UNIVERSIDAD NACIONAL AUTÓNOMA

DE MÉXICO

FACULTAD DE CIENCIAS

Implementación y adecuación del Repositorio

Institucional del IISUE

REPORTE DE APOYO A

LA INVESTIGACIÓN

QUE PARA OBTENER EL TÍTULO DE:

Licenciado en Ciencias de la Computación

PRESENTA:

Pablo Enrique Olivares Paz

TUTORA

M. en I. Beatriz Peralta Cortés

1. Datos del alumno
   Olivares
   Paz
   Pablo Enrique
   5557751098
   Universidad Nacional Autónoma de México
   Facultad de Ciencias
   Licenciatura en Ciencias de la Computación
   092241655
2. Datos del tutor
   M. en I.
   Beatriz
   Peralta
   Cortés
3. Datos sinodal 1
   M. en C.
   María Guadalupe Elena
   Ibargüengoitia
   González
4. Datos sinodal 2
   M. en I.
   Karla
   Ramírez
   Pulido
5. Datos sinodal 3
   M. en I.
   Yasmine
   Macedo
   Reza
6. Datos sinodal 4
   L. en C.C.
   Magali Odalinda
   Morales
   Rojas
7. Datos del trabajo escrito
   Implementación y adecuación del Repositorio Institucional del IISUE
   90 páginas
   2019

A mis padres.

A mis hijos y esposa.

A mi familia.

En especial a la Familia Cruz Garrido†y a mi primo Beto†.

Agradecimientos

Agradezco a la Universidad Nacional Autónoma de México y a la Facultad de Ciencias por sus conocimientos que hoy son un factor clave en mi vida profesional.

Agradezco al personal académico y administrativo que labora en el Instituto de Investigaciones sobre la Universidad y la Educación, en especial a los miembros del Departamento de Cómputo y la Coordinación de la Biblioteca por invitarme a participar en el proyecto Conacyt 286975 como becario, además de brindarme su apoyo académico y moral para llevar a buen fin este trabajo.

De manera especial mi gratitud a la Mtra. Beatriz Peralta por la confianza depositada, además de brindarme su amistad, me acompañó en la realización de este escrito como asesora. Le agradezco la oportunidad de participar en este proyecto y el apoyo total para alcanzar la meta.

Asimismo, estoy muy agradecido con cada una de las sinodales, las maestras Lupita Ibargüengoitia, Karla Ramírez, Yasmine Macedo y la licenciada Magali Morales por la lectura, los comentarios y las correcciones para mejorar y enriquecer este trabajo.

También expreso mi gratitud al Mtro. Sergio Arreguín Meneses y a mi colega Don Arturo Mena Barraza por compartir sus conocimientos y apoyo en las distintas actividades de configuración del repositorio.

Índice general

Índice de figuras

Índice de códigos

Capítulo 1

Introducción

En este capítulo se describe como fue que el Instituto de Investigaciones Sobre la Universidad y la Educación (IISUE) dió pie a su participación para la puesta en marcha de un Repositorio Institucional, también se expone el objetivo a cumplir en las actividades de configuración, necesarias para el desarrollo del repositorio.

1.1. Antecedente

El Instituto de Investigaciones Sobre la Universidad y la Educación (IISUE, UNAM) mantiene un esquema tradicional en el manejo y difusión de su información respecto a sus investigaciones y promoción de trabajo.

Actualmente ante las nuevas herramientas para la digitalización y almacenamiento

de la información, así como la reciente iniciativa gubernamental para promover a partir de recursos públicos el Acceso Abierto (del término en inglés *Open Access*), esta ha establecido una estrategia de alineamiento con estas nuevas tendencias tecnológicas como parte del proyecto "Toda la UNAM en línea", a fin de estar a la vanguardia y en condiciones de dar visibilidad a las investigaciones realizadas por los académicos del instituto, así como brindar el intercambio con instituciones a nivel nacional, internacional y al público en general apegándose conforme a las disposiciones de acceso abierto. En este sentido, el IISUE se vio beneficiado con el proyecto CONACyT 286975 a través de la "Convocatoria 2016 para desarrollar los repositorios institucionales de acceso abierto a la información científica, tecnológica y de innovación". Con base en dicha convocatoria se está trabajando en el repositorio institucional para el IISUE, en colaboración con la Coordinación de la Biblioteca y el Departamento de Cómputo. En la fase inicial, el Instituto llevó a cabo las siguientes labores:

- Definición de topología documental.
- Políticas de conversión de los recursos de información en objetos digitales.
- Políticas de metadatos.
- Instalación básica de un repositorio utilizando software libre, *DSpace*.

1.2. Objetivo del proyecto.

Apoyar en las actividades de integración a nivel de configuración, para adecuar el funcionamiento del Repositorio Institucional (RI), conforme a las características solicitadas por el CONACyT a través de las normas establecidas en el documento *Lineamientos Específicos para Repositorios*(CONACyT, 2017a), con mayor énfasis, en esta primera fase:

- Asegurar la integridad del registro de los elementos al repositorio, mediante la inclusión de los catálogos provistos por el CONACyT.
- Habilitar los servicios de consulta respectivos, para proporcionar información estadística correspondiente a la búsqueda y descarga del número de elementos, al igual que el total de búsquedas por autores e información de los depositarios.
- Realizar modificaciones puntuales a la presentación de algunos datos de información y la elaboración de la memoria técnica respecto a las configuraciones elaboradas en este trabajo para el respectivo mantenimiento.

El proyecto se encaminó a una solución que aproveche las herramientas sugeridas por el CONACyT, específicamente *DSpace*, así como complementar con otras tecnologías para cubrir los requerimientos de configuración y estar en condiciones de poner en marcha el Repositorio Institucional.

En el presente capítulo se describió el fundamento de la colaboración del IISUE en el ámbito de los repositorios institucionales. Se listaron las primeras tareas concluidas y se

presentaron aquellas que conforman el alcance del proyecto, por ejemplo la configuración requerida para la puesta en marcha del Repositorio Institucional del IISUE.

Capítulo 2

Repositorios digitales

En este capítulo se hace una introducción a los repositorios digitales, precisando en las particularidades de *DSpace*. También se describe el entorno del CONACyT respecto a su labor en la difusión de aquellos trabajos desarrollados con recursos públicos, finalizando con una reseña de las etapas que formaron parte de la puesta en marcha del repositorio del IISUE.

2.1. ¿Qué son los repositorios digitales?

La palabra repositorio se deriva del latín *repositorium* que significa armario o alacena, connotación que posteriormente fue generalizada por la Real Academia Española como *"Lugar donde se guarda algo"* (Academia, 2019), siendo este el punto de referencia para

identificar aquellos medios que permiten depositar la información digital.

Un repositorio digital está conformado por software y hardware que permite almacenar un conjunto de objetos digitales, garantizar la identificación de cada elemento digital almacenado así como brindar la gestión para cada uno, facilitando el acceso controlado y estandarizado a cada objeto además de tener la condición para la interoperabilidad en el intercambio de información con otros sistemas.

La implementación de repositorios digitales es una de las formas que las universidades disponen para minimizar la falta de visibilidad de su producción intelectual (Silva and Tomaél, 2012).

La clasificación de un repositorio se define conforme al manejo de sus contenidos, sin embargo se aluden dos principales tipos: institucionales y temáticos.

- Repositorios institucionales

  Almacenan, preservan y dan acceso al material intelectual producido por los miembros de una institución. Algunos ejemplos de este tipo son los siguientes:

  - Instituto Politécnico Nacional

    https://www.repositoriodigital.ipn.mx/

  - Universidad Autónoma Metropolitana, campus Azcapotzalco

    http://zaloamati.azc.uam.mx/

  - Universidad Veracruzana

https://cdigital.uv.mx/

- Universidad Autónoma de Zacatecas

  http://ricaxcan.uaz.edu.mx/

- Universidad Autónoma de Chiapas

  http://repositorio.unach.mx/jspui/

- Universidad de Guadalajara

  https://wdg.biblio.udg.mx/index.php/repo-inv

- Repositorios temáticos

Almacenan, preservan y dan acceso a los contenidos de una disciplina o área temática. Algunos ejemplos de este tipo son los siguientes:

- Instituto Nacional de Bellas Artes

  http://todocultura.mx/detalle/repositorio-inba-digital

- ArXiv

  Particularmente este repositorio, es uno de los pioneros en impulsar el libre acceso a la información, difundiendo la importancia del acceso e intercambio de información entre los distintos grupos de trabajo interdisciplinarios.

  https://arxiv.org/

Entre los principales contenidos almacenados por estos repositorios se encuentran:

- Científicos.

  Tesis, ponencias, *preprints*, materiales audiovisuales y revistas de divulgación.

- Institucionales.

  Revistas de información institucional, reglamentos y normas.

- Académicos.

  Guías de estudio, apuntes de clase, material audiovisual, guías y exámenes en línea.

El desarrollo de repositorios institucionales inició en el año 2002, como una nueva estrategia que les permitió a las universidades asumir el papel de editoras, modernizando los procesos de publicación y divulgación de la producción académica en contenido digital (Lynch, C. A., 2003).

Conforme a lo comentado por Facundo G. Adorno (Adorno, 2018), los repositorios institucionales son herramientas para impulsar la corriente del Acceso Abierto a la información.

El Acceso Abierto es un movimiento impulsado por la iniciativa Acceso Abierto de Budapest (del inglés *Budapest Open Access Initiative*), la cual trabajó en hacer difusión, promover estrategias para hacer una realidad este modo de cooperación y efectuó un llamado a todas la instituciones en distintos países a favor de este movimiento. El Acceso Abierto tiene como premisa compartir libremente la información, sin barreras que limiten a ciertos grupos de personas, contrarrestando parte de lo que empezó a ser un obstáculo para tener

acceso a la información; como lo es el alto costo de las publicaciones. De lo anterior deriva la implementación de software para cubrir dicho propósito, para fines de este proyecto se usó la plataforma *DSpace*.

2.1.1. *DSpace*

DSpace es un software de código abierto, su primera versión surge en 2002, resultado de un proyecto de colaboración entre *Massachusetts Institute of Technology Library* y *Hewlett Packard Labs*.

DSpace se ha consolidado como un repositorio apegado a la mayoría de los estándares para el registro y administración de objetos digitales, de la misma manera cumple con lo necesario para establecer una interoperabilidad entre repositorios, todo esto gracias a las mejoras por parte de los distintos grupos de contribución a lo largo de estos últimos 17 años.

A continuación se listan las cualidades de *DSpace* que se tomaron en consideración para su aplicación en la implementación del Repositorio Institucional IISUE (RI IISUE).

- Software bajo la Iniciativa para el Código Abierto.

  En el año 2018, la *Open Source Initiative* (OSI, en español Iniciativa para el Código Abierto) celebró su 20^o aniversario, de modo que, se aprovecha parte de su propuesta en el sentido de ajustar esta distribución de *DSpace* conforme a las necesidades del

Instituto de Investigaciones Sobre la Universidad y la Educación (IISUE).

- Un software probado.

  Demostrado por figurar durante estos últimos años que han transcurrido a partir de su primera versión; el aporte que ha brindado *DSpace* como herramienta respecto a la implementación y operación de un repositorio digital institucional a los distintos centros académicos y/o de investigación, brinda una seguridad respecto a la estabilidad del servicio que se tiene planeado proporcionar. Según el Directorio de Repositorios de Acceso Abierto OpenDOAR, por sus siglas en inglés *Directory of Open Access Repository* (OpenDOAR, 2019) *DSpace* registra una presencia del 45 % como plataforma para la implementación de repositorios de acceso abierto (ver Figura 2.1).

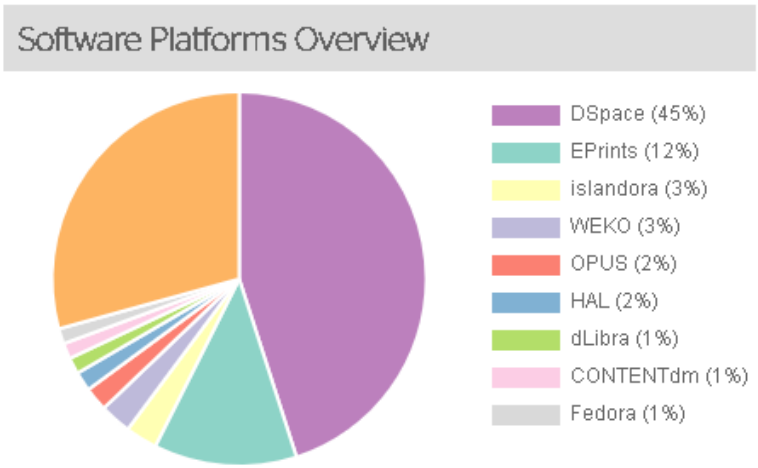

Figura 2.1: Plataformas disponibles (OpenDOAR, 2019).

- Soporta diversos formatos de archivos (Carlos André Rosa and Domingues, 2017).

Tipo	Formato	Descripción
Dato	XML	*eXtensible Markup Language*
	HTML	*HyperText Markup Language*
	PDF	*Portable Document Format*
	RTF	*Rich Text Format*
	XSL	*eXtensible Stylesheet Language*
	TXT, MS Word, Power Point, Open Office, DOCX y DOC	Identifica aquellos archivos que derivan de las herramientas convencionales de escritorio.
Imagen	BMP	*BitMap*
	JPEG	*Joint Photographic Experts Group*
	GIF	*Graphics Interchange Format*
	JPEG	*Joint Photographic Experts Group*
	PNG	*Portable Network Graphics*
	TIFF	*Tagged Image File Format*
	TGA	*Truevision Graphics Adapter*

Tipo	**Formato**	**Descripción**
	RAW	Del significado *crudo* dado que contiene la totalidad de los datos de la imagen tal y como ha sido captada.
Video	AVI	*Audio Video Interleave*
	FLV	*FLash Video*
	MOV	Denota formato de contenido multimedia
	MPEG	*Moving Picture Experts Group*
	SWF	*Small Web Format*
	WMV	*Windows Media Video*
Audio	AC3	*Audio Coding 3*
	AIFF	*Audio Interchange File Format*
	MP3	*Music file (MPEG Layer 3)*
	WAV	*Waveform Audio Format*
	WMA	*Windows Media Audio*

- Soporta los principales estándares en metadatos y protocolos de interoperabilidad (Carlos André Rosa and Domingues, 2017).

