Joomla!, Plone y Drupal.

En la siguiente tabla se utilizará el acrónimo ASC para indicar *Agregable sin costo.*

Característica	**Drupal 5.1**	**Joomla! 1.0.7**	**Plone 2.5**	**Peso**
Puede exportar el contenido a RSS/XML para que otros sitios lo puedan publicar	Si	Si	Si	3
Permite subir contenido vía FTP	Limitado	ASC	Si	2

Característica	**Drupal 5.1**	**Joomla! 1.0.7**	**Plone 2.5**	**Peso**
Soporta codificación UTF-8 para habilitar sitios con múltiples lenguajes para evitar el uso diferentes páginas codificadas para cada lenguaje	Si	Limitado	Si	4
Cumple con el estándar WAI	Limitado	No	Si	3
Permite subir contenido vía WebDAV	No	No	Si	2
Cumple las especificaciones para XHTML	Si	No	Si	3

Cuadro 2.7: Comparación de interacción con otros sistemas

Cabe mencionar que Plone es el único que cumple con todas las características de la tabla 2.7.

Según las tablas anteriores Plone cumple con la mayoría de características, ya sea que las trae integradas o se le pueden agregar de forma gratuita. Sin embargo, en el sitio *CMF Matrix* Plone fue destacado por su difícil curva de aprendizaje, no obstante, fue elogiado por su poder, flexibilidad y personalización que ofrece.

En [25] se indica que para hacer la elección de un SMC hay que definir primeramente los objetivos y necesidades del sitio para formar una lista de requerimientos. No basta con utilizar el SMC con más éxito en la Web, sino el que más se acople a los requerimientos propios de una organización.

Plone fue considerado el más completo cumpliendo de forma más completa lo requerido por el IMATE, que fue puesto a lado de cada característica por medio de un peso; y además cumple con un mayor número de características que Joomla! y Drupal, por eso fue elegido.

2.7. Aplicaciones Web

Una aplicación Web es un programa de computadora que los usuarios invocan utilizando un explorador Web. Las aplicaciones Web crean sus páginas de forma dinámica por medio de un programa que puede ser escrito en varios lenguajes de programación.

En los sitios que son construidos dinámicamente no es necesario visitar página por página para poder actualizar el contenido o su diseño, sino que permiten definir el diseño y distribución del contenido para un conjunto de páginas del sitio.

2.8. Servidor de aplicaciones Web

Un ambiente de trabajo que permite crear aplicaciones Web es conocido como servidor de aplicaciones Web. La mayoría de estos servidores de aplicación permiten desarrollar varias tareas descritas en [15]: adaptar la presentación de forma dinámica, proveer características de búsqueda, facilidades de integración de bases de datos, manejo del contenido y

presentación, construir aplicaciones de comercio electrónico, proveer mecanismo de control de acceso, integración con diversos sistemas y construir sistemas manejadores de contenido.

2.9. Zope

Zope es un servidor de aplicaciones Web, distribuido bajo una licencia de código abierto[3], y se puede obtener de forma gratuita. El nombre Zope se deriva de *Z Object Publishing Environment* (ambiente de publicación de objetos Z), donde la 'Z' no tienen ningún significado en particular. La mayor parte de Zope está escrita en Python, con algunas partes críticas en C. Dado que Python es un lenguaje orientado a objetos, también Zope hereda esta característica.

Zope incluye todos los componentes necesarios para iniciar a desarrollar una aplicación, como manejador de bases de datos y explorador de Internet, pero no es necesario utilizar todos los componentes de Zope, además existen muchos productos hechos por personas externas al equipo de desarrollo de Zope que se pueden agregar, y la mayoría de éstos de forma gratuita.

Zope es una plataforma de desarrollo colaborativo, que provee herramientas para que equipos de desarrollo puedan trabajar sin interferir unos con otros, como historial, manejo de versiones, recuperación de errores, entre otras. Además puede ser extendido utilizando Python [4]. Corre en los sistemas operativos más populares: Linux, Windows NT/2000/XP/2003, Solaris, FreeBSD, NetBSD, OpenBSD y Mac OS X.

Tiene una estructura jerárquica, es decir, un objeto puede contener otros objetos, y éstos a la vez otros; y provee un sistema de control de acceso para estos objetos, el cual consta de un conjunto de permisos, los cuales pueden variar dependiendo de las funciones del objeto.

A cada tipo de usuario se le puede asignar un rol, y la forma en la que se definen los permisos de cada objetos es por medio de una lista en la que se definen los permisos que tiene cada rol. La figura 2.3 muestra una parte de la lista de permisos relacionados con un objeto. En ésta se pueden ver roles como: *Anonymous, Authenticated, Manager y Owner.* Además se pueden ver permisos como: *Access content information, Change permissions* y *Copy or Move*

Se le puede asignar un permiso a un rol poniéndole una paloma en la intersección del permiso y el rol, o se puede heredar el permiso del objeto que lo contiene al ponerle una paloma del lado izquierdo donde dice: *Acquire permission settings.* De esta forma si se quiere que todos los objetos que se encuentra dentro de un objeto padre tengan los mismos permisos que éste, basta con definir los permisos en el objeto padre y todos sus hijos heredaran los mismos permisos, a menos que se modifique la lista de permisos de cada hijo.

[3]Está distribuido bajo la licencia *Zope Public Licence (ZPL)*, en `http://www.zope.org/Resources/ZPL` se puede encontrar más información sobre las condiciones de uso.

[4]Python es un lenguaje de programación de alto nivel, interpretado, orientado a objetos, de propósito general, multiplataforma, fácil de aprender y con muchas librerías disponibles.

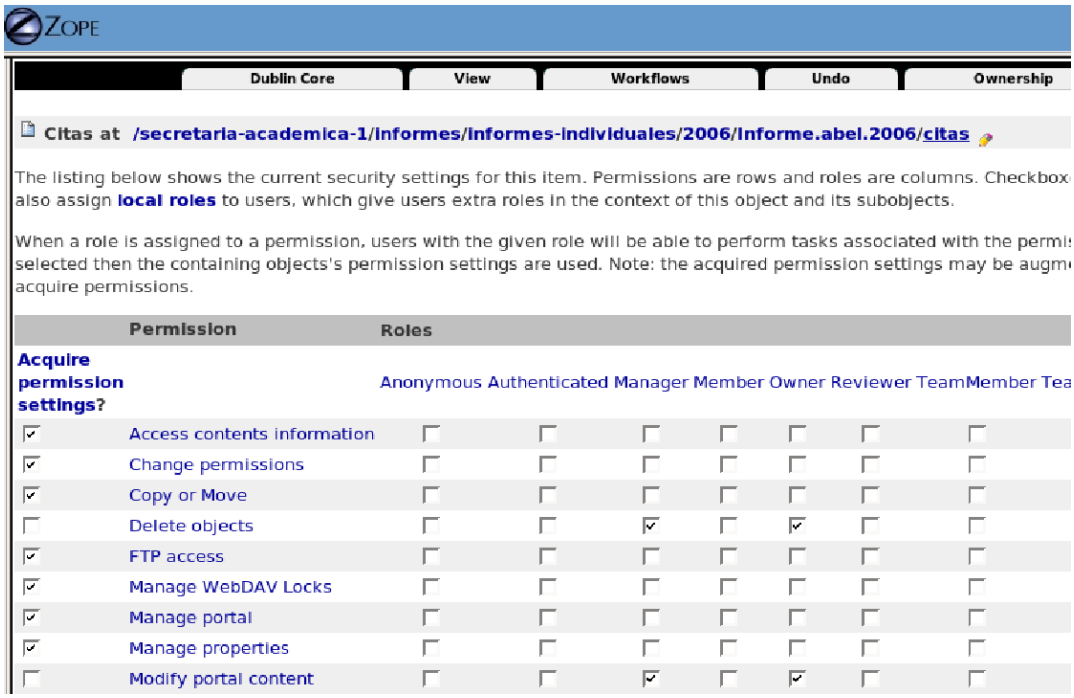

Figura 2.3: Lista de permisos asociados a un objeto

2.10. Plone

Plone es un Sistema Manejador de Contenido para Web, de código abierto, gratuito y distribuido bajo la licencia *GNU GPL*. Permite construir sitios ricos en contenido rápidamente. Por sus muchas características se puede decir que es comparable o mejor, a sistemas manejadores de contenido que cuestan cientos de miles de dólares [27]. Además, es considerado como uno de los mejores [5] SMC disponibles hoy en día.

La filosofía de que sea código abierto tiene la finalidad de que muchos usuarios, desarrolladores, expertos en usabilidad, escritores técnicos, traductores y diseñadores gráficos trabajen en el desarrollo de Plone. Actualmente se tiene una comunidad de cientos de desarrolladores y compañías que le dan soporte a Plone alrededor del mundo.

Plone está construido sobre Zope, por lo que la mayoría de las características antes mencionadas también aplican para Plone.

A diferencia de Zope, Plone está listo para usarse, basta con configurar algunas opciones, como la forma en que se verá el sitio, usuarios y permisos, para poder empezar a utilizarlo.

Una de las herramientas que se le pueden agregar a Zope es el CMF [6] *(Content Mana-*

[5] En abril de 2006 obtuvo un premio por *eWeek* (`http://www.eweek.com`) por ser una de las mejores soluciones de código abierto para portales de compañías e Intranets.

[6] En `http://www.zope.org/Products/CMF/` se puede obtener la última versión del CMF e información

gement Framework), que provee herramientas que permiten crear SMC. Plone es una capa sobre CMF que se encuentra corriendo sobre Zope, por lo que la mayor parte de tareas de administración requieren utilizar la interfaz de Zope, llamada *ZMI* (Zope Management Interface).

A continuación se muestran algunas características de Plone mencionadas en [19]:

- Su interfaz está traducida a más de 35 idiomas y es fácil agregar una traducción propia.
- Ofrece altos niveles de usabilidad y accesibilidad ya que es compatible con el estándar gubernamental *WAI-AAA* [7] y la sección 508 de los Estados Unidos [8]. Además, esto permite obtener mejores búsquedas, al estilo Google.
- Ya que es un SMC, separa el contenido de las plantillas utilizadas para presentarlo. Estas plantillas son escritas principalmente en HTML, ZPT (Zope Page Templates - Plantillas de Páginas Zope) [9] y hojas de estilo (CSS).
- Tiene un completo sistema de registro de usuarios, con el cual se puede personalizar la interfaz de cada usuario.
- Utiliza workflows que pueden ser configurados con herramientas gráficas por medio de Web.
- Tiene un robusto control de acceso, que se se hereda de Zope, con el cual se define quién va hacer qué y dónde.
- Ya que es de código abierto, cualquiera con el conocimiento necesario puede extender casi cualquier aspecto que se requiera. Además hay muchos paquetes y herramientas para Plone que se le pueden agregar, de las cuales la mayoría son gratuitas [10]. También existen herramientas que permiten modificar o generar código para Plone de manera sencilla utilizando UML.
- Utiliza por defecto una base de datos orientada a objetos, propia de Zope, pero brinda la posibilidad de utilizar otras, inclusive relacionales.

al respecto de forma gratuita.

[7] WAI (Web Accesibility Initiative - Iniciativa para la Accesibilidad Web) es un estándar con una serie de recomendaciones a seguir a la hora de diseñar un sitio Web para que éste sea accesible. Estas recomendaciones se dividen en tres categorías A, AA y AAA, donde la categoría AA implica que se cumple la A, y la AAA implica que se cumple la AA. El estándar AAA explica cómo hacer accesible el contenido Web para personas con discapacidades y es un estándar internacional. En `http://www.w3.org/` se puede obtener más información al respecto.

[8] La sección 508 es una ley que incluye estándares de accesibilidad en la que se indica que las agencias federales deben hacer sus tecnologías de la información accesible para personas con discapacidades. En `http://www.section508.gov` se puede obtener más información al respecto.

[9] ZPT es un lenguaje que permite definir la presentación de una página Web dinámicamente, por medio de etiquetas XML incrustadas dentro del código HTML de la página, es similar a PHP o JSP. La página es ejecutada en el servidor y el resultado es enviado al explorador Web.

[10] En `http://www.plone.org` hay se pueden encontrar repositorios gratuitos de Plone.

A pesar de sus beneficios, Plone tiene algunas limitantes que se mencionan en [22]:

- Se requiere mucho poder de CPU para ensamblar las páginas y tratar con la memoria conforme los objetos son cargados desde la base de datos. Por lo que Plone nunca será tan rápido en generar páginas Web como lo puede hacer un servidor Web puro, que es algo que los usuarios finales pueden notar. El tiempo que tarda en desplegar las páginas se debe a que realiza varias tareas importantes como comprobación de permisos para controlar el acceso al contenido, cargar los objetos que se van a utilizar y mostrarlos de acuerdo al perfil del usuario.
- Aunque se puede utilizar Apache en frente de Zope para dar mayor flexibilidad al servir a otras aplicaciones o páginas estáticas, y proveer seguridad, esto implica que se disminuya un poco el tiempo de respuesta.
- Depende de varios componentes principales actualizados constantemente por separado, como lo son: Python, Zope, Apache y el mismo Plone. Esto puede causar problemas de compatibilidad entre los componentes agregados y los productos previamente instalados, por lo cual se debe evitar realizar actualizaciones lo más posible para evitar inestabilidad y fallas en el sistema. El desarrollo en paralelo de cada uno de los componentes permite una evolución en paralelo, pero no coordinada, lo que puede dificultar la compatibilidad.

2.11. Bases de datos relacionales y orientadas a objetos

Las bases de datos relacionales proveen un modelo simple para organizar datos dentro de tablas, y la mayoría puede tratar con grandes cantidades de datos de manera eficaz. Debido a que su modelo de datos es simple, las bases de datos son fáciles de entender, cuando menos para problemas pequeños. Además, son neutrales en cuanto al lenguaje de programación, puesto que los datos son almacenados en tablas, que es independiente del lenguaje. Desafortunadamente, estas bases de datos se pueden convertir en algo engorroso cuando el dominio del problema no encaja en una simple organización tabular.

Una aplicación orientada a objetos que utiliza una base da datos relacional, debe acceder a la base de datos e inicializar sus variables leídas de las tablas antes de poder utilizarlas; de igual forma debe acceder a la base de datos para escribir en las tablas los datos modificados cuando sea necesario. Esto impone significante carga en el desarrollador de la aplicación y en la lógica de ésta al tener que estar traduciendo información entre un modelo relacional y uno orientado a objetos.

Otra opción es mantener una estructura tabular dentro del programa en vez de poblar los objetos con los valores de las tablas. La desventaja de esto es que se forza la aplicación a ser escrita según el modelo relacional en vez de utilizar el modelo orientado a objetos perdiendo propiedades como la encapsulación de datos y la lógica de asociación.

En [13] se menciona que las bases de datos relacionales difícilmente encajan con las necesidades de los sistemas de información complejos, las cuales requieren que la información sea modelada en tablas. Sin embargo, las bases de datos orientadas a objetos permiten un modelado más natural de entidades y relaciones entre éstas, con lo que se obtiene un diseño

más integrado con la lógica de la aplicación. Además, con los sistemas orientados a objetos se pueden implementar objetos complejos que soporten diferentes tipos de contenido, como multimedia, al definir nuevas clases que soporten este tipo de información.

2.12. Base de datos de ZOPE *(ZODB)*

La ZODB es una base de datos para Python, orientada a objetos, que permite un alto grado de transparencia. Las aplicaciones en Python se ven beneficiadas de esta base de datos al no tener que hacer cambios o siendo éstos mínimos en la lógica de la aplicación.

La ZODB es utilizada para almacenar todas las páginas, archivos y objetos creados. Casi no requiere modificar la configuración o darle mantenimiento. Almacena los objetos sobre múltiples registros, donde cada objeto almacenado tiene su propio registro, y cuando éste es modificado, sólo su registro es afectado. Todos los objetos tienen un identificador que los distingue de forma única dentro de la base de datos [13].

Cada petición realizada a la ZODB por medio de la Web es manejada como una trasacción separada y sólo es confirmada si no ocurre ningún error. Pero además la ZODB provee un mecanismo para deshacer múltiples transacciones, aunque ya hayan sido confirmadas anteriormente. Esto puede provocar que la base de datos crezca mucho al guardar diferentes copias de un mismo objeto, pero se puede indicar que las copias anteriores a ciertos días se borren. Cabe mencionar que esto no implica ninguna carga al desarrollador de aplicaciones, pues es transparente.

La base de datos tiene un objeto raíz por diseño. Una aplicación típicamente tiene un objeto raíz y todos los demás objetos son accedidos a través de éste por medio de sus atributos o métodos, es decir, regularmente una aplicación forma una estructura de árbol y para alcanzar cualquier hoja hay que iniciar el recorrido a partir de la raíz.

La ZODB no impone ninguna restricción a la aplicación sobre la organización de los objetos, tampoco es necesaria ninguna noción sobre bases de datos relacionales, sino que los aplicaciones son libres de imponer la organización dentro de la base de datos, hasta se puede implementar una base de datos relacional sobre ZODB.

La principal actividad que debe realizar el administrador con respecto a la ZODB es la de purgarla, ya que la base de datos guarda varias copias de un mismo objeto cada que éste es modificado. Lo cual permite que uno pueda regresar a una copia anterior de un objeto, pero provoca que la base de datos crezca bastante. La tarea del administrador es de indicarle a Plone que borre copias de un objeto anteriores a cierta fecha; por ejemplo, para guardar sólo modificaciones realizadas en la última semana o último mes.

Esta característica de la ZODB brinda varios beneficios a los investigadores, porque si cometen algún error, pueden volver a una copia guardada anteriormente. Por ejemplo, si un investigador importa artículos o borra actividades de forma incorrecta, tiene la posibilidad de deshacer esa acción y regresar a copias anteriores.

2.13. Productos

Los productos son una de las claves principales del CMF y Plone que permiten extender la funcionalidad de Zope. Con los productos se pueden definir nuevos tipos de contenido, herramientas y funcionalidad al sitio. Hay aproximadamente 500 productos [11] disponibles al público en general dentro de la página de Plone [7] con una gran variedad de funcionalidades.

Una vez que un producto ha sido instalado, éste puede hacer uso de toda la funcionalidad del sitio, aún de la provista por otros productos. De esta forma es posible que algunos productos sean dependientes de otros, es decir, un producto utiliza la funcionalidad de otro y sin ese no puede ser instalado.

Si no se encuentra disponible algún producto que cumpla con las necesidades requeridas se puede crear uno, y para esto se han creado diferentes herramientas que facilitan esta tarea.

Hay dos herramientas muy útiles para desarrollar productos: los arquetipos y ArchGenXML. El primero es un producto de Plone con varios componentes útiles en la creación de contenido, y el segundo es un generador de código que toma como entrada un modelo UML en formato XMI y lo convierte en código Python basado en los arquetipos. En la sección 4.2 se explica más sobre estas herramientas.

Un producto es un componente de software el cual encapsula alguna funcionalidad expuesta a través de interfaces. Dado que en Plone a los componentes también se les llama productos, a lo largo de esta tesis se le llamará indistintamente componente o producto.

2.14. Vistas y lógica del negocio

En esta tesis se referirá a las interfaces que permiten interactuar con el usuario como *vistas*. Los productos basados en los arquetipos utilizan frecuentemente las vistas propias de los arquetipos, las cuales permiten editar y visualizar contenido, modificar algunos metadatos, compartir contenido, etc.

No es aconsejable mezclar la lógica dentro de las vistas, como se menciona en [19], y para esto Plone ofrece diferentes maneras de manejarlo, como se muestra a continuación:

- **Expresiones de plantilla.** Son pequeñas expresiones que permiten insertar lógica dentro de las vistas, pero de forma restringida.
- **Scripts de Python (internos).** Son scripts escritos en Python, dentro de Plone, y ejecutados en un ambiente restringido.
- **Scripts de Python (externos).** Son scripts escritos en Python, fuera de Plone, y ejecutados en un ambiente no restringido.
- **Métodos de los objetos.** Son los métodos de los objetos que se encuentran en el sitio.

[11] Este dato fue obtenido el 26 de Junio de 2007.

Las vistas y los scripts pueden hacer uso de a una variable llamada *context*, la cual hace referencia al objeto sobre el cual se invoca. De esta forma una misma vista o script se puede reutilizar varias veces en diferentes contextos (con diferentes objetos).

Para invocar una vista o script primero se debe especificar la URL del objeto seguida del nombre de la vista o script. Por ejemplo, para utilizar la vista *base_view* con un objeto que tiene la URL *http://misitio/objetoX*, se debería invocar utilizando la URL *http://misitio/objetoX/base_view*.

Además, existen vistas y scripts controladores, los cuales permiten definir el orden en que se irán invocando los scripts y vistas dependiendo de ciertas circunstancias. Por ejemplo, en caso de que un usuario presione un botón dentro de una vista o que un script regrese cierto valor se va a seguir un flujo u otro. De esta forma las vistas y scripts controladores permiten definir la lógica del negocio sin mezclar la lógica con las vistas y de una forma ordenada.

2.15. Arquetipos

Para entender bien los arquetipos es necesario conocer el significado de *campo*, *widget* y *esquema*, que se definen a continuación para el ámbito de Plone:

- **Campo**. Es una instancia de una clase que se encarga de almacenar y recuperar cierta información sobre un contenido. Tiene varios atributos configurables que definen la forma como se va a manejar la información.

  Dentro de la configuración de los campos se puede definir: la forma en que será visualizado el valor almacenado dentro del propio campo, permisos de lectura y escritura, y forma en que se va a manejar el campo dentro de los catálogos, entre otras.

  Existen diferentes tipos de campos y dependiendo del que se escoja es el tipo de información que va a almacenar. Algunos ejemplos de campos son: *StringField* (para guardar cadenas de caracteres), *FileField* (para guardar archivos), *TextField* (para guardar textos largos) y *DateTimeField* (para guardar fechas).

- **Widget**. Es una instancia de una clase que se almacena como un atributo de un campo, el cual define la forma en que será visto o editado el valor de dicho campo.

- **Esquema**. Es una lista de campos.

Según lo que se definió arriba, un esquema tiene uno o varios campos, y un campo tiene un widget que define como será visto ese campo. La figura 2.4 muestra la relación entre estos tres.

Los arquetipos son otra aplicación de Zope y el CMF que simplifica la creación de nuevos tipos de contenido [12]. La clave de los arquetipos es que el contenido está basado en un *esquema*, además permiten hacer referencias robustas entre objetos mediante un identificador único que los diferencía de cualquier otro objeto del sistema. Por ejemplo,

[12] La creación de nuevos tipos de contenido sin utilizar los arquetipos no es una tarea sencilla, que requiere de un mayor conocimiento de Zope y de cómo funciona el CMF.

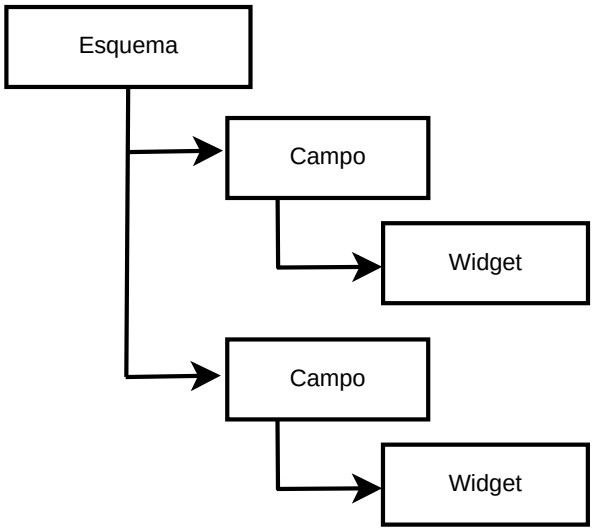

Figura 2.4: Relación entre campo, widget y esquema

si se quiere tener un contenido para mostrar archivos, se puede crear un esquema con los siguientes campos: título, descripción y archivo fuente.

Un esquema es definido una vez dentro de una clase de arquetipo y utilizado por todas las instancias de esa clase. De esta forma se pueden crear diferentes tipos de contenido, cada uno basado en un esquema diferente.

