Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Repositorio de Objetos de Aprendizaje de Contenidos Abiertos Reutilizables bajo una Arquitectura Orientada a Servicios

Trabajo Especial de Grado presentado ante la Ilustre

Universidad Central de Venezuela por el Bachiller:

Juan Carlos Morantes Taccetti

C.I. 17.705.291

Para optar al título de Licenciado en Computación

Tutor:

Prof. Yosly Hernández

Caracas, Octubre 2013

ACTA

Quienes suscriben, Miembros del Jurado designado por el Consejo de la Escuela de Computación para examinar el Trabajo Especial de Grado, presentado por el Bachiller Juan Carlos Morantes C.I.: 17.705.291, con el título “Repositorio de Objetos de Aprendizaje de Contenidos Abiertos Reutilizables bajo una Arquitectura Orientada a Servicios”, a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído el trabajo por cada uno de los Miembros del Jurado, se fijó el día 22 de octubre de 2013, a las 02:00pm, para que su autor lo defendiera en forma pública, en la Sala I de la Escuela de Computación, Facultad de Ciencias de la Universidad Central de Venezuela, lo cual realizó mediante una exposición oral de su contenido, y luego respondió satisfactoriamente a las preguntas que le fueron formuladas por el Jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo.

En fe de lo cual se levanta la presente acta, en Caracas el 22 de octubre de 2013.

_____

Prof. Yosly Hernández

Tutor

_____ _____

Prof. Franklin Sandoval Prof. Carlos Acosta

Jurado Jurado

Agradecimientos y dedicatoria

Agradezco principalmente a Dios por permitirme estar aquí cumpliendo un sueño y por no dejarme solo en ningún momento.

A mi padre, por todo el apoyo profesional y humano que me ha dado toda la vida.

A mis hermanos y mi cuñada, por el inmenso apoyo que me han brindado a lo largo de mi carrera.

A mi abuela Gioconda y mi tía Nena, por sus orientaciones y toda la experiencia que han compartido conmigo toda mi vida.

A Yasibit Figueredo por brindarme todo su apoyo desde siempre, igualmente a Tibisay Petit, Carlos Figueredo y el resto de la familia Figueredo.

A mi amigo y hermano David Rojas por todo el apoyo durante toda la carrera.

A Ignacio Ortiz y Vicente Ordoñez por todo el apoyo que me han brindado y por compartir conmigo sus conocimientos y experiencias en el ámbito académico y laboral.

A Carlos Sarmiento, Lizardo Lugo, Joel Carnevali, Jose Luis Carnevali, Carlos Sarpi, Francesco Sarpi, Sabrina Osorio, Raibel Nuñez, Iñaki Plessman, Mantura Kabchi, Alexander Santaniello y Sheila Hernández, por la amistad y hermandad que me han brindado por años. Son pilares y símbolos de motivación en muchos sentidos de mi vida.

A Patricia Montoya e Issis Heredia por su inmenso apoyo y compañía.

A Elvia Stifano y Antonella Capodicasa por estar siempre conmigo.

A todas aquellas personas que de una u otra manera me ayudaron durante el desarrollo de este Trabajo Especial de Grado como Yubisay Abraham, Cristina Lofiego, Wilfredo Gómez y Alexander Gómez.

A mi tutora Yosly Hernández por el increíble apoyo y confianza que ha tenido en mi trabajo y por haberme guiado en el desarrollo de este Trabajo Especial de Grado. Igualmente al profesor Franklin Sandoval por haberme alentado a alcanzar esta meta desde el primer momento. También al profesor Carlos Acosta por haberme ayudado a corregir los errores del TEG y haber sido un ejemplo de excelencia.

Este Trabajo Especial de Grado está dedicado a Bertha Irene Taccetti.

Resumen

La Unidad de Educación a Distancia y la Escuela de Computación en conjunto con otras Escuelas de la UCV y otras instituciones universitarias, están llevando a cabo una serie de proyectos en donde se hace uso de Objetos de Aprendizajes de Contenidos Abiertos, los cuales son desarrollados por alumnos y profesores de la universidad. Bajo este contexto, el problema que se presenta es que no se cuenta con un espacio digital apropiado para albergar todos estos recursos de manera eficiente y permitir la reutilización de los mismos. Tomando en cuenta esta situación, el objetivo que se persigue con el presente Trabajo Especial de Grado es construir un Repositorio para llevar la gestión de los Objetos de Aprendizaje de Contenidos Abiertos. Este Repositorio debe estar construido sobre arquitectura orientada a servicios REST que facilite la futura comunicación con otras aplicaciones. Para la realización de este Trabajo Especial de Grado se emplearon frameworks, aplicaciones y tecnologías libres. Entre ellas podemos mencionar: Postgresql como Sistema Manejador de Bases de Datos. PHP, como lenguaje de programación del lado del servidor. Symfony2, como framework de desarrollo para PHP. JQuery, como framework de Javascript para la programación del lado del cliente. APACHE, como servidor de aplicaciones. Modelo Vista Controlador, como patrón de desarrollo. REST, como arquitectura para la elaboración de Servicios Web. Linux, como sistema operativo, entre otros. Siguiendo los principios y actividades de la Programación Extrema (XP), se dividió la elaboración en varias iteraciones, durante las cuales se ponían en producción pequeñas secciones del Repositorio, las cuales eran sometidas a un conjunto de pruebas antes de continuar el desarrollo.

Palabras Clave: Objetos de Aprendizaje, Contenidos Abiertos, Repositorios, Arquitectura REST, Symfony2, Programación Extrema, Patrón MVC, Postgresql.

Tabla de Contenido

Índice de Figuras

Índice de Tablas

Índice de Gráficos

Introducción

El gran fortalecimiento que ha tenido la internet en los últimos tiempos, y la incorporación de los computadores y la tecnología en nuestra vida cotidiana, ha provocado una transformación radical de las formas de producción, difusión y consumo del conocimiento y la cultura. Así mismo, han surgido nuevos enfoques educativos con innovadoras técnicas para la enseñanza y el aprendizaje que se apoyan en recursos digitales (De la Cueva, 2007). Estos recursos digitales son los que mejor se conocen como Objetos de Aprendizaje, los cuales están destinados a apoyar tanto la formación a distancia como los cursos presenciales. Cuando los Objetos de Aprendizaje se desarrollan en base al conocimiento libre y se distribuyen bajo los lineamientos del acceso abierto, toman el nombre de Objetos de Aprendizaje de Contenidos Abiertos, y para poder publicar, descargar, compartir y reutilizar dichos objetos se utilizan los Repositorios de Objetos de Aprendizaje de Contenidos Abiertos, los cuales almacenan de forma estandarizada información adicional del recurso (metadatos) para facilitar los procesos de búsqueda.

Otra importante tecnología que ha surgido como producto del crecimiento de la internet y también de la necesidad de la interoperabilidad entre aplicaciones son los Servicios Web, los cuales se pueden desarrollar bajo distintas arquitecturas como por ejemplo Representational State Transfer (REST). Los Servicios Web buscan estandarizar el transporte, la codificación y el protocolo de intercambio de información, permitiendo de esta forma la interacción entre aplicaciones desarrolladas bajo lenguajes y plataformas distintas. En este sentido, el objetivo principal del presente Trabajo Especial de Grado consiste en la construcción de un Repositorio de Objetos de Aprendizaje y Contenidos Abiertos Reutilizables bajo una arquitectura REST, cuyo acceso será abierto y estará dirigido a todas las personas interesadas en el uso de OACA, tanto de la Universidad Central de Venezuela como de otras universidades. La arquitectura REST mencionada anteriormente va a facilitar la comunicación con otras aplicaciones que sean desarrolladas a futuro o que estén operativas actualmente.

Para llevar a cabo el objetivo anteriormente mencionado, se describe de forma general, los capítulos que se presentan en este documento.

Capítulo I, Problema de investigación: se describe la problemática en la cual se enmarca el presente Trabajo Especial de Grado. Así mismo, se propone una solución y se justifica la misma; se define el objetivo general, los objetivos específicos

y el alcance. Adicionalmente, se explica la implementación que se llevó a cabo de la Programación Extrema (XP), la cual se utilizó como metodología de desarrollo.

Capítulo II, Marco Conceptual: se presentan las bases teóricas de la investigación. Este capítulo abarca todos los conceptos, referencias y definiciones relacionadas con los OACA y los ROACA, así como la descripción de las herramientas tecnológicas que se utilizaron para el desarrollo del Repositorio, como el framework Symfony2, la Arquitectura REST, el patrón Modelo Vista Controlador (MCV), el sistema manejador de bases de datos Postgresql, entre otros.

Capítulo III, Marco Aplicativo: en este capítulo se describen cada una de las iteraciones que se llevaron a cabo durante todo el proceso de desarrollo, especificando la planificación y ejecución de las tareas correspondientes a la metodología XP.

Capítulo IV, Resultados: este capítulo muestra los resultados obtenidos luego de culminar el desarrollo del Trabajo Especial de Grado y se describen resumidamente todas las funcionalidades del Repositorio con sus respectivas interfaces.

Capítulo I. Problema de Investigación

En este capítulo se describe la situación actual en cuanto a la gestión de los Objetos de Aprendizaje de Contenidos Abiertos (OACA) dentro de la Unidad de Educación a Distancia y la Facultad de Ciencias de la Universidad Central de Venezuela, así como la propuesta del Repositorio de Objetos de Aprendizaje y Contenidos Abiertos Reutilizables para optimizar dicha gestión y ampliar su alcance a toda la Universidad Central de Venezuela y otras universidades. Se presentan los objetivos del Trabajo Especial de Grado, la importancia y justificación del mismo.

1. Planteamiento del Problema

En la Unidad de Educación a Distancia y la Facultad de Ciencias de la Universidad Central de Venezuela ha surgido la iniciativa de desarrollar diversos proyectos relacionados con el diseño y construcción de OACA, un caso particular es la asignatura de Objetos de Aprendizaje: aspectos pedagógicos y aspectos tecnológicos, donde los estudiantes ponen en práctica los conocimientos que adquieren a lo largo de la materia y desarrollan recursos de alta calidad dirigidos a apoyar los procesos de enseñanza aprendizaje en diversas áreas de conocimiento. Dichos recursos se almacenan en un computador de escritorio de la Unidad de Educación a Distancia que funciona como servidor y que no dispone de ninguna interfaz de usuario adecuada; y su acceso está limitado únicamente a aquellas personas que conozcan el enlace (http://ead.ciens.ucv.ve/oa/).

Para que un usuario pueda acceder a los recursos educativos almacenados, se le debe facilitar la dirección del servidor, y luego el usuario debe realizar una búsqueda manual hasta encontrar el recurso que sea de su interés, a partir de una lista que contiene el nombre de todos los archivos (Ver Figura 1). En esta lista no se especifica ningún tipo de metadato ni información adicional para los OACA, por lo tanto, se debe intuir el contenido de los mismos a partir de su nombre y, a medida que va aumentando el número de archivos y carpetas, se hace complicado y tedioso ubicar cualquier material que sea de interés. Tampoco existen políticas de seguridad, como por ejemplo el registro y validación de usuarios y, por otro lado, son muchos los casos en donde el archivo inicial o “index” no se encuentra en la carpeta raíz y es necesario buscarlo en diversas subcarpetas para visualizar de forma correcta el recurso. Además

de esto, la forma como están alojados actualmente los OACA impide compartir los recursos con otras escuelas u otras universidades y no se fomenta la reutilización, por lo tanto, el alcance de los OACA se limita solo al uso interno de Universidad Central de Venezuela, específicamente a la Escuela de Computación de la Facultad de Ciencias.

Por tales motivos, el problema que se presenta es la falta de un espacio accesible, robusto, seguro y usable que permita la publicación, búsqueda y reutilización de OACA desarrollados dentro y fuera de la Universidad Central de Venezuela.

Por todo lo antes expuesto se plantea como pregunta de investigación, cómo se puede crear e implementar un mecanismo para llevar a cabo la gestión de los OACA de forma eficiente.

Figura 1. Almacenamiento actual de los OACA en la Unidad de Educación a Distancia

2. Justificación

Los Repositorios de Objetos de Aprendizaje de Contenidos Abiertos (ROACA) almacenan una gran cantidad de recursos educativos de forma organizada y categorizada para facilitar el proceso de búsqueda y permiten aprovechar al máximo la característica de reutilización de los OACA, ya que dichos recursos pueden ser descargados las veces que sea necesario y, gracias a los metadatos y los estándares por los que se rigen, se pueden reutilizar en múltiples entornos educativos. Por otra parte, se incrementa la disponibilidad y accesibilidad de los contenidos de aprendizaje, lo cual ayuda a los docentes a centrarse más en la información a transmitir y no en las herramientas que se usarán para transmitir dicha información. Al mismo tiempo y gracias a la Internet, los recursos digitales contenidos en los Repositorios pueden ser accedidos desde cualquier ubicación geográfica en cualquier momento y motivan a algunos estudiantes a indagar en ciertas áreas del conocimiento, ya que ofrecen alternativas de aprendizaje no tradicionales.

Por estas razones, surgió la necesidad de crear un ROACA; para estudiantes, docentes y cualquier persona involucrada en el área; que ofreciera las ventajas señaladas anteriormente y que proveyera búsquedas eficientes e importación y exportación de OACA al Repositorio, con un rendimiento comparable al de otros Repositorios existentes y que se apoyara en una arquitectura orientada a servicios de tipo Representational State Transfer (REST) para mantener la conexión con otras aplicaciones.

3. Objetivo General

Construir un Repositorio de Objetos de Aprendizaje de Contenidos Abiertos bajo una arquitectura orientada a servicios de tipo Representational State Transfer (REST) para la Universidad Central de Venezuela, que permita la gestión y reutilización de los recursos educativos de forma libre y gratuita.

4. Objetivos Específicos

a) Definir las funcionalidades del ROACA.
b) Seleccionar las tecnologías y herramientas empleadas para el desarrollo del ROACA.
c) Diseñar la interfaz de usuario del ROACA.

d) Desarrollar la interfaz de usuario del ROACA.
e) Diseñar el modelo conceptual de la Base de Datos.
f) Implementar el modelo físico de la Base de Datos en el Sistema Manejador de Bases de Datos Postgresql.
g) Realizar pruebas de funcionalidad y usabilidad al ROACA
h) Implementar el ROACA en el servidor de la Unidad de Educación a Distancia de la Facultad de Ciencias de la UCV.

5. Alcance

Este ROACA está dirigido tanto a los alumnos como a los docentes de la Universidad Central de Venezuela o de otra universidad o institución, así como también a cualquier persona interesada en el uso de OACA. El Repositorio cuenta con un Módulo de Usuario, el cual no tiene ninguna restricción de acceso, y un Módulo de Administración que solo puede ser accedido por el Usuario Administrador. El Módulo de Usuario cumple con las siguientes funcionalidades:

a) Registrar usuarios en el ROACA.
b) Almacenar OACA, para ello, se deben llenar una serie de metadatos, algunos son obligatorios y otros son opcionales.
c) Visualizar OACA directamente en el navegador o descargar el archivo que contiene el OACA.
d) Realizar búsquedas por categoría de los OACA.
e) Realizar búsquedas avanzadas de los OACA, para ello, se deben completar algunos campos como Título, Autor, Año y Categoría.
f) Listar los OACA de un usuario, con la posibilidad de editarlos o eliminarlos.
g) Realizar comentarios sobre los OACA.
h) Visualizar los OACA más recientes y los más descargados.

Por otro lado, las funcionalidades que corresponden al Módulo de Administración son las siguientes:

a) Crear nuevas Categorías y Subcategorías para clasificar a los OACA.
b) Aprobar o rechazar los OACA que se deseen publicar en el Repositorio.
c) Aprobar o rechazar los comentarios que se deseen publicar en el Repositorio.
d) Colocar una valoración de estrellas a los OACA que se publiquen en el Repositorio.

e) Gestionar los usuarios del Repositorio.

6. Metodología de Desarrollo

Se propuso utilizar la Programación Extrema (XP) como metodología de desarrollo ágil, la cual busca aumentar la productividad a la hora de desarrollar aplicaciones, ya que se basa en una serie de tecnologías de desarrollo de software, donde se le da prioridad a aquellas actividades que generan un resultado directo, disminuyendo el esfuerzo en las actividades menos relevantes. El objetivo de XP es facilitar al máximo el desarrollo de sistemas, aplicando el sentido común.

6.1 Principios básicos de la Programación Extrema

La Programación Extrema se basa en 12 principios básicos que se describen a continuación (Zambrano & León, 2010):

a) El principio de pruebas: se tiene que establecer un período de pruebas del programa donde se definen las entradas al sistema y los resultados esperados. Es muy recomendable automatizar estas pruebas para poder hacer varias simulaciones del sistema en funcionamiento. Para hacer estas simulaciones se pueden utilizar Ambientes de Prueba.

b) Proceso de planificación: en esta fase, el usuario debe escribir sus necesidades, definiendo las actividades que realizará el sistema. Se crea entonces un documento llamado “Historias de Usuario”.

c) El cliente en el sitio: se le da el poder para determinar los requerimientos, definir la funcionalidad, señalar las prioridades y responder las preguntas de los programadores. Esta fuerte interacción directa con el desarrollador disminuye el tiempo de comunicación y la cantidad de documentación, junto con los altos costes de su creación y mantenimiento. Este representante del cliente debe estar con el equipo de trabajo durante toda la realización del proyecto.

d) Programación en parejas: requiere que todos los programadores XP escriban su código en parejas, compartiendo un solo computador. De acuerdo con los experimentos, este principio puede producir mejores aplicaciones, de manera consistente, a iguales o menores costes. Aunque la programación en pareja puede no ser para todo el mundo, las estadísticas del uso de XP demuestran un gran éxito.

e) **Integración continua:** permite al equipo hacer un rápido progreso implementando las nuevas características del software. En lugar de crear "bulids" (o versiones) estables de acuerdo a un cronograma establecido, los equipos programadores XP pueden reunir su código y reconstruir el sistema varias veces al día. Esto reduce los problemas de integración comunes en proyectos largos y estilo cascada.

f) **Refactorización:** permite a los equipos de programadores XP mejorar el diseño del sistema a través de todo el proceso de desarrollo. Los programadores evalúan continuamente el diseño y recodifican lo necesario. La finalidad es mantener un sistema enfocado a proveer el valor de negocio mediante la minimización del código duplicado y/o ineficiente.

g) **Entregas pequeñas:** colocan un sistema sencillo en producción rápidamente que se actualiza de forma rápida y constante permitiendo que el verdadero valor de negocio del producto sea evaluado en un ambiente real. Estas entregas no pueden pasar las 2 o 3 semanas como máximo.

h) **Diseño simple:** se basa en la filosofía de que el mayor valor de negocio es entregado por el programa más sencillo que cumpla los requerimientos. El Diseño Simple se enfoca en proporcionar un sistema que cubra las necesidades inmediatas del cliente, ni más ni menos. Este proceso permite eliminar redundancias y rejuvenecer los diseños obsoletos de forma sencilla.

i) **Metáfora:** es una descripción general del sistema, que se establece al comenzar el proyecto, que fortifica la integridad conceptual, ayuda a guiar el proceso de desarrollo y mantiene una visión unificada entre los actores. La Metáfora determina un estándar en el vocabulario que será utilizado por los programadores y el cliente, que luego ayudará a establecer las clases y métodos del sistema.

j) **Propiedad colectiva del código:** un código con propiedad compartida. Nadie es el propietario de nada, todos son el propietario de todo. Este método difiere en mucho a los métodos tradicionales en los que un simple programador posee un conjunto de código. Los defensores de XP argumentan que mientras haya más gente trabajando en una pieza, menos errores aparecerán.

k) **Estándar de codificación:** define la propiedad del código compartido así como las reglas para escribir y documentar el código y la comunicación entre las diferentes piezas de código desarrolladas por diferentes equipos. Los programadores las han de seguir de tal manera que el código en el sistema se vea como si hubiera estado escrito por una sola persona.

l) **La semana de 40 horas:** la programación extrema sostiene que los programadores cansados escriben código de menos calidad. Minimizar las horas extras y mantener los programadores frescos, generará código de mayor calidad. Los autores afirman que está bien trabajar horas extras pero sólo cuando es necesario, y que no ocurra por más de dos semanas seguidas.

6.2 Actividades de la Programación Extrema

La Programación Extrema está compuesta de una serie de actividades o prácticas que pueden agruparse en cuatro bloques o fases (Zambrano & León, 2010).

a) **Planificación:** Se escriben las Historias de Usuario, las cuales deben ser hechas por el cliente, de tal manera, que entienda perfectamente cada una de ellas y se pueda hacer una estimación de tiempo para el desarrollo del mismo. Al reunir suficientes historias, se forma el llamado “Plan de Liberación”, el cual define de forma específica los tiempos de entrega de la aplicación, para recibir retroalimentación por parte del usuario. Por regla general, cada historia suele necesitar de una a tres semanas de desarrollo. Son muy importantes y tienen que ser una constante las reuniones periódicas durante esta fase de planificación. Pueden ser a diario, con todo el equipo de desarrollo para identificar problemas, proponer soluciones y señalar aquellos puntos a los que se les ha de dar más importancia por su dificultad o por su punto crítico.
b) **Diseño:** Es en esta fase donde se pone en práctica el principio del Diseño Simple. Si alguna parte del sistema es de desarrollo complejo, lo apropiado es dividirla en varias. Si hay fallos en el diseño, deben ser corregidos lo antes posible. Se crea la Metáfora del sistema, que se basa en un Gráfico que muestre su arquitectura y una descripción de las funcionalidades, de manera que pueda ser comprendida por cualquier persona. Se escriben las tarjetas CRC (Clase, Responsabilidad y Colaboración) que ayudan al equipo a definir actividades durante el diseño del sistema. Cada tarjeta representa una clase en la programación orientada a objetos y define sus responsabilidades (lo que ha de hacer) y las colaboraciones con las otras clases (cómo se comunica con ellas).La Refactorización entra también en esta fase. Se debe cambiar un diseño si se considera necesario, o eliminar un código que ya no se necesite. El propósito es minimizar en lo posible, ineficiencias del código o duplicados del mismo.

c) **Codificación:** El equipo de trabajo procede a plasmar las ideas a través del código, y el representante del cliente debe mantenerse presente en todo el desarrollo del proyecto, para escribir las Historias de Usuario, participar en la elección de las que entrarán en el Plan de Liberación, y hacer las pruebas de funcionalidad. La idea es usar el tiempo del cliente para hacer estas tareas, en vez de para crear una minuciosa y detallada descripción de requisitos. Debe regirse bajo el principio del Estándar de Codificación mencionado anteriormente, para asegurar la consistencia y facilitar la comprensión y refactorización del código. Es en esta fase donde se pone en práctica la Integración Continua de los últimos avances del proyecto, las Entregas Pequeñas al cliente, la propiedad colectiva del código y la Programación en Parejas, para asegurar una alta calidad del trabajo.

d) **Pruebas:** Todo el código debe tener pruebas unitarias, y deben ser aprobadas antes de ser lanzado. Cuando se encuentra un error de codificación, se desarrollan pruebas para evitar volver a caer en el mismo. Se realizan pruebas de funcionalidad frecuentemente, publicando los resultados de las mismas. Estas pruebas son generadas a partir de las Historias de Usuario elegidas para la iteración. Cuando se pasa la prueba, se considera que la historia correspondiente se ha completado.

6.3 Implementación de la Programación Extrema

A continuación se describe cómo se aplicaron las 4 actividades de XP al desarrollo del Repositorio. Debe tenerse en cuenta que no se cumplieron algunos de los principios de la metodología dadas las características de este proyecto.

6.3.1 Planificación

Lo primero que se llevó a cabo, fue una reunión con los docentes de la materia Objetos de Aprendizaje: aspectos pedagógicos y aspectos tecnológicos, para poder determinar las funcionalidades que debía cumplir el Repositorio y en base a ello escribir una lista de Historias de Usuario. Para cumplir con el Plan de Liberación de esta fase, se definió una bitácora de desarrollo para la planificación de los requerimientos en base a días o semanas. Se hizo un desarrollo por módulos, cada uno con una fecha inicial, fecha estimada de culminación, precedencia de la tarea específica, número de días estimados y número de días realizados. En este sentido, el formato de la bitácora se observa en la Tabla 1.

Tabla 1. Tabla para la bitácora de desarrollo

Nº	Tarea	Precede	Fecha Inicio	Fecha Fin	Días estimados	Días realizados

Se trabajó en función del tiempo, implementando cada Historia de Usuario en una cantidad de días estimados (entre 1 y 3 semanas como indica XP). El formato de estas historias se observa en la Tabla 2.

Tabla 2. Formato para las Historias de Usuario

Número:	**Prioridad:**	**Estimación:**
Nombre:		
Descripción:		

La planificación se realizó en iteraciones, en las cuales se dividió la implementación de las Historias de Usuario, y las entregas al cliente para correcciones y realización de pruebas que se llevaron a cabo al final de cada iteración.

6.3.2 Diseño

En esta etapa se realizó el análisis global del sistema, creando una representación sencilla de las partes que la conforman y cómo se comunican entre ellas. Esto cumple con el principio de XP, de escribir una metáfora que represente de forma general, el resultado que se busca en el desarrollo del sistema.

Para llevar a cabo el principio del Diseño Simple, se mantuvieron conversaciones con los docentes de la materia de Objetos de Aprendizaje: aspectos pedagógicos y aspectos tecnológicos proponiendo soluciones entre ambas partes, que definieron las funcionalidades de cada módulo del sistema, evitando las complejidades en lo posible. Las tarjetas CRC no aplican por ser un solo desarrollador el que estuvo a cargo del Repositorio. Sin embargo, el principio de Refactorización se llevó a cabo, en vista de la revisión constante del código y de la documentación, refinando los detalles en las iteraciones subsecuentes.

6.3.3 Codificación

Lo primero que se estableció en esta etapa, fue un estándar básico de codificación para facilitar la programación y la comprensión del código. El estándar se definió de la siguiente manera:

a) Los nombres de las entidades se escriben en singular y con letra mayúscula al comienzo, y los atributos se escriben en letra minúscula. Si el nombre de la entidad está conformado por más de una palabra, se coloca la primera letra de cada palabra en mayúscula y se omiten las conjunciones y/o preposiciones. Por ejemplo, el nombre de la entidad correspondiente a Ciclo de Vida es CicloVida.
b) Los nombres de los controladores del Repositorio llevan el mismo nombre de la entidades a la cuales pertenecen pero con el sufijo "Controller". Por ejemplo, el controlador correspondiente a la entidad Usuario lleva el nombre de UsuarioController.
c) Todas las entidades deben tener sus atributos privados o protegidos, y deben poseer las funciones "get" y "set" necesarias para acceder o modificar cada uno de sus atributos.
d) Las claves primarias de todas las tablas son autoincrementales y se llaman "id".
e) Todo el código del Repositorio debe estar comentado lo más posible, tanto en las entidades como en los controladores y las vistas.
f) Los nombres de las tablas de la base de datos están en singular. Además, los nombres de las tablas que hacen la relación entre otras dos, están conformados exclusivamente de dos palabras, las cuales son los nombres de las tablas que están relacionando.
g) Todas las clases de Symfony2 donde se definen funciones especiales para consultas a Base de Datos, se guardan en una carpeta llamada "Repository".

Referente al principio de Entregas pequeñas, se hicieron entregas a los docentes cada dos semanas con los últimos avances, de forma que se pudieron realizar pruebas constantemente. Además de esto, se mantuvo una frecuente comunicación con los docentes vía e-mail, pudiendo modificar los requerimientos o prioridades, y aclarar las dudas rápidamente.

Las 40 horas semanales se distribuyeron en los 7 días de la semana, teniendo jornadas nocturnas de lunes a viernes y diurnas los fines de semana. El Repositorio fue desarrollado por una sola persona, por lo que la Programación en Parejas no aplica, sin embargo, se revisó el código constantemente para obtener una aplicación lo más refinada y eficiente posible.

6.3.4 Pruebas

En esta fase se realizaron las pruebas funcionales de cada Historia de Usuario, asegurando de esta forma, el buen funcionamiento de dicha implementación. Los restantes tipos de pruebas, como las unitarias y de integración, no se incluyeron en esta fase para evitar retrasos en los tiempos de entrega en vista de ser una sola persona a cargo de la programación.

Al finalizar cada iteración, se escribieron las pruebas de funcionalidad indicando los casos de prueba, sus resultados esperados y los obtenidos. Para cumplir este aspecto, se utilizó el siguiente formato que se observa en la Tabla 3.

Tabla 3. Formato de registro de pruebas del lado del cliente

Nº Caso de Prueba	Módulo	Caso de Prueba	Resultado Esperado	Resultado Obtenido

Capítulo II. Marco Conceptual

En este capítulo se describen las bases conceptuales sobre las cuales se desarrolla el presente Trabajo Especial de Grado. En primer lugar, se presenta todo el material referente a OACA, como lo son las definiciones, características, funciones, componentes, clasificación y estándares. Posteriormente, se describen los aspectos más relevantes relacionados con los ROACA y se analizan algunos Repositorios ya existentes. Finalmente, se describen las principales herramientas tecnológicas empleadas durante el proceso de desarrollo, entre las cuales destaca la implementación de Servicios Web a través de la Arquitectura REST.

1. Objetos de Aprendizaje

La razón de ser de los Repositorios de Objetos de Aprendizaje (ROA) es la de albergar los Objetos de Aprendizaje (OA) para permitir su reutilización. Por esta razón, se toma como punto de partida de este Marco Conceptual, el tema de OA.

1.1. Definiciones

Existen múltiples definiciones de OA, algunas de ellas son:

Mínima estructura independiente que contiene un objetivo, una actividad de aprendizaje, metadatos y un mecanismo de evaluación, la cual puede ser desarrollada con las tecnologías de la infocomunicación (TIC) con el fin de posibilitar su reutilización, interoperabilidad, accesibilidad y duración en el tiempo (APROA, 2005).

Elementos de un nuevo tipo de instrucción basada en el computador y fundamentada en el paradigma computacional de la orientación a objetos (Wiley, Connecting Learning Objects to Instructional Desig Theory: a Definition, a Metaphor, and Taxonomy, 2000).

Pequeñas unidades standalone de instrucción, que pueden ser etiquetadas y gestionadas en un Repositorio, y que más tarde pueden ser ensambladas junto con otras para formar módulos educativos (Singh, 2004).

Tomando en cuenta las definiciones de los autores anteriores podemos decir que un OA es cualquier recurso tecnológico que sirve de apoyo a un proceso de

enseñanza-aprendizaje y que tiene la capacidad de ser reutilizado en diferentes contextos ya que se fundamenta en el paradigma orientado a objetos y posee metadatos que facilitan su identificación, almacenamiento y recuperación.

1.2 Objetos de Aprendizaje de Contenidos Abiertos

Tomando en cuenta la obra de Wiley (2009) se puede definir a los OACA como aquellos OA que están desarrollados con herramientas de software libre y además cumplen con las siguientes características:

a) Reusabilidad: los usuarios tienen derecho de usar total o parcialmente el OA para sus propios propósitos.
b) Redistribución: los usuarios tienen derecho de compartir el OA con otros usuarios.
c) Revisión: los usuarios pueden adaptar, modificar, traducir o cambiar el contenido del OA.
d) Remezcla: los usuarios pueden tomar dos o mas OA existentes y combinarlos para crear un nuevo recurso.

Estas nuevas posibilidades dan cabida a que pueda haber un aumento en la calidad de los contenidos en línea debido a que los docentes o instructores pueden elaborar recursos, editar recursos existentes y/o mejorarlos.

1.2.1 Características

Tomando en cuenta la obra de Galeana (2004) se puede afirmar que los OACA cumplen con las siguientes características:

a) Capacidad de ser reutilizado.
b) Capacidad de generar aprendizaje.
c) Proporcionar interoperabilidad (se pueden integrar en estructuras y plataformas diferentes).
d) Facilidad de acceso a contenidos apropiados en tiempos apropiados.
e) Vigencia de la información de los OACA, sin necesidad de nuevos diseños.
f) Capacidad de construir contenidos y OACA nuevos, derivados de el.
g) Capacidad de ser actualizados o modificados, aumentando sus potenciales a través de la colaboración.
h) Flexibilidad, versatilidad y funcionalidad, con elasticidad para combinarse en diversas propuestas de áreas del saber diferentes.

i) Interactividad, permitiendo generar actividades y comunicación entre los sujetos involucrados.
j) Acoplamiento a las necesidades de aprendizaje de cada individuo.
k) Autocontención conceptual, capacidad de auto explicarse y posibilitar experiencias de aprendizaje integral.
l) Escalabilidad, permitiendo la integración con estructuras más complejas.

Por otro lado, Herández (2009) menciona que la fundamentación de un OACA se soporta en lo pedagógico, lo tecnológico y la interacción humano computador en estrecha relación, y plantea las siguientes dimensiones:

a) Dimensión pedagógica: se establece la intención educativa que permita la construcción, recreación y difusión del conocimiento. Dentro de las características más resaltantes se pueden mencionar: diversidad de estilos de aprendizaje, intencionalidad de aprendizaje, contenidos de aprendizaje, actividades de aprendizaje, recursos, interactividad y evaluación.
b) Dimensión tecnológica: se sustenta en los OA que abarcan aspectos tecnológicos, a través de software, donde se imbrican la reusabilidad, la interoperabilidad, la accesibilidad, la portabilidad, la flexibilidad y la granularidad, entre otros.
c) Dimensión de interacción humano computador: es la acción constante del sujeto con el recurso tecnológico. Los OACA deben poder motivar e interesar a los aprendices, para propiciar el trabajo con el mismo y así impulsar el aprendizaje. Se debe tener presente que como un recurso digital debe cumplir con ciertos atributos que lo hagan atractivo al aprendiz y éste no los rechace, como lo es el uso apropiado de los colores, las fuentes, presentación y disposición de la información, navegabilidad, entre otros.

1.2.2 Funciones

En base a la obra de Galeana (2004) se pueden listar las siguientes funciones de los OACA:

a) Favorecer la generación, integración y reutilización de OACA.
b) Estimular el estudio autogestivo.
c) Promover el trabajo colaborativo.
d) Posibilitar el acceso remoto a la información y contenidos de aprendizaje.

e) Posibilitar la integración de diferentes elementos multimedia a través de una interfaz gráfica.
f) Contribuir a la actualización permanente de profesores y alumnos.
g) Estructuración de la información en formato de hipertexto.
h) Facilitar la interacción de diferentes niveles de usuarios (administrador, diseñador, alumno).

Las funciones de un OACA varían de acuerdo al uso que cada usuario le vaya a dar. Según las expectativas de cada quien, un OACA debe cumplir con ciertas funcionalidades o no. Sin embargo, la función principal y que siempre deben cumplir los OACA, es la reutilización, la cual promueve el trabajo colaborativo y evita la creación de nuevos OACA, cuando existen otros con las mismas características que han sido elaborados con anterioridad.

1.2.3 Componentes Externos: Los Metadatos

Están constituidos por información externa al OACA, y reciben el nombre de metadatos, los cuales son identificadores que describen los atributos y propiedades de un OACA con el fin de optimizar su almacenamiento, búsqueda, selección y recuperación. Los metadatos pueden definirse también como datos sobre los datos. En el campo de las tecnologías de aprendizaje, los metadatos son utilizados para la descripción de recursos educativos. En definitiva, los metadatos educativos son el medio utilizado para la descripción, caracterización y catalogación de los recursos educativos. Esta última característica permite que los metadatos puedan ser utilizados en los sistemas de intermediación para la búsqueda y localización eficiente de los recursos que describen. (Rodriguez, 2001)

Según Rodriguez (2001), los metadatos pueden clasificarse de acuerdo a su objetivo y de acuerdo a su tipo de contenido. A continuación se explica cada uno de ellos:

De acuerdo a su objetivo:

a) Metadatos descriptivos: Brindan un resumen del contenido del OACA y lo identifican. Permiten la búsqueda y recuperación del OACA dentro de algún sistema local o internet.

b) **Metadatos estructurales:** Facilitan la navegación y presentación de los OACA, proporcionan información sobre la estructura interna de los OACA, describen posibles relaciones con otros OACA y describen posibles relaciones entre recursos dentro del mismo OACA.
c) **Metadatos administrativos:** Facilitan la gestión y procesamiento de los OACA, tanto a corto como a largo plazo; incluyen datos técnicos sobre autor, organización, administrador, fecha de creación y control de calidad e incluyen gestión de derechos, así como requisitos de control de acceso y utilización.

De acuerdo a su tipo de contenido:

a) **Objetivos:** Como el tamaños de un archivo, el nombre del creador, la fecha de creación, la versión, sus requerimientos mínimos, entre otros.
b) **Subjetivos**: Como palabras claves, descriptores, a quién va dirigido, qué usos se recomienda, entre otros.

Los metadatos objetivos presentan pocas dificultades para su manejo, basta con utilizar un determinado formato para que todo el que busque o utilice el OACA los entienda. Por ejemplo, para la colocación de la fecha de creación del OACA, se puede definir un formato específico, bien sea dd/mm/aa o mm/dd/aa, de forma que no se preste a confusión cuando algún usuario lea esta información. En cambio, los metadatos subjetivos requieren de un esfuerzo mayor por parte de los creadores de los OACA. Por ejemplo, sería difícil colocar el nivel de estudios al que pertenece el OACA, ya que existen diferencias entre los términos utilizados para referirse a estos niveles de estudios en los diferentes países de habla hispana; algunos países hablan de secundaria, mientras que otros emplean el término nivel medio básico.