  - Los metadatos juegan un papel importante ya que como lo comenta Adorno (2018, p.6) describen y brindan información respecto a los objetos digitales almacenados. *DSpace* tiene la capacidad de manejar los siguientes estándares de metadatos:

    - Dublin Core, especificación desarrollada por la *Dublin Core Metadata Initiative* (DCMI, 2019).
    - MARC por sus siglas en inglés *MAchine-Readable Cataloging*, especificación desarrollada por la *Library Congress US* (Library of Congress, 2006).
    - PREMIS por sus siglas en inglés *PREservation Metadata: Implementation Strategies*, especificación desarrollada por la *Online Computer Library Center* (OCLC) y la *Research Libraries Group* (RLG) (Group, 2005).
    - METS por sus siglas en inglés *Metadata Encoding & Transmission Standard*, especificación desarrollada por la *Digital Library Federation* (DLF) (Federation, 2010).
    - MODS por sus siglas en inglés *Metadata Object Description Schema*, especificación desarrollada por la *Network Development and MARC Standards Office Library of Congress US* (Library of Congress, 2018).

- DIDL por sus siglas en inglés *Digital Item Declaration Language*, especificación desarrollada por la *Motion Pictures Experts Group* (Jeroen Bekaert, 2003).

- Cubre los principales protocolos de interoperabilidad:
  - OAI por sus siglas en inglés *Open Archives Initiative*, iniciativa que se encarga del desarrollo de estándares para diseminar la información entre repositorios.
  - OAI-PMH por sus siglas en inglés *Open Archives Initiative Protocol for Metadata Harvesting*, provee la interoperabilidad para la cosecha de metadatos donde existe un *Proveedor de datos* que soporta el protocolo OAI-PMH, un *Proveedor de Servicios* aquel que genera servicios apartir de los metadatos cosechados vía OAI-PMH y un *Cliente* aquel componente usado por el *Proveedor de Servicio*, que se encarga de hacer la cosecha de metadatos sobre aquellos repositorios que dispongan del protocolo OAI-PMH.

- Otras características.

  Se aprovecha su configuración multi-idioma y compatibilidad con *Linux*[1], además está implementado en *Java*[2] lo que garantiza la existencia de diversos foros de apoyo,

[1]Sistema operativo constituido por varios programas fundamentales que necesita la computadora para poder comunicar y recibir instrucciones de los usuarios; tales como leer y escribir datos en el disco duro, cintas, e impresoras; controlar el uso de la memoria; y ejecutar otros programas (Debian, 2015).

[2]Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems (Corporation, 2019).

ya sea en video tutoriales o por documentación.

Con base en lo mencionado por Adorno (2018, p.8) el crecimiento de los repositorios digitales ha ido en aumento sobre todo en la promoción del acceso abierto a la información. Lo cual deja entrever la tendencia e importancia que se va dando al promover el acceso a contenidos digitales e intercambio de información, como se puede observar en la Figura 2.2.

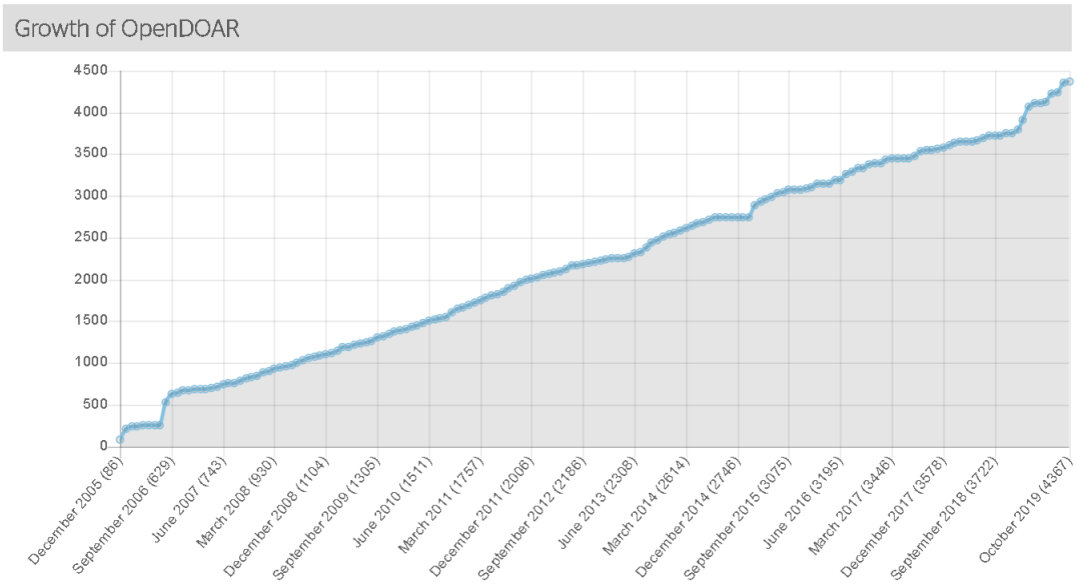

Figura 2.2: Progreso en la implementación de repositorios (OpenDOAR, 2019).

2.2. CONACyT y la Ciencia Abierta

"*El Consejo Nacional de Ciencia y Tecnología fue creado por disposición del H. Congreso de la Unión el 29 de diciembre de 1970, como un organismo público descentra-*

lizado de la Administración Pública Federal, integrante del Sector Educativo, con personalidad jurídica y patrimonio propio.

...

La misión del CONACyT es impulsar y fortalecer el desarrollo científico y la modernización tecnológica de México, mediante la formación de recursos humanos de alto nivel, la promoción y el sostenimiento de proyectos específicos de investigación y la difusión de la información científica y tecnológica "(CONACyT, 2014b).

Ciencia abierta se denomina a la práctica de aprovechar y hacer llegar la producción de trabajos científicos, que se han elaborado con recursos o infraestructura pública, para su aprovechamiento hacia otros grupos interdisciplinarios de distintas instituciones dentro del país, apoyando y fortaleciendo las actividades de investigación y enseñanza de distintas áreas de conocimiento (CONACyT, 2017b).

Para lograr este acceso y acercamiento el CONACyT trabaja los siguientes ejes:

- Editorial.

  Apoyar la clasificación, elaboración y publicación de revistas referente a trabajos, proyectos e iniciativas de investigación (CONACyT, 2018b).

- CONRICyT (Consorcio Nacional de Recursos de Información Científica y Tecnológica)

  En el mes de septiembre de 2010, "*... la Secretaría de Educación Pública (SEP),*

el Consejo Nacional de Ciencia y Tecnología (CONACyT), la Asociación Nacional de Universidades e Instituciones de Educación Superior (ANUIES), la Universidad Autónoma Metropolitana (UAM); la Universidad Nacional Autónoma de México (UNAM), el Instituto Politécnico Nacional (IPN), el Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional (CINVESTAV), la Universidad de Guadalajara (UdeG) y la Corporación Universitaria para el Desarrollo de Internet (CUDI), firmaron un Convenio de Colaboración para Constituir formalmente el Consorcio Nacional de Recursos de Información Científica y Tecnológica (CONRICyT); y de esta manera, ampliar y agilizar el acceso a la información científica -a través de bases de datos y revistas científicas reconocidas a nivel mundial- en las Instituciones de Educación Superior (IES) y Centros de Investigación del país" (CONRICyT, 2018).

- Comunicación Pública de la Ciencia

  Derivado de un estudio de sondeo en la sociedad mexicana, el CONACyT se percata que a la población le resulta importante el tema de ciencia y tecnología. Sin embargo, identifica que un número considerable de personas, no tienen un panorama claro en cuanto al conjunto de actividades que estas áreas involucran, por lo cual el CONACyT procede a implementar estrategias de difusión entre las cuales se encuentran principalmente (CONACyT, 2014a):

  - Semana Nacional de Ciencia y Tecnología.
  - Revista Ciencia y Desarrollo.
  - Revista Ciencia y Desarrollo suplemento Helix para niños.

- Sistema Integrado de Información sobre Investigación Científica, Desarrollo Tecnológico e Innovación (Siicyt)

  "*El Siicyt integrará los esfuerzos de diferentes instituciones educativas, centros de investigación, organismos públicos, empresas y personas físicas y morales del sector público y privado, a fin de promover el desarrollo y la vinculación de la ciencia básica y la innovación tecnológica, así como convertir a la ciencia y la tecnología en un elemento fundamental de la cultura general de la sociedad.*"(Siicyt, 2015).

- Programa de Conectividad

  Apoyar para que puedan existir las condiciones de interconexión de comunicación con las distintas instituciones públicas y/o privadas de investigación a fin de establecer los canales necesarios para el intercambio de información (CONACyT, 2018b).

- Programa de Repositorios de Información

  Iniciativa (CONACyT, 2017a) que promueve el CONACyT con la finalidad de impulsar, facilitar y garantizar el acceso a la información referente a los trabajos de investigación por medio de la implantación de un Repositorio Nacional (RN), pla-

taforma digital cuya función principal es fungir como núcleo receptor y punto de diseminación a nivel nacional a partir de la alimentación provista por todas aquellas unidades de investigación públicas o privadas que participen por medio de la puesta en marcha de su propio Repositorio Institucional (RI) (ver la Figura 2.3). Tanto el Repositorio Nacional como los institucionales asegurarán el acceso a sus contenidos de información sin existir impedimento financiero, legal y técnico, sólo se considerará aquellos que surjan por la misma naturaleza del Internet.

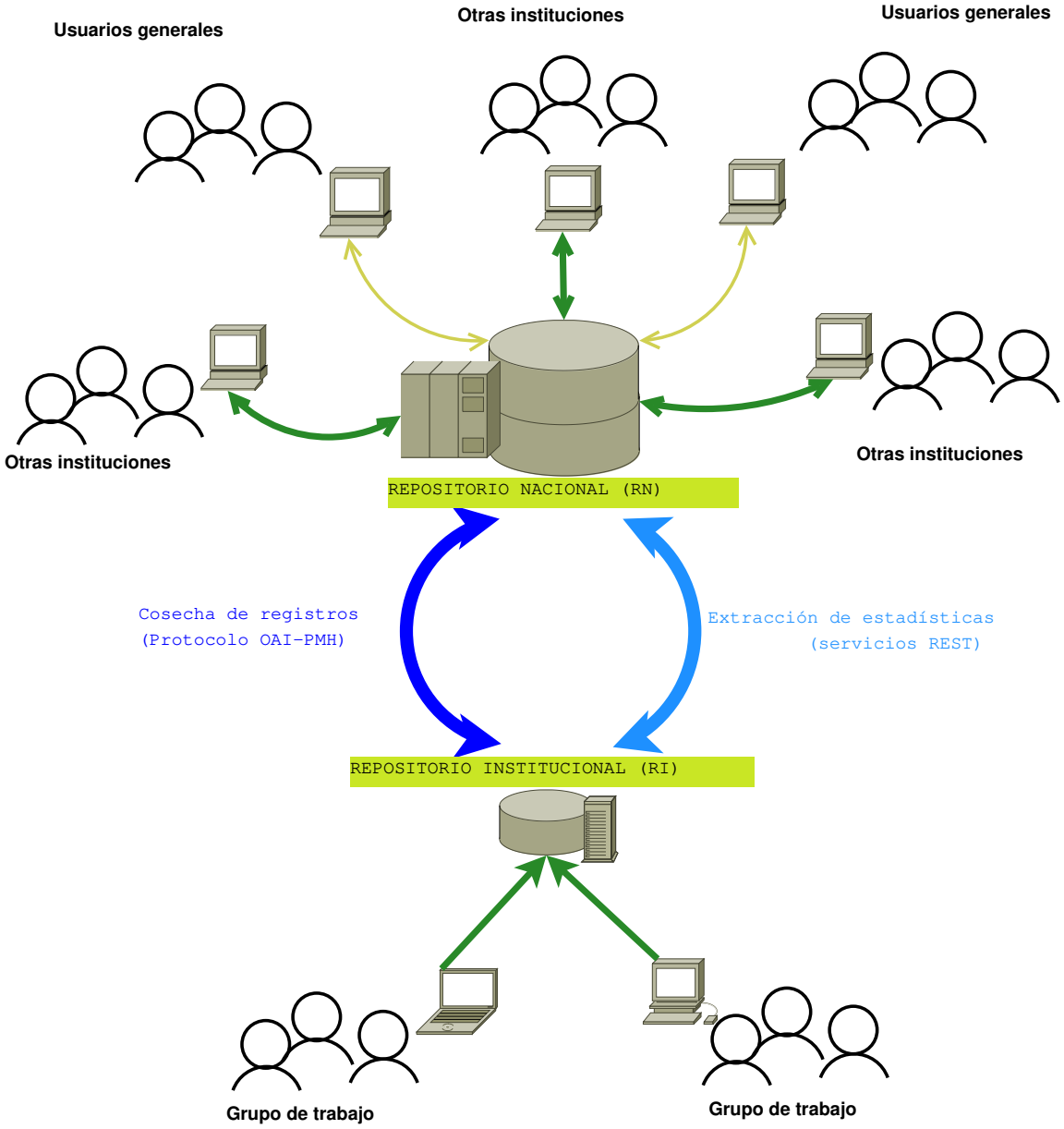

Figura 2.3: Interacción del Repositorio Nacional con un Repositorio Institucional.