Al utilizar arquetipos se obtienen las siguientes ventajas, mencionadas en [17]:

- *Automáticamente se generan los formularios.* Los arquetipos traen por default definidas varias vistas (para editar y mostrar los campos del esquema, entre otras) y scripts integrados para procesar la información. De esta forma se puede diseñar un tipo de contenido sin preocuparse por las vistas o métodos utilizados por éstas.

- *Provee una biblioteca de diferentes tipos de campos, widgets y validadores de campos.* Existen una variedad de campos y widgets que definen cómo se verán estos campos. Además para validar la información a ingresar dentro de los campos existen diferentes validadores para asegurar que un campo sólo tenga información válida.

- *Realiza una integración entre los campos, widgets y validadores de campos.* Basta con definir el esquema del contenido para que esta integración sea realizada de forma transparente para el programador.

Para crear un contenido basado en los arquetipos se puede hacer creando una clase que herede de alguna de las siguientes:

- **BaseContent**. Se utiliza éste si el objeto no se debe comportar como una carpeta guardando otros objetos dentro.

- **BaseFolder**. Se utiliza si el objeto guarda otros objetos comportándose como una carpeta.

- **BaseBTreeFolder**. Se utiliza si el objeto va a contener cientos de miles o millones de objetos dentro.

También se debe definir un esquema, que es la lista de campos del contenido. Hay tres esquemas base que se pueden utilizar: *BaseSchema*, *BaseFolderSchema*, y *BaseBTreeFolderSchema*, dependiendo de si se heredó de la clase BaseContent, BaseFolder o BaseBTreeFolder, respectivamente.

Los tres esquemas base ya incluyen algunos campos: un identificador para diferenciar el contenido dentro de la carpeta en que se encuentra y el título del contenido. Además tiene otros campos estándares para guardar metadatos sobre los contenidos, como el nombre del creador del contenido, fecha de última modificación y fecha de creación, entre otras.

Una vez definidos los campos propios de un contenido se debe crear un esquema con éstos y sumárselos al esquema base que le corresponda, obteniendo un esquema con campos estándares y campos propios del contenido. Es decir:

$$\textit{Esquema del contenido} = \textit{Esquema base} + \textit{Esquema con campos propios}$$

Con esto ya es suficiente para comenzar a utilizar el contenido con la funcionalidad y vistas estándares que proveen los arquetipos.

2.16. Catálogo de Zope *(ZCatalog)*

El *ZCatalog* es una máquina de búsqueda incluida dentro de Zope que permite la rápida localización de todo tipo de objetos mediante la previa organización de éstos.

Para que los objetos o contenidos se puedan catalogar, primero se deben definir índices que le permiten saber al catálogo cómo organizar estos objetos.

Existen varios tipos de índices, a continuación se muestran algunos de los más utilizados:

- **FieldIndex**. Busca objetos por un valor específico. Es útil para búsquedas de objetos, números o cadenas específicas.

- **KeywordIndex**. Busca objetos por una colección de valores específicos. Es similar al índice *FieldIndex* sólo que en vez de buscar por un valor específico, busca por varios.

- **DateIndex**. Es una versión de *FieldIndex* pero optimizada para valores de fecha y hora.

- **ZCTextIndex o TextIndex**. Divide el texto por palabras. Cuando se hace búsqueda por éste índice los resultados son ordenados según el número de coincidencias en las palabras del texto, permitiendo tener búsquedas al estilo Google.

Al momento de hacer una búsqueda con el catálogo no se busca objeto por objeto dentro del sitio, esto sería muy lento, en su lugar primero se catalogan los objetos de acuerdo a los índices y se almacena el resultado para no tener que cargar cada objeto en memoria y buscar uno por uno. De esta forma se obtienen búsquedas más rápidas [16].

Cada que un objeto es modificado debe de avisarle al catálogo para que éste lo reindexe con los nuevos valores que obtuvo, para tener en todo momento un catálogo actualizado.

Cuando se consulta un catálogo se obtiene como resultado una lista de objetos, que no son propiamente los contenidos catalogados, sino que son unos objetos más pequeños con referencias a los contenidos buscados. A estos objetos les llamaremos *objetos de catálogo*.

Los *objetos de catálogo* tienen datos sobre los contenidos a los cuales hacen referencia, conocidos como metadatos dentro del catálogo. Estos metadatos evitan tener que cargar en memoria los contenido al guardar sólo una parte de éste. Por ejemplo, el título de los contenidos de un sitio es un dato muy utilizado y no vale la pena cargar todos los contenidos en memoria sólo para consultar su título o algunos otros datos. Es más eficiente guardar el título como un metadato y de esta forma al consultar el catálogo se evita cargar el objeto completo en memoria, cargando sólo una parte de éste, ya que los contenidos suelen ser mucho más grandes que los *objetos de catálogo*.

Aunque los metadatos son muy útiles, en [15] se menciona que existe un sacrificio al utilizar muchos. Entre más metadatos se utilicen en el catálogo, éste se vuelve más grande y lento debido a que utiliza más memoria afectando así el desempeño. Por eso es recomendable agregar entre los metadatos sólo aquellos que se vayan a utilizar muy frecuentemente en las búsquedas, además de procurar que no sean datos muy grandes como un documento completo.

Los *objetos de catálogo* además de tener metadatos sobre los contenidos, tienen algunas funciones básicas. Entre éstas se tiene la función *getObject* que permite obtener el contenido referenciado, aunque provocando que éste se cargue en memoria, pero esto llega a ser necesario en ocasiones.

Plone trae un catálogo integrado llamado *portal_catalog*, para indexar todos los objetos del sitio. Por default, ya tiene varios índices y metadatos definidos, pero se le pueden agregar más. El *portal_catalog* hereda del *ZCatalog* y define nuevos métodos y atributos utilizados por los contenidos de un sitio Plone.

Los arquetipos utilizan por default el *portal_catalog* para indexar su información, pero proveen herramientas para poder utilizar otros catálogos de forma simultánea.

2.17. Algunos estándares

Dentro del sistema que se desarrolló se utilizaron algunos estándares, además de los que ya implementa Plone, que se describen a continuación:

BibTex

BibTex es un estándar de Latex utilizado para hacer referencias bibliográficas y darles formato. Fue creado por Oren Patashnik y Leslie Lamport en 1985.

BibTex permite citar información de forma fácil y consistente separando la información bibliográfica de la información de la presentación. Entre los tipos de información que se pueden referenciar se encuentran: artículos, libros, conferencias, manuales, tesis de maestría, tesis de doctorado, entre otros.

Este estándar es muy utilizado por revistas, comunidad científica y por quien quiere hacer referencia a trabajos de otras personas.

MSC

La MSC es una clasificación de materias de matemáticas desarrollado en forma conjunta por los editores de *Mathematical Review (MR)* y *Zentralblatt MATH (Zbl)* desde 1991.

La MSC está compuesta por 63 áreas con más de $5,000$ códigos divididos en tres niveles: el primero tiene dos dígitos, el segundo tiene tres dígitos y el tercero de cinco dígitos. Esta clasificación es muy utilizada por varias revistas, las cuales le piden a sus expositores que listen los códigos de la MSC en sus artículos.

2.18. Conclusiones

El manejo de información es fundamental para cualquier empresa y debido a ésto la mayoría de las organizaciones han implementado un sitio Web para compartir, distribuir o intercambiar información, enfrentando el problema de cómo manejar el contenido que tienen.

Los sistemas manejadores de contenido (SMC) son la respuesta a los problemas de creación y manejo de información dentro de un sitio, lo cuales permiten crear, manejar, distribuir y buscar contenido; cubriendo todo el ciclo de vida de las páginas Web, desde la creación y publicación de contenido hasta que es quitado y almacenado.

Plone es un SMC muy completo y comparado con muchos otros, es de los mejores por sus características. Está implementado sobre Zope y un producto conocido como CMF.

El sistema curricular se va a desarrollar sobre Plone, utilizando la base de datos de Zope (ZODB), que es orientada a objetos. Para esto se va a crear un producto basado principalmente en los arquetipos, lo cuales permiten crear contenidos basados en un esquema definido.

Al implementar el sistema sobre Plone se busca obtener un sistema robusto aprovechando los productos o componentes disponibles, los cuales ya han sido probados por muchas personas y depurados.

Capítulo 3

Análisis, diseño e integración

3.1. Introducción

La metodología que se va a seguir para el desarrollo del Sistema de Captura y Recolección de Información Curricular del IMATE, que de ahora en adelante nos referiremos a él con las siglas *SCRICI*, es una metodología para el desarrollo basado en componentes propuesta por Pressman [23]:

- Establecer los requisitos del sistema
- Establecer un diseño arquitectónico
- Examinar componentes existentes para integrarlos al sistema
- Tratar de modificar o eliminar aquellos requisitos que no se puedan implementar con los componentes existentes
- En caso de ser necesario, desarrollar componentes necesarios y no existentes.
- Desarrollo del sistema integrando los componentes

En este capítulo se va a seguir la metodología arriba propuesta. Se inicia mostrando los requisitos del sistema, se muestran dos arquitecturas: una física y otra de capas. Después se muestran los componentes existentes que se utilizaron y los que se implementaron. Finalmente se muestra cómo se implementaron los casos de uso para satisfacer los requerimientos y cómo se integraron los componentes.

3.2. Requerimientos del sistema

Para establecer los requisitos del sistema se hizo una toma de requerimientos la cual se dividió en requerimientos generales y requerimientos por perfil operacional. Finalmente se hicieron casos de uso sobre los requerimientos obtenidos.

3.2.1. Requerimientos generales

A continuación se muestran los requerimientos para el sistema SCRICI:

1. Debe haber un lugar para cada investigador donde éste pueda guardar exclusivamente lo relacionado con sus actividades curriculares. En lo que sigue de esta tesis nos referiremos a este lugar como *MIC* o *Módulo de Información Curricular*.

2. Cada investigador debe tener en su carpeta personal de trabajo, también llamada *home*[1], un MIC. Este MIC es donde el investigador va a ir almacenando sus actividades académicas.

3. Las actividades deben estar agrupadas según el rubro al que correspondan, siendo un rubro un tipo de actividad. Con esto se busca que las actividades se puedan localizar fácilmente teniendo bien organizada la información.

4. Los rubros, referentes a las actividades de los investigadores, que debe tener cada MIC se listan a continuación:

   - Proyectos
   - Artículos
   - Libros y monografías
   - Capítulos de libros
   - Informes y reportes
   - Tesis
   - Tutorías
   - Patrocinios
   - Visitas a otras instituciones
   - Asistencias a congresos, coloquios y talleres
   - Conferencias plenarias o magistrales impartidas
   - Conferencias impartidas y participación en mesas redondas
   - Participación o dirección de seminarios de investigación y de formación
   - Organización o coordinación de congresos, coloquios y talleres
   - Investigadores invitados
   - Cursos o cursillos impartidos
   - Producciones de material de divulgación

[1]Dentro de Plone todos los usuarios tienen una carpeta personal, llamada *home* que es donde ellos pueden agregar, modificar y eliminar contenido para el cual tengan los permisos de hacerlo. Esto funciona de forma similar a Unix donde también los usuarios tienen una carpeta de trabajo con tres permisos: lectura, escritura y ejecución. A diferencia de Unix, Plone maneja una gran variedad de permisos para cada tipo de objeto.

- Traducciones
- Construcciones de prototipos o tecnologías transferidas
- Participaciones en comités editoriales
- Participaciones en consejos editoriales
- Revisiones de trabajos
- Participaciones en comisiones académicas
- Desarrollos de infraestructura institucional
- Coordinaciones de cursos, talleres y programas docentes
- Elaboraciones de planes de estudio
- Materiales didácticos
- Trabajos institucionales ligados a la docencia
- Distinciones académicas recibidas
- Puestos académico-administrativo
- Patentes

5. Dentro del rubro *libros y monografías* deben haber los siguientes tipos de actividades: libros escritos por el investigador, libros editados por el investigador, memorias de congresos y monografías.

6. Dentro de los proyectos se pueden escribir algunos artículos, por lo tanto se debe poder relacionar un artículo con uno o varios proyectos donde haya participado el investigador.

7. Los proyectos son una actividad donde pueden participar varios miembros del IMATE, y no se desea que esta información se repita por cada miembro. Si un investigador reporta un proyecto como responsable e incluye a otros miembros del IMATE, estos miembros ya no tienen que volver a repetir esta información, sino que automáticamente deben de ver reflejado dentro en su MIC como si ellos lo hubieran creado; pero sólo lo puede modificar el responsable o corresponsable, y sólo lo puede borrar el responsable del proyecto. Es decir sólo el responsable del proyecto es el que debe poder agregar los proyectos para evitar redundancia.

8. Las actividades se deben clasificar por fecha para saber en qué periodo se realizaron. Algunas actividades pueden durar más de un año por lo que se debe de tener una fecha inicial y una final, otras sólo la fecha de realización puesto que duran a lo más un año y otras pueden tener más de dos fechas dependiendo los estados por los que vaya pasando. La siguiente tabla muestra las fechas que debe tener cada rubro para poderlas clasificar.

Rubro	**Fechas que debe tener**

Proyectos	Fecha inicial y fecha final
Artículos	Fecha de envío, fecha de aceptación y fecha de publicación
Libros y monografías	Fecha de publicación
Capítulos de libros	Fecha de publicación
Informes y reportes	Fecha de publicación
Tesis	Fecha de inicio y fecha de titulación
Tutorías	Fecha inicial y fecha de graduación
Patrocinios	Fecha inicial y fecha final
Visitas a otras instituciones	Fecha inicial y fecha final
Asistencias a congresos, coloquios y talleres	Fecha de realización
Conferencias plenarias o magistrales impartidas	Fecha de realización
Conferencias impartidas y participación en mesas redondas	Fecha de realización
Participación o dirección de seminarios de investigación y de formación	Fecha inicial y fecha final
Organización o coordinación de congresos, coloquios y talleres	Fecha de realización
Investigadores invitados	Fecha inicial y fecha final
Cursos o cursillos impartidos	Fecha inicial
Producciones de material de divulgación	Fecha inicial y fecha final
Traducciones	Fecha inicial y fecha final
Construcciones de prototipos o tecnologías transferidas	Fecha inicial y fecha final
Participaciones en comités editoriales	Fecha inicial y fecha final
Participaciones en consejos editoriales	Fecha inicial y fecha final
Revisiones de trabajos	Fecha de realización
Participaciones en comisiones académicas	Fecha inicial y fecha final
Desarrollos de infraestructura institucional	Fecha inicial y fecha final
Coordinaciones de cursos, talleres y programas docentes	Fecha inicial y fecha final
Elaboraciones de planes de estudio	Fecha inicial y fecha final
Materiales didácticos	Fecha inicial y fecha final
Trabajos institucionales ligados a la docencia	Fecha inicial y fecha final
Distinciones académicas recibidas	Fecha de recepción
Puestos académico-administrativo	Fecha inicial y fecha final
Patentes	Fecha de solicitud y fecha de realización

9. En todos los atributos de fecha que se utilice dentro del MIC se debe dar la posibilidad de que no se proporcione el día o el mes, puesto que llega a ser complicado recordar una fecha exacta. Si no se proporciona el día, el sistema debe de considerar que la actividad se pudo realizar cualquier día del mes, sino se proporciona mes, entonces que se pudo realizar cualquier mes del año. Ésto para efectos de clasificar las actividades por fecha.

10. Cada MIC que se encuentre dentro del *home* de algún investigador debe de tener una forma estándar, es decir, los MIC de los investigadores deben tener el mismo nombre, misma estructura, descripción, etc.

11. No se debe poder modificar la estructura de los MIC.

12. Las actividades de los MIC deben utilizar la clasificación de la MSC.

13. Debe de haber compatibilidad entre BibTex y las actividades del MIC.

14. Los MIC deben permitir la importación/exportación de información curricular entre BibTex.

15. Al importar información, el sistema debe detectar y sugerir el tipo de contenido al que se va a importar la información, pero el usuario debe poder cambiar el tipo a importar.

16. Si hay errores al importar información se debe de avisar al usuario cuáles fueron los errores.

17. Las tesis de licenciatura no tienen compatibilidad con BibTex, pero se deben de poder importar/exportar con BibTex de forma similar a las tesis de maestría y doctorado que utilizan el tipo BibTex masterthesis y phdthesis, respectivamente, pero utilizando el nombre *undergradthesis*.

18. Para poder realizar los diferentes reportes del IMATE cada investigador debe enviar un informe de sus actividades realizadas. Estas actividades deberán ser copiadas en algún lugar del sitio para ser procesadas posteriormente. A esta acción se le denominará *envío de actividades* y será realizada por el sistema cuando se lo indique el investigador.

19. Para realizar el envío de actividades, se le debe de especificar al sistema una fecha inicial y una final. Estas fechas definen el rango en el cuál el sistema debe buscar las actividades, es decir, las actividades que se encuentren entre estas fechas deberán ser tomadas automáticamente por el sistema y confirmadas por el investigador.

20. Con cada *envío de actividades*, se debe generar un acuse de recibido que le de seguridad al investigador de que su envío fue correcto y haga constatar que sus actividades realmente fueron enviadas. Este acuse debe ser como los acuses de papel, una copia de lo que se están entregando, pero no pudiendo ser modificada.

21. El lugar a donde serán copiadas las actividades enviadas debe ser un parámetro configurable.

22. Las actividades recibidas deben de ser clasificadas por año para su fácil localización.

23. Se debe de poder controlar la recepción de actividades indicándole al sistema cuándo se inicia y cuándo termina. En otro momento no se recibirán reportes de actividades.

24. Una vez que algún investigador realizó su *envío de actividades*, éste no puede reenviarlo a menos que se le de autorización.

25. Se debe de poder configurar quiénes son los investigadores que deben enviar su reporte de actividades y así poder consultar quiénes son los que ya enviaron y los que no han enviado su actividades.

26. El sistema debe de permitir obtener diferentes resúmenes de las actividades: resumen numérico (permite saber cuántas actividades se tienen por cada rubro), listado de actividades (muestra todas las actividades, cada una de forma completa, es decir, con todos sus atributos), en tabla (muestra todas las actividades en tablas, cada actividad en una fila), para imprimir (muestra todas las actividades con un formato para entregar, como currículum vítae).

27. Se deben de poder obtener los resúmenes de actividades restringiéndolas por rangos de fechas y por condiciones referentes a los campos de éstas.

28. Una vez recibidas las actividades de los investigadores debe de ser posible realizar consultas sobre éstas por medio de formularios donde se puedan definir diferentes parámetros de búsqueda. A estas consultas se les referirá como *consultas de actividades*.

29. Las consultas de actividades deben permitir obtener diferentes vistas de los resultados: en forma de tabla (permitiendo así transportar la información a otros medios de procesamiento), resumen numérico (permite obtener un conjunto de datos cuantificados referentes a las actividades de cierto rubro) y resumen formateado (permite obtener los resultados con un formato predefinido).

3.2.2. Tipos de usuarios

Dentro del sitio hay varios tipos de usuarios, cada uno con sus propios atributos, pero por las actividades que pueden realizar interactuando con el sistema SCRICI, se pueden reducir a los que se mencionan a continuación:

- **Investigador.** Es el actor principal dentro del sistema SCRICI, pues es quien va a ir agregando actividades dentro del MIC que se encuentre en su *home*.
- **Administrador de SCRICI.** Es el responsable de realizar los diferentes tipos de reportes y de administrar los MIC que se encuentren en el sistema.
- **Invitado.** Dentro del contexto de un MIC, se le llamará así al usuario que no es dueño ese MIC. Dentro de este tipo usuario se encuentran los no autenticados, que son los usuarios ajenos al IMATE; y los usuarios autenticados que no son dueños

del MIC en cuestión. Por ejemplo, si se está hablando del MIC del investigador X, el investigador Y será un usuario *Invitado* puesto que no es dueño de ese MIC.

3.2.3. Requerimientos por perfil operacional

A continuación se muestran las actividades directamente relacionadas con el sistema SCRICI, que le debe permitir hacer a cada tipo de usuario:

Invitado

- Debe poder acceder a cualquier MIC para visualizar la información que se encuentre dentro.
- Debe poder exportar la información curricular de cualquier MIC.

Investigador

1. Debe poder acceder a cualquier MIC para visualizar la información que se encuentre dentro.
2. Debe poder exportar la información curricular de cualquier MIC.
3. Debe poder realizar el *envío de actividades* desde su MIC.
4. Debe poder importar y agregar información dentro de su MIC.
5. Debe poder modificar las actividades dentro de su MIC.

Administrador de SCRICI

El administrador de SCRICI va a poder realizar varias actividades propias de un investigador debido a que es posible que los investigadores soliciten ayuda con sus actividades por cualquier problema que se llegue a presentar, ya sea interno o externo al sistema, además de otros tareas de administración.

1. Debe poder acceder a cualquier MIC para visualizar la información que se encuentre dentro.
2. Debe poder exportar la información curricular de cualquier MIC.
3. Debe poder realizar el *envío de actividades* desde el MIC de un investigador.
4. Debe poder importar y agregar información dentro del MIC de un investigador.
5. Debe poder modificar las actividades dentro del MIC de un investigador.
6. Debe poder agregar, borrar o modificar el módulo de información curricular en el *home* de los investigadores.

7. Debe poder controlar el envío de actividades (cuándo inicia y cuándo termina).
8. Debe poder definir la ruta a la que llegarán las actividades enviadas por los investigadores.
9. Debe poder definir consultas sobre cada rubro para que otros las puedan utilizar, así como definir quiénes pueden utilizarlas.

3.2.4. Casos de uso

Los casos de uso (CU) son escenarios que describen la forma en que los actores van a interactuar con el software bajo determinadas situaciones. Éstos son una parte del análisis que permite definir lo que debe hacer el sistema.

A continuación se muestran los casos de uso indicando los actores que lo pueden realizar y una descripción.

Consultar MIC de un investigador

Actor: Invitado, Investigador y Administrador de SCRICI
Requerimientos que satisface. 26 y 27.
Descripción. Un MIC está compuesto por diferentes rubros que contienen las actividades de los investigadores. La forma en que se consultan las actividades, es navegando a través los rubros del MIC y seleccionando actividades para verlas.

Flujo 1: Consulta de una actividad

Actores	**Sistema**
Dentro del *home* de un investigador da clic en la carpeta *Mi currículum*.	Se posiciona dentro del MIC del investigador y muestra los diferentes rubros que tiene el MIC.
Da clic en alguno de los rubros mostrados.	Muestra las actividades que se encuentran dentro del rubro seleccionado.
Da clic en alguna actividad.	Muestra información referente a la actividad.

Flujo 2: Consulta de varias actividades utilizando diferentes formatos

Actores	**Sistema**
Dentro del *home* de un investigador da clic en la carpeta *Mi currículum*.	Se posiciona dentro del MIC del investigador y muestra los diferentes rubros que tiene el MIC.
Selecciona la opción *Elegir un tipo de resumen*.	Muestra un formulario en el que se puede elegir el rango de fechas de las actividades y tipo de resumen.
Selecciona el rango de fechas de las actividades y da clic en un tipo de resumen.	Muestra las actividades según el tipo de resumen elegido y filtradas por las fechas proporcionadas.

Exportar información del MIC de un investigador

Actor: Invitado, Investigador y Administrador de SCRICI

Requerimientos que satisface. 14 y 17

Descripción. Dentro del MIC existen actividades que son compatibles con BibTex. Ya que éste es un formato estándar para referencias bibliográficas se pueden exportar estas actividades para poderlas reutilizar.

Flujo 1: Exportar información de todos los rubros al mismo tiempo.

Actores	**Sistema**
Dentro de un MIC, posicionado en la raíz, selecciona la opción *Exportar*.	Muestra un formulario en el que da la opción de elegir entre diferentes formatos a exportar.
Selecciona un formato y da clic en el botón *Exportar*.	Devuelve un archivo con la información exportada al formato elegido sobre todas las actividades compatibles con BibTex dentro del MIC.

Flujo 2: Exportar información sólo de un rubro.

Actores	**Sistema**
Dentro de un MIC, posicionado en la raíz, selecciona un rubro que se pueda exportar.	Se posiciona en ese rubro.
Selecciona la opción *Exportar*.	Muestra un formulario en el que da la opción de elegir entre diferentes formatos a exportar.
Selecciona un formato y da clic en el botón *Exportar*.	Devuelve un archivo con la información exportada al formato elegido sobre todas las actividades dentro del rubro seleccionado, compatibles con BibTex.

