UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Dirección de Investigaciones

Sistema integrado de gestión de Ciencia, Tecnología e Innovación en la Universidad de las Ciencias Informáticas

Trabajo final presentado en opción al título de
Máster en Informática Aplicada

Autora: Ing. Yunaysy Ortiz Batista

Tutor: Dr C Jorge Gulín González

Ciudad de La Habana, julio de 2011

AGRADECIMIENTOS

Agradecimiento especial a mi tutor por la confianza depositada en mí.

A mi compañero Ricardo por su apoyo en cada momento.

A mis padres por darme confianza y seguridad.

A mis amigos, por ser los mejores.

A todos los que me han ayudado en este trabajo.

DECLARACIÓN JURADA DE AUTORÍA

Declaro por este medio que yo, Yunaysy Ortiz Batista, con carné de identidad 83020122255, soy la autora principal del trabajo final de maestría “Sistema integrado de gestión de Ciencia, Tecnología e Innovación en la Universidad de las Ciencias Informáticas”, desarrollado como parte de la Maestría en Informática Aplicada y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Y para que así conste, firmo la presente declaración jurada de autoría en Ciudad de La Habana a los _____ días del mes de _____ del año 2011.

_____

Yunaysy Ortiz Batista

RESUMEN

La gestión de la información es un proceso crítico para el funcionamiento adecuado del sistema de Ciencia, Tecnología e Innovación (CTI) en un centro de educación superior. En el presente, el procedimiento que gestiona esta información en la Universidad de las Ciencias Informáticas es impreciso y dificultoso. Por otra parte, el funcionamiento aislado de los sistemas informáticos utilizados en la gestión editorial de la Serie Científica UCI (*Open Journal System*) y en la medición de la producción científica (SIndiCIT), provoca inexactitud e inconsistencia en la información manipulada. En este trabajo se presenta la concepción de un sistema informático que se propone como solución al problema, considerando los diferentes procesos que se integran dentro del sistema de CTI. Como parte importante de la solución se hizo un análisis de los sistemas mencionados a fin de integrarlos en la propuesta de solución, lo cual conllevó a hacer determinadas modificaciones en el *Open Journal System* con ese objetivo. La propuesta resulta novedosa en el contexto nacional e iberoamericano, pues se desarrolla una aplicación que integrada en un sistema único la información asociada a los procesos claves dentro del sistema de CTI de la universidad. Esta constituye una herramienta de gran utilidad, ya que posibilita el acceso efectivo a la información, lo cual permite su mayor control.

Palabras clave: integración, sistema de gestión de información, sistema de ciencia, tecnología e innovación.

ABSTRACT

Information management is a critical process for the appropriate functioning of R&D system in universities and other scientific institutions. At present, the procedure to manage this information in the University of Informatics Sciences (UIS) is slow, awkward, and sensitive to introducing errors. Otherwise, isolated operation of informatics applications, like Scientific Series UIS editorial management system and the scientometrics indicators system for measure scientific production, causes inaccuracy and inconsistency in manipulated data. This paper embodies the general conception of a new computer system, which is proposed as a solution to the above mentioned problem, considering the different processes involved in the R&D system. As important part of the solution, an analysis about above mentioned systems was made, in order to integrate in the solution proposal, which led to make adjustments to the Open Journal System for adapt it to integration needs. This proposal is novel because it is a computer system that integrates, in a single platform, the complete information related to the scientific and technological activity in our university. By implementing this system, all the information related to the scientific activity that takes place in the university, will be efficiently managed. Additionally, the application potential and its flexibility will facilitate extending this methodology to other universities and research institutions not only in Cuba but also overseas.

Keywords: information management system, integration, science, technology and innovation system.

ÍNDICE

INTRODUCCIÓN

La investigación científica es un aspecto clave en el desarrollo socioeconómico de las naciones y por ende está incluida dentro de los procesos sustantivos del sistema universitario. La Universidad de las Ciencias Informáticas (UCI) como centro de educación superior, promueve la vinculación docencia-investigación-producción y el empleo eficiente del potencial científico de profesores y estudiantes, con el objetivo de elevar el nivel y la calidad de la docencia universitaria y contribuir directamente a mejorar las condiciones económicas y sociales del país [1].

La Dirección de Investigaciones es la estructura funcional encargada de orientar metodológicamente la actividad de Ciencia, Tecnología e Innovación (CTI) en la universidad y por ello desempeña un rol importante. Un documento fundamental en este proceso es la Política Científica, que establece las directrices de esta actividad en un período de cinco años.

Todo sistema de CTI debe partir de una definición clara y un control adecuado de las líneas de investigación, así como de los proyectos I+D+i asociados a las mismas, pues estas constituyen las direcciones de trabajo por las cuales los grupos de investigación deben orientar su labor científica y con ello obtener resultados visibles que aporten al desarrollo socioeconómico del país.

Las publicaciones científicas constituyen el indicador de mayor impacto reconocido internacionalmente, considerando que son el principal medio usado por la comunidad científica para dar a conocer sus resultados.

Como parte de este proceso se realiza la gestión editorial de las publicaciones en la Serie Científica UCI[1] (SC-UCI) y en la Revista Cubana de Ciencias Informáticas[2] (RCCI). Se hace además un seguimiento de las publicaciones de estudiantes y profesores en diferentes fuentes de publicación (revistas, memorias de eventos y repositorios institucionales) externas a la universidad, así como un seguimiento de las publicaciones seriadas de mayor impacto.

Para la gestión editorial de las publicaciones en la SC-UCI se utiliza actualmente un sistema de gestión de contenidos especializado en publicaciones científicas. Este

[1] Publicación seriada en la cual la comunidad universitaria puede publicar sus resultados asociados a las líneas de investigación definidas en la Política Científica de la universidad.

[2] Publicación seriada con la misión de socializar resultados vinculados con la ciencia de la computación y sus aplicaciones.

sistema permite el registro y arbitraje de artículos, así como su publicación para la comunidad universitaria.

Por otro lado, a través de los eventos científicos son socializados los resultados científicos de alto nivel, lo cual también proporciona visibilidad a la entidad. Este proceso incluye los eventos organizados por la universidad como: Jornada Científico-Estudiantil, Fórum de Ciencia y Técnica o la Conferencia Científica UCIENCIA, y los externos en los cuales participa la comunidad universitaria.

Otro de los procesos es la gestión de la información relacionada con los premios de CTI. Estos son reconocimientos que entregan determinadas instituciones a un resultado o la labor científica de una persona en un período de tiempo determinado, tales como los premios CITMA, TWAS, UNESCO y otros.

Estos resultados generados de la actividad científica, son procesados en los balances de CTI, en los cuales se mide el desarrollo científico que ha alcanzado la universidad en el período de un año y donde se reconoce la labor realizada por profesores, estudiantes y colectivos de trabajo, que resultan destacados en esta esfera. El balance es una de las actividades en la cual se manipula mayor cantidad de información.

Para el análisis del desarrollo científico, se utiliza un sistema informático de indicadores cienciométricos (SIndiCIT). Este sistema permite un conjunto de salidas, de acuerdo al sistema de indicadores establecidos por el CITMA-MES y de acuerdo a un sistema de indicadores adaptado a las características de la UCI, que posibilita comparar la universidad con otras instituciones en el país, así como establecer un *ranking* entre las diferentes áreas del centro, respectivamente.

Una de las dificultades existentes es que pese a que la información asociada a estos procesos se recopila digitalmente, los métodos para el procesamiento y la gestión de la misma son los tradicionales y no los más efectivos. Esto dificulta el análisis estadístico o cuantitativo con posterioridad, sobre todo en aquellos procesos en los cuales se gestiona gran volumen de información, como la organización de los eventos científicos que se desarrollan en la universidad y el procesamiento de la información necesaria para realizar los balances de CTI.

Otra dificultad es la utilización de diferentes sistemas informáticos en la gestión de algunos de los procesos de CTI (SIndiCIT para medir la producción científica y el sistema de gestión editorial de las publicaciones en la SC-UCI), lo cual –al utilizarse de manera aislada- provoca inconsistencia en la información relacionada con dichos procesos.

Dada esta problemática, se identificó el siguiente **problema de investigación**: ¿cómo facilitar y gestionar de manera consistente la información asociada a la actividad de CTI que se desarrolla en la UCI?

Se definió como **objeto de estudio** los procesos claves del sistema de CTI en la UCI y centra su investigación en la gestión, de forma integrada, de la información asociada a dichos procesos, constituyendo este el **campo de acción**.

Por tanto, se propone como **objetivo general**: integrar en un sistema único los procesos claves de CTI que se desarrollan en la UCI, para evitar inconsistencias en la información que gestionan.

Como **hipótesis de trabajo** se propone: el desarrollo de una aplicación que integre en un sistema único los procesos claves de CTI que se desarrollan en la UCI, facilitará y permitirá que se gestione de manera consistente la información asociada a los mismos.

Para dar cumplimiento al objetivo general se definieron los siguientes **objetivos específicos**:

- Analizar el estado del arte en relación a sistemas de información de CTI.
- Valorar la integración con los sistemas informáticos utilizados actualmente en la gestión de algunos procesos de CTI en la UCI (*OJS* y SIndiCIT).
- Realizar la propuesta de integración de los procesos claves de CTI en un sistema único de información.

Se utilizó el método teórico **analítico-sintético** para el análisis de los diferentes procesos del sistema de CTI y su posterior integración con el fin de identificar sus relaciones; y el **hipotético-deductivo** que permitió la formulación de la hipótesis.

La **observación** fue uno de los métodos empíricos utilizados, que es el registro visual del comportamiento de los procesos que fueron investigados. Este es uno de los métodos más utilizados, pues permite obtener información directa e inmediata sobre el fenómeno u objeto que se investiga. La **entrevista** fue otro de los métodos empíricos empleados en la obtención de información sobre los procesos que se ejecutan dentro del sistema de CTI.

La propuesta resulta novedosa en el contexto nacional e iberoamericano, pues se desarrolla una aplicación que integra en un sistema único los procesos claves de CTI

que se desarrollan en la universidad. Su concepción, así como las tecnologías utilizadas en su desarrollo permiten interoperabilidad para integraciones futuras.

El presente trabajo se estructura con una introducción, tres capítulos, conclusiones, recomendaciones y bibliografías que se referencian en el cuerpo del trabajo.

En el capítulo 1 se realiza un análisis del marco teórico, haciendo énfasis en el estado del arte referente a sistemas de información utilizados para la gestión de la actividad de CTI.

En el capítulo 2 se describen las tecnologías y herramientas utilizadas en el desarrollo de la propuesta de solución.

Por último, en el capítulo 3 se hace un análisis de los sistemas informáticos que serán integrados en la solución. Se describe la propuesta y se valida a partir de un conjunto de pruebas de sistema.

Cada capítulo cuenta con una introducción y conclusiones parciales.

CAPÍTULO 1. FUNDAMENACIÓN TEÓRICA

1. Introducción

A partir de la necesidad de

En este capítulo se realiza una valoración del estado del arte en relación a sistemas de información para la gestión de la actividad de CTI a nivel internacional y en Cuba. Para ello, se reflejan resultados de investigaciones y proyectos desarrollados (en el contexto iberoamericano, nacional y en la UCI), dentro del área de conocimiento en el que se desarrolla la investigación.

2. Estado del arte en relación con sistemas de gestión de Ciencia, Tecnología e Innovación

Para promocionar, divulgar y gestionar información científico–tecnológica, lograr mayor visibilidad de producciones científicas de universidades y centros de investigación, apoyar a la formulación de políticas y a la toma de decisiones cada vez más eficiente y efectiva en relación con la ciencia y la tecnología, la utilización de sistemas de información ha sido indispensable.

En este sentido, no han sido pocas las instituciones -a nivel nacional e internacional-, que han realizado grandes esfuerzos con el fin de informatizar sus procesos vinculados con la ciencia, la tecnología y la innovación.

2.1. Experiencias en el uso de sistemas de información científica en el contexto iberoamericano

2.1.1. Red Internacional de Fuentes de Información y Conocimiento para la Gestión de la Ciencia, Tecnología e Innovación (ScienTI)

ScienTI es una red internacional de información que tiene como objetivo brindar un espacio para compartir las experiencias nacionales e internacionales relativas a las políticas de CTI y el rol de los sistemas de información en apoyo a la gestión de la actividad científica y tecnológica. [2]

ScienTI se propone:

- Promover acciones en red colaborativa, buscando el intercambio de contenido, metodología y sistemas de gestión en CTI, respectando la soberanía, las políticas y los intereses nacionales de los países participantes.

- Propiciar y fortalecer el intercambio de información y conocimiento en gestión de CTI.

Esta Red surgió a partir de la cooperación entre el Consejo Nacional de Investigación Científica y Tecnológica (CNPq) de Brasil y la Organización Panamericana de la Salud (OPS), firmada en 2001 con el objetivo de transferir la tecnología y metodología de la Plataforma Lattes[3] a los Países de América Latina, en particular, para el desarrollo de las fuentes de Información de la Biblioteca Virtual en Salud. Al extenderse el marco de acción a todas las áreas de conocimiento, propició la formación de la Red ScienTI. [3]

Actualmente a esta Red pertenecen más de doce países de Latinoamérica y el Caribe. Entre ellos: Argentina, Cuba, Brasil, Ecuador, Venezuela, Colombia, Chile, México, Panamá y otros.

Esta Red cuenta con un conjunto de sistemas de información, bases de datos (de *Currículum Vitae* (CV), grupos de investigación, proyectos, instituciones, etc.) y portales que aseguran el acceso a sus fuentes de información, con el fin de lograr el intercambio, complementariedad y socialización de experiencias, productos y servicios de CTI de cada país miembro.

Tanto la captura y el almacenamiento de la información nacional de CTI, como su divulgación por la Web, son flexibles a las decisiones de cada participante de la red ScienTI.

En el caso de cada país para adoptar los instrumentos apropiados, su participación en el directorio internacional es condicional a la compatibilización de los archivos XML (*Extensible Markup Language*) de la información y a la responsabilidad de la publicación de esta información, en forma periódica, en el directorio correspondiente de ScienTI.

El proceso de desarrollo de la Red incluyó, en su primera fase, el establecimiento de convenios bilaterales entre el CNPq y los Organismos Nacionales de Ciencia y Tecnología, firmados en el ámbito de la Red ScienTI. Además de la transferencia de tecnología y metodología Lattes, el CNPq promovió acciones de desarrollo, instalación e implantación de la primera generación de sistemas de la Red ScienTI con base en la Plataforma Lattes.[3]

En 2002 después del desarrollo de una versión en español de Curriculum Lattes, se firman acuerdos binacionales con CNPq para la transferencia de la Plataforma Lattes

[3] Sistema informático para la gestión de información científica.

a seis países. Así, el Curriculum Lattes fue desplegado en países como Colombia, Ecuador, Chile, Perú, Argentina y Portugal y otros que están en el proceso de despliegue.