2.3. Repositorio Nacional (RN)

Derivado del programa Repositorios de Información, cuyo punto de partida inicia en 2014 con las reformas a las leyes en Ciencia y Tecnología, seguido de la expedición en 2015 de los lineamientos generales y técnicos del Repositorio Nacional e Institucional. Complementando con las primeras dos convocatorias 2016 y 2017 para el desarrollo de repositorios institucionales y su integración al Repositorio Nacional.

El Repositorio Nacional actualmente se puede acceder a través de la URL, por sus siglas en inglés *Uniform Resource Locator*[3]:

https://www.repositorionacionalcti.mx/

2.3.1. Contenidos de información

Se consideran los siguientes contenidos de información disponibles tanto en el repositorio nacional e institucionales que salvo a guarda por derechos de autor o temas de seguridad nacional sea confidencial o reservada:

- "*Publicaciones científicas, comprende todo el universo de publicaciones resultado de la investigación. Dentro de estas se incluyen: artículos, libros, capítulos de libros, tesis o trabajos para la obtención de grado, documentos presentados en conferencias nacionales e internacionales y otros materiales enfocados en la producción de*

[3]Última consulta en mayo 2019.

conocimiento;

- *Productos del desarrollo tecnológico y la innovación, tales como patentes, desarrollos tecnológicos, innovaciones, transferencias tecnológicas, creación o mejora de prototipos, productos, procesos o servicios, o diagnósticos tecnológicos dirigidos al estado de la tecnología;*

- *Datos primarios de las investigaciones, comprende toda aquella información recolectada y utilizada para la investigación académica, científica, tecnológica y de innovación, además de ser aquella comúnmente aceptada por la comunidad científica como un elemento necesario para validar los resultados de las investigaciones.*"(CONACyT, 2018b, p.11).

2.3.2. Esquema de alimentación de Contenidos Digitales

El Repositorio Nacional (RN) y los repositorios institucionales lograrán el intercambio de los contenidos de información por medio de la aplicación correspondiente a la estructura de metadatos provista por el CONACyT y la integración del conjunto de catálogos. La función de estos catálogos es homogenizar los distintos tipos de identificadores o clasificadores que pudieran aplicarse al momento de capturar cualquier contenido de información digital en cada Repositorio Institucional.

El Repositorio Nacional cuenta con un componente llamado Metabuscador (CO-

NACyT, 2018a), que es un conjunto de funciones encargadas de realizar principalmente las siguientes tareas:

- Llevar a cabo la cosecha a los distintos repositorios institucionales para importar todos aquellos contenidos que cumplan con las defincciones de metadatos establecidos. Esta cosecha la realizará por medio de la explotación del protocolo OAI-PMH.

- Recabar información sobre las estadísticas de uso a cada repositorio, conectándose a cada Repositorio Institucional para extraer información sobre el número de materiales consultados, descargados, al igual que el número de autores consultados e información sobre los usuarios que alimentan de información al Repositorio Institucional correspondiente.

2.4. El IISUE y su participación en la convocotaria 2017

Nuestra Máxima Casa de Estudios (UNAM) ya cuenta con Repositorios Universitarios (RU), de hecho se ha formado la red de Repositorios Universitarios digitales de la UNAM [4] que constituye parte de los esfuerzos colectivos para administrar y diseminar los materiales digitales producidos por la comunidad de académicos de la UNAM. Contribuye y es parte del programa institucional “Toda la UNAM en línea”. De esta forma, la red de Repositorios Universitarios complementa y fortalece un esfuerzo a nivel universitario para

[4] http://www.rad.unam.mx, última consulta junio 2019.

mejorar la presencia de la UNAM en Internet.

En este sentido, el Instituto de Investigaciones sobre la Universidad y la Educación siendo parte integral de la misma no puede mantenerse al margen de las actividades que se van desarrollando día a día dentro de la universidad. Una de estas es el acceso abierto de materiales biblio-hemerográficos con los que cuenta la biblioteca, para ello es importante participar como dependencia en un proyecto orientado en la administración, procuración y diseminación de recursos digitales. Por ende participa en la convocatoria "Convocatoria 2017 para desarrollar repositorios institucionales de Ciencia Abierta".

2.4.1. Etapas del proyecto

Para contextualizar la etapa que dió lugar a la contribución descrita en el presente reporte, se menciona en general que el proyecto consistió de cinco fases (ver la Figura 2.4), en la primera se trabajó en el registro y aceptación del proyecto, en la segunda tuvo espacio un taller de presentación y orientación referente al campo de repositorios digitales, introduciendo a los equipos participantes a las herramientas, protocolos y tecnologías disponibles en la actualidad para su aplicación en el desarrollo del proyecto. La tercer fase consistió en la adquisición de recursos y la conformación del equipo de trabajo necesario para el desarrollo del Repositorio Institucional. La cuarta fase dió pie a solventar la necesidad de realizar las adecuaciones a la herramienta *DSpace* para cumplir con lo establecido en el documento de lineamientos específicos para repositorios, que permita al Repositorio

Institucional la interoperabilidad con el Repositorio Nacional primordialmente atendiendo los puntos D y E (CONACyT, 2017a, p.9-10):

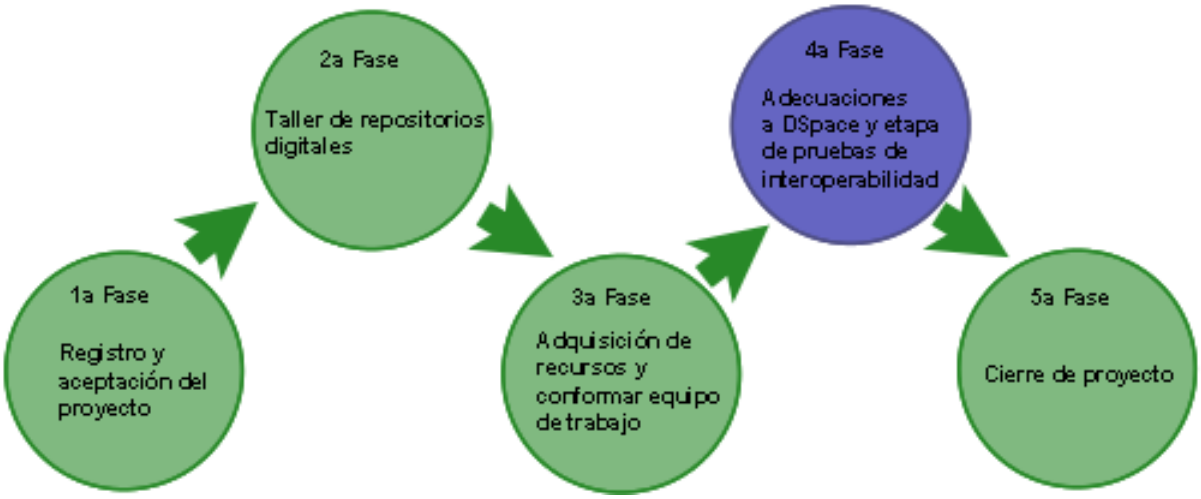

Figura 2.4: Fases correspondientes al desarrollo del proyecto.

D. *A nivel de estadísticas y datos de uso*. Permitirá la agregación e intercambio de los datos y la información sobre el uso de los materiales desde diferentes repositorios y sistemas de información.

E. *A nivel de catálogos*. La identificación y nombramientos de los autores, elementos, ubicación de los elementos, instituciones, agencias de financiamiento, proyectos, etcétera, serán consistentes dentro del proceso de organización de los materiales de los repositorios.

De forma adicional se realizaron las pruebas de cosecha de metadatos y se validó el acceso a los servicios de estadística. En el cierre del proyecto o etapa final se aprueba la conclusión del proyecto donde el CONACyT entregó al IISUE la carta de aceptación correspondiente.

En este capítulo se definió lo que es un repositorio digital, se mencionaron las cualidades principales de la herramienta *DSpace*, que la hacen sobresalir en su ámbito. Enseguida se presentó el contexto referente a la labor del CONACyT por impulsar el acceso abierto a la información, para asentar los motivos que encaminaron al IISUE a sumarse a este esfuerzo. Por último se realizó un breve recuento de las etapas desarrolladas para lograr el funcionamiento del Repositorio Institucional del IISUE.

Capítulo 3

Configuración y adecuación

En este capítulo se describen las actividades realizadas para satisfacer los lineamientos requeridos por el CONACyT, a fin de poner en operación el Repositorio Institucional del IISUE.

3.1. Configuración para la interoperabilidad entre el Repositorio Institucional del IISUE y el Repositorio Nacional

Como se establece en el documento de lineamientos (CONACyT, 2015), la interoperabilidad se define como la capacidad de intercambiar información entre el Repositorio

Nacional y los repositorios institucionales. Los puntos primordiales que debe cumplir el Repositorio Institucional para lograr dicho intercambio son:

- Manejar el protocolo OAI-PMH, necesario para lograr la cosecha de metadatos.

- Tener soporte para un esquema de metadatos conforme a lo propuesto por OpenAIRE 3.0.

  Por sus siglas en inglés OpenAIRE *(Open Access Infrastructure for Research in Europe)* iniciativa Europea cuya finalidad es promover la Ciencia Abierta, dentro del esfuerzo que lleva a cabo para garantizar el intercambio y distribución de información entre repositorios, implementa guías con las recomendaciones a cubrir destacando los siguientes puntos (Barrueco Cruz et al., 2017):

  1. Sugiere y define características referente a como implementar una cosecha a través del protocolo OAI-PMH.
  2. Marca y define los elementos Dublin Core obligatorios y recomendados para la descripción de los recursos.

  Durante los últimos años han elaborado las guías *Guidelines OpenAIRE 1.0*, *Guidelines OpenAIRE 2.0*, *Guidelines OpenAIRE 3.0* y está en curso la elaboración de *Guidelines OpenAIRE 4.0*.

- Integrar el uso de los catálogos provistos por el CONACyT.

- Implementar o habilitar servicios para consultar la estadística de usabilidad del repositorio.

DSpace como parte de su madurez[1] en el campo de los repositorios digitales, permite cubrir los puntos mencionados, los primeros dos ya son parte de su estructura estándar consolidada al paso del tiempo. Los dos puntos restantes se tuvieron que trabajar aparte debido a la peculiaridad de la solicitud. Sin embargo, gracias a la flexibilidad de *DSpace* como código abierto facilitó la realización de los cambios, mismos que se describen a continuación:

- Se inicia exponiendo el trabajo efectuado para integrar los catálogos provistos por el CONACyT.

- Se concluye detallando las actividades realizadas para generar los servicios referente a la estadística de usabilidad del repositorio.

3.2. Estructura de *DSpace*

En primer instancia se consultó la documentación referente a los componentes que integran a *DSpace* para conocer sus características y tener un panorama respecto a cuales

[1] Por *madurez* se considera la persistencia que tiene el software a través de los años. Adicional a lo anterior un software se considera que ha madurado cuando presenta estabilidad, liderazgo en su campo, cuenta con el respaldo de una comunidad de trabajo, cuenta con documentación consistente, existe soporte para los usuarios finales y presenta liberaciones regulares que incorporan nuevas funcionalidades (Cherukodan Surendran, 2014).

se tendrían que modificar. Si bien no es parte del alcance del presente reporte describir en su totalidad la estructura del *DSpace*, se considera importante mencionar los principales módulos que se exploraron, donde sobresalen JSPUI y REST, por sus siglas en inglés *Java Server Pages User Interface* y *REpresentational State Transfer* ya que con ellos se trabajó directamente para cubrir la necesidad respecto a la integración de catálogos e implementar los servicios de estadísticas, como se puede observar en la Figura 3.1.

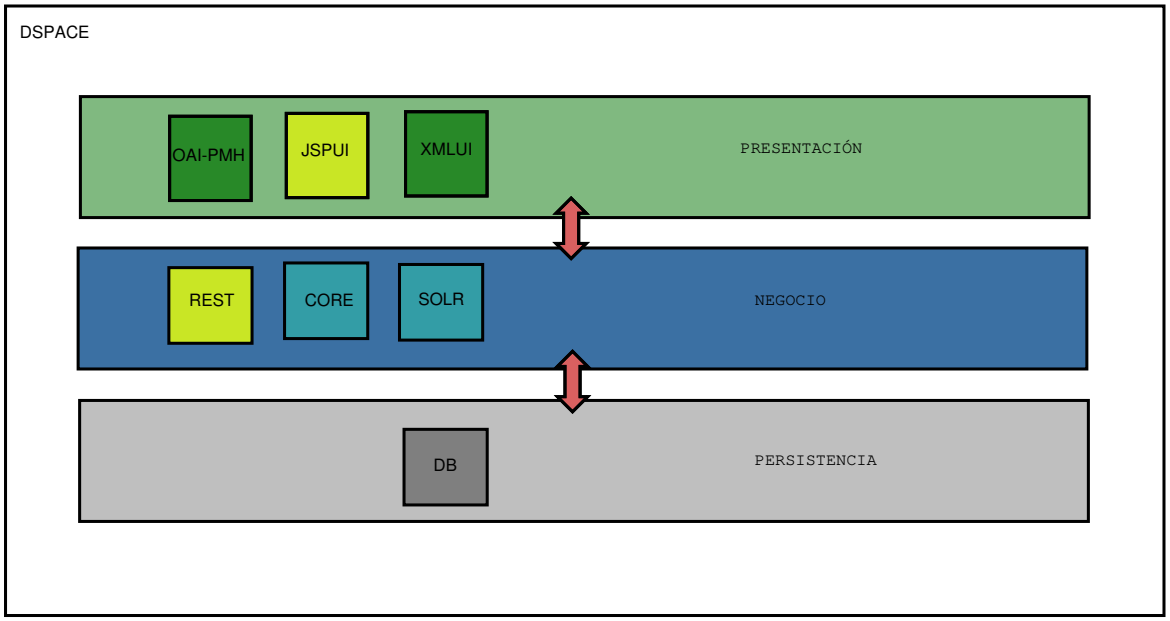

Figura 3.1: Módulos explorados en *DSpace*.