Importar información del MIC de un investigador

Actor: Invitado, Investigador y Administrador de SCRICI

Requerimientos que satisface. 14, 15, 16 y 17

Descripción. Dentro del MIC existen actividades que son compatibles con BibTex y se pueden importar al sistema para evitar volver a capturar esta información. Este CU explica cómo importar estas actividades.

Flujo 1: Importar información de varios rubros al mismo tiempo.

Actores	**Sistema**
Dentro de un MIC, posicionado en la raíz, selecciona la opción *Importar*.	Muestra un formulario en el que da la opción de elegir entre diferentes formatos y permite ingresar el código o archivo a importar.
Llena el formulario y da clic en el botón *Importar*.	Muestra un formulario con las actividades a importar junto con el tipo de contenido al que se importará cada una por default, dando la posibilidad de modificar el tipo. Si hay algún error al importar se le muestra al usuario.
Si lo requiere, modifica el tipo a importar de algunas actividades y da clic en aceptar.	Importa las actividades y muestra los resultados de la importación.

Flujo 2: Importar información sólo de un rubro.

Actores	**Sistema**
Dentro de un MIC, posicionado en la raíz, selecciona un rubro que se pueda importar.	Se posiciona en ese rubro.
Selecciona la opción *Importar*.	Muestra un formulario en el que da la opción de elegir entre diferentes formatos y permite ingresar el código o archivo a importar.
Llena el formulario y da clic en el botón *Importar*.	Muestra un formulario con las actividades a importar junto con el tipo de contenido al que se importará cada una por default, dando la posibilidad de modificar el tipo, pero sólo permite importar contenidos que se encuentren dentro del rubro seleccionado. Si hay algún error al importar se le muestra al usuario.
Si lo requiere, modifica el tipo a importar de algunas actividades y da clic en aceptar.	Importa las actividades y muestra los resultados de la importación.

Enviar reporte de actividades

Actor: Investigador y Administrador de SCRICI

Requerimientos que satisface. 18, 19, 20 y 22.

Descripción. Cada investigador tiene dentro de su MIC un conjunto de actividades que pueden corresponder a diferentes años. Al menos una vez al año se les pide a los investigadores que entreguen un reporte sobre sus actividades realizadas durante el año o varios años atrás. Este CU sirve para realizar este reporte.

Flujo:

Actores	**Sistema**
Dentro de un MIC, posicionado en la raíz, selecciona la opción *Enviar Informe*.	Muestra un formulario en el que solicita el tipo de reporte a enviar y el rango de fechas en las que se encuentran las actividades a reportar.
Selecciona un tipo de reporte: *PRIDE*, *PAIPA* o *Anual*; y da clic en aceptar.	Muestra un formulario en el que muestra las actividades a reportar, dando la opción de no reportar algunas.
Selecciona las actividades a reportar y da clic en enviar.	Crea dos MIC en los que copia las actividades a reportar; uno lo deposita dentro de la carpeta de acuses del MIC del investigador y el otro lo deposita por años en una carpeta previamente definida que recibe los reportes de todos los investigadores. Por último informa si el envío fue exitoso o no.

Agregar actividades dentro de un MIC

Actor: Investigador y Administrador de SCRICI

Requerimientos que satisface. 5, 6, 7, 8, 9, 12, 13

Descripción. El MIC está diseñado para guardar las actividades de los investigadores. Con este CU se muestra como se pueden agregar actividades.

Flujo:

Actores	**Sistema**
Dentro del MIC, selecciona el rubro correspondiente a la actividad a agregar.	Se posiciona en el rubro seleccionado.
Selecciona la opción *Agregar un nuevo ítem*.	Muestra un formulario con campos propios de la actividad.
Llena el formulario y da clic en *Guardar*.	Crea y agrega la actividad con la información proporcionada.

Eliminar actividades dentro de un MIC

Actor: Investigador y Administrador de SCRICI

Descripción. Los MIC guardan actividades de los investigadores, y en caso de tener una actividad agregada de forma incorrecta se puede eliminar.

Flujo:

Actores	**Sistema**
Dentro del MIC, selecciona el rubro donde se encuentra la actividad a eliminar.	Se posiciona en el rubro seleccionado.
Selecciona la actividad a eliminar.	Muestra la actividad seleccionada.
Selecciona la opción *Eliminar*.	Pide confirmación para eliminar la actividad.
Da clic en aceptar.	Elimina la actividad.

Modificar actividades dentro de un MIC

Actor: Investigador y Administrador de SCRICI

Descripción. En caso de tener una actividad agregada de forma incorrecta o si hay nuevos datos a agregar en la misma actividad o que cambiar se puede modificar la actividad.

Flujo:

Actores	**Sistema**
Dentro del MIC, selecciona el rubro donde se encuentra la actividad a modificar.	Se posiciona en el rubro seleccionado.
Selecciona la actividad a modificar.	Muestra la actividad seleccionada.
Selecciona la opción *Editar*.	Muestra un formulario con los campos de la actividad con la información previamente guardada.
Modifica los campos necesarios y da clic en *Guardar*.	Guarda la actividad con los cambios realizados.

Agregar un MIC dentro del *home* un usuario

Actor: Administrador de SCRICI

Requerimientos que satisface. 1, 2, 3, 4, 10 y 11.

Descripción. Cuando un usuario se autentica en el sistema por primera vez automáticamente se crea su *home*, el cual se encuentra vacío. El investigador no puede agregar un MIC en su *home*, pues ésto sólo lo puede hacer un usuario con mayores privilegios. Este CU explica cómo agregar un MIC dentro del *home* de un usuario.

Cabe mencionar que para agregar varios MIC de forma colectiva se pueden utilizar scripts de Python, pero no es una tarea muy común puesto que regularmente no ingresan muchos investigadores al IMATE.

Flujo:

Actores	**Sistema**
Busca el *home* de un investigador y da clic en esa carpeta.	Se posiciona en el *home* del investigador.
Da clic en *Agregar un nuevo ítem → currículum.*	Muestra un formulario con campos del MIC: *Título*, *Descripción* y *Dueño del currículum.*
Llena el formulario y da clic en *Guardar.*	Agrega un MIC con varias carpetas que hacen referencia a los diferentes rubros y con una estructura estándar. Además, con permisos para que nadie pueda modificar su estructura.

Eliminar MIC dentro de un *home*

Actor: Administrador de SCRICI

Descripción. El siguiente CU muestra cómo eliminar un MIC del *home* de un usuario

Flujo:

Actores	**Sistema**
Busca el *home* de un investigador y da clic en el MIC.	Se posiciona en el MIC del investigador.
Da clic en *Eliminar.*	Pide confirmación para eliminar el MIC.
Da clic en aceptar.	Elimina el MIC.

Agregar *consulta de actividades*

Actor: Administrador de SCRICI

Requerimientos que satisface. 28.

Descripción. Para poder realizar los diferentes reportes del IMATE referentes a las actividades académicas de los investigadores se pueden utilizar las *consultas de actividades*, las cuáles de forma rápida y sencilla regresan datos cuantificados, concentrado de actividades en tablas para una manipulación externa en hojas de cálculo y reportes con formato definido por el usuario. Este CU explica cómo agregar una consulta de actividades.

Flujo:

Actores	**Sistema**
Dentro de cualquier carpeta del sitio da clic en *Agregar un nuevo ítem → Reporte.*	Muestra un formulario con varios campos: tipo de reporte, lugar de búsqueda, formato del reporte, descripción y título.
Llena el formulario y da clic en aceptar.	Agrega una *consulta de actividades.*

Eliminar *consulta de actividades*

Actor: Administrador de SCRICI

Descripción. El siguiente CU muestra cómo eliminar una *consulta de actividades*.

Flujo:

Actores	**Sistema**
Da clic en la *consulta de actividades* a eliminar.	Se posiciona en la *consulta de actividades*.
Da clic en *Eliminar*.	Pide confirmación para eliminar la *consulta de actividades*.
Da clic en aceptar.	Elimina la *consulta de actividades*.

Realizar consultas de actividades

Actor: Administrador de SCRICI

Requerimientos que satisface. 28 y 29.

Descripción. El siguiente CU muestra cómo consultar las actividades de los investigadores utilizando las *consultas de actividades*.

Flujo:

Actores	**Sistema**
Da clic en una *consulta de actividades*.	Se posiciona en la *consulta de actividades* mostrando un formulario con un menú que permite restringir la búsqueda de actividades.
Llena el formulario y da clic en el tipo de reporte que desea obtener (en tabla, numérico o con formato).	Muestra un reporte de las actividades que coincidieron con los parámetros de búsqueda.

Definir ruta de recepción de los reportes de actividades

Actor: Administrador de SCRICI

Requerimientos que satisface. 21.

Descripción. El siguiente CU muestra cómo configurar el lugar al que llegarán los reportes de los investigadores cuando éstos envíen su reporte de actividades.

Flujo:

Actores	Sistema
Desde la interface de Zope, en la raíz del sitio, da clic en el objeto *portal_curriculum*.	Se posiciona en el objeto *portal_curriculum*.
Da clic en la pestaña *View*.	Muestra el objeto *portal_curriculum*.
Da clic en la pestaña *Editar*.	Muestra un formulario en el que permite modificar la ruta donde se recibirán los reportes de actividades.
Llena el formulario y da clic en *aceptar*.	Guarda los valores proporcionados por el usuario.

Controlar la recepción de los reportes de actividades

Actor: Administrador de SCRICI

Requerimientos que satisface. 23, 24 y 25.

Descripción. El siguiente CU muestra cómo controlar la recepción de los reportes de actividades que envían los investigadores.

Flujo:

Actores	Sistema
Desde la ruta en la cual se espera el reporte, da clic en *Agregar ítem* → *Folder de informes enviados*.	Muestra un formulario.
Llena el formulario poniendo el año del cual se espera recibir reportes.	Guarda los valores proporcionados.
Da clic en la opción *Abrir recepción de informes* o *Cerrar recepción de informes*.	Abre o cierra la recepción de informes, permitiendo que los usuarios envíen o bloqueando el envío del reporte correspondiente a ese año.

3.2.5. Diagramas de casos de uso

En la figura 3.1 se encuentra el diagrama de casos de uso, donde se muestra la relación entre los actores y los casos de uso del sistema, mostrados anteriormente.

3.3. Arquitectura

El sistema se dividió en dos partes: para recolección y para recuperación de la información.

La recolección de la información es la parte del SCRICI que permite ingresar información a los MIC, ya sea de forma manual, mediante captura de datos, o mediante la importación de información.

La recuperación de la información es la parte del SCRICI que permite extraer información o datos de varios MIC mediante consultas especializadas. Se busca que cualquier

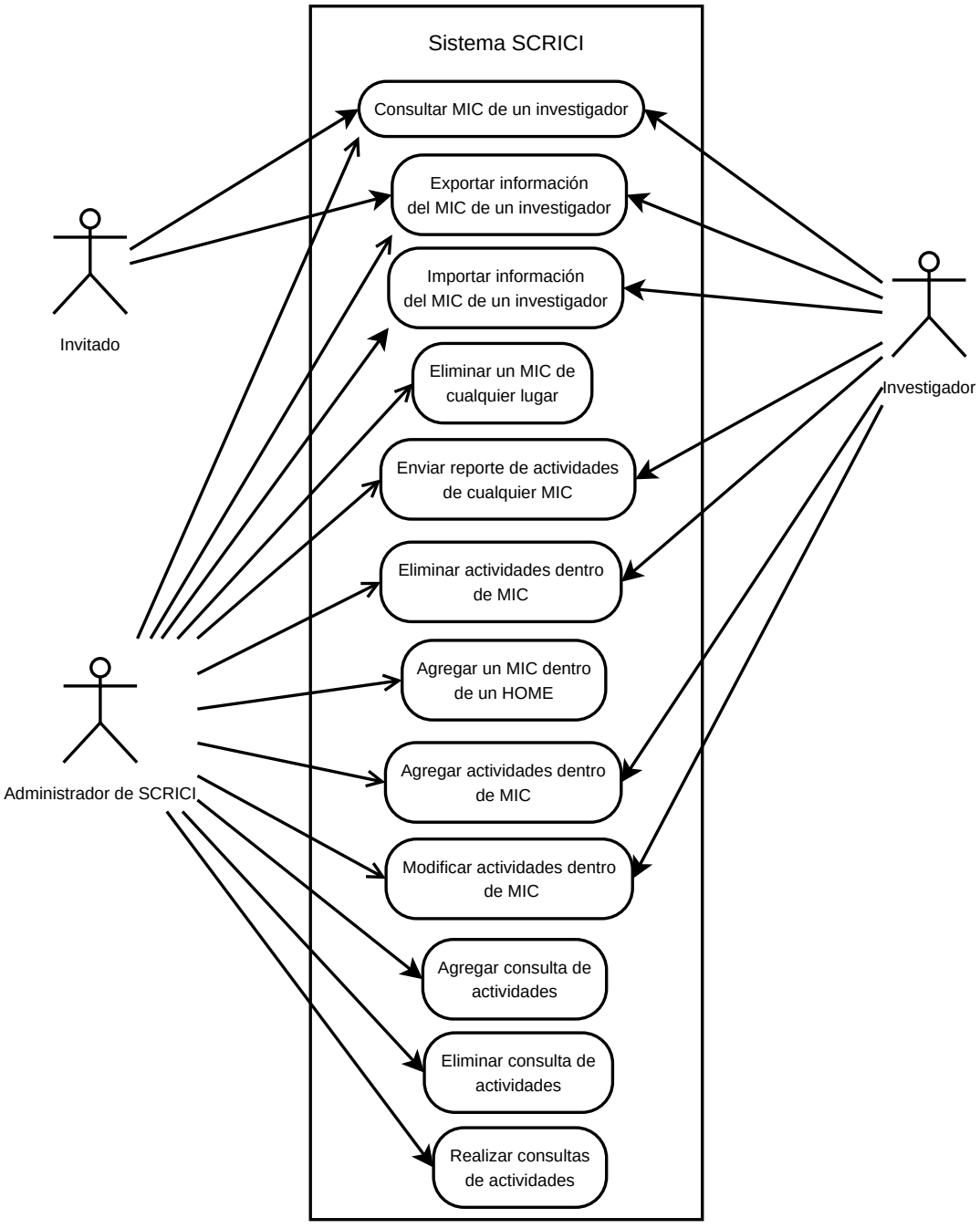

Figura 3.1: Diagrama de casos de uso del sistema SCRICI

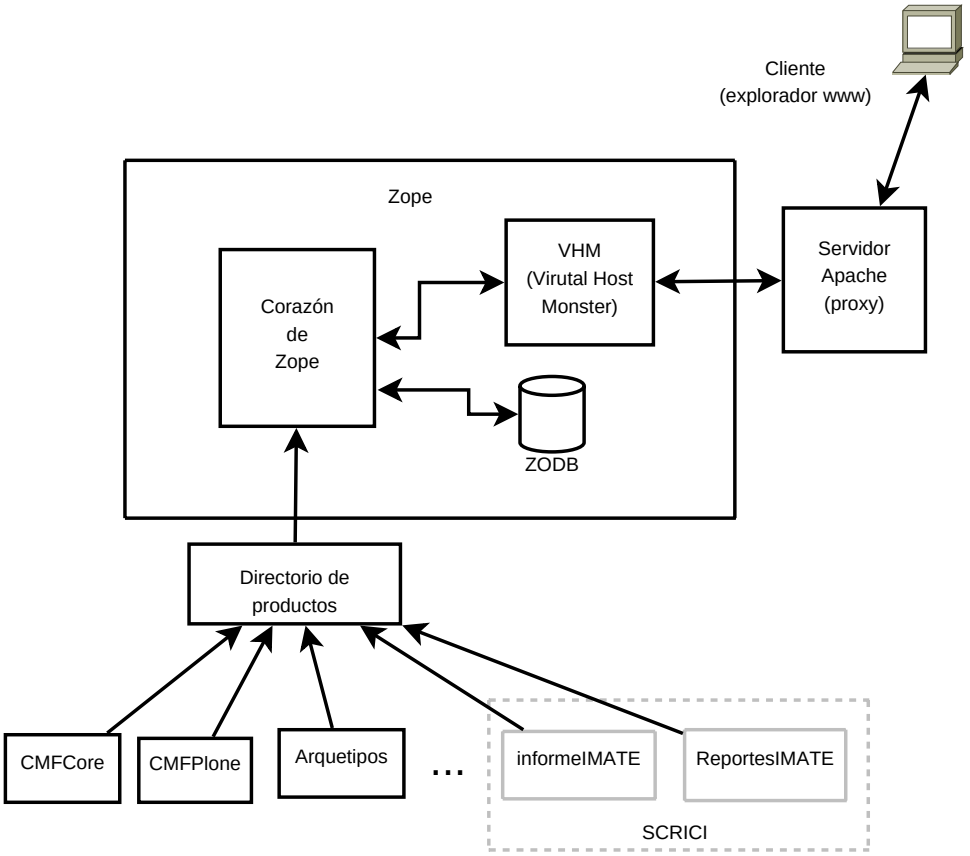

Figura 3.2: Arquitectura física del sistema de información del IMATE

persona pueda realizar esta tarea sin necesidad de conocer aspectos técnicos, como el lugar o forma en que se encuentra almacenada la información o formas de recuperación de información.

Para recolectar la información se creó un producto llamado *informeIMATE* y para recuperarla de forma colectiva se creó otro llamado *ReportesIMATE*.

3.3.1. Arquitectura física

La figura 3.2 muestra la arquitectura física que se siguió para implementar el *Sistema de Información del Instituto de Matemáticas*, dentro del cual se encuentra el sistema SCRICI.

Como servidor Web que va a recibir las peticiones directamente de los usuarios se tiene Apache, el cual maneja comunicación segura mediante SSL y direcciona las peticiones de los usuarios hacia el VHM (Virtual Host Monster), quien a su vez manda las peticiones hacia Zope. Como repositorio se va a utilizar la ZODB, la cual es parte de Zope.

Como se mencionó en el capítulo 2, Plone está sobre Zope y el CMF. Tanto el CMFPlone (mejor conocido como Plone) como el CMFCore (mejor conocido como el CMF)

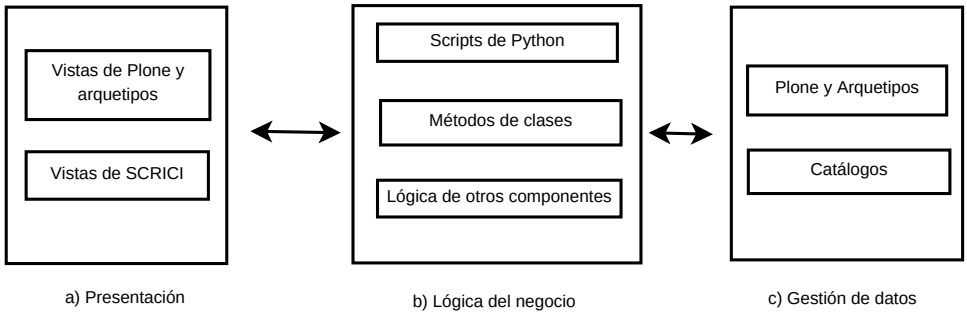

Figura 3.3: Capas del sistema SCRICI

están implementados sobre Zope como productos, pero Plone depende del CMF para que se pueda instalar. Además hay muchos otros productos instalados sobre Zope, que requieren de Plone, representados en la figura 3.2 como tres puntos suspensivos, puesto que son muchos. El sistema SCRICI está conformado por dos productos: *informeIMATE* y *ReportesIMATE*, que se muestran en la figura 3.2 dentro de las líneas punteadas.

Dado que Plone trabaja con componentes, si se requiere alguna funcionalidad que ya la provee algún componente, ésta se puede utilizar para crear un nuevo producto.

Cabe mencionar que los productos *informeIMATE* y *ReportesIMATE* están basados principalmente en los arquetipos. Es decir, utilizan la funcionalidad provista por los arquetipos y además le agregan funcionalidad propia.

3.3.2. Arquitectura de capas

Plone permite el desarrollo de productos por capas, separando la presentación, la lógica del negocio y la administración de los datos. Al desarrollar por medio de capas se obtiene la ventaja de que se puede modificar una capa sin que las demás capas se vean afectadas. La figura 3.3 muestra cómo se tienen estas capas dentro del sistema SCRICI.

Capa de presentación

Es la parte utilizada para mostrar la aplicación a los usuarios. Esta capa incluye código y otros recursos que permiten definir la interface con el usuario (como texto HTML e imágenes). Típicamente tiene un poco de código relacionado con el formato y presentación de los datos.

Esta capa está conformada por vistas ya definidas por Plone y los arquetipos; además se implementaron otras vistas utilizando plantillas de Zope (ZPT), como se muestra en la figura 3.3a.

Dentro de Plone, las vistas no se encuentran ligadas a ningún objeto en particular. Pero dentro de cada definición de contenido, que se hace mediante clases derivadas de los arquetipos, se deben configuran las vistas a utilizar por cada acción. Por ejemplo, se debe definir la vista a utilizar para ver y para editar un contenido.

A través de las vistas se le pueden enviar mensajes a los objetos del sitio, siempre y cuando los mensajes enviados sean conocidos por los objetos que los reciben. Es decir, los objetos deben tener implementados los métodos invocados por las vistas para que pueda haber comunicación entre ambos.

Capa de lógica del negocio

Dentro de esta capa se encuentra sólo el código relacionado con la lógica del negocio. Idealmente esta capa debería ignorar el código utilizado por la capa de presentación.

La capa de presentación le envía mensajes a los objetos del sitio para que éstos realicen ciertas acciones, las cuales son parte de la lógica del negocio.

Como se muestra en la figura 3.3b, la lógica del negocio se encuentra conformada de la siguiente forma:

- Mediante métodos de clases. Cada objeto dentro del sitio está implementado mediante una o más clases, dentro de las que se pueden tener varios métodos.
- Mediante scripts de Python. Dentro del sistema SCRICI se crearon varios scripts de Python que permiten definir parte de la lógica del negocio.
- Lógica de otros productos. Se aprovecharon algunos scripts y métodos implementados dentro de otros productos.

Capa de gestión de datos

Esta capa, mostrada en la figura 3.3c se encarga de interactuar con la base de datos para guardar o recuperar datos. Permite encapsular las especificaciones de la base de datos (SQL, Oracle, ZODB, archivos de texto, etc) a la siguiente capa.

Esta capa es llevada a cabo por Plone y los arquetipos de forma transparente para el programador. Cada que se agrega o se modifica un contenido creado con los arquetipos, de forma transparente se le avisa a Plone para que guarde el objeto tal cual, como si fuera una fotografía, dentro de la base de datos. Cuando se quiere acceder al objeto, Plone se encarga de recuperarlo de la base de datos y cargarlo.

Para buscar objetos dentro de la base de datos y recuperarlos de manera eficiente se utilizan catálogos que indexan los contenidos del sitio.

Por default Plone viene con un catálogo llamado *portal_catalog*, el cual se encarga de indexar todos los contenidos del sitio y realizar búsquedas sobre ellos. Sin embargo, dentro del sistema SCRICI se tienen consultas especializadas que requieren definir bastantes índices y metadatos, lo cual afectaría el desempeño de todos los objetos del sitio si se definieran dentro del *portal_catalog*. Por esto se definió otro catálogo propio del sistema SCRICI, el cual se llama *curriculum_catalogo*, el cual se encarga específicamente de catalogar la información que se encuentra en todos los MIC.

Además, existen otros catálogos que se utilizaron para realizar consultas, como el *member_catalog* que indexa información sobre los miembros del sitio y el *uid_catalog* que indexa los contenidos creados con los arquetipos.

3.4. Componentes existentes

En el sitio de Plone [7] hay aproximadamente 500 productos, que se pueden obtener de forma gratuita. Para saber cuál utilizar se tiene una lista de todos con una clara descripción de cada uno. Además dentro del sitio de Plone se pueden realizar búsquedas al estilo *Google* para encontrar el producto que más se acople a las necesidades.