Con esto, actualmente la mayoría de los países miembros de la Red cuentan con algún tipo de sistema de información para la gestión, fundamentalmente, de *currículum vitae* de sus comunidades científicas.

Con el desarrollo de la Red y la necesidad de asegurar la actualización local de la información de estos sistemas, se estableció una topología descentralizada, basada en la interoperabilidad entre las diversas fuentes de información de CTI disponibles [3].

Estas fuentes de información (currículos, grupos de investigación, instituciones y proyectos) son descritas, representadas y estandarizadas usando el lenguaje XML. El intercambio de información entre los países miembros de la Red es realizado por servicios web que establecen un protocolo de comunicación entre los sistemas y fuentes de información de la red [4].

Esta topología de la Red promueve y facilita el desarrollo de proyectos de información y conocimiento específicos, que comparten la misma red de fuentes de información.

2.1.1.1. Plataforma Lattes. Brasil

La Plataforma Lattes es una base de datos de currículos de la experiencia del CNPq en la integración de bases de datos de los programas e instituciones de ciencia y tecnología en un sistema único de información, el cual puede ser utilizado tanto para apoyar la actividad de gestión como para apoyar a la formulación de políticas.[5]

A partir de Curriculum Lattes, el CNPq desarrolla un formato estándar para la recopilación de información curricular adoptada hoy por la mayoría de las universidades e institutos de investigación en el país. [5]

Lattes es una base de datos pública, tanto en lo que respecta a la entrada como a la recuperación de la información a través de Internet.

A pesar de la información pública disponible, instituciones educativas, de investigación e innovación en el país, han solicitado a CNPq el acceso a los datos de sus profesores, investigadores y estudiantes, con el objetivo de realizar estudios a través de la aplicación de minería de datos, apoyar la aplicación de políticas de gestión, generar indicadores internos de la producción científica y tecnológica, e integrar los datos de Lattes a sus sistemas de información.

En este sentido, la Agencia ofrece a las instituciones tres modelos de acuerdos que facilitan el acceso y la extracción de información de la Plataforma Lattes.

Uno de ellos consiste en disponer de todos los datos en detalles de la Plataforma y de los archivos de currículos actualizados diariamente, para ser replicados en la base espejo de la institución que realiza el convenio.

Puesto que este modelo implica la necesidad de apoyo y el control frecuente por el equipo técnico del CNPq, el número de acuerdos en este modelo está limitado a veinte instituciones, con un costo anual de R$ 36,000.00 para el mantenimiento de la infraestructura de apoyo a la institución del convenio.

El segundo modelo está dirigido a las instituciones encargadas de la ejecución de proyectos de interés del gobierno, y consiste en el suministro de una copia mensual de todos los currículos de la base de Lattes, salvo los datos personales contenidos en ella. El acuerdo para obtener una copia de la base Lattes tiene un costo anual de R$ 12.000,00.

El tercer modelo de acuerdo es extracción en línea, el cual está disponible para todas las instituciones educativas y de investigación e innovación en el país que deseen obtener datos de sus profesores, investigadores y estudiantes registrados en la Plataforma Lattes. Consiste en la disponibilidad de servicios web que posibilitan la recuperación en línea de los currículos de la institución convenida. Este modelo tiene un costo anual de R$ 6,000.00.

Los datos obtenidos a través de los tres tipos de acuerdo, estarán siempre en formato estándar XML definido por la comunidad, responsable de la normalización de la información existente en la Plataforma Lattes. [6]

2.1.1.2. Sistema de Información de Ciencia y Tecnología Argentino (SICyTAR)

Según su sitio oficial, el SICyTAR es un sistema de información para la gestión y administración de recursos básicos para el sector científico tecnológico.

El mismo tiene como objetivo organizar y mantener actualizados registros unificados nacionales de científicos y tecnólogos, de grupos, de proyectos de investigación en los que estos intervengan y de los órganos que integran el Sistema Nacional de Ciencia, Tecnología e Innovación, así como los nomencladores básicos utilizados a los fines de categorizar y clasificar dicha información.

El SICyTAR está constituido por los siguientes sistemas de información:

- CvLAC *(Currículum Vítae de Latinoamérica y el Caribe)*: reúne la información curricular de los usuarios del Sistema Científico y Tecnológico.

- GrupLAC *(Grupo Latinoamérica y el Caribe)*: es un Directorio de Grupos de investigación.

- Instituciones: contiene información de instituciones Científicas y Tecnológicas.

Con el subsistema CvLAC cada investigador podrá modificar total o parcialmente sus datos sin más intervención de la Secretaría de Ciencia, Tecnología e Innovación Productiva, ni de otro organismo, otorgándose de esta forma carácter de declaración jurada a la información que incluirán los currículos. El sistema le solicitará al usuario que necesite importar o migrar su currículo, una contraseña, garantizándose la exclusividad de las actualizaciones únicamente al titular de cada currículo.

El desarrollo del Proyecto SICyTAR se basa tecnológicamente en la Plataforma Lattes. [7]

2.1.1.3. Instituto Colombiano para el Desarrollo de la Ciencia y la Tecnología (COLCIENCIAS). CvLAC

La plataforma COLCIENCIAS cuenta con varios subsistemas para la gestión de *currículum vitae*, información de grupos de investigación e instituciones relacionadas con la actividad de CTI. Entre estos subsistemas se encuentran: CvLAC, GrupLAC e InsituLAC.

CvLAC es una herramienta que permite a los usuarios realizar el registro o actualización en línea (vía Web) de su hoja de vida (*currículum vitae*) [8].

CvLAC adopta la tecnología y metodología, inicialmente desarrollada por el CNPq de Brasil y con la contribución de los países participantes en el proyecto, la desarrolla, la adapta y la hace viable para su progresiva implantación y operación, a escala regional e internacional [9].

GrupLAC es un software originalmente desarrollado por el Grupo Stela de la Universidad Federal de Santa Catarina, Brasil, cuyo objetivo es mantener un directorio de los grupos de investigación, instituciones e investigadores que participan activamente en el desarrollo de nuevas estrategias en el ámbito de la ciencia, la tecnología y la innovación.

Con motivo de la Convocatoria de Grupos Colombianos de Investigación de 2002 se desarrolló la versión en línea para Colciencias en 2003, la cual dispone de varias

mejoras relacionadas con: producción científica del grupo basada en las hojas de vida de sus integrantes, pertenencia a más de una institución, relaciones con empresa, reportes de revisión de información, entre otras [10].

InstituLAC es un servicio de Colciencias, que contiene la información de la conformación y detalles de instituciones relacionadas en la Red ScienTI.

Esta aplicación está diseñada para que el funcionario responsable de la institución defina la estructura de la organización que debe estar visible en la plataforma, defina los roles de otros actores frente a la plataforma, por ejemplo: los directores de programas que interactúan con el sistema de doctorados; además, asociar los currículos de las personas vinculadas; avale a los grupos de investigación que pertenecen a ella, y las horas que dedican los integrantes del grupo al mismo; publique las solicitudes de personas altamente calificadas y registre los proyectos de investigación en el Banco Nacional de Proyectos de la plataforma.

En resumen, permitirá al actor institucional definir y mantener la información que oriente su presencia institucional en la plataforma.[11]

2.1.1.4. Secretaría Nacional de Ciencia y Tecnología (SENACYT). Ecuador. CvLAC

SENACYT también utiliza CvLAC para la gestión de los currículos vitae de todas las personas que se relacionan con la actividad de CTI en Ecuador. El mismo adopta la tecnología-metodología inicialmente desarrollada por el CNPq de Brasil y la desarrolla, la adapta y la viabiliza para su progresiva implantación y operación a escala regional e internacional.

Este sistema permite el registro y la actualización de la información de la hoja de vida de las personas vinculadas al SINACYT. En CvLAC, las personas incluyen toda la información de su hoja de vida: datos personales, formación, experiencia académica y profesional; su producción de tipo intelectual: bibliográfica, técnica, tecnológica o artística, entre otros [12].

2.1.2. Sistema Integrado de Información sobre Investigación Científica y Tecnológica. (SIICYT). México

El SIICYT es un sistema de información que tiene por objeto integrar, homogeneizar y estandarizar toda la información disponible sobre investigación científica y tecnológica, técnicas y servicios que ofertan las instituciones educativas, centros, organismos, empresas y personas físicas del sector público, privado y social. A fin de conocer las capacidades del sistema científico y tecnológico nacional, y en consecuencia, planear

el mejoramiento del mismo y aprovechar sus potencialidades para la solución de los problemas nacionales, de las empresas y del propio gobierno.

El SIICYT es un instrumento que apoya la divulgación de la ciencia y la tecnología y se logra una conciencia creciente en la sociedad sobre la importancia de la investigación y el conocimiento en la elevación del nivel de vida. Este sistema gestiona información de *currículum vítae*, proyectos de investigación e instituciones y empresas de Ciencia y Tecnología. [13]

2.1.3. Universidad de Barcelona, España. (Sistema GREC)

GREC es una aplicación de gestión de investigación desarrollada por la Universidad de Barcelona, actualmente utilizada por diversas instituciones y organismos de investigación en España. GREC incluye un conjunto de bases de datos como por ejemplo: Currículo Vítae (CV), grupos de investigación, proyectos y publicaciones.

La aplicación Currículo permite la actualización y gestión del propio currículo, así como la obtención de éste en diversos formatos. Mediante un acceso seguro e individualizado, se puede introducir el CV y actualizarlo desde cualquier punto de acceso a Internet. La aplicación permite a cada usuario visualizar su CV, modificarlo, guardarlo en disco, tramitarlo o imprimirlo en el formato deseado y con la selección de datos que sea necesario. El servidor seguro y el acceso mediante un código y una palabra clave (contraseña) garantizan la seguridad y la confidencialidad de los datos introducidos.

La aplicación Currículo para grupos de investigación permite la actualización y gestión del currículo de dichos grupos, a partir de los currículos individuales de los investigadores que forman parte del grupo.[14]

2.1.4. Sistema de Información Científica de Andalucía (SICA). España

El Sistema de Información Científica de Andalucía, SICA, puede ser definido como un conjunto de personas, procedimientos y equipos diseñados, construidos, operados y mantenidos para recoger, registrar, procesar, almacenar, recuperar y visualizar información relacionada con las actividades y resultados producidos por los investigadores en sus centros de desarrollo o en colaboración con otras instituciones nacionales o internacionales.

Es un sistema de ámbito regional que agrupa la producción científica y que, a medida que ésta se genera, es validada en poco tiempo, facilitando un análisis fiable y evolutivo de las políticas científicas.

SICA ha sido el primer sistema de información a nivel nacional en España, en implementar un formato de salida del currículum de los investigadores, bajo la norma CVN2 -Currículum Vitae Normalizado- y cuyo objetivo es permitir el intercambio de información entre los diferentes sistemas de información de las entidades del Sistema Español de Ciencia y Tecnología-Empresa (SECTE). [15]

2.2. Experiencias en el uso de sistemas de información científica en Cuba

Cuba no se queda al margen del empleo de las nuevas Tecnologías de la Información y las Comunicaciones para gestionar su actividad científico–tecnológica y de innovación. La mayoría de los centros del país vinculados a la actividad de ciencia, tecnología e innovación, se han propuesto desarrollar sistemas de información con el propósito de hacer visible sus resultados científicos, publicar informaciones de CTI y gestionar todo el quehacer científico en las diferentes instituciones.

En este sentido, se identificó un conjunto de sistemas que en su gran mayoría consisten en portales web informativos, sin ningún componente transaccional que permita un análisis estadístico de la información registrada.

En este caso se pueden encontrar diferentes sitios web de instituciones que se vinculan con la actividad de CTI, como:

- "Redcien". Red cubana de la ciencia, disponible en: http://www.redciencia.cu/
- Academia de Ciencias de Cuba, disponible en: http://www.academiaciencias.cu/
- Observatorio Cubano de Ciencia y Técnica, disponible en: http://www.occyt.cu/
- Cátedra Cubana de CTS+i, disponible en: http://www.vriep.uh.cu/ctsuh/

Y otros portales web de centros de altos estudios del país, donde se promociona la actividad científico–tecnológica de sus comunidades, como:

- Vicerrectoría de Investigaciones de la Universidad de La Habana, disponible en: http://www.uh.cu/infogral/areasuh/vri/
- Centro Universitario "José Martí" de Sancti Spíritus, disponible en: http://www.suss.co.cu/apache2-default/
- Universidad de Cienfuegos "Carlos Rafael Rodríguez", disponible en: http://www.ucf.edu.cu/

- Universidad de Camagüey, disponible en: http://www.reduc.edu.cu/
- Centro Universitario “Vladimir Ilich Lenin” de las Tunas, disponible en: http://www.ult.edu.cu/
- Universidad de Granma, disponible en: http://www.udg.co.cu/

Se identificaron otros sistemas, no solamente informativos, que han sido desarrollados para el registro, actualización y visualización de información científica, posibilitando el análisis de la misma para el apoyo a la toma de decisiones en relación con la actividad de CTI de las diferentes instituciones.

2.2.1. Sistema de Información para la Gestión de Programas y Proyectos de Ciencia e Innovación Tecnológica (SIPROCIT)

SIPROCIT constituye un sistema de apoyo a la planificación, control, evaluación y proyección de los Programas y Proyectos de Ciencia e Innovación, acorde a las prioridades establecidas para un período determinado en Cuba y que contribuye a incrementar el nivel de integración del Sistema de Proyectos y Programas (SPP).

De manera general, con esta aplicación se puede acceder a los listados actualizados de programas y proyectos nacionales, territoriales y ramales, de proyectos no asociados a programas y los internacionales, así como a los datos consolidados, series cronológicas y gráficos estadísticos de programas y proyectos en ejecución y terminados con información pública sobre la planificación, gerencia y participación de los territorios, organismos y entidades.

El sistema fue desarrollado sobre una arquitectura cliente-servidor y soportado por una plataforma mixta con prevalencia de tecnologías de código abierto. En concordancia, se seleccionó el servidor MySQL como gestor de la base de datos y los lenguajes PHP y Java Script para la interfaz Web. [16]

Los programas y proyectos que se gestionan con SIPROCIT son los gerenciados por el CITMA y que están a cargo de diferentes instituciones como GEPROP, Delegación del CITMA de la Ciudad de La Habana, ICIMAF, CUBAENERGÍA, entre otras, aunque también el MINAZ y el MINAGRI han introducido alguna información pero en menor cuantía que la del CITMA.