JSPUI denota aquel módulo localizado en la capa de presentación encargado de ofrecer todo el ambiente gráfico para la interacción con el usuario, está implementado mediante la tecnología JSP[2], por sus siglas en inglés *Java Server Pages*.

[2]Tecnología que permite del lado del servidor añadir contenido dinámico a las páginas *web* antes de enviarlas al navegador que las solicita (IBM, 2019).

REST es el módulo que permite habilitar aquellos servicios bajo el paradigma de *Transferencia de Estado Representacional* donde la comunicación entre el cliente y el servidor se reduce a una petición vía HTML, asegurando una independencia entre ambos.

DSpace está estructurado en tres capas, la primera corresponde a la parte de vista, aquí residen sus dos principales módulos que integran las interfaces gráficas JSPUI y XMLUI.

XMLUI por sus siglas en inglés *(eXtensible Markup Language User Interface)* es el módulo que está constituido por tres capas interconectadas, la capa de *Aspectos* que se encarga de extraer en formato XML del contenido del repositorio, la capa de *Temas* la cual se encarga de construir la estructura que tendrá cada página y por último la capa *Estilo* que se encarga de manejar la presentación que se le da a las páginas *(look and feel)* (De Giusti Marisa Raquel, 2012).

Las dos interfaces JSPUI y XMLUI convergen a la misma fuente de información, su diferencia consiste respecto a la flexibilidad que ofrece cada una de sus tecnologías para personalizar la presentación de la información, así como la complejidad que cada una demanda.

En el primer acercamiento que se tuvo con *DSpace*, los coordinadores del proyecto, se ambientaron a la interfaz JSPUI e incluso un primer esfuerzo derivó en un manual de operación para el equipo de bibliotecarios, encargados de proceder con la primera carga de información al repositorio, se ilustró sobre dicha interfaz.

Por lo anterior se decidió realizar los ajustes de integración de los catálogos del CONACyT sobre el módulo mencionado, mismos que se describen más adelante.

En la siguiente capa de negocio se identifican los módulos REST y *Solr*. El módulo *Solr* provee un motor de indización mediante el cual, se pueden hacer búsquedas referente a la actividad operativa del repositorio. Por último se menciona la capa de persistencia donde el principal componente es la base de datos donde se lleva a cabo el almacenamiento de toda la información que integra al repositorio.

3.3. Integración de los Catálogos del CONACyT

En seguimiento al apartado *DECIMO CUARTO.CATÁLOGOS* del documento Lineamientos Específicos para Repositorios (CONACyT, 2017a). En esta primera fase de implementación del Repositorio Institucional se eligieron los catálogos a trabajar:

- Catálogo de Autores.
- Catálogo de Área de conocimiento.
- Catálogo de Colaboradores.
- Catálogo de Resultado Científico.

La integración de estos catálogos se realizó en dos partes:

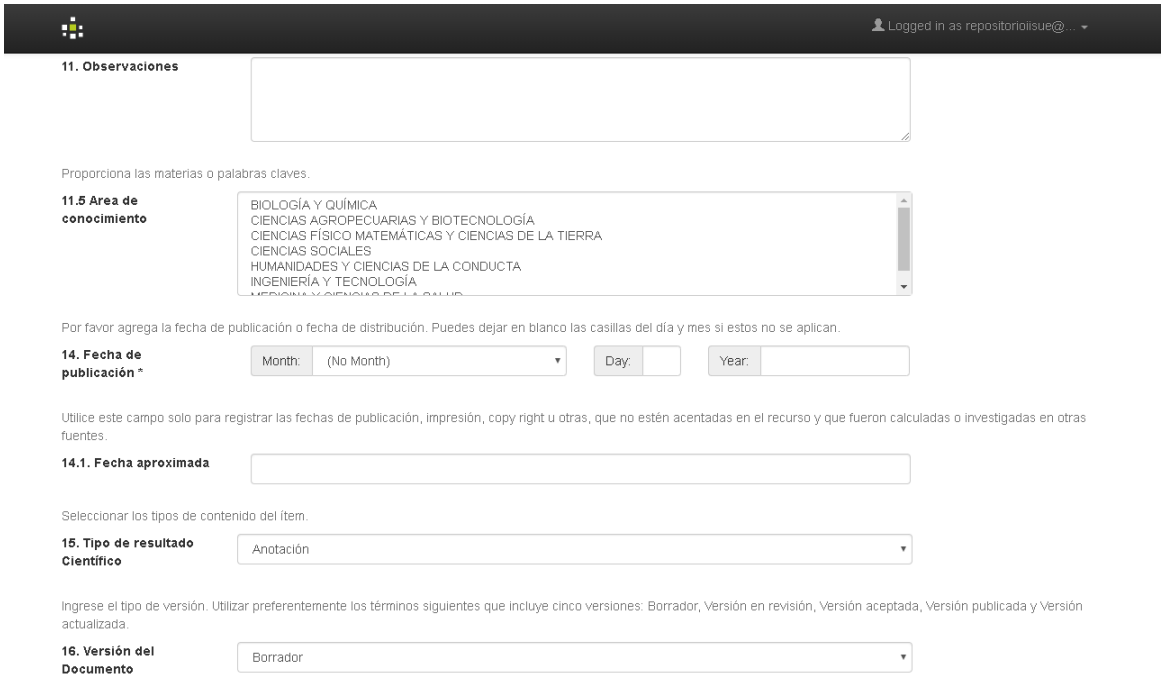

Figura 3.2: Interfaz JSPUI (Dspace, 2015b).

1. Reconocer en la forma de captura situada en JSPUI los campos ha modificar, como se ilustra en la Figura 3.2, donde se observan algunos de los campos que dispone el formulario, en negrita se denota la descripción para cada uno. Con base en esta exploración se identificaron los campos a cambiar y adicionalmente se eligieron los tipos apropiados con respecto a la utilidad para el usuario.

2. Realizar las modificaciones para lograr la comunicación entre los servicios provistos por CONACyT y los campos de la interfaz que corresponda.

Al añadir un elemento digital[3] al Repositorio Institucional, se tiene que realizar un

[3]También identificado como *item*.

proceso de carga conocido como flujo de trabajo[4], el cual está integrado por una serie de pasos que conforme a su configuración puede consistir simplemente de un paso, donde se registra e ingresa el recurso digital y al instante se hace su publicación, hasta un proceso de registro que requiera validación, edición y aprobación. Según el tipo de flujo de trabajo en cada paso del proceso, se presenta un formulario dependiendo del paso que corresponda, el formulario puede ser de edición o lectura[5], se determina que únicamente es necesario conectar los campos que residen en el formulario de inserción o edición.

3.3.1. Componente tipo *combo*

Derivado de lo anterior, se identificaron los dos componentes a emplear para la integración de los catálogos. El primero corresponde al tipo *combo*[6], como se muestra en la Figura 3.3 este componente al seleccionarse despliega su contenido, permitiendo al usuario elegir la opción que necesite. Por lo anterior, se considera para integrar aquellos catálogos cuyo contenido es limitado, con el propósito de mantener un despliegue y selección de información apropiado.

[4]En inglés *workflow*.
[5]Se denota aquel formulario que sólo presenta información para lectura del usuario.
[6]En *DSpace* se identifica como *dropdown*.

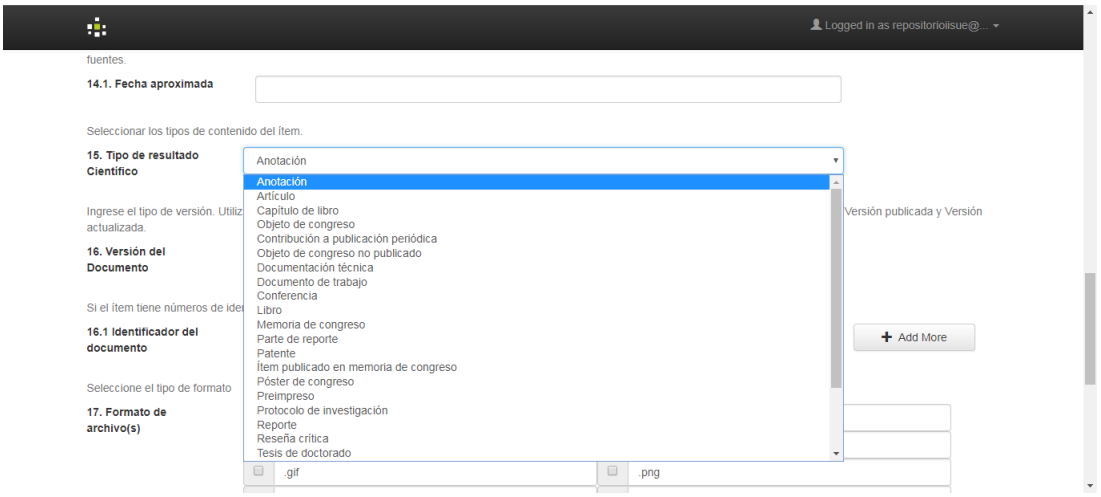

Figura 3.3: Componente tipo *combo* (Dspace, 2015b).

A continuación se describen los servicios de catálogos que se enlazaron a este componente:

- **Catálogo de Área de Conocimiento**

  El catálogo se puede consultar en la siguiente liga[7]:

  http://catalogs.repositorionacionalcti.mx/webresources/areacono

  Este servicio provee la información referente a las áreas de conocimiento establecidas por el CONACyT.

  De este servicio se requieren los campos:

[7]Servicio activo consultado en mayo 2019.

- cveArea: identificador asignado al área de conocimiento. Este identificador no es visible al usuario. Sin embargo, se almacena en base de datos y forma parte de los datos cosechados; resultado de la interoperabilidad con el Repositorio Nacional.
- descripcion: detalle asignado al área de conocimiento.

- **Catálogo de Resultado Científico**

  El catálogo se puede consultar en la siguiente liga[8]:

  http://catalogs.repositorionacionalcti.mx/webresources/tipopublicacion

  Este servicio contiene la información relacionada a la clasificación de tipos de publicación que están consideradas por el CONACyT.

  De este catálogo se toman los campos relacionados a:

  - clave: identificador asignado al tipo de publicación. Este identificador no es visible al usuario. Sin embargo, se almacena en base de datos y forma parte de los datos cosechados; resultado de la interoperabilidad con el Repositorio Nacional.
  - descCorta: nombre asignado al área de conocimiento.

[8] Servicio activo consultado en mayo 2019.

3.3.2. Componente tipo *caja de texto*

Para los catálogos *Autores* y *Colaboradores* cuyo contenido son de miles de registros, se consideró emplear el componente *caja de texto* [9] añadiendo la funcionalidad de autocompletado, con la finalidad de mostrar la información provista por los catálogos cuando el usuario comience a teclear en el campo de texto, el nombre o apellido determinado. Se consideró este modo, para facilitar la tarea de buscar y seleccionar el autor o colaborador, tal y como se muestra en la Figura 3.4. El usuario comienza a teclear en el campo de texto el nombre o apellido determinado, con esto se da inicio al proceso de búsqueda, como resultado sólo se muestran aquellas coincidencias, con esto se reduce significativamente el universo de opciones.

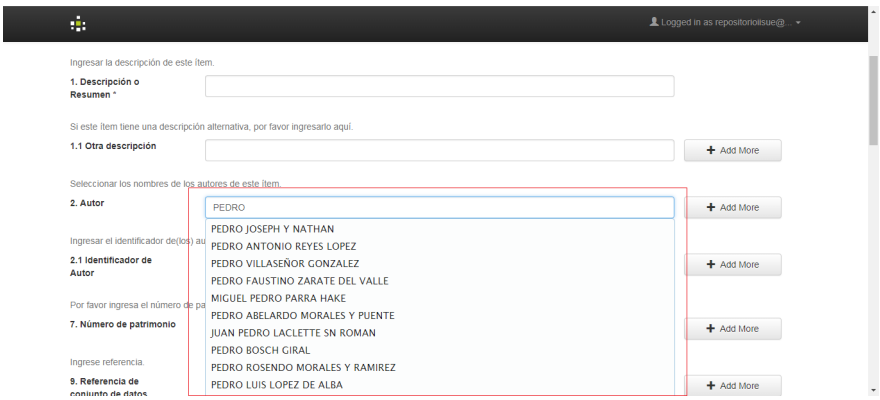

Figura 3.4: Componente tipo *caja de texto* (Dspace, 2015b).

[9] En *DSpace* se identifica como *onebox*.

- **Catálogo de Persona**

  El catálogo se puede consultar en la siguiente liga[10]:

  http://catalogs.repositorionacionalcti.mx/webresources/persona

  Este servicio contiene la información referente a los autores y colaboradores que el CONACyT tiene registrados y se actualiza a partir de la retroalimentación que cada instituto realice. En vista de que los institutos pueden añadir un autor que no esté registrado para el Repositorio Nacional, se le deberá notificar al CONACyT para que haga la incorporación correspondiente.

  De este servicio se toman los campos relacionados a:

  - nombres: corresponde al nombre del autor o colaborador.
  - primerApellido: corresponde al apellido paterno del autor o colaborador.
  - segundoApellido: corresponde al apellido materno del autor o colaborador.
  - Para estructurar el identificador de autor se contempla tomar uno de los siguientes valores:
    - **CVU**: corresponde al número de Curriculum Vitae Único asignado por el CONACyT.

[10]Servicio activo consultado en mayo 2019.

- **IDORCID**: por sus siglas en inglés *Open Researcher and Contributor ID* identificador único e independiente con la finalidad de eliminar la ambigüedad de los nombres de autores en publicaciones científicas.
- **CURP**: corresponde a la Clave Única de Registro de Población.
- **DNI**: corresponde a la clave del Documento Nacional de Identidad.
- **RN**: corresponde a la clave del Registro de Autor asignado por el CONACyT.
- **ND**: en caso que el autor o colaborador no tenga asociado algún identificador y esto es cuando el servicio no devuelve ningún valor asociado de CVU, IDORCID, CURP, DNI Y RN, se especifica la cadena ND con la finalidad de evitar su rechazo al transferirse al Repositorio Nacional.