Otro aspecto importante de los metadatos es el referente al idioma. En el mundo académico y de investigación el idioma dominante es el inglés, por lo tanto, para que un OACA pueda ser visible desde muchos países alrededor del mundo, es importante que los metadatos aparezcan en inglés. Así mismo, al momento de realizar una búsqueda, lo más conveniente es usar descriptores en inglés para poder obtener la mayor cantidad de resultados posibles.

1.2.4 Clasificación de acuerdo a su composición

Se puede realizar una clasificación de los OACA de acuerdo a su composición de la siguiente manera: (Wiley, Connecting Learning Objects to Instructional Desig Theory: a Definition, a Metaphor, and Taxonomy, 2000).

a) **Fundamentales:** Son OACA que no pueden ser descompuestos o subdivididos. Generalmente corresponden con una ayuda visual que permite ejemplificar o mostrar la información. Por ejemplo, video, audio, imagen, texto, entre otros.
b) **Combinados-Cerrados:** Son OACA compuestos por recursos digitales que están combinados, pero que no pueden ser accedidos de manera individual para su reutilización en otros OACA. Por ejemplo, un video instruccional de guitarra acompañado con un OACA de audio y un OACA de texto integrado que muestra las partituras a medida que transcurre el video.
c) **Combinados-Abiertos:** Son OACA compuestos por recursos digitales que pueden ser accedidos directamente para su reutilización**.** Por ejemplo, una página Web que contenga fotos que muestren la colocación de los dedos para diferentes acordes de guitarra, una pista de audio y partituras en formato de texto. Todos los OACA presentes en la página Web pueden ser extraídos y combinados con otros OACA o reutilizados en otros contextos.
d) **Presentación Generativa:** Este tipo de OACA es más complejo y tienen una alta reusabilidad intra-contextual (pueden ser reutilizados en contextos similares), pero tienen una baja reusabilidad extra-contextual (no pueden ser usados en dominios para los cuales no fueron creados). Proporcionan la estructura lógica para combinar, generar o reorganizar los OACA de nivel inferior (fundamentales y combinados cerrados), y son capaces de recibir y pasar mensajes de otros OACA. Por ejemplo, un applet de Java capaz de generar gráficamente un conjunto de notas y posicionarlas adecuadamente dentro de un pentagrama para explicar el pasaje de una melodía de guitarra.
e) **Generativos Instruccionales:** Este tipo de OA está relacionado con ejercicios prácticos a desarrollar, es decir, los OACA de ésta categoría se encargan de instruir y proveer prácticas. Están desarrollados para evaluar la capacidad de un aprendiz de recordar una serie de pasos. Proporcionan la estructura lógica para combinar los OACA de nivel inferior y evaluar las interacciones del estudiante con estas combinaciones. Por ejemplo, se presenta la teoría de intervalos y composición de los acordes dentro de las escalas musicales, y

luego se pide al aprendiz que en base a una tonalidad dada, escoja de una lista de alternativas, el conjunto de acordes correspondientes a la escala. Los OACA generativos instruccionales poseen reusabilidad intra y extra contextual.

En la Tabla 4 se muestra un cuadro comparativo de los diferentes OACA según la taxonomía explicada anteriormente. La comparación se hace en base a ciertas características.

Tabla 4. Taxonomía según composición de los OACA. Willey (2000)

	Fundamentales	Combinados-Cerrados	Combinados-Abiertos	Presentación Generativa	Generativos Instruccionales
1	Uno	Pocos	Muchos	Pocos-Muchos	Pocos-Muchos
2	Uno	Fundamentales o Combinados-Cerrados	Todos	Fundamentales o Combinados-Cerrados	Fundamentales, Combinados-Cerrados
3	No se aplica	No	Si	Si/ No	Si/ No
4	Mostrar	Prediseño de instrucciones o prácticas	Prediseño de instrucciones y/o prácticas	Mostrar	Instrucciones o prácticas generadas por computador
5	No	No	Si	Si/ No	Si
6	No se aplica	No o respuestas basadas en ítems	No, o instrucciones de dominio específico y de estrategias de evaluación	Dominio específico y estrategias de presentación	Presentaciones, estrategias instruccionales y de evaluación independientes del dominio
7	Alto	Medio	Bajo	Alta	Alta
8	Bajo	Bajo	Medio	Alta	Alta

Leyenda

Nro	Característica
1	Número de elementos individuales combinados
2	Tipo de OACA combinados
3	OACA como componentes reusables. Describe si un OACA puede ser reutilizado como parte de otros OACA.
4	Funciones comunes. Forma en la cual un OACA es generalmente usado.
5	Dependencia con otros OACA
6	Tipo de lógica contenida en el OACA. Describe los Algoritmos y procedimientos comunes de un OACA
7	Potencial para reuso intercontextual. Número de contextos en los que el OACA puede ser usado.
8	Potencial para reuso intracontextual. Número de veces que un OACA puede ser usado en el mismo contexto

1.2.5 Ventajas y desventajas

A pesar de que los OACA constituyen una manera innovadora de desarrollar recursos didácticos que apoyen los procesos de enseñanza y aprendizaje, no todas sus características resultan positivas para quién los usa. Los OAcA tienen tanto ventajas como desventajas, las cuales se explican a continuación.

Ventajas

a) Pueden ser utilizados en contextos diferentes, por lo tanto evita la necesidad de crear OACA ya existentes.
b) Tienen la capacidad de acoplarse con otros OACA para generar OACA más complejos.
c) Se adaptan a las necesidades específicas y al ritmo de aprendizaje del alumno.
d) Pueden ser accedidos fácilmente en el momento que se desee.
e) Son independientes de la tecnología empleada, por lo tanto no necesitan ser rediseñados cuando ocurren cambios tecnológicos.
f) Se acceden independientemente de la plataforma y hardware.
g) No requieren de otros OACA para cumplir con su objetivo de aprendizaje.
h) Disminuyen el tiempo invertido en el desarrollo del material didáctico, para de esta manera centrarse en la información a transmitir y no en las herramientas que se usarán para transmitir dicha información.
i) Ofrecen caminos de aprendizaje alternativos.
j) No existen restricciones legales ni económicas para hacer uso de ellos.

En general, los OACA reducen considerablemente el tiempo de preparación de recursos de aprendizaje de calidad, y le ofrecen la posibilidad al docente de elegir aquellos que se adapten fácilmente a su estilo de enseñanza, a la diversidad de estilos de aprendizaje de sus alumnos y a su programa de estudios. Por otra parte, los OACA combinan una variedad de herramientas tecnológicas para la elaboración de materiales didácticos innovadores, despertando el interés de algunos estudiantes por el uso de estos nuevos recursos.

Desventajas

a) Requieren de tecnología de información, incluyendo acceso a internet de banda ancha y un sistema de gestión de aprendizaje.

b) Para publicar información es necesario contar con un espacio en un servidor.
c) La transmisión de conocimiento es en un solo sentido: de instructor a aprendiz.
d) El aprendizaje de un tema está limitado a la disponibilidad de OACA que lo aborden.
e) Falta de experiencia en el uso de OACA en el proceso de instrucción.
f) Es necesario que el alumno se sienta cómodo utilizando el computador como herramienta de instrucción.

Las principales desventajas de los OACA radican por un lado en el consumo de recursos, sin embargo, sabemos que en la actualidad el acceso a la tecnología es masivo, y la presencia en el hogar de un computador con conexión a internet es algo muy común en las sociedades del siglo XXI. Y por otro lado, no hay suficiente experiencia por parte de docentes y aprendices en el uso de los OACA. Para el uso de OACA, los docentes más allá de ser transmisores de conocimientos deben ejercer el rol de tutores, orientando a los estudiantes para adquirir las destrezas necesarias para desenvolverse en los nuevos ambientes tecnológicos.

2. Estándares

Existen diversos estándares que buscan unificar la estructura y lenguaje de los metadatos de los OACA, así como también el empaquetado de los mismos. La finalidad de estos estándares es mejorar la compatibilidad y la portabilidad de las distintas descripciones asociadas a los distintos OACA o recursos, como la independencia de barreras técnicas, culturales y de lenguaje que en cierto modo limitan la reutilización, intercambio o difusión en distintos ámbitos. (Instituto Nacional de Estadísticas, Geografía e Informática de México, 2003). A continuación se presentan dos de los estándares más importantes.

2.1 Learning Object Metadata

El origen del estándar Learnig Object Metadata (LOM) se remonta al año 1997, cuando el consorcio EDUCOM (agrupación de instituciones americanas) empezó a desarrollar el proyecto IMS, con el objetivo de crear estándares de metadatos de materiales educativos. En ese mismo año un grupo catalogado P.1484 del IEEE se propuso el mismo objetivo. IMS trabajando en conjunto con ARIADNE (proyecto de investigación europeo) sometió su proyecto a la consideración del grupo de la IEEE y crearon en conjunto una propuesta que fue la base para el borrador del IEEE LOM, el

cual fue aprobado como estándar IEEE en el año 2002 con la referencia 1484.12.1. El estándar LOM define un conjunto de elementos de meta-datos para ser utilizado en la descripción de recursos de aprendizaje. Incluye los nombres de los elementos, los tipos de datos y la longitud de cada campo. IEEE-LOM define una estructura conceptual para los metadatos, requisitos de conformidad sobre cómo deben ser organizados, de igual forma define cómo deben comportarse las aplicaciones sobre los mismos para que ésta pueda ser adaptada por el comité IEEE-LOM (Burgos, Galve, García, & Sutil, 2002).

2.1.2 Categorías y elementos

El esquema de LOM está compuesto de 9 categorías, las cuales se muestran en la Tabla 5.

Tabla 5. Categorías y elementos del estándar LOM. (IEEE, 2002)

Categorías	Elementos
1. General	Identificador, Entrada de Catálogo, Lenguaje, Descripción, Clave, Cobertura, Estructura, Nivel de agregación.
2. Ciclo de vida	Versión, Estatus, Contribución, Rol, Entidad, Fecha.
3. Meta-Metadata	Identificador, Entrada de Catálogo, Contribución, Rol, Entidad, Fecha, Esquema de metadata, Lenguaje.
4. Técnico	Formato, Tamaño, Ubicación, Requisitos, Comentarios sobre la instalación, sobre la instalación, otros requerimientos de plataforma, Duración.
5. Educacional	Tipo de interacción, Tipo de recurso educativo, Nivel de interacción, Densidad semántica, Papel del usuario, Contexto, Segmento de edad típico, Dificultad, Tiempo típico de aprendizaje, Descripción, Lenguaje.
6. Derechos	Coste, Derecho de Autor y otras restricciones, Descripción.
7. Relación	Tipo, Identificador, Descripción, Entrada de Catalogo.
8. Anotación	Entidad, Fecha, Descripción.
9. Clasificación	Propósito, Camino taxonómico, Descripción, Clave.

A continuación se explican cada una de las categorías y sus elementos:

Categoría General: Esta categoría agrupa la información general que describe el OACA como un todo.

a) **Identificador:** Una etiqueta única que identifica unívocamente al OACA.
b) **Entrada en catálogo:** Entrada en un determinado catálogo. El valor para este metadato debe ser un par formado por un nombre de catálogo, así como por el nombre de la entrada en dicho catálogo. Este metadato puede especificarse para seleccionar el recurso cuando éste se encuentra indexado en un catálogo externo**.**
c) **Lenguaje:** El idioma o los idiomas utilizados en el material para comunicarse con los posibles consumidores del mismo.
d) **Descripción:** Texto con la descripción del OACA.
e) **Clave:** Una palabra o frase que describe el tópico del OACA.
f) **Cobertura:** El tiempo, la cultura, geografía o región en donde el OACA aplica.
g) **Estructura:** La estructura interna del material. LOM define el siguiente vocabulario controlado para definir la estructura: colección (collection), mixta (mixed), lineal (linear), jerárquica (hierarchical), en red (networked), ramificada (branched), compartimentada (parceled), atómica (atomic). Sin embargo, los autores pueden utilizar sus propios vocabularios, adaptados a sus necesidades pedagógicas particulares.
h) **Nivel de agregación**: Define la granularidad del material.

Categoría Ciclo de vida: Esta categoría agrupa metadatos referidos a la historia, estado actual y mantenimiento del material educativo por parte de los educadores.

a) **Versión:** La edición o versión del material.
b) **Estatus:** El estado de producción del material. LOM propone el siguiente vocabulario (aunque puede utilizarse cualquier otro): borrador (draft), final (final), revisado (revised), no disponible (unavailable).
c) **Contribución:** Aquellas entidades (personas, organizaciones, entre otros.) que han contribuido con el OACA durante su ciclo de vida, por ejemplo creación, edición, publicación, entre otros.
d) **Rol:** Tipo de contribución por parte de los contribuyentes.
e) **Entidad:** identificación e información sobre las entidades contribuyentes. Las entidades deben ser ordenadas desde la más relevante.

f) Fecha: Fecha de la contribución.

Categoría Meta-metadata: Agrupa información relativa a los metadatos en sí. En esta categoría aparecen elementos ya contemplados en las anteriores categorías, sin embargo, esta vez su significado es diferente, ya que se refieren a la producción de los metadatos en sí como recurso digital, y no a la producción del material educativo que se está describiendo.

a) Identificador: Etiqueta que identifica al conjunto de metadatos para el recurso. Este identificador se puede utilizar para seleccionar el conjunto de metadatos cuando éste se encuentra almacenado externamente.
b) Entrada en catálogo: Un catálogo y una entrada en dicho catálogo en el que el conjunto de metadatos para el recurso reside. Esto permite seleccionar los metadatos de un catálogo externo.
c) Contribución: Aquellas entidades que han afectado el estado de los metadatos durante su ciclo de vida.
d) Rol: Tipo de contribución por parte de los contribuyentes.
e) Entidad: identificación e información sobre las entidades contribuyentes. Las entidades deben ser ordenadas desde la más relevante.
f) Fecha: Fecha de la contribución.
g) Esquema de los metadatos: El nombre y versión de la especificación usada para crear los metadatos (por ejemplo, LOM v1.0).
h) Lenguaje: El idioma utilizado para proporcionar los metadatos.

Categoría Técnico: Esta categoría describe los requerimientos técnicos y características del OACA.

a) Formato: Formato del material. El material puede incluir múltiples formatos (por ejemplo HTML, JPG, MP4, entre otros). Una manera adecuada de describir los formatos es mediante su denominación MIME.
b) Tamaño: EL tamaño del OACA en bytes.
c) Ubicación: forma de ubicar el material, puede ser un URL o una descripción acerca de cómo llevar a cabo la localización.
d) Requisito: Plataforma informática necesaria para utilizar el material. Aquí se define el tipo de plataforma, nombre de la plataforma, versión mínima requerida y versión máxima requerida.
e) Comentarios sobre la instalación: Descripción de cómo instalar el OACA.

f) **Otros requerimientos de plataforma:** Información sobre otros requerimientos de software y hardware.
g) **Duración:** Tiempo que dura un OACA continuo cuando se reproduce a su velocidad normal (esto aplica a materiales educativos donde tenga sentido una duración de reproducción, como por ejemplo un video o una reproducción en flash).

Categoría Educacional: Esta categoría define las características educativas y pedagógicas fundamentales del OACA.

a) **Tipo de interacción:** Modo predominante de aprendizaje apoyado por el OACA. LOM propone el siguiente vocabulario para caracterizar este tipo de interacción: contenidos interactivos (active), contenidos pasivos (expositive), contenidos que comparten ambas características (mixed), contenidos donde no se especifica el tipo de interacción (undefined).
b) **Tipo de recurso educativo**: Tipo específico de OACA. Un mismo material puede tener varios tipos asociados, en cuyo caso, el tipo dominante debe estar definido de primero. LOM propone el siguiente vocabulario para caracterizar el tipo de material: ejercicio (exercise), simulación (simulation), cuestionario (questionnarie), diagrama (diagram), Figura (figure), Gráfico (graph), inicio (index), diapositiva (slide), Tabla (table), texto narrativo (narrative text), examen (exam), experimento (experiment), enunciado de problema (problem statement), autoevaluación (selfassessment).
c) **Nivel de interacción:** Especifica el grado de interacción del material. LOM propone el siguiente vocabulario: muy bajo (very low), bajo (low), medio (medium), alto (hight), muy alto (very hight).
d) **Densidad semántica:** Una medida subjetiva de la utilidad educativa del material en comparación con su tamaño y/o duración. Para expresar la densidad semántica LOM propone el mismo vocabulario propuesto en el nivel de interacción.
e) **Papel del usuario:** Indica el rol que ejerce el usuario final del material. LOM propone el siguiente vocabulario: maestro (teacher), autor (author), aprendiz (learner), gestor (manager).
f) **Contexto:** El entorno principal dentro del cual el aprendizaje y el uso del OACA deberían tener lugar. El vocabulario propuesto por LOM para especificar el contexto es el siguiente: educación primaria (primary education), educación secundaria (secundary education), educación superior (higher education),

primer ciclo universitario (university first cycle), segundo ciclo universitario (university second cycle), postgrado (university postgrade), primer ciclo de escuela técnica (technical school first cycle), segundo ciclo de escuela técnica (technical school second cycle), formación profesional (professional formation), formación continua (continuous formation), formación vocacional (vocational training)

g) **Segmento de edad típico:** rango de edades típico de los usuarios previstos.
h) **Dificultad:** nivel de dificultad del material. LOM propone el siguiente vocabulario para caracterizar dicho nivel: muy fácil (very easy), fácil (easy), medio (medium), difícil (difficult), muy difícil (very dificult).
i) **Tiempo típico de aprendizaje:** Tiempo aproximado que le toma a los usuarios previstos, trabajar con el OACA para alcanzar el aprendizaje.
j) **Descripción:** Comentarios sobre cómo debe ser usado el OACA desde el punto de vista pedagógico.
k) **Lenguaje:** Idioma usado por el usuario final.

Categoría Derechos: Esta categoría describe los derechos de propiedad intelectual y condiciones de uso del OACA.

a) **Costo:** Especifica si el recurso debe ser pagado. LOM propone el vocabulario controlado siguiente: si (yes), no (no).
b) **Derecho de autor y otras restricciones:** Establece si el recurso está sujeto o no a derechos de copia y otras restricciones.
c) **Descripción:** Comentarios sobre las condiciones de uso del OACA.

Categoría Relación: Define la relación del OACA con otros OACA. Un mismo OACA puede tener relación con múltiples recursos (que se denominarán para esta categoría como OACA objetivo).

a) **Tipo:** Define la naturaleza de la relación entre el OACA y el OACA objetivo. LOM propone el siguiente vocabulario: el material es parte de otro más complejo (isPartOf), el material tiene a otro como parte integrante (hasPart), el material es una versión de otro (isVersionOf), el material tiene a otro como una versión (hasVersion), el material es la descripción de un formato de otro material (isFormatOf), el material tiene a otro como formato (hasFormat), el material refiere al otro (references), el material está referido por el otro (isReferencedBy), el material está basado en otro (isBasedOn), el material es

la base de otro (isBasedOn), el material es la base de otro (isBasisFor), el material requiere la presencia de otro (requires), el material es requerido por el otro (isRequiredBy).

b) **Identificador:** Identificador único del OACA objetivo.
c) **Descripción:** Descripción del OACA objetivo.
d) **Entrada de catálogo**: Una entrada en un catálogo para el OACA objetivo.

Categoría de Anotación: Esta categoría provee comentarios sobre el uso educacional del OACA, y la información de cuándo y quién creó los comentarios.

a) **Entidad:** Entidad que creó la anotación.
b) **Fecha:** Fecha en que la anotación fue creada.
c) **Descripción**: El contenido de la anotación.

Categoría de Clasificación: Describe el lugar que ocupa el OACA dentro de un sistema de clasificación particular. Para definir múltiples clasificaciones, puede haber numerosas instancias de esta categoría.

a) **Propósito:** El propósito de clasificar el OACA. LOM propone el siguiente vocabulario controlado de propósitos: disciplina (discipline), idea (idea), requisito previo (prerequisite), objetivo educativo (educational objective), restricciones de acceso (accesibility restrictions), nivel educativo (educational level), nivel de destreza (skill level), nivel de seguridad (security level)
b) **Camino Taxón:** una serie de rutas en distintas taxonomías.
c) **Descripción:** Una descripción textual del material relativa al propósito de clasificación establecido.
d) **Clave:** Un conjunto de palabras clave relativas al propósito de clasificación establecido.

2.1.3 Representación de Metadatos LOM

Según (Blanco, 2004) IEEE LOM define y especifica un esquema de metadatos que permite múltiples implementaciones, e incluye atributos, definiciones y una estructura jerárquica que los relaciona entre ellos, y por lo tanto los aspectos teóricos del sistema, pero no incluye información acerca de cómo presentar estos metadatos o con qué mecanismos se puede transmitir y procesar esta meta

información o metadata. Es importante entonces, la utilización de un mecanismo de representación para los metadatos.

Los metadatos LOM pueden representarse en múltiples formatos (por ejemplo XML o RDF), sin embargo, XML ofrece ciertas ventajas para la codificación de los metadatos, como lo es la portabilidad de la información, ya que es independiente de la plataforma, o la extensibilidad del lenguaje, lo cual permite que se puedan añadir nuevas etiquetas cuando sea necesario sin complicación alguna. Además de esto, la sencillez de la estructura XML facilita la compresión de los metadatos contenidos en dicha estructura. Para la representación LOM en XML, se emplean las categorías descritas anteriormente, y se introducen elementos para cada categoría. Las presencia de todos los elementos es opcional, por lo tanto, si no existen elementos para una categoría dada, no se especifica el elemento para dicha categoría. En la Figura 2 se muestra la estructura gramatical de alto nivel para la codificación XML de metadatos LOM, y en la Figura 2 se muestra el aspecto general de un documento XML usado para la representación de los metadatos.

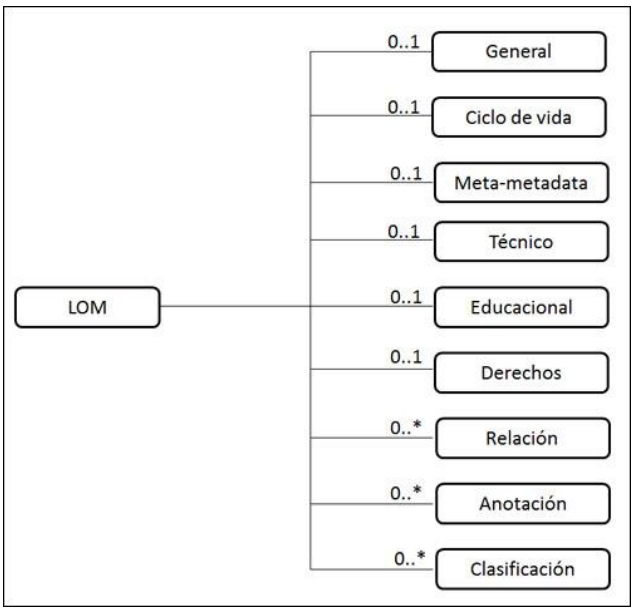


Figura 2. Estructura gramatical de LOM

```
<lom: lom>
    <lom: general>
        <lom: identificador> ... </lom: identificador>
        <lom: titulo> ... </lom: titulo>
        <lom: idioma> ... </lom: idioma>
        <lom: descripcion> ... </lom: descripcion>
        <lom: clave> ... </lom: clave>
        <lom: clave> ... </lom: clave>
    </lom: general>
    <lom: ciclodevida> ... </lom: ciclodevida>
    <lom: metametadata> ... </lom: metametadata>
    <lom: tecnico> ... </lom: tecnico>
    <lom: educaional> ... </lom: educaional>
    <lom: derechos> ... </lom: derechos>
    <lom: relacion> ... </lom: relacion>
    <lom: relacion> ... </lom: relacion>
    <lom: relacion> ... </lom: relacion>
    <lom: anotacion> ... </lom: anotacion>
    <lom: anotacion> ... </lom: anotacion>
    <lom: clasificacion> ... </lom: clasificacion>
    <lom: clasificacion> ... </lom: clasificacion>
    <lom: clasificacion> ... </lom: clasificacion>
</lom: lom>
```

Figura 3. Aspecto general documento XML

2.2 Sharable Content Object Reference Model

Sharable Content Object Reference Model (SCORM) es un estándar que permite la estructuración, empaquetamiento y distribución del contenido de los OACA en los ambientes virtuales. A continuación veremos detalladamente este estándar.

2.2.1 Organización

Las especificaciones de SCORM están agrupadas en tres documentos técnicos importantes que son los siguientes (Laguna, 2011):

a) **Modelo de Agregación de Contenido (Content Aggregation Model, CAM):** detalla los mecanismos necesarios para el empaquetamiento de OACA de modo que puedan ser ejecutados posteriormente por cualquier LMS.
b) **Entorno de Tiempo de Ejecución (Run-time Evironment, RTE):** indica la forma como se debe ejecutar el contenido y los mecanismos de comunicación entre el contenido y el LMS.
c) **Secuenciación y Navegación:** especifica la secuencia que se debe seguir al momento de ejecutar los diferentes componentes de un OACA. El manual recoge los eventos que pueden ser generados por los alumnos o por el sistema

y que deber ser procesados por el LMS para decidir cuál es el recurso educativo que debe ser ejecutado a continuación.

Los aspectos fundamentales descritos en los 3 libros técnicos de SCORM se pueden resumir de la siguiente manera:

a) Estructura y empaquetado de los OACA
b) Mecanismos para el intercambio de información entre el LMS y el OACA
c) Identificación de los OACA con metadatos
d) Secuencia de los contenidos de acuerdo a las interacciones de los usuarios.

2.2.2 Arquitectura

Según (Laguna, 2011) SCORM se basa en la filosofía cliente-servidor, donde el cliente es una combinación del estudiante y el recurso educativo, y el servidor es el entorno de ejecución que normalmente es soportado por la plataforma de aprendizaje o LMS. El LMS contiene la lógica necesaria para gestionar los OACA y para entregar un recurso en el momento correcto de acuerdo a las interacciones que haya tenido el estudiante con el sistema. En la Figura 4 se puede observar la arquitectura del modelo SCORM.

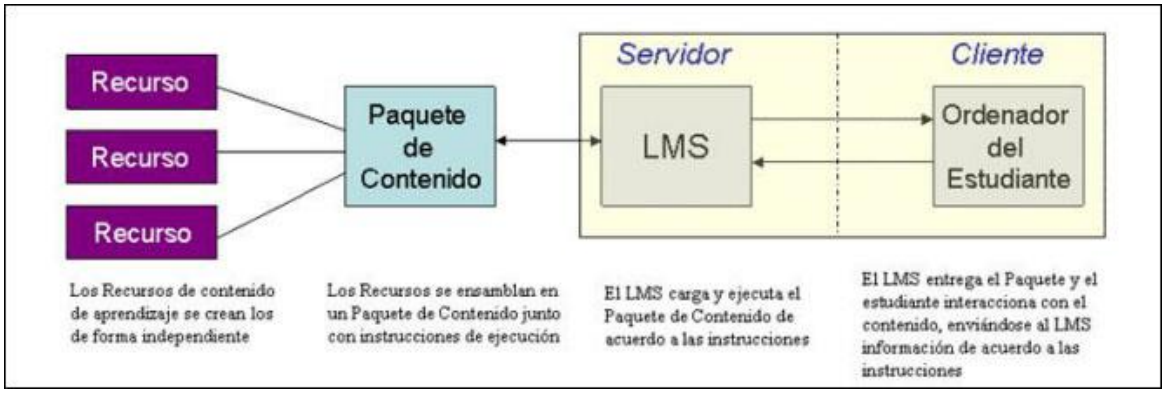


Figura 4. Arquitectura del Modelo SCORM. (Laguna, 2011)

2.2.3 Estándares y especificaciones

Según (Laguna, 2011), SCORM no consiste en una especificación que compita con las ya existentes en el campo. Al contrario, SCORM agrupa diversas especificaciones de diversos organismos y colabora con dichos organismos en la evolución de éstas especificaciones. En particular SCORM integra las siguientes

especificaciones de grupos como IMS Global Consortium, ARIADNE, AICC o IEEE-LTSC:

a) **IEEE Learning Object Meta-data 1484.12 (IEEE LOM):** Empleado en el Modelo de Agregación de Contenido para definir los metadatos de los OACA de contenido.
b) **IEEE ECMAScript API for Content to Runtime Services Communication 1484.11.2 (IEEE EACRSC):** Empleado por el Entorno de Tiempo de Ejecución para definir el mecanismo de comunicación entre el contenido y el LMS.
c) **IEEE Data Model for Content Object Communication 1484.11.1 (IEEE DMCOC):** Empleado por el Entorno de Tiempo de Ejecución para definir el modelo de datos empleado en la comunicación entre el contenido y el LMS.
d) **AICC/Web-Based CMI Guidelines (AICC WBCMIG):** Empleado para definir la estructura del contenido en el Modelo de Agregación de Contenido.
e) **IMS Content Packaging (IMS CP):** Empleado en el Modelo de Agregación de Contenido para agrupar OACA de contenido.
f) **IMS Simple Sequencing (IMS SS):** Empleado para el secuenciamiento de actividades en curso.

3. Repositorios de Objetos de Aprendizaje

Los ROA permiten publicar, descargar, compartir y reutilizar gran cantidad de recursos educativos en formato digital, lo cual beneficia a aquellos estudiantes y docentes que desean trabajar con dichos recursos y probar nuevas técnicas de enseñanza y aprendizaje.

3.1 Definiciones

Los ROA son la infraestructura clave para el desarrollo, almacenamiento, administración, localización y recuperación de todo tipo de contenido digital (ADL, 2002). Un Repositorio es un concepto tan amplio que va desde sencillos sistemas de almacenamiento hasta complejos entornos que incorporan, además de los sistemas de almacenamiento, conjuntos de herramientas que ayudan al proceso de reutilización. Dado el origen conceptual que tienen los OA a partir de filosofías de programación informáticas, puede pensarse que el término de Repositorio también se hereda de este campo, en el que se conciben como bases de datos para el almacenamiento de recursos simples de software reutilizables (assets), pero que han evolucionado hacia

complejos métodos de almacenamiento, búsqueda, navegación, evaluación de assets y recuperación (García, 2000).

Las definiciones anteriores también aplican al concepto de ROACA, con la diferencia de que los ROACA no almacenan cualquier tipo de OA sino aquellos que sean de contenidos abiertos, es decir, almacenan OACA. Por lo tanto, un ROACA es un almacén de recursos educativos de contenidos abiertos u OACA con metadatos asociados, donde los recursos se identifican y se clasifican de una manera específica, de forma que se facilite su posterior búsqueda y utilización. A diferencia de una base de datos, el Repositorio no solo contiene las referencias o diccionarios de datos, sino los propios contenidos u objetos digitales. Los ROACA además, deben ser capaces de interactuar con todos los sistemas, herramientas y usuarios que hagan uso de contenidos, así como con aquellos otros Repositorios o recursos que pueda agregar a su catálogo o con los que pueda intercomunicarse para hacer posibles cualquier tipo de búsquedas. En la Figura 5 se muestra la interacción de los ROACA.

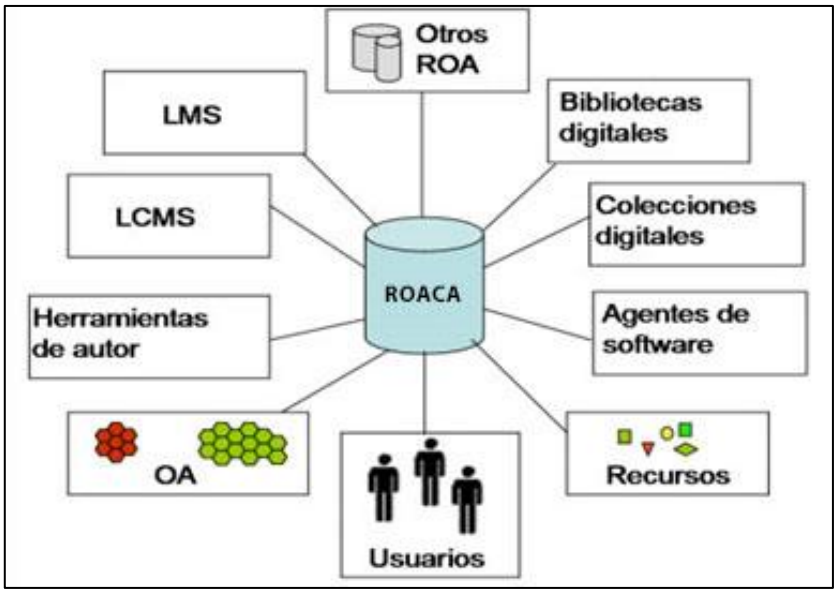


Figura 5. Interacciones de los ROACA. Adaptación de (López, 2005)

3.2. Características

A pesar de la gran diversidad de Repositorios digitales que podemos encontrar, hay un conjunto de aspectos en la arquitectura que todos los Repositorios tienen en común según (Tramullas, 2002):

a) Colección: desarrollo y gestión de colecciones de recursos digitales, locales o distribuidos, sin restricción de formato.
b) Servicios de valor añadido: productos y facilidades creados para dar valor al contenido de la colección, adecuados a las necesidades y requisitos de los usuarios.
c) Personalización: funcionalidad para que el usuario o la institución pueda definir su espacio de interacción con el Repositorio y seleccionar en listas propias los elementos de la colección.
d) Ciclo de vida de la información: los contenidos digitales pueden tener fases diferentes en sus diversas etapas y debe llevarse un seguimiento del ciclo de cada recurso.
e) El acceso a los contenidos es universal, ya que no hay restricciones de tiempo ni de espacio, debido a que el acceso se realiza a través de internet.
f) Los materiales educativos pueden ser descargados del Repositorio tantas veces como sea necesario, sin que afecte el archivo original ni que se deteriore la calidad de la copia.
g) Almacenan las diferentes revisiones de un archivo de forma eficiente, es decir, sólo almacenan los cambios realizados al archivo, no todo el material completo.
h) Los Repositorios asignan números a las distintas revisiones, lo cual permite recoger una determinada versión del mismo. Además, es posible dar nombres a un grupo de revisiones de varios archivos o incluso a todos los del proyecto entero, lo cual se suele usar para hacer referencia a las distintas versiones definitivas de un programa, archivo o proyecto.

Con base a la obra de Leslie, Landond, Lamb, & Poulin (2011) se pueden mencionar 10 características deseables en un ROACA:

a) Herramientas de búsqueda: considera la búsqueda a través de palabras claves u otros metadatos, la posibilidad de que el usuario pueda realizar exploraciones en listados predefinidos en alguna categorización o clasificación,

así como la capacidad del sistema de notificar a los usuarios sobre eventos determinados en el Repositorio.

b) **Herramientas de recopilación:** creación de “bookmarks” o “favoritos” de recursos o colecciones personales y posibilidad de creación de paquetes con varios recursos.
c) **Colectividad y evaluación:** posibilidad de que los usuarios puedan evaluar formal o informalmente un OACA, mecanismos para registrar los diferentes contextos en los que el OACA ha sido utilizado, y listas de OACA que el usuario desearía se incluyeran o modificaran.
d) **Meta-etiquetado:** herramienta de etiquetado, soporte de estándares y/o varios esquemas, importación y exportación de metadatos, mecanismo de identificación única de los recursos especialmente importantes.
e) **Administración de contenidos:** seguimiento del flujo de creación y publicación de un OACA, control de versiones y funciones de almacenamiento, herramientas de autoría.
f) **Administración y cumplimiento de derechos digitales de autor:** registro, transmisión interpretación y cumplimiento de los derechos de autor, así como un sistema de pago cuando sea necesario.
g) **Presentación y salidas de consorcio:** accesibilidad, salidas en múltiples formatos para diferentes dispositivos, cambios de apariencia de la interfaz, soporte de caracteres de diferentes idiomas, habilidad para servir como puerta de entrada para varias colecciones, transformación de formatos.
h) **Integración e interoperabilidad:** federación y búsqueda en otros Repositorios, integración con un administrador de recursos, soporte de Servicios Web y de aplicaciones API (Aplication Program Interface) que puedan extraer información de actividades dentro del Repositorio.
i) **Consideraciones técnicas:** autenticación, autorización y personalización, informe de uso, soporte para diferentes sistemas operativos, especificaciones de: la base de datos requerida por el Repositorio, escalabilidad, arquitectura del modelo de software, soporte, requisitos técnicos y humanos para su puesta en marcha, cliente del navegador.
j) **Costo/licenciamiento/Otros:** información de la compañía u organización que provee el software, número de instalaciones, modelo de costo o licenciamiento.

En términos generales, la característica más deseable en un ROACA es la calidad de los contenidos que alberga. Esta característica se concreta en los siguientes aspectos:

a) OACA con metadatos estándar orientados a facilitar ciertas funcionalidades tales como las búsquedas.
b) Inclusión de herramientas de evaluación a la calidad de los contenidos, preferiblemente anónima y externa.
c) Implementación de funciones de utilidad, preferiblemente de acuerdo a estándares o especificaciones ampliamente aceptadas tales como IMS DRI u otras que pudieran surgir.
d) Regulación de los derechos de propiedad intelectual de los contenidos.
e) Políticas específicas de actualización y mantenimiento tanto del Repositorio como de los materiales contenidos en él.