Se analizaron los componentes existentes para trata de encontrar aquellos que proporcionen parte de la funcionalidad que debe realizar el sistema SCRICI. A continuación se muestran los que se eligieron.

- **Archetypes**. Este componente permite definir nuevos contenidos, editarlos, visualizarlos, indexarlos en uno o varios catálogos, navegar a través de ellos y proporciona varias opciones para manejarlos. Fue utilizado para definir las actividades y carpetas del MIC, así como las consultas de actividades.

- **CMFPlone**. Además de todas las características que proporciona, ya mencionadas en la sección 2.10, también permite definir catálogos, característica que se utilizó para definir el catálogo llamado *curriculum_catalogo*, que se encarga de indexar la información de los MIC.

- **ATCountryWidget, AddRemoveWidget, MasterSelectWidget, CompoundField, DataGridField, ATExtensions, ATReferenceBrowserWidget**. Permiten definir diferentes campos y widgets para que sean utilizados dentro de los esquemas de los contenidos. Estos componentes permiten visualizar y editar las actividades de los investigadores de forma más amigable y más a la medida.

- **CMFCore**. Este componente proporciona muchas características que son utilizadas por Plone y que pueden utilizar otros productos. Se utilizaron diferentes funciones que proporciona: para localizar objetos únicos dentro del sitio, revisar permisos de un usuario dentro de un objeto, instalar y desinstalar productos. Además dentro de este componente se definen varios permisos que se utilizaron.

3.5. Desarrollo de componentes propios

No todos los componentes encontrados satisficieron los requerimientos del sistema, y por eso se implementaron esos requerimientos mediante nuevos componentes, que se muestran a continuación:

- **ATFechas**. Este componente define un campo y widget para almacenar fechas, las cuales pueden ser incompletas. Es decir, para guardar fechas que no tengan día o que sólo tengan el año. Actualmente el componente *Archetypes* tiene a su vez los componentes *DateTimeField* y *DateTimeWidget*, los cuales permiten almacenar fechas, sólo que éstas deben ser completas, con día, mes y año.

- **ReportesIMATE**. Este componente permite realizar consultas especializadas sobre las actividades de los MIC. Actualmente existe un componente llamado *CustomSearch* que permite también realizar consultas sobre los contenidos de un sitio, sólo que se requiere una consulta hecha a la medida, fácil de agregar y configurar, lo cual no lo permite el componente *CustomSearch*.

- **informeIMATE**. Este componente utiliza la funcionalidad de los arquetipos, sin embargo, para cumplir con todos los requerimientos se tuvo que extender ésta, para lo cual se crearon vistas, scripts, métodos y se acopló con otros componentes.

En el capítulo 4 se muestra cómo desarrollar un componente propio integrándolo con otros componentes.

3.6. Desarrollo del sistema integrándolo con otros componentes

Para mostrar cómo se desarrolló el sistema integrándolo con otros componentes primero se mostrará la forma en que se implementaron los casos de uso y después cómo está integrado con otros componentes.

3.6.1. Implementación de casos de uso

Antes de explicar cómo implementar los casos de uso en Plone, es necesario explicar los diferentes archivos que usa Plone, cómo se interconectan y el orden en el que serán invocados.

Para implementar los casos de uso se utilizaron varios tipos de archivos que se muestran a continuación:

- **.pt** Es una vista escrita en ZPT.

- **.py** Es un script escrito en Python que se utiliza como si fuera un método de un objeto.

- **.cpt** Es una vista escrita en ZPT, la cual envía un mensaje a un archivo del tipo *metadata* dependiendo de los eventos realizados por el usuario. Por ejemplo si el usuario llena un formulario y presiona un botón, esa información la recibirá el archivo del tipo *metadata*.

- **.cpy** Es un script de Python, el cual envía el resultado de su ejecución a un archivo del tipo *metadata*.

- **.metadata** Este archivo atrapa los resultados arrojados por los archivos del tipo *cpt* y *cpy*, y con base en éstos define la siguiente acción a realizar, que puede ser: ejecutar un script o mostrar una vista.

Para interconectar y definir el orden en que los archivos serán invocados por Plone, se debe crear un archivo del tipo *metadata* por cada archivo del tipo *cpy* y *cpt*, con el mismo nombre sólo agregando al final la extensión *metadata*. Por ejemplo, una vista llamada evalúa.cpy debe enviarle información a un archivo llamado evalua.cpy.metadata, de esta forma se puede ir formando un autómata que define el caso de uso.

A continuación se mostrará la implementación de cada caso de uso mediante un diagrama que muestra el flujo de los archivos acompañado de un diagrama de flujo.

Caso de uso: *Consultar MIC de un investigador*

Este caso de uso, mostrado en la sección 3.2.4, tiene dos flujos:

- Flujo 1: Consulta de una actividad. Éste no se tuvo que implementar, puesto que esta funcionalidad ya la trae Plone y los arquetipos.
- Flujo 2: Consulta de varias actividades utilizando diferentes formatos. Éste se implementó como se muestra en la figura 3.4. La navegación a través del sitio ya está implementada por Plone, así que los diagramas 3.4a y 3.4b inician a partir de que un usuario llega a la vista resúmenes.cpt.

  La vista resúmenes.cpt muestra un formulario en el que se le solicita al usuario el rango de fechas de las actividades a buscar y el tipo de vista o resumen a mostrar. El archivo resumenes.cpt.metadata valida que las fechas ingresadas sean válidas y decide qué vista mostrar según el botón presionado por el usuario. Los archivos ResumenTexto.pt, Numérico.pt y ResumenTabla.pt son diferentes vistas que se le muestran al usuario dependiendo de la opción que seleccione.

Caso de uso: *Exportar actividades*

Este caso de uso, mostrado en la sección 3.2.4, tiene dos flujos. Sin embargo, ambos están implementados de la misma forma, como se muestra en la figura 3.5. La única diferencia entre ambos flujos son las actividades que va a exportar el sistema, en un flujo exporta sólo las de un rubro y en el otro exporta todas las del MIC.

La plantilla *bibliografia_exportForm.pt* es una vista que permite elegir entre los diferentes tipos de formatos a exportar. Después de elegir el formato a exportar, y dar clic en aceptar, se manda a llamar el script *bibliografia_export.py* que se encarga de exportar y devolver la información en un archivo.

Caso de uso: *Importar actividades*

Este caso de uso, mostrado en la sección 3.2.4, tiene dos flujos. Sin embargo, ambos están implementados de la misma forma, como se muestra en la figura 3.6. La única diferencia entre ambos flujos es que en un flujo sólo permite importar las actividades correspondientes a un rubro y en el otro se pueden importar todo tipo de actividades.

El proceso inicia en la plantilla *bibliografia_importForm.cpt* que muestra un formulario en el que se ingresa la información a importar y el formato. Al dar clic en *importar* se pasa

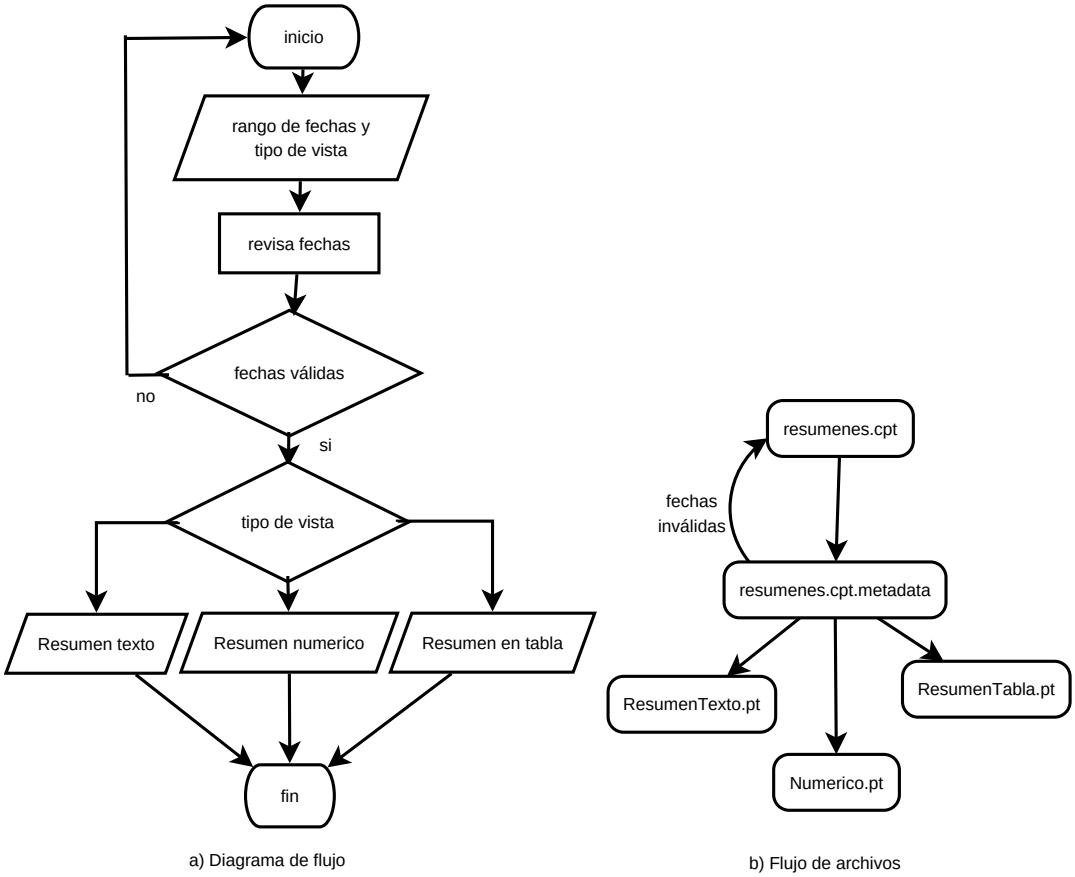

Figura 3.4: Consultar MIC de un investigador (flujo 2)

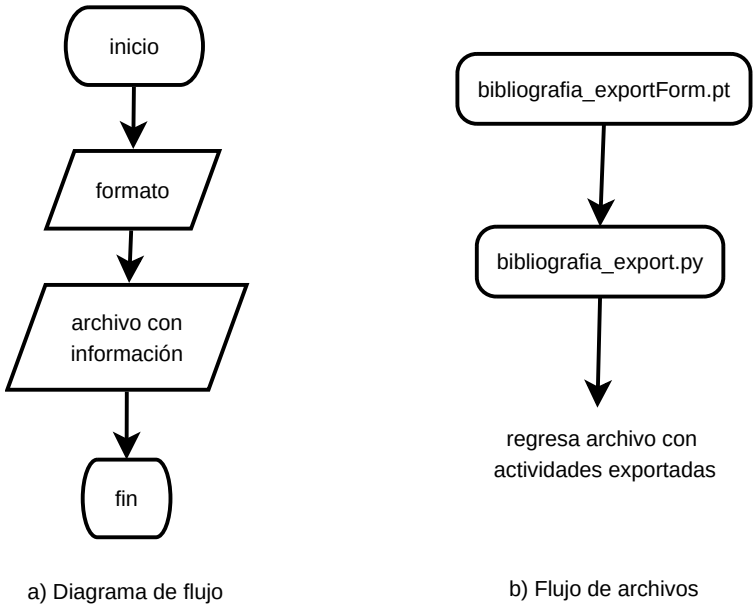

Figura 3.5: Diagrama de flujo de la exportación de actividades

el control al archivo *bibliografia_importForm.cpt.metadata* que verifica que la información ingresada tenga un formato correcto y define la siguiente acción.

La vista *bibliografia_importConfirmacionForm.cpt* muestra un formulario el cual permite definir el tipo de contenido al que se importará cada actividad, proponiendo un tipo por default según la información proporcionada.

El script *realiza_importacion.cpy* se encarga de realizar la importación. El resultado de la importación se le manda a la plantilla *bibliografia_importForm.cpt* para que lo muestre.

Caso de uso: *Enviar reporte de actividades*

Este caso de uso está mostrado en la sección 3.2.4, y su implementación en la figura 3.7.

Este proceso inicia con la vista *envioInformeFecha.cpt* que muestra un formulario en el que se debe especificar el tipo de reporte a enviar y las fechas entre las cuáles se van a buscar las actividades.

El archivo *envioInformeFecha.cpt.metadata* valida que las fechas ingresadas sean válidas y decide la siguiente vista a mostrar.

La vista *envio_informe.cpt* muestra las actividades que se van a enviar dando la opción al usuario de que no reporte algunas de éstas. Además en el caso del reporte *Anual del IMATE* se le pide al investigador que llene datos adicionales.

El archivo *envio_informe.cpt.metadata* realiza cierta validación sobre los datos proporcionados por el usuario y decide la siguiente acción a realizar.

El archivo *script_envio.cpy* es un script que se encarga de realizar el envío de actividades, cuyo resultado se muestra en la vista llamada *resultado_envioForm.pt*.

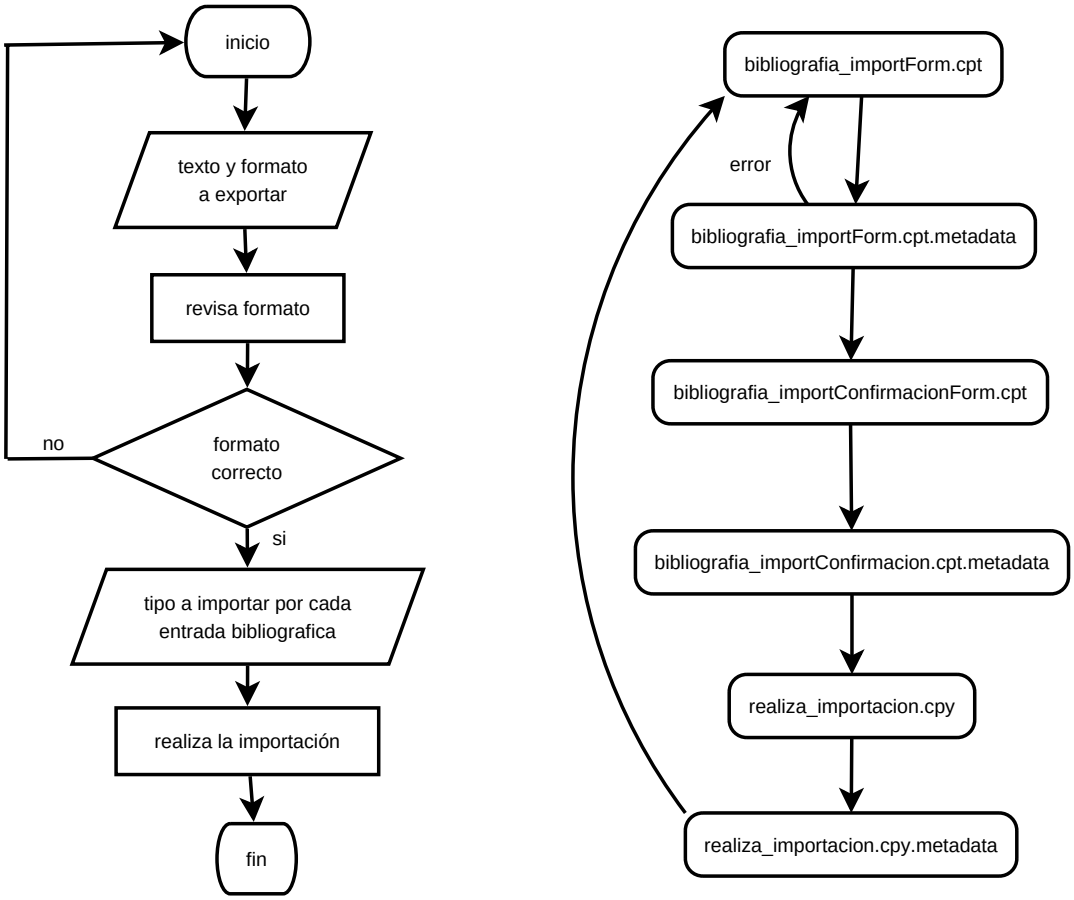

Figura 3.6: Diagrama de flujo de la importación de actividades

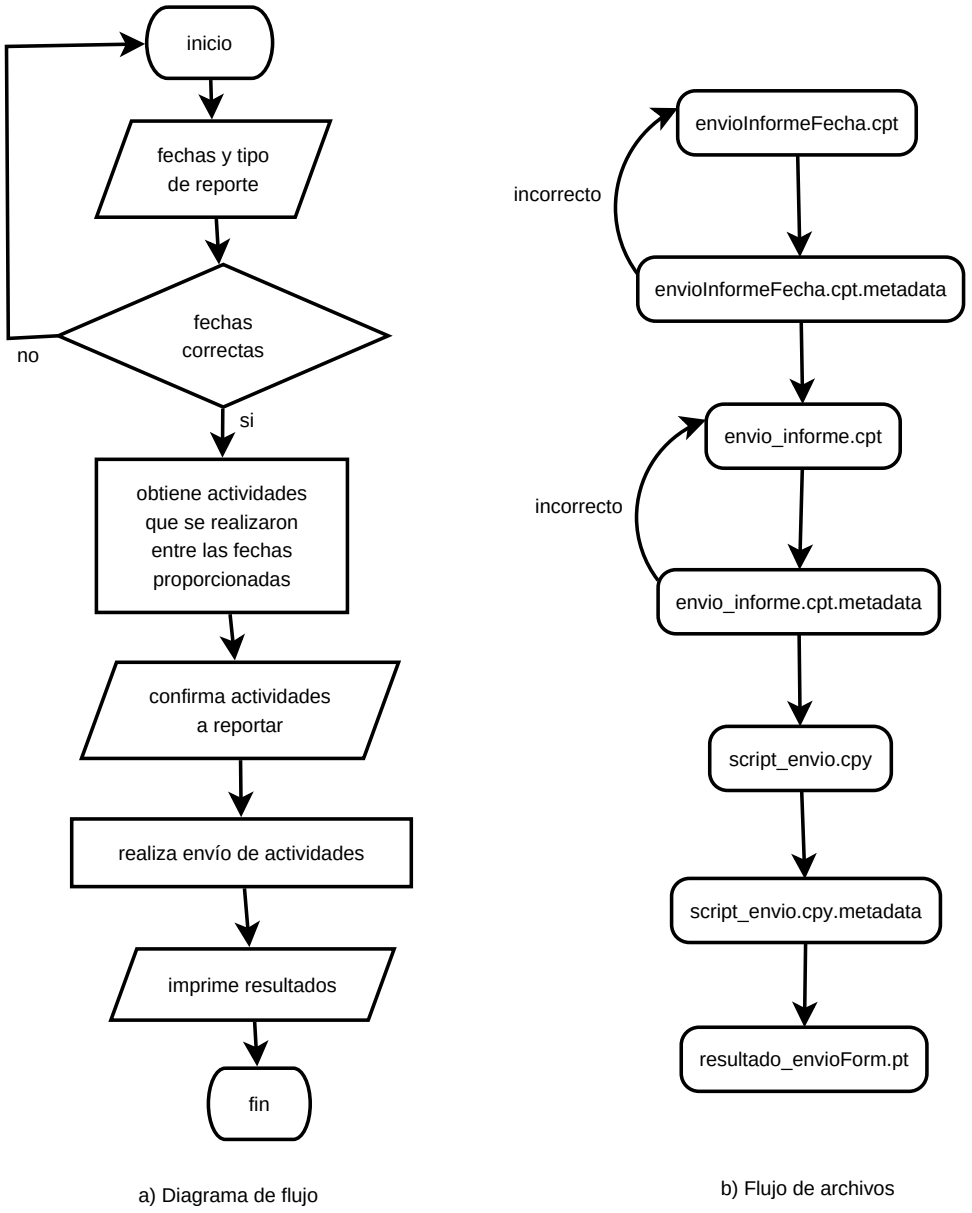

Figura 3.7: Diagrama de flujo del envío del reporte de actividades

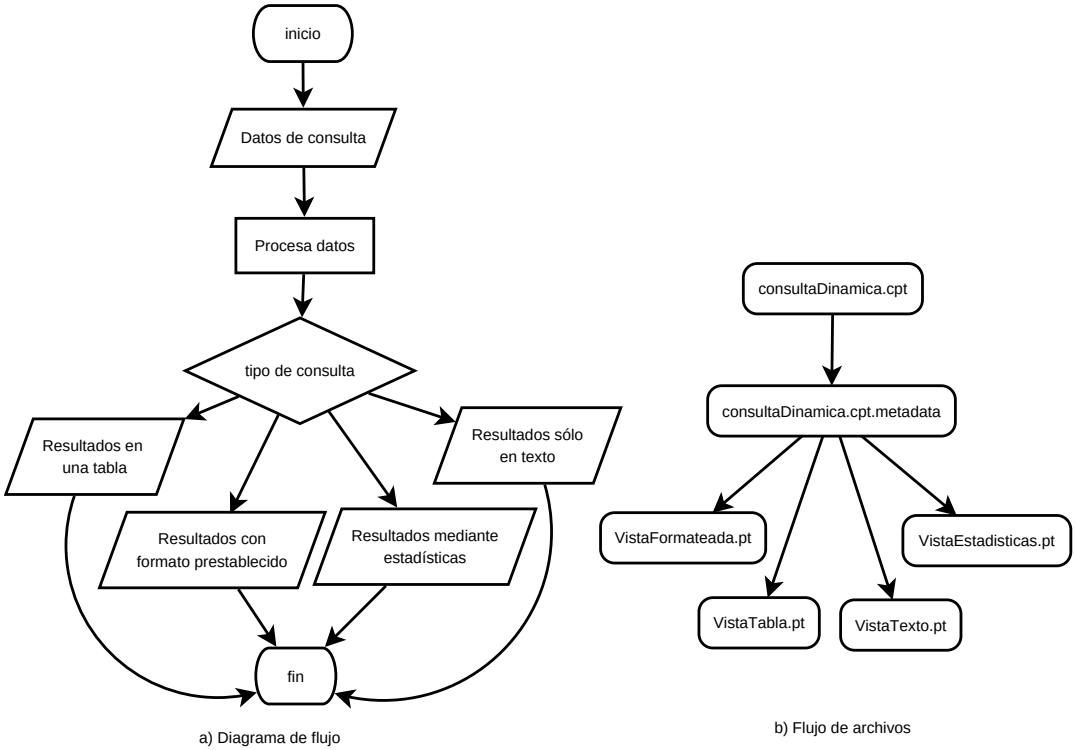

Figura 3.8: Diagrama de flujo del envío del de reporte de actividades

Caso de uso: *Realizar consulta de actividades*

Este caso de uso está mostrado en la sección 3.2.4, y su implementación en la figura 3.8.

La vista *consultaDinamica.cpt* muestra un formulario con varias opciones para restringir la búsqueda de actividades y permite elegir entre diferentes formas de ver los resultados. Las opciones seleccionadas por los usuarios son enviadas al archivo *consultaDinamica.cpt.metadata*.

El archivo *consultaDinamica.cpt.metadata* decide cuál es la siguiente vista que se mostrará dependiendo de lo que el usuario haya seleccionado.

Las vistas *VistaFormateada.pt*, *VistaTabla.pt*, *VistaTexto.pt* y *VistaEstadisticas.pt* muestran de diferente forma las actividades que coincidan con los parámetros de búsqueda proporcionados por el usuario.

3.6.2. Integración de componentes

Plone trabaja con componentes para reducir el esfuerzo realizado al desarrollar aplicaciones y que sean de mejor calidad.

La figura 3.9 muestra el componente *informeIMATE* y su relación con otros componentes. El componente *informeIMATE* coordina la interacción con los demás componentes.