El identificador no es visible para el usuario, sin embargo se guarda en base de datos y se transfiere en la cosecha como resultado de la interoperabilidad con el Repositorio Nacional.

3.4. Integración de los servicios CONACyT

A continuación, se describe la manera que se trabajó la conexión entre los catálogos y los componentes de la interfaz de edición, que corresponden al tipo *caja de texto* y al tipo *combo*.

3.4.1. Invocación de cliente-servicio a catálogo

Debido a que los servicios de los catálogos *Área de Conocimiento* y *Tipo de Publicación* devuelven un número reducido de elementos, se decide integrarlos a la interfaz de edición a través de un *combo*, por medio de la implementación de un cliente que tenga la tarea de consumir directamente al servicio del catálogo y regresar un arreglo con los elementos a presentar en dicho componente, lo anterior se conceptualiza en la Figura 3.5.

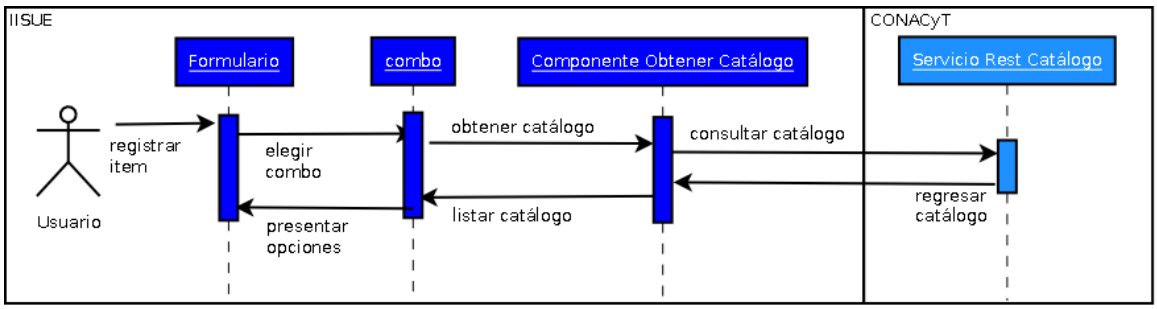

Figura 3.5: Invocación Cliente-Servicio.

3.4.1.1. Descripción de un cliente que dispone de un servicio de catálogo

En el Código 3.1 se ilustra la conexión del cliente *catalogoResultadoCientifico* al servicio del catálogo *TipoPublicacion* (líneas 5 y 6) y la recepción de su respuesta (línea 13). Puesto que el servicio devuelve un arreglo de tipo JSON[11] (por sus siglas en inglés *JavaScript Object Notation*), la respuesta se convierte a un objeto de tipo *List* (línea 15) para estar en condiciones de extraer los valores respectivos a la clave que corresponde al identificador junto con la descripción de cada elemento que integran al catálogo (líneas 19 y 20).

```
1  public ArrayList catalogoResultadoCientifico (){
2          ArrayList searchList =new ArrayList();
3     try {
4       ...
5      URL url = new URL("http://catalogs.repositorionacionalcti.mx/webresources
           /tipopublicacion/");
6      HttpURLConnection conn = (HttpURLConnection) url.openConnection();
7      conn.setRequestMethod("GET");
8      conn.setRequestProperty("Accept", "application/json");
9        if (conn.getResponseCode() != 200) {
10          throw new RuntimeException("Failed : HTTP error code : "
11                + conn.getResponseCode());
12       }
13       BufferedReader br = new BufferedReader(new InputStreamReader((conn.
             getInputStream())));
14      ...
15      List<ResultadoCientificoItem> lista =converter.fromJson(br.readLine(),type);
16      Iterator <ResultadoCientificoItem> iterator = lista . iterator ();

17      while ( iterator .hasNext()) {
18            ResultadoCientificoItem resultado=iterator.next();
19            searchList .add(resultado.getDescCorta());
20            searchList .add(resultado.getClave());
```

[11]Formato elemental para el intercambio de información. Se define por una estructura de pares *(variable, valor)* y que dicha estructura forma un arreglo. Lo anterior, favorece a su interpretación y manipulación en distintos lenguajes de programación (JSON, 2019).

```
21      }
22        conn.disconnect();
23       } catch (Exception e) {
24         ...
25       }
26      return searchList;
27    }
```

Código 3.1: Consulta y extracción de información a un servicio del CONACyT.

En el siguiente apartado se describe como desde la interfaz de edición se invoca al cliente y obtiene el contenido del *combo* a partir del arreglo *searchList* (línea 26 del Código 3.1).

3.4.1.2. Descripción de la invocación del cliente desde la interfaz de edición

En el Código 3.2 se presenta la función *doDropDown* que forma parte de la interfaz de edición y que tiene como labor la construcción del campo *combo*. Aunado a lo anterior, al revisar la estructura que compone al JSP encargado de construir el formulario de edición, se identificó la existencia del arreglo *valueList* que es usado para transferir los valores[12] hacia el componente *combo*, el arreglo es de tipo *ArrayList* donde cada elemento se conforma de un *identificador* y una *descripción*, al introducir un elemento al arreglo se debe conservar el siguiente orden:

1. Ingresar la descripción.

2. Ingresar el identificador.

[12]Estos valores integran el contenido del *combo*, se obtienen de los catálogos internos que maneja *DSpace*.

Al observar lo anterior, se decidió invocar al cliente conforme al campo de la interfaz de edición requerido, verificando por el tipo de metadato (líneas 7 y 13); el metadato *dc_subject_firstLevel* identifica al *combo Área de Conocimiento* y *dc_type* al *combo Resultado Científico*. Al recibir la respuesta con la lista de valores por parte del cliente (líneas 8 y 14) se reemplaza el contenido del arreglo *valueList* (líneas 10 y 16) de esta manera para la rutina encargada de construir el *combo* es transparente integrar los valores provistos en la respuesta del servicio (líneas 21 y 22) .

```
1  void doDropDown(javax.servlet.jsp.JspWriter out, Item item,
2        String fieldName, String schema, String element, String  qualifier , boolean
           repeatable,
3        boolean required, boolean readonly, List  valueList , String label)
4        throws java.io.IOException
5     {
6        ...
7        if(fieldName.equals("dc_subject_firstLevel")){
8           areaconocimiento = cbap.catalogoAreaConocimiento();
9            if (!areaconocimiento.isEmpty()){
10              valueList = areaconocimiento;
11           }
12       }
13       if(fieldName.equals("dc_type")){
14           resultadocientifico = cbap.catalogoResultadoCientifico();
15           if (! resultadocientifico .isEmpty()){
16              valueList = resultadocientifico ;
17           }
18       }
19       for (int i = 0; i < valueList.size(); i += 2)
20       {
21          display = (String) valueList.get(i);
22          value = (String) valueList.get(i+1);
23          for (j = 0; j < defaults.length; j++)
24          {
25              if (value.equals( defaults[j].value))
26                 break;
```

```
27          }
28          sb.append("<option ")
29            .append(j < defaults.length ? " selected=\"selected\" " : "")
30            .append("value=\"")
31            .append(value.replaceAll("\"", "&quot;"))
32            .append("\">")
33            .append(display)
34            .append("</option>");
35        }
36        sb.append("</select></span></div><br/>");
37        out.write(sb.toString());
38      }
```

Código 3.2: Llamado a la clase cliente desde el JSP de edición (Dspace, 2015a).

3.4.2. Invocación de Ajax-Servlet-Servicio del Catálogo

Los catálogos de *Autores* y *Colaboradores* son bastante extensos, siendo poco práctico mostrar sus contenidos por medio del componente tipo *combo*, de manera que, se optó por usar el componente *caja de texto*. Además para agilizar la búsqueda se contempló añadir la funcionalidad de autocompletado por medio de la integración de la tecnología *Ajax*[13]; dicha funcionalidad consiste en que el usuario al comenzar a teclear el nombre del autor o colaborador que desea registrar, activa un proceso cuya tarea es realizar la petición al servicio catálogo de persona, que como parte de su diseño, este servicio provee un método que permite iniciar la búsqueda dentro del universo de información en el Repositorio Nacional, la cual, sólo extraerá aquellos nombres que coincidan con lo que el usuario tecleó, de esta

[13]Por sus siglas en inglés *Ajax (Asynchronous JavaScript And XML)*. Tradicionalmente una vez cargada la página *web*, su contenido no podía variar hasta que nuevamente se volviera a presentar una recargar por completo. Sin embargo, este paradigma cambia al introducir esta tecnología que permite cambiar el contenido de una página *web* sin necesidad que sea recargada, a partir de eventos asíncronos, se pueden hacer invocaciones para solicitar información hacia el servidor y una vez recibida reflejarla en la página (Jonas Kluge, 2007).

manera se reduce en un número considerable el área de búsqueda. Así, sólo se le presenta al usuario el listado de coincidencias conforme al nombre de la persona especificado, como se puede observar en la Figura 3.6.

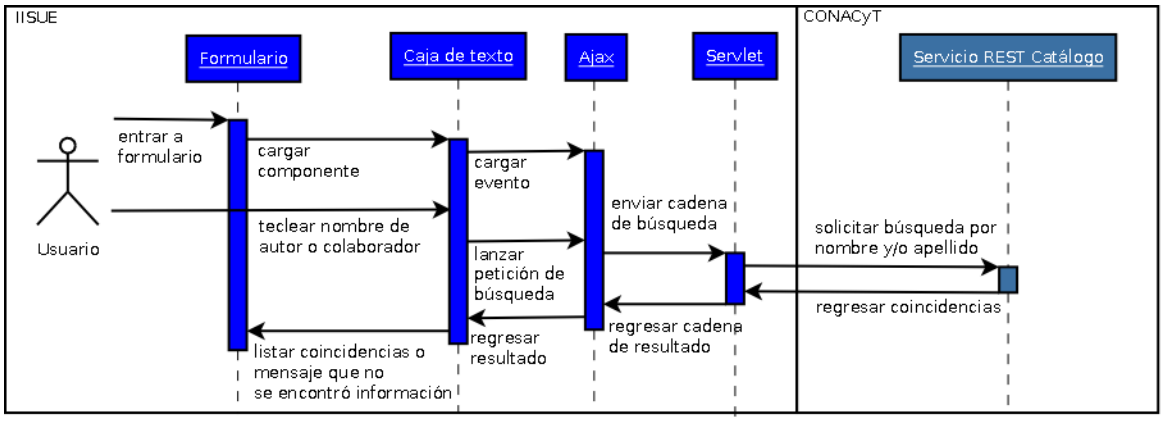

Figura 3.6: Invocación Cliente-Ajax-Servlet-Servicio.

3.4.2.1. Integración de *Ajax* en el componente *caja de texto*

En el Código 3.3 se muestra la función *doIisuebap* que se integró al JSP de edición para añadir al elemento *caja de texto* la funcionalidad de autocompletado (líneas 10 y 21) y la invocación al *servlet*[14] para hacer la petición de contenido al servicio del catálogo (líneas 11 y 22) una vez que el *servlet* responde, la misma función en *Ajax* se encarga de recibir la respuesta y presentarla en la *caja de texto* localizada en la interfaz de edición. De igual

[14]En la arquitectura *web* cliente-servidor, son módulos en *Java* que se ejecutan de lado del servidor a partir de una invocación para proveer o recibir información. Para este caso, en particular, se tomaron en cuenta ya que se consideraron los adecuados para conseguir la información a partir de una petición desde *Ajax*, con la cual hacer la construcción del contenido de los catálogos en la interfaz de edición.

forma como se mencionó anteriormente, por medio de los metadatos se hace la distinción cuando aplica la petición para el campo *Autor* (línea 5) y al de *Colaborador* (línea 14).

Aunado a lo anterior, en pruebas subsecuentes se reportó la necesidad de registrar más de un autor o colaborador; si bien, la interfaz de administración de formularios permite configurar un elemento de tipo múltiple, en un inicio únicamente se tenía considerado hacer peticiones hacía el catálogo de *Autor* o *Colaborador* solamente por campo, al presentarse esta necesidad, se trabajaron los ajustes en la función *doIisuebap* añadiendo un índice como identificador (líneas 7, 8, 18 y 19), este índice es propio de la función que construye el componente *caja de texto* en el formulario, la finalidad de añadirlo es para que cada campo pueda hacer invocaciones de manera independiente al *servlet* y poder obtener la información correspondiente.

```
1  String dolisuebap(boolean repetable,int index,String typeField){
2   StringBuffer sb = new StringBuffer();
3   String campoCreador ="dc_creator";
4   String campoIDCreador="dc_creator_ID";
5      if(typeField.equals("dc_creator")){
6          if(index >0 && repetable){
7           campoCreador+="_"+index;
8           campoIDCreador+="_"+index;
9        }
10     sb.append("<script type=\"text/javascript\">$(document).ready(function() {
           $(function() {if( document.getElementById(\""+campoCreador+"\") !=
           null){$(\"#"+campoCreador+"\").autocomplete({");
11     sb.append("source : function(request, callback) {$.ajax({url : \"/jspui/
           CONACyTController\",type : \"GET\",data : {term : request.term,id:
           1},dataType : \"json\",");
12      ...
13    }
14    if(typeField.equals("dc_contributor")){
```

```
15          campoCreador ="dc_contributor";
16          campoIDCreador="dc_contributor_ID";
17          if(index >0 && repetable){
18              campoCreador+="_"+index;
19              campoIDCreador+="_"+index;
20          }
21      sb.append("<script type=\"text/javascript\">$(document).ready(function() {
            $(function() {if( document.getElementById(\""+campoCreador+"\") !=
            null){$(\"#"+campoCreador+"\").autocomplete({");
22      sb.append("source : function(request, callback) {$.ajax({url : \"/jspui/
            CONACyTController\",type : \"GET\",data : {term : request.term,id:
            1},dataType : \"json\",");
23       ...
24      }
25      return sb.toString();
26  }
```

Código 3.3: Integración de *Ajax* al componente *caja de texto*.