3.3. Funciones

Los ROACA deben proveer un conjunto de funciones con el fin de dar acceso a los OACA en un ambiente seguro. Estas funciones son las siguientes: (ADL, 2002).

a) Buscar/encontrar: Es la habilidad para localizar un OACA apropiado. Esto incluye la habilidad para su despliegue.
b) Pedir: Es la habilidad de solicitar un OACA que ha sido localizado.
c) Recuperar: Recibir un OACA que ha sido pedido.
d) Enviar: Entregar a un ROACA un OACA para ser almacenado.
e) Almacenar: Poner dentro de un registro de datos un OACA, con un identificador único que le permita ser localizado.
f) Colectar: Obtener metadatos de los OACA de otros Repositorios por búsquedas federadas.
g) Publicar: Proveer metadatos a otros Repositorios.

3.4 Clasificación

Para clasificar a los ROACA se debe tomar en cuenta la concentración de los recursos y la distribución de los metadatos. A continuación veremos la clasificación:

3.4.1 Según la concentración de los recursos

Describen la clasificación de los ROACA de acuerdo a la forma en que se encuentran los recursos (Rehak & Manson, 2003):

ROACA Locales: En este tipo de Repositorio, los OACA y sus descriptores se encuentran dentro de un mismo sistema e incluso dentro de un mismo servidor. En la Figura 6 se muestra la estructura de los ROACA locales.

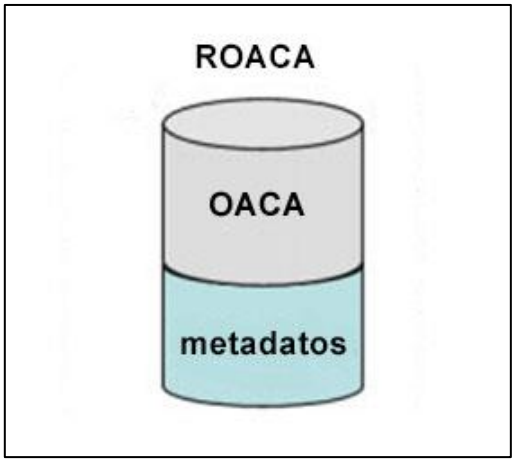


Figura 6. ROACA locales. Adaptación de (López, 2005)

ROACA Distribuidos: Este tipo de Repositorio contiene solo los descriptores y se accede al OACA a través de una referencia a su ubicación física que se encuentra en otro sistema o en otro ROACA. En la Figura 7 se muestra la estructura de los ROACA distribuidos.

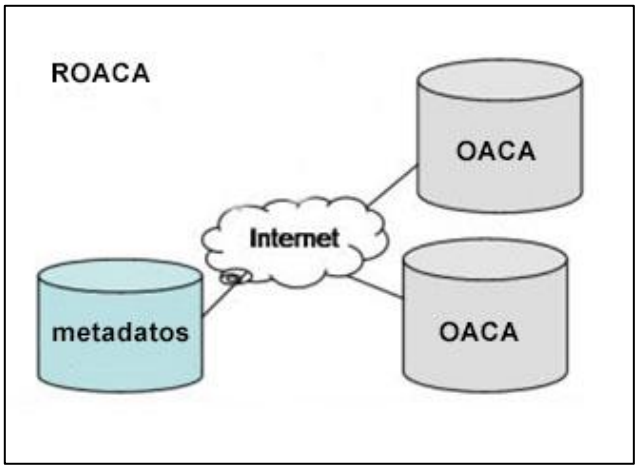


Figura 7. ROACA distribuidos. Adaptación de (López, 2005)

ROACA Mixtos: En estos ROACA se hace una combinación de los dos tipos mencionados anteriormente.

3.4.2. Según la distribución de los metadatos

De acuerdo a la distribución de los metadatos los ROACA pueden clasificarse en centralizados y distribuidos (Downes, 2004):

ROACA Centralizados: Los metadatos de los OACA están contenidos en un mismo servidor, aunque el OACA esté localizado en algún otro, en la Figura 8 se muestra la estructura de los ROACA centralizados.

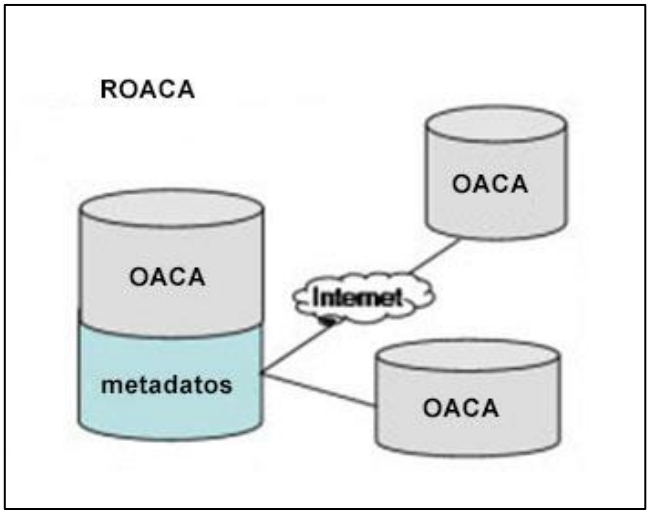


Figura 8. ROACA centralizados. Adaptación de (López, 2005)

ROACA Distribuidos: Operan a través de varios servidores, cada uno contiene diferentes grupos de metadatos y se comunican entre ellos para intercambiarlos, tal como se puede apreciar en la Figura 9.

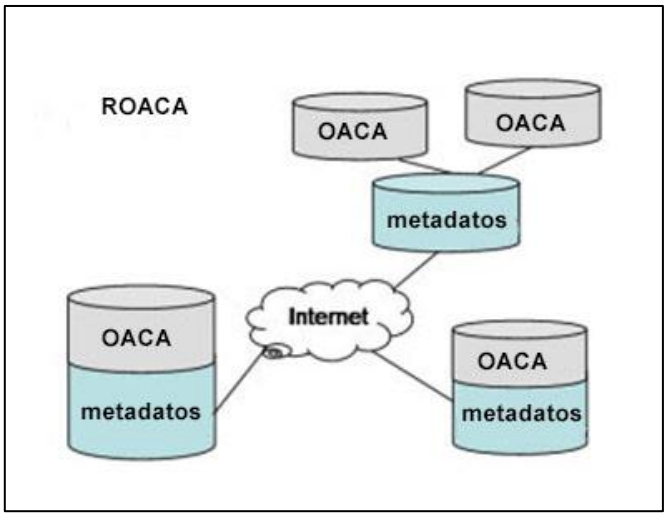


Figura 9. ROACA de metadatos distribuidos. Adaptación de (López, 2005)

Es importante destacar que también los ROACA se pueden clasificar en otros dos tipos: los ROACA que trabajan de forma independiente y los que operan sólo como módulos adicionales a otros productos (Learning Managment System). Los que trabajan de forma independiente (stand-alone) son los más comunes, y son aplicaciones con una interfaz Web, un mecanismo de búsqueda y listados con algún tipo de clasificación. Los que operan como módulos adicionales, utilizan los contenidos de forma exclusiva y sin que el usuario tenga acceso al Repositorio. Lo deseable es que los ROACA tengan ambas capacidades, tanto ofrecer una interfaz Web, para que los usuarios puedan acceder a la colección, así como la capacidad de comunicarse directamente con las plataformas de aprendizaje y hacer posible la interoperabilidad entre sistemas de diferente naturaleza.

3.5 Ejemplos de Repositorios de Objetos de Aprendizajes

A continuación se muestran algunos ROA desarrollados para América Latina que están operativos actualmente.

3.5.1 Multimedia Educational Resource for Learning and Online Teaching

Multimedia Educational Resource for Learning and Online Teaching (MERLOT) es un Repositorio diseñado sobre todo para los profesores y estudiantes de educación superior. Fue creado en 1997 por la California State University (Estados Unidos), y tiene más de 79 miembros alrededor del mundo y más de 21 mil materiales de aprendizaje incluyendo simulaciones, casos de estudio, presentaciones, entre otros,

en las más diversas áreas de enseñanza. La participación en MERLOT está basada en colaboración y soporte creativo de Miembros Individuales y Socios Institucionales.

MERLOT es un Repositorio centralizado que contiene solo los metadatos y los enlaces a los materiales ubicados en sitios remotos. Cualquier usuario puede tener acceso a todos los OA contenidos en el Repositorio, pero solo los miembros contribuyen agregando OA, comentarios y asignaciones, sin embargo, para ser miembro lo único necesario es inscribirse y no se adquiere ninguna responsabilidad. Para contribuir con algún OA, se debe contar con una dirección URL donde esté alojado el recurso, sin embargo, en caso de no poseer ninguna dirección URL, el Repositorio ofrece la posibilidad de adquirir una dirección gratuitamente. La revisión por pares es una actividad que se realiza en este Repositorio para evaluar la calidad de los OA agregados, y se toma en cuenta la calidad de los contenidos, la efectividad y la facilidad de uso. Por otro lado, Los materiales agregados no se borran del Repositorio sino que van clasificándose en un ranking de categoría.

Una característica interesante de MERLOT es que hace posible crear colecciones personales, donde se pueden organizar los materiales que se han ido encontrando y que son de interés para el miembro de la comunidad. También tiene colecciones generales de interés para todos los miembros. MERLOT es también una comunidad de personas en el área educativa. Los miembros de la comunidad ayudan a crecer a MERLOT contribuyendo con materiales y agregando asignaciones y comentarios. Muchos miembros de la comunidad dejan su información profesional disponible en el directorio de miembros de MERLOT. Este Repositorio está disponible en: http://www.merlot.org.

En Julio de 2009 el Centro de Innovación en Educación de la Universidad Tecnológica de Chile (INACAP) firma un convenio con MERLOT para crear la primera comunidad de habla hispana de MERLOT, llamada MERLOT CHILE. El objetivo de MERLOT CHILE es mejorar la efectividad de la enseñanza y el aprendizaje en los países hispano-hablantes, para esto busca incrementar la cantidad y la calidad de materiales de aprendizaje en línea revisados por pares expertos que puedan ser fácilmente incorporados a los cursos diseñados por docentes. En la Figura 9 se puede observar la página de inicio de MERLOT CHILE y en la Figura 10 se muestra la interfaz empleada por este Repositorio para que los usuarios realicen búsquedas de OA. MERLOT CHILE está disponible en: http://www.merlotchile.cl

Figura 10. Página de inicio de MERLOT CHILE

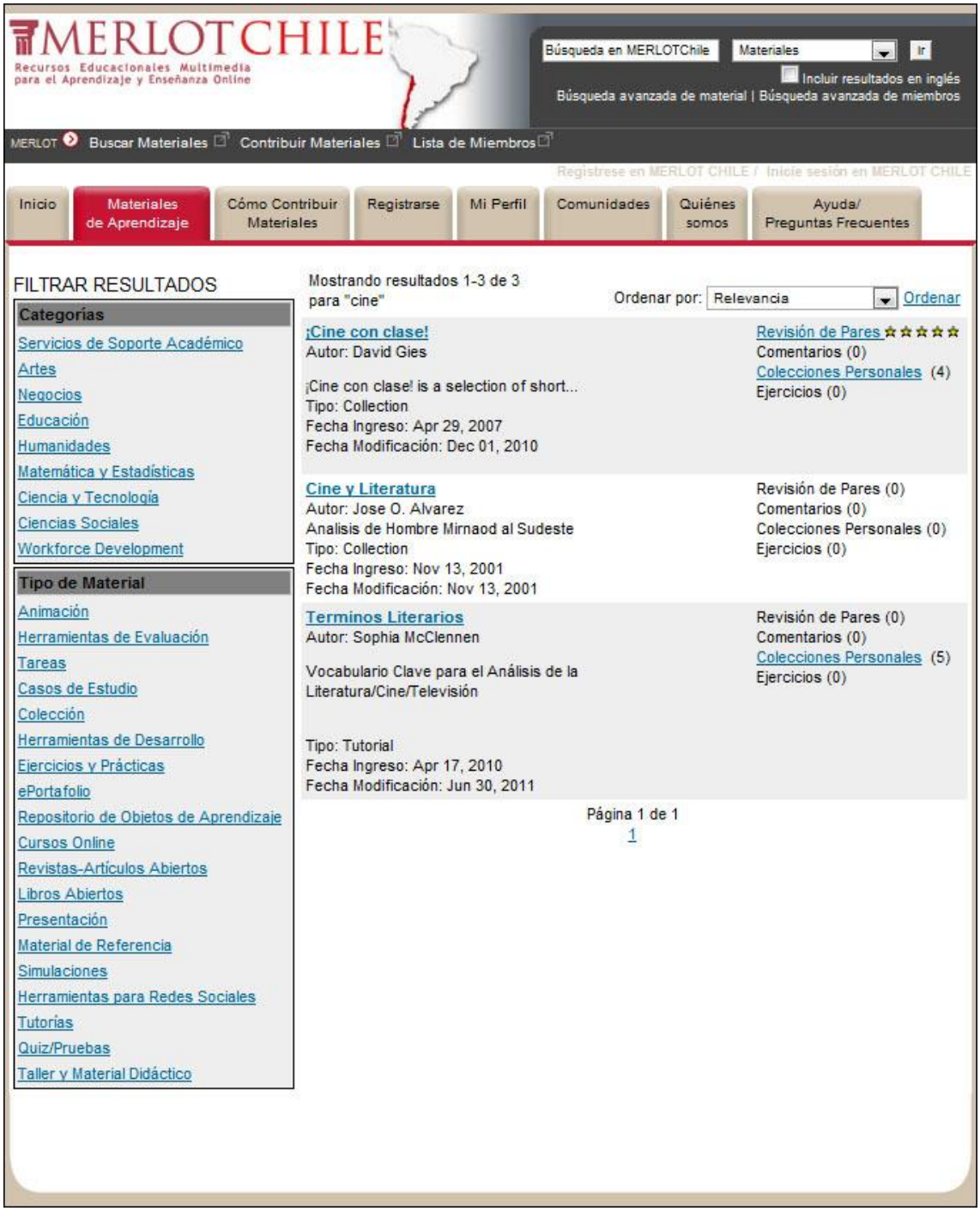


Figura 11. Interfaz para la realización de búsquedas de MERLOT CHILE

3.5.2 Iniciativa Colaborativa de Objetos de Aprendizaje Utilizables y ReUtilizables

Iniciativa Colaborativa de Objetos de Aprendizaje Utilizables y ReUtilizables (ICOA URU) es un ROACA desarrollado por el Ministerio del Poder Popular para la Educación Universitaria del Gobierno Bolivariano de Venezuela. Es un espacio virtual que pone a la disposición de las comunidades académicas una colección de OACA reutilizables, bajo la Figura de aprendizaje de acceso abierto. ICOA URU ofrece servicios en línea vinculados con la gestión de los OACA como: generación, evaluación, consulta y actualización. El acróstico con el que se designa este Repositorio es, al mismo tiempo, el nombre de una indígena legendaria: Icoa urú, con lo cual se rinde homenaje a nuestros pueblos originarios. Los OACA se almacenan en formato .zip siguiendo el estándar SCORM. En la Figura 12 se muestra la página principal de este Repositorio.

Existen varios roles de usuarios en el Repositorio, cada rol hereda los derechos del anterior, conformándose una jerarquía de roles en el siguiente orden:

a) Invitado: Puede acceder a los OACA publicados sin tener que estar registrado. Puede descargarlos, buscarlos, visualizarlos.
b) Usuario: Es un usuario registrado que tiene un área de trabajo para incluir, excluir y comentar OACA y puede comunicarse con otros usuarios del Repositorio a través de mensajería interna.
c) Autor: Este usuario puede subir al Repositorio nuevos OACA o crearlos dentro del mismo.
d) Revisor: Realiza una labor de auditoría sobre los OACA. Una vez concluido este proceso los OACA aprobados son publicados para que puedan ser accedidos por cualquier usuario que acceda al Repositorio.
e) Administrador: Administra y configura el Repositorio.

Este Repositorio también ofrece una sección de estadísticas, donde se puede observar de manera gráfica los usuarios más activos, los usuarios que más OACA han subido o creado, los usuarios con más cantidad de publicaciones y la cantidad de usuarios que interactúan por categoría. Las estadísticas de los usuarios se pueden filtrar por cantidad de usuarios, fecha, rol de usuario, país y estado. También se pueden generar gráficos que reflejen los OACA más descargados, la cantidad de OACA por calificación y la cantidad de OACA por categorías y subcategorías así como

también la cantidad de visitas al Repositorio por día en una semana. En la Figura 13 se muestra la sección de estadísticas de ICOUA URU. Este Repositorio está disponible en: http://roa.mppeu.gob.ve/

Figura 12. Página principal de ICOA URU

Figura 13. Sección de estadísticas de ICOA URU

3.5.3 Repositorio de Objetos para el Aprendizaje de la Universidad Simón Bolívar

El Repositorio de Objetos para el Aprendizaje de la Universidad Simón Bolívar (Esopo) es el ROA de la Universidad Simón Bolívar (Venezuela), esta plataforma hace visible y accesible la productividad del cuerpo profesoral de esta universidad, así como también promueve el desarrollo de contenidos digitales que pueden ser utilizados para la virtualización de los curos. A través de ESOPO, los profesores pueden publicar, clasificar, catalogar y etiquetar OA que utilizan en su gestión académica, tales como artículos, monografías, libros, tesis, recursos técnicos, entre otros. Los recursos pueden estar en formato de texto (HTML, PDF, PostScript, Texto plano, RTF, PowerPoint, Excel, Word), de imágenes (JPEG, PNG, GIF, BMP, TIFF), video (MPEG, QUICK TIME, AVI, WMV, MP4, FLASH, AVCHD) o de audio (WAV, MP3, OGG, FLAC, WMA). La publicación de los contenidos es bajo la licencia de Creative Commons.

Para publicar en ESOPO, es necesario ser miembro de la Universidad Simón Bolívar y hacer uso del “USBID”, el cual es un nombre de usuario único que posee cada miembro de la universidad. El sistema permite buscar tanto en la Web como en cualquier unidad de almacenamiento que posea el usuario, el recurso que se va a publicar en el Repositorio. Existen 4 grandes áreas para clasificar los OA dentro del sistema, las cuales son: Ciencias Físicas y Matemáticas, Ciencias Sociales y Humanidades, Ciencias Biológicas, y por último Ciencias y Tecnologías Administrativas e Industriales; y estas áreas poseen a su vez varias materias. En la Figura 14 se puede observar esta clasificación.

Figura 14. Clasificación de los OA en el ROA

En la Figura 15 se muestra la página principal de ESOPO. Este Repositorio se encuentra disponible en: http://esopo.usb.ve/

Figura 15. Página principal de ESOPO

En la Tabla 6 se muestra un cuadro comparativo de los ROA antes mencionados, se toma en cuenta la organización que los promueve, la región a la que pertenecen, el tipo de acceso, la organización de los metadatos, la organización de los OA, el estándar utilizado para los metadatos, el tipo de descarga de los OA y la capacidad de búsqueda en otros Repositorios.

Tabla 6. Cuadro comparativo de los ROA analizados

ROA	Organización	Región	OA	Metadatos	Estándar	Descarga	Búsquedas en otros ROA
MERLOT CHILE	California State University	Chile	Distribuidos	Centralizados	IEEE LOM	No ofrece descarga de paquete comprimido estandarizado	Si
ICOA URU	Ministerio del Poder Popular para la Educación Universitaria del Gobierno Bolivariano de Venezuela	Venezuela	Locales	Centralizados	IEEE LOM	Paquete comprimido estandarizado	No
ESOPO	Universidad Simón Bolívar	Venezuela	Locales y distribuidos	Centralizados	No especificado	No ofrece descarga de paquete comprimido estandarizado	No

En la Tabla 6 se puede observar que en cuanto a la organización de los metadatos, todos los Repositorios son iguales, es decir, todos son de metadatos centralizados. También es importante mencionar que, a excepción de ESOPO, el estándar LOM es utilizado por todos los Repositorios, lo cual refleja la importancia de dicha especificación para la interoperabilidad de los OA. En cuanto a la descarga en forma de paquete comprimido estandarizado, ICOA URU llama la atención por ser el único que ofrece esta opción. Finalmente, MERLOT es capaz de realizar búsquedas en otros Repositorios, pero de forma transparente al usuario.

4. Tecnologías de desarrollo Web

A continuación se describen algunas tecnologías que existen en la actualidad para el desarrollo de una aplicación Web, como por ejemplo la arquitectura orientada a servicios Representation State Transfer, el framework Symfony2 de PHP, el sistema manejador de bases de datos Postgresql, y el frameworkJQuery para la programación en Javascript.

4.1 Representation State Transfer

La arquitectura Representation State Transfer (REST) es una técnica de arquitectura de software para sistemas hipermedia distribuidos tales como la Web. REST se refiere estrictamente a una colección de principios para el diseño de arquitecturas en red. REST abarca una arquitectura cliente-servidor sin estado en el que los Servicios Web son vistos como recursos y pueden ser identificados por sus direcciones URL. REST tiene como principales principios u objetivos la utilización de los estándares Web, tales como HTTP y URI, y aprovechar la arquitectura de la Web para su beneficio. Según (Los Santos, 2009) algunas características de REST son:

a) Cliente-Servidor: estilo de interacción bajo demanda
b) Sin estado: cada petición desde un cliente hacia un servidor debe contener toda la información necesaria para comprender la petición, y no puede tomar provecho de ningún contexto almacenado en el servidor.
c) Caché: para mejorar la eficiencia de la red, las respuestas deber guardarse en la memoria caché del navegador.
d) Interfaz uniforme: todos los recursos son accesibles mediante una interfaz genérica. Por ejemplo HTTP GET, POST, PUT, DELETE.
e) Los recursos son nombres: el sistema se compone de recursos que son nombrados usando una URL.
f) Representaciones de recursos interconectados: las representaciones de los recursos están interconectadas usando URLs, por lo tanto un cliente está capacitado para progresar de un estado a otro.
g) Componentes en capas: intermediarios tales como servidores proxy, servidores caché, gateways, entre otros, pueden ser introducidos entre los clientes y los recursos, para ofrecer rendimiento, seguridad, entre otros.

4.1.2 Estándares usados por Representation State Transfer

REST está basado en los siguientes estándares:

a) HTTP
b) URL
c) Representación de los recursos: XML, HTML, GIF, JPEG, entre otros.
d) Tipos MIME: text/xml, text/html, entre otros.

4.1.3 Funcionamiento de Representation State Transfer

Para explicar el funcionamiento de la Arquitectura REST partiremos de un ejemplo de (Los Santos, 2009). Supongamos que una empresa de componentes informáticos ofrece un servicio para permitir que sus clientes puedan obtener una lista de productos. El recurso Web dispone de una URL para un recurso con la lista de productos. Por ejemplo, el cliente usaría la siguiente URL para obtenerla: http://www.empresa.com/productos

La forma como el Servicio Web genera la lista de productos es completamente transparente al cliente. Todo lo que el cliente sabe es que ha solicitado la URL anterior y que el documento que contiene la lista de productos ha sido devuelto. Puesto que la implementación es transparente, la empresa es libre de modificar la implementación que hay por debajo de este recurso, sin influenciar a los clientes. Esto es lo que se conoce como bajo acoplamiento. En la Figura 16 podemos ver un ejemplo del documento que el cliente recibe

```
<p:Parts xmlns:p="http://www.empresa.com"  xmlns:xlink="http://www.w3.org/1999/xlink">
    <Part id="00345" xlink:href="http://www.empresa.com/producto/00345"/>
    <Part id="00346" xlink:href="http://www.empresa.com/producto/00346"/>
    <Part id="00347" xlink:href="http://www.empresa.com/producto/00347"/>
    <Part id="00348" xlink:href="http://www.empresa.com/producto/00348"/>
</p:Parts>
```

Figura 16. Mensaje de respuesta de un Servicio Web basado en REST

Asumimos que el servicio ha determinado a través de procesos de negociación que el cliente quiere la respuesta como XML. Vemos que la lista de componentes tiene enlaces para obtener información detallada sobre cada uno. Esto es una característica clave de REST, el cliente se traslada de un estado al siguiente examinando y eligiendo las URLs de entre la alternativas que dispone el documento respuesta.

4.2 Symfony2

Symfony2 es un framework de PHP que facilita la utilización de la arquitectura Modelo-Vista-Controlador (MVC) y fue creado por una gran comunidad liderada por Fabien Potencier. Symfony2 fue desarrollado principalmente para aplicaciones Web, proporcionando herramientas para agilizar aplicaciones complejas y guiar al desarrollador a acostumbrarse al orden y buenas prácticas dentro del proyecto.

4.2.1 Características de Symfony2

A continuación se presentan las principales características de Symfony2 (Pacheco, 2011):

a) **Versátil:** Symfony2 puede funcionar como un framework full-stack (como por ejemplo CakePHP), como un framework de componentes (como por ejemplo Zend Framework) o como un micro framework (como por ejemplo FuelPHP).
b) **Poco intrusivo:** resuelve problemas de seguridad, persistencia de datos, manejo de formularios, validaciones, plantillas, caché, enrutamiento, rendimiento, entre otros, pero le da libertad al programador de desarrollar su proyecto de la forma que quiera.
c) **Buenas prácticas:** se basa en las mejores prácticas de otros frameworks tanto de PHP como de otros lenguajes de programación. Como por ejemplo Rails, Django, Spring, Java Hibernate, Pyhton, entre otros.
d) **Flexible:** Symfony2 es flexible en cuando al tipo de archivos de configuración (PHP, XML o YAML), en cuanto al motor de plantillas (Twig o PHP), en cuando al almacenamiento (SQL o NoSQL) y en cuanto al flujo de trabajo (anotaciones o archivos).
e) **Rendimiento:** para mejorar el rendimiento, Symfony2 exige que se trabaje con PHP 5.1.3 como mínimo, por otro lado, todos los archivos en Symfony2 (archivos de configuración, plantillas, anotaciones, entre otros) se convierten en PHP al ejecutarse, lo cual mejora la rapidez de la aplicación. También utiliza un proxy inverso en PHP para responder a las peticiones del cliente en el menor tiempo posible.
f) **Documentación:** cuenta con una amplia documentación gratuita en varios idiomas, entre ellos español.
g) **Comunidad:** hay más de 200 programadores de todo el mundo que son colaboradores del framework Symfony2. Este número de programadores es bastante alto comparado con otros frameworks.
h) **Popularidad:** es uno de los frameworks de PHP más usados en España y America Latina.
i) **Gestión de dependencias:** Symfony2 se integra ampliamente al proyecto un "Manejador de Dependencias" para PHP llamado Composer. Composer contiene una base de datos online de muchas librerías desarrolladas por terceros a fin de centralizarlas en un Repositorio llamado Packagist. Esto le permite a los programadores olvidarse de descargar las librerías que desean utilizar y almacenarlas dentro del proyecto base, ya que simplemente a través

de un archivo de configuración se declaran cuáles son las librerías que el proyecto necesita para funcionar.

j) **Varios entornos:** Symfony2 permite trabajar en dos entornos, el entorno de desarrollo y el entorno de producción. El entorno de desarrollo se utiliza durante la construcción de la aplicación, y en él se puede visualizar una traza detallada de los errores, la información de la sesión, los tokens, los tiempos de respuesta, entren otros. El entorno de producción se aplica cuando la aplicación está lista y comienza a ser utilizada por el cliente.

k) **Enrutamiento:** Symfony2 posee un archivo con todas las rutas de la aplicación. Cada ruta tiene asociado un Controlador y una función de dicho Controlador. En este archivo también se definen los métodos HTTP y los formatos permitidos para cada una de las rutas.

4.2.2. Arquitectura MVC con Symfony2

La arquitectura MCV implementada por Symfony2 corresponde a la siguiente: (Pacheco, 2011).

a) **Modelo**: Es el responsable de la conexión a la base de datos y la manipulación de los datos mismos. Esta capa está pensada para trabajar con los datos y obtenerlos, pero no mostrarlos, ya que la capa de presentación de datos es la Vista. Todas las clases del Modelo se guardan en la carpeta “Entity” de la aplicación. Las clases del Modelo se pueden generar automáticamente por consola a través del comando “generate:entity” de Doctrine. Una vez creado el Modelo, se puede crear la base de datos ejecutando el comando “schema:create”. Para cambiar de Sistema Manejador de Bases de Datos solo es necesario acceder a la página de configuración de Symfony2 y modificar algunos datos, sin necesidad de hacer cambios en el Modelo. Gracias a Doctrine, el programador nunca manipula directamente la Base de Datos.

b) **Vista:** Todo lo que se refiere a la visualización de la información, el diseño, colores, estilos y la estructura visual en sí de las páginas de la aplicación. Las vistas se manejan por defecto a través del motor de plantillas Twig, sin embargo Symfony2 le da la opción al programador de trabajar las vistas en formato PHP si así lo desea. A través de Twig se pueden ejecutar ciclos y condicionales, lo cual ofrece gran flexibilidad para trabajar con las variables que son pasadas a la vista desde el controlador. También es posible la herencia de plantillas con Twig, lo cual facilita el diseño de las diferentes

páginas que componen la aplicación y además se evita repetir código innecesariamente.

c) **Controlador:** Su responsabilidad es procesar y mostrar los datos obtenidos por el Modelo. Es decir, este último trabaja de intermediario entre los otros dos, encargándose también de la lógica de negocio. Los controladores en Symfony2 llevan el mismo nombre de la clase del Modelo a la cual pertenecen, pero con el sufijo Controller. Por ejemplo, el Controlador correspondiente a una clase llamada Persona, debe llamarse PersonaController. Dentro de los controladores se definen varias funciones, las cuales llevan como sufijo la palabra Action y su ejecución depende de la solicitud que haga el cliente y el método HTTP con el que se realice dicha solicitud (GET, POST, CREATE, DELETE, UPDATE). Las funciones básicas dentro de un Controlador de Symfony2 son indexAction, newAction, editAction y deleteAction. Los controladores pueden pasarle a la Vista cualquier tipo de objeto o variable para que se le muestre al usuario.

En la siguiente Figura 17 se muestra la interacción que tiene cada uno de los componentes del patrón MVC en Symfony2

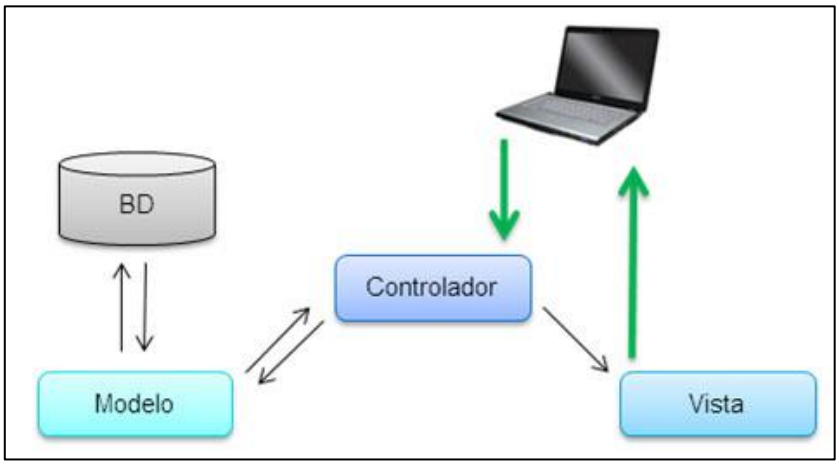


Figura 17. Patrón MVC en Symfony2

4.3 PostgresSQL

PostgreSQL es un sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS) que ha sido desarrollado desde la década de 1980. El proyecto PostgreSQL sigue actualmente un activo proceso de desarrollo a nivel mundial gracias a un equipo de desarrolladores y contribuidores de código abierto. Es ampliamente considerado como una de las mejores alternativas de sistema de bases de datos de código abierto (Ginestà & Perez, 2007).

4.3.1 Características de PostgreSQL

Según Ginestà & Perez (2007) las principales características de PostgreSQL son las siguientes:

a) **Atomicidad:** es la propiedad que asegura que una transacción se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.
b) **Consistencia:** es la propiedad que asegura que solo se ejecutarán aquellas acciones que no van a romper las reglas y directrices de integridad de la base de datos.
c) **Aislamiento:** es la propiedad que asegura que dos transacciones sobre la misma información nunca generarán ningún tipo de error.
d) **Durabilidad:** es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.
e) **Ejecución de triggers:** permite la ejecución de acciones antes o después de un evento de BD.
f) **Soporte de tipos y funciones de usuario:** soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario. También incorpora una estructura de datos Array.

4.3.2 Ventajas de PostgreSQL

Según Ginestà & Perez (2007) las principales ventajas de PostgreSQL son:

a) **Gratuito:** no hay costo asociado a la licencia del software.
b) **Estabilidad y confiabilidad:** es muy común que compañías reporten que PostgreSQL nunca ha presentado caídas en varios años de operación de alta actividad. Simplemente funciona.

c) **Extensible:** el código fuente está disponible para todos sin costo. Cualquiera puede extender o personalizar PostgreSQL sin costos adicionales. Esto se

complementa con la comunidad de profesionales y entusiastas alrededor del mundo que también extienden PostgreSQL todos los días.

d) **Multiplataforma:** está disponible en 24 plataformas en la última versión estable, y ahora en versión nativa para Windows.
e) **Diseñado para ambientes de alto volumen:** usa una estrategia de almacenamiento de filas llamada MVCC para conseguir una mejor respuesta en ambientes de grandes volúmenes. Los principales proveedores de sistemas de bases de datos comerciales usan también esta tecnología, por las mismas razones.
f) **Herramientas gráficas de diseño y administración de BD:** existen varias herramientas gráficas de alta calidad para administrar las bases de datos (pgAdmin, pgAcces, entre otras) y para hacer diseño de bases de datos (Tora, Data Architect).

4.4 JQuery

JQuery es un framework de JavaScritpt, entendiéndose por framework a un conjunto de librerías o a una librería que sirve como base para la programación avanzada de aplicaciones, aportando una serie de funciones que facilitan la ejecución de tareas habituales. De esta forma, JQuery ofrece una infraestructura con la que se tiene mucha mayor facilidad para la creación de aplicaciones complejas del lado del cliente, como trabajo y validación de formularios, efectos dinámicos, manejo de eventos, entre otros. (Tamayo Yero, Lemes Báez, & Naranjo Ortiz, 2011).

Capítulo III. Marco Aplicativo

En este capítulo se describe el proceso de construcción del ROACA, siguiendo la metodología XP y cumpliendo con una serie de iteraciones a lo largo de todo el desarrollo.

1. Análisis Global del Repositorio

Cumpliendo con los principios de XP, se realiza un análisis general de la aplicación, levantando los requerimientos necesarios para el desarrollo de la misma. A partir de ello, se obtienen las Historias de Usuario y se construye un esquema del funcionamiento de la aplicación.

1.1 Historias de Usuario

Como parte del análisis global, se crearon treinta y cinco (35) Historias de Usuario con el propósito de definir con exactitud las funcionalidades que debía cumplir el ROACA (Ver Anexo). A partir de esa lista de funcionalidades se creó un plan de trabajo dándole prioridad a las actividades más relevantes y que generarían un resultado inmediato y piezas de software completamente funcionales. A continuación se muestran algunas de las Historias de Usuario creadas:

Tabla 7. Iteración 1

Número: 1	**Prioridad:** MEDIA	**Estimación:** 5 días
Nombre: Diseñar el prototipo de interfaz del Módulo de Usuario		
Descripción: Diseño de las principales pantallas del Módulo de Usuario en código HTML.		

Tabla 8. Iteración 2

Número: 2	**Prioridad:** MEDIA	**Estimación:** 2 días
Nombre: Diseñar logo del Repositorio		
Descripción: Diseñar el logo del Repositorio usando la herramienta Photoshop.		

Tabla 9. Iteración 3

Número: 3	**Prioridad:** ALTA	**Estimación:** 1 día
Nombre: Realizar la configuración inicial de Symfony2		
Descripción: Chequear los requisitos para la correcta instalación de Symfony2 y realizar la configuración para el acceso a la Base de Datos		

Las Historias de Usuario previamente descritas son el resultado del levantamiento de requerimientos que se realizó durante el análisis global del Repositorio. Para continuar con este proceso se menciona a continuación los componentes y especificaciones técnicas.

1.2. Interfaz Gráfica

Es el sitio Web con el que interactúan los usuarios del Repositorio. Cuenta con un diseño sencillo, práctico, intuitivo y sin sobrecarga de información; además se emplean colores análogos, que son todos aquellos que contienen un matiz común con variaciones de valor o intensidad. En este caso se utilizaron principalmente los colores verde y azul con variaciones de tonalidad, pero manteniendo siempre los tonos suaves para hacerlos frescos y agradables a la vista. Adicionalmente, se incluyen botones de accesibilidad que permiten aumentar o disminuir el tamaño de las fuentes y cambiar el contraste de los colores.

1.3 Portal Web del Repositorio

Portal Web que se encarga de llevar a cabo todas las funcionalidades del ROCA. Dichas funcionalidades varían de acuerdo al tipo de usuario que se encuentre conectado. Los usuarios pueden ser Contribuyentes, No Contribuyentes o Administradores. Los usuarios Contribuyentes son aquellos que han insertado OACA en el ROACA, mientras que los No Contribuyentes son aquellos usuarios que solo se limitan a navegar a través del Portal y a descargar y visualizar los contenidos que ofrece el ROACA. El usuario Administrador es el que posee la mayor cantidad de privilegios dentro de la jerarquía, es conocido también como el Super Usuario, y es el encargado de gestionar los usuarios y los contenidos del ROACA a través del módulo de Administración.