La información se introduce de manera descentralizada por cada uno de los secretarios de los programas y es accedido por otros usuarios para recuperar información.

El sistema es administrado por GEPROP, quien tiene como uno de sus objetos sociales la gestión de programas y proyectos nacionales de ciencia e innovación. [17]

2.2.2. Uso de herramientas informáticas para la gestión de proyectos de investigación

La Dirección de Ciencia y Técnica del Ministerio de Educación Superior en colaboración con el Instituto Superior Politécnico José Antonio Echeverría (CUJAE) llevaron a término un proyecto que proponía una metodología para la planificación y control de proyectos de investigación, así como el uso de herramientas informáticas (*Microsoft Project* 2002, integrado con el uso de *Microsoft Office Word, Outlook, Excel* y *Access*) para dar soluciones a determinados problemas profesionales.

El uso de las Tecnologías de la Información y las Comunicaciones (TIC), con un enfoque sistémico, les permitía establecer un sistema de Dirección por Proyectos capaz de garantizar su desarrollo desde la etapa de Concepción, Planificación y Programación hasta el Control de la Ejecución, haciendo uso de técnicas de diagnóstico y pronósticos con la utilización de los sistemas informáticos profesionales, apoyados de las redes informáticas.

El sistema era alojado en un servidor donde cada director de proyecto podía inicialmente depositar su documentación y posteriormente actualizarla desde el lugar de origen a través del correo electrónico, disquetes o directamente en el servidor, de acuerdo con los cortes programados para informar del cumplimiento de los objetivos asociados a los criterios de medida.

En la medida en que se desarrollaban los proyectos en formato electrónico se hacía imprescindible la elaboración de la base de datos de proyectos en ejecución, con el objetivo de facilitar la información primaria a las partes interesadas. Desde la base de datos era posible acceder a los proyectos a través de la herramienta *Project* y obtener mayor información sobre el avance de los mismos. En los cortes y en el cierre del proyecto también era necesario evaluar la producción científico técnica reflejada en el contenido del proyecto en formato electrónico y que implicaba: publicaciones nacionales e internacionales, publicaciones en revistas de alto impacto, monografías, libros, solicitudes y aprobaciones de patentes, participación en eventos científicos nacionales e internacionales, premios en eventos, certificados de calidad, resultados de los doctorados, maestrías y diplomados; resultados del trabajo científico estudiantil, diplomas estudiantiles y otros.

Los proyectos terminados con todos sus resultados y producción científica también eran almacenados convenientemente en base de datos de proyectos concluidos para

ser consultados oportunamente y debían tener referenciada toda la producción científica alcanzada durante su período de ejecución.

En los sistemas nacionales, donde se generaba información en los diversos lugares del país era necesario crear sistemas de bases de datos territoriales y enviarlas a una base de datos central donde podían ser publicados los resultados para un uso compartido de los diferentes proyectos.

El proyecto era llevado a un formato electrónico para facilitar su publicación en la red y el acceso de las partes interesadas, tanto a la concepción como a su control de ejecución y establecer nexos de intercambio de información a distancia en beneficio del cumplimiento de los objetivos de los proyectos y programas. [18]

2.3. Experiencias en el uso de sistemas de información científica en la UCI

2.3.1. Sistema de indicadores cienciométricos (SIndiCIT)

La existencia de un sistema de indicadores que permita evaluar la producción científica de los profesores, investigadores y estudiantes de la universidad, que potencie los resultados científicos y de innovación en las ramas de las Ciencias Informáticas y de la Computación, que premie el trabajo en equipo y que se adapte a las características de la universidad, han sido algunos de los principales objetivos que se ha trazado la Dirección de Investigaciones de la UCI. [19]

Con tal propósito, esta dirección se dio a la tarea de adaptar el Sistema de Indicadores de CTI vigente en las instituciones y universidades del Ministerio de Educación Superior (MES) y el Ministerio de Ciencia Tecnología y Medio Ambiente de Cuba (CITMA) a las condiciones existentes en la UCI; que consisten fundamentalmente en que la formación curricular y el núcleo fundamental de las investigaciones científicas que se desarrollan se centran en las ramas mencionadas. Además del trabajo académico y de investigación y desarrollo, la UCI se propone participar de manera decisiva en la producción de software enfocados en la informatización de la sociedad y la exportación de productos informáticos con alto valor agregado. [20]

Todo esto se materializó con el desarrollo de un sistema informático que implementó dichas adecuaciones (SIndiCIT), capaz de ponderar de manera diferenciada aquellas investigaciones científicas de ciclo completo (Investigación + Desarrollo + Producción + Comercialización) con relación a aquellas investigaciones puramente académicas y medir, en sentido general, la actividad de CTI en la universidad.

Este sistema actualmente depende de una serie de entradas que consiste en información cuantitativa de los diferentes indicadores cienciométricos que son introducidos por los asesores de investigación en las diferentes áreas. A partir de estas entradas y a través de un mecanismo de ponderación, el sistema permite un conjunto de salidas que posibilitan la evaluación de la producción científica de la universidad.

Este sistema se ha estado utilizando en los balances de CTI de la universidad desde el año 2006 con resultados satisfactorios.

2.3.2. Sistema para la gestión editorial de las publicaciones de la Serie Científica UCI. Open Journal System (OJS)

La SC-UCI es una publicación seriada interna de la universidad donde estudiantes y profesores publican los resultados de sus investigaciones científicas.

Esta publicación seriada está montada sobre un sistema de administración y publicación de revistas (*Open Journal System*, *OJS* por sus siglas en inglés), que se utiliza en la universidad con este propósito.

El *OJS* está diseñado para reducir el tiempo dedicado al manejo exhaustivo de las tareas que involucra la edición de una publicación seriada. Este sistema permite un manejo eficiente y unificado del proceso editorial, con lo que se busca acelerar el acceso a las investigaciones producidas por las universidades y centros de investigación productores del conocimiento. [21]

Este sistema posee un conjunto de características que se relacionan a continuación:

- Permite al editor configurar los requerimientos, secciones, procesos de revisión, entre otros.
- Los envíos de artículos son controlados en línea y así mismo se administra el proceso editorial.
- Contiene un módulo para ofrecer opcionalmente acceso libre a la información publicada.
- Maneja notificaciones de correo electrónico del avance del proceso editorial y permite a los usuarios registrados comentar los artículos publicados.
- El sistema asiste en cada fase del proceso de publicación, desde el envío hasta la publicación en línea.
- Es un software de código abierto y de acceso gratuito para publicaciones de todo el mundo.

Por todas estas características antes mencionadas y sobre todo por la reducción del tiempo dedicado a las tareas que involucra la edición de una publicación seriada, la Dirección de Investigaciones optó por el uso del *OJS* para la gestión de las publicaciones en la SC-UCI.

Este se prevé que sea utilizado también en la gestión de las publicaciones de la Revista Cubana de Ciencias Informáticas (RCCI).

3. Conclusiones del capítulo

De acuerdo a la bibliografía consultada, se puede apreciar en sentido general el impacto que han alcanzado los sistemas de información, ya que automatizan procesos operativos y suministran una plataforma de información importante para promover y potenciar la actividad científica.

En el contexto iberoamericano existe un conjunto de sistemas de información para la gestión de la actividad de CTI en varios países. La mayoría de estos sistemas pertenecen a la red internacional ScienTI y muchos de ellos han implantado y adaptado a las condiciones reales de cada país, el sistema CvLAC para la gestión del *currículum vítae* de todo el personal que se vincula con la investigación científica.

La concepción de CvLAC no se corresponde con las necesidades existentes en el sistema de CTI de la universidad, pues con este sistema se controla la producción científica a partir del *currículum vítae* de los usuarios; lo cual no permite el control a partir de la recopilación de información validada en cada uno de los procesos que se desarrollan.

Existe otro conjunto de sistemas que se han desarrollado para dar solución a determinadas instituciones, por lo cual no se encuentran disponible, de forma libre, para su utilización y –en el caso necesario- adaptación por cualquier otro centro.

En el ámbito nacional, la mayoría de estos sistemas constituyen portales web informativos, los cuales no están concebidos para la gestión de la información relacionada con la actividad científica en las distintas instituciones que los utilizan, siendo esto una necesidad importante dentro del sistema de CTI. En otros casos solo gestionan parte de la actividad de CTI, concentrándose en la gestión de proyectos y grupos de investigación, lo cual no cumple con las necesidades identificadas.

En el contexto de la universidad también se han realizado esfuerzos por informatizar la actividad de CTI, lo cual ha constituido una informatización parcial, debido a que se dispone de soluciones informáticas para gestionar solo la información de algunos procesos de CTI de manera independiente.

A pesar de los diferentes sistemas informáticos que se han identificado con el propósito de gestionar la actividad científica, no ha sido posible la adopción de ninguno como solución al problema.

Por lo que se propone el desarrollo de un sistema informático que responda a las necesidades del sistema de CTI del centro y que permita la gestión de manera integrada de la información generada de la actividad científico, tecnológica y de innovación en la universidad.

CAPÍTULO 2. TECNOLOGÍAS UTILIZADAS

1. Introducción

En el presente capítulo se hace una descripción de las tecnologías utilizadas en el desarrollo de la solución propuesta. Se mencionan las tecnologías del lado del cliente como XHTML, JavaScript, Ajax y jQuery y las tecnologías del lado del servidor como symfony, PHP, Doctrine y PostgreSQL. También se menciona el *framework* Lime utilizado en las prueba automáticas del sistema.

2. Aplicación web

Según plantea Conallen en el libro "*Building Web applications with UML*", las aplicaciones web utilizan tecnologías que posibilitan hacer su contenido dinámico y le permiten al usuario del sistema interactuar y afectar la lógica del negocio en el servidor.[22]

Son aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet, mediante un navegador. Estas son populares debido a lo práctico del navegador web como cliente ligero, así como a la facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software a miles de usuarios potenciales, ya que el control (los accesos, los recursos, la integridad de los datos) se encuentra centralizado en el servidor, lo cual facilita a su vez la tarea de mantener al día los datos y la consistencia de la información.

3. Arquitectura cliente-servidor

Se vive en una era basada en el concepto de redes informáticas, en la que los datos residen en una o varias computadoras (servidores), brindando la posibilidad a los usuarios de que puedan acceder a la información desde cualquier ordenador (cliente) a través de una red.

Según Sommerville [23], cuando un sistema se organiza como un conjunto de servicios y servidores asociados, más clientes que acceden y usan los servicios, se está en presencia del modelo arquitectónico cliente-servidor, el cual está compuesto por los siguientes componentes:

1. Un conjunto de servidores que ofrecen servicios a otros subsistemas.
2. Un conjunto de clientes que llaman a los servicios ofrecidos por los servidores.

3. Una red que permite a los clientes acceder a estos servicios. Esto no es estrictamente necesario, ya que los clientes y los servidores podrían ejecutarse sobre una misma máquina.

Los clientes acceden a los servicios proporcionados por un servidor a través de llamadas a procedimientos remotos, usando un protocolo de petición-respuesta tal como el protocolo *http* usado en la *World Wide Web*. Este estilo arquitectónico es utilizado en el desarrollo de aplicaciones web.

4. Tecnologías y lenguajes del lado del cliente

Las tecnologías del lado del cliente son aquellas que se utilizan para el desarrollo de la interfaz gráfica, permitiendo el intercambio amigable entre el usuario y el sistema. Estas son independientes del servidor, de manera que solo se ocupan de aspectos relacionados directamente con el cliente de la aplicación y su procesamiento queda a cargo del navegador.

4.1. Lenguaje de Marcado de Hipertexto Extensible

XHTML (acrónimo en inglés de *Extensible Hypertext Markup Language*), según lo expuesto en el sitio web de *World Wide Web Consortium* (W3C) [24], es una nueva versión del HTML[4], pensado para sustituirlo como estándar para las páginas web y se basa en XML[5] cumpliendo sus especificaciones más estrictas.

XHTML tiene una serie de ventajas sobre HTML [25], entre ellas:

- Un navegador no necesita implementar heurísticas para detectar lo que quiso poner el autor, ya que al cumplir la especificación de XML de estar bien formado[6], elimina la ambigüedad y el analizador sintáctico (*parser*) puede ser más sencillo.

[4] Es un tipo especial de documento de texto utilizado por los navegadores web para representar gráficos, textos, imágenes. Es la lengua franca para publicar hipertexto en la *World Wide Web*.

[5] Acrónimo de *Extensible Markup Language,* es un estándar para el intercambio de información estructurada entre diferentes plataformas.

[6] Los documentos bien formados son aquellos que cumplen con todas las definiciones básicas de formato y pueden, por ende, analizarse correctamente por cualquier *parser* que cumpla con la norma.

- Permite una correcta interpretación de la información independientemente del dispositivo desde el que se accede a ella, debido a que es un lenguaje cuyo etiquetado es más estricto que HTML.
- Se pueden incorporar elementos de distintos espacios de nombres XML (como *MathML* y *Scalable Vector Graphics*)
- Como es XML, se pueden utilizar fácilmente herramientas creadas para el procesamiento de documentos XML genéricos.

4.2. JavaScript

JavaScript es un lenguaje de programación interpretado que permite aumentar el potencial de HTML con animaciones, interactividad con el usuario y efectos visuales dinámicos. [26]

Permite a las páginas web responder instantáneamente ante acciones como: hacer un clic en un enlace, llenar un formulario o simplemente mover el *mouse* sobre una determinada zona de la pantalla. Esto es debido a que JavaScript no sufre de la demora asociada a los lenguajes de programación del lado del servidor como PHP, que se basan en la comunicación entre el navegador y el servidor web. [26]

JavaScript, de los de su tipo, es uno de los lenguajes más utilizado en los navegadores web [27], por lo que no debe ser preocupación la compatibilidad con los mismos.

Este lenguaje es utilizado en el desarrollo de la aplicación mediante el empleo de la tecnología Ajax, la cual se describe a continuación.

4.3. Ajax

Ajax es la abreviatura de *Asynchronous JavaScript + XML,* y según James [28], quien introdujera el término en febrero de 2005, Ajax no es una tecnología en sí, sino la unión de varias tecnologías independientes que se unen de formas nuevas y sorprendentes.