3.4.2.2. Estructura central del *servlet*

En el Código 3.4 se muestra el método *doGet* encargado de atender la invocación realizada por *Ajax* desde la interfaz de edición. En primera instancia el método recibe la cadena de texto que corresponde al nombre del autor o colaborador que el usuario está buscando (línea 6), enseguida hace la petición hacía el método *ejecuta()*, el cual se encarga de invocar al servicio del catálogo (línea 8).

```
1  ...
2  protected void doGet(HttpServletRequest request,
3                  HttpServletResponse response) throws ServletException, IOException {
4                  response.setContentType("application/json");
5                  try {
6                          String term = request.getParameter("term");
7                          int idservicio = Integer.parseInt(request.getParameter("id"));
8                          response.getWriter().write(ejecuta(term, idservicio ));
9                  } catch (Exception e) {
10                         System.err.println(e.getMessage());
11                 }
12         }
13 ...
```

Código 3.4: Método que atiende la invocación *Ajax* desde el componente *caja de texto*.

3.4.2.3. Invocación al servicio del catálogo de *Persona*

El Código 3.5 ilustra como el *servlet* hace la petición de información al servicio del catálogo, este ejemplo en particular corresponde al método *ejecuta_autor*, el cual se usa para la búsqueda de autores, dicho método recibe la cadena de texto correspondiente al nombre o apellido de la persona e inicia la búsqueda a partir de la invocación al servicio de *Persona* (línea 5). Por cada coincidencia recibida en la respuesta (línea 16) se procede ir agregando en un arreglo (línea 51) el identificador, la descripción del identificador (líneas 20 a la 49) y el nombre del autor (línea 50) con la finalidad que el componente (*caja de texto*) al recibir este arreglo presente las coincidencias, para que el usuario elija la deseada y esta se pueda guardar con el identificador que le corresponda.

```
1  private String ejecuta_autor(String term){
2          ...
3     try {
4      ...
5          URL url = new URL("http://catalogs.repositorionacionalcti.mx/
               webresources/persona/byNombreCompleto/params;nombre="+URLEncoder.
               encode(term, "UTF-8").replaceAll("\\+","%20")+";limit=15;offset=0");
6          HttpURLConnection conn = (HttpURLConnection) url.openConnection();
7           ...
8          conn.setRequestMethod("GET");
9          conn.setRequestProperty ("Authorization", basicAuth);
10         conn.setRequestProperty("Accept", "application/json");
11        if (conn.getResponseCode() != 200) {
12           throw new RuntimeException("Failed : HTTP error code : "
13                                      + conn.getResponseCode());
14        }
15        BufferedReader br = new BufferedReader(new InputStreamReader((conn.
              getInputStream())));
16        List<PersonaItem> lista=converter.fromJson(br.readLine(),type);
17        List tempo = new ArrayList();
18        Iterator <PersonaItem> iterator = lista.iterator();
19        AutoresItem aut = new AutoresItem();
20        while ( iterator.hasNext()) {
21            PersonaItem cveO = iterator.next();
22            aut = new AutoresItem();
23            if(cveO.getCvu() !=null){
24                aut.setValue(cveO.getCvu());
25                aut.setIdentificador ("cvu");
26            }else{
27                if(cveO.getIdOrcid() !=null){
28                      aut.setValue(cveO.getIdOrcid());
29                      aut.setIdentificador ("orcid");
30                 }else{
31                      if(cveO.getCurp() !=null){
32                          aut.setValue(cveO.getCurp());
33                          aut.setIdentificador ("curp");
34                      }else{
35                          if(cveO.getDni() !=null){
36                              aut.setValue(cveO.getDni());
37                              aut.setIdentificador ("dni");
38                          }else{
39                              if(cveO.getRn() !=null){
40                                  aut.setValue(cveO.getRn());
```

```
41                                  aut. setIdentificador ("rn");
42                              }else{
43                                  aut.setValue("ND");
44                                  aut. setIdentificador ("ND");
45                              }
46                          }
47                      }
48                  }
49              }
50              aut.setLabel(cveO.getNombres()+" "+cveO.getPrimerApellido()+cveO.
                    getSegundoApellido());
51              tempo.add(aut);
52          }
53          searchList = converter.toJson(tempo);
54          conn.disconnect();
55      } catch (Exception e) {
56          ...
57      }
58     return searchList;
59  }
```

Código 3.5: Método *ejecuta_autor* encargado de hacer la conexión al servicio *Persona*.

3.4.2.4. Revisión de los identificadores de autor

Con base en la guía proporcionada por el CONACyT, el registro de autor o colaborador se debe guardar y enviar al Repositorio Nacional mediante la siguiente estructura:

- *IDENTIFICADOR* cadena que está conformada por la estructura:

NOMBRE\ID\VALOR DEL ID

Donde:

- *NOMBRE*: está conformado por el nombre, apellido paterno y materno.
- *ID*: corresponde alguna de las siguientes claves:
  - CVU: corresponde al número de Curriculum Vitae Único asignado por el CONACyT.
  - ORCID: por sus siglas en inglés *Open Researcher and Contributor ID* identificador único e independiente con la finalidad de eliminar la ambigüedad de los nombres de autores en publicaciones científicas.
  - CURP: corresponde a la Clave Única de Registro de Población.
  - DNI: corresponde a la clave del Documento Nacional de Identidad.
  - RN: corresponde a la clave del Registro de Autor asignado por el CONACyT.
  - ND: en caso que el autor o colaborador no tenga asociado algún identificador y esto es cuando el servicio no devuelve ningún valor asociado de CVU, IDORCID, CURP, DNI o RN; se asigna esta cadena con la finalidad de evitar su rechazo al transferirse al Repositorio Nacional.
- *VALOR DEL ID*: corresponde al valor asociado al identificador, por ejemplo para el CURP su valor asociado es el CURP asignado al autor o colaborador.

- *VALOR*: integrado por el nombre, apellido paterno y materno del autor o colaborador.

En un principio se consideró suficiente manejar sólo el *Open Researcher and Contributor ID* (ORCID); sin embargo, al realizar las primeras cosechas, se identificó que algunos autores registrados en el catálogo del Repositorio Nacional no contaban con valor de ORCID. Por lo cual, se reajustó la asignación de este identificador con base en las siguientes prioridades definidas:

1. Si el autor tiene CVU y cuenta con valor asociado, se asigna el identificador CVU.
2. Si el autor tiene ORCID y cuenta con valor asociado, se asigna el identificador ORCID.
3. Si el autor tiene CURP y cuenta con valor asociado, se asigna el identificador CURP.
4. Si el autor tiene DNI y cuenta con valor asociado, se asigna el identificador DNI.
5. Si el autor tiene RN y cuenta con valor asociado, se asigna el identificador RN.
6. Si no encontró ninguno de los anteriores, se asignará como ID la cadena ND y como valor asociado ND.

   Este valor deriva a partir de que el grupo de bibliotecarios reportó que era recurrente que el campo *Autor* o *Colaborador* no se habilitaba para capturar; pues en un inicio se consideró que cuando no se tuviera respuesta del servicio sobre una búsqueda no debía permitirse su captura manual y notificar al usuario mediante el mensaje:

   Autor no localizado en el catálogo

> En seguimiento a este hallazgo, el equipo de bibliotecarios comunicó que dicho caso se presentaba de forma recurrente conforme al listado de autores considerados a registrar. Ante el hecho que el catálogo del CONACyT no cuente con el registro de uno o varios autores, el equipo del IISUE acordó con el CONACyT que, en el caso de los autores no localizados se les hará llegar un listado; además se consideró necesario habilitar el campo *Autor* en el formulario para su captura y asignar el identificador ND para diferenciarlos en el Repositorio Nacional.

Lo anterior se plasmó en las líneas 23 a la 44 del Código 3.5 en el apartado 3.4.2.3. De esta forma el identificador cumple con lo solicitado y para aquellos autores que no cuenten con alguno, se puedan distinguir fácilmente de lado del Repositorio Nacional para su posterior registro.

Cuando se concluyó la implementación de la conexión hacía los catálogos, se inició la etapa de exploración de la interfaz por parte del equipo de bibliotecarios encargados del ingreso de los primeros registros al repositorio.

3.5. Reporte de estadísticas

La estadística en el ámbito de los repositorios digítales juega un papel importante, ya que se puede tener información referente a la calidad del contenido y uso; por ejemplo, conocer que materiales u autores tienen más consultas realizadas, o que material tiene

mayor o menor número de descargas. Los equipos de trabajo pueden tomar las decisiones necesarias para reforzar las áreas que consideren más pertinentes con la finalidad de mejorar la difusión de sus contenidos.

Según lo expone Facundo Adorno en su trabajo de titulación (Adorno, 2018), las estadísticas promueven las siguientes oportunidades:

- Incorporar nuevos idiomas, a partir del origen de los usuarios.
- Optimizar las páginas *web* para maximizar su visibilidad en los buscadores.
- Reorganizar los contenidos para darles mayor relevancia a aquellos menos utilizados.
- Promocionar servicios con bajo nivel de uso.
- Desarrollar servicios, herramientas y estrategias para aumentar el acceso desde ciertos dispositivos.
- Mejorar las herramientas de búsqueda y exploración en el repositorio.

3.5.1. Bitácoras en *DSpace*

DSpace cuenta con dos módulos donde *Solr* forma parte; *Solr* está basado en *Java Lucene*, que es un biblioteca de alto desempeño para realizar indización y búsquedas sobre texto. Se considera que *Solr* es la versión de Lucene en modalidad *web* a través de servicios. Lucene maneja una estructura de indexado conocida como *índice invertido*, que permite

establecer una correspondencia entre un conjunto de datos hacia su ubicación o nombre del documento origen.

El primero de los módulos se llama *Discovery* que está orientado principalmente a presentar los contenidos del repositorio agrupados por tema y relevancia. Por ejemplo en la Figura 3.7, se puede observar el listado de Autor (enmarcado en rojo) a partir del número de obras que tiene registrado cada uno, de mayor a menor contenido.

Figura 3.7: Listado de autores por cantidad de contenido (Dspace, 2015c).

La vía de acceso al módulo *Discovery* es a través de la siguiente URL local[15]:

```
http://localhost:8080/solr/search
```

El segundo módulo se llama *Statistics*, el cual está enfocado a recopilar los eventos que se presentan en las interfaces JSPUI y XMLUI. Este módulo provee la información necesaria para la implementación de los servicios requeridos por el CONACyT.

El módulo *Statistics* se habilitó a partir de la versión 1.6 de *DSpace*, cada que un archivo o página del repositorio es accedido, la actividad se almacena en las bitácoras de *logs*, las cuales se localizan en el directorio:

```
{directorio_DSpace}/logs
```

Y sus archivos de configuración residen en:

```
{directorio_DSpace}/solr/statistics/conf/schema.xml
{directorio_DSpace}/solr/statistics/conf/solrconfig.xml
```

Donde `{directorio_DSpace}` refiere al directorio de instalación.

El módulo *Statistics* cuenta con dos archivos principales de configuración: *schema.xml* y *solrconfig.xml*; en el archivo *schema.xml* se encuentran definidos los campos que contienen la configuración de indización, dichos campos se emplean para realizar las consultas correspondientes. El archivo que corresponde a *solrconfig.xml* contie-

[15]Por seguridad este módulo no pueden ser accedido de forma externa.

ne las configuraciones propias del motor de búsqueda. La vía de acceso al módulo[16] es: `http://localhost:8080/solr/statistics/select`

Solr cuenta con su propio lenguaje, mediante el cual se formulan las peticiones o consultas para buscar y extraer información requerida. A grandes rasgos una consulta se conforma de términos y campos; donde los términos pueden ser desde una palabra "título" hasta una frase, que es un conjunto de palabras encerradas entre doble comilla *"colecciones literarias"*; y los campos conformados por una etiqueta (nombre de los índices definidos en el archivo *schema.xml*) y valor (definimos lo que necesitamos buscar). Por ejemplo *q=type:2*.

Por otro lado *Solr* tiene una cualidad en la búsqueda y clasificación de la información, por medio de agrupamiento o *facet*, la cual permite obtener el *total de elementos* existentes en cada agrupación. Puesto que tres de los servicios solicitados por el CONACyT corresponden a listar por relevancia el número de materiales consultados, descargados e información sobre el número de autores buscados en el repositorio, se identificó que esta propiedad facilitaría obtener el total de los datos de cada concepto.

3.5.2. Examinar y generar consultas para solicitar información a *Solr*

Como primer paso para estar en condiciones de construir cada uno de los servicios, se investigó como formular las consultas para llevar a cabo la petición a *Solr*, a continuación

[16]Nótese el uso de *localhost*, por seguridad sólo se tiene acceso de manera interna.

se presentan las estructuras de cada una.

3.5.2.1. Información del número de visitas por recurso

Se requiere reportar lo referente a:

- Identificador del recurso (URI[17]).
- Número de visitas.

```
http://localhost:8080/solr/statistics/select?q=type:2&wt=json&
version=2&indent=on&facet.limit=15&facet.field=id&facet.mincount=1&fq=
-isBot:true&fq=statistics_type:view&facet=true
```

Donde:

- *q=type:2*, indica a *Solr* extraer únicamente información referente a documentos.
- *facet=true*[18]: indica a *Solr* que agrupe y a la vez contabilice el número de ocurrencias de aquellos documentos que coincidan con los criterios de búsqueda especificados.
- *facet.field=id*: indica a *Solr* que la agrupación y el conteo sea a través del campo *id*, mismo que está definido en el archivo *schema.xml*; recordemos que en este archivo se declaran los valores, que en un momento dado, se almacenan en las bitácoras. Usando

[17] Acrónimo en inglés *(Uniform Resource Identifier)*, este identificador corresponde a la ruta que ubica y diferencia a cada documento en el repositorio de manera única.