1.4 Especificaciones Técnicas

El desarrollo del Repositorio se lleva a cabo utilizando las siguientes herramientas tecnológicas:

a) Programación del lado del servidor: PHP,con Symfony 2.1 sobre el servidor APACHE
b) Programación del lado del cliente: HTML5, JQuery
c) Sistema Manejador de Bases de Datos: Postgresql

2. Plan de Iteración

Debido a la metodología de desarrollo utilizada para el presente Trabajo Especial de Grado, fue necesaria la definición y ejecución de un conjunto de iteraciones, las cuales se les estableció fecha de inicio y fecha de culminación.

2.1 Iteración 0

En esta iteración se detalló el planteamiento del problema, el cual se basa en la falta de un espacio accesible, robusto y seguro que permita la gestión de los OACA. Además se presentó la propuesta y justificación de la solución, así como el diseño general del Repositorio, mejor conocido en XP como “Metáfora del Sistema”. Adicionalmente se estableció el nombre “ROACAR” al Repositorio, siendo el acrónimo de Repositorio de Objetos de Aprendizaje y Contenidos Abiertos Reutilizables.

2.1.1 Metáfora del Sistema

Para realizar la Metáfora del Sistema fue necesario tomar en cuenta la concentración de los recursos en el Repositorio. En este caso se decidió desarrollar un ROACA Local, de manera que los OACA y sus metadatos estuvieran dentro del mismo servidor, logrando de esta forma aumentar la rapidez en los tiempos de respuesta del Repositorio. En base a esto, se creó la Metáfora del Sistema, donde se incluyó además los tres tipos de usuario posibles. Los usuarios pueden ser de tipo Contribuyente, No Contribuyente o Administrador, y todos ellos se conectan al Repositorio a través de una interfaz gráfica común, la cual es accedida mediante una conexión a internet y un navegador Web. Por otro lado, se observa el servidor de aplicaciones sobre el cual se instaló el Repositorio y la base de datos. En la Figura 18 se puede observar la Metáfora del Sistema.

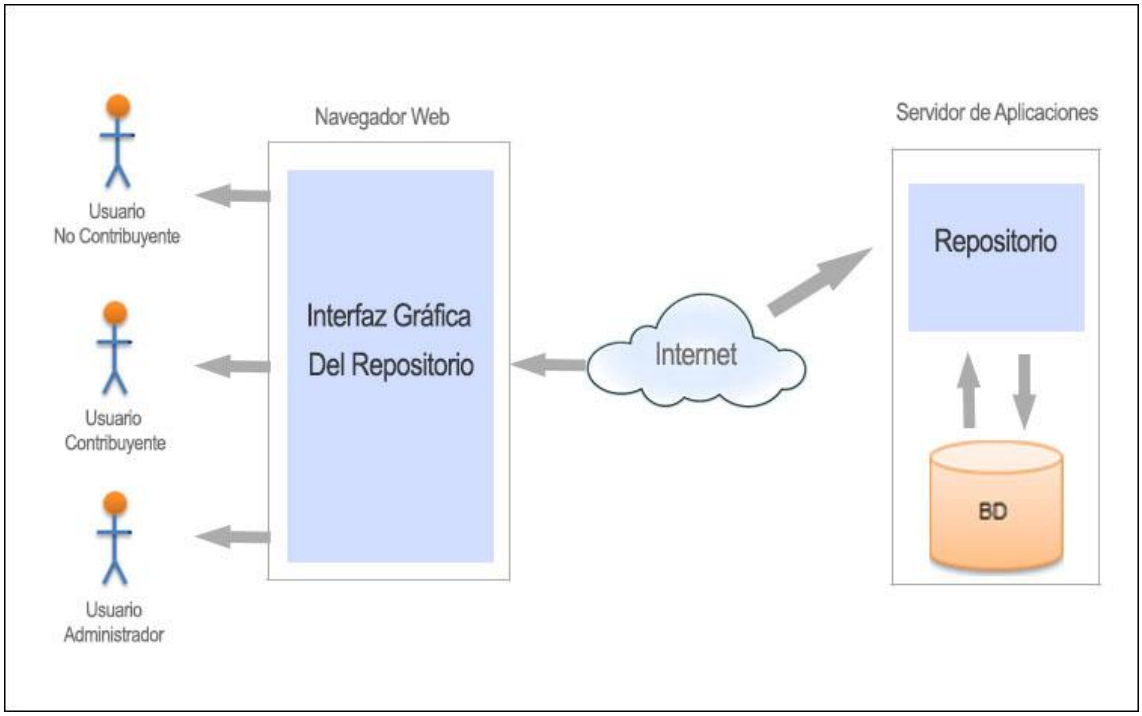


Figura 18. Metáfora del Sistema

2.2 Iteración 1

Esta iteración se basó en la creación de un prototipo de interfaz para el Módulo de Usuario, siendo éste el módulo principal del Repositorio, y el que será accedido por todos los tipos de usuario. Además, se desarrolló el inicio de sesión y el registro de usuario con sus respectivas validaciones.

Planificación

Iteración 1	
Descripción	Creación de prototipo de Interfaz del módulo de Usuario, e implementación del inicio de sesión y registro de usuarios.
Historias de Usuario	1. Diseñar prototipo de interfaz del módulo de Usuario 2. Diseñar logo del Repositorio. 3. Realizar la configuración inicial de Symfony2 4. Desarrollar la Clase Usuario 5. Desarrollar el controlador UsuarioController 6. Configurar el archivo de seguridad de Symfony2 y crear las funciones necesarias para el inicio de sesión. 7. Desarrollar las vistas de la Clase Usuario.

	8. Desarrollar las validaciones del formulario de Registro de Usuario.
Tiempo Estimado	21 días
Fecha Inicio - Fin	15/04/2013 – 6/05/2013

Bitácora de desarrollo

En la presente fase se incluye la bitácora de desarrollo, describiendo el tiempo estimado y real de las actividades generales que se realizaron durante este período. En este caso, se trata de la creación del prototipo de interfaz del módulo de Usuario y la implementación del Inicio de Sesión y Registro de Usuarios. La presente bitácora de desarrollo se puede observar en la Tabla 7.

Tabla 10. Bitácora de desarrollo - iteración 1

Nº	Tarea	Precede	Fecha Inicio	Fecha Fin	Días estimados	Días realizados
1	Diseño de interfaz y montaje de plantillas (CSS + HTML)	N/A	15/04/2013	18/04/2013	3	3
2	Análisis e implementación	N/A	18/04/2013	30/04/2013	12	12
3	Desarrollo de la interfaz	N/A	30/04/2013	03/05/2013	3	3
4	Realización de pruebas	N/A	03/05/2013	06/05/2013	3	3

Diseño

Para llevar a cabo el diseño de interfaz, se tomó como referencia otros Repositorios, así como también algunas aplicaciones Web educativas. En base a esto, se decidió utilizar un diseño sencillo e intuitivo, donde predominan los colores azul y verde con diferentes tonalidades. Este diseño se puede observar en la Figura 19

Figura 19. Primer prototipo de interfaz del Módulo Usuario

Como se puede observar en la Figura 19, la interfaz principal contiene un menú en la parte superior que permite acceder a todas las secciones del Módulo de Usuario. Dichas secciones se explican a continuación:

a) Inicio: Corresponde a la página inicial del ROACA, la cual contiene un mensaje de bienvenida junto con un texto explicativo de la importancia de los OACA. Dicho texto está acompañado de dos imágenes pertenecientes a la Universidad Central de Venezuela.

b) Gestionar OACA: A través de esta sección, los usuarios podrán listar, crear, editar y eliminar OACA. Esta sección se divide en dos subsecciones:

- ✓ **Insertar OACA:** A través de esta opción, los usuarios podrán insertar nuevos OACA en el Repositorio.
- ✓ **Mis OACA:** Esta sección contiene una lista de todos los OACA insertados por el usuario. Dicha lista contiene un resumen de los metadatos de cada OACA y un conjunto de botones que permiten realizar varias acciones sobre cada uno de los elementos de la lista. Las acciones que se pueden realizar son: Ver, Descargar, Ficha Descriptiva, Comentar, Editar y Eliminar.

c) **Buscar OACA:** A través de esta sección, los usuarios podrán realizar búsquedas avanzadas y también ver los OACA más descargados y los más recientes. Esta sección se divide en tres subsecciones:

- ✓ Búsqueda Avanzada
- ✓ Lo Más Reciente
- ✓ Lo Más Descargado

d) **Créditos:** Esta sección contiene los créditos del desarrollador y la información de contacto.

La interfaz también contiene un pequeño menú lateral donde se muestran las categorías de los OACA, permitiendo al usuario realizar una navegación rápida a través de dichas categorías.

Por otra parte, para desarrollar el inicio de Sesión, se creó en primer lugar la Clase Usuario. Para crear dicha Clase se tomó en cuenta las características que van a poseer los usuarios. En este sentido, los usuarios pueden ser de tipo Administrador, Contribuyente o No Contribuyente, por lo tanto, estos tres roles deben estar reflejados en la Clase. Por otra parte, el usuario debe tener otros atributos como por ejemplo: nombre, apellido, email y password. En la Figura 20 se muestra un Diagrama de Clases que refleja la estructura de los usuarios.

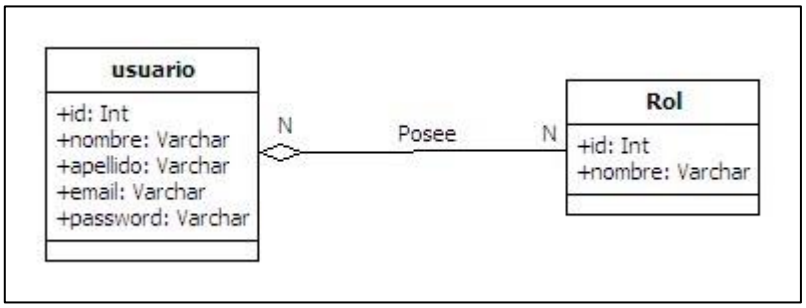


Figura 20. Diagrama de Clases con las Clases Usuario y Rol

Para definir los roles de usuario, se tomaron en cuenta las distintas necesidades de los usuarios que interactúan con el Repositorio. Estos roles también deben ir en contraste con los niveles de seguridad que debe ofrecer el Repositorio. En la Tabla 11 se muestran los roles que fueron definidos.

Tabla 11. Roles de usuario del Repositorio

Rol	Descripción
Usuario No Contribuyente	Este rol corresponde a los usuarios que no han insertado ningún OACA en el Repositorio pero pueden navegar por todas las secciones de la aplicación y pueden ver y descargar cualquiera de los OACA publicados.
Usuario Contribuyente	Este rol corresponde con los usuarios que han insertado OACA en el Repositorio. Los usuarios Contribuyentes pueden realizar las mismas acciones de los No Contribuyentes, pero además pueden editar y eliminar los OACA que sean de su autoría.
Administrador	Este rol corresponde al usuario administrador que posee todos los privilegios en el uso del Repositorio. Además se encarga de velar por la calidad de los contenidos que se publican.

Continuando con la fase de diseño de la presente iteración, se realizaron las vistas correspondientes al Inicio de Sesión y Registro de Usuario. En las Figuras 21 y 22 se pueden apreciar dichas vistas.

Figura 21. Vista correspondiente al Registro de Usuarios

Figura 22. Vista correspondiente al Inicio de Sesión

Codificación

Con el fin de cumplir los requerimientos planteados en la presente iteración, se creó en primer lugar, la hoja de estilo asociada a la interfaz del módulo de Usuario, incluyendo el mensaje de error que pueda generarse al autenticar al usuario. Esta hoja de estilo se puede apreciar en la Figura 23.

```css
body {

    font-family: Arial, Helvetica, sans-serif;
    background-repeat:no-repeat;
    background-attachment:fixed;

}
#logo {
    padding-top: 40px;
    padding-bottom: 5px;
    float: left;
}
.menu li a{

    text-decoration: none;
    color: white;
    display: block;
    text-align: center;

}
#div_sesion {

    float: right;
    position: relative;
    text-align: right;
}
.error {

    color: #FE2E2E;
    background-color: #FDE4E1;
    border: solid 1px #FBD3C6;
    padding: 4px;
    -webkit-border-radius: 6px;
    border-radius: 6px;
    margin-bottom: 10px;
}
```

Figura 23. Código css para la interfaz general del Módulo de Usuario

Posteriormente, se creó la Clase Usuario, tomando en cuenta los atributos y las características que van a poseer los usuarios. Adicionalmente, se agregaron ciertos parámetros en la Clase que permitirán la realización de las validaciones del lado del servidor. En la Figura 24 se puede ver dicha Clase.

```php
<?php
namespace ROA\ROABundle\Entity;
/**
 * @ORM\Entity
 * @ORM\Table(name="usuario")
 */
class Usuario implements UserInterface, \Serializable{
    /**
     * @ORM\Id
     * @ORM\Column(type="integer")
     * @ORM\GeneratedValue(strategy="IDENTITY")
     */
    protected $id;
    /**
     * @ORM\Column(type="string", length=60)
     * @Assert\NotBlank(message="Este campo no puede ser vacio")
     */
    protected $email;
    /**
     * @ORM\Column(type="string", length=25)
     * @Assert\NotBlank(message="Este campo no puede ser vacio")
     */
    protected $nombre;
    /**
     * @ORM\Column(type="string", length=25)
     * @Assert\NotBlank(message="Este campo no puede ser vacio")
     */
    protected $apellido;
    /**
     * @ORM\Column(type="string", length=8)
     * @Assert\NotBlank(message="Este campo no puede ser vacio")
     */
    protected $password;
    /**
     * @ORM\ManyToMany(targetEntity="Role")
     * @ORM\JoinTable(name="usuario_role",
     *      joinColumns={@ORM\JoinColumn(name="usuario_id", referencedColumnName="id")},
     *      inverseJoinColumns={@ORM\JoinColumn(name="role_id", referencedColumnName="id")}
     * )
     */
    protected $roles;
```

Figura 24. Codificación de la Clase Usuario

Seguidamente, se creó la Clase Rol, ya que como se puede observar en la Figura 24, la Clase Usuario posee una relación de muchos a muchos con Rol. En la Figura 25 se observa la Clase Rol.

```php
<?php

namespace ROA\ROABundle\Entity;

/**
 * @ORM\Entity
 * @ORM\Table(name="rol")
 */
class Rol implements RoleInterface
{
    /**
     * @ORM\Id
     * @ORM\Column(type="integer")
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;

    /**
     * @ORM\Column(name="nombre", type="string", length=255)
     */
    protected $nombre;

}
```

Figura 25. Codificación de la Clase Rol

Luego de crear la Clase Usuario y la Clase Rol, se creó el controlador UsuarioController, y el controlador DefaultController, donde se agregó la función encargada de manejar la sesión. En la Figura 26 se muestra dicha función.

```php
public function loginAction(){

    $request = $this->getRequest();
    $session = $request->getSession();

    if ($request->attributes->has(SecurityContext::AUTHENTICATION_ERROR)){
        $error = $request->attributes->get(SecurityContext::AUTHENTICATION_ERROR);
    } else {
        $error = $session->get(SecurityContext::AUTHENTICATION_ERROR);
    }

     return $this->render('ROABundle:Default:login.html.twig', array(
        'last_username' => $session->get(SecurityContext::LAST_USERNAME),
        'error'         => $error,
    ));
}
```

Figura 26. Función para el manejo de sesión

Seguidamente se configuró el archivo security.yml de Symfony2, el cual se encarga de manejar la autenticación y autorización de los usuarios. En este caso, se agregaron al archivo security.yml la jerarquía de roles del Repositorio y el tipo de codificación para la contraseña. Por otro lado, se establecieron los parámetros del

control de acceso, especificando que solo los usuarios con rol de Administrador pueden acceder al Módulo de Administración. En la Figura 27 se muestra dicho archivo.

```
security:
    encoders:
        Symfony\Component\Security\Core\User\User: md5
        ROA\ROABundle\Entity\Usuario: md5

    role_hierarchy:
        ROLE_SUPER_USER: [ROLE_CONTRIBUYENTE]
        ROLE_CONTRIBUYENTE: [ROLE_NO_CONTRIBUYENTE]

    providers:
        user_db:
            entity: {class: ROA\ROABundle\Entity\Usuario, property: email}

    firewalls:
        main:
            pattern: /.*
            provider: user_db
            form_login:
                default_target_path: /index
                login_path: /index
                check_path: /login_check
                use_referer: true
                failure_handler: authentication_handler
            logout:
                path: /logout
                target: /index
            security: true
            anonymous: true

    access_control:
        - { path: /admin, roles: ROLE_SUPER_USER }
```

Figura 27. Archivo security.yml de Symfony2

Luego se creó dentro del controlador usuarioController una función para restaurar la contraseña. Esta función se encarga de enviarle al usuario una contraseña provisional por correo electrónico. Para el envío de correos se hizo uso de las llamadas “Inyecciones de Dependencias” de Symfony2, que no son más que funciones que pueden ser accedidas desde cualquier controlador de la aplicación, pasándole por parámetro aquellas Clases de las cuales dependen dichas funciones. En este caso, se trata de la función “Mailer”, que es la encargada de enviar los correos. En la Figura 28 se puede observar el uso de esta función dentro del controlador usuarioController.

```
//Envio de email
$clave = $this->get('generador_clave')->getClave();//clave provisional
$mensaje = \Swift_Message::newInstance()
    ->setSubject('Restauracion de Contrasena')
    ->setFrom('juansneak@gmail.com')
    ->setTo($usuario->getEmail())
    ->setBody(
            $this->renderView(
            'ROABundle:Usuario:email.html.twig',
            array('clave' => $clave)), 'text/html');
    $this->get('mailer')->send($mensaje);
    $content = $this->renderView('ROABundle:Usuario:recuperar_contrasena.html.twig', array());
```

Figura 28. Envío de correos en el controlador UsuarioController

Finalmente, se creó una función llamada loginService la cual se encarga de ofrecer el inicio de sesión como un servicio REST. Esta función recibe como parámetro el email y la contraseña del usuario y puede ser accedida desde una aplicación externa. La función se encarga de verificar la existencia del usuario a partir de los datos pasados por parámetro. Si el usuario existe se retornan todos los datos del usuario en un formato estándar (JavaScript Object Notation) de forma que pueda ser leído e interpretado por cualquier aplicación independientemente del lenguaje de programación o plataforma sobre la cual esté construida.

Pruebas

Las pruebas en esta iteración se basaron en el acceso a usuarios registrados y al registro de nuevos usuarios, así como también la restauración de contraseña. La última prueba refiere también al acceso del Administrador, el cual debe tener permiso de ingresar al Módulo de Administración. Estas pruebas se pueden apreciar en la Tabla 12.

Tabla 12. Casos de prueba - Iteración 1

Nº Caso Prueba	Módulo	Caso de Prueba	Resultado Esperado	Resultado Obtenido
1	Usuario	Acceder al Repositorio con usuario y contraseña otorgados.	Otorgar acceso al usuario registrado.	Se le concedió acceso al Repositorio, indicando su nombre de usuario y permitiendo el cierre de sesión.

2	Usuario	Registrar un nuevo usuario	Creación de un nuevo usuario.	Se guardó la información básica del usuario. Incluyendo el nombre de usuario y contraseña
3	Usuario	Restaurar la contraseña de un Usuario	Enviar una clave provisional por correo al usuario	Se envió correctamente la contraseña provisional por correo y el usuario pudo restaurar su contraseña
4	Administración	Acceder al Repositorio con rol de Administrador	Otorgar acceso al Administrador y permitir el acceso al Módulo de Administración	Se le concedió acceso al Módulo de Administración

2.3 Iteración 2

La presente iteración se basó en la creación de la Clase OA y las funciones CRUD (Create, Read, Update, Delete), que en conjunto con lo creado en la iteración anterior (registro de usuarios), otorgó continuidad al desarrollo del Módulo de Usuario, a través del cual, los usuarios registrados podrán almacenar, ver, editar y eliminar sus OACA.

Planificación

Iteración 2	
Descripción	Creación de la Clase OA y las funciones CRUD (Create, Read, Update, Delete)
Historias de Usuario	9. Crear la Clase OA 10. Crear todos las Clases correspondientes a los metadatos 11. Crear el controlador OAController 12. Desarrollar las funciones CRUD dentro del OAController 13. Configurar las rutas de la Clase OA 14. Desarrollar las vistas correspondientes a la Clase OA
Tiempo Estimado	24 días
Fecha Inicio - Fin	07/05/2013 – 31/05/2013

Bitácora de desarrollo

La bitácora de desarrollo de la presente iteración describe el tiempo estimado y real de las actividades que se llevaron a cabo durante este período, en donde se comenzó el desarrollo de la Clase más importante del Repositorio (OA). Las funcionalidades asociadas a esta Clase, refieren específicamente a la inserción de OA junto con sus respectivos metadatos, y, a la edición y eliminación de los recursos almacenados, así como el diseño y desarrollo de las vistas para tales propósitos. La presente bitácora de desarrollo se puede apreciar en la Tabla 13.

Tabla 13. Bitácora de desarrollo - Iteración 2

Nº	Tarea	Precede	Fecha Inicio	Fecha Fin	Días estimados	Días realizados
1	Diseño de interfaz y montaje de plantillas (CSS + HTML)	N/A	7/05/2013	10/05/2013	3	3
2	Análisis e implementación	N/A	10/05/2013	24/05/2013	14	16
3	Desarrollo de la interfaz	N/A	24/05/2013	28/05/2013	4	4
4	Realización de pruebas	N/A	28/05/2013	31/05/2013	3	3

Diseño

En esta fase de diseño se llevó a cabo la primera refactorización, en donde se recodificó la hoja de estilos de la interfaz general del Módulo Usuario, y se creó un diseño mucho más llamativo. También se creó un nuevo logo y se añadió el logo de la UCV en la cabecera. Por otro lado, se agregaron los botones de accesibilidad y se cambió la tipografía y tamaño de letra en casi todos los contenidos. En la Figura 29 la interfaz general del Módulo Usuario.

Figura 29. Interfaz general del Módulo de Usuario (rediseño)

Basado en la nueva interfaz de usuario, se procedió a diseñar el formulario que permite insertar nuevos OACA. En dicho formulario se especifican los campos que son obligatorios y recomendados, y además se incluye un botón de ayuda para cada campo. En la Figuras 30 se muestra el diseño de los formularios.

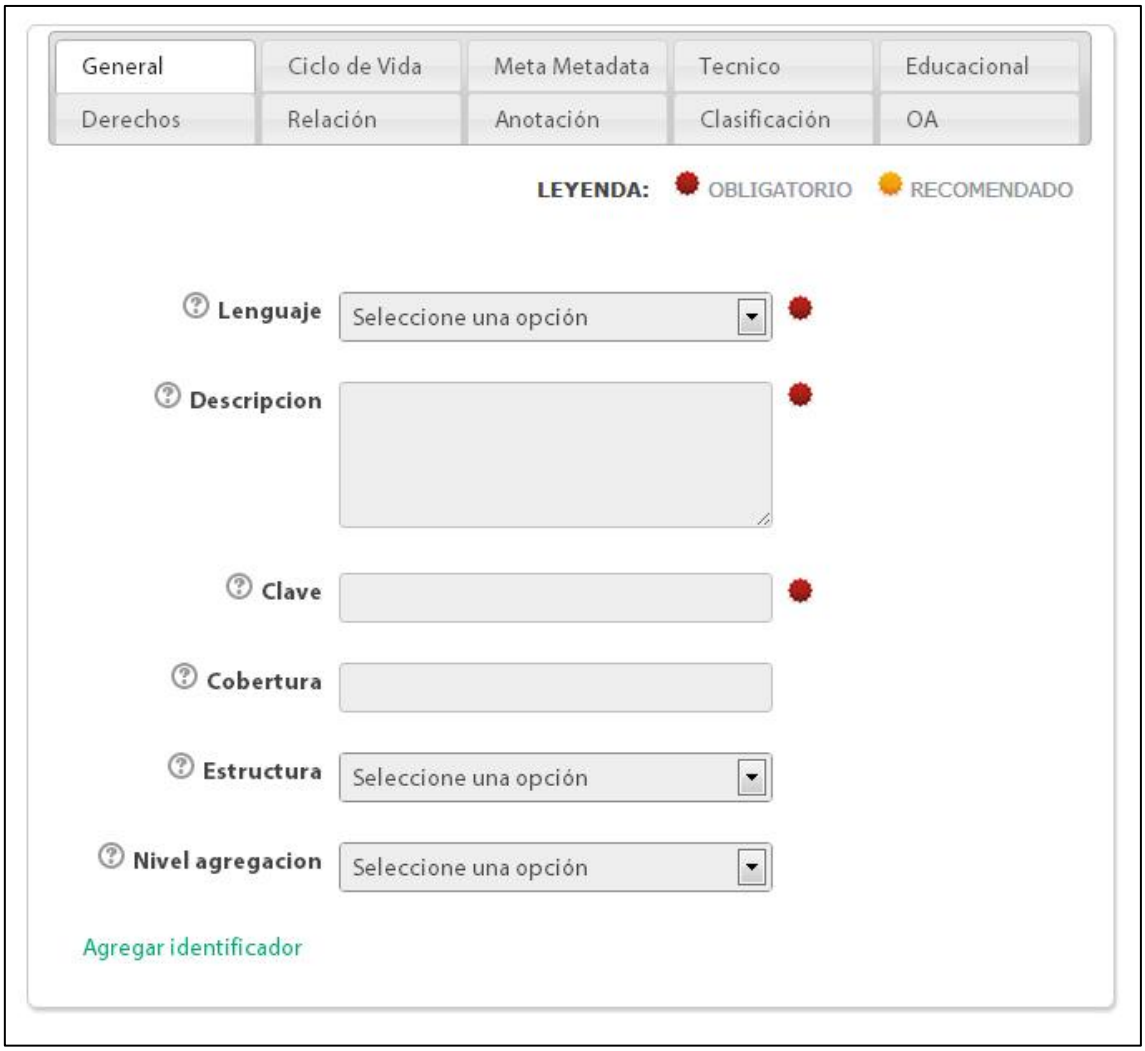


Figura 30. Detalle del formulario para la inserción de OACA

El diseño de la presente iteración lo conformó también la creación de un diagrama para representar la Clase OA. Cabe destacar que para la representación de los metadatos se empleó el estándar LOM, por lo que se puede observar que la Clase OA viene siendo la raíz del diagrama y se relaciona con otras 9 Clases, las cuales representan las 9 categorías de LOM (General, Ciclo de Vida, Meta Metadata, Técnico, Educacional, Derechos, Relación, Anotación y Clasificación). Algunos atributos de las categorías pueden ser representados a su vez por otras Clases, las cuales son: Identificador, Contribución, Entidad, Requerimiento y Recurso. En la Figura 31 se puede observar la representación de la Clase OA junto con sus metadatos a través de un diagrama de clases.

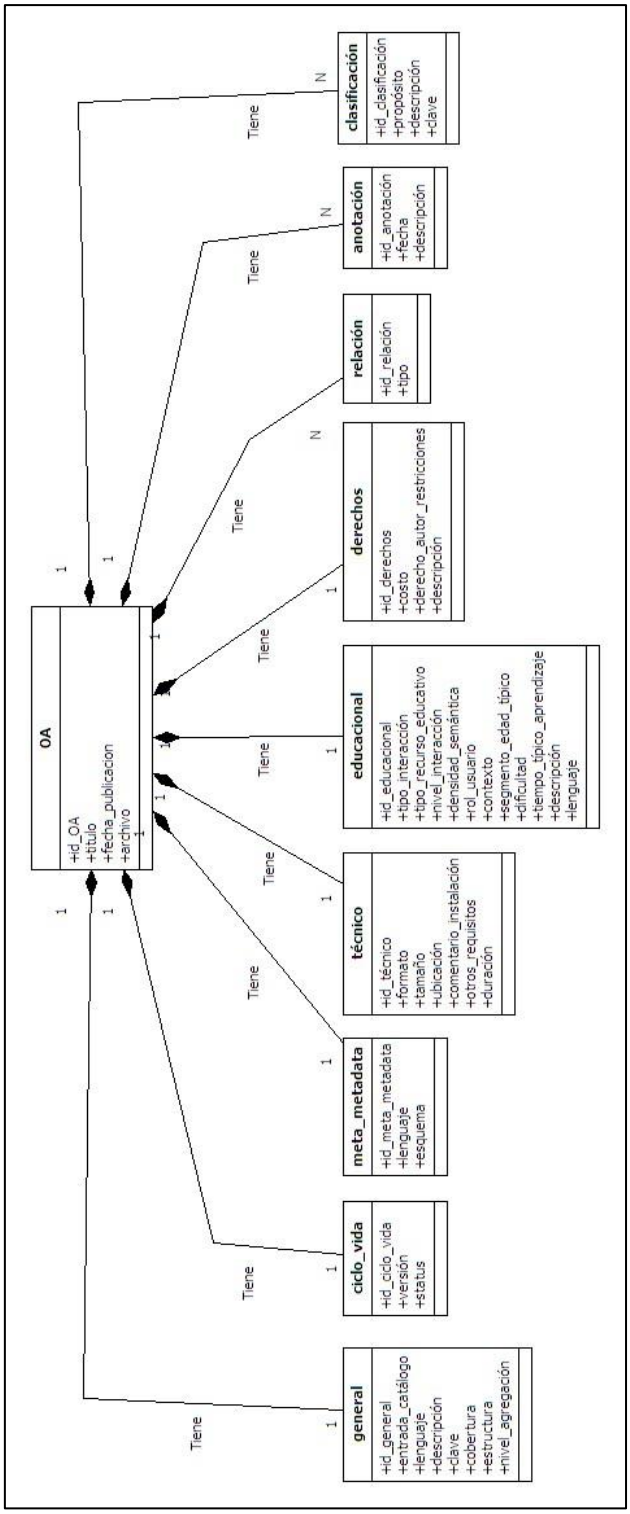


Figura 31. Diagrama de Clases de los metadatos

Otro aspecto importante de la Clase OA, es lo referente al tipo de OACA que se pueden almacenar dentro del Repositorio. En este sentido, se estableció que las categorías permitidas son: fundamentales y combinados abiertos (clasificación según su composición). En el caso de los fundamentales, está permitido cualquier formato abierto de video, audio, imagen o texto. Para los combinados cerrados se estableció como norma general que los OACA deben estar comprimidos en formato .zip y deben contener un archivo llamado “index” en la raíz de la carpeta. En este archivo “index” debe estar presente el contenido principal y todos los enlaces a las diferentes secciones del OACA. Toda esta información debe estar disponible al usuario en el botón de ayuda.

Codificación

En esta fase se comenzó por crear la Clase OA. Para crear la Clase OA, fue necesario crear las Clases que se relacionan con OA, y las Clases que forman parte a su vez de las Clases relacionadas con OA. De esta forma, para crear la Clase OA se crearon en primer lugar las siguientes Clases: General, CicloVida, MetaMetadata, Técnico, Educacional, Derechos, Relación, Anotación, Clasificación, Identificador, Contribución, Entidad, Requerimiento y Recurso. Es importante destacar que para el nombre de las Clases se siguió el estándar básico de codificación descrito en la implementación de la metodología XP.

En todas las relaciones se usó el comando de Doctrine “cascade={‘persist’, ‘remove’}” lo que permite realizar operaciones de inserción y eliminación en cascada. De esta forma, al crear un objeto de tipo OA y hacerlo persistente en la Base de datos, todas sus relaciones persisten automáticamente. Por otro lado, al eliminar un objeto de tipo OA, se eliminan también todas sus relaciones. En la Figura 32 se puede observar la codificación final de la Clase OA.

```php
<?php

namespace ROA\ROABundle\Entity;
class OA
{
    private $id;

    private $titulo;

    private $fecha_publicacion;

    public $file;

    /**
     * @ORM\OneToOne(targetEntity="General", cascade={"persist","remove"})
     * @ORM\JoinColumn(name="general_id", referencedColumnName="id")
     */
    private $general;

    /**
     * @ORM\OneToOne(targetEntity="CicloVida", cascade={"persist","remove"})
     * @ORM\JoinColumn(name="ciclovida_id", referencedColumnName="id")
     */
    private $ciclovida;

    /**
     * @ORM\OneToOne(targetEntity="MetaMetadata", cascade={"persist","remove"})
     * @ORM\JoinColumn(name="metametadata_id", referencedColumnName="id")
     */
    private $metametadata;

    /**
     * @ORM\OneToOne(targetEntity="Tecnico", cascade={"persist","remove"})
     * @ORM\JoinColumn(name="tecnico_id", referencedColumnName="id")
     */
    private $tecnico;

    /**
     * @ORM\OneToOne(targetEntity="Educacional", cascade={"persist","remove"})
     * @ORM\JoinColumn(name="educacional_id", referencedColumnName="id")
     */
    private $educacional;

    /**
     * @ORM\OneToOne(targetEntity="Derechos", cascade={"persist","remove"})
     * @ORM\JoinColumn(name="derechos_id", referencedColumnName="id")
     */
    private $derechos;

    /**
     * @ORM\OneToMany(targetEntity="Anotacion", mappedBy="oa", cascade={"persist","remove"})
     */
    private $anotaciones;

    /**
     * @ORM\OneToMany(targetEntity="Relacion", mappedBy="oa", cascade={"persist","remove"})
     */
    private $relaciones;

    /**
     * @ORM\OneToMany(targetEntity="Clasificacion", mappedBy="oa", cascade={"persist","remove"})
     */
    private $clasificaciones;
```

Figura 32. Codificación de la Clase OA

Luego de crear la Clase OA se procedió a desarrollar el controlador que corresponde con dicha Clase, en este caso se trata del OAController. Dentro del controlador se definieron las funciones del CRUD (Create, Read, Update, Delete) las cuales permitirán la gestión de los OACA. Es importante destacar que dentro del controlador se hizo uso de los llamados “Repositories” en Symfony2. Los “Repositories” son los archivos que contienen las consultas a Base de Datos para

cada Clase, por lo tanto, los controladores no contienen código sql, sino llamadas a funciones definidas en los “Repositories”. Además de los “Repositories” también se utilizan las Clases “Type”, estas Clases son las encargadas de mostrar los formularios de la aplicación.

Otro aspecto importante del OAController, es que se toma en cuenta el tipo de formato con que se hace la petición al servidor. Esto se hace con la finalidad de devolver al cliente una respuesta distinta dependiendo del tipo de formato, y cumplir con la entrega de recursos de acuerdo a la Arquitectura REST.

Por otro lado, también se manejaron dentro del controlador los mensajes flash, los cuales son pasados a las vistas en caso de que se genere algún error dentro de las funciones del controlador.

En la Figura 33 se muestra la función indexAction del controlador OAController

```php
<?php
    public function indexAction(){

            $mensaje = $this->getRequest()->getSession()->getFlash('mensaje');
            $error = $this->getRequest()->getSession()->getFlash('error');
            $usuario = $this->get('security.context')->getToken()->getUser();
            $format = $this->getRequest()->getRequestFormat();

            $em = $this->getDoctrine()->getEntityManager();

            switch($format){

                case 'json':
                   $objetos = $em->getRepository('ROABundle:OA')->findAll(array('fecha_publicacion'=>'DESC'));
                   $content = $this->get('serializador')->serializar($objetos, 'json');
                    break;

                default:
                    $objetos = $em->getRepository('ROABundle:OA')->findAll(array('fecha_publicacion'=>'DESC'));
                    $variables =
                    array(
                            'error'=> $error,
                            'usuario'=> $usuario,
                            'objetos'=>$objetos,
                            'mensaje' =>$mensaje
                            );

                    $content = $this->renderView('ROABundle:OA:index.html.twig', $variables);
                    break;

            }

        return new Response($content);
    }
```

Figura 33. Codificación de la función indexAction del controlador OAController

En la Figura 34 se muestra la función newAction del controlador OAController

```php
<?php
public function newAction(){

        $error = $this->getRequest()->getSession()->getFlash('error');
        $usuario = $this->get('security.context')->getToken()->getUser();
        $format = $this->getRequest()->getRequestFormat();

        $OA = New OA();

        //Creación de los formularios
        $OA_form= $this->createForm(new OAType(), $OA);

        //Variables que son pasadas a la vista
        $variables = array(
                    'error'=> $error,
                    'usuario'=> $usuario,
                    'OA_form' => $OA_form->createView());

        $content = $this->renderView('ROABundle:OA:new.html.twig', $variables);

        return new Response($content);

}
```

Figura 34. Función newAction del controlador OAController

En la Figura 35 se muestra la codificación de la función CreateAction del controlador OAController.

```php
<?php
public function createAction()
    {

        $error = $this->getRequest()->getSession()->getFlash('error');
        $usuario = $this->get('security.context')->getToken()->getUser();
        $format = $this->getRequest()->getRequestFormat();

        $OA  = new OA();
        $OA_form= $this->createForm(new OAType(), $OA);
        $OA_form->bind($request);

        if( $OA_form->isValid()){

            //Persistencia y almacenamiento en Base de Datos
            $em = $this->getDoctrine()->getEntityManager();

            if ($usuario->esContribuyente()== false){
                $role_contribuyente = $em->getRepository('ROABundle:Role')->findOneByDescripcion('Contribuyente');
                $usuario->changeToContribuyente($role_contribuyente);
            }

            $OA->setUsuario($usuario);
            $em->persist($OA);
            $em->flush();

            $variables = array('error'=> $error,'usuario'=> $usuario);

            $this->getRequest()->getSession()->setFlash('mensaje', '¡Objeto almacenado exitosamente!');

            return $this->redirect($this->generateUrl('OA_index', array());

        }

        $variables = array(
                    'error'=> $error,
                    'usuario'=> $usuario,
                    'OA_form' => $OA_form->createView(),
                    'mensaje' => '');

        return $this->render('ROABundle:OA:new.html.twig', $variables);
    }
```

Figura 35. Codificación de la función createAction del controlador OAController

En la función createAction se puede apreciar que al insertar el OACA en la Base de Datos se verifica si el usuario es No Contribuyente. En caso de ser No Contribuyente, el rol del usuario cambia a Contribuyente.