Las tecnologías que forma Ajax son:

- XHTML y CSS: para crear una presentación basada en estándares.
- DOM: para la interacción y manipulación dinámica de la presentación.
- XML y XSLT: para el intercambio y la manipulación de información.

- XMLHttpRequest: para el intercambio asíncrono de información.
- JavaScript: para unir todas las demás tecnologías [28].

La manera tradicional como se creaban las aplicaciones web no daba una buena sensación al usuario, ya que al realizar peticiones continuas al servidor, este debía esperar a que se recargara la página completa con los cambios solicitados. Si la aplicación debe realizar peticiones continuas, su uso se convierte en algo molesto.

Ajax permite mejorar la interacción del usuario con la aplicación, evitando las recargas constantes de la página, pues el intercambio de información con el servidor se produce en un segundo plano. Esto permite además, aumentar la velocidad y la usabilidad de las aplicaciones, lo cual ha posibilitado que se reduzca la brecha que existe entre las aplicaciones web y las aplicaciones de escritorio.

La utilización de Ajax brinda un nuevo enfoque a las aplicaciones web y representa un cambio fundamental en lo que se puede hacer en la Web [28].

4.4. jQuery

jQuery es una librería de clases o *framework* JavaScript de código abierto [29], que permite simplificar la manera de interactuar con los documentos HTML, manejar eventos, crear animaciones y agregar interacción a las páginas web con la tecnología Ajax. Utilizar esta librería posibilita obtener resultados con pocas líneas de código y menor tiempo [30].

jQuery brinda un conjunto de beneficios como:

- Las aplicaciones funcionan de igual manera en la mayoría de los navegadores web: Internet Explorer, Mozilla Firefox, Opera y otros.
- Sintaxis sencillas.
- Cuenta con una gran comunidad de desarrollo en línea.
- Posee abundante documentación.
- Cuenta con extensiones opcionales que añaden funcionalidades, como: jQueryUI [31].

jQuery es actualmente uno de los *framework* JavaScript más populares [32] y es usado en el desarrollo de numerosas aplicaciones en todo el mundo [33] [29].

5. Tecnologías y lenguajes del lado del servidor

Las tecnologías del lado del servidor son aquellas que permiten el procesamiento (operaciones y/o acceso a bases de datos) de las peticiones de usuarios mediante la interpretación de un *script* en el servidor web, que genera el código HTML dinámicamente como respuesta.

5.1. PHP

PHP responde a un acrónimo recursivo: "*PHP: Hypertext Preprocessor*", pero también es conocido por su nombre original: *Personal Home Page* o *Personal Home Page Tool*. Este es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Este lenguaje es uno de los más populares en el desarrollo de aplicaciones web [34].

Según reportes de *Netcrasft survey*, hasta abril de 2008 PHP había sido usado en el desarrollo de alrededor de 30 millones de sitios web [35].

A continuación, algunas de las características de PHP que lo convierten en una de las mejores alternativas [36].

- **Rendimiento**: los *scripts* de PHP se ejecutan más rápidamente que los escritos en otros lenguajes interpretados. El motor de PHP 5.0 fue rediseñado por completo con un gestor de memoria optimizado para mejorar el rendimiento y es notablemente mucho más rápido que versiones anteriores.

- **Portabilidad**: PHP está disponible para casi todos los sistemas operativos (UNIX, Microsoft Windows, Mac OS, OS/2), por lo que los programas escritos en PHP son portables. Esta capacidad es valiosa, especialmente cuando se opera en un entorno multiplataforma [37].

- **Facilidad de uso**: PHP es un lenguaje de programación extremadamente sencillo. Su sintaxis es clara, consistente y posee una amplia documentación donde se describen las más de cinco mil funciones incluidas en el *core*. Esto reduce significativamente la curva de aprendizaje, tanto para los programadores novatos como para los experimentados y es una de las razones por las cuales PHP es uno de los favoritos en la creación de aplicaciones web.

- **Código abierto**: este lenguaje es desarrollado por voluntarios de la *world wide web* que ponen el código fuente a disposición de la comunidad de desarrollo gratuitamente y puede, por ende, ser utilizado sin pago por concepto de

licencias. El hecho de que sea un lenguaje de código abierto permite que cualquier desarrollador, en cualquier parte, pueda revisar el código, identificar errores y sugerir posibles soluciones. Esto hace de PHP un producto estable y robusto, donde los errores de programación una vez descubiertos, son resueltos rápidamente.

- **Comunidad de soporte**: una de las ventajas que tiene PHP es la gran comunidad de soporte que posee. Cientos de programadores alrededor del mundo implementan componentes reutilizables que son almacenados en PEAR (*Extension and Application Repository*).

5.2. Symfony

Un *framework* simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver tareas comunes. Además, proporcionan estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, facilita la programación de aplicaciones, pues encapsula operaciones complejas en instrucciones sencillas [38].

Symfony es un completo *framework* diseñado para optimizar el desarrollo de las aplicaciones web y se ajusta a los requisitos siguientes: [38]

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares)
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de *phpDocumentor* y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel [38]

Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza tareas comunes permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación, por ejemplo:

- La capa de internacionalización que incluye symfony permite la traducción de los datos y de la interfaz, así como la adaptación local de los contenidos.
- La capa de presentación utiliza plantillas y *layouts* que pueden ser creados por diseñadores HTML sin ningún tipo de conocimiento del *framework*. Los *helpers* incluidos permiten minimizar el código utilizado en la presentación, ya que encapsulan grandes bloques de código en llamadas simples a funciones.
- Los formularios incluyen validación automatizada y relleno automático de datos ("*repopulation*"), lo que asegura la obtención de datos correctos y mejora la experiencia de usuario.
- Los datos incluyen mecanismos de escape que permiten una mejor protección contra los ataques producidos por datos corruptos.
- La gestión de la caché reduce el ancho de banda utilizado y la carga del servidor.
- La autenticación y la gestión de credenciales simplifican la creación de secciones restringidas y la gestión de la seguridad de usuario.
- El sistema de enrutamiento y las URL limpias permiten considerar a las direcciones de las páginas como parte de la interfaz, además de estar optimizadas para los buscadores.
- Los *plugins* y las factorías (patrón de diseño *Factory*) permiten realizar extensiones a medida de symfony [38].

Crear aplicaciones con symfony permite crear páginas XHTML válidas, crear una configuración sencilla, abstracción de la base de datos utilizada, enrutamiento con URL limpias, varios entornos de desarrollo y muchas otras utilidades para la creación de aplicaciones.

Existe un gran número de *framework* PHP, como Zend Framework, CodeIgniter, CakePHP, PRADO, PHPDevShell, Yii, Seagull, Akelos, Zoop, QPHP y otros. [39] [40] [41] Escoger un *framework* u otro puede resultar una tarea compleja, pues cada uno posee sus ventajas y desventajas, teniendo en cuenta tamaño del proyecto, tiempo de desarrollo, nivel de conocimiento del equipo sobre la tecnología y demás. Aun así, las ventajas son indiscutibles y trabajar sin uno significaría más esfuerzo, más tiempo, menos seguridad y más dificultades para modificaciones futuras.

5.2.1. Patrón MVC

Symfony está basado en el patrón clásico de diseño Modelo Vista Controlador (MVC), que está formado por tres capas [38]:

- El Modelo representa la información con la que trabaja la aplicación (lógica de negocio).
- La Vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El Controlador se encarga de procesar las interacciones del usuario y realizar los cambios en el modelo o en la vista.

Este patrón separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones.

Con el objetivo de simplificar la programación, symfony subdivide estas capas de la siguiente manera [38]:

- La capa del Modelo
    - Abstracción de la base de datos
    - Acceso a los datos
- La capa de la Vista
    - Plantilla
    - Layout
- La capa del Contralodor
    - Controlador frontal
    - Acción

La capa del modelo se divide en la capa de acceso a los datos y en la capa de abstracción de la base de datos. De esta forma, las funciones que acceden a los datos no utilizan sentencias ni consultas que dependa del SGBD, sino que utilizan otras

funciones para realizar las consultas. La capa de abstracción es totalmente transparente para el programador.

La vista se separa en un *layout* y en plantillas. El primero almacena el código HTML de los elementos comunes a todas las páginas de la aplicación symfony, con el objetivo de no repetirlos en cada una de ellas, como: cabeceras de la página, el pie de página y la navegación global. Las plantillas representan las salidas HTML de las acciones que se ejecutan y estas se insertan en el *layout*.

Por otra parte la capa del controlador se divide en el controlador frontal y en acciones. El controlador frontal es el único punto de entrada a una aplicación symfony y es el encargado de cargar todas las configuraciones y determinar la acción a ejecutarse. Este, muy rara vez hay que modificarlo, por lo que solo se tendrán en cuenta las acciones a la hora de modelar el sistema. Las acciones por su parte, contienen toda la lógica de la aplicación, verifican la integridad de las peticiones, utilizan el modelo y preparan los datos requeridos para la capa de presentación. Las acciones son métodos con el nombre *execute<NombreAccion>* de una clase llamada *<nombreModulo>Actions* que hereda de la clase *sfActions* y se encuentran agrupadas por módulos symfony (casos de uso) [38].

La implementación que hace symfony del patrón MVC es un aspecto importante a tener en cuenta a la hora de diseñar el sistema, específicamente para modelar los diagramas de clases del diseño.

5.2.2. Plugins

Un *plugin* permite agrupar todo el código diseminado por diferentes archivos y reutilizarlo en otros proyectos. Al igual que un proyecto, un *plugin* puede tener clases, *helpers*, configuraciones, tareas, módulos, esquemas, e incluso recursos Web (CSS, JavaScript, etc.) El directorio de un *plugin* se organiza de forma similar al directorio de un proyecto [38].

Existen los *plugins* públicos y los privados. Los públicos son aquellos que están disponibles para toda la comunidad y pueden ser descargados e instalados con el objetivo de agregar funcionalidades a un proyecto [42]. Un ejemplo es el *sfDoctrineGuardPlugin,* que proporciona funciones de autenticación y autorización, siendo uno de los más populares [43]. Los *plugins* privados son los que su uso está restringido a una empresa o institución [42], tal y como se proponen en la solución.

Una de las ventajas de los *plugins* es que permiten extender la aplicación mediante la adición de nuevas funcionalidades encapsuladas en un directorio *plugin,* como pueden ser funcionalidades de integración con otros componentes.

5.2.3. Tareas

Una tarea es una porción de código que puede ser ejecutada mediante la línea de comandos utilizando el *script* PHP symfony que se encuentra en la raíz del proyecto. Symfony incluye un conjunto de tareas de propósito general, pero también brinda la posibilidad de desarrollar tareas propias para ampliar las existentes [44]. Una tarea es una clase que debe heredar de *sfBaseTask*.

La ejecución de la tarea puede ser manual o automática cada determinado tiempo. Para ejecutar la tarea automáticamente, existe en la mayoría de los sistemas UNIX y GNU/Linux un mecanismo denominado *cron* que dispone de un archivo de configuración (*crontab*), en el que busca los comandos que se deben ejecutar en cada momento. Las tareas de symfony se pueden integrar fácilmente a este mecanismo [45].

5.3. Doctrine

Las bases de datos siguen una estructura relacional, mientras que la mayoría de las aplicaciones están diseñadas para usar la orientación a objetos. Por tanto, para acceder a la base de datos desde una aplicación, es necesaria una interfaz que traduzca la lógica de los objetos a la lógica relacional y viceversa. Esta interfaz se denomina “mapeo de objetos a bases de datos” (ORM, por sus siglas en inglés).

La utilización de un ORM aporta varias ventajas como las siguientes: [38]

- Portabilidad, porque hace posible cambiar la aplicación a otra base de datos, incluso a mitad de desarrollo. Si se comenzara a desarrollar el proyecto con un gestor de base de datos determinado, por ejemplo, MySQL y se decidiera cambiar para PostgreSQL, Oracle o cualquier otro, bastaría cambiar una línea en un archivo de configuración y todo seguiría funcionando correctamente.

- Reutilización, ya que permite llamar a los métodos de un objeto desde varias partes de la aplicación e incluso desde diferentes aplicaciones.

- Seguridad, pues los ORM implementan mecanismos para evitar tipos de ataque como pudieran ser las inyecciones SQL.

- La utilización de objetos en lugar de registros y de clases en lugar de tablas, permite añadir métodos de acceso en los objetos que no tienen relación directa con una tabla.

Existen varios ORM, entre los que se encuentran Propel para PHP, Hibernate para la tecnología Java, LINQ para .Net, entre otros [46]. Desde la versión 1.3 de symfony, Doctrine se ha convertido oficialmente en su ORM por defecto [47].

5.4. PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente [48]. Está considerado como el sistema de gestión de bases de datos de código abierto más avanzado y potente del mercado [49] [50].

A continuación, algunas de sus características [49] [50] [51]:

- Soporta alta concurrencia: mediante un sistema denominado MVCC (acceso concurrente multiversión, por sus siglas inglés) permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo *commit.*

- Soporta integridad referencial: utilizada para garantizar la validez de los datos de la base de datos.

- Es altamente extensible: los usuarios pueden definir nuevos operadores, funciones, índices y tipos de datos.

- Cumple con la atomicidad, consistencia, aislamiento y durabilidad (ACID por sus siglas en inglés).
  Atomicidad: es la propiedad que asegura que la operación se ha realizado o no y por tanto ante un fallo del sistema no puede quedar a medias.
  Consistencia: asegura que solo se empieza aquello que pueda acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos.
  Aislamiento: asegura que una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generarán ningún error.
  Durabilidad: asegura que una vez realizada la operación, esta persistirá y no se podrá deshacer aunque falle el sistema.

- Está disponible para los sistemas operativos Linux, Unix (para todas sus variantes) y Windows.

- Posee amplia documentación, bien organizada, pública y libre.

Se ha probado que PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios [49].

6. Comunicación entre sistemas

Existen varias vías para lograr la comunicación entre aplicaciones web y la obtención de información. Una de ellas es a través de peticiones HTTP mediante la simulación de clientes web capaces de obtener el contenido de un sitio. Para PHP existen varias librerías de clases que implementan esta técnica, como la clase *HttpClient* [52] [53], *PHP HTTP protocol client* [54] o *Snoopy* [55].

Para symfony existe el *plugin sfWebBrowserPlugin* que también propone un cliente HTTP capaz de hacer peticiones web basado en una URL [56].

Esta técnica es especialmente útil cuando se necesita obtener información de un sistema y no se dispone de otro mecanismo para lograr la integración.