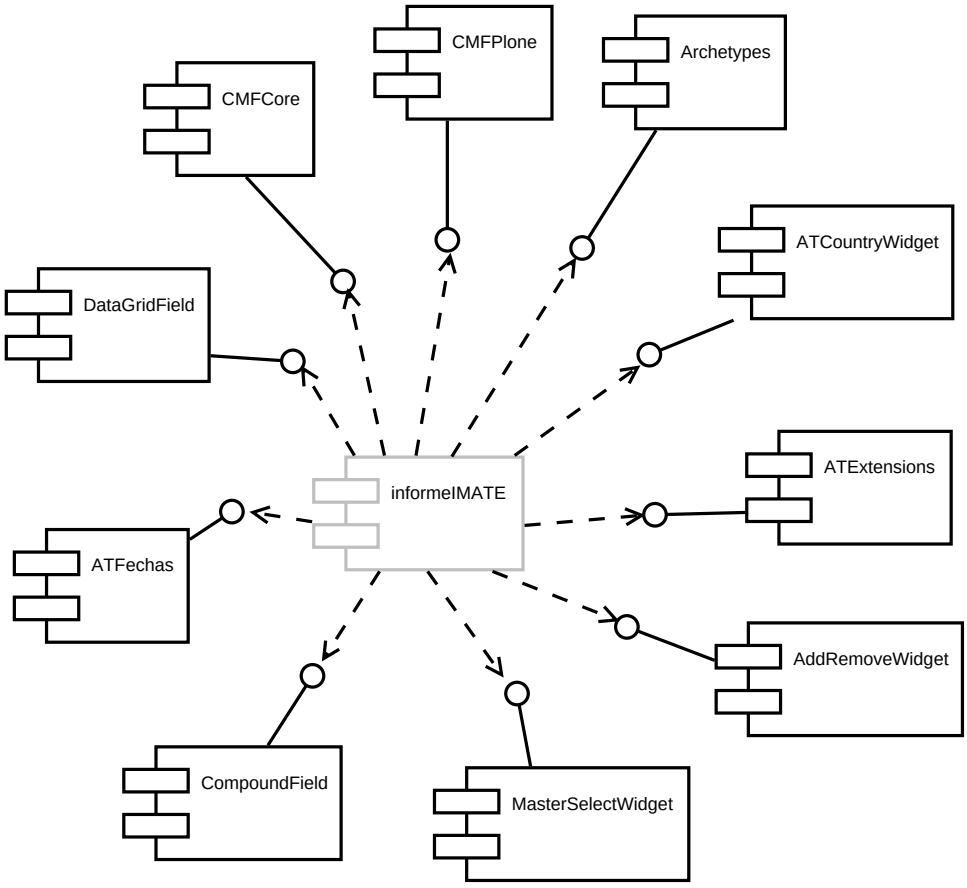

Figura 3.9: Diagrama de componentes del producto informeIMATE

Adicional a éste, se creó el componente *ATFechas*, el cual define un campo y un widget para almacenar fechas incompletas.

La figura 3.10 muestra el componente *ReportesIMATE* y su relación con otros componentes, donde está incluído el componente *informeIMATE*.

En el capítulo 4 se mostrará cómo realizar la conexión entre componentes para formar una aplicación personalizada.

Para una mayor referencia sobre el desarrollo utilizando arquetipos y su interconexión con otros componentes se recomienda [8] y [24].

3.7. Conclusiones

Este capítulo mostró cómo se implementó el sistema SCRICI siguiendo los requerimientos solicitados por el IMATE y aprovechando los componentes gratuitos disponibles para Plone.

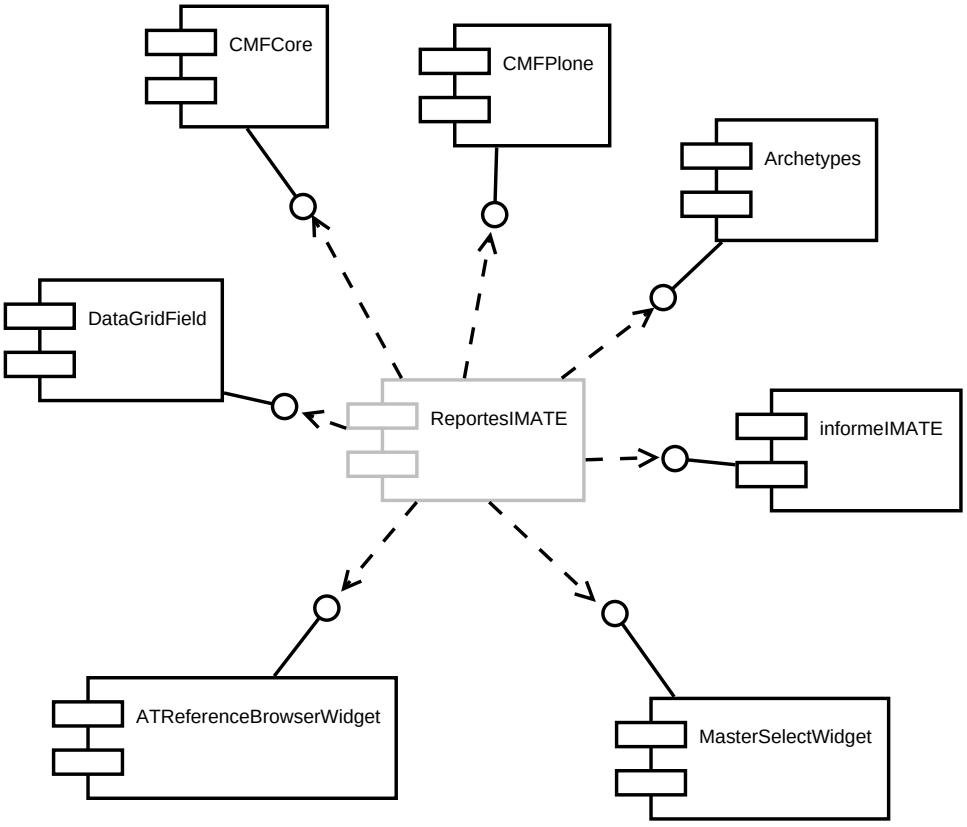

Figura 3.10: Diagrama de componentes del producto ReportesIMATE

En el desarrollo de este sistema se buscó compatibilidad con BibTex que es un estándar muy utilizado para hacer referencias bibliográficas, se siguió además la clasificación de la MSC para agrupar las actividades, además se agregaron varios campos importantes a los rubros que se venían manejando anteriormente para tener información mejor clasificada que se pueda intercambiar en un futuro con otros sistemas.

Al utilizar componentes se reduce mucho el trabajo a realizar, tanto al desarrollar como al diseñar, puesto que no es necesario conocer cómo funciona internamente cada componente sino qué es lo que hace y cómo integrarlo.

En este capítulo se mostraron los componentes utilizados y para qué sirven, pero no se mostró cómo se conectan y se configuran, puesto que ésto se mostrará en el siguiente capítulo.

Capítulo 4

Implementación del sistema SCRICI

4.1. Introducción

En este capítulo se verán algunos aspectos técnicos sobre el desarrollo de componentes y la implementación del sistema SCRICI con las siguientes finalidades: servir como guía a desarrolladores novatos y a los encargados de darle mantenimiento al sistema, puesto que Plone llega a ser difícil de entender al principio; y además para ampliar más la visión sobre la forma en que trabaja Plone, ya que éste es muy diferente a otros SMC y tiene una curva de aprendizaje muy prolongada, según se mencionó en el concurso de sistemas manejadores de contenido 2006, realizado por *Packt Publishing Ltd* [6].

4.2. Software utilizado

El software que se utilizó principalmente para el desarrollo del sistema *SCRICI* es: *ArgoUML*, *ArchGenXML* y *eclipse*. Este software permite reducir de manera considerable el tiempo requerido para desarrollar una aplicación sobre Plone facilitando esta tarea.

4.2.1. ArgoUML

ArgoUML [1] es una herramienta gratuita y de código abierto que permite hacer diagramas UML. Corre sobre java, lo que permite que corra sobre cualquier plataforma soportada por java.

ArgoUML almacena los diagramas UML en formato *XMI* [1] junto con información adicional de los diagramas dentro de un archivo con extensión *.zargo*. Este archivo es un zip del archivo XMI y de otros adicionales.

Así como esta herramienta, hay otras que también se pueden utilizar puesto que producen archivos en formato *XMI*, como lo son: *ObjectDomain* (comercial), *Poseidon* (comercial

[1] XMI o XML Metadata Interchange (XML de Intercambio de Metadatos) es un formato basado en XML que permite intercambiar metadatos entre herramientas de modelado de UML.

basada en ArgoUML), *Umbrello* (gratuita y de código abierto) y *Sybase Powerdesigner* (comercial). A pesar de que algunas de estas herramientas son comerciales, también se pueden encontrar versiones gratuitas de demostración.

Los diagramas que son de utilidad, dentro de los modelos de UML, para desarrollar sobre Plone son:

- **Diagrama de clases**. Con este diagrama se definen los contenidos y sus relaciones, así como métodos, esquemas, campos y tipo de contenido, entre otras.
- **Diagrama de estados**. Con este diagrama se define el workflow que tendrán los contenidos.

Para el sistema SCRICI sólo se hicieron diagramas de clases, donde se representan los contenidos, puesto que no fue necesario definir nuevos workflows.

El archivo obtenido con ArgoUML, u otra herramienta que produzca archivos con formato *XMI*, es utilizado como entrada de la herramienta ArchGenXML.

4.2.2. ArchGenXML

ArchGenXML [2] es una herramienta gratuita y de código abierto que permite crear aplicaciones sobre Plone basadas en los arquetipos. Estas aplicaciones se pueden agregar a Plone como productos para incrementar su funcionalidad. ArchGenXML no sólo genera las clases definidas en los diagramas de clases, sino que además genera otros archivos con métodos implementados.

ArchGenXML recibe como entrada un archivo UML con formato *XMI* (extensiones .xmi, .zargo o .zuml) y devuelve un producto para Plone listo para instalarse sin tener que haber programado nada, en relativamente poco tiempo. Este producto tiene una funcionalidad y vistas por default propias de los arquetipos. Si se requiere modificar la funcionalidad del producto se debe modificar éste utilizando Python y/o el lenguaje para plantillas ZPT.

Entre las vistas por default que tienen los productos creados con ArchGenXML se tienen las siguientes: para editar, visualizar y compartir contenido, además de modificar metadatos de éste, que son las proporcionadas por los arquetipos.

4.2.3. Eclipse

Eclipse [3] es un IDE [3] gratuito y de código abierto que permite desarrollar software dentro de un ambiente de trabajo que incluye: manejador de archivos, manejador de versiones, diferentes editores de texto y unidad de pruebas para depurar errores, entre otras.

Es más fácil programar en Python, así como en otros lenguajes, si se utiliza un IDE apropiado que revise la sintaxis, permita depurar errores de programación y proporcione un ambiente agradable con las herramientas necesarias para trabajar.

[2] Se puede obtener mayor información sobre ArchGenXML y descargar gratuitamente del sitio de Plone.

[3] IDE (Integrated Development Environment - Ambiente de desarrollo integrado). Es un software que proporciona un ambiente que permite a los programadores desarrollar software.

Al utilizar eclipse se puede mejorar mucho el tiempo de desarrollo de una aplicación para Plone ya que Python exige indentar los bloques de código que pertenecen a una estructura, además de respetar la demás sintaxis propia del lenguaje, y sin un editor adecuado, puede llegar a ser muy molesto programar y depurar errores, sobre todo cuando las líneas de código van aumentando.

Para trabajar eclipse con Python, se deben bajar otras tres herramientas gratuitas, que son: *PyDev*, *Subclipse* y *Webtools*. Unas vez configuradas correctamente, se tendrá en eclipse: un editor para Python, editor de HTML (que también sirve para ZPT), posibilidad de iniciar y detener Zope con sólo presionar un botón, realizar pruebas de productos, entre otras.

4.3. Creación del producto *informeIMATE*

Dentro del producto *informeIMATE* se definió el módulo de información curricular (MIC); un catálogo llamado *curriculum_catalogo* que indexa la información de los MIC; y una herramienta llamada *portal_curriculum* que permite importar/exportar información y definir el lugar a donde serán enviados los reportes de los investigadores.

Dentro de esta sección se mostrará cómo se hizo este producto sin entrar en profundidad en aspectos técnicos de programación o algoritmos.

4.3.1. Diagrama de clases en ArgoUML

Para definir los contenidos y crear una estructura básica primero se hicieron diagramas en ArgoUML, como se mencionó en la sección 4.2.1.

La figura 4.1 muestra el diagrama de clases del producto *informeIMATE* realizado con ArgoUML. En esta figura se muestran dos carpetas:

- **tool** Dentro de esta carpeta se define la herramienta *portal_curriculum* y el catálogo *curriculum_catalogo*. Sólo que ArchGenXML no maneja catálogos, así que el catálogo se tuvo que agregar en esta carpeta después de haber creado el producto.

  Nótese que a la clase *portal_curriculum* se le asignó el estereotipo [4] *tool*, que sirve para definir un objeto único dentro del sitio. La razón de tener un sólo objeto es que se guarde información en un sólo lugar, la cual pueda ser consultada por otros objetos del sitio. En este caso, la herramienta *portal_curriculum* guarda la ruta donde serán depositados los reportes de los investigadores y se lo comunica a los MIC del sitio.

- **content** Dentro de esta carpeta se define el MIC, que está formado por varias carpetas y contenidos.

  La carpeta raíz del MIC es una carpeta del tipo *FolderCurricula*, dentro de la cual hay 31 carpetas que representan los diferentes rubros. Por espacio sólo se muestran tres en la figura 4.1: *FolderProyectos*, *FolderPatrocinios* y *FolderTesis*. Además dentro

[4] Los estereotipos permiten clasificar los objetos de acuerdo a varios criterios.

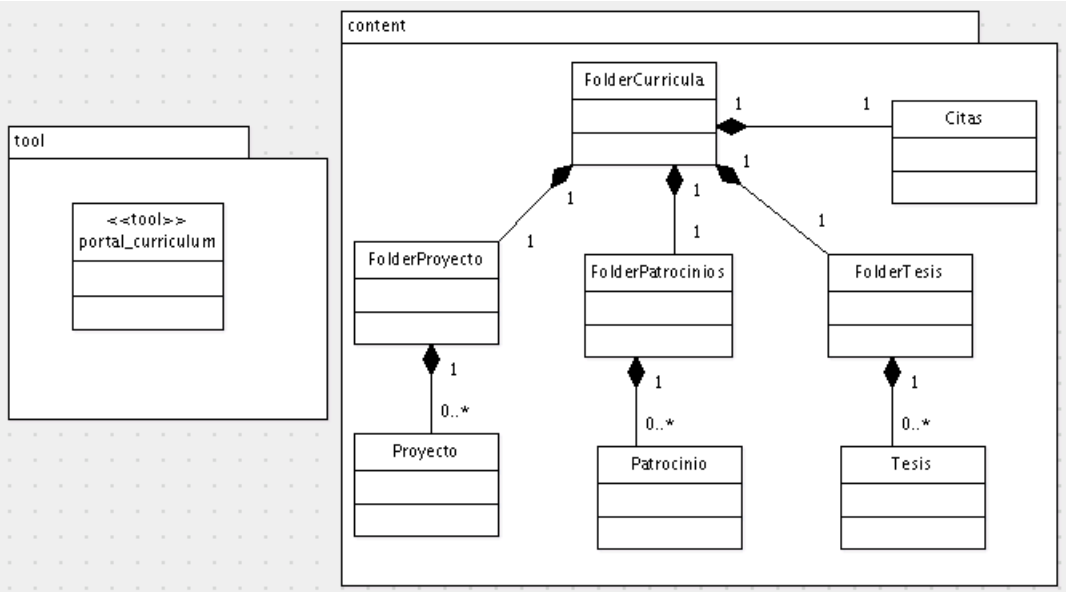

Figura 4.1: Diagrama de clases hecho en ArgoUML

de la carpeta raíz se guarda un contenido para hacer referencia a las citas de los investigadores. Dentro de cada carpeta que hace alusión a un rubro se guardan las actividades, por ejemplo, en la figura 4.1 se muestran las actividades *Proyecto*, *Patrocinio* y *Tesis*.

Nótese que no es necesario indicar cuáles clases sirven para definir carpetas y cuáles para definir contenidos mediante estereotipos, pues al utilizar la relación de *composición* (la línea que tiene un rombo relleno en un extremo) se le está indicando ésto implícitamente. Aunque también se le puede indicar explícitamente mediante los estereotipos *folder* y *content_class*, para denotar una carpeta y un contenido respectivamente.

Hasta ahora ya se definieron las carpetas y los contenidos que van a existir, pero no se han configurado. Para configurar el comportamiento de los objetos que serán instanciados a partir de las clases definidas en el diagrama, se deben agregar algunas propiedades con un valor asociado. Las propiedades que se pueden agregar dependen del tipo de contenido o estereotipo, y se puede encontrar una lista de éstas en [8] y [11].

La figura 4.2 muestra un ejemplo de algunas propiedades asignadas a la clase *FolderCurricula*, las cuales permiten definir su comportamiento. Dentro de algunas de sus propiedades se encuentran: *allowed_content_types*, que indica qué tipos de contenido admite el folder; y *global_allow*, que indica si el contenido puede ser agregado en cualquier lugar del sitio o sólo en ciertos lugares. Si no se definen algunas propiedades de configuración entonces toman un valor por default, por ejemplo, la propiedad *archetype_name* define el nombre con el que los usuarios van a conocer el tipo de contenido, sino se define esta

propiedad toma por default el nombre de la clase.

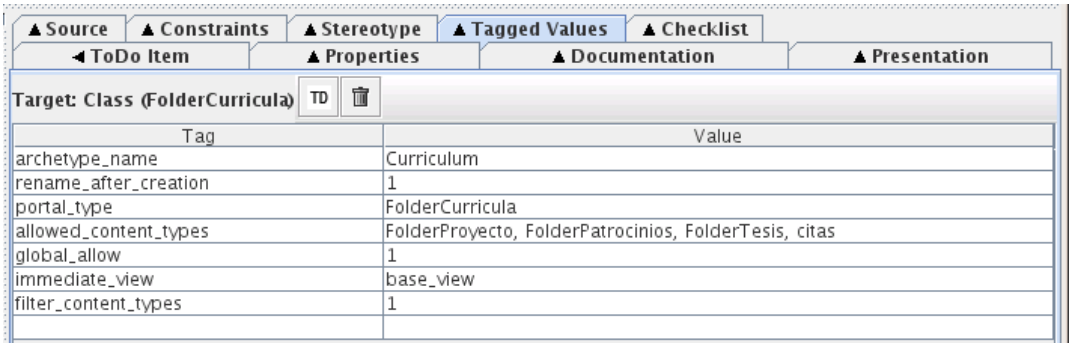

Figura 4.2: Atributos de una clase de contenido

Dentro de ArgoUML también se puede definir el esquema, que es un conjunto de campos. Para esto, cada campo se debe agregar como un atributo de la clase, y cada uno de éstos también se puede configurar mediante una serie de propiedades, de igual forma que las clases. La figura 4.3 muestra un ejemplo de algunas propiedades del campo *Tipo de proyecto*, que es un campo dentro del esquema del contenido *Proyectos*.

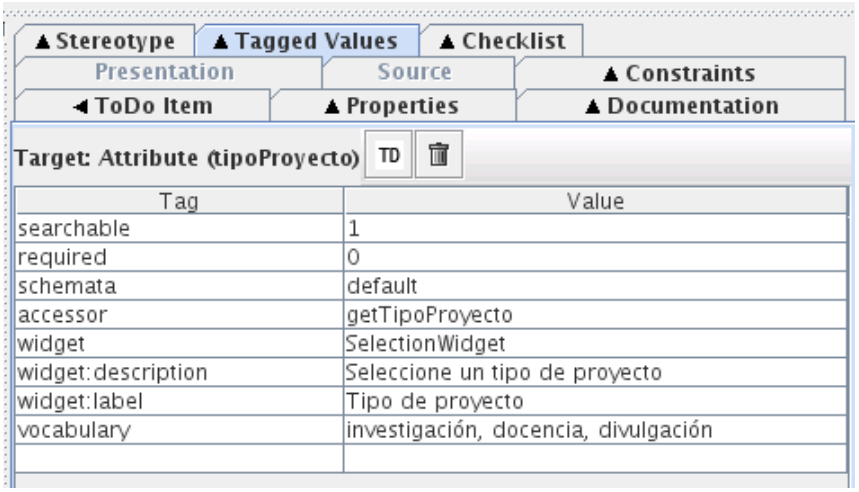

Figura 4.3: Atributos de un campo

De esta forma se debe hacer con cada contenido, tomando en cuenta que una carpeta también es un contenido, con la diferencia que puede guardar otros contenidos dentro. De la misma manera, la herramienta *portal_curriculum* se debe definir como un contenido que tiene un esquema, con la diferencia que sólo puede haber uno en el sitio.

4.3.2. Creación del producto con *ArchGenXML*

Una vez que ya se tiene un archivo con el modelo UML de las clases en formato .xmi, .zargo o .zuml, se debe utilizar ArchGenXML para crear el producto mediante una pequeña instrucción, similar a la siguiente, según el formato del archivo y nombre del producto:

```
$ ArchGenXML/ArchGenXML.py -o informeIMATE informeIMATE.zargo
```

La instrucción anterior produce como resultado un producto listo para instalarse, el cual está compuesto por una carpeta que tiene dentro otras carpetas y archivos, que se explican a continuación:

- **content** Dentro de esta carpeta se guardan las clases que hacen referencia a los contenidos. Por ejemplo, se guarda la clase *FolderCurricula*, *FolderProyecto*, *citas*, *TesisReference*, etc.
- **i18n** Dentro de esta carpeta se guardan archivos para manejar la traducción del contenido a diferentes idiomas.
- **Extensions** Dentro de esta carpeta se guardan archivos para manejar la instalación y desinstalación del producto.
- **skins** Inicialmente esta carpeta está vacía, pero dentro de ésta se pueden guardan las diferentes vistas del producto, así como scripts de Python que interactúan con éstas.
- **tool** Dentro de esta carpeta se guardan herramientas que puede utilizar el producto, si es que fueron definidas dentro del modelo UML.
- **config.py** Es un archivo que tiene varias variables de configuración, como permisos, nombre del producto, dependencias del producto, entre otras.

4.4. Implementación de casos de uso

En la sección 3.6.1 se mostraron las vistas y scripts utilizados para implementar los casos de uso, así como su interacción. En esta parte se va a mostrar más a detalle cómo crear las vistas y scripts para implementar los casos de uso. Dado que todos los casos de uso se implementan de forma similar, aquí sólo se mostrará cómo se creó el caso de uso: *Envío de reporte de actividades*.

Vista *envioInformeFecha.cpt*

El caso de uso *Envío de reporte de actividades* inicia con la vista *envioInformeFecha.cpt*, la cual permite seleccionar las fechas que abarcan las actividades y el tipo de reporte a enviar. Básicamente se compone la vista de un formulario hecho en HTML con posibilidad de seleccionar un tipo de reporte, una fecha inicial y una final. La figura 4.4

muestra esta vista, donde la parte que se realizó se encuentra marcada como *Sección principal*.

Nótese que esta vista además de mostrar la sección principal también muestra otras, como son: navegación, logotipo del sitio, eventos y noticias, entre otras. Estas secciones ya venían implementadas dentro Plone y algunas se personalizaron un poco; pero lo que importa es que no se tiene que estar escribiendo el código de estas secciones en todas las vistas, sino que al utilizar macros [5] sólo es necesario redefinir algunas secciones, reutilizando aquellas que ya fueron definidas anteriormente.