En la Figura 36 se muestra la función editAction del controlador OAController

```php
<?php
public function editAction($id)
    {

        $mensaje = $this->getRequest()->getSession()->getFlash('mensaje');
        $usuario = $this->get('security.context')->getToken()->getUser();

        $em = $this->getDoctrine()->getManager();

        $objeto = $em->getRepository('ROABundle:OA')->find($id);

        if (!$objeto) {
            throw $this->createNotFoundException('Error tratando de localizar el Objeto');
        }

        $editForm = $this->createForm(new OAType(), $objeto);
        $deleteForm = $this->createDeleteForm($id);

        return $this->render('ROABundle:OA:edit.html.twig', array(
            'objeto'      => $objeto,
            'OA_form'   => $editForm->createView(),
            'delete_form' => $deleteForm->createView(),
            'usuario' => $usuario,
            'mensaje' => $mensaje,
        ));
    }
```

Figura 36. Codificación de la función editAction del controlador OAController

En la Figura 37 se muestra la función updateAction del controlador OAController

```
<?
public function updateAction($id)
    {
        $usuario = $this->get('security.context')->getToken()->getUser();

        $em = $this->getDoctrine()->getManager();

        $objeto = $em->getRepository('ROABundle:OA')->find($id);

        if (!$objeto) {
            throw $this->createNotFoundException('Error tratando de localizar el Objeto');
        }

        $em = $this->getDoctrine()->getManager();
        $em->persist($objeto);
        $em->flush();

        $this->getRequest()->getSession()->setFlash('mensaje', '¡Objeto editado exitosamente!');

        return $this->redirect($this->generateUrl('OA_edit', array('id' => $id));
        }

        return $this->render('ROABundle:OA:edit.html.twig', array(
            'objeto'      => $objeto,
            'OA_form'   => $editForm->createView(),
            'delete_form' => $deleteForm->createView(),
            'usuario' => $usuario,
            'mensaje'=>''
        ));
    }
```

Figura 37. Codificación de la función updateAction del controlador OAController

En la Figura 38 se muestra la función deleteAction del controlador OAController

```
<?
public function deleteAction($id){

        $error = $this->getRequest()->getSession()->getFlash('error');
        $usuario = $this->get('security.context')->getToken()->getUser();

        $em = $this->getDoctrine()->getEntityManager();
        $objeto = $em->getRepository('ROABundle:OA')->find($id);
        $em->remove($objeto);
        $em->flush();

        $Oas = $usuario->getOasArray();
        if(count($Oas) == 0) {
            $role_no_contribuyente = $em->getRepository('ROABundle:Role')->findOneByDescripcion('No Contribuyente');
            $usuario->changeToNoContribuyente($role_no_contribuyente);
        }

        $em->persist($usuario);
        $em->flush();

        $variables = array('error'=> $error,'usuario'=> $usuario);

        $this->getRequest()->getSession()->setFlash('mensaje', '¡Objeto eliminado exitosamente!');

        return $this->redirect($this->generateUrl('OA_index', array()));
    }
```

Figura 38. Codificación de la función deleteAction del controlador OAController

En la función deleteAction se puede apreciar que al eliminar el OACA en la Base de Datos se verifica si el usuario no posee otros OACA almacenados. En caso de no poseer más OACA, su rol cambia a No Contribuyente.

Luego de crear todas las funciones CRUD, se creó un archivo de rutas llamado OA.yml. Este archivo contiene todas las rutas de la aplicación que corresponden a la Clase OA. A cada ruta se le asocia un nombre y una función del controlador OAController. Por otro lado, el método HTTP de acceso al recurso OA es distinto para cada función del controlador. Esto se hace con la finalidad de seguir el esquema de la Arquitectura REST.

En la Tabla 14 se muestran los métodos HTTP asociados a cada función del controlador.

Tabla 14. Métodos HTTP asociados a las funciones CRUD

Método HTTP	Funciones CRUD
POST	Create
GET	Read
PUT	Update
DELETE	Delete

En la Figura 39 se muestra el archivo con las rutas asociadas la Clase OA.

```
OA_index:
    pattern:  /OA.{_format}
    defaults: { _controller: "ROABundle:OA:index", _format: html }
    requirements:
        _method: GET
        _format: html|xml|json

OA_new:
    pattern:  /OA/new
    defaults: { _controller: "ROABundle:OA:new" }
    requirements:
        _method: GET

OA_create:
    pattern:  /OA/create
    defaults: { _controller: "ROABundle:OA:create" }
    requirements:
        _method: POST

OA_edit:
    pattern:  /OA/{id}/edit
    defaults: { _controller: "ROABundle:OA:edit" }
    requirements:
        _method: GET

OA_update:
    pattern:  /OA/{id}/update
    defaults: { _controller: "ROABundle:OA:update" }
    requirements:
        _method: PUT|POST

OA_delete:
    pattern:  OA/{id}/delete
    defaults: { _controller: "ROABundle:OA:delete" }
    requirements:
        _method: DELETE
```

Figura 39. Archivo de rutas asociadas a la Clase OA

Pruebas

Las pruebas en esta iteración se basaron en la inserción de nuevos OACA y en la modificación de OACA que ya están registrados en la Base de Datos. Además, se hicieron pruebas introduciendo datos inválidos para probar las validaciones de los campos. Por otro lado, se chequeó la persistencia y destrucción de las relaciones al insertar y eliminar OACA. Las pruebas de esta iteración se pueden apreciar en la Tabla 15.

Tabla 15. Casos de prueba - Iteración 2

Nº Caso Prueba	Módulo	Caso de Prueba	Resultado Esperado	Resultado Obtenido
1	Usuario	Insertar un nuevo OA	Crear un nuevo OA en la Base de Datos con todos sus metadatos.	Se creó correctamente el OA y todos sus metadatos asociados
2	Usuario	Colocar datos inválidos en el formulario de inserción de OA	Impedir la inserción del OA	Se impidió la inserción del OA y se mostró un mensaje de error.
3	Usuario	Editar la información de un OA existente en la Base de Datos	Mostrar todos los metadatos del OA y permitir la edición de los mismos	Se mostraron correctamente los metadatos del OA y se lograron editar exitosamente
4	Usuario	Eliminar un OA existente en Base de Datos	Eliminar el OA junto con todos sus metadatos asociados.	Se eliminó correctamente el OA y todos sus metadatos asociados.

2.4 Iteración 3

En esta iteración se continuó con el desarrollo del Módulo de Usuario, con el fin de permitir a los usuarios registrados publicar comentarios, y además, visualizar y descargar los OA almacenados.

Planificación

Iteración 3	
Descripción	Creación de la Clase y las funciones necesarias para publicar comentarios y para visualizar y descargar OACA.
Historias de Usuario	15. Crear la Clase Comentario 16. Crear el controlador comentarioController. 17. Crear una función dentro del controlador comentarioController que permita la publicación de comentarios. 18. Añadir la función de visualización en el controlador OAController 19. Añadir la función de descarga en el controlador OAController 20. Actualizar el archivo de rutas de la Clase OA
Tiempo Estimado	12 días
Fecha Inicio - Fin	1/06/2013 – 13/06/2013

Bitácora de desarrollo

La bitácora de desarrollo de la presente iteración describe el tiempo real y estimado de las actividades que se llevaron a cabo durante el tiempo planificado. En este caso se refiere al agregado de las nuevas opciones para los OACA, como lo son la publicación de comentarios, y la visualización y descarga de los recursos almacenados. La presente bitácora de desarrollo se puede apreciar en la Tabla 16.

Tabla 16. Bitácora de desarrollo - Iteración 3

Nº	Tarea	Precede	Fecha Inicio	Fecha Fin	Días estimados	Días realizados
1	Diseño de interfaz y montaje de plantillas (CSS + HTML)	N/A	1/06/2013	3/06/2013	2	2
2	Análisis e implementación	N/A	3/06/2013	9/06/2013	6	7
3	Desarrollo de la interfaz	N/A	9/06/2013	11/06/2013	2	2
4	Realización de pruebas	N/A	11/06/2013	13/06/2013	2	1

Diseño

Esta fase de la presente iteración tuvo el propósito de reflejar la publicación de comentarios en el modelo de la base de datos, así como el diseño de las interfaces relacionadas a la visualización de los OACA.

La Clase Comentario posee dos atributos importantes que son: descripción y fecha de publicación. La descripción corresponde con el contenido en sí del comentario, y la fecha de publicación se utiliza para mantener el orden cronológico. Adicionalmente, los comentarios están relacionados a un OACA en específico, de ahí se deriva una relación de uno a muchos con la Clase OA. Por otro lado, cada comentario pertenece a un determinado usuario, y cada usuario puede tener muchos comentarios asociados, por lo tanto, fue necesario crear una relación entre la Clase Comentario y la Clase Usuario.

En la Figura 40 se muestra la integración de la Clase Comentario con las Clases OA y Usuario.

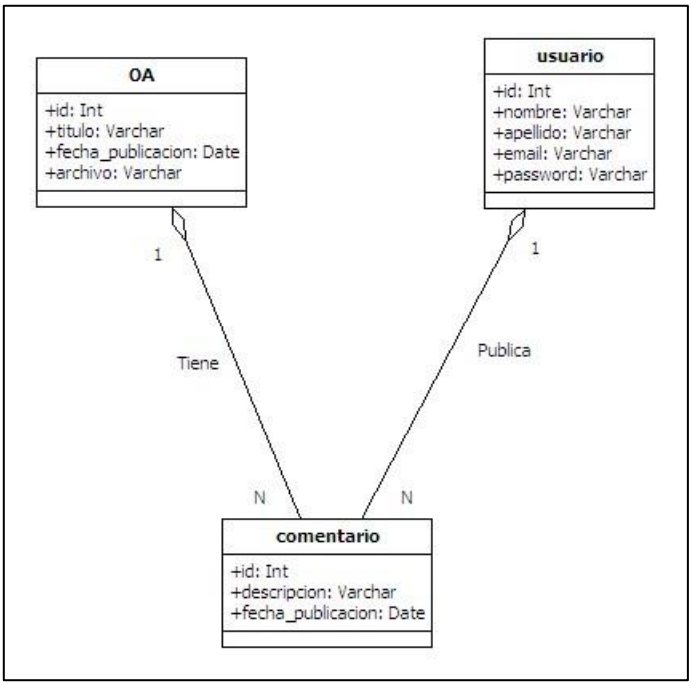


Figura 40. Integración de la Clase Comentario con las Clases OA y Usuario (Diagrama de Clases)

En cuanto a la interfaz para publicar comentarios, se empleó un diseño sencillo basado en páginas como Youtube, Facebook, Twitter, entre otras. El diseño consiste en una ventana Modal que contiene en la parte superior un campo te texto para escribir el comentario y un botón llamado “Publicar”. En la Figura 41 se puede observar dicho diseño.

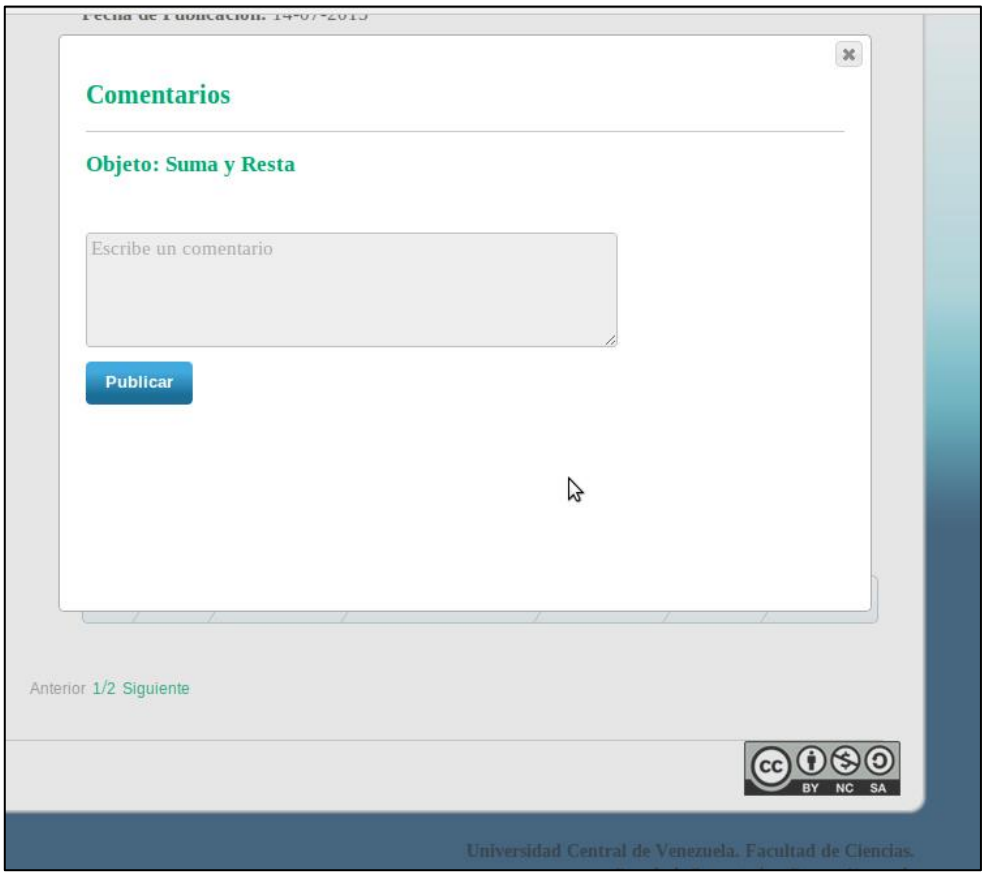


Figura 41. Interfaz usada para la publicación de comentarios

Para la interfaz de visualización de OA se tomó en cuenta el formato de los OACA. Para los formatos de audio o video, se utilizaron las respectivas etiquetas de HTML5 que permiten cargar y visualizar dichos formatos en los navegadores. En la Figura 42 se puede observar el uso de HTML5 para visualizar un OACA en formato de video.

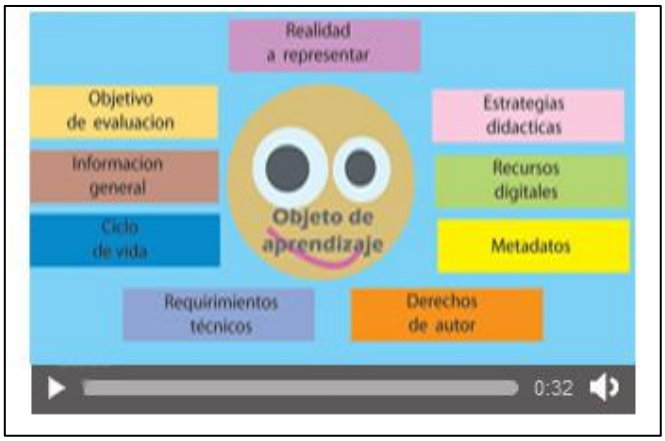


Figura 42. Visualización de un OACA en formato de video

En la Figura 43 se puede observar el uso de HTML5 para visualizar un OA en formato de audio.

.

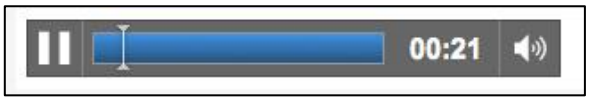


Figura 43. Visualización de un OACA en formato de audio

Los OA en cualquier otro formato se despliegan directamente en el navegador.

Codificación

Esta fase comienzo con la creación de la Clase Comentario, tomando en cuenta las relaciones con las Clases OA y Usuario que fueron descritas en la fase de diseño. En la Figura 44 se puede observar la codificación de la Clase Comentario.

```php
<?php
namespace ROA\ROABundle\Entity;
use Doctrine\ORM\Mapping as ORM;

class Comentario
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="IDENTITY")
     */
    private $id;
    /**
     * @var string
     *
     * @ORM\Column(name="descripcion", type="string", length=255)
     */
    private $descripcion;
    /**
     *
     * @ORM\Column(name="fecha_publicacion", type="datetime", nullable=true)
     */
    private $fecha_publicacion;
    /**
     * @ORM\ManyToOne(targetEntity="OA", inversedBy="comentarios")
     * @ORM\JoinColumn(name="oa_id", referencedColumnName="id")
     * @return integer
     */
    private $oa;
    /**
     * @ORM\ManyToOne(targetEntity="Usuario", inversedBy="comentarios")
     * @ORM\JoinColumn(name="usuario_id", referencedColumnName="id")
     * @return integer
     */
    private $usuario;

}
```

Figura 44. Codificación de la Clase Comentario

Luego de crear la Clase Comentario se procedió a desarrollar el controlador que corresponde con dicha Clase, en este caso se trata del controlador comentarioController. Dentro de este controlador se crearon las funciones CRUD (Create, Read, Update, Delete), aunque en principio solo se usará la función Create para crear nuevos comentarios. A diferencia de las funciones del controlador OAController que fueron desarrolladas desde cero, en esta ocasión se utilizó un comando de Doctrine para agilizar el proceso de creación de las funciones CRUD.

Dicho comando se ejecuta por consola y es el siguiente: “doctrine:generate:crud comentario.php”. Este comando no solo crea las funciones

dentro del controlador, sino que también genera todas las vistas y el archivo de configuración de rutas correspondiente a la Clase.

Es importante destacar que las llamadas al controlador comentarioController se hacen de forma asíncrona, para impedir la recarga de toda la página cada vez que un usuario desee publicar un comentario. Para realizar estas llamadas asíncronas se emplea la tecnología AJAX. Para ello se crea una función de JQuery que se encarga de manejar la petición. Dicha función recibe 4 parámetros:

a) URL: La dirección o URL de la petición.
b) Location: La ubicación dentro del documento HTML donde se cargará la respuesta obtenida del servidor.
c) BeforeSend: El nombre de la función a ejecutar antes de hacer la llamada asíncrona
d) Success: El nombre de la función a ejecutar después de hacer la llamada asíncrona.

En la Figura 45 se puede observar la función descrita anteriormente.

```
function ajax(url, location, beforeSend, success){
    $.ajax({
            type: 'GET',
            url: url,
            data: '',
            beforeSend: function(){
                eval(beforeSend);
                $('#'+location).html('Cargando...');

            },
            success: function(datos) {
                $('#'+location).html(datos);
            },
        });
        return false;
}
```

Figura 45. Función para manejar las llamadas asíncronas

Para la visualización de OACA directamente en los navegadores, se creó una función llamada visualizacionAction dentro del controlador OAController. Esta función se comporta de manera distinta dependiendo del tipo de OACA que se desee visualizar. A continuación se explica dicho comportamiento de acuerdo a lo distintos casos que se pueden presentar:

a) **OACA combinados abiertos:** estos OACA se almacenan en el Repositorio en formato comprimido (.zip), por lo tanto la función de visualización se encarga de descomprimir dichos OACA y ubicar el archivo principal que debe llevar por nombre “index”. Finalmente se hace una redirección a la ruta donde está contenido dicho archivo.

b) **OACA fundamentales:** dentro de esta categoría se presentan dos casos especiales: los OACA de tipo audio y los OACA de tipo video. A continuación se explican ambos casos.

   - ✓ **OACA de Audio:** Para los OACA cuyo formato sea de audio, se crea una vista donde se inserta la ruta del OACA dentro de una etiqueta de audio de HTML5, y luego se hace una redirección a dicha vista.

   - ✓ **OACA de Video:** El comportamiento para los OACA cuyo formato sea de video es igual al caso anterior, pero se utiliza la etiqueta de video de HTML5.

Para el resto de los casos no se realiza ninguna acción adicional, simplemente se hace una redirección a la ruta donde está contenido el OACA.

En la Figura 46 se puede observar la codificación de la función visualizacionAction.

```
public function visualizacionAction($id){
        $em = $this->getDoctrine()->getEntityManager();
        $webpath = $em->getRepository('ROABundle:OA')->find($id)->getWebPath();
        $original_name = $em->getRepository('ROABundle:OA')->find($id)->getOriginal_name();
        $original_name = explode('.', $original_name);
        $name = $original_name[0];
        $extension = $original_name[1];

        if($extension == 'zip'){
            $this->get('zip_manager')->unZip($webpath, $id);
            $index = $this->get('dir_reader')->read('uploads/objetos', 'index');
            return new RedirectResponse('http://localhost/ROA/web/downloads/'.$id.'/'.$original_name[0].'/'.$index);
        }else{
            switch ($this->get('extension_checker')->check($extension)) {
                case 'audio':
                    if (!(file_exists('downloads/'.$id.'/audio.html'))) {

                        mkdir('downloads/'.$id, 0777);
                        $fileManager = fopen('downloads/'.$id.'/audio.html', "w");
                        $data = $this->get('html_generator')->html_audio($webpath);
                        fwrite($fileManager, $data);
                    }

                    return new RedirectResponse('http://localhost/ROA/web/downloads/'.$id.'/audio.html');
                    break;

                case 'video':
                    if (!(file_exists('downloads/'.$id.'/video.html'))) {

                        mkdir('downloads/'.$id, 0777);
                        $fileManager = fopen('downloads/'.$id.'/video.html', "w");
                        $data = $this->get('html_generator')->html_video($webpath);
                        fwrite($fileManager, $data);
                    }

                    return new RedirectResponse('http://localhost/ROA/web/downloads/'.$id.'/video.html');
                    break;

                default:
                    return new RedirectResponse('http://localhost/ROA/web/'.$webpath);
                    break;
            }
        }
```

Figura 46. Codificación de la función visualizaciónAction

Adicionalmente se creó una función para descargar los OACA, la cual consiste en añadir el OACA como un archivo adjunto cuando se envía la respuesta del servidor. Esta función se puede observar en la Figura 47.

```
public function descargaAction($id){
    $em = $this->getDoctrine()->getEntityManager();
    $path = $em->getRepository('ROABundle:OA')->find($id)->getWebPath();
    $filename = $em->getRepository('ROABundle:OA')->find($id)->getPath();

    $content = file_get_contents($path);

    $response = new Response();

    $response->headers->set('Content-Type', 'text/csv');
    $response->headers->set('Content-Disposition', 'attachment;filename="'.$filename);

    $response->setContent($content);
    return $response;

}
```

Figura 47. Codificación de la función de descarga de OA

Por último, se añadieron las rutas correspondientes a las funciones de visualización y descarga dentro del archivo de rutas de la Clase OA. En la Figura 48 se puede apreciar dicho archivo de rutas.

```
OA_visualizacion:
    pattern:  /{id}/visualizacion.{_format}
    defaults: { _controller: ROABundle:OA:visualizacion, _format: html }
    requirements:
        id: \d+
        _method: POST|GET

OA_descarga:
    pattern:  /{id}/descarga.{_format}
    defaults: { _controller: ROABundle:OA:descarga, _format: html }
    requirements:
        id: \d+
        _method: POST|GET
```

Figura 48. Actualización del archivo de rutas de la Clase OA

Pruebas

En esta fase se procedió a verificar que efectivamente se pueden publicar comentarios y que las funciones de visualización y descarga funcionan correctamente para cualquier tipo de OACA.

Tabla 17. Casos de prueba - Iteración 3

Nº Caso Prueba	Módulo	Caso de Prueba	Resultado Esperado	Resultado Obtenido
1	Usuario	Publicar un comentario sobre alguno de los OACA almacenados	Asignar el comentario al OACA correspondiente.	El comentario fue añadido al OACA y se almacenó exitosamente en la Base de Datos.
2	Usuario	Visualizar en el navegador un OACA de tipo fundamental en formato de video.	Visualizar el OACA en el navegador.	El OACA se mostró correctamente en el navegador gracias al uso de la etiqueta de video de HTML5.
3	Usuario	Visualizar en el navegador un OACA de tipo fundamental en formato de audio.	Visualizar el OACA en el navegador.	El OACA se mostró correctamente en el navegador gracias al uso de la etiqueta de audio de HTML5.
4	Usuario	Visualizar un OACA combinado abierto en el navegador.	Visualizar el OACA en el navegador.	Se abrió correctamente la página principal del OACA en el navegador.
5	Usuario	Descargar un OACA	Añadir el OA como un archivo adjunto.	El OACA se añadió correctamente como un archivo adjunto.

2.5 Iteración 4

En la iteración actual se siguieron añadiendo nuevas funcionalidades al Módulo de Usuario. Inicialmente, se incorporó en el Repositorio la búsqueda avanzada de OACA, donde se puede filtrar por título, autor y año. Por otra parte, se desarrollaron dos secciones importantes: la sección de los recursos más descargados y la sección de los más recientes. Finalmente, se creó para cada OACA, una interfaz llamada ficha descriptiva donde están contenidos los metadatos. En esta ficha se incluyen además los comentarios que hayan sido publicados.

Planificación

Iteración 4	
Descripción	Búsqueda avanzada, OACA más recientes, OACA más descargados y ficha descriptiva.
Historias de Usuario	21. Diseñar la interfaz de búsqueda avanzada. 22. Programar las funciones necesarias para realizar la búsqueda avanzada. 23. Actualizar la Clase OA para manejar el número de descargas. 24. Desarrollar método para obtener los OACA más descargados. 25. Desarrollar método para obtener los OACA más recientes. 26. Diseñar interfaz de la ficha descriptiva.
Tiempo Estimado	11 días
Fecha Inicio – Fin	14/06/2013 – 25/06/2013

Bitácora de desarrollo

La bitácora de desarrollo de la iteración actual presenta las actividades que se realizaron en el período planificado para ello. Las mismas refieren al diseño de interfaz de la búsqueda avanzada, así como el desarrollo del método que permita llevar a cabo dicha búsqueda. Así mismo, se creó la sección de OACA más recientes y la sección de OA más descargados, y finalmente, se desarrolló la interfaz de la ficha descriptiva. La bitácora de desarrollo de esta iteración se puede apreciar en la Tabla 18.

Tabla 18. Bitácora de desarrollo - Iteración 4

Nº	Tarea	Precede	Fecha Inicio	Fecha Fin	Días estimados	Días realizados
1	Diseño de interfaz y montaje de plantillas (CSS + HTML)	N/A	14/06/2013	16/06/2013	2	2
2	Análisis e implementación	N/A	16/06/2013	22/06/2013	6	5
3	Desarrollo de la interfaz	N/A	22/06/2013	24/06/2013	2	2
4	Realización de pruebas	N/A	24/06/2013	25/06/2013	1	1

Diseño

En esta fase de la actual iteración, se comenzó por diseñar la interfaz para la búsqueda avanzada, la cual se compone de un pequeño formulario con tres campos: Autor, Año y Título. El usuario puede combinar estos tres campos de la manera que desee para filtrar la búsqueda, o puede dejar todos los campos en blanco y se mostrará una lista con todos los OACA contenidos en el Repositorio. En la Figura 49 se puede observar la interfaz de búsqueda avanzada.

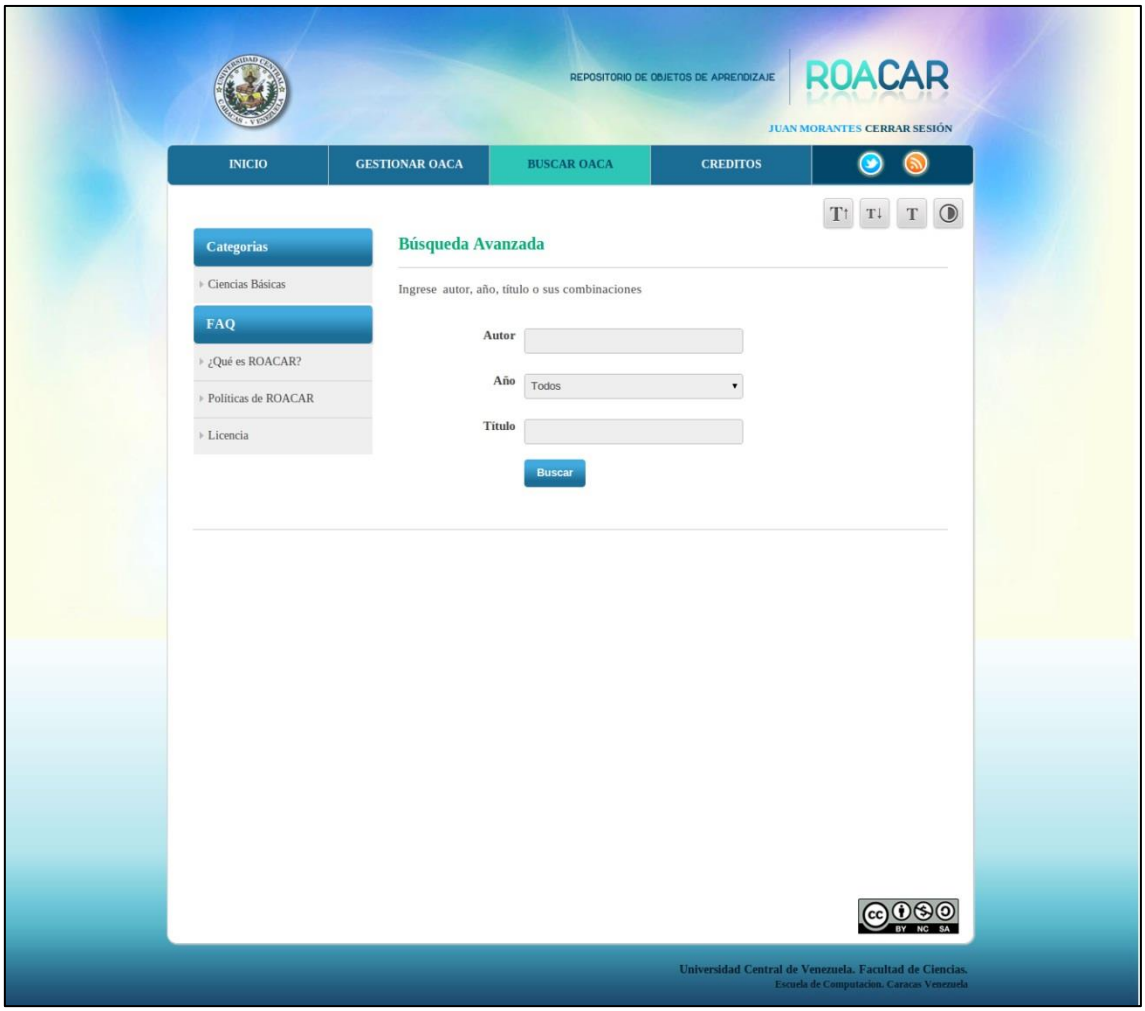


Figura 49. Formulario de búsqueda avanzada

También fue importante diseñar la interfaz de resultados de la búsqueda, la cual consiste en una lista paginada de tres OACA por página. Por cada OACA se muestran los metadatos más relevantes. En la Figura 50 se puede observar la interfaz de resultados.

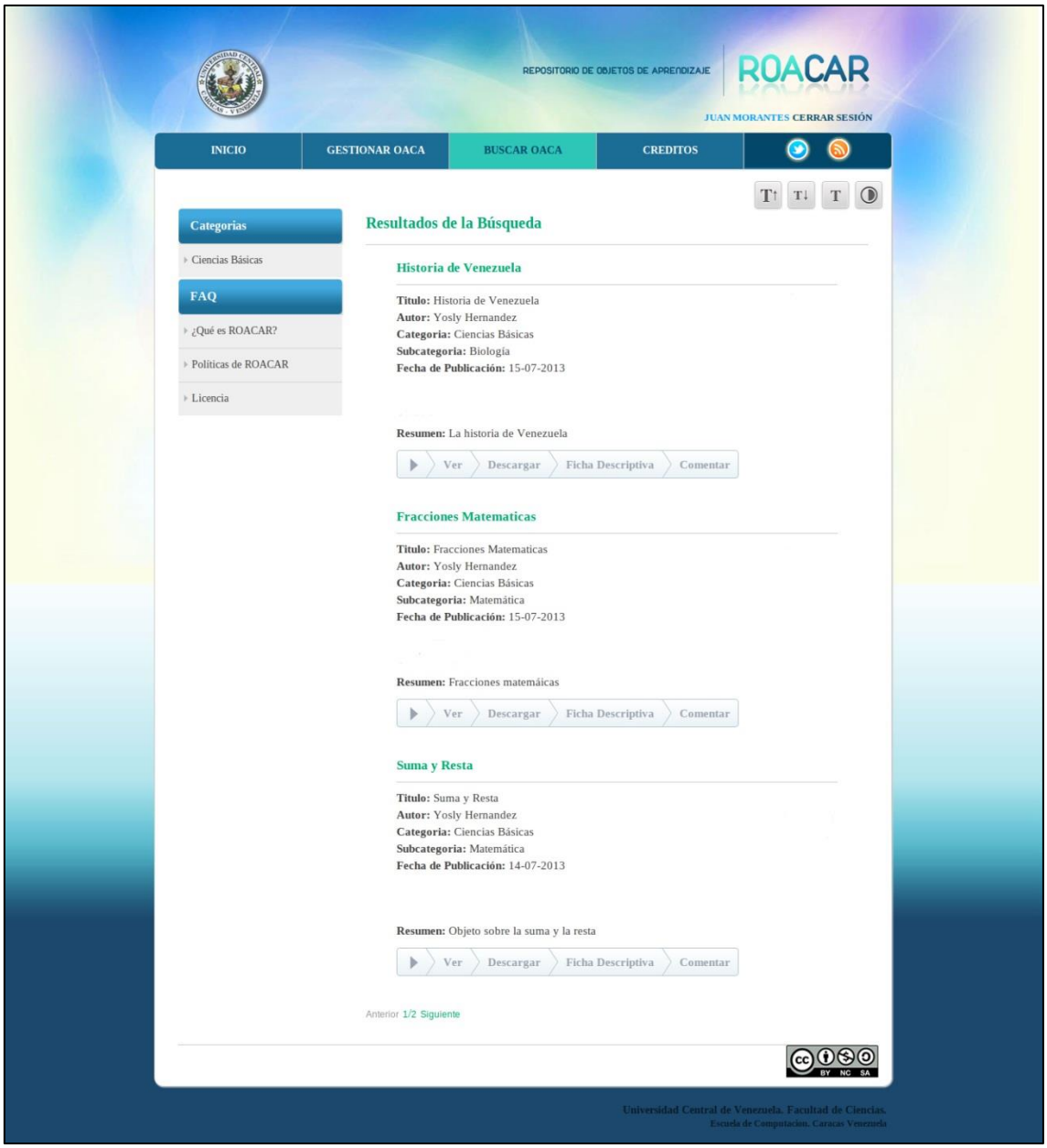


Figura 50. Interfaz de resultados de la búsqueda avanzada

Hasta este momento, no fue necesario modificar ningún atributo o método de la Clase OA, ya que los campos por los cuales se filtra la búsqueda pertenecen a la Clase. Sin embargo, para consultar los OACA más descargados hubo que tomar en cuenta ciertos aspectos que obligaron a hacer una refactorización de la Clase OA. Dichos aspectos se listan a continuación:

a) **Cada descarga debe estar asociada a una dirección ip:** esto se hace con la finalidad de impedir que se contabilicen las descargas que haga repetidamente un mismo usuario sobre un mismo OACA.
b) **No se deben repetir las direcciones ip en las tablas de la base de datos:** Si un usuario ha hecho al menos una descarga de cualquier OACA, su ip debe quedar registrada, y no se debe almacenar nuevamente en posteriores descargas.

Tomando en cuenta los aspectos señalados anteriormente, se procedió a crear una Clase para representar las direcciones ip, y se creó una relación de muchos a muchos entre dicha Clase y la Clase OA. La relación de muchos a muchos se debe a que un OACA puede ser descargado por múltiples usuarios (direcciones ip distintas), y un mismo usuario (una misma dirección ip) puede descargar varios recursos diferentes. Adicionalmente, se añade a los OACA un campo denominado "num_descargas" que se va a incrementar cada vez que el OACA se descargue desde una dirección ip distinta. En la Figura 51 se muestra la refactorización de la Clase OA para contabilizar el número de descargas por OACA.

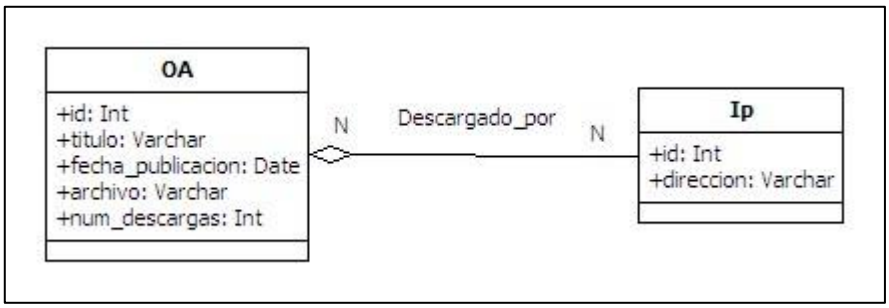


Figura 51.Diagrama de Clases con la Refactorización de la Clase OA.

Por otro lado, para crear el método que permita obtener los OACA más recientes, se debe hacer una consulta en base de datos ordenada por fecha de publicación en forma descendente. En cuanto al diseño de las vistas, se puede decir que la interfaz para los OACA más descargados y los más recientes es la misma de la búsqueda avanzada, lo único que cambia es el título de la página.