Una de las desventajas de esta forma de comunicación es que no siempre es posible obtener del contenido HTML todos los datos necesarios. Además de esto, obtener información por esta vía constituye un riesgo, teniendo en cuenta la posibilidad de ocurrencia de cambios en la estructura del HTML. Esta técnica tampoco es viable cuando el sitio consultado tiene comportamiento Ajax y ejecuta código *javascript*.

En cambio, los servicios web proporcionan mecanismos de comunicación estándares entre aplicaciones, que interactúan entre sí para presentar información dinámica al usuario, proporcionando interoperabilidad y extensibilidad [57]. Esta comunicación se establece independientemente del lenguaje de programación con que se hayan desarrollado los sistemas y de la plataforma en que se ejecutan [58], por tanto, constituyen la solución para lograr la integración con los sistemas propuestos: SIndiCIT y sistema de gestión editorial de la SC-UCI.

Uno de los estándares asociados a los servicios web es el *Web Service Description Language* (WSDL), que permite que un servicio y un cliente establezcan un contrato sobre los detalles de transporte de mensajes y su contenido, a través de un documento procesable por dispositivos. WSDL especifica la sintaxis y los mecanismos de intercambio de mensajes [57].

Dentro del proceso de creación de servicios web, la generación del WSDL es una de las tareas más dificultosas. Aun cuando existen editores gráficos que facilitan la creación, edición, visualización y validación de WSDL [59, 60], pueden ocurrir inconsistencias entre los servicios y el contrato que los describe, debido a que –independientemente del nivel de automatización que tengan estas herramientas- este proceso no es automatizado totalmente [61].

Existe un *plugin* para symfony llamado *ckWebServicePlugin* que permite crear un API de servicios web para aplicaciones desarrolladas con este *framework.* El *plugin* tiene integrado un componente, que a diferencia de los editores, genera el WSDL a partir del código fuente [62], de modo que el desarrollador solo debe indicar las acciones que serán expuestas como servicios y ejecutar la tarea *webservice:generate-wsdl* para que se realice el proceso.

7. Pruebas automáticas

El proceso de prueba en el desarrollo de *software* es una de las tareas más difíciles. Constantemente los requerimientos cambian, por lo que se debe reimplementar parte de la aplicación para adaptarla a las nuevas necesidades. Lograr diseñar casos de pruebas que garanticen la funcionalidad de cada uno de los componentes, analizar los resultados y ejecutarlos cada vez que ocurra un cambio, es una tarea que requiere de mucho tiempo y esfuerzo.

La automatización de las pruebas es uno de los mayores avances en la programación desde la invención de orientación a objetos [38]. El uso de herramientas *software* ayuda a reducir el tiempo y el esfuerzo implicado en los procesos de prueba [23]. Permite conocer en todo momento si la aplicación funciona correctamente antes los cambios internos. Después de una refactorización del código la posibilidad de contar con pruebas automáticas garantiza encontrar de forma inmediata los errores introducidos y las incompatibilidades surgidas entre los componentes.

En el ámbito de PHP existen varios *framework* para la realización de pruebas automáticas, siendo los más conocidos PhpUnit y SimpleTest [38]. Symfony incluye su propio *framework* llamado Lime.

7.1. El framework de pruebas Lime

Es el *framawork* de pruebas que utilizan las versiones 1.x de symfony. Está basado en la librería *Test::More* de Perl y es compatible con *Test Anything Protocol (TAP)* lo que facilita la lectura de los resultados de las pruebas [38]. Algunas de las ventajas que posee son:

- Ejecuta los archivos de prueba en un entorno independiente para evitar interferencias entre las diferentes pruebas.
- Las pruebas de Lime son fáciles de leer, al igual que sus resultados. En los sistemas operativos que lo soportan, los resultados de Lime utilizan diferentes colores para mostrar de forma clara la información más importante.
- El núcleo de Lime se valida mediante pruebas unitarias.
- Está escrito con PHP, es muy rápido y está bien diseñado internamente. Consta únicamente de un archivo, llamado *lime.php*, y no tiene ninguna dependencia [38].

Con el *framework* Lime se pueden crear dos tipos de pruebas automáticas: las unitarias y las funcionales.

Las pruebas unitarias comprueban que todos los métodos funcionan correctamente y cada una de estas pruebas debe ser completamente independiente de las demás [63].

Por otro lado, las pruebas funcionales verifican que la aplicación funciona correctamente en su conjunto, desde la petición realizada por un navegador hasta la respuesta enviada por el servidor, es decir, que prueban todas las capas de la aplicación: el sistema de enrutamiento, el modelo, las acciones y las plantillas (que todos los elementos se muestren correctamente en la página, al pulsar sobre enlaces y botones), tal y como se haría manualmente. En resumen, se prueba un escenario correspondiente a un caso de uso [63].

Esta herramienta permite describir los escenarios de la aplicación de forma sencilla. Una vez definidos los escenarios se pueden ejecutar automáticamente una y otra vez, simulando el comportamiento de un usuario con el navegador.

8. Conclusiones del capítulo

En este capítulo se expusieron las tecnologías utilizadas en el desarrollo de la solución, las cuales han sido aprobadas por comisiones de arquitectura de la universidad para el desarrollo de aplicaciones web.

Existe gran variedad de *framework* para PHP y escoger uno u otro puede resultar una tarea compleja considerando sus ventajas y desventajas, pero independientemente de ello, las ventajas de usar un *framework* son indiscutibles.

En el caso del *framework* symfony proporciona una arquitectura bien definida y probada, la cual establece una clara definición entre las capas de presentación y la

lógica de negocio. Dentro de esta, la concepción de *plugin* permite que sea fácil de extender, lo que permite su integración con aplicaciones desarrolladas por terceros. Esto demuestra la alta escalabilidad que se puede obtener en las aplicaciones desarrolladas con symfony. Además, cuenta con el *plugin ckWebServicePlugin* que permite crear fácilmente un API de servicios web para aplicaciones desarrolladas con este *framework.*

Como metodología de desarrollo se utiliza el Proceso Unificado (RUP, por sus siglas en inglés), que utiliza el Lenguaje Unificado de Modelado (UML) para representar los esquemas del software, y como herramienta CASE para UML: *Visual Paradigm*.

CAPÍTULO 3. PROPUESTA DE SOLUCIÓN

1. Introducción

En este capítulo se presenta la propuesta de solución a partir del análisis de integración de los sistemas informáticos utilizados en la universidad en la gestión editorial de la SC-UCI y en la medición de la producción científica. Se explican las modificaciones realizadas en el *OJS* con el objetivo de integrarlo en el sistema de gestión y se exponen varios elementos a tener en cuenta para integraciones futuras. Se muestra además la forma en que se ejecutaron las pruebas, así como los resultados que permitieron validar la propuesta de solución.

2. Concepción general del sistema integrado

La propuesta consiste en un sistema capaz de integrar las aplicaciones utilizadas en algunos procesos de CTI (*OJS* para la gestión editorial de la SC-UCI y sistema de indicadores cienciométricos SIndiCIT), con el objetivo de gestionar de manera consistente los datos manipulados en el sistema de CTI.

En la Fig. 1 se muestra la concepción general del sistema integrado de CTI.

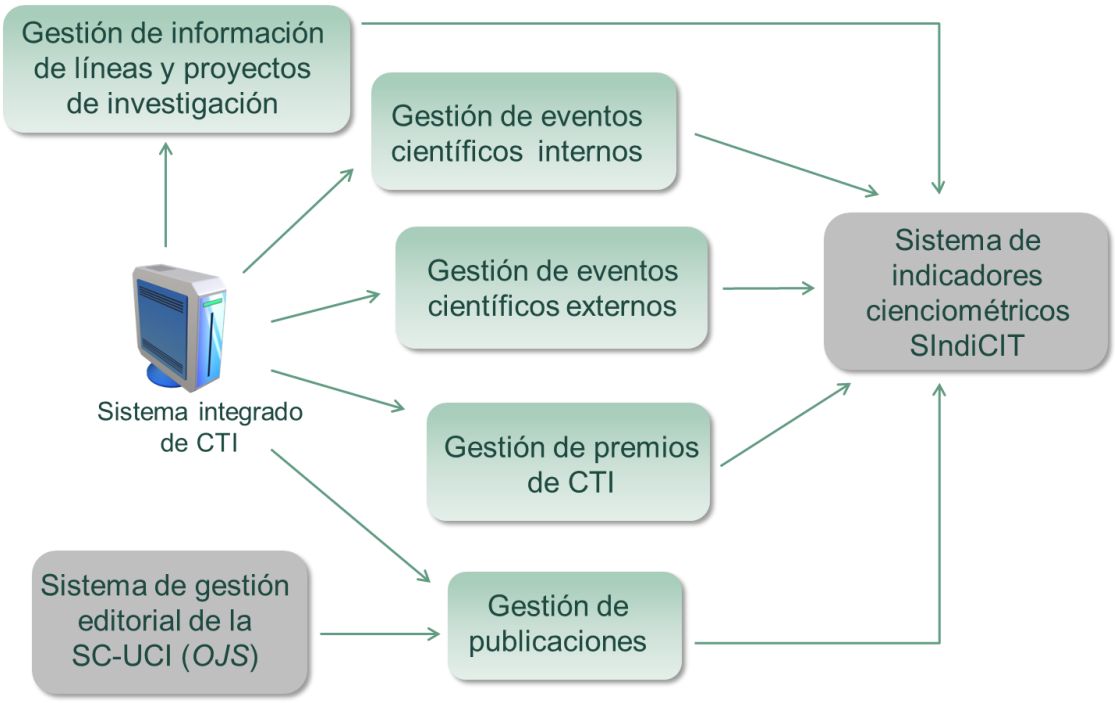

Fig. 1. Concepción del sistema integrado de CTI

El sistema dispone de un adecuado control de acceso que se maneja a través de la asignación de permisos a los diferentes roles. Esta aplicación será accedida desde las diferentes estructuras del sistema de CTI, donde se introducirá la información básica. Esto permitirá a una estructura central (Dirección de Investigaciones) validar dicha información y poder contar en todo momento con los datos más actualizados.

Para lograr el sistema integrado se proponen adecuaciones al *OJS* (acápite 4.1.1) y se define un conjunto de servicios web en el sistema de gestión de CTI (acápite 4.2), que garantizan la interoperabilidad en la propuesta de solución.

3. Diseño de la propuesta de solución

Según Jacobson *et al.*, en el diseño se formulan los modelos que soporten todos los requisitos, incluyendo los requisitos no funcionales y otras restricciones y sus propósitos son:

- Adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, tecnologías de distribución y de interfaz de usuario.
- Crear una entrada apropiada y un punto de partida para actividades de implementación subsiguiente, capturando los requisitos o subsistemas individuales, interfaces y clases.
- Descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo.

Capturar las interfaces entre los subsistemas de forma temprana en el ciclo de vida del software, lo cual es muy útil cuando se utilizan interfaces como elementos de sincronización entre diferentes equipos de desarrollo [64].

Para realizar el diseño de la aplicación es esencial tener en cuenta la arquitectura que propone el *framework* symfony, basada fundamentalmente en el patrón MVC (acápite 5.2.1 del capítulo 2) que separa la lógica del negocio, la lógica de la aplicación y la presentación de los datos, lo cual permite un fácil mantenimiento de las aplicaciones.

3.1. Subsistemas de diseño

Los subsistemas de diseño son una forma de organizar los artefactos del diseño en piezas más manejables. Puede constar de clases del diseño, realizaciones de casos de uso y otros subsistemas (recursivamente). También pueden proporcionar interfaces que representan la funcionalidad que exportan en términos de operaciones [64]

Los subsistemas deben ser cohesivos, es decir, sus contenidos deben estar fuertemente asociados, y deben además estar débilmente acoplados, o sea, que las dependencias entre unos y otros o entre sus interfaces deben ser mínimas. Estos suelen tener trazas directas hacia los paquetes del análisis [64].

En la Fig. 2 se muestran los subsistemas de diseño identificados de acuerdo al conjunto de funcionalidades afines que agrupan, así como sus dependencias.

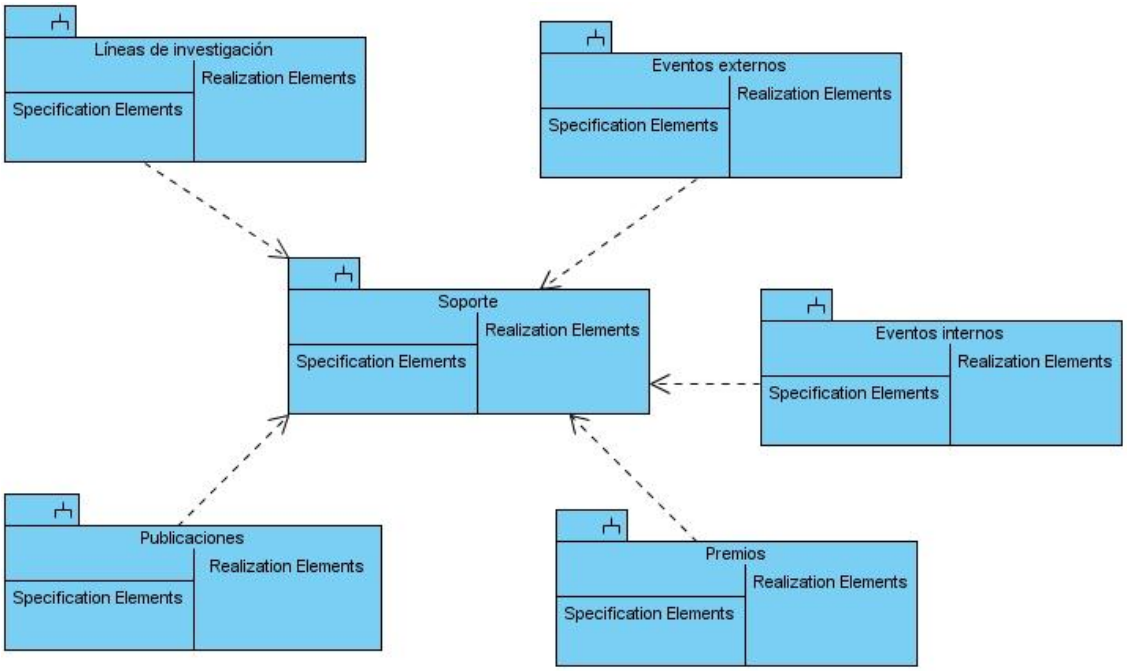

Fig. 2 Relaciones entre los subsistemas del diseño

En el módulo Soporte se gestiona la información común al resto de los subsistemas, como son los datos de las personas; por ende, los subsistemas Publicaciones, Eventos externos, Eventos internos, Premios y Líneas de investigación tienen una relación de dependencia con el subsistema Soporte.