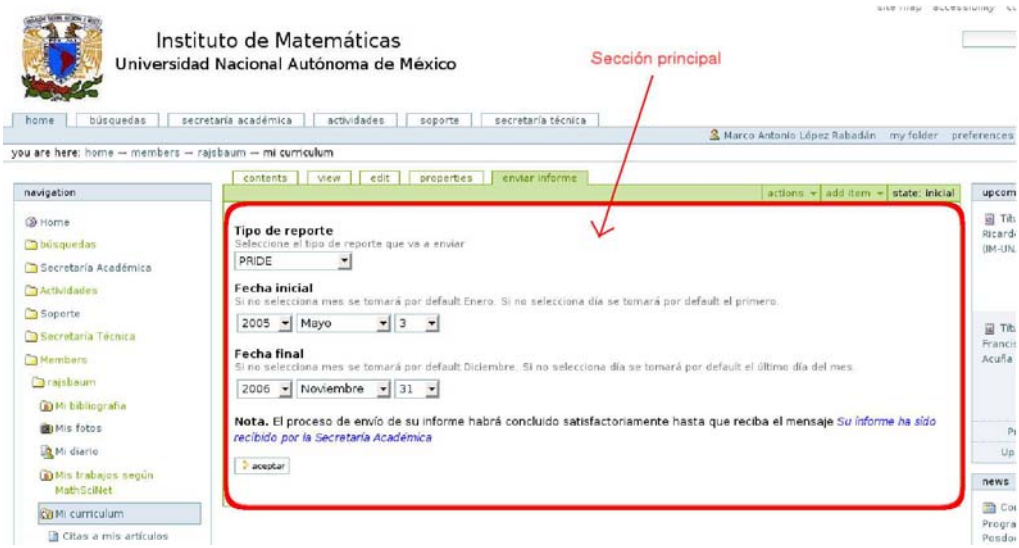

Figura 4.4: Vista envioInformeFecha.cpt

Para crear una vista con todas las secciones del sitio se debe utilizar una macro hecha en ZPT, llamada *master*. La sección que regularmente se redefine se llama *main*, que es la que está marcada como *Sección principal* dentro de la figura 4.4. La vista *envioInformeFecha.cpt*, así como todas las demás que se hicieron dentro del sistema SCRICI, sólo redefinen la sección principal, y tiene la siguiente forma:

```
<html metal:use-macro="here/main_template/macros/master">
   <div metal:fill-slot="main">
      ...código de la sección principal
   </div>
</html>
```

La vista *envioInformeFecha.cpt* envía la información ingresada por el usuario a un archivo del tipo *metadata*, el cual se encarga de decidir la siguiente vista o script a ejecutar.

[5] Una macro es un conjunto de instrucciones que se pueden reutilizar para evitar la tarea de volver a escribir el mismo código.

Para indicar que se le va a enviar la información a ese archivo, el cual se llama igual que la vista pero con extensión metadata, se debe poner el nombre de la vista sin extensión en el parámetro *action*, dentro de la etiqueta *form*, como se muestra a continuación:

```
<html metal:use-macro="here/main_template/macros/master">
  <div metal:fill-slot="main">
    ...
    <form action="envioInformeFecha" >
      ...Código dentro del formulario
    </form>
    ...
   </div>
</html>
```

De esta forma al presionar el botón de aceptar, la vista *envioInformeFecha.cpt* envía la información del formulario hacia el archivo *envioInformeFecha.ctp.metadata*.

Archivo *envioInformeFecha.cpt.metadata*

Los archivos *metadata* se encargan de decidir la siguiente acción a ejecutar, que puede ser mostrar una vista o ejecutar un script. Dentro del archivo *envioInformeFecha.cpt.metadata* se define qué acción tomar por cada botón que tenga el formulario, que en el caso de la vista *envioInformeFecha.cpt* sólo se tiene un botón llamado *aceptar*.

Además, los archivos *metadata* brindan la posibilidad de validar las formas para decidir la siguiente acción.

A continuación se muestra el archivo *envioInformeFecha.cpt.metadata*, que primero valida las fechas mediante un script llamado *valida_fechasEnvio.vpy*, si la validación fue exitosa solicita la vista *envio_informe.cpt*, y si no es exitosa vuelve a mostrar la vista *envioInformeFecha.cpt*.

```
[default]
title=Envío de informe

[validators]
validators..aceptar=valida_fechasEnvio

[actions]
action.success..aceptar=traverse_to:string:envio_informe
action.failure..aceptar=traverse_to:string:envioInformeFecha
```

Los archivos del tipo metadata se dividen en tres secciones: *default*, *validators* y *actions*. En la sección *validators* se definen los validadores que se van a ocupar, y en la sección *actions* se definen las acciones a tomar dependiendo el botón presionado por el usuario o lo que haya devuelto algún script y dependiendo del resultado de las validaciones.

En el archivo *envioInformeFecha.cpt.metadata*, dentro de la sección *validators*, se define que al presionar el botón aceptar se va realizar una validación mediante un script llamado *valida_fechasEnvio.vpy*. Dentro de la sección *actions* se define que si la validación fue exitosa y se presionó el botón *aceptar* se muestre la vista *envio_informe.cpt*, en caso de que la validación no sea exitosa y se haya presionado el botón *aceptar* se muestre nuevamente la vista *envioInformeFecha.cpt*.

Cabe mencionar que dentro del archivo metadata no hace falta poner la extensión de los archivos para indicar si se trata de un script o de una vista, ya que Plone busca tanto en las vistas como en los scripts hasta encontrar un archivo con el nombre dado dentro de ciertas rutas previamente especificadas.

Archivo *valida_fechasEnvio.vpy*

Los archivos con extensión *vpy* son scripts de Python que realizan validaciones sobre los formularios definidos en las vistas. A continuación se muestra sólo un fragmento del script *valida_fechasEnvio* que se encarga de validar las fechas. Primero se obtienen los valores del formulario y después de haber revisado si son correctos se regresa una variable llamada *state* donde se indica si la validación fue correcta o incorrecta.

```
...
fechaInicialYear=context.REQUEST.get("fechaInicial|Year",'')
fechaFinalYear=context.REQUEST.get("fechaFinal|Year", '')
...
if (correcto):
   return state.set(status='success')
return state.set(status='failure')
```

Los campos que se obtienen del formulario son: `fechaInicial|Year` y `fechaFinal|Year`. Después de verificar si los valores proporcionados tienen valores válidos se regresa la variable *state* poniéndole el valor *success* si fue correcta la validación o *failure* en caso contrario. Como ya se mencionó anteriormente, el valor de esta variable servirá para que el archivo *envio_informe.cpt.metadata* tome la decisión de la siguiente acción a realizar.

Archivo *envio_informe.cpt*

Este archivo define una vista donde se le muestran al usuario las actividades que fueron elegidas por el sistema según los años proporcionados. Lo interesante de esta vista es la interacción que tiene con el MIC. Para esto se utiliza la variable *here* o *context*, que hacen alusión a la ruta actual del navegador.

Plone utiliza una estructura de árbol, por lo que para llegar a un contenido particular se debe conocer la ruta del objeto. Para invocar una vista primero se pone la ruta de un objeto, sobre el que va a aplicar la vista, y después el nombre de la vista. Por ejemplo, si se proporciona la ruta *sa/Members/marco/curriculum/envio_informe* se estará invocando la vista *envio_informe.cpt* en el contexto del objeto *curriculum*, cuya URL completa es

sa/Members/marco/curriculum. En este caso la variable *context* y *here* harán alusión al objeto *curriculum*.

Utilizando la variables *here* y *context* se pueden mandar a llamar funciones de los contenidos. Por ejemplo, la vista *envio_informe* manda a llamar la función *doReporte* del objeto *curriculum* para obtener todas las actividades que fueron vigentes entre las fechas proporcionadas, e imprime estas actividades para que el usuario las confirme.

A continuación se muestra un fragmento del archivo *envio_informe.cpt* donde se manda a llamar la función *doReporte*.

```
<span tal:define="informe python:here.doReporte(fechaIni, fechaFin)">
     Aquí se imprimen las actividades para
     que el usuario las confirme
</span>
```

De esta forma una misma vista puede ser utilizada por diferentes objetos dentro del sitio, siempre y cuando las llamadas a las funciones sean válidas para el objeto referenciado por la variable *here* o *context*. Es decir, si esta vista se trata de utilizar con otro objeto, se lanzará un error si este objeto no tiene la función *doReporte*.

Archivo *script_envio*

Este archivo es un script que se encarga de copiar las actividades seleccionadas por el investigador a una carpeta común. Para esto se auxilia de la herramienta *portal_curriculum* quien proporciona la ruta de la carpeta destino.

Para acceder a la herramienta *portal_curriculum*, el MIC tiene la función *getToolCurriculum* que devuelve esta herramienta, la cual tiene las funciones *getCarpetaPride*, *getCarpetaPaipa* y *getCarpetaInformeAnual*, para obtener las rutas de los diferentes reportes. A continuación se muestra un fragmento del archivo *script_envio* donde se muestra esto:

```
...
tool = context.getToolCurriculum()
if tipo_reporte == 'PRIDE':
   carpeta_destino = tool.getCarpetaPride()
elif tipo_reporte == 'PAIPA':
   carpeta_destino = tool.getCarpetaPaipa()
elif tipo_reporte == 'ANUAL':
   carpeta_destino = tool.getCarpetaInformeAnual()
...
```

Cabe mencionar que la herramienta *portal_curriculum* es un contenido único dentro del sitio, el cual debe ser editado antes de enviar el reporte de actividades para que éste regrese una carpeta válida.

4.5. Restricción de actividades

Cada tipo de usuario tiene ciertas actividades que puede realizar dentro del sistema SCRICI, las cuales son restringidas mediante permisos.

Plone ofrece varias formas de restringir las actividades que pueden realizar los usuarios por medio de permisos. A continuación se muestran las que se utilizaron dentro del sistema SCRICI.

Al agregar contenido

Dentro del código para instalar un producto se puede definir el permiso que se debe de tener para agregar los contenidos de dicho producto. De esta forma, sólo el usuario que tenga ese permiso podrá agregar los contenidos definidos en el producto.

ArchGenXML crea un archivo llamado permissions.py o config.py donde se tiene una variable llamada *DEFAULT_ADD_CONTENT_PERMISSION* la cual define el permiso para agregar contenido. En caso de querer cambiar este permiso, se puede modificar esta variable.

Dentro del producto *informeIMATE* se le asignó a la variable *DEFAULT_ADD_CONTENT_PERMISSION* el valor: *'Add curriculum content'*; y dentro del producto *ReportesIMATE* se le asignó el valor: *'Add reporte content'*. De esta forma, con el permiso *'Add curriculum content'* se controla quién puede agregar contenido dentro de los MIC y quién puede agregar un MIC. De igual forma, con el permiso *'Add reporte content'* se controla quién puede agregar consultas especializadas sobre las actividades de los MIC.

Además, para definir permisos iniciales para un contenido dentro de un producto se puede utilizar la función *setDefaultRoles*, que pertenece al producto *CMFCore*, la cual define permisos iniciales para un rol dentro de un contenido. A continuación se muestra un fragmento del archivo *config.py* del producto *informeIMATE*.

```
from Products.CMFCore.CMFCorePermissions import setDefaultRoles
...
DEFAULT_ADD_CONTENT_PERMISSION = "Add curriculum content"
ROL_ADMIN_MIC="secretaria_ac"
setDefaultRoles(DEFAULT_ADD_CONTENT_PERMISSION,
                                        ('Manager', ROL_ADMIN_MIC))

...
```

En el fragmento de arriba se muestra la definición del permiso `DEFAULT_ADD_CONTENT_PERMISSION`, que permite agregar contenido. También se define el rol de administración de MIC con la variable `ROL_ADMIN_MIC`. Por último se definen permisos iniciales para los objetos del producto *informeIMATE*, de forma que los roles *Manager* y *ROL_ADMIN_MIC* tendrán el permiso de agregar contenido.

Al acceder a una vista

Plone trae una herramienta llamada *portal_membership*, la cual tiene una función llamada *checkPermission* que permite identificar si el usuario actualmente identificado en el sistema tiene cierto permiso sobre un objeto. A continuación se muestra cómo se revisa dentro de una vista si un usuario tiene el permiso *Modify portal content* sobre un contenido ligado con la variable *context*. En caso de que el usuario no tenga el permiso *Modify portal content*, se le mostraría un error de permisos insuficientes.

```
<span tal:condition="python: \
         not (checkPermission('Modify portal content', context)"
   tal:replace="python: context.raiseUnauthorized()" />
```

Dentro de las vistas del sistema SCRICI se revisa si los usuarios tienen permiso de verlas. Esto evita que los usuarios realicen acciones no permitidas, como editar contenidos o importar información dentro de un MIC ajeno.

Cabe mencionar que la función *checkPermission* también se puede utilizar dentro de los métodos de las clases de los contenidos, pero hay una manera más conveniente de controlar el acceso a estos métodos, como se verá a continuación.

Al invocar un método

A pesar de que se pueden restringir los scripts o vistas para que no las puedan acceder los usuarios sin autorización, es posible que un usuario invoque un método perteneciente a un objeto desde la URL; o que una vista o script invoquen un método al cuál no deberían de poder acceder.

Sino se restringen los métodos, cualquier persona podría invocarlos y en cada método se tendría que revisar si el usuario tiene o no permiso de acceso. Para evitar esto, y dado que Python no maneja control de acceso a los métodos, Zope tiene una clase llamada *ClassSecurityInfo* que implementa el control de acceso a los métodos permitiendo tener métodos privados, públicos y protegidos de la siguiente forma:

- **Público.** Todos pueden acceder a este método, incluso se puede acceder desde la URL.
- **Privado.** Sólo puede se accesado después de haber traspasado la barrera de seguridad, es decir, desde dentro de otro método del mismo objeto.
- **Protegido.** Si el usuario tiene cierto permiso entonces puede acceder al método.

Para declarar que un método es privado, público o protegido se utilizan los métodos *declarePrivate*, *declarePublic* y *declaraProtected*, respectivamente.

A continuación se muestra un ejemplo donde se declara una clase llamada *articulo* con tres métodos: el primero público, la segundo privado y el tercero protegido por el permiso *'Modify portal content'*:

```
from AccessControl import ClassSecurityInfo

class articulo(BaseContent):
   security = ClassSecurityInfo()

   security.declarePublic('getFieldValue')
   def getFieldValue(self, field_name):
      ... Cuerpo del método ...

   security.declarePrivate('getClassName')
   def getClassName(self):
      ... Cuerpo del método ...

   security.declareProtected('Modify portal content', 'edit')
   def edit(self):
      ... Cuerpo del método ...
```

El acceso a los métodos de los contenidos del sistema SCRICI está controlado de esta forma para que sólo los usuarios autorizados realicen las acciones permitidas.

4.6. Uso del catálogo

Como se mencionó en la sección 4.3, el catálogo no se puede crear con ArgoUML y ArchGenXML, así que se debe agregar manualmente. En esta sección se explica cómo hacer esto y cómo utilizar el catálogo para hacer consultas sobre las actividades indexadas.

4.6.1. Agregar un catálogo

El catálogo se agregó en la carpeta *tool*, dentro del producto *informeIMATE*. Dentro de esta carpeta se agregó un archivo llamado *CatalogTool.py*, donde se define el catálogo.

Basta con heredar de la clase *CatalogTool* que se encuentra dentro del producto *CMFPlone* para tener una clase que se puede instanciar en un sólo catálogo. Además se le debe definir un identificador para poderlo localizar y redefinir el método *manage_afterAdd*, el cual se manda a llamar después de agregar el catálogo al sitio. Dentro de este método se le va a indicar al catálogo que solamente indexe los contenidos del MIC. Con esto ya se puede instanciar un catálogo que indexe las actividades de los MIC, pero aún no se han definido índices ni metadatos.

A continuación se muestran las partes más relevantes de la definición del catálogo:

```
from Products.CMFPlone.CatalogTool import CatalogTool
from Products.informeIMATE.config import contenidosMIC
...

class CatalogCurriculumTool(CatalogTool):
```

```
    id = 'curriculum_catalogo'
    ....

    def manage_afterAdd(self, item, container):
        portal = getToolByName(self,'portal_url').getPortalObject()
        apply_path='/'.join(portal.getPhysicalPath())
        self.ZopeFindAndApply(portal, obj_metatypes=contenidosMIC, search_sub=1,
                apply_func=self.catalog_object, apply_path=apply_path)
```

En la primer línea se importa la clase *CatalogTool* para poder heredar de ella, después se importa la variable *contenidosMIC*, que es una tupla con los nombres de las actividades que se van a indexar. Después se define la clase *CatalogCurriculumTool* la cual hereda de la clase *CatalogTool*. Después se define el atributo *id* que es el nombre con el que se podrá buscar el catálogo cuando se quiera ocupar. Y por último se define la función *manage_afterAdd*, la cual será llamada sólo una vez al momento de agregar el catálogo, ésta sirve para indicarle al catálogo que sólo indexe las actividades de los MIC.

4.6.2. Definición de índices

Los índices se agregaron de dos formas: al agregar el catálogo y al definir cada uno de los contenidos del MIC. Estas dos formas se muestran a continuación.

Definición de índices al definir los contenidos

Para agregar un índice al catálogo el cual hace referencia a un campo de un contenido se debe configurar la propiedad *index*, perteneciente al campo. Esta propiedad se configura poniendo primero el nombre del catálogo, seguido por el tipo de índice y por último se puede indicar si además se quiere agregar ese campo como un metadato dentro del catálogo.

A continuación se muestra un ejemplo de cómo se agregó un índice encargado de indexar el campo *'Tipo de artículo'*, desde ArgoUML.

El código generado por ArchGenXML sobre el campo *'Tipo de proyecto'* se muestra a continuación.

```
schema=Schema((
   ...
   StringField('tipoProyecto',
        searchable=1,
        required=0,
        schemata='default',
        index='curriculum_catalogo/FieldIndex:brains',
        widget=SelectionWidget(label="Tipo de proyecto',
            description='Seleccione un tipo de proyecto',
            ),
        accessor='getTipoProyecto',
        ),
```

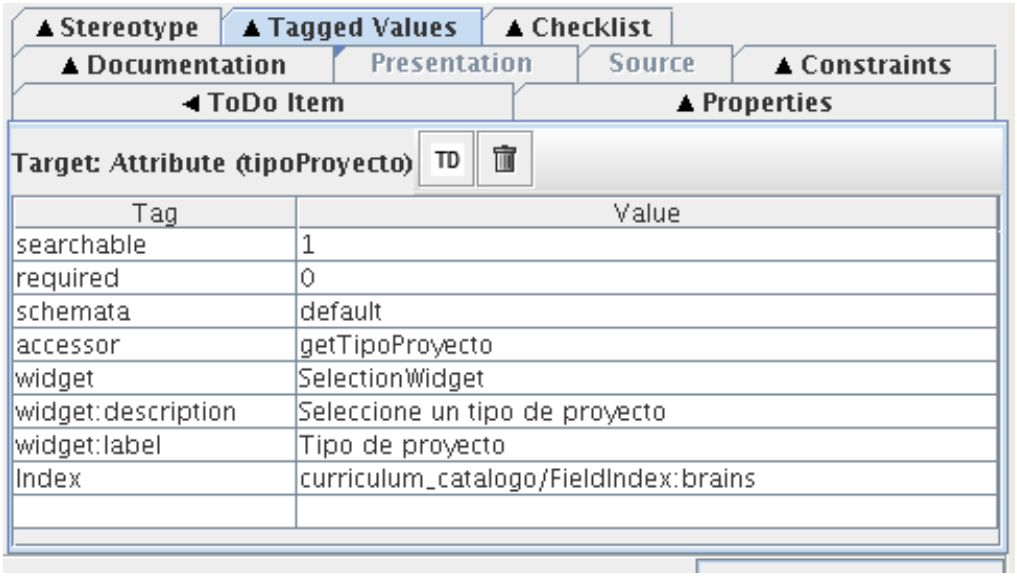

Figura 4.5: Definición de un índice desde ArgoUML

...

Con la propiedad *index*, del ejemplo anterior, se está indicando que se va a utilizar el catálogo *curriculum_catalogo* para indexar el campo con un índice del tipo *FieldIndex* y además con la opción *:brains* se le dice que agregue ese campo como un metadato dentro del catálogo. El catálogo va a buscar todos los objetos que tengan la función definida con la propiedad *accessor*, para almacenar su resultado. En el caso de los campos, el *accessor* devuelve el valor almacenado.

Definición de índices al agregar el catálogo

Cada que se instala un producto en Plone, se manda a llamar la función *install* del script llamado *Install.py* dentro del folder *Extensions*. Dentro de esta función se agregó la instalación del catálogo y además se agregaron algunos índices al catálogo.

El código para agregar un catálogo y sus índices es un poco extenso y por eso no se mostrará aquí. Pero básicamente primero se agrego el catálogo si no existía ya, después se agregaron los índices y metadatos si es que no existían ya, como se muestra a continuación:

```
1 if no existe catálogo then
2     Agregar catálogo

3 for each índice a agregar do
4     if no existe índice then
5         Agrega índice

6 for each metadato a agregar do
7     if no existe metadato then
8         Agregar metadato
```

Cabe mencionar, que cada que un contenido es modificado, éste se vuelve a reindexar, actualizando todos sus índices y metadatos con los nuevos valores obtenidos.

En la figura 4.6 se muestran algunos de los índices del catálogo *curriculum_catalogo*. La primer columna muestra el nombre de los índices; la segunda el tipo de índice; la tercera el número de valores distintos catalogados; y la cuarta la fecha de actualización del índice. Como se puede observar, aparece el índice *getTipoProyecto* que se agregó en el ejemplo anterior.

getNivel	FieldIndex	8	2007-07-26 22:16
getNoProyecto	FieldIndex	123	2007-07-26 22:16
getObjPositionInParent	FieldIndex	1	2007-07-26 22:16
getPais	FieldIndex	38	2007-07-26 22:16
getPaisVisita	FieldIndex	25	2007-07-26 22:16
getPatrocinador	FieldIndex	61	2007-07-26 22:16
getResponsableInterno	FieldIndex	1	2007-07-26 22:16
getStatus	FieldIndex	4	2007-07-26 22:16
getStatusTesis	FieldIndex	3	2007-07-26 22:16
getTipo	FieldIndex	54	2007-07-26 22:16
getTipoActividad	FieldIndex	6	2007-07-26 22:16
getTipoArticulo	FieldIndex	3	2007-07-26 22:16
getTipoCorresponsable	FieldIndex	4	2007-07-26 22:16
getTipoInvestigacion	FieldIndex	1	2007-07-26 22:16
getTipoLibro	FieldIndex	4	2007-07-26 22:16
getTipoProyecto	FieldIndex	4	2007-07-26 22:16
getTipoResponsable	FieldIndex	1	2007-07-26 22:16

Figura 4.6: Índices dentro del catálogo

4.6.3. Consulta de información

Para consultar información de la base de datos a través del catálogo se puede hacer utilizando una función propia del catálogo llamada *searchCatalog*, la cual recibe como

parámetros los índices del catálogo.

Dependiendo del tipo de índice que se utilice es el tipo de búsqueda que se puede realizar. En la sección 2.16 se explicaron algunos índices.

A continuación se muestra un ejemplo para consultar todos los artículos científicos dentro del sitio. La primera instrucción sirve para obtener el catálogo *curiculum_catalogo*, y la segunda para realizar la consulta sobre los índices *portal_type* y *getTipoArtículo*. La consulta regresa referencias a aquellas contenidos cuyo atributo *portal_type* sea *articulo* y cuyo campo *getTipoArticulo* sea *científico*.

```
curriculum_catalogo=getToolByName(self, 'curriculum_catalogo')
consulta=curriculum_catalogo.searchCatalog(portal_type="articulo",
                getTipoArticulo="científico")
```

Cabe mencionar que el resultado de la consulta no devuelve los contenidos que cumplan las condiciones, sino objetos más pequeños que tienen referencias a los contenidos. Esas referencias tienen como atributos los metadatos definidos.

Entre más refinada se quiera la consulta se deben ir agregando más índices; por ejemplo, para buscar además los artículos publicados y nacionales se deberían agregar otros dos índices. Pero entre más índices se tengan más tiempo toma el catálogo en reindexar sus objetos y entre más metadatos tenga, más tiempo tarda en recuperar los objetos del catálogo. Por eso se debe examinar cuidadosamente cuáles realmente se requieren.

Las consultas del producto *ReportesIMATE* se hicieron mediante formularios con varias opciones, casi una por índice, y el valor elegido por el usuario se le pasa como parámetro al catálogo mediante la función *searchCatalog*. Sin embargo, no todas las opciones de búsqueda se le pueden pasar a la función *searchCatalog*, y en estos casos se utilizan los metadatos para realizar la búsqueda.

Cuando se requieren consultas sobre las actividades que entren dentro de un rango de fechas, primero se hace la consulta al catálogo por los demás parámetros de búsqueda y posteriormente los resultados se meten a un filtro que descarta las actividades por rango de fechas, valiéndose de los metadatos. Para esto se crearon los metadatos: *fechaInicial* y *fechaFinal* que indica cuando inició y terminó una actividad, respectivamente.