Para culminar con esta fase de la presente iteración, se creó la interfaz de la ficha descriptiva, la cual consiste en una venta Modal donde se muestran los metadatos de forma priorizada. En principio, solo se muestran los metadatos más

importantes, pero se le ofrece al usuario la opción de ver el resto de los metadatos si así lo desea. Cabe destacar que la ficha descriptiva también contiene los comentarios del OACA, los cuales se añaden en la categoría anotación de los metadatos. La interfaz de la ficha descriptiva se puede observar en las Figuras 52.

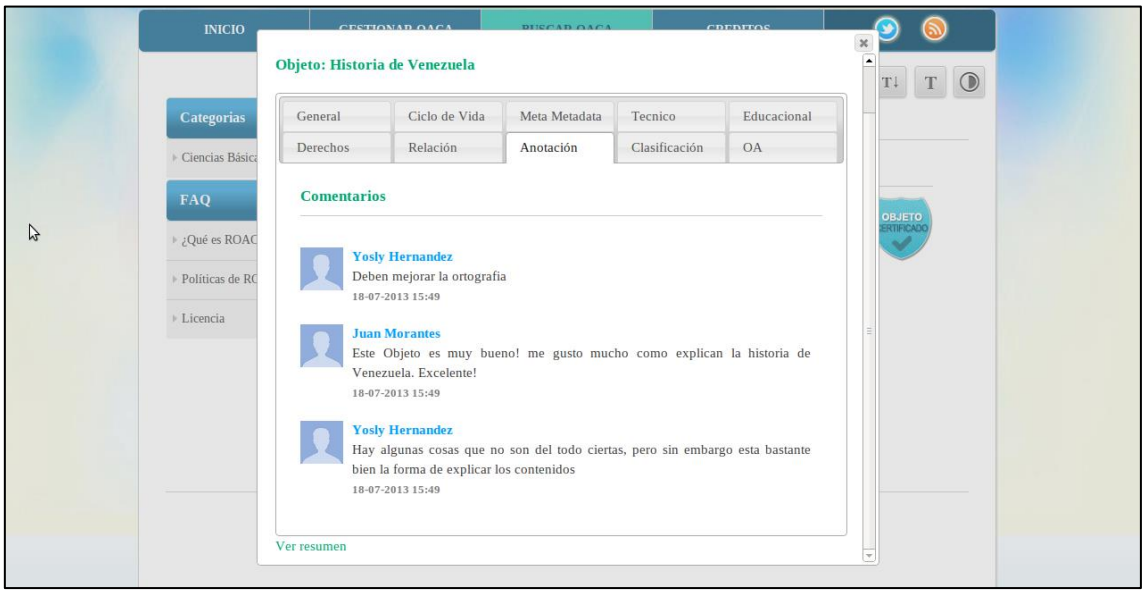


Figura 52. Interfaz de la ficha descriptiva

Codificación

La codificación de la iteración actual culminó el desarrollo del Módulo de Usuario. En primer lugar, se creó el método de búsqueda avanzada dentro del controlador OAController.

El método de búsqueda avanzada recibe las variables autor, año y título; y luego llama a una función denominada “buscar” que está definida dentro del archivo OARepository.php. La función “buscar” es la encargada de realizar la consulta en base de datos y devolver una lista de OACA filtrados por los valores introducidos por el usuario. En la Figura 53 se muestra la codificación del método de búsqueda avanzada.

```
public function busquedaAction(){

    $usuario = $this->get('security.context')->getToken()->getUser();
    $em = $this->getDoctrine()->getEntityManager();

    if($this->getRequest()->getMethod() == 'POST'){

        $request = $this->getRequest();

        /*Variables pasadas desde la vista*/
        $ano = $request->request->get('ano');
        $autor = $request->request->get('autor');
        $titulo = $request->request->get('titulo');

        $objetos = $em->getRepository('ROABundle:OA')->buscar($autor, $ano, $titulo);
        return $this->render('ROABundle:OA:resultados.html.twig', array('objetos'=>$objetos));

    }
    return $this->render('ROABundle:OA:busqueda.html.twig', $variables);
}
```

Figura 53. Codificación del método de búsqueda avanzada

El siguiente paso dentro de la fase de codificación de esta iteración consistió en refactorizar el método de descarga para poder llevar una contabilización adecuada. Antes de completar la descarga y devolver una respuesta al usuario se realizan las siguientes actividades dentro del método de descarga definido en el controlador:

a) Se obtiene la dirección ip del usuario conectado a través de la función "getClientIp()" de Symfony2.
b) Se añade un condicional que, haciendo uso de la Clase Ip, verifica si la dirección ip del usuario ya está registrada.
c) Si la dirección ip no está registrada se procede a añadirla en la Base de Datos y luego se incrementa el número de descargas para el OACA a través de una función definida previamente en la Clase OA. Dicha función lleva por nombre "IncrementarNum_descargas()".
d) Si la dirección ip ya está registrada, se añade un segundo condicional donde se verifica si ya existe una asociación entre dicha ip y el OACA en cuestión. Si ya la dirección ip está asociada al OACA no se incrementa el número de descargas, pero en caso contrario si se realiza la contabilización.

En la Figura 54 se puede observar la refactorización del método de descarga.

```
public function descargaAction($id){
        $em = $this->getDoctrine()->getEntityManager();
        $OA = $em->getRepository('ROABundle:OA')->find($id);
        $filename = $OA->getPath();
        $content = file_get_contents($path);
        $response = new Response();
        $response->headers->set('Content-Type', 'text/csv');
        $response->headers->set('Content-Disposition', 'attachment;filename="'.$filename);

        $response->setContent($content);

        /*Se obtiene la direccion ip del usuario conectado*/
        $direccion_ip = $this->get('request')->getClientIp();

        $ip = $em->getRepository('ROABundle:Ip')->findOneByDireccion($direccion_ip);

        /*Si le direccion ip no esta registrada se agrega*/
        if (!($ip)){

            $ip = New Ip();
            $ip->setDireccion($direccion_ip);
            $em->persist($ip);
            $em->flush();

            $OA->addIp($ip);
            $OA->IncrementarNum_descargas();
            $em->persist($OA);
            $em->flush();

        }else{

            /*Si la direccion ip no esta asociada al objeto se incrementa el numero de descargas*/
            if (!($OA->existeIp($ip))){
                $OA->addIp($ip);
                $OA->IncrementarNum_descargas();
            }
            $em->persist($OA);
            $em->flush();
        }
        return $response;
```

Figura 54. Refactorización del método de descarga

Una vez terminada la refactorización anterior, se facilitó el desarrollo del método para obtener los OACA más descargados, ya que solo basta con realizar una consulta por número de descargas estableciendo un límite máximo de 3 registros. Dicha consulta se realiza a través de una función llamada “BuscarMas_descargados()” que está definida en el archivo OARepository.php. En la Figura 55 se puede observar la codificación del método para obtener los OACA más descargados.

```
public function mas_descargadosAction(){

    $usuario = $this->get('security.context')->getToken()->getUser();

    $em = $this->getDoctrine()->getEntityManager();

    //Funcion para buscar los 3 Objetos mas descargados
    $objetos = $em->getRepository('ROABundle:OA')->BuscarMas_descargados(3);

    return $this->render('ROABundle:OA:mas_descargados.html.twig', array('objetos'=>$objetos));

}
```

Figura 55. Codificación del método para obtener los OA más descargados

En la Figura 56 se muestra la codificación de la función "BuscarMas_descargados()", la cual es invocada por el método descrito anteriormente.

```
public function BuscarMas_descargados($limit){

    $em = $this->getEntityManager();
    $dql = "select o from ROABundle:OA o ORDER BY o.num_descargas DESC";
    $query = $em->createQuery($dql);
    $query->setMaxResults($limit);
    $OAs = $query->getResult();
    return $OAs;

}
```

Figura 56. Codificación de la función BuscarMas_descargados()

Seguidamente, se creó el método para obtener los OACA publicados recientemente. En este caso, se realizó el mismo procedimiento del método anterior, pero cambiando el atributo de número de descargas por la fecha de publicación. En este sentido, la consulta devuelve los tres últimos OACA que fueron publicados en el Repositorio. En la Figura 57 se puede observar la codificación del método para obtener los OACA más recientes.

```
public function recientesAction(){

    $usuario = $this->get('security.context')->getToken()->getUser();

    $em = $this->getDoctrine()->getEntityManager();

    //Funcion para buscar los 3 Objetos mas recientes
    $objetos = $em->getRepository('ROABundle:OA')->BuscarRecientes(3);

    return $this->render('ROABundle:OA:recientes.html.twig', array('objetos'=>$objetos));

}
```

Figura 57. Codificación del método para obtener los OA más recientes

En la Figura 58 se muestra la codificación de la función “BuscarRecientes ()”, la cual es invocada por el método descrito anteriormente.

```
public function BuscarRecientes($limit){

    $em = $this->getEntityManager();
    $dql = "select o from ROABundle:OA o ORDER BY o.fecha_publicacion DESC";
    $query = $em->createQuery($dql);
    $query->setMaxResults($limit);
    $OAs = $query->getResult();
    return $OAs;

}
```

Figura 58. Codificación de la función BuscarRecientes()

Para culminar con esta fase de la presente iteración, se desarrolló el método que permite obtener los metadatos que son mostrados en la ficha descriptiva. Este método realiza las siguientes acciones:

a) Busca el OACA y sus metadatos a través del id que es pasado por parámetro.
b) Busca todos los comentarios asociados al OACA en cuestión.
c) Le pasa a la vista el OACA y sus comentarios.

Finalmente, en la vista de la ficha descriptiva se muestran todos los metadatos que fueron pasados desde el controlador. Los metadatos se organizan de acuerdo a las 9 categorías del estándar LOM, y se añaden los comentarios en la categoría Anotación, tal como fue descrito en la fase de diseño de esta iteración. En la Figura 59 se muestra el método para obtener la ficha descriptiva.

```
public function metadatosAction($id){

    $em = $this->getDoctrine()->getEntityManager();
    $objeto = $em->getRepository('ROABundle:OA')->find($id);
    $comentarios = $em->getRepository('ROABundle:Comentario')->findByOa($id);
    return $this->render('ROABundle:OA:metadatos.html.twig',
                array('objeto'=>$objeto, 'comentarios'=>$comentarios));
}
```

Figura 59. Método para obtener la ficha descriptiva

Pruebas

En esta fase se llevaron a cabo las pruebas para verificar el correcto funcionamiento de la búsqueda avanzada, realizando varios filtros y corroborando la veracidad de los resultados obtenidos. Del mismo modo, se verificó la veracidad de los OACA más recientes. Por otro lado, se comprobó que las descargas de los OACA se contabilizaran correctamente y que no se tomaran en cuenta las direcciones ip repetidas, de modo que la lista de los más descargados contuviera los datos correctos. Finalmente, se realizaron pruebas en la ficha descriptiva, asegurándose de que los metadatos se mostraran correctamente.

Las pruebas mencionadas pueden apreciarse en la siguiente Tabla 19.

Tabla 19. Casos de prueba - Iteración 4

Nº Caso Prueba	Módulo	Caso de Prueba	Resultado Esperado	Resultado Obtenido
1	Usuario	Realizar una búsqueda avanzada.	Obtener una lista de OACA que corresponden con el filtro aplicado en la búsqueda avanzada.	Se obtuvo correctamente la lista de todos los OACA que coincidían con los parámetros establecidos en la búsqueda.
2	Usuario	Realizar una búsqueda avanzada sin aplicar ningún filtro.	Obtener una lista de todos los OACA almacenados en el Repositorio.	Se Obtuvo correctamente la lista de todos los OACA.
3	Usuario	Obtener los OACA más recientes.	Obtener una lista con los tres últimos OACA que fueron publicados.	Se obtuvo correctamente la lista de los OACA más recientes, ordenados por su fecha de publicación.

4	Usuario	Descargar varias veces un OACA desde una misma dirección ip.	Incrementar una sola vez el número de descargas del OACA.	Se incrementó una sola vez el número de descargas del OACA, a pesar de que se realizaron varias descargas.
5	Usuario	Obtener los OACA más descargados	Obtener una lista con los tres OACA más descargados.	Se obtuvo correctamente la lista de los tres OACA más descargados, ordenados por el número de descargas.
6	Usuario	Visualizar la ficha descriptiva del OACA	Mostrar en una ventana Modal los metadatos del OACA en forma priorizada.	Se mostraron correctamente todos los metadatos del OACA.

2.6 Iteración 5

Le presente iteración se centró en el desarrollo del Módulo de Administración, cuyo objetivo principal es la gestión de usuarios, contenidos y solicitudes de publicación de OACA y comentarios en el Repositorio. Para empezar, se diseñó la interfaz principal de este módulo y se crearon los métodos para gestionar los contenidos que se muestran en el módulo desarrollado en las iteraciones anteriores (Módulo de Usuario).

Planificación

Iteración 5	
Descripción	Desarrollo de la interfaz principal del Módulo de Administración y desarrollo de las funciones CRUD de las categorías de los OACA.
Historias de Usuario	27. Diseñar la interfaz principal del Módulo de Administración. 28. Desarrollar la Clase Categoría. 29. Desarrollar las funciones CRUD de las categorías de los OACA. 30. Desarrollar las vistas correspondientes a la Clase Categoría.
Tiempo Estimado	9 días
Fecha Inicio – Fin	26/06/2013 – 5/07/2013

Bitácora de desarrollo

La bitácora de desarrollo de la presente iteración muestra las actividades que se realizaron en el período planificado, las cuales refieren a la creación de la interfaz del Módulo de Administración y al desarrollo de las funciones CRUD (Create, Read, Update, Delete) de las Categorías a las cuales pertenecen los OACA. Puede apreciarse esta información en la Tabla 20.

Tabla 20. Bitácora de desarrollo - Iteración 5

Nº	Tarea	Precede	Fecha Inicio	Fecha Fin	Días estimados	Días realizados
1	Diseño de interfaz y montaje de plantillas (CSS + HTML)	N/A	26/06/2013	28/06/2013	2	2
2	Análisis e implementación	N/A	28/06/2013	2/07/2013	4	4
3	Desarrollo de la interfaz	N/A	2/07/2013	4/07/2013	2	2
4	Realización de pruebas	N/A	4/07/2013	5/07/2013	1	1

Diseño

La fase de diseño se inició con la creación de la interfaz del Módulo de Administración, la cual es muy similar a la del Módulo de Usuario, con la diferencia de que en este caso se sustituyen las opciones del menú principal por las siguientes:

a) **Inicio:** Corresponde a la página de bienvenida del Módulo de Administración.
b) **Usuarios:** A través de esta opción se gestionan los usuarios registrados en el Repositorio, y se dividen de acuerdo al rol que posean (Contribuyentes y No Contribuyentes).
c) **Contenidos:** Esta sección está destinada a gestionar los diferentes contenidos que se muestran en el Módulo de Usuario, como por ejemplo, el texto de bienvenida, la información de los créditos, etc. Sin embargo, el presente TEG sólo abarcará la gestión de las categorías de los OACA.
d) **Solicitudes:** Esta sección contiene las solicitudes que hagan los usuarios para publicar OACA o comentarios en el Repositorio.

En la Figura 60 se muestra la interfaz principal del Módulo de Administración y las opciones del menú mencionadas anteriormente.

Figura 60. Interfaz principal del Módulo de Administración

Posteriormente se procedió a diseñar la Clase correspondiente a las categorías, tomando en cuenta los siguientes aspectos:

a) Cada categoría posee varias subcategorías (relación de uno a muchos).
b) Cada OACA pertenece a una determinada subcategoría de la categoría padre.
c) Cada subcategoría puede tener varios OACA asociados (relación de uno a muchos).

En base a esto, se puede observar que para crear la Clase Categoría fue necesario crear también una Clase para las subcategorías. Además, es importante establecer una relación de uno a muchos entre Subcategoría y OA; y al mismo tiempo una relación de uno a muchos entre Categoría y Subcategoría. El diseño de la Clase Categoría junto con sus relaciones se puede observar en la Figura 61.

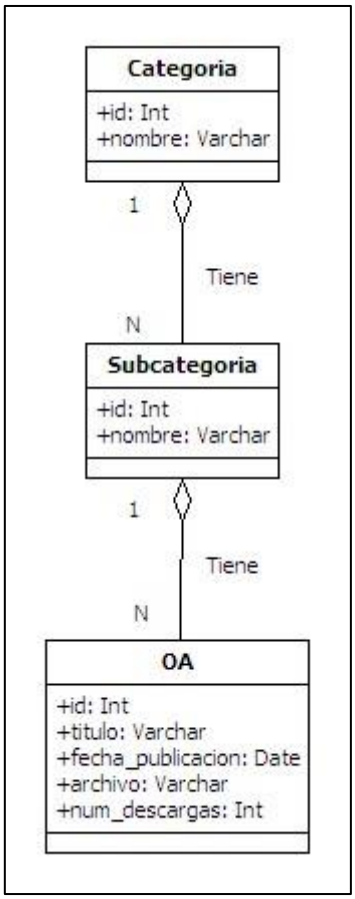


Figura 61. Diagrama de Clases de Categoría, Subcategoría y OA.

Seguidamente se desarrollaron las interfaces correspondientes a la Clase Categoría, las cuales se listan a continuación:

a) **Index:** Muestra la lista de todas las categorías junto con los botones de ver, editar y crear nueva categoría.

b) **New:** A través de esta vista se crean nuevas categorías. A cada categoría se le puede ir agregando la cantidad de subcategorías que se desee a través de un formulario anidado.

c) **Edit:** A través de esta vista se puede editar la información de la categoría seleccionada. Igualmente, se pueden eliminar, editar o añadir nuevas subcategorías utilizando el formulario anidado

d) **Show:** Esta vista muestra el nombre de la categoría seleccionada junto con una lista de las subcategorías que tiene asociadas.

Codificación

La fase de codificación de la iteración actual se inició con la creación de la interfaz de la página principal del Módulo de Administración, para lo cual se tomó como base la interfaz del Módulo de Usuario y se hicieron las adaptaciones y cambios pertinentes de acuerdo a lo explicado en la fase de diseño de la actual iteración.

Seguidamente, se creó la Clase Categoría, la cual se muestra en la Figura 62

```php
namespace ROA\ROABundle\Entity;
use Doctrine\ORM\Mapping as ORM;

/**
 * Categoria
 *
 * @ORM\Table()
 * @ORM\Entity(repositoryClass="ROA\ROABundle\Repository\CategoriaRepository")
 */
class Categoria
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="nombre", type="string", length=255)
     */
    private $nombre;

    /*RELACION DE UNO A MUCHOS CON SUBCATEGORIA*/
    /**
     * @ORM\OneToMany(targetEntity="Subcategoria", mappedBy="categoria", cascade={"persist","remove"})
     */
    private $subcategorias;
}
```

Figura 62. Codificación de la Clase Categoría

Como se puede observar en la Figura 62, se establece una relación de uno a muchos entre la Clase Categoría y Subcategoría, tal como se mencionó en la fase de diseño de la actual iteración. Además, se añadió en la Clase la eliminación en cascada, lo que permite eliminar todas las subcategorías asociadas cuando se elimina una categoría padre.

A continuación se creó la Clase Subcategoría, la cual se muestra en la Figura 63

```php
namespace ROA\ROABundle\Entity;
use Doctrine\ORM\Mapping as ORM;
/**
 * Subcategoria
 *
 * @ORM\Table()
 * @ORM\Entity(repositoryClass="ROA\ROABundle\Entity\SubcategoriaRepository")
 */
class Subcategoria {
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="nombre", type="string", length=255)
     */
    private $nombre;

    /*RELACION DE UNO A MUCHOS CON OA*/

    /**
     * @ORM\OneToMany(targetEntity="OA", mappedBy="subcategoria", cascade={"persist", "remove"})
     */
    private $oas;

    /*RELACION DE MUCHOS A UNO CON CATEGORIA*/

    /**
     * @ORM\ManyToOne(targetEntity="Categoria", inversedBy="subcategorias")
     * @ORM\JoinColumn(name="categoria", referencedColumnName="id")
     * @return integer
     */
    private $categoria;

}
```

Figura 63. Codificación de la Clase Subcategoría

Como se puede observar en la Figura 63, se creó una relación de uno a muchos con la Clase OA, tal como se mencionó en la fase de diseño de esta iteración. Y nuevamente se establece la eliminación en cascada, con lo cual se eliminan todos los OACA asociados a una subcategoría cuando ésta última es eliminada.

Una vez creado las Clases Categoría y Subcategoría, se procedió a crear el controlador CategoriaController junto con los métodos del CRUD (Create, Read, Update, Delete). En esta oportunidad se empleó nuevamente el comando “doctrine:generate:crud” de Doctrine. Dicho comando fue utilizado en la iteración 3 para crear los métodos CRUD de la Clase Comentario, por lo tanto no se ahondará nuevamente en el desarrollo de dichas funciones.

Por otro lado, se refactorizó la vista de la página principal del Módulo Usuario, permitiendo que las categorías que se muestran en el menú lateral, se carguen

directamente de Base de Datos y no como un contenido estático tal como estaba diseñado hasta este momento. Para hacer el recorrido por todas las categorías y subcategorías dentro de la vista, se hace uso de los iteradores del motor de plantillas Twig. En la Figura 64 se puede observar el uso de Twig para mostrar la lista de categorías.

```
<div class="menu_lateral">
    <ul id="acordion">
    <li><div class="cabecera_acordion">Categorias</div></li>
        {% for categoria in categorias %}
            <li><div>{{ categoria.nombre }}</div>
                <ul>
                {% for subcategoria in categoria.subcategorias %}
                    <li class="submenu_acordion"><div><a href="{{path('OA_index', {'filtro':'subcategoria', 'id':
                    subcategoria.id})}}" class="">{{ subcategoria.nombre}}</a></div></li>
                {% endfor %}
                </ul>
            </li>
        {% endfor %}
    </ul>
</div>
```

Figura 64. Uso de Twig para mostrar las categorías cargadas desde Base de Datos

Para terminar con la codificación de esta iteración, se realizó también una refactorización al método buscar definido en el controlador OAController. El objetivo de esta refactorización fue añadir un parámetro adicional al método mencionado anteriormente, para permitir que las búsquedas avanzadas también se puedan filtrar por categoría.

Pruebas

En esta fase se llevaron a cabo las pruebas para comprobar el correcto funcionamiento de las funciones para listar, crear, editar y eliminar categorías. De igual manera, se comprobó que en la página principal del Módulo de Usuario se cargaran las categorías creadas previamente a través del Módulo de Administración.

A continuación se observa la Tabla 21 con los casos de prueba mencionados.

Tabla 21. Casos de prueba - Iteración 5

Nº Caso Prueba	Módulo	Caso de Prueba	Resultado Esperado	Resultado Obtenido
1	Administración	Consultar las categorías existentes.	Obtener una lista con todas las categorías.	Se listaron correctamente todas las categorías.
2	Administración	Crear una nueva categoría.	Crear una nueva categoría junto con sus respectivas subcategorías.	Se creó correctamente la categoría y sus subcategorías asociadas.
3	Administración	Editar una categoría existente.	Editar la información de una categoría existente y sus subcategorías.	Se editó correctamente la información de la categoría y sus subcategorías asociadas.
4	Administración	Eliminar una categoría existente	Eliminar una categoría existente y también todas las subcategorías que tenga asociadas.	Se eliminaron correctamente tanto la categoría como las subcategorías asociadas.
5	Usuario	Consultar las categorías y subcategorías en el menú lateral de la página principal.	Mostrar en el menú lateral todas las categorías y subcategorías.	Se mostraron correctamente todas las categorías y subcategorías en el menú lateral de la página principal

2.7 Iteración 6

La presente iteración conforma la última serie de actividades del desarrollo de ROACAR. Se agregó la funcionalidad de gestionar los usuarios y de aprobar o no los comentarios y/o recursos que se deseen publicar en el Repositorio. Por otro lado, también se añadió la funcionalidad de puntuar o valorar los OACA de acuerdo al estándar Learning Object Review Instrument (LORI). Todas estas funcionalidades son manejadas desde el Módulo de Administración.

Planificación

Iteración 6	
Descripción	Valoración de los recursos, gestión de usuarios y gestión de solicitudes para comentar y/o publicar OACA en el Repositorio.
Historias de Usuario	31. Desarrollar las funciones del CRUD de usuarios. 32. Añadir un nuevo campo en la Clase OA que corresponda con la valoración del OACA. 33. Añadir un nuevo campo en la Clase OA que corresponda con el estado del OACA (en revisión, certificado y no autorizado). 34. Añadir un nuevo campo en la Clase Comentario que corresponda con su estado (publicado o no publicado). 35. Actualizar las consultas de Base de Datos referentes a los OACA y los comentarios
Tiempo Estimado	8 días
Fecha Inicio – Fin	6/07/2013 – 14/07/2013

Bitácora de desarrollo

En la presente iteración se definió la última bitácora de desarrollo, que define las actividades que se realizaron en un tiempo estimado, las cuales refieren a la gestión de usuarios, valoración de OACA y aprobación de recursos y comentarios. En la Tabla 22 se presenta el tiempo planificado y real del desarrollo de estas actividades.

Tabla 22. Bitácora de desarrollo - Iteración 6

Nº	Tarea	Precede	Fecha Inicio	Fecha Fin	Días estimados	Días realizados
1	Diseño de interfaz y montaje de plantillas (CSS + HTML)	N/A	6/07/2013	7/07/2013	1	1
2	Análisis e implementación	N/A	7/07/2013	12/07/2013	5	7
3	Desarrollo de la interfaz	N/A	12/07/2013	13/07/2013	1	1
4	Realización de pruebas	N/A	13/07/2013	14/07/2013	1	1

Diseño

Esta fase de la presente iteración tuvo el propósito de diseñar las vistas necesarias para gestionar los usuarios, así como también las interfaces que permitan aprobar o rechazar los comentarios y OACA.

Antes de comenzar el desarrollo de dichas interfaces, fue necesario agregar algunos atributos nuevos en las Clases OA y Comentario. En la Clase OA se añadieron dos atributos: status y valoración. En la Figura 65 se puede observar cómo queda la Clase OA luego de estos cambios.

Figura 65. Clase OA con los atributos status y valoración

Del mismo modo, se modificó la Clase Comentario y se añadió un atributo nuevo llamado status. En la Figura 66 se puede observar el resultado de añadir este nuevo atributo.

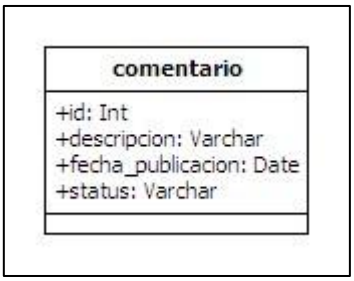


Figura 66. Clase Comentario con el atributo status

Luego de haber modificado las Clases anteriormente señaladas, se procedió a desarrollar las interfaces. En las Figuras 67 y 68 se muestran las interfaces donde se listan los usuarios registrados. Cabe destacar que los usuarios se muestran de acuerdo al rol que posean. En este caso se crean dos listados, uno con los usuarios contribuyentes y otro con los no contribuyentes.

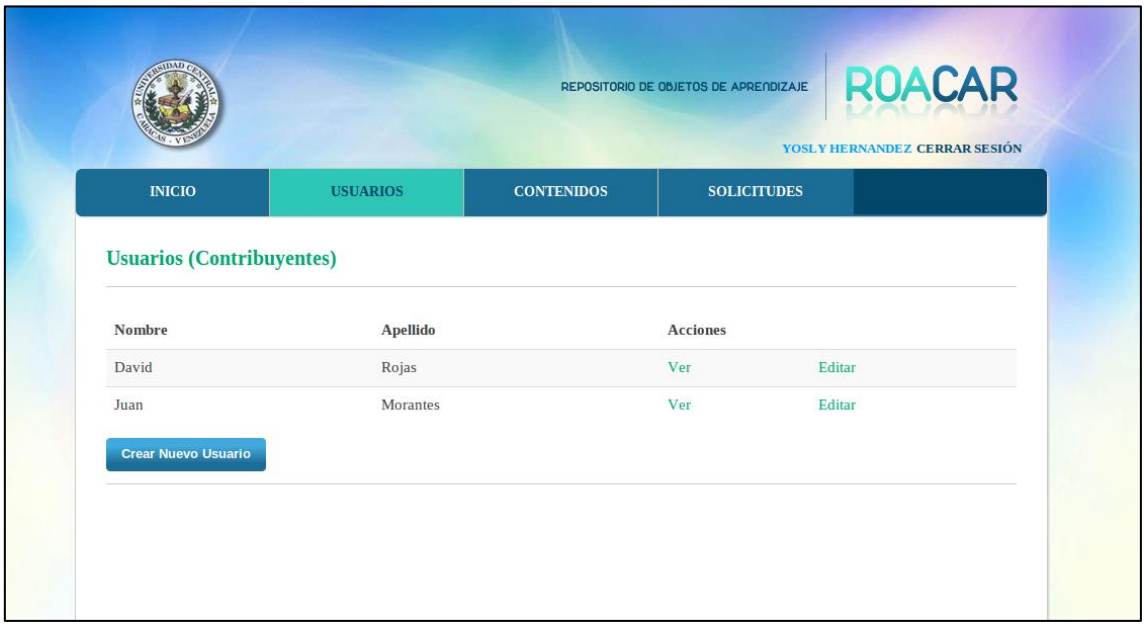


Figura 67. Lista de usuarios contribuyentes

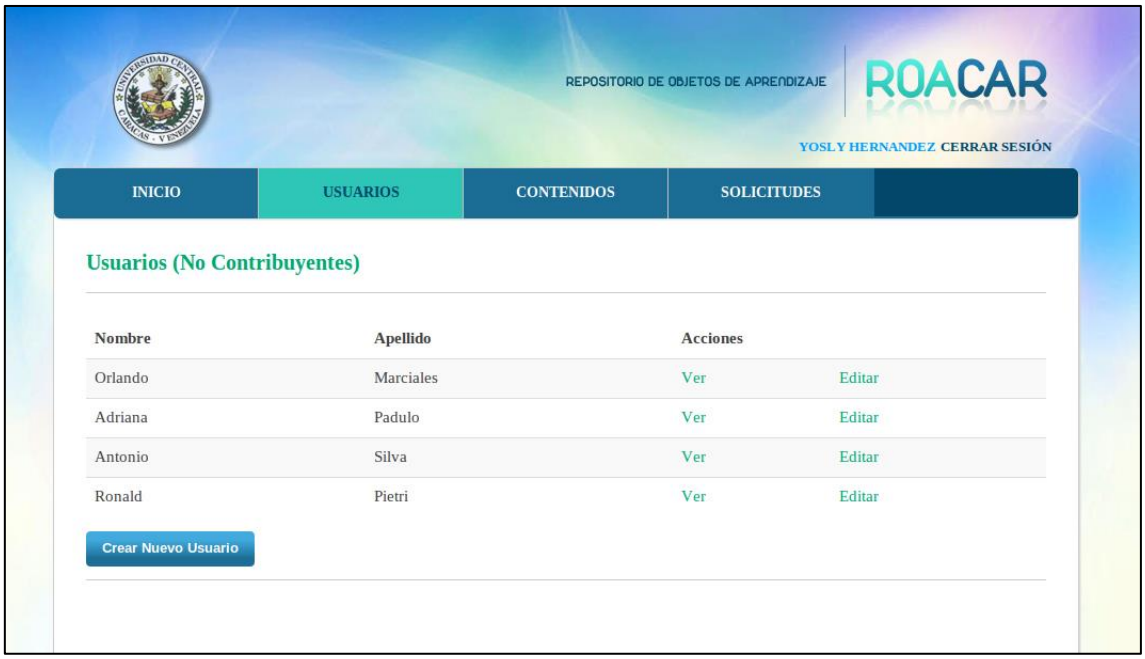


Figura 68. Lista de usuarios no contribuyentes

Seguidamente, se desarrollaron las interfaces que permitirán gestionar los OACA desde el Módulo de Administración. En la Figura 69 se puede ver la interfaz que muestra la lista de todos los OACA almacenados en el Repositorio.

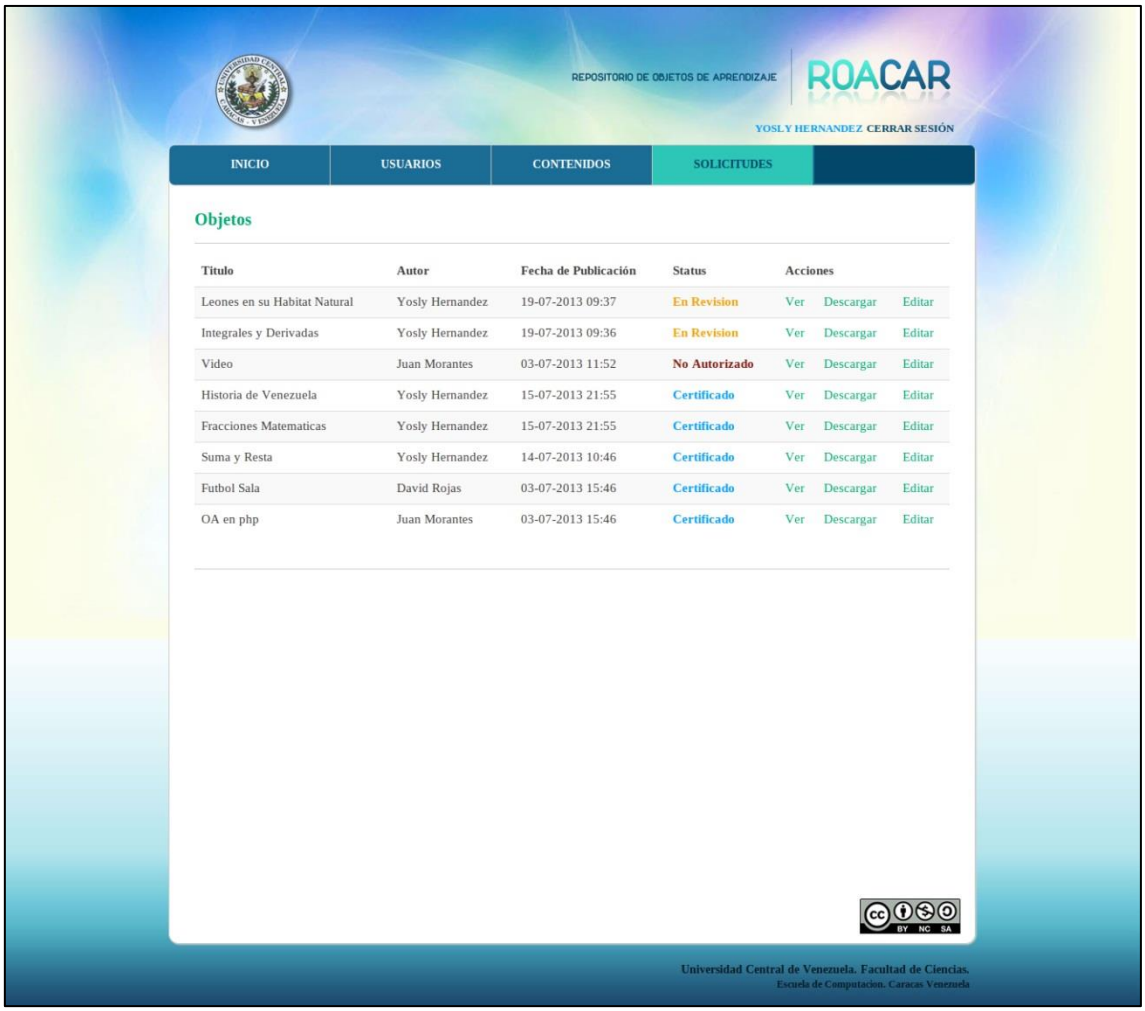


Figura 69. Interfaz para mostrar todos los OACA almacenados

Como se puede ver en la Figura 69, todos los OACA poseen un campo denominado status. El significado de dicho campo se explica a continuación:

a) **En Revisión:** El OACA está siendo sometido a un proceso de revisión para saber si cumple con los requisitos mínimos de calidad de ROACAR. Mientras el OACA está en revisión no se muestra en las publicaciones y tampoco se le permite al autor editarlo hasta tanto no sea Certificado.
b) **Certificado:** El OACA ha sido revisado y cumple con los requisitos mínimos de calidad de ROACAR, por lo tanto ya está visible en las publicaciones. Los OACA certificados pueden ser editados por sus autores, sin embargo, una vez que se editen vuelven a ser sometidos a un proceso de revisión.
c) **No Autorizado:** El OACA no cumple con los requisitos mínimos de calidad de ROACAR o viola alguna de las políticas del Repositorio. En este caso el

autor debe ponerse en contacto con el administrador o para conocer los detalles y tomar las acciones pertinentes.

Es importante destacar que el estado por defecto de un OACA al ser insertado es “En Revisión”, y cada vez que el Administrador edita el estado del recurso se envía un correo al usuario para mantenerlo al tanto.

Por otra parte, para cambiar el estado del OACA, se debe presionar el botón de editar y acto seguido se debe cambiar el valor del campo status. En la Figura 70 se muestra la interfaz para cambiar el estado del OACA.