Estos subsistemas son desarrollados como *plugins* para agrupar los archivos según su funcionalidad, por ejemplo: guardar juntos los archivos relacionados con la gestión de publicaciones (modelos, módulos symfony y plantillas), logrando así una mejor organización en la aplicación y aprovechar las potencialidades de los *plugins*.

4. Integración

En el sistema de gestión editorial de la SC-UCI se registran las publicaciones de la comunidad universitaria en esa revista. El SIndiCIT necesita de una serie de entradas relacionadas con los indicadores cienciométricos para medir la producción científica.

Por otro lado, en la propuesta de solución se gestionará información de las publicaciones y de otros indicadores como eventos científicos, premios de CTI y demás, que necesita el SIndiCIT para su funcionamiento.

Teniendo en cuenta lo anterior y la necesidad de evitar inconsistencia en la información de CTI que se gestiona, se propone:

- Extender el *OJS* mediante la implementación de servicios web que permitan obtener las publicaciones registradas en la SC-UCI.
- Extender el sistema de gestión de CTI mediante la implementación de servicios web que brinden la información necesaria al SIndiCIT.

La integración con dichos sistemas se logra a través de servicios web.

4.1. Integración con el *Open Journal System,* sistema de gestión editorial de la SC-UCI

4.1.1. Implementación del servicio web en el OJS

El *OJS* es un producto de s*oftware* libre y liberado bajo la licencia *GNU General Public License* (*GNU GPL*), por lo que se puede descargar y modificar su código, por tanto, puede instalarse y controlarse de manera local en un servidor propio [21]. Estas características dan la posibilidad de extenderlo mediante la implementación de servicios web e integrarlo a la propuesta de solución.

El *OJS* en su arquitectura define un directorio con nombre *page,* en el cual se crea una carpeta por cada uno de los casos de uso. Estas contienen dos ficheros: *index.php* y un fichero con la definición de una clase que hereda de *Handler* (perteneciente al núcleo del sistema). Esta es la estructura básica que se encarga de atender las solicitudes del cliente. El fichero *index.php* se encarga de *enrutar* las peticiones y en las subclases de *Handler* se implementa la lógica de la aplicación, mediante la definición de las acciones del sistema [65].

Para implementar el servicio web se creó un directorio *webService* dentro de la carpeta *page,* que contiene el fichero *index.php* y la clase *WebServiceHandler*. El *index.php* solo tiene como función incluir esta clase, ya que determinar el método que debe ser ejecutado es una responsabilidad que se deja a la clase *SoapServer*[7]*,* pues el método invocado viene incluido dentro del mensaje *Soap* que envía el cliente.

[7] Clase que define PHP para la creación de servicios web.

Por defecto, el *OJS* invoca el método *index* de la clase *WebServerHandler,* punto donde se establece el servicio web que brinda las publicaciones de la SC-UCI. Para obtener esta información se utiliza el mecanismo de recuperación de datos que posee el *OJS*.

En el listado 1 se muestra la definición de la clase *WebServerHandler* con su método *index.*

Listado 1. Definición del servicio web en el método *index* de la clase *WebServerHandler*.

```
class WebServiceHandler extends Handler {
    function index($args) {
        $server = new SoapServer(null, array('uri' => 'urn:webservices'));
        $server->setObject($this);
        $server->handle();
     }
    // Definición del resto de los métodos…
}
```

4.1.2. Consumo del servicio web desde el sistema de gestión

Para obtener las publicaciones de la SC-UCI se consume el servicio web que fue implementado en el *OJS* y se registra la información obtenida en el sistema de gestión de CTI. De esta manera, el sistema contiene todas las publicaciones en su base de datos sin necesidad de conectarse al *OJS* cada vez que se hace una solicitud relacionada con este indicador.

El registro de las publicaciones de la SC-UCI en el sistema de gestión, se hace a través de la ejecución de una tarea propia que se implementó para ello.

En la Fig. 3 se muestra el diagrama de clases del diseño correspondiente a este requerimiento.

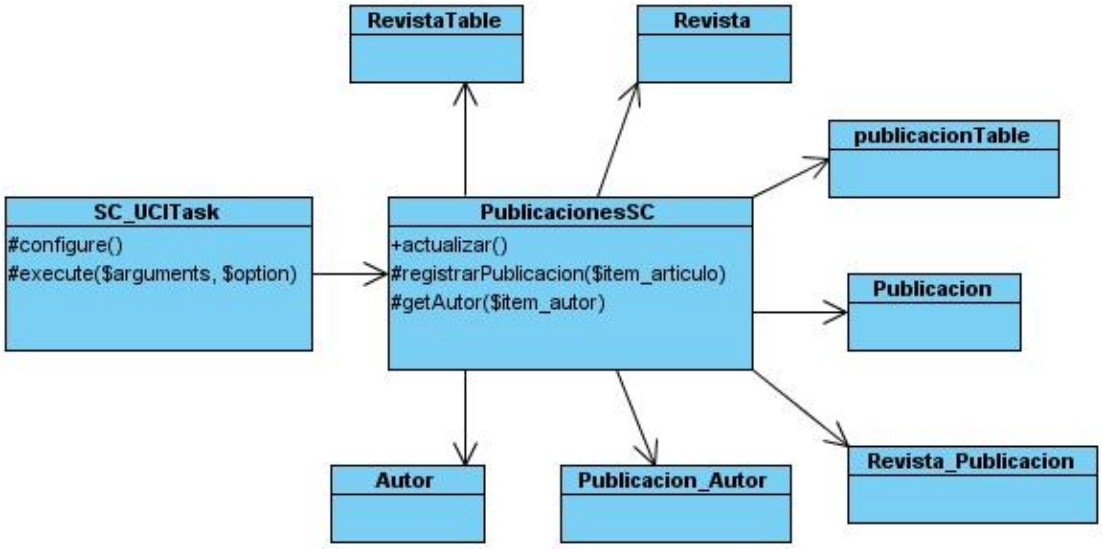

Fig. 3 Diagrama de clases del diseño del caso de uso “Obtener publicaciones de la SC-UCI”

En la clase *SC-UCITask* se define una tarea que es ejecutada mediante la línea de comandos de symfony y se encarga de obtener las publicaciones de la SC-UCI y registrarlas en la base de datos.

Tiene un método *configure* donde se realiza la configuración de la tarea. Es en este método donde se define el nombre de la tarea, una descripción breve y otra detallada de la misma.

En el método *execute* se declara lo que debe realizar la tarea. En el caso particular de *SC_UCITask* utiliza la clase *PublicacionesSC* que realiza la conexión al servicio web del *OJS,* obtienen los artículos con sus autores y los registra en la base de datos.

De este modo se puede ejecutar en la línea de comandos la tarea: *inv:scuci-update* para obtener las publicaciones de la SC-UCI.

4.2. Integración con el sistema de indicadores cienciométricos SIndiCIT

Este sistema depende para su procesamiento, de una serie de entradas referente a los indicadores cienciométricos, los cuales son introducidos por las diferentes estructuras de CTI. A partir de estas entradas y de un mecanismo de ponderación, el sistema brinda un conjunto de salidas que posibilitan el análisis y evaluación de la producción científica de la universidad.

Actualmente la información que se introduce en el sistema de indicadores consiste en valores cuantitativos. Por otro lado, el sistema de gestión propuesto manipula datos que pueden servir de entrada al SindiCIT. Teniendo en cuenta esto, se decide la

integración de ambos sistemas para evitar inconsistencia en la información. Por tanto, el sistema de indicadores utilizará para sus cálculos la información que será registrada a través del sistema de gestión de CTI y de esta manera los datos serán introducidos por una única vía.

Considerando lo anterior, se decide extender el sistema de gestión de CTI mediante la creación de servicios web que brinden la información que utilizará el SIndiCIT para los cálculos correspondientes. Para esto se utiliza el *plugin* para symfony *ckWebServicePlugin*.

Este *plugin* genera el WSDL a partir del código fuente, por lo que el desarrollador solo debe indicar las acciones que serán expuestas como servicios y ejecutar la tarea *webservice:generate-wsdl* para que se realice el proceso.

En sentido general, a todos los servicios de la propuesta de solución se les debe pasar como parámetro un intervalo de tiempo (fecha de inicio y fecha de fin) y un área.

Los servicios que brinda el sistema de gestión de CTI son los siguientes:

- *cantidadPublicaciones(fecha_inicio, fecha_fin, area, nivel)*: este método devuelve la cantidad de publicaciones científicas, dependiendo del nivel MES[8] que sea enviado como parámetro.

- *cantidadParticipacionEventos(fecha_incio, fecha_fin, area, tipo_evento)*: devuelve la cantidad de participación en eventos UCI, municipales, provinciales, nacionales o internacionales, en dependencia del parámetro *tipo_evento.*

- *cantidadPremios(fecha_incio, fecha_fin, area)*: devuelve la cantidad de premios obtenidos por un área.

- *cantidadProyectosID(fecha_inicio, fecha_fin, area, nivel_proyecto)*: devuelve la cantidad de proyectos I+D en que participa un área, tanto universitarios, territoriales, nacionales o internacionales, teniendo en cuenta el parámetro *nivel_proyecto*.

En la Fig. 4 se muestra un ejemplo de diagrama de clases del diseño donde se muestra la acción *executeCantidadPublicaciones,* que se publicará como servicio web*.*

[8] Es una clasificación definida por el MES para establecer un *ranking* de publicaciones.

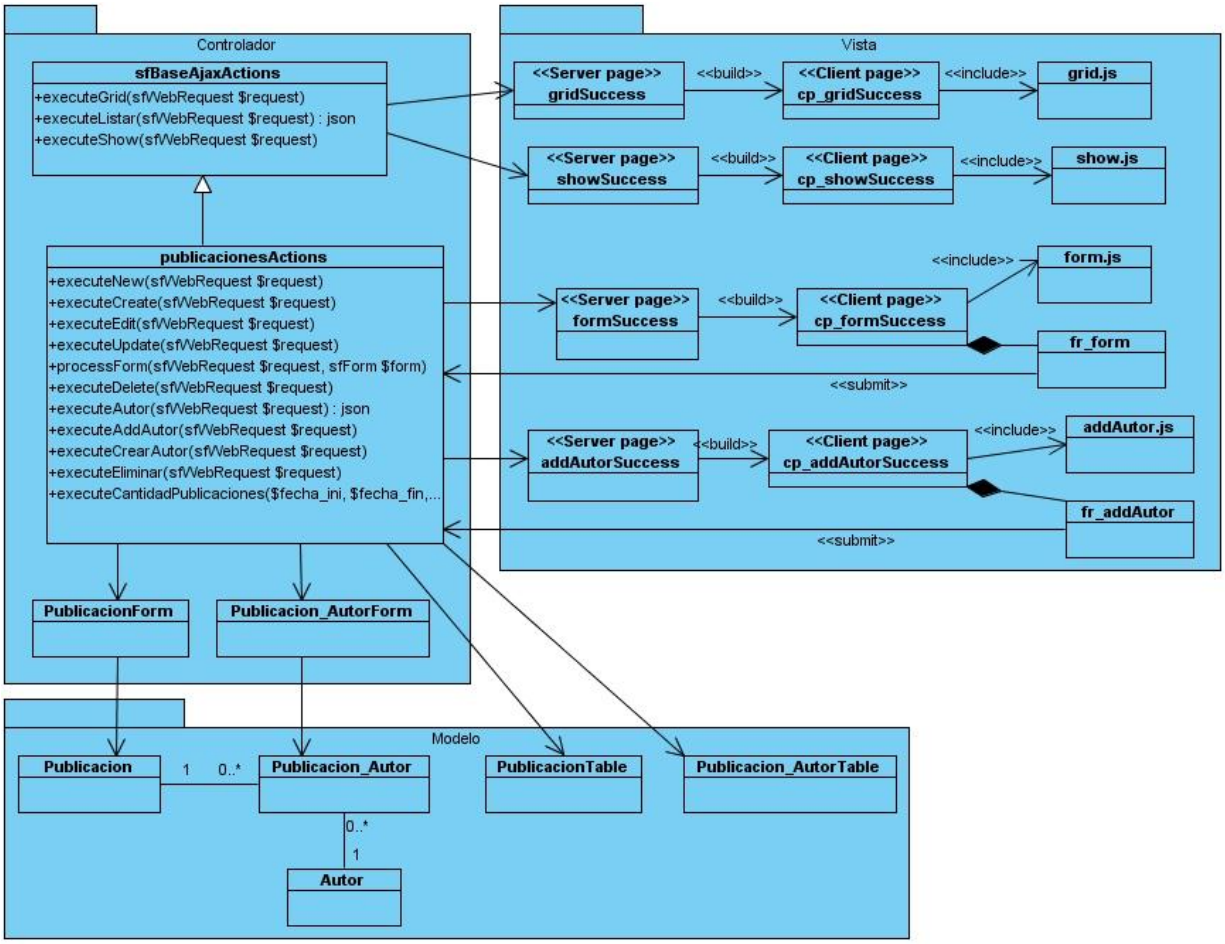

Fig. 4 Diagrama de clases del diseño del caso de uso “Gestionar publicaciones”

La clase *publicacionesActions* contiene la lógica de la aplicación referente al caso de uso “Gestionar publicaciones”. En esta se definen las acciones –métodos con nombre *execute<NombreAcción>-,* que acceden al modelo para obtener o registrar los datos de las publicaciones con sus autores y se envían a las plantillas correspondientes. También contiene la acción *executeCantidadPublicaciones* que se publicará como servicio web.

Regularmente las plantillas son cargadas por defecto y en este caso el *framework* busca la plantilla con nombre *<nombreAcción>Success.php,* a menos que en la acción se defina otra.

La clase *publicacionesActions* hereda de *sfBaseAjaxActions,* pues en esta última se definió un conjunto de acciones que son comunes al resto de los módulos (*plugins*).

Las clases *Publicacion*, *PublicacionTable, Publicacion_Autor, Publicacion_AutorTable* y *Autor* constituyen parte del modelo de la aplicación y contienen la lógica del negocio

referente al caso de uso “Gestionar publicaciones”. El modelo es generado por Doctrine a partir del esquema, el cual genera tres clases por entidad que relacionan las tablas de la base de datos con los objetos de la aplicación:

- *Base<NombreEntidad>:* esta clase se genera directamente a partir del esquema y no debería modificarse, pues cada vez que se ejecuta la tarea *doctrine: build --all-classes* se borra y se vuelve a generar.