A continuación se muestra un fragmento código de cómo se pueden realizar las consultas por rango de fechas utilizando los metadatos:

```
cc=getToolByName(self, 'curriculum_catalogo')
brains=cc.searchCatalog(**parametrosBusqueda)
if fInicial and fFinal:
  res=[]
  for i in brains:
    iniBrain=i.fechaInicial
    finBrain=i.fechaFinal

    if (iniBrain <= fInicial or iniBrain <= fFinal) and \
             (finBrain >= fInicial or finBrain >= fFinal):
```

```
        res.append(i)
  brains=res
return brains
```

En el ejemplo anterior, las dos primeras líneas permiten consultar el catálogo pasándole como parámetro una variable con varias opciones de búsqueda. Después si se tiene un rango de fechas, es decir, una fecha inicial y una final, entonces se obtienen los metadatos *fechaInicial* y *fechaFinal* para finalmente comparar aquellos que si pudieron ocurrir dentro del rango de fechas a buscar.

El producto *ReportesIMATE* permite realizar consultas sobre las actividades de los MIC, pero si se requiriera alguna consulta no contemplada dentro del esquema de este producto se tendría que realizar un script en Python y consultar los catálogos, aunque para esto se requiere un conocimiento más técnico sobre la forma en que se encuentra organizada la información para consultar los índices y metadatos correctos.

4.7. Conclusiones

Para el desarrollo de aplicaciones sobre Plone se recomienda mucho la utilización de arquetipos en conjunto con las herramientas ArgoUML y ArchGenXML, puesto que permiten ahorrar mucho tiempo de codificación, ya que generan un producto completo y listo para instalarse, sin haber escrito una línea de código. En el caso del sistema SCRICI, después de haber generado el producto, fue necesario realizar varias modificaciones a éste para que cumpla con todos los requisitos solicitados. Además, para modificar un producto, se recomienda un IDE como eclipse para ahorrar mucho tiempo en la codificación y depuración de errores.

Este capítulo es un complemento de los capítulos anteriores, en el que se mostraron aspectos claves del desarrollo sobre Plone, con la finalidad de se entienda mejor el presente trabajo. Además, tiene la finalidad de servir como guía para quienes quieran realizar un sistema sobre Plone utilizando arquetipos o modificar el actual sistema SCRICI. Debido a esto, no se profundizó en todas las opciones de configuración de los componentes utilizados, para mayor información se recomiendan las siguientes páginas Web:

- `http://plone.org/documentation/manual/archetypes-developer-manual`
- `http://plone.org/documentation/tutorial/archgenxml-getting-started`
- `http://plone.org/products/archetypes/documentation/old/quickref`
- `http://plone.org/documentation/manual/definitive-guide`

Capítulo 5

Resultados y conclusiones

5.1. Introducción

En esta sección se mostrarán los resultados obtenidos con el desarrollo del sistema SCRICI. Para obtener estos resultados se realizaron algunos cuestionarios, que se pueden encontrar en los anexos A, B y C; se hicieron algunas pruebas de rendimiento, que se encuentran más detalladamente en el anexo D; y se analizó el cumplimiento de los objetivos del sistema.

Posteriormente se muestran las conclusiones de la tesis y se habla sobre el trabajo futuro para el sistema SCRICI.

5.2. Resultados

Se realizó un cuestionario a los investigadores del IMATE para conocer cuál fue su experiencia con el sistema SCRICI hasta el momento. De los 101 investigadores, 10 contestaron el cuestionario que se muestra en el anexo C, del que se obtuvieron las siguientes conclusiones, siendo 1 la calificación más baja y 5 la calificación más alta:

- En promedio, la facilidad de uso del sistema SCRICI fue calificada con 4, es decir, que es fácil de usar.
- En promedio, la facilidad para cambiar entre la forma anterior de reportar los reportes de actividades y la actual forma fue calificada con 4.33, es decir, les pareció fácil y muy fácil el cambio.
- Lo beneficios de la importación y exportación de información de los MIC lo calificaron con 4.55, es decir, que les brinda varios o muchos beneficios.
- La calificación que le dieron al tiempo empleado para realizar el reporte de actividades fue de 3.88, es decir, entre que no mejoró y que mejoró un poco.
- El 90 % si recomendaría el sistema para que otras instituciones lo usen.

- El 44.44 % actualiza su información conforme van realizando sus actividades a lo largo del año. Es decir, menos de la mitad lo actualiza durante el año. Si se actualizara a lo largo del año, al final de éste se podría enviar el reporte de actividades sin mayor trabajo, pero se debe considerar que es algo a lo que se están apenas acostumbrando.

- El 88.88 % utiliza el sistema durante el año para consultar información. Lo que muestra la importancia del sistema.

Se cumplieron todos los objetivos propuestos al inicio de la tesis como se muestra a continuación:

- Se redujo el tiempo y trabajo necesario para realizar los diferentes reportes. Aproximadamente, antes del sistema SCRICI, Ángel Carrillo utilizaba un mes y medio en obtener la información, gráficas y cálculos necesarios para redactar el informe anual de actividades. En el informe anterior, Mónica Leñero realizó esta tarea utilizando el sistema SCRICI, y se tardó un poco menos de un mes. Es decir, se redujo aproximadamente un 33 % el tiempo para realizar esta tarea. En los anexos A y B se encuentran los cuestionarios realizados a Mónica Leñero y Ángel Carrillo, donde indican el tiempo utilizado para realizar la tarea mencionada.

  En cuanto al reporte del PRIDE y PAIPA, el IMATE no realiza un documento entregable, como es el caso del informe anual, sino que una comisión llamada *comisión PRIDE* o *comisión PAIPA* revisa los reportes de cada investigador. En este caso la ventaja que tienen las comisiones es que pueden revisar un poco más rápido los reportes, puesto que pueden ver directamente los resúmenes de sus actividades. En el caso de los investigadores, si tienen actualizadas sus actividades, el reporte se puede enviar inmediatamente, lo que ahorra tiempo y esfuerzo.

- Se disminuyó la intervención humana en la generación de los reportes, puesto que antes se tenía que organizar y capturar la información dentro de hojas de cálculo. Para realizar el informe anual anterior, utilizando el sistema SCRICI, ya no se tuvo que hacer ésto, sino que utilizando diferentes criterios de búsqueda se obtuvieron directamente las hojas de cálculo del sistema y además se obtuvieron varios de los anexos, que son datos de cada actividad con cierto formato.

- El sistema SCRICI permite la importación de información para ahorrarle tiempo al investigador y aprovechando la información disponible en formatos estándares. Las actividades de los investigadores se pueden exportar utilizando los formatos: BibTex, EndNote [1] y XML (MODS) [2]. La referencia a la mayoría de los artículos y libros publicados se encuentra disponible en BibTex, así que fácilmente se pueden subir las publicaciones de los investigadores importándolas.

  Además, según el cuestionario que se realizó a los investigadores, los que lo contestaron opinan que esta opción les brinda varios o muchos beneficios. Sin embargo,

[1] EndNote es un software comercial para el manejo de referencias y bibliografías.

[2] MODS (Metadata Object Description Schema o Esquema de descripción de objeto de metadatos) es un esquema de descripción bibliográfica basada en XML.

varios de ellos no se animaron a utilizar esta opción. Más información al respecto se puede encontrar en el anexo C.

- Se puede acceder al sistema desde cualquier lugar con conexión a Internet. Esto permite agilizar el envío y recepción de información. Además la comunicación es segura pues es cifrada con SSL utilizando Apache.

  Anteriormente se tenía el problema de que algunos investigadores entregaban tarde su reporte de actividades, y si se suma a esto el tiempo de envío, el retraso era aún mayor. Además, el envío de información por paquetería es vulnerable a retrasos y extravío; mientras que mediante una conexión segura por Internet la respuesta es inmediata, confiable y sin un costo de envío, lo cual permitió ahorrar tiempo y dinero.

- Al momento de que un investigador está a punto de enviar su reporte de actividades el sistema le muestra las actividades que se van a reportar marcando aquellas que ocurrieron uno o más años atrás y, que aún están vigentes. De esta manera el investigador no tiene que preocuparse por volver a ingresar las actividades anteriormente reportadas dentro de su nuevo reporte, además de que le permite detectarlas en caso de que se le haya olvidado ponerle fecha final a alguna de éstas. Esto también disminuye el tiempo empleado por el investigador para realizar su reporte de actividades, aunque por tratarse de la primera vez que se utilizó el sistema SCRICI, no se vio tan notorio este año.

- Evita capturas redundantes. Una vez que un investigador registró sus actividades realizadas, básicamente tiene que seleccionar las fechas a reportar y dar clic en *aceptar* para enviar su reporte de actividades para el PRIDE, PAIPA o para el informe anual de actividades. Es decir, se aprovecha la información ya disponible, ahorrándole tiempo al investigadores para entregar cada reporte, y al IMATE al tener organizada la información de forma que la pueda explotar eficientemente. En este año 17 investigadores enviaron su reporte de actividades par el PRIDE y obtuvieron los beneficios de no volver a ingresar la información del 2007 que ya habían ingresado para su reporte anual.

Se utilizó un catálogo llamado *curriculum_catalogo* para indexar sólo las actividades de los MIC, el cual trajo ventajas y desventajas. Cabe mencionar que era necesario agregar varios índices y metadatos sobre las actividades de los MIC dentro de algún catálogo, pero además todos los objetos de un sitio Plone deben estar catalogados dentro del catálogo llamado *portal_catalog*.

A continuación se muestran las ventajas y desventajas de agregar los índices y metadatos, ya sea dentro del *portal_catalog* o dentro del *curriculum_catalogo*. Para obtener estos resultados se realizaron 100 veces pruebas con los índices y metadatos dentro del *portal_catalog* y 100 veces pruebas con los índices y metadatos dentro del *curriculum_catalogo*. Estas pruebas se pueden ver más a detalle en el anexo D.

- Se consultaron sólo los objetos de los MIC dentro sitio y se cargaron en memoria. En promedio se tardó 22.39 segundos con los índices y metadatos dentro

del *portal_catalog* y 19.95 segundos con los índices y metadatos dentro del *curriculum_catalogo*. Es decir, se tuvo una mejora del 10.89 % al utilizar el *curriculum_catalogo*.

- Se consultaron todos los objetos del sitio y se cargaron en memoria. En promedio se tardó 22.36 segundos con los índices y metadatos dentro del *portal_catalog* y 24.21 segundos con los índices y metadatos dentro del *curriculum_catalogo*. Es decir, se tuvo una mejora del 7.64 % al utilizar el *curriculum_catalogo*.

- Se reindexaron objetos que no pertenecen a los MIC. En promedio se tardó 68.78 segundos con los índices y metadatos dentro del *portal_catalog* y 59.60 segundos con los índices y metadatos dentro del *curriculum_catalogo*. Es decir, se tuvo una mejora del 13.34 % al utilizar el *curriculum_catalogo*.

- Se reindexaron sólo los objetos de los MIC. En promedio se tardó 62.85 segundos con los índices y metadatos dentro del *portal_catalog* y 81.37 segundos con los índices y metadatos dentro del *curriculum_catalogo*. Es decir, se tuvo un empeoramiento del 22.76 % al utilizar el *curriculum_catalogo*.

Finalmente, cabe mencionar que el sistema SCRICI cumplió con todos los requisitos solicitados por el IMATE. En el capítulo 3 se mostraron los requisitos obtenidos para el sistema SCRICI, los cuales fueron implementados.

5.3. Conclusiones

Con base en los resultados obtenidos se pueden obtener las siguientes conclusiones:

- El sistema SCRICI es un sistema fácil de usar, y la introducción de éste al IMATE no implicó mayor complicación para los investigadores, pues fue un cambio fácil.

- Los investigadores aún no ven gran mejora en el tiempo que empleaban antes y el que emplearon en esta ocasión utilizando el sistema SCRICI. Este se atribuye a que es la primera vez que ingresan sus actividades al sistema, pero en lo sucesivo van a poder ver la mejora, al no volver a ingresar las mismas actividades.

- La opción de importar y exportar información dentro del sistema SCRICI es algo que les brinda varios beneficios a los investigadores, puesto que les ahorra tiempo y esfuerzo.

- El sistema SCRICI redujo en un 33 % el tiempo empleado para obtener los datos necesarios para realizar el informe anual de actividades. Además de que también redujo la intervención humana reduciendo los posibles errores humanos.

- El catálogo *curriculum_catalogo*, que se utilizó para indexar sólo información de los MIC, no afectó a todos los demás objetos del sitio ajenos al sistema SCRICI, pero sí mejoró el tiempo empleado en las búsquedas de los objetos propios del sistema

SCRICI, que es una tarea que se realiza muy frecuentemente dentro de las diferentes vistas del sistema SCRICI. Pero se empeoró un poco el tiempo que tardan los objetos del sistema SCRICI en reindexarse. Sin embargo, era necesario agregar los índices y metadatos dentro de algún catálogo, y el *curriculum_catalogo* fue el lugar más conveniente, puesto que sólo se reindexan los contenidos al crearlos o modificarlos, lo cuál es una tarea muy poco frecuente en comparación con las consultas realizadas.

- El sistema SCRICI cumple con todos los requisitos solicitados, por lo tanto, este sistema hace todo lo que debe de hacer.

Además, como conclusiones globales se tienen la siguientes:

- Los SMC son una solución al problema de manejo de grandes cantidades de información cambiante dentro un sitio Web.
- Plone es un SMC que permite desarrollar componentes de forma muy rápida debido al gran número de componentes disponibles. Es considerado como uno de los mejores SMC disponibles actualmente por el número de características que tiene. Además, el desarrollo sobre Plone con arquetipos, utilizando ArchGenXML y ArgoUML se reduce bastante.
- El sistema SCRICI (que fue desarrollado sobre Plone), permite ingresar información desde la Web de forma segura y mediante interfaces amigables haciendo el sistema fácil de usar. Reduce el tiempo requerido para realizar reportes sobre la información almacenada y evita realizar actividades redundantes al sólo tener que ingresar las actividades una sola vez, sin importar el número de veces que se vaya a utilizar para diferentes reportes.

  Además, mediante pláticas informales con diversos usuarios del sistema, a la mayoría le agradó y están a favor de éste. Aunque por ser la primera vez, muchos investigadores no notaron ventaja en cuanto a tiempo para entregar su primer reporte, pero están consientes de esta ventaja para los consecutivos reportes.

5.3.1. Trabajo futuro

A continuación se menciona lo que queda como trabajo futuro, y que no se pudo realizar por cuestiones que están fuera de mi alcance o porque aún no se define bien completamente:

- Actualmente el CONACYT tiene un sistema llamado *CVU* o *Currículum Vítae Único* donde se almacena información sobre las actividades académicas de alumnos, profesores e investigadores, entre otros. El CVU es muy similar al sistema SCRICI, pero tiene algunas diferencias: el CVU maneja 20 rubros, mientras que el sistema SCRICI maneja 31; además varias de las actividades coinciden y otras no en los campos que manejan.

  Todas las personas que reciben algún apoyo del CONACYT deben actualizar su CVU para poder continuar con ese apoyo, y gran parte de los investigadores del

IMATE reciben algún apoyo del CONACYT, por lo que éstos deben actualizar su currículum tanto en el CVU como en el SCRICI.

Recientemente el CONACYT ha estado desarrollando interfaces de comunicación para poder compartir la información de sus CVU con otros sistemas, mediante SOAP [3]. Actualmente hay varias instituciones que están probando estas interfaces para ponerlas en producción.

Después de haber terminado el sistema SCRICI y casi por concluir esta tesis, el IMATE solicitó al CONACYT acceso a la información de los CVU de sus investigadores mediante las interfaces desarrolladas con SOAP, pero ésto aún sigue en trámite.

Queda como trabajo futuro la interacción del sistema SCRICI con el CVU del CONACYT. Sin embargo, esta interacción sólo será inicialmente para obtener información del CVU y no para enviar información a éste, ya que el CONACYT actualmente tiene la mentalidad de ser únicamente proveedor de información y no consumidor.

- Recientemente el IMATE ha pensado en cambiar la forma en que se están manejando los proyectos y dejarlo en manos de la Secretaría Técnica, para esto se tendría que sacar de los MIC de los investigadores los proyectos reportados y ponerlos en una carpeta propia de la Secretaría Técnica, y sólo dejar que se reflejen dentro del MIC de los investigadores. Esta decisión se está pensando tomar para tener un mayor control de los proyectos que se inician y terminan por año, y de aquellos que quedan pendientes.

- Cada investigador debe reportar anualmente un plan de trabajo donde indique las tareas que va a estar realizando a lo largo del año. Si al iniciar el año, el investigador ingresa dentro del sistema SCRICI aquellas actividades que él sepa en las que va a estar trabajando, entonces se podría obtener el plan de trabajo directamente del sistema SCRICI sin que el investigador tenga redactarlo. Esto aún se está analizando dentro del IMATE.

- El sistema SCRICI se dejó preparado para ser traducido a diferentes idiomas, que es una característica que permite Plone. Se queda como trabajo futuro la traducción a los diferentes idiomas requeridos por el IMATE.

[3] SOAP es un protocolo de comunicación basado en XML, que permite la comunicación entre componentes mediante el uso de HTTP.

Anexo A

Cuestionario 1

A continuación se muestra un cuestionario que se le hizo a Ángel Manuel Carrillo Hoyo, quien fue el secretario académico del IMATE de 1993 a 2006 y realizó 13 informes anuales de actividades.

1. **¿Desde hace cuántos años se realiza el informe anual de actividades en el IMATE?**
   Hace aproximadamente 17 años.

2. **¿Hace cuántos años realiza usted el informe anual de actividades del IMATE?**
   Desde hace 13 años.

3. **¿Cuánto tiempo en promedio le tomó realizar cada informe de actividades?**
   Aproximadamente dos meses y medio desde la captura de información de cada investigador hasta la entrega del informe a Publicaciones, dedicándole casi tiempo completo. Y en una ocasión me llegué a tardar tres meses y medio.

4. **¿En qué fechas realizaba el informe?**
   Iniciaba a principios de Diciembre y tenía que terminar a más tardar por el 16 de Marzo.

5. **¿Cuál es el procedimiento que seguía para realizar el informe?**
   Utilizaba un programa parecido a Excel en el cual iba capturando la información de cada investigador de forma ordenada en 4 archivos.

   En un archivo capturaba todo lo relacionado con publicaciones de los investigadores, como son libros, artículos, capítulos de libros, etc.

   En otro archivo capturaba los datos personales de los investigadores, como son promociones de puestos, fechas de ingreso, egreso, nivel de SNI, etc.

   En otro archivo que llamaba *Relaciones* capturaba información que tiene que ver con las relaciones de los investigadores con otras personas o instituciones, como conferencias, visitas a otras instituciones, etc.

En otro archivo que llamaba *Docencia* capturaba información de cursos, cursillos, tesis, asesorías y premios recibidos.

Una vez capturada la información en los archivos realizaba un análisis de la información para evitar duplicidad o erróres en la información.

Cuando la información ya estaba correctamente organizada entonces empezaba a realizar cálculos que se requerían para escribir el informe.

6. **¿Cuánto tiempo le tomaba en promedio el proceso de capturar la información al Excel, procesar la información y tenerla lista junto con cálculos y gráficas que se incluirían en el informe anual? (Sin redacción)**
   Aproximadamente un mes y medio

7. **¿Con qué problemas se encontró?**
   Algunos investigadores no entregaban a tiempo su reporte de actividades, así que después de ya haber procesado la información, se tenía que volver a procesar cuando recibía información tarde.

   Algunos investigadores entregaban su reporte en papel, así que tenía que capturar la información a la computadora para poderla procesar.

Anexo B

Cuestionario 2

A continuación se muestra un cuestionario que se le hizo a Mónica Leñero Padierna, quien se encargó, entre sus demás tareas, de obtener del sistema SCRICI la información, cálculos de éstos y gráficas que se incluyeron en el informe anual de actividades correspondiente al año 2006, además de darle formato y edición al informe.

El siguiente cuestionario permitirá evaluar el módulo de consultas sobre las actividades de los investigadores, que permite realizar diferentes reportes, entre ellos gran parte del Informe Anual de actividades.

1. **¿Qué te pareció la interfaz?**
   La primera interfaz para consultas era bastante aceptable. Con las últimas modificaciones se volvió mucho más amigable.

2. **¿Qué tan fácil de usar es?**
   Para mí, muy fácil, aunque a lo mejor valdría la pena preguntarle a alguno de los administrativos su opinión.

3. **¿Qué tan efectivo lo consideras? (es decir, ¿qué tanto te permite realizar las tareas solicitadas?)**
   Como herramienta para generar la información requerida diría que efectivo. Para generar el informe anual fue necesario conseguir mucha más información, que no está presente en el sistema (apoyo de posgrado, proyectos de intercambio, histórico de movimientos respecto a nombramientos, PRIDE, SNI, etc.)

4. **¿Qué tan eficiente lo consideras? (es decir, ¿qué tanta facilidad y rapidez te proporciona para realizar las tareas?)**
   Creo que eficiente. Sería mucho más útil si pudieran generarse las hojas de Excel directamente, o mejor aún, las gráficas sin tener que pasar por Excel. También sería bueno poder generar los documentos directamente, exportar la información a algún procesador de texto.

5. **¿Te sientes a gusto con los resultados obtenidos?**
   Si

6. **Mejoras que le harías para reducir el tiempo o esfuerzo empleado para obtener el informe Anual**
   Lo que mencioné respecto a hojas de cálculo, gráficas y procesador de texto.

7. **Opinión personal sobre el sistema**
   Es un muy buen trabajo.

8. **¿Cuánto tiempo te tomó en promedio tener lista la información, cálculos y gráficas que se incluirían en el informe anual? (Sin redacción)**
   Casi un mes.

Anexo C

Cuestionario 3

En este anexo se muestra el cuestionario que se le realizó a los investigadores del IMATE. Se le pidió a todos los investigadores que lo contestaran, pero sólo lo hicieron 12 de ellos, de los cuales 2 no respondieron poniendo una calificación, sino sólo con comentarios abiertos, por lo cual no se pusieron dentro de la tesis. Primero se mostrará el cuestionario y después las respuestas de los investigadores.

Cuestionario aplicado

1. **¿Cómo calificarías la facilidad de uso del MIC?**
   1=Muy difícil, 2=Difícil, 3=Regular, 4=Fácil y 5=Muy Fácil
2. **¿Qué tan fácil se te hizo el cambio de como se venían reportando las actividades antes, a como se reportan actualmente por medio del MIC?**
   1=Muy difícil, 2=Difícil, 3=Regular, 4=Fácil y 5=Muy Fácil f
3. **¿Crees que la opción de importar/exportar información entre BibTex y otros formatos estándares te traiga beneficios?**
   1=Ninguno, 2=Pocos, 3=Algunos, 4=Varios y 5=Muchos
4. **¿Qué mejoras o cambios le harías al MIC?**
5. **¿Con qué problemas te has encontrado dentro del MIC?**
6. **¿El tiempo que empleabas al año antes, para hacer tu reporte de actividades para el PRIDE e informe anual mejor con este sistema?**
   1=Empeoró mucho, 2=Empeoró un poco, 3=No mejoró, 4=Mejoró un poco, 5=Mejoró mucho
7. **¿Recomendarías este sistema para que otras instituciones lo usen?**
   1=Si, 0=No
8. **¿Usas el MIC durante al año para actualizar tu producción o te esperas al final del año cuando se solicita el Informe Anual?**
   1=Si, 0=No

9. **¿Usas el MIC durante al año para consultar información?**
   1=Si, 0=No

10. **Cualquier comentario o sugerencia adicional será bienvenido**

Respuestas de los investigadores

A continuación se muestran las respuestas a las preguntas de opción múltiple mediante una tabla que muestra cada investigador que contestó el cuestionario y su respuesta:

No. Pregunta / Investigador	1	2	3	6	7	8	9
Mónica Alicia Clapp	5	5	5	5	1	1	1
Carlos Prieto de Castro	3	3	4	4	1	1	1
María Emilia Caballero	2	2	5	3	0	0	1
Ernesto Rosales González	4	5	5	2	1	0	0
José Luis Cisneros Molina	5	5	5	4	1	1	1
Sergio Rajsbaum Gorodezky	5	5	5	5	1	1	1
Luz de Teresa de Oteyza	4	5	4	3	1	0	1
Martha Gabriela Araujo Pardo	4	5	4	5	1	0	1
Juan José Montellano Ballesteros	4	4	4	4	1	0	1
Déborah Oliveros Braniff	4	-	-	-	1	0	1
Promedio	4.00	4.33	4.55	3.88	0.9	0.44	0.88

A continuación se muestran las respuestas a las preguntas abiertas:

¿Qué mejoras o cambios le harías al MIC?