[18] Si esta propiedad se usa, es necesario definir por lo menos el parámetro *facet.field*.

el *id* y la clase *Item* que forma parte de la base principal de *DSpace* se obtiene el identificador de cada documento conocido como URI.

- *facet.mincount=1*: indica a *Solr* que responda con al menos una agrupación y su conteo correspondiente.

- *facet.limit=15*: indica a *Solr* que responda con un máximo de 15 agrupaciones junto con los conteos correspondientes.

- *fq=statiscts_type:view*: indica a *Solr* extraer información únicamente de las consultas realizadas al repositorio.

- *fq=-isBot:true*: indica a *Solr* que no tome en cuenta las peticiones ejecutadas por motores de clasificación de información, comúnmente buscadores. *DSpace* cuenta con el directorio de configuración *[DSpace]/config/spiders* para registrar aquellas *IP* que pertencen este tipo de motores o *bots*.

3.5.2.2. Información del número de descargas por recurso

Se requiere reportar lo referente a:

- Identificador del recurso (URI).

- Número de descargas.

```
http://localhost:8080/solr/statistics/select?q=type:0&wt=json&
version=2&indent=on&facet.limit=15&facet.field=id&facet.mincount=1&fq=
-isBot:true&facet=true
```

Donde:

- *q=type:0*: indica a *Solr* extraer únicamente información sobre las descargas realizadas.

- *facet=true*: indica a *Solr* que agrupe y a la vez contabilice el número de ocurrencias de aquellos documentos que coincidan con los criterios de búsqueda especificados.

- *facet.field=id*: indica a *Solr* que la agrupación y el conteo sea a través del campo *id*, mismo que está definido en el archivo *schema.xml*.

- *facet.mincount=1*: indica a *Solr* que responda con al menos una agrupación y su conteo correspondiente.

- *facet.limit=15*: indica a *Solr* que responda con un máximo de 15 agrupaciones junto con sus conteos correspondientes.

- *fq=-isBot:true*: indica a *Solr* que no tome en cuenta las peticiones ejecutadas por motores de clasificación de información, comúnmente buscadores.

3.5.2.3. Información del número de consultas por autor

Se requiere reportar lo referente a:

- Nombre y apellido(s) del autor.
- Número de consultas.

```
http://localhost:8080/solr/statistics/select?q=statistics_type:
search&facet.field=query&facet=true&fq=query:/author_keyword:.*/
```

Donde:

- *q=statistics_type:search*: indica a *Solr* que la búsqueda se realiza únicamente sobre los registros que correspondan a las consultas efectuadas.
- *facet=true*: indica a *Solr* que agrupe y a la vez contabilice el número de ocurrencias de aquellos documentos que coincidan con los criterios de búsqueda especificados.
- *facet.field=query*: indica a *Solr* que la agrupación y el conteo sea a través del campo *query*, mismo que está definido en el archivo *schema.xml*, esta cadena contiene el criterio que, en su momento el usuario empleó para realizar su búsqueda.
- *facet.limit=25*: indica a *Solr* el total de elementos que debe regresar en el listado de coincidencias. Se amplía a 25 ya que cuando se realiza la búsqueda, si en su momento el usuario usó otro filtro al realizar su búsqueda, *Solr* lo incluye en los elementos de

salida, por lo tanto pudiera traer más registros que no corresponden necesariamente a una búsqueda por autor.

- *fq=query:/autor_keyword:.*/*: indica a *Solr* considerar sólo las consultas efectuadas por autor.

3.5.2.4. Información referente a los depositarios en activo identificados en el Repositorio Institucional

Se requiere reportar lo relacionado a:

- Nombre del depositario.
- Apellidos (paterno y materno) del depositario.
- Correo electrónico del depositario.
- Número telefónico del depositario.

Para cubrir esta necesidad se implementó un catálogo local, en el cual se registraron los nombres y teléfonos de los usuarios que están a cargo de la población del repositorio.

3.5.3. Construcción de los servicios de estadística

Una vez definidas las consultas y criterios para obtener la información referente a las visitas y descargas por recursos así como el número de consultas por autor, pasamos a

exponer parte de la construcción de estos servicios, para lo cual se usó el módulo *DSpace-Rest*. Este módulo está enfocado al manejo y publicación de servicios tipo REST[19]. Los servicios REST han tenido una gran aceptación ya que se identifican como *recursos* donde no es necesario la creación de una sesión para interactuar con ellos, lo que permite mayor acoplamiento pues no están orientados a un cliente en específico, su uso se reduce a una petición por HTTP. Bajo este esquema de servicios REST, CONACyT requiere la implementación de los cuatro servicios de estadística.

A continuación se expone como se usó la API[20] *SolrJ* para plasmar en servicios las consultas vistas en la subsección 3.5.2; para cada uno se expondrán ciertas modificaciones realizadas para obtener la información requerida.

En la Figura 3.8 se ilustra el consumo de los servicios por CONACyT y la comunicación interna entre los módulos principales para proveer la información, el servicio encargado de proveer la información de los depositarios no entra en este esquema ya que su fuente de información proviene de un catálogo de usuarios.

[19]Por sus siglas en inglés REST *(**RE**presentational **S**tate **T**ransfer)*.

[20]Por sus siglas en inglés API *(Application Programming Interface)*, brinda las clases necesarias para lograr una comunicación transparente con el motor *Solr*.

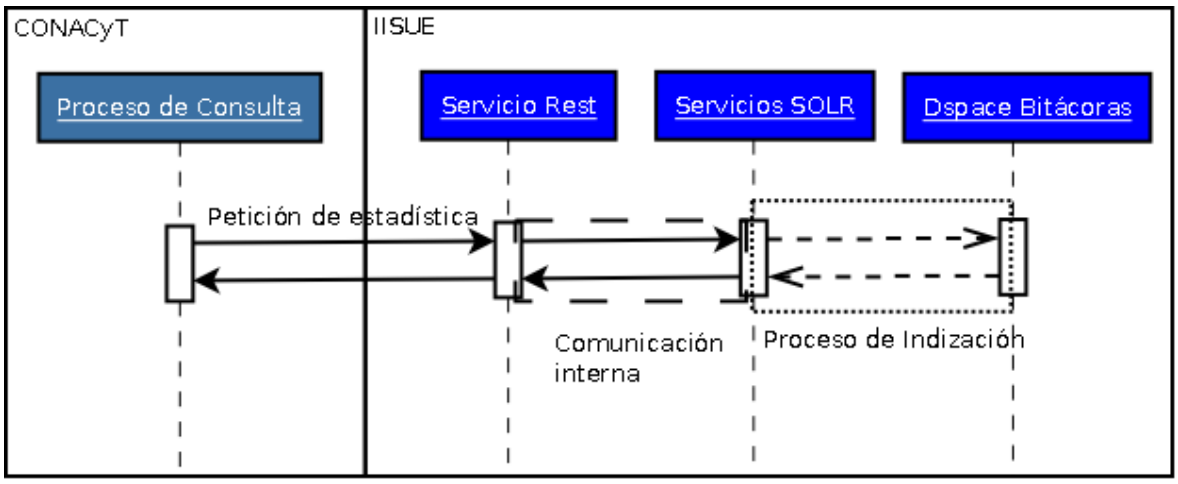

Figura 3.8: Servicios de estadística.

3.5.3.1. Estructura del servicio para obtener el número de visitas por recurso

En el Código 3.6 se muestra, de la línea 6 a la 14, el procedimiento para definir la consulta vista en el apartado 3.5.2.1 mediante los métodos *set* que ofrece la API.

Puesto que la búsqueda se agrupa por el identificador de cada documento (*id*, línea 17) , se usa la clase *Item* (línea 24) para obtener la instancia de cada documento y conseguir el identificador solicitado (URI) :

```
Item item = Item.find(contexto,Integer.parseInt(oct.getName()));
```

Al obtener la instancia del objeto *Item* se extrae el URI mediante la especificación del metadato *dc.identifier.uri* (línea 28). Por medio de la clase *Recursos* (línea 27) se va construyendo con cada URI (línea 28) y conteo obtenido (línea 29), los cuales forman el listado (línea 35) que integrará la respuesta del servicio (línea 42).

```
1  public String rankingArticulos (@QueryParam("q") String arg, @QueryParam("rows")
       String rows, @QueryParam("filter") String filter, @HeaderParam("authorization")
       String authString ){
2        ...
3        try{
4           String urlString = "http://localhost:8080/solr/statistics";
5           HttpSolrServer solr = new HttpSolrServer(urlString);
6           SolrQuery query = new SolrQuery();
7           query.setQuery("type:2");
8           query.setParam("shards",ClientUtils.escapeQueryChars("localhost:8080/solr/
                statistics"));
9           query.setRows(0);
10          query.addFacetField("id");
11          query.setFacetLimit(15);
12          query.setFacet(true);
13          query.setFacetMinCount(1);
14          query.addFilterQuery("-isBot:true","statistics_type:view");
15          QueryResponse resp = solr.query(query);
16           ...
17          FacetField field = resp.getFacetField("id");
18           ...
19          if ( field != null){
20             List <FacetField.Count> fc = field.getValues();
21              Iterator <FacetField.Count> oc =fc.iterator();
22             while(oc.hasNext()){
23                FacetField.Count oct=oc.next();
24                Item item = Item.find(contexto, Integer.parseInt(oct.getName()));
25                if(item != null){
26                   if( item.getMetadata("dc.identifier.uri")!=null){
27                      Recursos recurso = new Recursos();
28                      recurso.setId(item.getMetadata("dc.identifier.uri"));
29                      recurso.setNumero((int)oct.getCount());
30                      tempo.add(recurso);
31                   }
32                }
33             }
34              ...
35             salida =converter.toJson(tempo);
36          }else{
37              ...
38          }
39       }catch(Exception e){
40         ...
```

```
41      }
42        return salida ;
43    }
```

Código 3.6: Servicio para obtener el número de visitas por recurso.

3.5.3.2. Estructura del servicio para obtener el número de descargas por recurso

En el Código 3.7 se muestra, de la línea 8 a la 15, el procedimiento para definir la consulta vista en el apartado 3.5.2.2 mediante los métodos *set* que ofrece la API.

Puesto que la búsqueda se agrupa por el identificador de cada documento (*id*, línea 18) , se usa la clase *Item* (línea 24) para obtener la instancia de cada documento y conseguir el identificador solicitado (URI) :

```
Item item = Item.find(contexto,Integer.parseInt(oct.getName()));
```

Al obtener la instancia del objeto *Item* se extrae el URI mediante la especificación del metadato *dc.identifier.uri* (línea 28). Por medio de la clase *Recursos* (línea 27) se va construyendo con cada URI (línea 28) y conteo obtenido (línea 29), los cuales forman el listado (línea 35) que integrarán la respuesta del servicio (línea 42).

```
1  public String descargas (@QueryParam("q") String arg, @QueryParam("rows") String rows,
       @QueryParam("filter") String filter, @HeaderParam("authorization") String
     authString ){
2        ...
3       try{
4           ...
5          String urlString = "http://localhost:8080/solr/statistics";
6          HttpSolrServer solr = new HttpSolrServer(urlString);
7          SolrQuery query = new SolrQuery();
```

```
8           query.setQuery("type:0");
9           query.setParam("shards",ClientUtils.escapeQueryChars("localhost:8080/solr/
                statistics"));
10          query.setRows(0);
11          query.addFacetField("id");
12          query.setFacetLimit(20);
13          query.setFacet(true);
14          query.setFacetMinCount(1);
15          query.addFilterQuery("-isBot:true");
16             QueryResponse resp = solr.query(query);
17          List tempo = new ArrayList();
18          FacetField field = resp.getFacetField("id");
19          if ( field != null){
20               List <FacetField.Count> fc = field.getValues();
21               Iterator <FacetField.Count> oc =fc.iterator();
22               while(oc.hasNext()){
23                    FacetField.Count oct=oc.next();
24                    item = Item.find(contexto, Integer.parseInt(oct.getName()));
25                    if(item != null){
26                         if( item.getMetadata("dc.identifier.uri")!=null){
27                              Recursos recurso = new Recursos();
28                              recurso.setId(item.getMetadata("dc.identifier.uri"));
29                              recurso.setNumero((int)oct.getCount());
30                              tempo.add(recurso);
31                         }
32                    }
33               }
34               ...
35                    salida =converter.toJson(tempo);
36          }else{
37               ...
38          }
39          }catch(Exception e){
40               ...
41       }
42          return salida;
43    }
```

Código 3.7: Servicio para obtener el número de descargas por recurso.

3.5.3.3. Estructura del servicio para obtener el número de consultas por autor

En el Código 3.8 se muestra, de la línea 7 a la 14, el procedimiento para definir la consulta vista en el apartado 3.5.2.3 mediante los métodos *set* que ofrece la API. Este servicio difiere respecto a los dos anteriores en que no es necesario generar una instancia de alguna clase, para este caso los nombres se obtienen de la bitácora que maneja *Solr* en las líneas 14 y 24 a la 29, mediante la clase *Autores* se va construyendo el listado junto con los totales para cada uno (líneas 25 y 29) que integrará la respuesta del servicio (línea 40).

```
1  public String rankingAutores (@QueryParam("q") String arg, @QueryParam("rows") String
       rows, @QueryParam("filter") String filter, @HeaderParam("authorization") String
       authString ){
2        ...
3       try{
4          String urlString = "http://localhost:8080/solr/statistics";
5          HttpSolrServer solr = new HttpSolrServer(urlString);
6          SolrQuery query = new SolrQuery();
7          query.setQuery("statistics_type:search");
8          query.setParam("shards",ClientUtils.escapeQueryChars("localhost:8080/solr/
               statistics"));
9          query.setRows(0);
10         query.addFacetField("query");
11         query.setFacetLimit(25);
12         query.setFacet(true);
13         query.setFacetMinCount(1);
14         query.addFilterQuery("query:/author_keyword:.*/");
15         QueryResponse resp = solr.query(query);
16         List tempo = new ArrayList();
17         FacetField field = resp.getFacetField("query");
18         if ( field != null){
19            List <FacetField.Count> fc = field.getValues();
20            Iterator <FacetField.Count> oc =fc.iterator();
21            while(oc.hasNext()){
22               FacetField.Count oct=oc.next();
23               oct.getName();
```

```
24                  if(oct.getName().contains("author_keyword:")){
25                     Autores autor = new Autores();
26                     String separa[]=oct.getName().split(":");
27                     autor.setNombre(separa[1]. replaceAll ("\\"+"\\",""));
28                     autor.setNumero((int)oct.getCount());
29                     tempo.add(autor);
30                  }
31               }
32               ...
33                  salida =converter.toJson(tempo);
34           }else{
35               ...
36           }
37       }catch(Exception e){
38            ...
39       }
40         return salida ;
41    }
```

Código 3.8: Servicio para obtener el número de consultas por autor.