- *<NombreEntidad>:* un objeto de esta clase representa un registro de la tabla. Es en esta clase donde se definen los métodos propios para manipular un objeto (salvar, modificar y eliminar).

- *<NombreEntidad>Table:* esta clase define los métodos que regularmente devuelven colecciones de objetos.

Las clases *PublicacionForm* y *Publicacion_AutorForm* pertenecen a la capa Controlador y son las clases que se encargan de las validaciones de los datos introducidos por el usuario.

5. Consideraciones para la integración

En PHP la conexión a un servicio web se hace a través de la clase *SoapClient*, a la cual se le debe pasar en el constructor la dirección del WSDL que describe al servicio y un parámetro *options* que constituye un arreglo opcional [66].

Uno aspecto importante a tener en cuenta es que desde varias partes de la aplicación se podría necesitar consumir un servicio web determinado, para lo cual habría que especificar la dirección del WSDL cada vez que se establezca la conexión con el servicio. Esto pudiera provocar que si ocurre un cambio en dicha dirección, habría que actualizarla en cada punto donde se consumió el servicio web. Esto sucede a menudo cuando en tiempo de desarrollo se utilizan servicios de prueba y en tiempo de explotación de las aplicaciones se utilizan los servicios reales.

Con el objetivo de evitar este inconveniente se utiliza un conjunto de clases que permite al desarrollador abstraerse del mecanismo de conexión, debido a que solo debe interactuar con la clase *slCompFabrica*. Esta recibe en su constructor el nombre del componente con el cual se desea establecer la conexión (y no la dirección del servicio web) [61].

De este modo el desarrollador solo debe conocer el nombre del componente al cual se desea conectar, ya que las especificaciones para la integración se establecen en el

fichero de configuración *componentes.yml* [61], ubicado en el directorio *config* del proyecto, el cual es utilizado por la clase *sICompFabrica*.

En el listado 2 se muestra un ejemplo de las configuraciones necesarias para conectarse con los diferentes componentes. Este es el único fichero que el desarrollador debe modificar si ocurriera algún cambio en la dirección de un determinado WSDL.

Listado 2. Fichero de configuración *componente.yml*.

```
all:
    Telematicos:
        class: TelematicosServicios
        param:
            wsdl: https://serviciosi2.uci.cu/servicios_telematicos_ver3.0/ws_telematicos.php?wsdl
            opciones:
                classmap:
                    Usuario:  UsuarioWS

    SC_UCI:
        class: SC_UCIServicios
        param:
            wsdl: ~
            opciones:
                location: http://publicaciones.uci.cu/index.php/SC/webService
                uri:         urn:webservices
```

Otro elemento importante a considerar, asociado a la integración con otras aplicaciones, es la posibilidad de cambios en la estructura del servicio web consumido, que aunque no es común que ocurra, podría darse la situación, sobre todo, si no existe un compromiso por parte del que publica el servicio.

Una vía de solución es mediante el mapeo del tipo de dato retornado por el servicio a una clase PHP definida en la aplicación. Esto es posible a través de la opción *classmap* que se le puede pasar al constructor de la clase *SoapClient*. A partir de la utilización del fichero de configuración *componentes.yml*, que se muestra en listado 2, esta opción puede definirse en dicho fichero.

La opción *classmap* es un arreglo asociativo donde la clave es el tipo de dato retornado por el servicio y el valor es el nombre de la clase a la cual se desea convertir en la aplicación [66]. Esta opción no es obligatoria. En caso de no definirse, la clase *SoapClient* retorna un objeto de tipo *stdClass*.

Para evitar el acoplamiento entre las clases de la aplicación y los servicios web, se propone la implementación de una clase genérica, una clase que representa a cada

componente al cual se desea integrar y una clase por cada tipo de dato retornado por los servicios.

La definición de las clases y la relación entre ellas se muestran en la Fig. 5. En este diagrama se pone el ejemplo de la clase *UsuarioWS*, que es la clase en la que se convertirá el resultado del método *AutenticarUsuario* (que devuelve una estructura de tipo *Usuario*) del servicio web *ws_telematicos,* utilizado para la autenticación en el sistema.

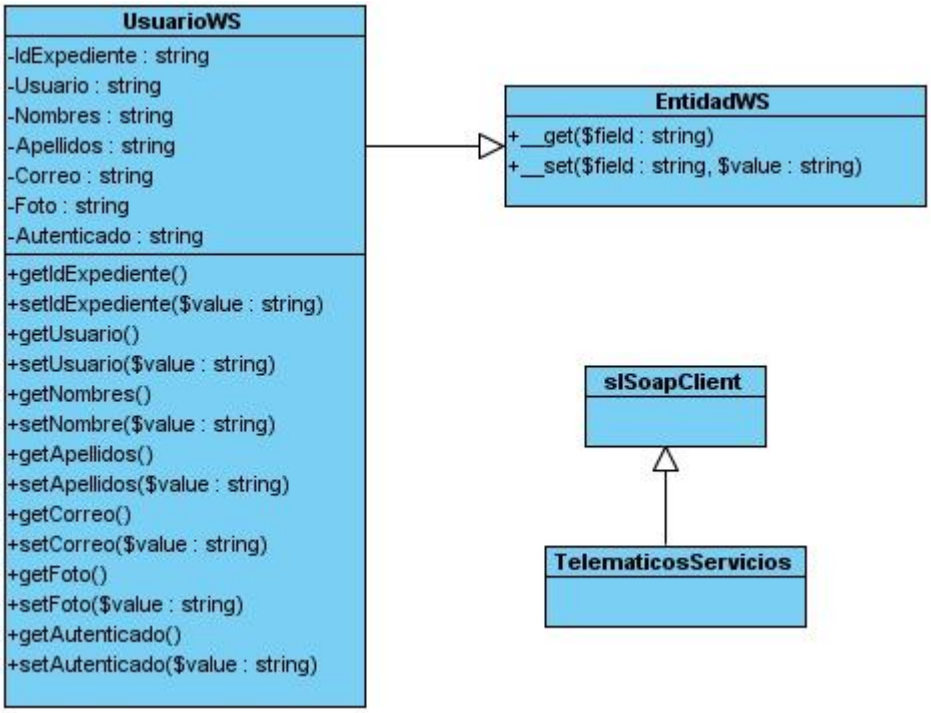

Fig. 5. Clases para el mapeo de servicios a clases de la aplicación

- *EntidadWS* es una clase genérica de la cual deben heredar las clases a las cuales se desean convertir los diferentes tipos de dato retornados por los servicios, en el ejemplo que se analiza: *UsuarioWS*. En *EntidadWS* se definen los métodos mágicos *__get* y *__set.*

- En la clase *UsuarioWS* es donde se definen los atributos del servicio, así como los métodos de acceso: *get* y *set*.

- *TelematicosServicios* es la clase que representa al servicio web *ws_telematicos* y se utiliza para hacer la conexión. Esta debe heredar de la clase *slSoapClient* y en ella se podrían redefinir algunos métodos si fuera necesario.

Estas clases estarán ubicadas en un directorio *clientes_servicios* dentro del *lib* del proyecto.

6. Pruebas de sistema

Durante y después de la implementación, el programa que se desarrolla debe ser probado para asegurar que satisface su especificación y realiza las funcionalidades que esperan los clientes. A este proceso de prueba se le conoce como verificación y validación (V&V).

La verificación implica comprobar que el software está de acuerdo a su especificación. Debe comprobarse que satisface sus requerimientos funcionales y no funcionales. La validación, en cambio, es un proceso más general y su objetivo es asegurar que el software satisface las expectativas del cliente. Teniendo cuenta lo anterior, se puede resumir la definición de verificación y validación a través de las siguientes preguntas:

Verificación: ¿estamos construyendo el producto correctamente?

Validación: ¿estamos construyendo el producto correcto? [67]

Una de las técnicas dentro del proceso V&V son las pruebas del software. Estas implican ejecutar una implementación del *software* con datos de prueba, por lo que constituyen técnicas dinámicas de verificación y validación. Se examinan las salidas del *software* y su entorno operacional para comprobar que funciona tal y como se requiere. El objetivo de estas pruebas es convencer a los desarrolladores del sistema y a los clientes de que el software es lo suficientemente bueno para su uso operacional [23].

Como parte de las pruebas de sistema se encuentran las pruebas de integración, que deben comprobar que los componentes integrados funcionen juntos, que sean llamados correctamente y transfieran los datos correctos a través de sus interfaces [23]. Durante estas pruebas las técnicas que prevalecen son las de diseño de casos de prueba de caja negra, aunque se pueden llevar a cabo algunas pruebas de caja blanca [68].

Una prueba de integración que se define es comprobar que se establezca la conexión al *OJS* y se obtengan los datos correctamente. Para ello se implementó una prueba de unidad utilizando la librería Lime de symfony. En el listado 3 se muestra la implementación de la prueba automática correspondiente.

Listado 3. Prueba automática correspondiente al caso de prueba “Publicaciones de la SC-UCI”.

```
require_once dirname(__FILE__).'/../bootstrap/unit.php';
$t = new lime_test(11);      //Se crea el objeto de prueba, indicando que son 11

$configuration = ProjectConfiguration::getApplicationConfiguration(
  'frontend', 'test', true
);
new sfDatabaseManager($configuration);
Doctrine::loadData(sfConfig::get('sf_test_dir').'/fixtures');

//Se obtiene la conexión al servicio web del OJS y se le cambia la dirección del servicio por la
//del servicio de prueba
$client = slCompFabrica::fabrica('SC_UCI', false);
$client->__setLocation('http://direccion_app/web/test/SC_WebServiceTest.php');

// Se llama al método actualizar para que registre las publicaciones obtenidas del OJS
PublicacionesSC::getInstance()->actualizar();

// Se comprueba que se hayan registrado las publicaciones
$t->ok(Doctrine::getTable('Publicacion')->findOneByTitulo('publicacion1_ART'));
$t->ok(Doctrine::getTable('Publicacion')->findOneByTitulo('publicacion2_ART'));
$t->ok(Doctrine::getTable('Publicacion')->findOneByTitulo('publicacion1_COMCORT'));
$t->ok(Doctrine::getTable('Publicacion')->findOneByTitulo('publicacion2_COMCORT'));

// Se comprueba que se hayan registrado los autores
$t->ok(Doctrine::getTable('Autor')->findOneByEmail('autor1@uci.cu'));
$t->ok(Doctrine::getTable('Autor')->findOneByEmail('autor2@uci.cu'));
$t->ok(Doctrine::getTable('Autor')->findOneByEmail('autor3@uci.cu'));
$t->ok(Doctrine::getTable('Autor')->findOneByEmail('autor4@uci.cu'));
$t->ok(Doctrine::getTable('Autor')->findOneByEmail('autor5@uci.cu'));
$t->ok(Doctrine::getTable('Autor')->findOneByEmail('autor6@uci.cu'));
$t->ok(Doctrine::getTable('Autor')->findOneByEmail('autor7@uci.cu'));
```

Para elaborar la prueba automática anterior, se simuló el servicio web que representa al *OJS*. Este servicio envía datos predefinidos que permiten hacer las comprobaciones necesarias.

El resultado de esta prueba demostró que se establece la conexión al *OJS*, se obtienen los datos del mismo y son registrados correctamente en el sistema de gestión de CTI; garantizando de esta manera la consistencia en la información manipulada en ambos sistemas.

Por otro lado, las pruebas funcionales se centran en establecer que el sistema cumple con sus requerimientos funcionales y no funcionales, por tanto deben basarse en la

especificación de los requerimientos del mismo, así como tener en cuenta las restricciones que pudieran estar presentes en los casos de uso.

Para un requerimiento no se escribe una única prueba, sino que por lo general deben escribirse varias pruebas para asegurar que cubre por completo el requerimiento [23]. Entre más pruebas se implementen, más confianza se podrá tener del sistema porque habrá una mayor probabilidad de exponer los defectos, ya que como dijera Dijkstra *et al.* [69]: las pruebas solo pueden demostrar la presencia de errores, no su ausencia.

A continuación se muestran las pruebas definidas para los requerimientos asociados al caso de uso “Gestionar publicaciones seriadas” del módulo “Publicaciones científicas”:

1. Probar que se registre una revista teniendo en cuenta la obligatoriedad de los datos.
2. Probar que se registre una revista teniendo en cuenta la validez de los datos.
3. Probar que solo se le muestra la opción registrar revista a los usuarios con roles “asesor central” y “responsables de área”.
4. Probar, para un usuario con rol “usuario simple”, que se listen solo las revistas que tienen estado “aprobado”.
5. Probar, para un usuario con rol “especialista de la editorial”, que se listen todas las revistas.
6. Probar, para un usuario con rol “responsable de área”, que se listen las revistas que tienen estado “aprobado” y las revistas en estado “propuesto” que hayan sido registradas por ese usuario.
7. Probar, para el usuario con rol “especialista de la editorial”, que a todas las revistas les salga el vínculo “eliminar”.
8. Probar, para el usuario con rol “usuario simple”, que a ninguna revista le salga el vínculo “eliminar”.
9. Probar, para el usuario con rol “responsable de área”, que solo le salga el vínculo “eliminar” a las revistas que hayan sido propuestas por él y que tengan estado “propuesto”.
10. Probar que cuando se acceda al vínculo “eliminar” haya una revista menos en la base de datos.

En el listado 4 se muestra la implementación de las pruebas automáticas funcionales correspondientes al escenario “Listar publicaciones seriadas”, del caso de uso “Gestionar publicaciones seriadas”.

Se parte sobre la base de tener en la *fixture* de prueba las siguientes publicaciones seriadas:

- Cinco publicaciones seriadas aprobadas.
- Tres publicaciones seriadas propuestas por el usuario “area_user1” con rol “responsable de área”.
- Tres publicaciones seriadas propuestas por el usuario “area_user2” con rol “responsable de área”.