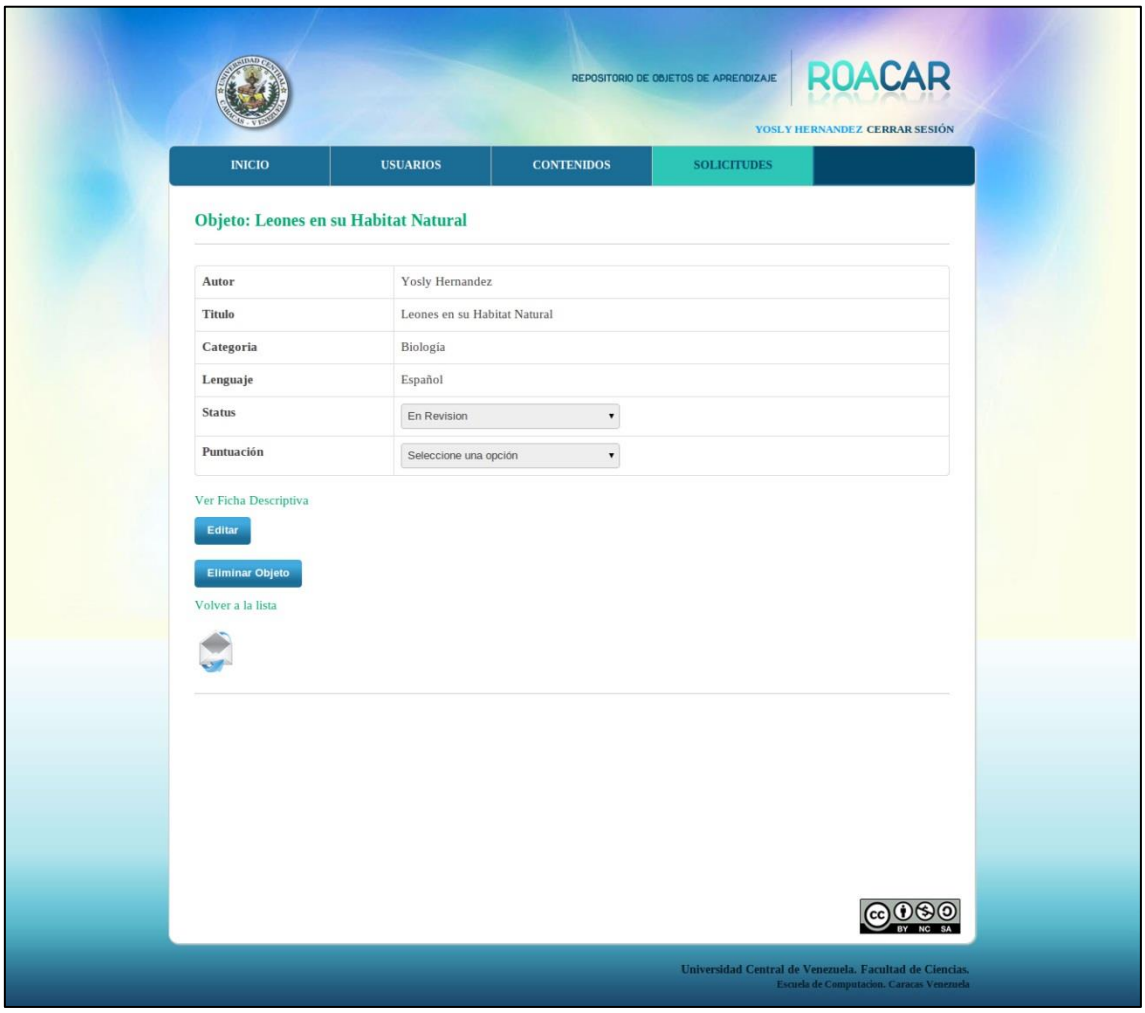


Figura 70. Interfaz para editar OACA desde el Módulo de Administración

Como se puede ver en la Figura 70, además de cambiar el estado del OACA, también se le puede asignar la valoración al recurso directamente en esta interfaz. Dicha valoración corresponde a una escala del 1 al 5 definida por el estándar LORI.

Otra de las opciones importantes que ofrece esta vista, es la de enviar un correo al usuario haciendo click en el ícono del correo ubicado en la parte inferior.

Adicionalmente, se crean 3 logos que representen cada uno de los posibles estados del OACA. Estos logos serán añadidos junto con la valoración de los recursos en todas las interfaces del Módulo de Usuario donde se muestren listas de OACA. En las Figura 71 se pueden observar dichos logos.

Figura 71. Logos para representar los diferentes estados de un OA

Finalmente, se creó la interfaz para listar los comentarios y editar su estado. El estado de un comentario puede ser uno de los siguientes:

a) **No publicado:** Este estado corresponde a los comentarios que aún no han sido revisados y aprobados por el Administrador. Por lo tanto, todavía no están visibles en los metadatos del OACA.
b) **Publicado:** Este estado corresponde a aquellos comentarios que ya fueron revisados y aprobados por el Administrador. Estos comentarios ya están presentes en los metadatos del OACA.

Cabe destacar que al añadir un comentario se le asigna por defecto el estado “No Publicado”.

En la Figura 72 se muestra la interfaz para ver la lista de comentarios.

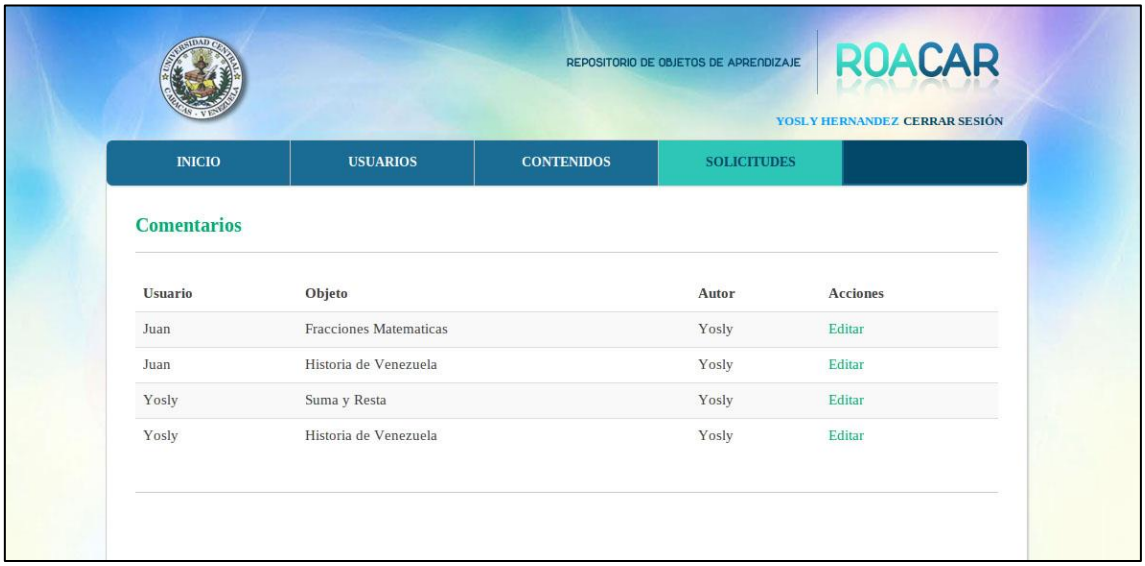


Figura 72. Interfaz para ver la lista de comentarios desde el Módulo de Administración

Codificación

La fase de codificación de la iteración actual se inició con la edición de las Clases OA y Comentario, para añadir los atributos que fueron mencionados en la etapa de diseño de la presente iteración.

Seguidamente, se procedió a editar las consultas de Base de Datos relacionadas tanto con los OACA como con los comentarios, con la finalidad de añadir un filtro que permitiera retornar únicamente los OACA certificados y los comentarios publicados. Esto se realiza en los llamados “Repositories” de Symfony2, que son los archivos que contienen las consultas a Base de Datos.

Es importante resaltar, que solo se modificaron las consultas que se realizan del lado del Módulo de Usuario, exceptuando la sección de “Mis OACA” donde se deben retornar todos los OACA del usuario independientemente del estado de los mismos.

Para dar un ejemplo, se muestran en la Figura 73 las consultas de los OACA más recientes y la de los OACA más descargados aplicando el filtro del campo status.

```
public function BuscarRecientes($limit){

    $em = $this->getEntityManager();
    $dql = "select o from ROABundle:OA o WITH o.status LIKE 'Certificado' ORDER BY o.fecha_publicacion DESC";
    $query = $em->createQuery($dql);
    $query->setMaxResults($limit);
    $OAs = $query->getResult();
    return $OAs;
}

public function BuscarMas_descargados($limit){

    $em = $this->getEntityManager();
    $dql = "select o from ROABundle:OA o WITH o.status LIKE 'Certificado' ORDER BY o.num_descargas DESC";
    $query = $em->createQuery($dql);
    $query->setMaxResults($limit);
    $OAs = $query->getResult();
    return $OAs;
}
```

Figura 73. Consulta de los OA más recientes y más descargados filtrando por por status

Finalmente se le agregó al método edit del controlador OAController la función encargada de informar al usuario sobre el cambio de estado de su OACA a través del envío de un correo electrónico. En la Figura 74 se puede observar dicha función.

```
//Envio de email
$email = \Swift_Message::newInstance()
            ->setSubject('ROACAR')
            ->setFrom('juansneak@gmail.com')
            ->setTo($autor->getEmail())
            ->setBody(
                    $this->renderView(
                        'ROABundle:Email:email_edicion_status_OA.html.twig',
                        array('objeto'=>$entity, 'autor' => $autor)), 'text/html');
$this->get('mailer')->send($email);
```

Figura 74. Función para informar al usuario sobre el cambio de estado de su OA

Pruebas

En esta fase se llevaron a cabo las pruebas para comprobar el correcto funcionamiento de las funciones para crear, editar y eliminar usuarios a través del Módulo de Administración. Por otro lado, se hicieron pruebas relacionadas al cambio de estado de los OACA y su visibilidad en las publicaciones de acuerdo a dicho estado. También se realizaron el mismo tipo de pruebas para los comentarios. Por último, se comprobó que se efectuaran correctamente los envíos de correo electrónico a los usuarios. A continuación se observa en la Tabla 23 los casos de prueba mencionados.

Tabla 23. Casos de prueba - iteración 6

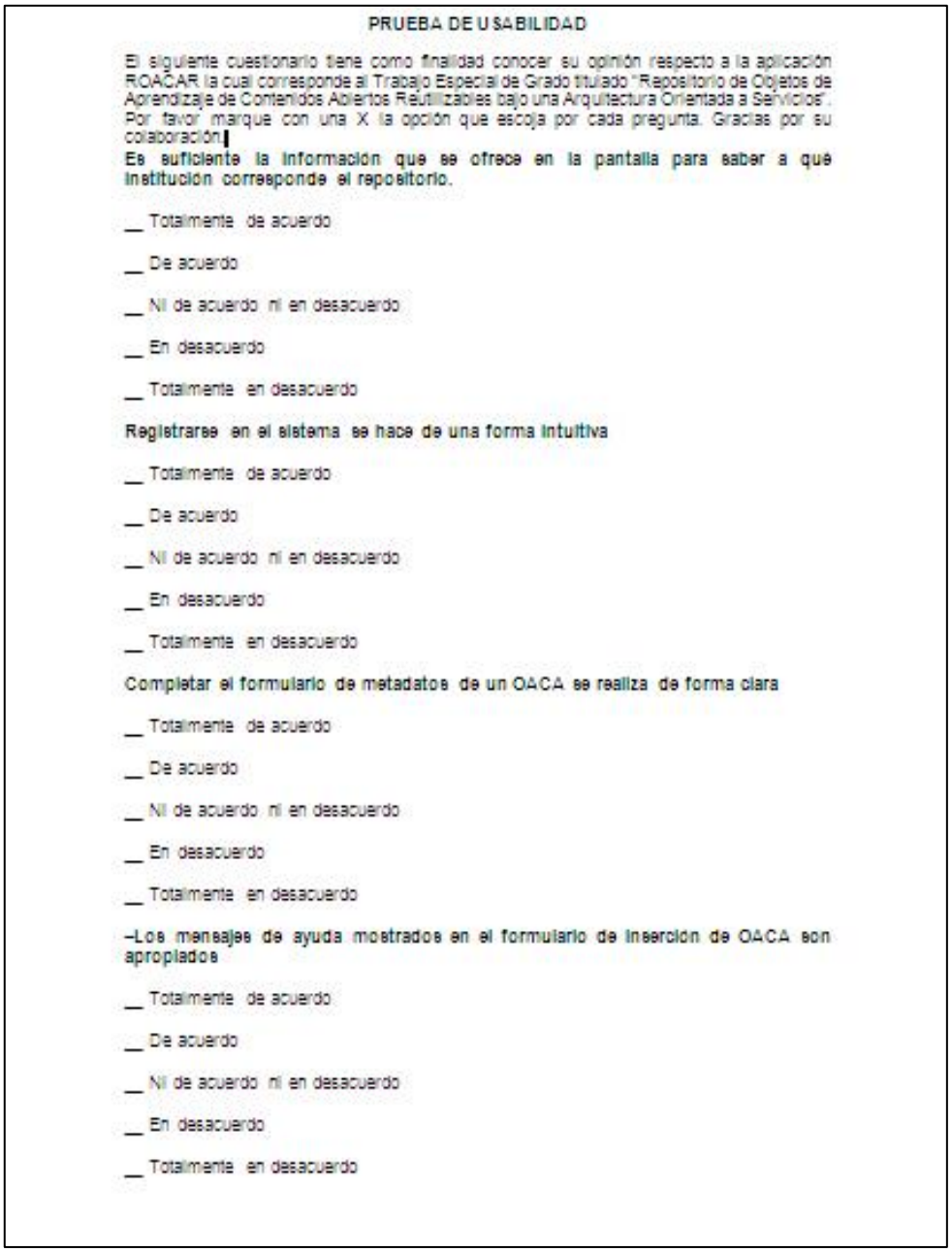
Nº Caso Prueba	Módulo	Caso de Prueba	Resultado Esperado	Resultado Obtenido
1	Administración	Insertar un nuevo Usuario en el Repositorio	Crear un nuevo Usuario.	Se creó correctamente el Usuario.
2	Administración	Editar la información de un usuario existente.	Mostrar todos los datos del usuario y permitir la edición de los mismos.	Se mostraron correctamente los datos del usuario y se lograron editar exitosamente.
3	Administración	Eliminar un usuario registrado.	Eliminar el usuario del Repositorio.	Se eliminó correctamente el usuario del Repositorio.
4	Administración	Listar todos los OACA y cambiar el estado de uno de ellos.	Mostrar todos los OACA y permitir la edición de su estado	Se listaron correctamente todos los OACA y se pudo cambiar el estado de uno de ellos. Además, se envió un correo electrónico al usuario para informarle sobre el cambio de estado de su OACA.
5	Administración	Listar todos los OACA y cambiar la valoración de uno de ellos.	Mostrar todos los OACA y permitir la edición de la valoración.	Se listaron correctamente todos los OACA y se pudo cambiar la valoración de uno de ellos.
6	Administración	Listar todos los comentarios y cambiar el estado de uno de ellos.	Mostrar todos los comentarios y permitir la edición del estado.	Se listaron correctamente todos los comentarios y se pudo cambiar el estado de uno de ellos.

7	Usuario	Consultar los OACA de un usuario y verificar que se muestre el estado y la valoración de los OACA.	Listar todos los OACA del usuario y mostrar el estado y la valoración de cada uno de ellos.	Se listaron correctamente todos los OACA del usuario y se mostró el estado y la valoración de cada uno de ellos.
8	Usuario	Consultar todos los OACA y corroborar que solo se muestren los OACA certificados.	Listar todos los OACA que estén certificados.	Se listaron correctamente todos los OACA certificados.
9	Usuario	Consultar los comentarios de un OACA y corroborar que solo se muestren los comentarios que hayan sido aprobados por el Administrador.	Listar los comentarios que hayan sido aprobados por el Administrador.	Se listaron únicamente los comentarios aprobados por el Administrador.

3. Pruebas de Usabilidad

Para culminar con el desarrollo de ROACAR fueron aplicadas pruebas de usabilidad con el objetivo de corroborar que el Repositorio desarrollado era un producto de software usable, tolerante a fallas y que cumplía con el funcionamiento esperado para finalmente ponerlo en producción. Las pruebas de usabilidad consistieron en un cuestionario donde se empleó la escala de Likert, en ésta se plantean enunciados positivos y negativos, ante los cuales el individuo debe mostrar su acuerdo o desacuerdo, se utilizaron cinco alternativas de respuestas para cada enunciado: totalmente de acuerdo, de acuerdo, ni de acuerdo ni en desacuerdo y totalmente de acuerdo (Fernadez Nogales, 2004). El cuestionario se realizó de forma escrita y fue aplicado a 8 estudiantes universitarios.

En la Figura 75 se muestra la primera parte del modelo de cuestionario utilizado.



PRUEBA DE USABILIDAD

El siguiente cuestionario tiene como finalidad conocer su opinión respecto a la aplicación ROACAR la cual corresponde al Trabajo Especial de Grado titulado "Repositorio de Objetos de Aprendizaje de Contenidos Abiertos Reutilizables bajo una Arquitectura Orientada a Servicios". Por favor marque con una X la opción que escoja por cada pregunta. Gracias por su colaboración.

Es suficiente la información que se ofrece en la pantalla para saber a qué institución corresponde el repositorio.

__ Totalmente de acuerdo

__ De acuerdo

__ Ni de acuerdo ni en desacuerdo

__ En desacuerdo

__ Totalmente en desacuerdo

Registrarse en el sistema se hace de una forma intuitiva

__ Totalmente de acuerdo

__ De acuerdo

__ Ni de acuerdo ni en desacuerdo

__ En desacuerdo

__ Totalmente en desacuerdo

Completar el formulario de metadatos de un OACA se realiza de forma clara

__ Totalmente de acuerdo

__ De acuerdo

__ Ni de acuerdo ni en desacuerdo

__ En desacuerdo

__ Totalmente en desacuerdo

–Los mensajes de ayuda mostrados en el formulario de inserción de OACA son apropiados

__ Totalmente de acuerdo

__ De acuerdo

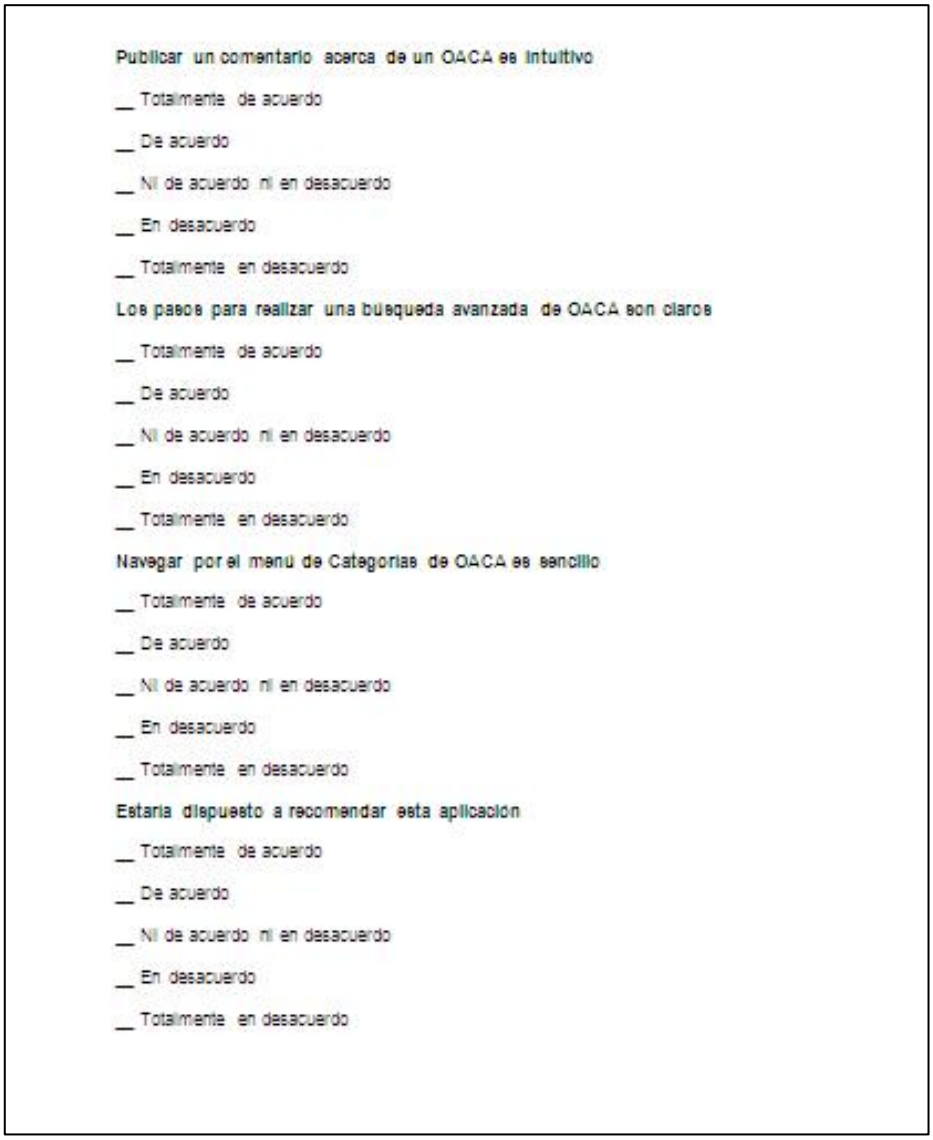
__ Ni de acuerdo ni en desacuerdo

__ En desacuerdo

__ Totalmente en desacuerdo

Figura 75. Primera parte del modelo de cuestionario

En la Figura 76 se muestra la segunda parte del modelo del cuestionario utilizado



Publicar un comentario acerca de un OACA es intuitivo

__ Totalmente de acuerdo

__ De acuerdo

__ Ni de acuerdo ni en desacuerdo

__ En desacuerdo

__ Totalmente en desacuerdo

Los pasos para realizar una búsqueda avanzada de OACA son claros

__ Totalmente de acuerdo

__ De acuerdo

__ Ni de acuerdo ni en desacuerdo

__ En desacuerdo

__ Totalmente en desacuerdo

Navegar por el menú de Categorías de OACA es sencillo

__ Totalmente de acuerdo

__ De acuerdo

__ Ni de acuerdo ni en desacuerdo

__ En desacuerdo

__ Totalmente en desacuerdo

Estaría dispuesto a recomendar esta aplicación

__ Totalmente de acuerdo

__ De acuerdo

__ Ni de acuerdo ni en desacuerdo

__ En desacuerdo

__ Totalmente en desacuerdo

Figura 76. Segunda parte del modelo de cuestionario

3.1 Resultados de la Prueba de Usabilidad

Del cuestionario presentado anteriormente, se pudieron obtener los siguientes resultados:

En la primera pregunta, que se puede apreciar en el Gráfico 1, el 75% de los usuarios estuvo totalmente de acuerdo con que era suficiente la información que se ofrece en la pantalla para saber a qué institución corresponde el Repositorio, y el 25% restante indicó que estaba de acuerdo.

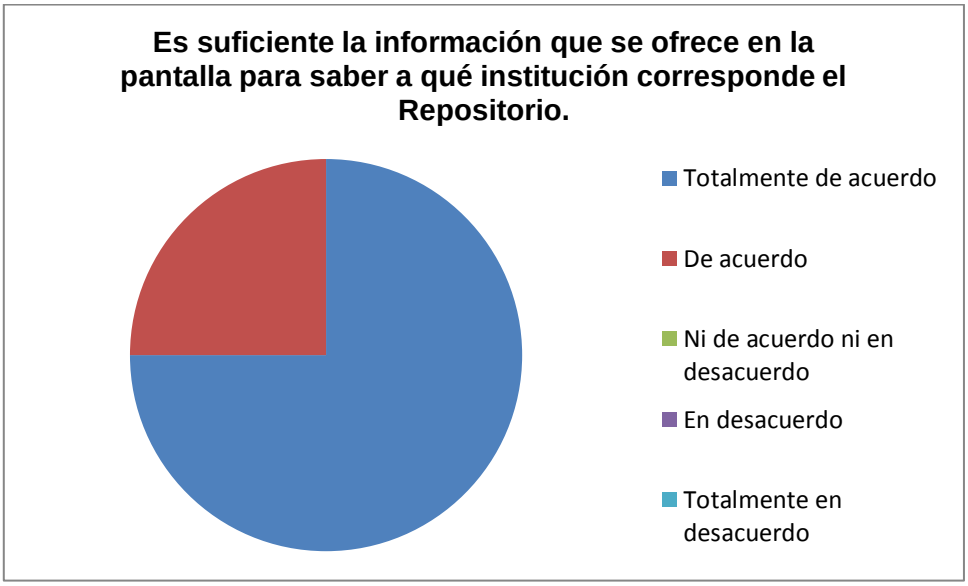


Gráfico 1. Gráfico de torta: Es suficiente la información que se ofrece en la pantalla para saber a qué institución corresponde el Repositorio

En la segunda pregunta, como se refleja en el Gráfico 2, el 25% de los usuarios opinó que estaba totalmente de acuerdo con que registrarse en el Repositorio se hace de una forma intuitiva, mientras que el 75% restante indicó que estaba de acuerdo.

Gráfico 2. Gráfico de torta: Registrarse en el Repositorio se hace de una forma intuitiva

En la tercera pregunta, el 25% de los usuarios estuvo totalmente de acuerdo con que completar el formulario de metadatos de un OACA se realiza de forma clara, y el 75% restante estuvo de acuerdo. Estos resultados se ven reflejados en el Gráfico 3.

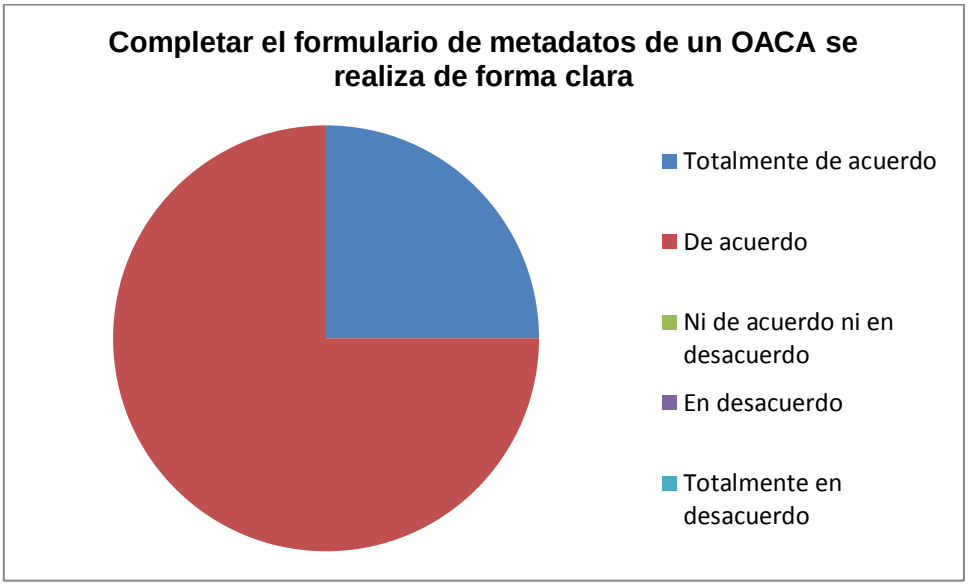


Gráfico 3. Gráfico de torta: Completar el formulario de metadatos de un OACA se realiza de forma clara

En la cuarta pregunta, como se refleja en el Gráfico 4, el 75% de los usuarios indicó que estaba completamente de acuerdo con que los mensajes de ayuda mostrados en el formulario de inserción de OACA son apropiados, mientras que el 25% restante opinó que estaba de acuerdo.

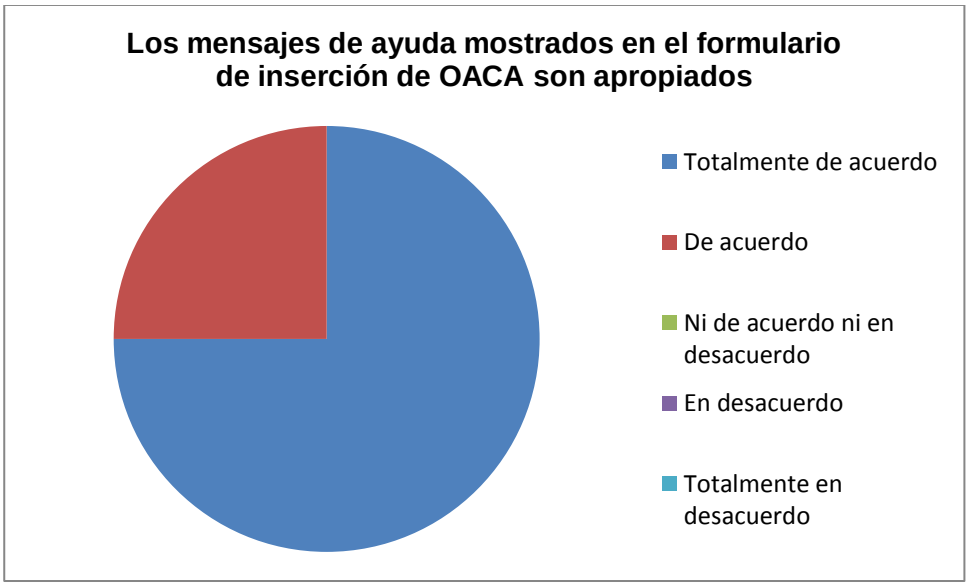


Gráfico 4. Gráfico de torta: Los mensajes de ayuda mostrados en el formulario de inserción de OACA son apropiados

En la quinta pregunta, como se refleja en el Gráfico 5, un 25% de los usuarios indicó que estaba completamente de acuerdo con que publicar un comentario es intuitivo, otro 25% indicó que estaba de acuerdo y el 50% restante señaló que no estaba ni de acuerdo ni en desacuerdo.

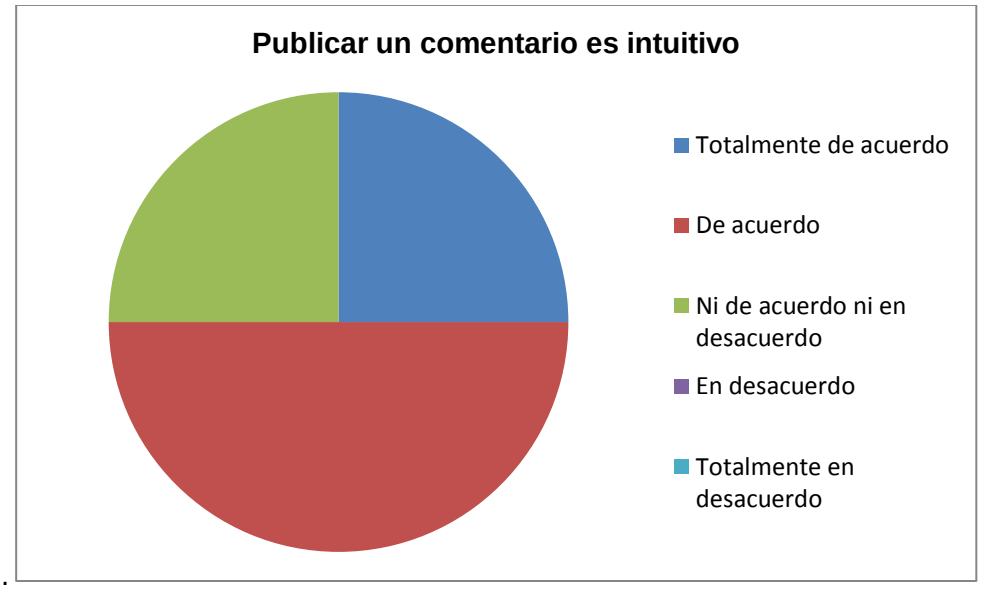


Gráfico 5. Gráfico de torta: Publicar un comentario es intuitivo

En la sexta pregunta, el 75% de los usuarios indicó que estaba completamente de acuerdo con que los pasos para realizar una búsqueda avanzada de OACA son claros, mientras que el 25% restante estuvo de acuerdo. Estos resultados se reflejan en el Gráfico 6.

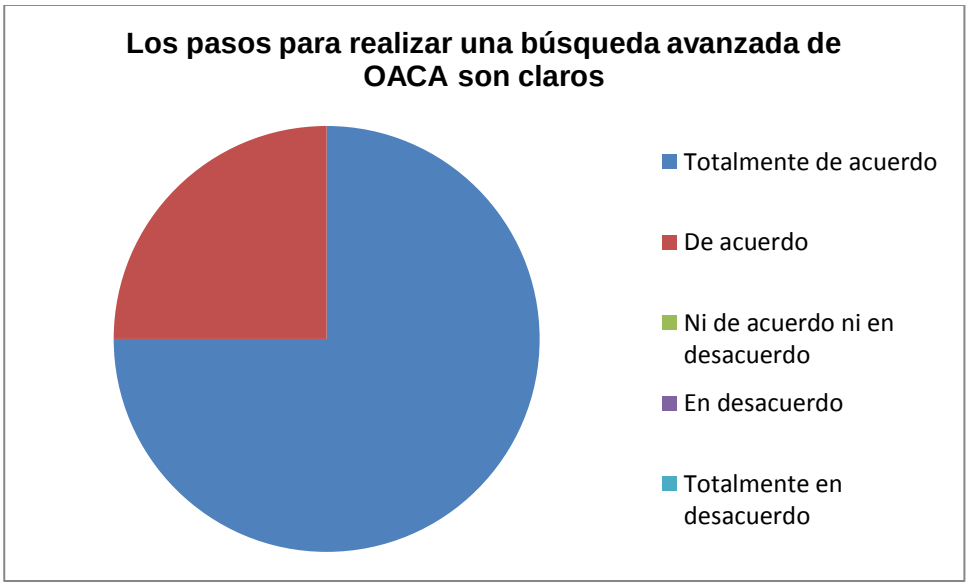


Gráfico 6. Gráfico de torta: Los pasos para realizar una búsqueda avanzada de OACA son claros

En la séptima pregunta, como se refleja en el Gráfico 7, el 50% de los usuarios estuvo completamente de acuerdo con que navegar por el menú de Categorías de OACA es sencillo, mientras que un 25% indicó que estaba de acuerdo y otro 25% señaló que no estaba ni de acuerdo ni en desacuerdo.

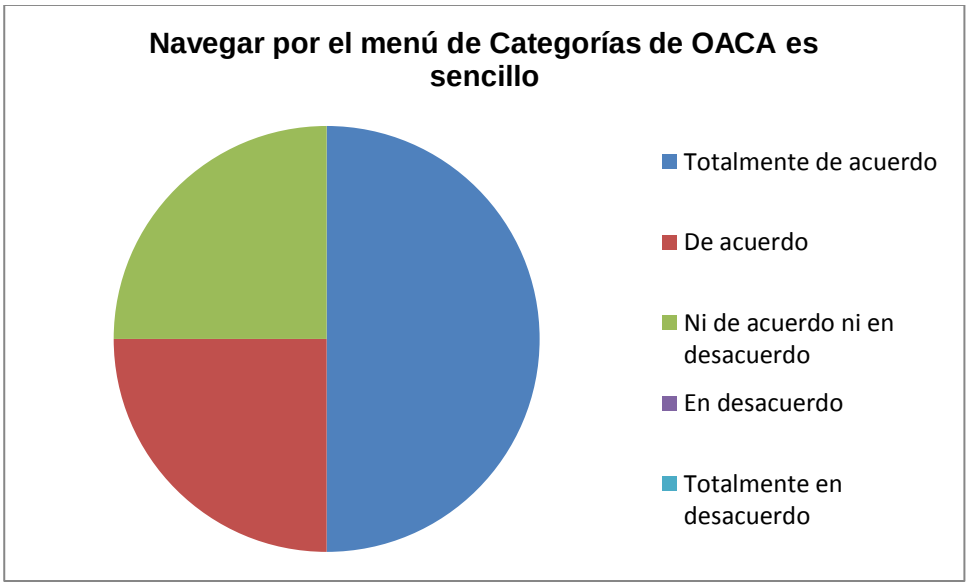


Gráfico 7. Gráfico de torta: Navegar por el menú de Categorías de OACA es sencillo

En la octava y última pregunta, como se refleja en el Gráfico 8, el 75% de los usuarios estuvo completamente de acuerdo con que estaría dispuesto a recomendar el Repositorio, mientras que el 25% restante indicó que estaría de acuerdo.

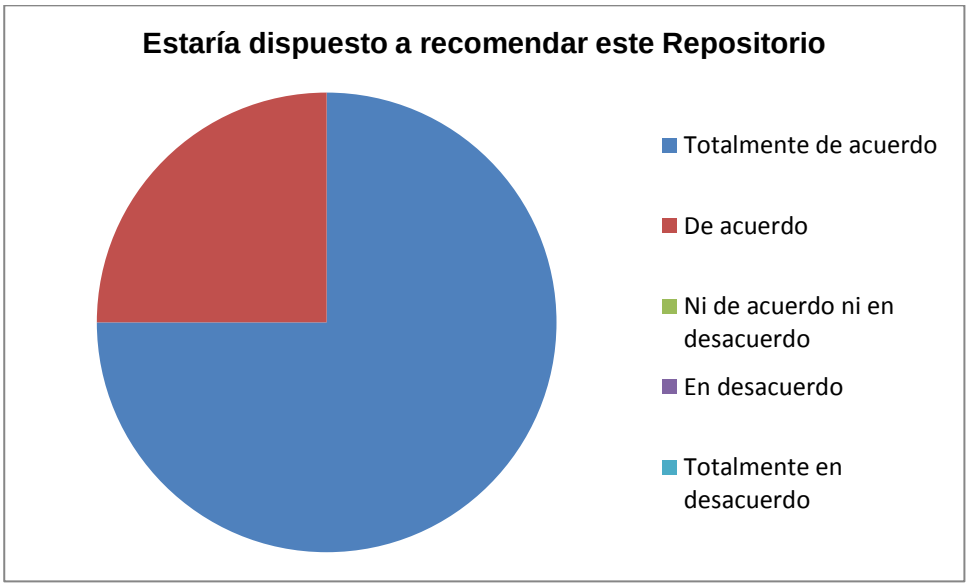


Gráfico 8. Gráfico de torta: Estaría dispuesto a recomendar este Repositorio

Es importante tomar en cuenta que todas las preguntas del cuestionario fueron formuladas en base a los criterios de usabilidad de Nielsen (1994), y en base a los resultados obtenidos que están reflejados en los gráficos; se puede concluir que ROACAR es usable, ya que se obtuvo un número significativo de respuestas favorables y de aceptación del Repositorio.