3.5.3.4. Estructura del servicio para obtener información de los usuarios depositarios

El servicio referente a la información de los usuarios encargados de capturar los datos en el repositorio se muestran en el Código 3.9, se decidió hacer un listado con los datos de los encargados de realizar dicha labor; en el concentrando se incluyó el nombre, apellido paterno, apellido materno, teléfono y correo electrónico. Por ser clasificada esta información como datos personales y con base en la especificación solicitada para este servicio, se añadió un esquema de seguridad para que su uso sea a partir de proporcionar un usuario y contraseña (ver Figura 3.9).

```
1  public String padron (@QueryParam("q") String arg, @QueryParam("rows") String rows,
       @QueryParam("filter") String filter, @HeaderParam("authorization") String
       authString ){
2            ...
3        if(authString==null||!isUserAuthenticated(authString)){
4            return ...
5          }
6        try{
7            ...
8          if(!l.isEmpty()){
9              List tempo = new ArrayList();
10             Iterator<String> itr = l.iterator();
11             while (itr.hasNext()){
12                 StringTokenizer tokens=new StringTokenizer(itr.next(),"|");
13                 while(tokens.hasMoreTokens()){
14                     Depositarios depos = new Depositarios();
15                     depos.setNombre(tokens.nextToken());
16                     depos.setPApellido(tokens.nextToken());
17                     depos.setSApellido(tokens.nextToken());
18                     depos.setCorreo(tokens.nextToken());
19                     depos.setNumTel(tokens.nextToken());
20                     tempo.add(depos);
21                 }
22             }
23              ...
24                 salida =converter.toJson(tempo);
25         }
26       }catch(Exception e){
27           ...
28       }
29         return salida ;
30   }
```

Código 3.9: Servicio para obtener información de los usuarios depositarios.

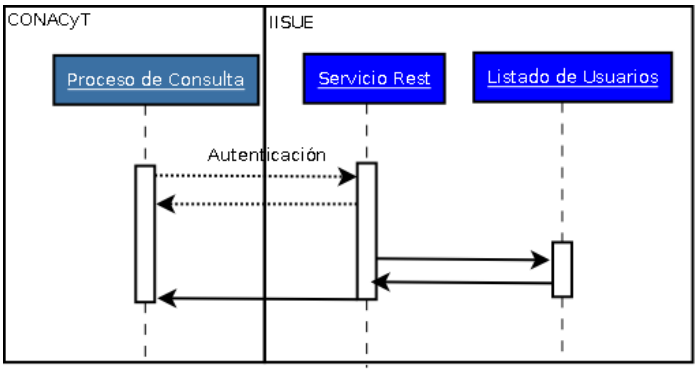

Figura 3.9: Servicio para obtener información de los depositarios.

Por último se realizaron las pruebas de estadísticas, proporcionando la URL de cada servicio, con la finalidad que el área del Repositorio Nacional del CONACyT hiciera la conexión a cada uno y verificara primero si podían conectarse al servicio; segundo validar que los servicios cumplan con las especificaciones establecidas y en su momento realizar la conexión definitiva para la puesta en operación del repositorio.

En este capítulo se describió el trabajo realizado para cumplir con la configuración solicitada por el CONACyT, se expuso el fundamento establecido para la interoperabilidad de los repositorios con el Repositorio Nacional. Se mostró en general la estructura que conforma a *DSpace*, resaltando los módulos esenciales con los cuales se trabajó directamente. Enseguida se presentó como se identificaron y determinaron los campos a usar para integrar los catálogos provistos por CONACyT; sin perder de vista que fuese útil para el usuario. Se describieron las clases, sus estructuras principales para lograr la integración y comunicación entre los campos de la interfaz de edición hacia los catálogos del CONACyT. Además

se explicó la construcción de los servicios de estadística, resaltando los puntos importantes y el porqué se solicitaron, se continuó con la descripción de las consultas elaboradas para obtener la información, finalizando con la exposición de las principales estructuras que se implementaron para cada uno de los servicios.

Capítulo 4

Conclusiones

A continuación se presenta un recuento de los resultados alcanzados, se hace el planteamiento del trabajo pendiente y se comparte una apreciación personal sobre la participación en este proyecto.

4.1. Conclusiones y trabajo a futuro

Al hacer uso y adecuar *DSpace* en la implementación y adecuación del Repositorio Institucional del IISUE se logró cubrir satisfactoriamente la integración de los catálogos provistos por el CONACyT al repositorio digital IISUE. Esto conforme a las normas establecidas en el documento de Lineamientos Específicos para Repositorios.

Como resultado de revisar la interfaz JSPUI, específicamente la que está dirigida

a la captura de información, se identificó y determinó el tipo de campo a utilizar para establecer la conexión a los catálogos, siendo estos reconocidos en *DSpace* como *onebox* y *dropdown*. Posteriormente, se averiguó a nivel de código el componente encargado de hacer la construcción del formulario de edición y se trabajó la conectividad con los catálogos del CONACyT.

Respecto a la implementación de los servicios de estadística, como parte de lo requerido para la interoperabilidad con el Repositorio Nacional, se analizó el motor *Solr* ubicado en el módulo *DSpace-Solr*, se halló que lo constituye un módulo llamado *Statistics* y que a su vez cuenta con unas bitácoras que registran la actividad que se lleva a cabo en las interfaces gráficas. Con el lenguaje que brinda *Solr* se elaboraron las consultas para obtener el número de materiales visitados y descargados; así como saber el número de autores consultados. Una vez que se contó con esta información se implementaron los servicios REST usando el módulo *REST-DSpace* y la *API SolrJ*.

La implementación de un repositorio digital no es una tarea exclusiva del área tecnológica, ya que demanda la participación de otras áreas "como la bibliotecología" puesto que es un trabajo interdisciplinario, aún cuando la cultura sobre el resguardo digital no está del todo permeada a nivel usuario en general. Lo anterior implica una labor continua para fomentar y acercar a los distintos grupos de trabajo.

En una siguiente etapa se puede contemplar asentar un esquema de conservación digital, el cual garantice que todo el material en el repositorio se mantenga legible y en

aquellos casos que por cambios tecnológicos esté en riesgo de quedar inaccesible algún material, tener la capacidad de asegurar su integridad y disponibilidad.

Para finalizar, haber tenido la oportunidad de formar parte de este proyecto, me permitió divisar un panorama respecto al trabajo y esfuerzo que involucra establecer un repositorio que resguarde contenidos digitales. Valorar el alcance y beneficio a la comunidad universitaria y la sociedad en general, pues se habilita una herramienta que además de contener parte del material elaborado en el IISUE facilita la creación de lazos de cooperación con otras comunidades científicas y universitarias, ya sea dentro o fuera del país.

Bibliografía

Academia, R. (2019). Diccionario de la lengua española. https://dle.rae.es/?id=W3mzJyE. [Consultado en: agosto 2019].

Adorno, F. G. (2018). Reportes estadísticos para repositorios digitales desarrollados en DSpace, Universidad Nacional de la Plata, Licenciatura en Sistemas.

Barrueco Cruz, J. M. U. d. V., Andrés Rodríguez, A. F. E. p. l. C. y. l. T., Rico Castro, P. F. E. p. l. C. y. l. T., Coslado Bernabé, M. n. F. E. p. l. C. y. l. T., Azorín, C. U. A. d. B., Bernal Martínez, I. C. S. d. I. C., Cívico Martín, R. U. d. H., Cózar Santiago, A. U. d. N., Guzmán Pérez, C. U. d. C., Losada Yáñez, M. U. P. F., Morillo Moreno, J. C. U. d. H., Nonó Rius, B. U. d. G., Padrós Cuxart, R. U. O. d. C., and Prats Prat, J. U. P. d. C. (2017). *Guía para la evaluación de repositorios institucionales de investigación*. Guia realitzada amb el suport de la Fundación Española para la Ciencia y la Tecnología i Crue Universidades Españolas, https://ddd.uab.cat/record/184795 [Consultado:agosto 2019].

Carlos André Rosa, O. C. and Domingues, P. (2017). Open source software for digital

preservation repositories: A survey. *International Journal of Computer Science & Engineering Survey (IJCSES)*, 8(3).

Cherukodan Surendran (2014). Measuring the Maturity of Open Source Software for Digital Libraries a Case Study of DSpace. *Cochin University of Science and Technology*, page 21. http://hdl.handle.net/10603/71778 [Consultado en: agosto 2019].

CONACyT (2014a). Comunicación Pública de la Ciencia, Tecnología e Innovación. https://www.conacyt.gob.mx/index.php/comunicacion. [Consultado en: octubre 2018].

CONACyT (2014b). Consejo Nacional de Ciencia y Tecnología. https://www.conacyt.gob.mx/index.php/el-conacyt. [Consultado en: septiembre 2018].

CONACyT (2015). *Lineamientos Técnicos para el Repositorio Nacional y los Repositorios Institucionales*. CONACyT.

CONACyT (2017a). Lineamientos específicos para repositorios. https://www.repositorionacionalcti.mx/documentos. [Consultado en: noviembre 2018].

CONACyT (2017b). Política de Ciencia Abierta. https://www.repositorionacionalcti.mx/documentos. [Consultado en: agosto 2019].

CONACyT (2018a). Interoperabilidad con el metabuscador del repositorio nacional. https://www.repositorionacionalcti.mx/documentos. [Consultado en: noviembre 2018].

CONACyT (2018b). Lineamientos generales de ciencia abierta. https://www.repositorionacionalcti.mx/documentos. [Consultado en: noviembre 2018].

CONRICyT (2018). Conócenos. https://www.conricyt.mx/acerca-del-consorcio/conocenos. [Consultado en: septiembre 2018].

Corporation, O. (2019). Java. https://www.java.com/es/download/faq/whatis_java.xml. [Consultado en: agosto 2019].

DCMI (2019). Dublin Core Metadata Initiative (DCMI). https://dublincore.org/specifications/dublin-core/. [Consultado en: agosto 2019].

De Giusti Marisa Raquel, Lira Ariel Jorge, O. N. F. (2012). Introducción a XMLUI. In *Curso de capacitación en repositorios y DSpace*, http://sedici.unlp.edu.ar/handle/10915/25158 [Consultado en: agosto 2019].

Debian (2015). ¿Qué es GNU/Linux? https://www.debian.org/releases/jessie/s390x/ch01s02.html.es. [Consultado en: agosto 2019].

Dspace (2015a). Edit metadata form source code. (DSpace 5.1)[source code] https://duraspace.org/dspace/.

Dspace (2015b). JSPUI Interfaz. (DSpace 5.1) https://duraspace.org/dspace/.

Dspace (2015c). XMLUI Interfaz. (DSpace 5.1) https://duraspace.org/dspace/.

Federation, D. L. (2010). Metadata Encoding and Transmission Standard. http://www.loc.gov/standards/mets/METSPrimer.pdf. [Consultado en: agosto 2019].

Group, P. W. (2005). Data dictionary for preservation data. https://www.loc.gov/standards/premis/v1/premis-dd_1.0_2005_May.pdf. [Consultado en: agosto 2019].

IBM (2019). Tecnología JSP (JavaServer Pages). `https://www.ibm.com/support/knowledgecenter/es/SS5JSH_9.1.1/org.eclipse.wst.webtools.doc.user/topics/cpdjsps.html`. [Consultado en: septiembre 2019].

Jeroen Bekaert, Patrick Hochstenbach, H. V. d. S. (2003). Using MPEG-21 DIDL to Represent Complex Digital Objects in the Los Alamos National Laboratory Digital Library. http://dlib.org/dlib/november03/bekaert/11bekaert.html. [Consultado en: agosto 2019].

Jonas Kluge, Frank Kargl, M. W. (2007). THE EFFECTS OF THE AJAX TECHNOLOGY ON WEB APPLICATION USABILITY. *WEBIST*.

JSON (2019). Introducing JSON. https://www.json.org/. [Consultado en: agosto 2019].

Library of Congress (2006). Frequently Asked Questions (FAQ). https://www.loc.gov/marc/faq.html#definition. [Consultado en: agosto 2019].

Library of Congress (2018). MODS (Metadata Object Description Standard). https://www.loc.gov/librarians/standards. [Consultado en: agosto 2019].

Lynch, C. A. (2003). Institutional repositories: essential infrastructure for scholarship in the digital age. http://www.arl.org/newsltr/226/ir.html. [Consultado en: noviembre 2018].

OpenDOAR (2019). Software Platforms Overview. http://v2.sherpa.ac.uk/view/repository_visualisations/1.html. [Consultado en: Octubre 2019].

Siicyt (2015). Sobre el Siicyt/Introducción. http://www.siicyt.gob.mx/index.php/el-siicyt/sobre-el-siicyt. [Consultado en: octubre 2018].

Silva, T. E. and Tomaél, M. I. (2012). Repositorios institucionales: directrices para políticas de información. consideraciones (1). *Ciencias de la Información*, 42(3):39–46.