Listado 4. Pruebas automáticas del caso de prueba “Listar publicaciones seriadas”

```
$browser->info('1 - Entrar como un usuario simple')->
   get('/revistas')->
   with('response')->begin()->
      checkElement('table.test_listado_revista tr', 5)->
      checkElement('table.test_listado_revista tr.test_propuesta', false)->
   end()
;

$browser->info('1 - Entrar como un usuario con rol responsable de area')->
   get('/login')->
   click('login', array('user' => 'area_user1', 'pass' => 'area_user_pass'))->
      isStatusCode(200)->
      isRequestParameter('module', 'sfGuardAuth')->
      isRequestParameter('action', 'signin')->
      isRedirected()->
   followRedirect()->
      isStatusCode(200)->
      isRequestParameter('module', 'general')->
      isRequestParameter('action', 'index');
   get('/revistas')->
   with('response')->begin()->
      checkElement('table.test_listado_revista tr', 8)->
      checkElement('table.test_listado_revista
tr.test_propuesta.test_propuesta_por_area_user1', 3)->
      checkElement('table.test_listado_revista tr.test_aprobadas', 5)->
    end()
;

$browser->info('1 - Entrar como un usuario especialista de la editorial’)->
   get('/login')->
   click('login', array('user' => 'administrador', 'pass' => 'admin_pass'))->
      isStatusCode(200)->
      isRequestParameter('module', 'sfGuardAuth')->
      isRequestParameter('action', 'signin')->
      isRedirected()->
```

```
    followRedirect()->
      isStatusCode(200)->
      isRequestParameter('module', 'general')->
      isRequestParameter('action', 'index');
    get('/revistas')->
    with('response')->begin()->
      checkElement('table.test_listado_revista tr', 11)->
    end()
;
```

Luego de varias iteraciones, los resultados finales de estas pruebas permitieron comprobar el correcto funcionamiento del escenario “Listar publicaciones seriadas”.

7. Conclusiones del capítulo

La característica de ser el *OJS* un software de código abierto y tener el equipo de desarrollo la posibilidad de modificar su código fuente, permitió implementar el servicio web con el cual se pueden obtener las publicaciones de la SC-UCI.

Por otro lado, a partir del análisis del funcionamiento del sistema de indicadores cienciométricos, fueron implementados en el sistema de gestión de CTI los servicios web necesarios para mantener la consistencia de los datos manipulados en ambas aplicaciones.

El diseño de la aplicación estuvo determinado por la arquitectura que propone el *framework* symfony, basada fundamentalmente en el uso del patrón arquitectónico MVC, que establece una clara definición de la responsabilidad de cada capa de la aplicación.

El uso de pruebas automáticas, utilizando para su implementación la librería Lime de syfmony, facilitó el proceso de pruebas al cual se sometió la aplicación. Con las distintas iteraciones realizadas y las correcciones necesarias en cada una de ellas, se validó que el sistema funciona correctamente de acuerdo a los requerimientos especificados.

Se comprobó además que la integración con el *OJS* y el SIndiCIT, garantizan la consistencia en la información gestionada en el sistema de CTI. La propuesta para lograr la interoperabilidad de las aplicaciones quedó abierta para integraciones futuras.

CONCLUSIONES

Considerando que en la bibliografía consultada no se reporta la existencia de un sistema libre que integre la mayor cantidad de procesos de CTI, se propuso el desarrollo de una aplicación que cumpliera con dicha especificación.

El análisis de los sistemas informáticos utilizados en la gestión editorial de la SC-UCI (*OJS*) y en la medición de la producción científica (sistema de indicadores cienciométricos SIndiCIT), demostró la necesidad de integrar ambos sistemas en la propuesta de solución y se definió además, la manera en que se logra dicha integración.

El desarrollo de una aplicación que integra en un sistema único los procesos claves de CTI, demuestra que se facilita y gestiona de manera consistente la información asociada a la actividad científica que se desarrolla en la UCI.

Su concepción, así como las tecnologías utilizadas en su desarrollo, se orientaron hacia lograr una alta flexibilidad y escalabilidad que permite la integración con otras aplicaciones.

RECOMENDACIONES

Profundizar en el tema de los *currículum vítae* con el objetivo de conformar el currículo de cada usuario a partir de la información recopilada en el sistema de gestión de CTI, así como a partir de la información que se pudiera obtener de otras aplicaciones utilizadas en las áreas de pregrado, postgrado y otras.

REFERENCIAS BIBLIOGRÁFICAS

1. Comisión de Política Científica, *Política Científica de la Universidad de las Ciencias Informáticas*, D.d. Investigaciones, Editor. 2010: La Habana.

2. BIREME, OPS, and OMS. *ScienTI. Red Internacional de Fuentes de Información y conocimiento para la Gestión de la Ciencia, Tecnología e Innovación*. [cited 2009 1 de octubre]; Available from: http://www.scienti.net.

3. *IV Reunión de Coordinación Regional de la Red ScienTI*. 2005 [cited 2009 6 de octubre]; Available from: http://bvs4.icml9.org/gt/scienti/presentation.php?lang=es.

4. BIREME/OPS/OMS, S.E., *La Red ScienTI: estado actual*. 2005: Salvador, Bahia.

5. CNPq. *Plataforma Lattes. A Plataforma Lattes*. [cited 2009 7 de octubre]; Available from: http://lattes.cnpq.br/conteudo/aplataforma.htm.

6. CNPq. *Plataforma Lattes. Acordos Institucionais*. [cited 2009 7 de octubre]; Available from: http://lattes.cnpq.br/conteudo/acordos.htm.

7. *CICYTAR. Sistema de Información de Ciencia y Tecnología Argentino*. [cited 2009 30 septiembre]; Available from: http://www.sicytar.secyt.gov.ar/.

8. *Instituto Colombiano para el Desarrollo de la Ciencia y la Tecnología COLCIENCIAS. CvLAC*. [cited 2009 1ro de octubre]; Available from: http://thirina.colciencias.gov.co:8081/scienti/jsp/cvlac.jsp.

9. *Instituto Colombiano para el Desarrollo de la Ciencia y la Tecnología COLCIENCIAS. Currículos*. [cited 2009 1ro de octubre]; Available from: http://thirina.colciencias.gov.co:8081/scienti/jsp/curriculo.jsp.

10. *Instituto Colombiano para el Desarrollo de la Ciencia y la Tecnología COLCIENCIAS. GrupLAC*. [cited 2009 1ro de octubre]; Available from: http://thirina.colciencias.gov.co:8081/scienti/jsp/gruplac.jsp.

11. *Instituto Colombiano para el Desarrollo de la Ciencia y la Tecnología COLCIENCIAS. InstituLAC*. [cited 2009 1ro de octubre]; Available from: http://thirina.colciencias.gov.co:8081/scienti/jsp/institulac.jsp.

12. *Secretaría Nacional de Ciencia y Tecnología. SENACYT*. [cited 2009 1ro de octubre]; Available from: http://cvlac.senacyt.gov.ec/curriculo.html.

13. *Sistema Integrado de Información sobre Investigación Científica y Tecnológica. SIICYT* [cited 2009 1ro de octubre]; Available from: http://www.siicyt.gob.mx/siicyt/quienesSomos.do?idenc=10#titulo5.

14. *Universidad de Barcelona. GREC*. [cited 2009 8 de octubre]; Available from: https://webgrec.ub.edu/.

15. Solís Cabrera, M. *El sistema de información científica de Andalucía, una experiencia pionera en España*. 7.

16. Sánchez, O.E., et al. *Sistema de Información para la Gestión de Programas de Ciencia e Innovación en Cuba*.

17. *CiPROCiT. Sistema de Información para la Gestión de Programas y Proyectos de Ciencia e Innovación Tecnológica*. 2008 [cited 2009 8 de octubre]; Available from: http://siprocit.redciencia.cu/.

18. Delgado Victore, R. (2003) *La dirección integrada por proyectos (project management) en el marco de la ciencia y la innovación tecnológica*.

19. Gulín González, J., D. Batard Lorenzo, and E. Mon Pérez. *Los indicadores de producción científica en la Universidad de las Ciencias Informáticas: ¿Cómo ponderar los resultados de las Ciencias Informáticas de forma diferenciada?* in *VII Congreso Iberoamericano de Indicadores de Ciencia y Tecnología*. 2007. São Paulo, Brasil.

20. Medina León, Y., *Sistema de indicadores para medir la ciencia en la UCI*. 2008, Universidad de las Ciencias Informáticas: Ciudad de La Habana. p. 110.

21. PKP Team. *Public Knowlegde Project*. [cited 2009 20 de octubre]; Available from: http://pkp.sfu.ca/espanol.

22. Conallen, J., *Building Web applications with UML*. 2da ed. 2003. 22.

23. Sommerville, I., *Ingenería del software*. 7ma ed. 2005, Madrid: Pearson Educación, S.A.

24. W3C. *World Wide Web Consortium*. [cited 2010 15 de noviembre]; Available from: http://www.w3c.es/.

25. *World Wide Web Consortium. Oficina Española. Guía Breve de XHTML*. 2008 [cited 2010 noviembre]; Available from: http://www.w3c.es/divulgacion/guiasbreves/XHTML.

26. Sawyer McFarland, D., *JavaScript: The Missing Manual*. 1ra ed. 2008: O´ Reilly Media, Inc.

27. Flanagan, D., *JavaScript. The Definitive Guide*. 5ta ed: O´ Reilly.

28. Garrett, J.J. (2005) *Ajax: A New Approach to Web Applications*.

29. Holzner, S., *jQuery. Visual QuickStart Guide*. 2009: Peachpit Press.

30. Project team of jQuery. *jQuery. Write less, do more.* 2010 [cited 2011; Available from: http://jquery.com/.

31. Lengstorf, J., *Pro PHP and JQuery*. 2010: Apress.

32. Harwani, B., *JQuery Recipes: A problem-Solution Approach*. 2010: Apress.

33. *jQuery. Sites using jQuery*. 2010 [cited 2011; Available from: http://docs.jquery.com/Sites_Using_jQuery.

34. Holzner, S., *The Complete Reference PHP*. 2008: The McGraw-Hill Companies.

35. The PHP Group. *PHP*. [cited 2010 15 de noviembre]; Available from: http://www.php.net/usage.php.

36. Vaswani, V., *PHP. A Beginner´s Guide*. 2009.

37. The PHP Group. *PHP ¿Qué se puede hacer con PHP?* [cited 2011 27 de abril]; Available from: http://www.php.net/manual/es/intro-whatcando.php.

38. Potencier, F. and F. Zaninotto, *Symfony 1.2, la guía definitiva*. 2008, Apress.

39. *BeezNest Open-Source specialists. Algunos framework para PHP más usados*. 2009 [cited 2011 6 de junio]; Available from: http://beeznest.wordpress.com/2009/01/14/algunos-frameworks-para-php-mas-usados/.

40. *Dream Css. 10 useful php framework from web*. 2009 [cited 2011 6 de junio]; Available from: http://www.dreamcss.com/2009/06/10-useful-php-framework-from-web.html.

41. *Mundo geek. Frameworks PHP*. 2010 [cited 2011 6 de junio]; Available from: http://mundogeek.net/archivos/2010/10/30/frameworks-php/.

42. *Symfony. Practical symfony. Día 20: Los Plugins*. Available from: http://www.symfony-project.org/jobeet/1_2/Propel/es/20.

43. *Symfony. Plugins*. [cited 2011 febrero]; Available from: http://www.symfony-project.org/plugins/.

44. Potencier, F., *Más con symfony*.

45. Potencier, F., *Más con symfony. Utilizando las tareas con un Crontab*.

46. Carrero, A. *Programación en castellano*. Conceptos básicos de ORM (Object Relational Mapping).

47. Potencier, F., *Más con symfony*, in *Utilizando la herencia de tablas de Doctrine (primera parte)*. 2010.

48. The PostgreSQL Global Development Group, *PostgreSQL 8.4. The SQL Language*. Vol. I. 2009.

49. *PostgreSQL-es. Sobre PostgreSQL*. 2010 [cited 2011 8 de junio]; Available from: http://www.postgresql.org.es/sobre_postgresql.

50. Douglas, K. and S. Douglas, *PostgreSQL. A comprehensive guide to building, programming, and administering PostgreSQL databases.* 1ra ed. 2003.

51. Quiñones, E. *PostgreSQLPE*. Introducción a PostgreSQL.

52. *Incutio. HttpClient - a PHP Web Client Class*. 2006 [cited 2011 mayo]; Available from: http://scripts.incutio.com/httpclient/.

53. *Incutio. HttpClient Manual*. 2006 [cited 2011 mayo]; Available from: http://scripts.incutio.com/httpclient/manual.php.

54. *PHP Classes. Class PHP HTTP protocol client* [cited 2011 junio]; Available from: http://www.phpclasses.org/package/3-PHP-HTTP-client-to-access-Web-site-pages.html.

55. *SourceForge. Snoopy*. [cited 2011 junio]; Available from: http://sourceforge.net/projects/snoopy/.

56. *Symfony. sfWebBrowserPlugin - 1.1.2. Standalone HTTP client*. 2009 [cited 2011 junio]; Available from: http://www.symfony-project.org/plugins/sfWebBrowserPlugin/1_1_2?tab=plugin_readme.

57. Web Services Architecture Working Group. *W3C. World Wide Web Consortium. Guía breve de servicios web*. 2010 [cited 2011 23 de junio]; Available from: http://www.w3c.es/divulgacion/guiasbreves/ServiciosWeb#intro.

58. *MDN. Servicios Web XML*. 2010 [cited 2011 23 de junio]; Available from: https://developer.mozilla.org/es/Servicios_Web_XML.

59. *Altova. WSDL Editor and Documentation Generator. XMLSpy 2011*. [cited 2011 25 de junio]; Available from: http://www.altova.com/xmlspy/wsdl-editor.html.

60. *Liquid Technologies. WSDL Editor.* 2011 [cited 2011 25 de junio]; Available from: http://www.liquid-technologies.com/wsdl-editor.aspx.

61. Ricardo Figueredo, R., *slSISaludPlugin, extensión de symfony para el desarrollo de aplicaciones en el Sistema de Información para la Salud*, in *Facultad 7*. 2010, Universidad de las Ciencias Informáticas: La Habana. p. 80.

62. Kerl, C., *ckWebServicePlugin*. 2008.

63. Potencier, F., *El tutorial Jobeet*. 2009. p. 309.

64. Jacobson, I., G. Booch, and J. Rumbaugh, *El Proceso Unificado de Desarrollo de Software*. 2da ed. 2000, Madrid: Addison Wesley. 464.

65. Smecher, A. (2008) *OJS Technical Reference*. 52.

66. The PHP Group. *PHP. SoapClient*. 2011 24 de junio de 2011 [cited 2011 julio]; Available from: http://www.php.net/manual/es/soapclient.soapclient.php.

67. Boehm, B.W. *University of Southern California*. Guidelines for Verifying and Validating Software Requerements and Design Specifications.

68. Pressman, R.S., *Ingenería de Software. Un enfoque práctico.* 5ta ed. 2002, Madrid, España: McGraw Hill. 642.

69. Dijkstra, E.W., O.J. Dahl, and C.A.R. Hoare, *Structured Programming*. 1972: Academic Press.