- **Mónica A. Clapp** Sería bueno incluir en cada rubro un parrafito describiendo la información que hay que capturar allí. En el CVU de CONACYT viene, por ejemplo, "Artículos Publicados. Se refiere a resultados originales de investigación publicados en revistas arbitradas o indizadas."

- **Carlos Prieto** Hacerlo compatible con otros formatos, especialmente el informe y la renovación para el SNI.

- **María Emilia C.** Agregar algunos rubros: tutorías de posgrado, sinodal de exámenes de licenciatura, participación en el posgrado.

- **Ernesto Rosales González** Primero que nada felicidades. Ahora bien, los menús están muy homogéneos, si se pudiera ponerlos de distintos colores por lo menos el de la sección que estás usando o que se pudiera resaltar más estaría mejor. Además me parece que cuando entras aparece demasiada información a la visita. Si se pudiera que aparezca a parte de la barra de menú solo la información que quieres consultar. De hecho si pudiera usarse una distribución de barras de menú más estándar quizá ayudaría a que el que entre le sea más familiar moverse dentro del programa.

- **José Luis Cisneros** Por lo pronto no se me ocurre, a mi me ha funcionado muy bien.
- **Sergio Rajsbaum** Traducirlo al inglés, para que se pueda consultar en inglés. Así mismo, más opciones de imprimir, en diversos estilos, y en inglés.
- **Luz de Teresa** Creo que ninguno, me parece bastante cómodo.
- **Martha Araujo** En este momento no se me ocurre nada.

¿Con qué problemas te has encontrado dentro del MIC?

- **Ernesto Rosales González** Sólo que todavía no lo se usar del todo. Quizá si hubiera forma de hacer un manual muy básico con ejemplos clave podría ayudar en especial a los que no conocen el programa. Para mi en particular es sólo tiempo.
- **José Luis Cisneros** He tenido muy pocos y todos los he reportado y los han resuelto rápidamente.
- **Sergio Rajsbaum** Que está muy lento.
- **Martha Araujo** Cuando había que llenar el informe anual se saturaba rápido pero creo que es el único.

Cualquier comentario o sugerencia adicional será bienvenido

- **Mónica A. Clapp** Me gustaría saber si hay modo de exportar información al CVU del CONACYT, a las solicitudes de renovación de PAPIIT, etc.
- **María Emilia C.** Creo que fue difícil por ser la primera vez, creo que será más fácil en adelante. Solo deseo que ya no se cambie y que en el futuro sirva para el SNI.
- **Sergio Rajsbaum** Poner más claras las etiquetas de cada cosa que se pide.
- **Martha Araujo** Me parece fácil de llenar y accesible, me gusta que vaya apareciendo cada "ítem" que añades porque vas viendo tu actualización directamente
- **Déborah Oliveros Braniff** Es el primer año que reporto actividades. Me pareció bastante accesible, sólo tuve algunas dificultades para enviarlo por que fui la primera conejilla de indios. Pero reporté los problemitas y parece que se solucionaron después.

Anexo D

Rendimiento del curriculum_catalogo

Para hacer las pruebas de rendimiento del curriculum_catalogo se utilizó una máquina como servidor con un procesador Intel Pentium 4 a 2.66 GHz, memoria cache de 512 KB, memoria RAM de 1GB y un bus de 32 bits. Se le instaló Gentoo Linux, Zope 2.8.8 y Plone 2.1.4. Para hacer las peticiones se utilizó una máquina directamente conectada al servidor mediante un cable ethernet para evitar retrasos en la red y en el servidor sólo se tenía corriendo Plone al momento de las pruebas.

Se hicieron cuatro tipo de pruebas, primero sin el curriculum_catalogo y luego con el curriculum_catalogo. Los índices referentes a los contenidos de los MIC se agregaron en las primeras pruebas dentro del catálogo portal_catalog, y en las segundas pruebas dentro del curriculum_catalogo.

En lo que sigue de este anexo se mostrarán cada una de las pruebas y los resultados obtenidos.

Consultas de contenidos de los MIC

Esta prueba consistió en consultar del catálogo todos los objetos de los MIC y cargarlos en memoria. Esta prueba se hizo porque cuando un usuario navega a través del MIC, se realizan consultas al catálogo para mostrar los objetos del MIC, sin que el usuario se percate de ésto.

La consulta se realizó por medio de un script, repitiendo el proceso 100 veces. A continuación se muestra el código que se utilizó:

```
# La variable CATALOGO tiene el nombre del catálogo
# a consultar.
CATALOGO="portal_catalog"
#CATALOGO="curriculum_catalogo"

from Products.CMFCore.utils import getToolByName
```

```
from DateTime import DateTime

catalogo=getToolByName(context, CATALOGO)

for i in range(100):
   inicio=DateTime()
   res=catalogo(portal_type=objetosMIC)
   for i in res:
      o=i.getObject()
   fin=DateTime()
   print "Tiempo=%s" % (float(fin)-float(inicio))
```

A continuación se muestran los resultados de las dos ejecuciones del script:

- La primer prueba se hizo sin el curriculum_catalogo, es decir, con todos los índices y metadatos dentro del catálogo portal_catalog. En promedio se tardó **22.39** segundo. A continuación se muestran los tiempos que regresó el script:

  20.99 23.67 20.97 23.73 20.95 23.73 20.97 23.75 20.96 23.77 20.99 23.78 21.01 23.84 21.09 23.77 21.01 23.75 21.10 23.87 21.04 23.75 20.96 23.76 21.08 23.83 21.04 23.74 20.99 23.84 21.06 23.81 21.03 23.72 21.00 23.78 21.03 23.81 21.03 23.76 20.98 23.73 22.40 20.98 23.83 21.04 23.77 20.98 23.76 21.06 23.81 21.07 23.77 20.98 23.76 21.04 23.82 21.07 23.73 20.98 23.81 21.06 23.86 21.02 23.80 20.98 23.79 21.04 21.03 23.80 21.01 23.78 21.00 23.82 21.07 23.86 21.01 23.75 20.98 23.84 21.09 23.82 20.98 23.79 21.02 23.86 21.10 23.82 20.95 23.74 21.06 23.86 21.03 23.77 20.99 23.77 21.02 23.81 21.03 23.84

- La segunda prueba se hizo con el curriculum_catalog, es decir, poniendo los índices y metadatos relacionados con el sistema SCRICI, sólo dentro de éste catálogo. En promedio se tardó **19.95** segundos. A continuación se muestran los tiempos que regresó el script:

  18.97 20.78 18.93 20.81 18.97 20.77 18.98 20.93 18.98 20.91 19.03 20.86 19.13 21.16 19.08 20.83 19.03 20.85 19.05 20.95 19.05 20.88 19.10 20.96 19.03 20.89 19.01 20.81 19.06 20.82 19.06 20.83 19.03 20.85 19.04 20.84 19.08 20.87 19.03 20.87 19.05 20.84 19.94 19.05 20.85 19.03 20.82 19.04 20.83 19.10 20.88 19.02 20.85 19.04 20.86 19.06 20.91 19.03 20.83 19.06 20.86 19.02 20.87 19.04 20.98 19.05 20.86 19.05 20.86 19.05 20.90 19.07 20.84 19.05 20.85 19.04 20.84 19.05 20.89 19.07 20.91 19.05 20.84 19.08 20.92 19.05 20.87 19.03 20.89 19.07 20.89 19.07 20.90 19.02 20.89 19.05 20.87 19.06 20.87 19.04

Consultas de todos los objetos del sitio

Esta prueba consistió en consultar del catálogo todos los objetos del sitio y cargarlos en memoria. Esta prueba se hizo para ver cuánto afectaría el desempeño de todo el sitio

si se agregaran los índices en un catálogo o en el otro. También la consulta se hizo con un script que repitió la tarea 100 veces. A continuación se muestra el script:

```
# La variable CATALOGO tiene el nombre del catálogo
# a consultar.
CATALOGO="portal_catalog"
#CATALOGO="curriculum_catalogo"

from Products.CMFCore.utils import getToolByName
from DateTime import DateTime

catalogo=getToolByName(context, CATALOGO)

for i in range(100):
   inicio=DateTime()
   res=catalogo()
   for i in res:
      o=i.getObject()
   num=len(res)
   fin=DateTime()
   print "Tiempo=%s" % (float(fin)-float(inicio))
```

A continuación se muestran los resultados de las dos ejecuciones del script:

- La primer prueba se hizo sin el curriculum_catalogo, es decir, con todos los índices y metadatos dentro del catálogo portal_catalog. En promedio se tardó **22.36** segundo. A continuación se muestran los tiempos que regresó el script:

  21.27 23.22 21.20 23.15 21.20 23.24 23.22 21.19 23.22 21.21 23.25 21.16 23.26 21.21 23.20 21.23 23.23 23.22 21.24 23.22 21.24 23.23 21.25 23.20 21.29 23.24 23.31 21.24 23.34 21.22 23.24 21.31 23.22 21.21 23.20 21.25 23.35 23.32 21.26 23.27 21.28 23.28 21.31 23.27 22.35 21.32 23.27 21.26 23.29 23.35 21.30 23.35 21.30 23.27 21.39 23.28 21.29 23.29 23.35 21.24 23.23 21.27 23.28 21.30 23.25 21.25 23.25 21.26 23.19 23.29 21.27 23.30 21.17 23.23 21.27 23.22 21.27 23.21 21.24 23.22 23.25 21.27 23.24 21.26 23.28 21.27 23.25 21.25 23.25 23.33 21.25 23.36 21.25 23.34 21.36 23.40 21.38 23.38 21.34 23.33

- La segunda prueba se hizo con el curriculum_catalog, es decir, poniendo los índices y metadatos relacionados con el sistema SCRICI, sólo dentro de éste catálogo. En promedio se tardó **24.21** segundos. A continuación se muestran los tiempos que regresó el script:

  25.39 22.52 25.43 25.40 22.51 25.41 22.54 25.46 22.59 25.46 22.58 25.50 22.64 25.46 25.53 22.56 25.48 22.58 25.55 22.56 25.49 22.57 25.54 25.54 22.62 25.49 22.57 25.50 22.60 25.56 22.60 25.46 22.60 25.50 25.55 22.60 25.53 22.59 25.48 22.60 25.49 22.62 24.20 25.51 22.56 25.46 25.53 22.60 25.48 22.63 25.49 22.59 25.50 22.60 25.49 25.56

22.59 25.50 22.57 25.51 22.62 25.54 22.58 25.56 22.59 25.51 25.53 22.61 25.51 22.66 25.54 22.65 25.53 22.64 25.50 25.56 22.59 25.53 22.63 25.58 22.66 25.51 22.58 25.52 22.66 25.48 25.55 22.63 25.51 22.62 25.60 22.62 25.51 22.61 25.49 22.63 25.51 25.54 22.59 25.56

Reindexado de los contenidos de los MIC

Esta prueba consistió en reindexar algunos objetos de los MIC, ya que cada que se modifica un objeto, éste se reindexa. La consulta se hizo con un script que repitió la tarea 100 veces. A continuación se muestra el script:

```
# La variable CATALOGO tiene el nombre del catálogo
# a consultar.
CATALOGO="portal_catalog"
#CATALOGO="curriculum_catalogo"

from Products.CMFCore.utils import getToolByName
from DateTime import DateTime

tiposObjetos=['ProyectoReference']
catalogo=getToolByName(context, CATALOGO)

for i in range(100):
   inicio=DateTime()
   res=c(portal_type=tiposObjetos)
   for i in res:
      o=i.getObject()
      o.reindexObject()
   fin=DateTime()
   print "Tiempo=%s" % (float(fin)-float(inicio))
```

A continuación se muestran los resultados de las dos ejecuciones del script:

- La primer prueba se hizo sin el curriculum_catalogo, es decir, con todos los índices y metadatos dentro del catálogo portal_catalog. En promedio se tardó **62.85** segundo. A continuación se muestran los tiempos que regresó el script:

  60.36 60.36 60.36 60.42 60.39 60.43 60.41 61.62 60.39 60.44 60.46 60.46 75.20 76.44 60.53 60.43 60.59 60.51 61.71 60.60 60.56 60.50 60.61 60.56 60.55 60.56 77.33 74.52 60.62 61.61 60.57 60.46 60.59 60.41 60.66 60.49 60.73 60.86 60.78 60.70 78.94 75.17 62.84 60.72 60.53 60.59 60.47 60.58 60.58 60.61 60.53 61.73 60.50 60.56 60.53 78.99 72.90 60.63 60.48 60.58 61.62 60.66 60.49 60.64 60.54 60.62 60.53 60.61 60.55 80.34 72.02 60.55 61.53 60.61 60.40 60.64 60.45 60.66 60.48 60.59 60.50 60.62 60.46 81.51 72.38 60.60 60.40 60.60 60.46 60.63 60.46 60.60 60.43 61.74 60.51 60.62 60.46 82.32 70.15 60.64

- La segunda prueba se hizo con el curriculum_catalog, es decir, poniendo los índices y metadatos relacionados con el sistema SCRICI, sólo dentro de éste catálogo. En promedio se tardó **81.37** segundos. A continuación se muestran los tiempos que regresó el script:

  77.74 78.48 78.27 78.98 78.86 79.50 80.23 80.15 80.24 78.64 77.96 77.94 100.01 95.63
  78.36 78.44 78.56 78.65 78.70 78.56 78.60 78.60 78.62 78.72 78.58 78.58 102.46 95.01
  78.30 78.48 78.57 78.51 78.52 78.47 78.59 78.62 78.61 78.61 78.65 78.61 103.64 94.06
  81.36 78.34 78.30 78.41 78.48 78.52 78.56 78.59 78.55 78.48 78.47 78.51 78.51 105.05
  93.35 78.49 78.44 78.53 78.46 78.48 78.53 78.50 78.47 78.60 78.56 78.57 78.54 105.38
  92.69 78.27 78.40 78.59 78.48 78.63 78.61 78.98 78.86 78.13 78.10 78.16 78.11 106.14
  91.01 77.88 78.03 78.20 78.19 78.24 78.13 78.13 78.20 78.18 78.26 78.22 78.15 107.64
  90.00 77.90

Reindexado de contenidos que no son de los MIC

Esta prueba consistió en reindexar algunos objetos que no son de los MIC. La consulta se hizo con un script que repitió la tarea 100 veces. A continuación se muestra el script:

```
# La variable CATALOGO tiene el nombre del catálogo
# a consultar.
CATALOGO="portal_catalog"
#CATALOGO="curriculum_catalogo"

from Products.CMFCore.utils import getToolByName
from DateTime import DateTime

tiposObjetos=['Member', 'Investigador', 'Becario', 'Externo', \
          'ATPhoto', 'ATPhotoAlbum', 'Event', 'File', 'Folder']

catalogo=getToolByName(context, CATALOGO)

for i in range(100):
   inicio=DateTime()
   res=catalogo(portal_type=tiposObjetos)
   for i in res:
      o=i.getObject()
      o.reindexObject()
   fin=DateTime()
   print "Tiempo=%s" % (float(fin)-float(inicio))
```

A continuación se muestran los resultados de las dos ejecuciones del script:

- La primer prueba se hizo sin el curriculum_catalogo, es decir, con todos los índices y metadatos dentro del catálogo portal_catalog. En promedio se tardó **68.78** segundo. A continuación se muestran los tiempos que regresó el script:

66.49 67.04 66.81 67.05 66.82 67.11 66.80 67.07 66.89 67.03 66.86 85.02 67.37 67.16 66.90 67.27 66.98 67.24 67.00 67.20 67.01 67.23 67.88 82.90 68.69 67.24 67.10 67.30 67.11 67.33 67.14 67.29 67.17 68.24 67.19 81.73 70.39 67.25 67.09 67.26 67.14 67.31 68.77 67.29 67.37 67.24 67.35 67.19 80.52 72.49 67.47 67.28 67.54 67.45 67.54 67.53 67.54 67.42 67.55 67.50 78.92 74.18 67.34 67.34 67.39 67.34 67.47 67.48 67.48 67.44 67.63 67.48 76.91 76.47 67.52 67.56 67.50 67.57 67.56 67.66 67.60 67.63 67.56 67.63 67.59 85.57 67.63 67.62 67.09 67.68 67.36 67.54 67.27 67.51 67.36 67.56 67.28 84.46 68.91 67.30

- La segunda prueba se hizo con el curriculum_catalog, es decir, poniendo los índices y metadatos relacionados con el sistema SCRICI, sólo dentro de éste catálogo. En promedio se tardó **59.60** segundos. A continuación se muestran los tiempos que regresó el script:

  57.55 57.88 58.76 57.89 57.71 57.93 57.79 57.91 57.75 57.93 57.77 74.97 58.21 58.02 57.91 58.05 57.96 58.02 57.91 58.04 57.98 58.07 57.91 73.74 59.37 58.04 57.92 58.13 58.98 58.12 57.90 58.10 57.92 58.13 57.99 71.40 61.85 58.05 57.93 58.07 57.94 58.12 57.95 59.61 58.10 57.93 58.09 58.01 70.50 62.34 58.10 57.98 58.19 58.07 58.18 59.08 58.12 58.12 58.14 58.13 67.81 64.76 58.09 58.13 58.18 58.15 58.19 58.16 58.19 58.61 58.61 58.55 66.51 67.57 58.42 58.49 58.50 58.59 58.49 59.65 58.55 58.57 58.53 58.54 58.56 75.77 58.85 58.55 58.54 58.58 58.54 58.54 58.57 58.55 58.51 58.64 58.73 74.44 59.99 58.74

Anexo E

Glosario

ArgoUML Herramienta para hacer diagramas de UML en formato XMI.

ArchGenXML Herramienta para crear aplicaciones sobre Plone basadas en los arquetipos.

Arquetipos Aplicación de Zope que simplica la creación de nuevos tipos de contenidos basados en un esquema, además permite hacer referencias robustas entre objetos.

BibTex Estándar de LaTex para hacer referencias bibliográficas y darles formato.

Componente Es una pieza de código pre-elaborada, que se puede ver como una caja negra, la cual realiza una tarea y encapsula sus funciones y datos, es reutilizable, puede contener otros componentes dentro y se define cómo interactúa con otros.

CMF Es un componente de Zope que provee herramientas para crear sistemas manejadores de contenido.

curriculum_catalogo Es el catálogo del sistema SCRICI para indexar sólo la información que se encuentra dentro del MIC.

CU Caso de uso.

home Es la carpeta personal de un usuario, donde puede agregar, modificar y eliminar contenido.

IMATE Instituto de Matemáticas (UNAM).

MIC Módulo de información curricular. Es una parte del sistema SCRICI, donde se guarda la información académica de los investigadores.

MSC En una clasificación de materias de matemáticas compuesta por 63 áreas con más de $5,000$ códigos divididos en tres niveles.

PAIPA Programa de apoyo a la incorporación de personal académico de tiempo completo. Los investigadores deben entregar un reporte de sus actividades al IMATE para participar en este programa.

portal_catalog Es el catálogo que utiliza Plone para indexar y consultar absolutamente todos los contenidos del sitio.

PRIDE Programa de primas al desempeño del personal académico de tiempo completo. Los investigadores deben entregar un reporte de sus actividades al IMATE para participar en este programa.

Producto Es un componente que permite extender la funcionalidad de Zope.

SCRICI Sistema de Captura y Recolección de Información Curricular del IMATE.

SMC Sistema manejador de contenido.

SSL o Secure Socket Layer es un protocolo que proporciona comunicación segura en Internet.

VHM Es un objeto de Zope que intercepta todas las peticiones entrantes, las altera y las envía a cierta parte específica de Zope. Permite utilizar otro servidor Web diferente al ZServer.

Workflow Es la definición de la lógica de negocios de cierta parte del sistema que se encuentra en movimiento o en estado cambiane.

XMI Es un formato basado en XML que permite intercambiar metadatos entre herramientas de modelado de UML.

ZMI Interface de manejo de Zope. Es donde se lleban a cabo varias tareas de administración tanto de Zope como de Plone.

ZODB Es una base de datos orientada a objetos para python. Es la base de datos por default de Zope.

Zope Es un servidor de aplicaciones Web. Incluye todos los componentes necesarios para desarrollar un SMC..

ZPT Lenguaje de programación de páginas Web sobre Zope. Permite darle dinamismo a las páginas mediante código HTML o XML con código python incrustrado.

ZServer Es el servidor Web de Zope.

Bibliografía

[1] *http://argouml.tigris.org/*. Sitio de ArgoUML donde se puede descargar gratuitamente.

[2] *http://www.cmsmatrix.org*. Sitio enfocado a aquellas personas que quieran manejar contenido Web. Tiene foros de discusión sobre sistemas manejadores de contenido y permite realizar comparaciones entre más de 750 SMC.

[3] *http://www.eclipse.org/*. Sitio de Eclipse donde se puede descargar gratuitamente.

[4] *http://www.gnu.org/licenses*. Página del proyecto GNU que está enfocado al desarrollo de software libre. Aquí se encuentran sus diferentes licencias.

[5] *http://www.internetworldstats.com/*. Sitio que se dedica a obtener estadisticas mundiales de Internet. Página consultada el 25 de Enero de 2007.

[6] *http://www.packtpub.com/*. Packt Publishing Ltd se dedica a publicar libros enfocados a tecnología y soluciones específicas. El sitio fue consultado en Enero de 2007.

[7] *http://www.plone.org/*. Sitio de Plone.

[8] K. Ayeva. *http://plone.org/documentation/manual/archetypes-developer-manual*, 2006. Manual para el desarrollo con arquetipos.

[9] B. Boiko. Content management bible, 2nd edition, 2005.

[10] R. Cook. Is a hybrid database in your future? *SunWorld*, Febrero 1997.

[11] S. da Silva. *Archetypes Quick Reference (Reference for the Archetypes framework)*, Octubre 2004. http://plone.org/products/archetypes/documentation/old/quickref.

[12] M. den Hengst and H. Sol. The impact of information and communication technology on interorganizational coordination. In *HICSS '01: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 7*, page 7006, Washington, DC, USA, 2001. IEEE Computer Societyf.

[13] J. Fulton. Introduction to the zope object database. In *Proceedings of the 8th International Python Conference*, 2000.

[14] I. Göhlert and T. Höning. Open source content management systems for small and medium-sized enterprises. Julio 2004.

[15] A. Latteier and M. Pelletier. *The Zope Book*. Sams, Julio 2001.

[16] R. M. Lerner. The book of zope. *Linux J.*, 2002(95):16, 2002.

[17] A. Limi. Archetypes developers guide, Noviembre 23.

[18] S. McClure. Objectdatabase vs. object-relational databases. *IDC Bulletin # 14821E*, Agosto 1997.

[19] A. McKay. *The Definitive Guide to Plone*. Apress, 2004.

[20] S. McKeever. Understanding web content management systems: evolution, lifecycle and market. *Industrial Management and Data Systems*, 103(9):686–692, 2003.

[21] J. C. Meloni. *Plone Content Management Essentials*. Sams, Indianapolis, IN, USA, 2004.

[22] S. Pastore. Web content management systems: using plone open source software to build a website for research institute needs. *icdt*, 0:24, 2006.

[23] R. S. Pressman. *Ingeniería de software: Un enfoque práctico*. McGraw-Hill, 2002.

[24] R. Ritz. Programming plone - the mysite tutorial. ITB, Humboldt-University Berlin, 2005.

[25] J. Robertson. How to evaluate a content management system. *KM Column*, Enero 2002.

[26] J. Robertson. So, what is a content management system? *KM Column, Step Two Designs,*, Junio 2003.

[27] G. K. Thiruvathukal and K. Laufer. Plone and content management. *Computing in Science and Engineering*, 06(4):88–95, 2004.

[28] R. Vidgen, S. Goodwin, and S. Barnes. Web content management. *14th Bled Electronic Commerce Conference*, 2001.