Capítulo IV. Resultados

En este capítulo son descritos los resultados en función de la interfaz así como las funcionalidades de ROACAR luego de emplear la metodología XP. El Repositorio cuenta con dos módulos: Módulo Usuario y Módulo de Administración. El Módulo de Usuario cuenta con una página de inicio, una sección para gestionar los OACA de cada usuario, una sección de búsqueda y los créditos. Dentro del Módulo de Administración se encuentran las funcionalidades para gestionar los usuarios, los comentarios, los OACA y las categorías de los OACA.

1. Módulo Usuario

El Módulo Usuario corresponde al Portal Web que puede ser accedido por cualquier tipo de usuario. A continuación se describen cada una las secciones de este Módulo.

1.1 Página de Inicio

La Figura 77 muestra la página de inicio del Módulo Usuario, ésta cuenta con una breve descripción acerca del ROACA y sus funcionalidades. Contiene dos menús; un menú lateral con las categorías y subcategorías a través de las cuales se pueden ubicar los OACA publicados y un menú superior con las principales funcionalidades del Repositorio: Gestionar OACA, Buscar OACA, Créditos. Adicionalmente, se incluye el formulario de inicio de sesión, los botones de accesibilidad debajo del menú principal y las licencias Creative Commons en la parte inferior de la página.

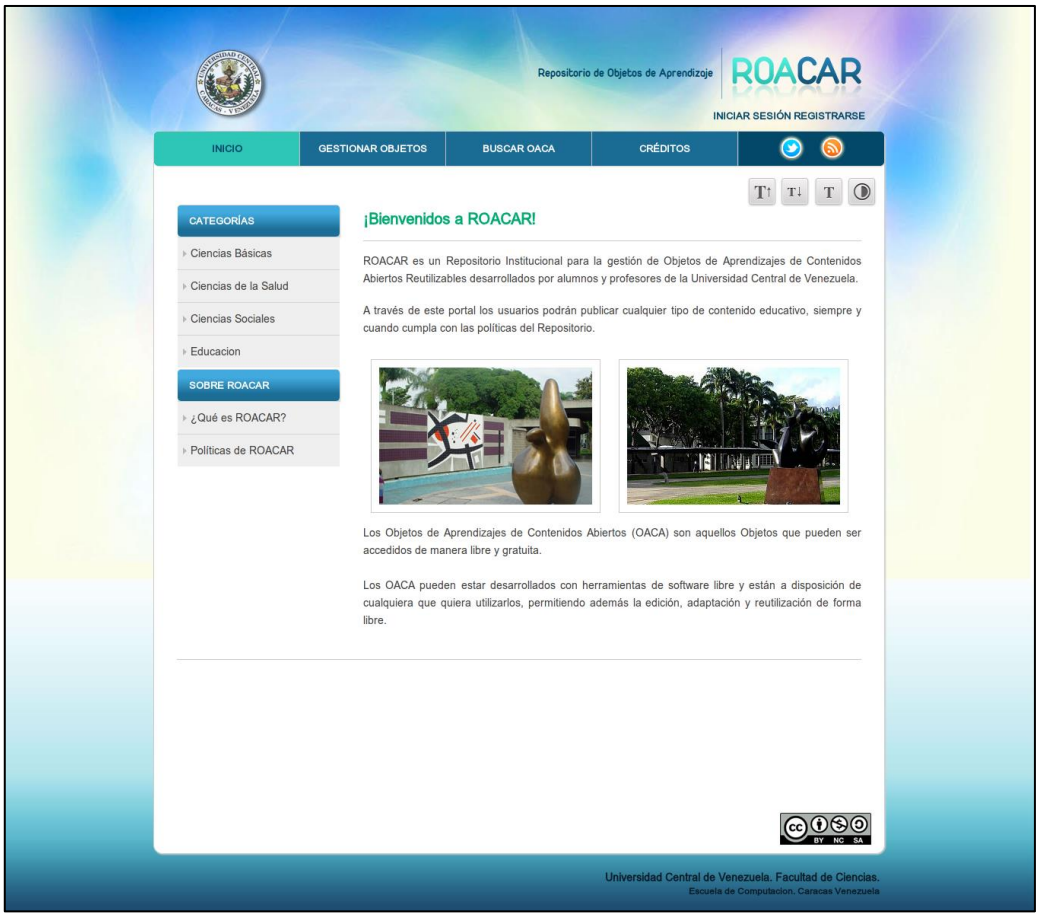


Figura 77. Vista de la página de Inicio del Módulo Usuario

A continuación se explican detalladamente cada uno de los elementos de la página de inicio.

1.1.1 Menú principal

El menú principal contiene las siguientes opciones:

a) **Gestionar OACA:** a través de esta opción los usuarios podrán tanto insertar OACA nuevos como editar sus OACA anteriores.
b) **Buscar OACA:** esta opción permite buscar los OACA mas descargados, los OACA más recientes y realizar búsquedas avanzadas.
c) **Créditos:** contiene los créditos del Repositorio.

1.1.2 Menú lateral

El menú lateral permite realizar una navegación rápida a través de las diferentes categorías y subcategorías mediante las cuales se clasifican los OACA publicados en el Repositorio. Adicionalmente, este menú incluye un enlace hacia la página de las políticas de ROACAR.

1.1.3 Formulario de inicio de Sesión

El formulario de inicio de sesión se despliega al hacer click en el botón “Iniciar Sesión” ubicado debajo del logo de ROACAR. Dicho formulario se puede observar en la Figura 78.

Figura 78. Vista del formulario de inicio de sesión

1.1.4 Botones de Accesibilidad

Los botones de accesibilidad están ubicados debajo del menú principal y del lado derecho de la página. Estos botones permiten aumentar y disminuir el tamaño de las fuentes y cambiar el contraste de los colores.

1.1.5 Licencias Creative Commons

Las licencias Creative Commons se encuentran en la parte inferior derecha de la página. Dichas licencias son:

a) **Atribución:** El beneficiario de la licencia tiene el derecho de copiar, distribuir, exhibir y representar la obra y hacer obras derivadas siempre y cuando reconozca y cite la obra de la forma especificada por el autor o el licenciante.

b) No Comercial: El beneficiario de la licencia tiene el derecho de copiar, distribuir, exhibir y representar la obra y hacer obras derivadas para fines no comerciales.

c) Compartir igual: El beneficiario de la licencia tiene el derecho de distribuir obras derivadas bajo una licencia idéntica a la licencia que regula la obra original.

1.2 Registro de Usuarios

La Figura 79 muestra la página de registro de usuarios, esta cuenta con un formulario que contiene los siguientes campos: nombre, apellido, email, tipo de usuario (estudiante o profesor), contraseña y foto de perfil. Es importante destacar, que todos los campos son de carácter obligatorio a excepción de la foto de perfil.

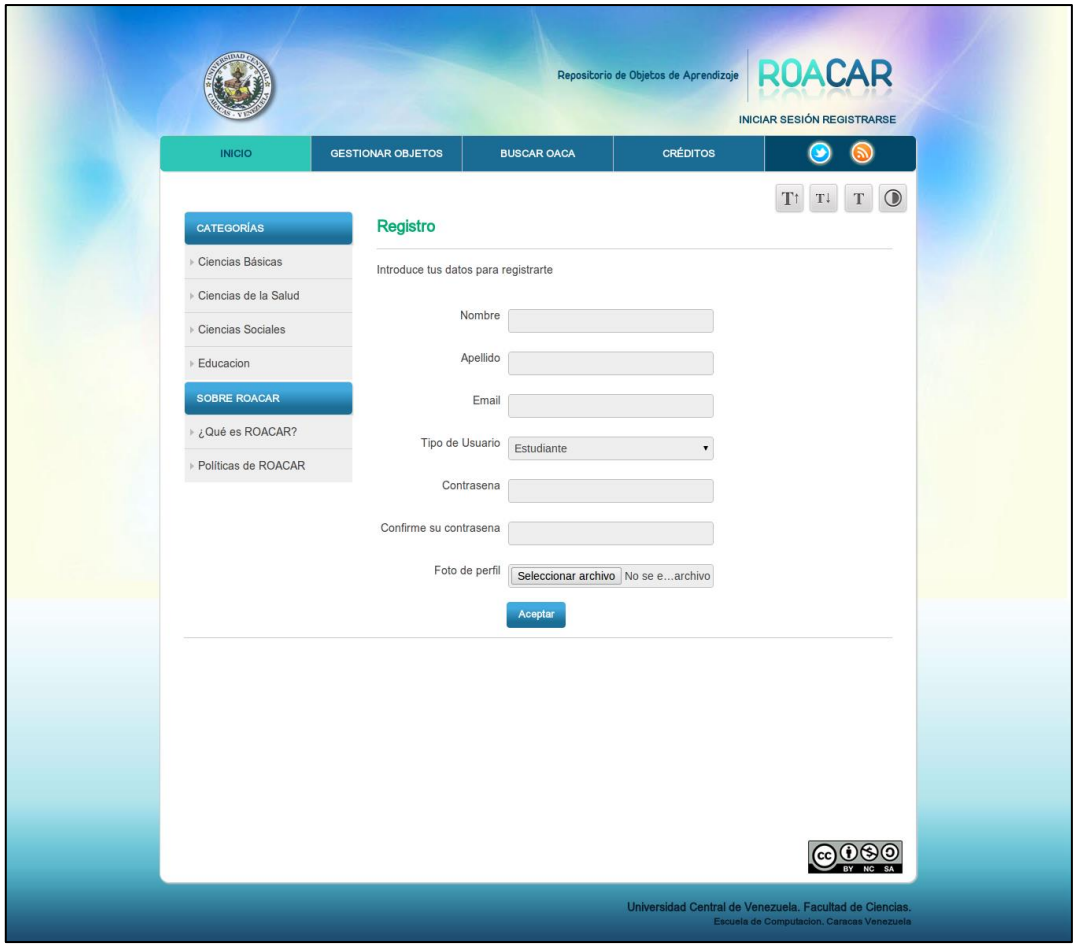


Figura 79. Vista del registro de usuario

1.3 Gestionar OACA

La opción de Gestionar OACA se divide en dos secciones: Nuevo OACA y Mis OACA. A continuación se explican ambas secciones.

1.3.1 Nuevo OACA

En la Figura 80 se muestra la sección de Nuevo OACA. Para acceder a esta sección los usuarios deben haberse registrado previamente en el Repositorio. Aquí se muestra un formulario que contiene todas las categorías correspondientes al estándar LOM divididas en varias pestañas. El usuario debe llenar al menos todos los datos obligatorios del formulario para poder insertar un nuevo OACA en el Repositorio. En la parte superior del formulario aparece también un enlace a las políticas de ROACAR.

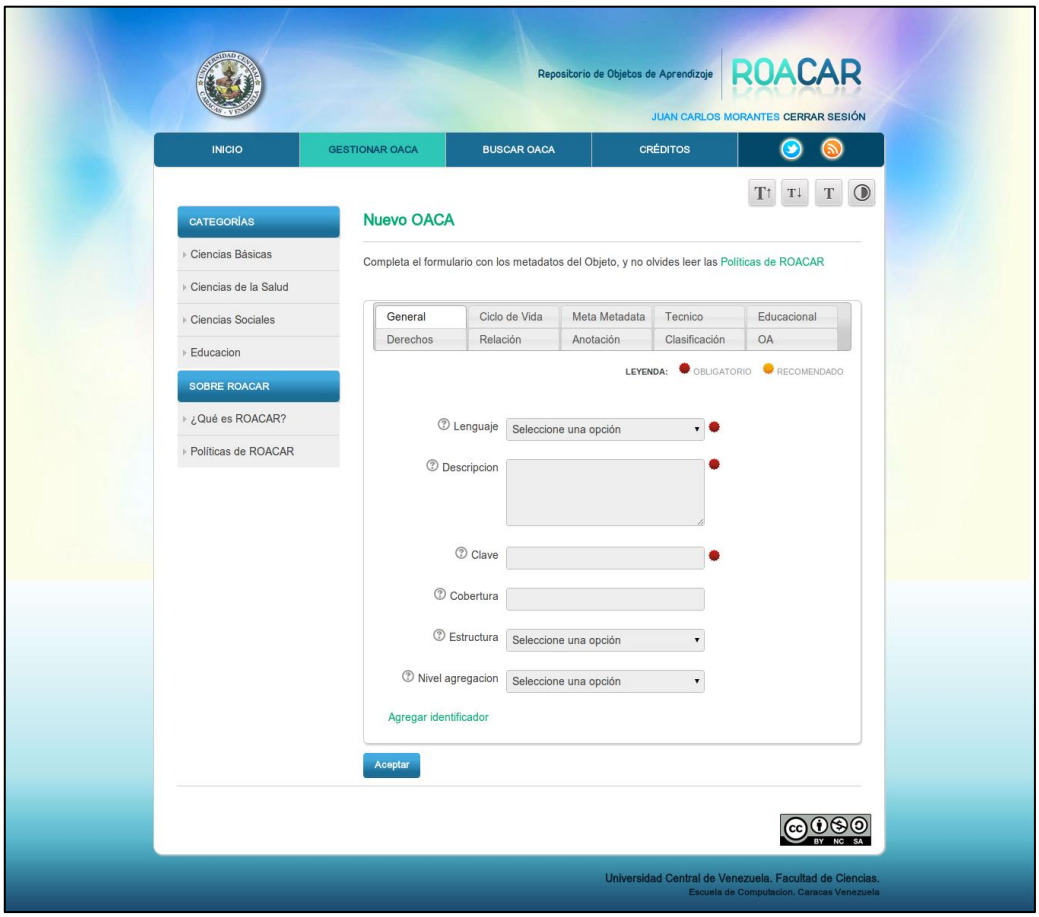


Figura 80. Vista de la sección Nuevo OACA

1.3.2 Mis OACA

La Figura 81 muestra la sección Mis OACA. Esta sección contiene una lista paginada de los OACA pertenecientes al usuario conectado. Por cada OACA se muestran los siguientes datos: Titulo, Autor, Categoría, Subcategoría y Fecha de publicación. Adicionalmente se muestra la valoración del OACA y un logo del lado derecho que indica el estado del OACA (No Autorizado, En Revisión o Certificado).

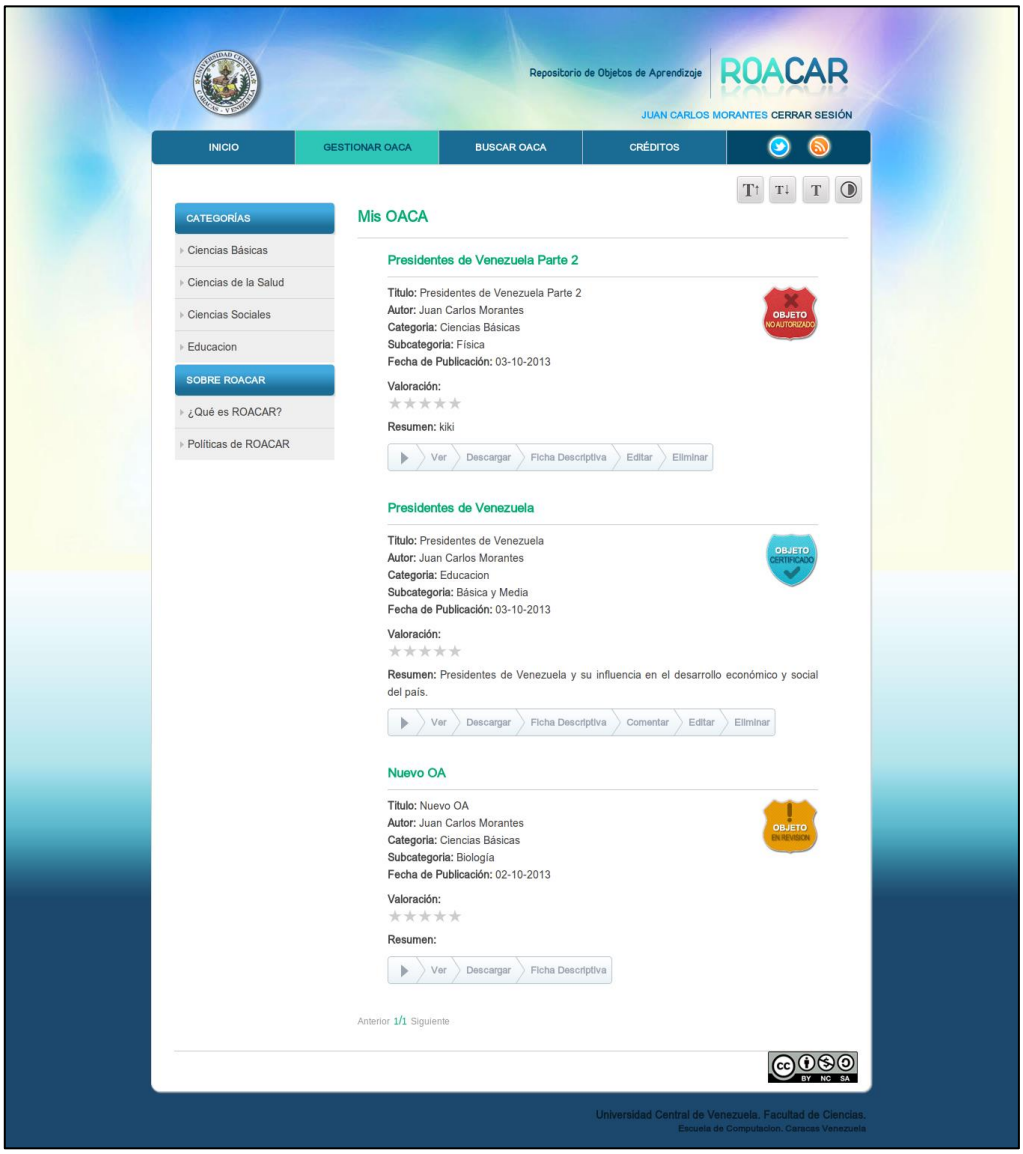


Figura 81. Vista de la sección Mis OACA

En la Figura 82 también se puede observar que cada OACA contiene un conjunto de botones que permiten realizar las siguientes acciones:

a) **Ver:** permite visualizar el OACA directamente en el navegador
b) **Descargar:** permite descargar un paquete comprimido con el OACA
c) **Ficha descriptiva:** despliega una ventana emergente con los metadatos del OACA. La ficha descriptiva se muestra en la Figura 80.
d) **Editar:** permite editar los metadatos del OACA seleccionado. En la Figura 83 se muestra el formulario de edición de los OACA.
e) **Eliminar:** permite borrar el OACA y todos sus metadatos.
f) **Comentar:** permite publicar comentarios sobre el OACA seleccionado. Esta opción solo está disponible para aquellos OACA que estén certificados.

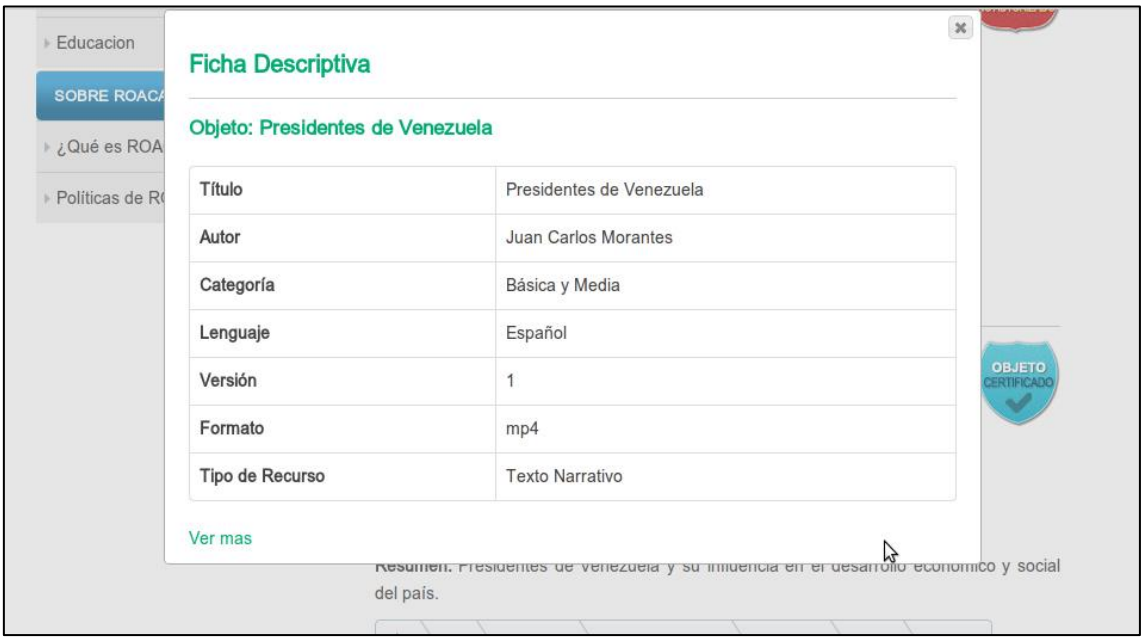


Figura 82. Vista de la ficha descriptiva de un OACA

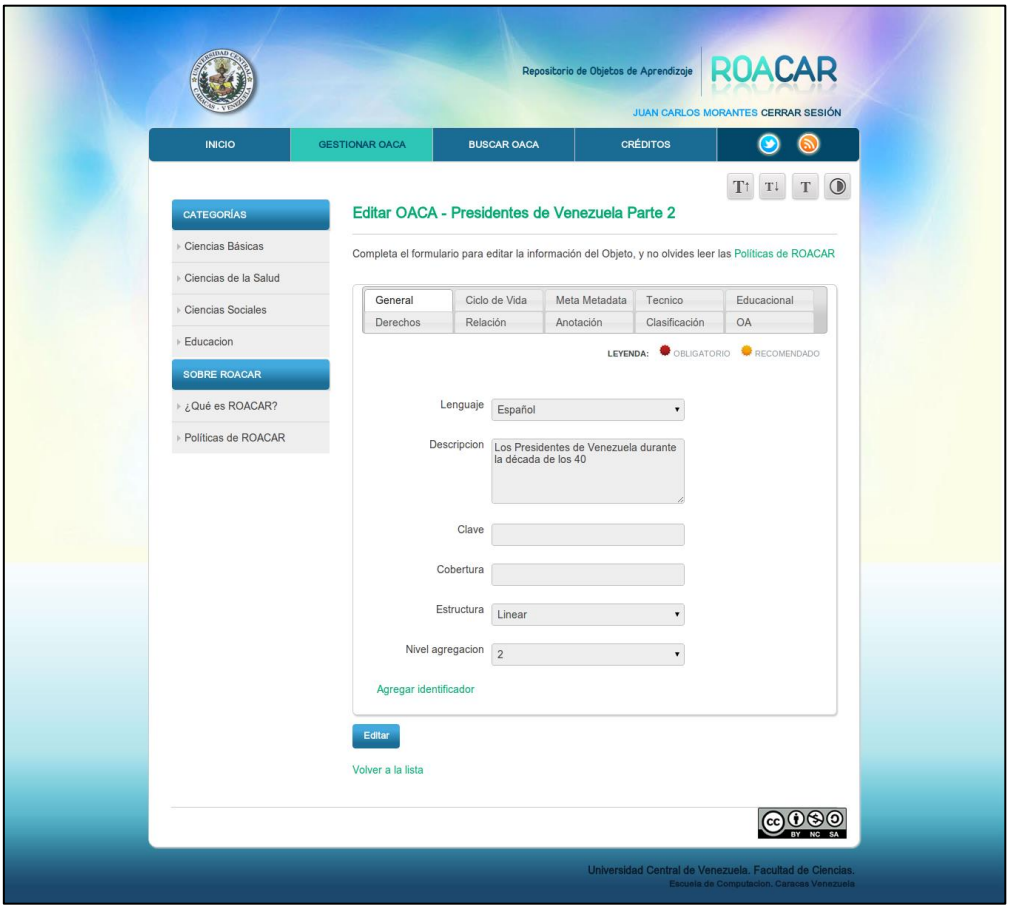


Figura 83. Vista del formulario de edición de OACA

1.4 Buscar OACA

La opción de Buscar OACA se divide en tres secciones: Búsqueda Avanzada, Lo Mas Reciente y Lo Mas Descargado. A continuación se explican dichas secciones.

1.4.1 Búsqueda Avanzada

La Figura 84 muestra la sección de Búsqueda Avanzada. Esta sección permite buscar cualquier OACA dentro del Repositorio filtrando por Categoría, Autor, Año y Título. Los filtros se aplican a través de un sencillo formulario que contiene los campos mencionados anteriormente. El resultado de la búsqueda se muestra a través de una lista paginada igual a la de la sección Mis OACA. Dicha lista se puede observar en la Figura 85.

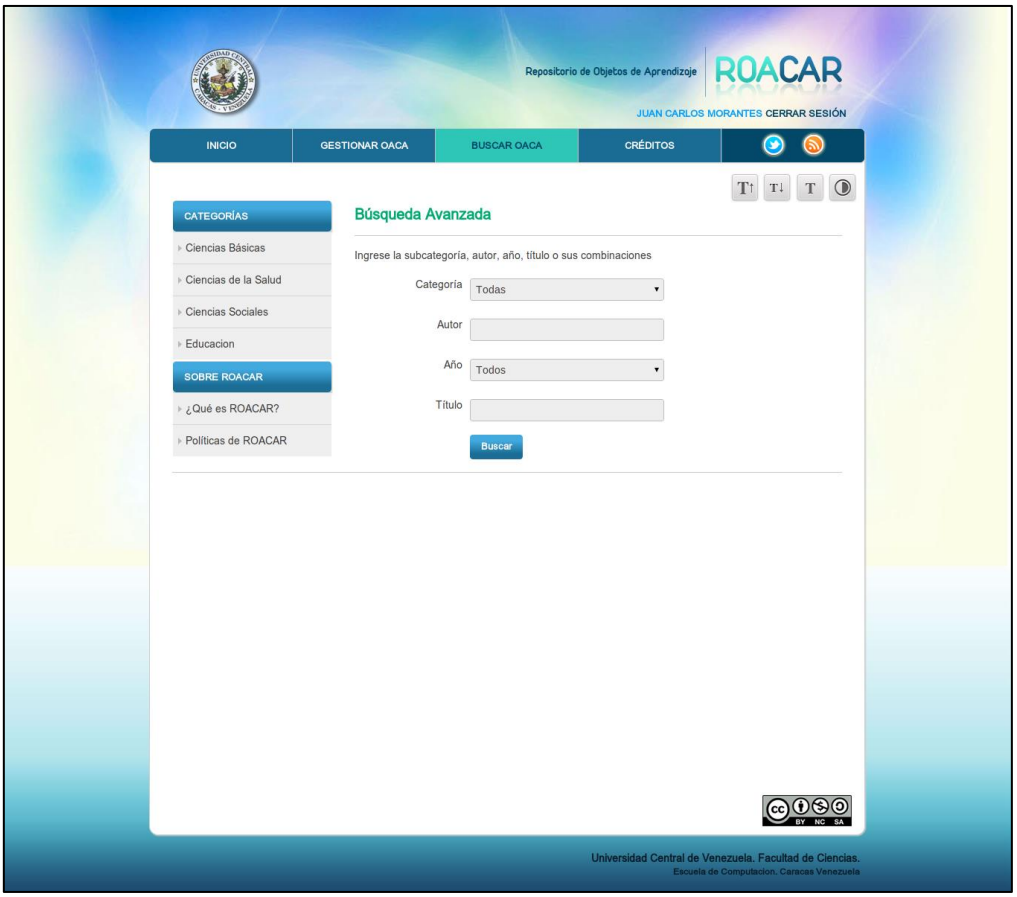


Figura 84. Vista de la sección Búsqueda Avanzada

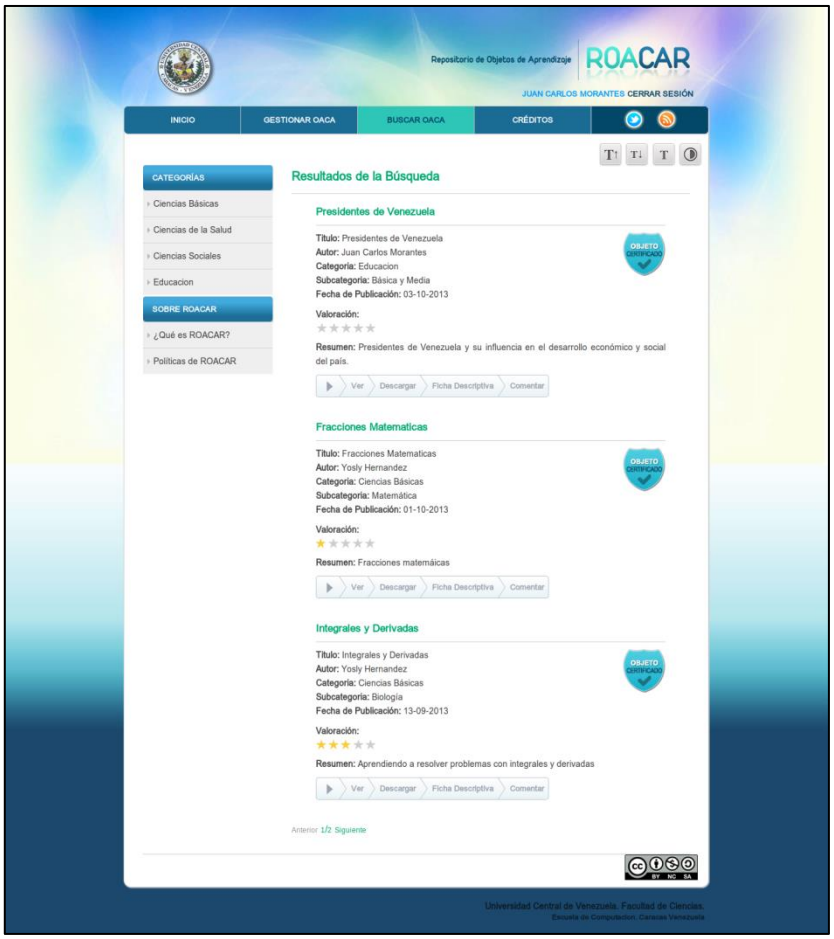


Figura 85. Vista de resultados de la Búsqueda Avanzada

1.4.2 Lo Más Reciente

La Figura 86 muestra la sección de Lo Mas Reciente. En esta sección se listan los últimos tres OACA que fueron publicados en el Repositorio.

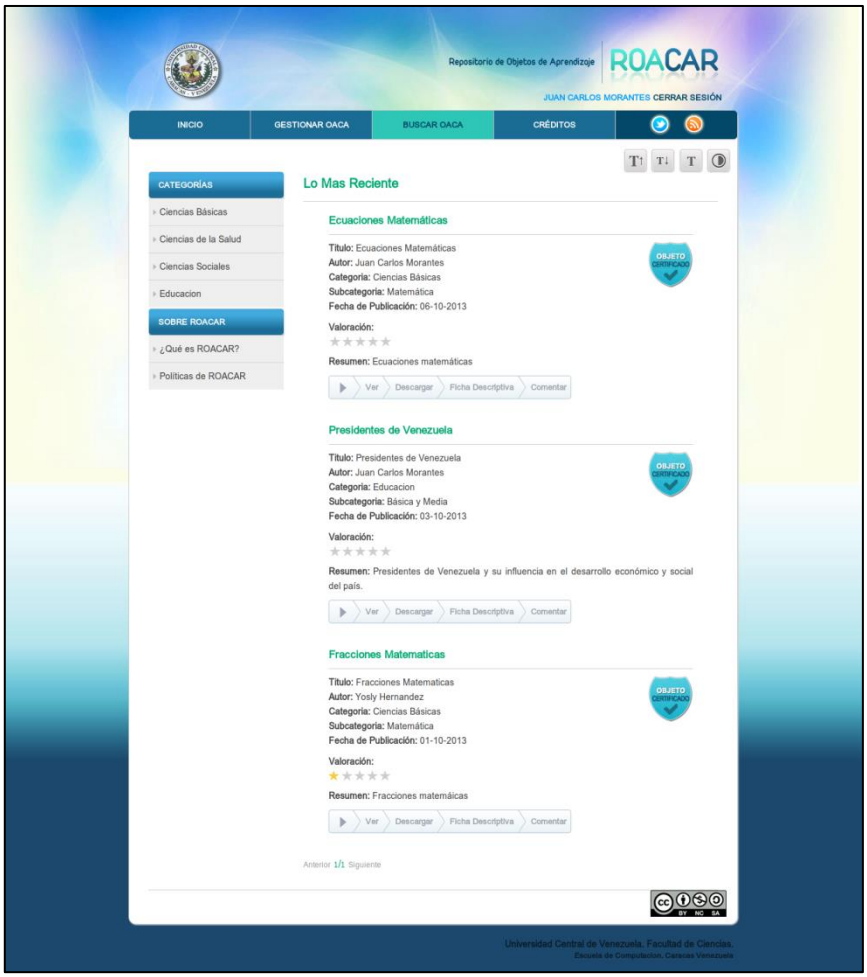


Figura 86. Vista de la sección Lo Mas Reciente

1.4.3 Lo Más Descargado

La Figura 87 muestra la sección de Lo Mas Descargado. En esta sección se listan los tres OACA que más han sido descargados por los diferentes usuarios del Repositorio.

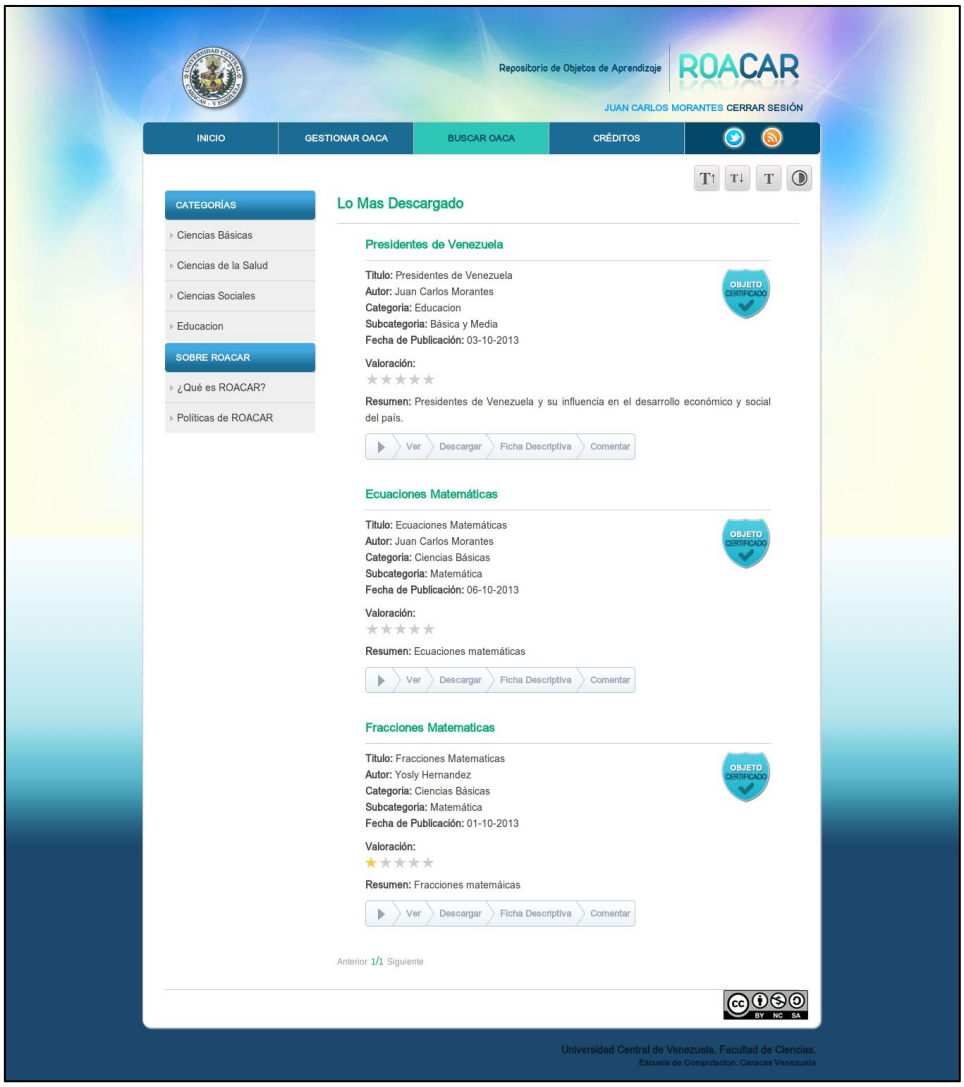


Figura 87. Vista de la sección Lo Mas Descargado

1.5 Créditos

En la Figura 88 se pueden apreciar los créditos del Repositorio. Los créditos contienen los logos de la Universidad Central de Venezuela, la Facultad de Ciencias y la Escuela de Computación. También se muestra el nombre completo y correo electrónico tanto del desarrollador como del tutor del Trabajo Especial de Grado.

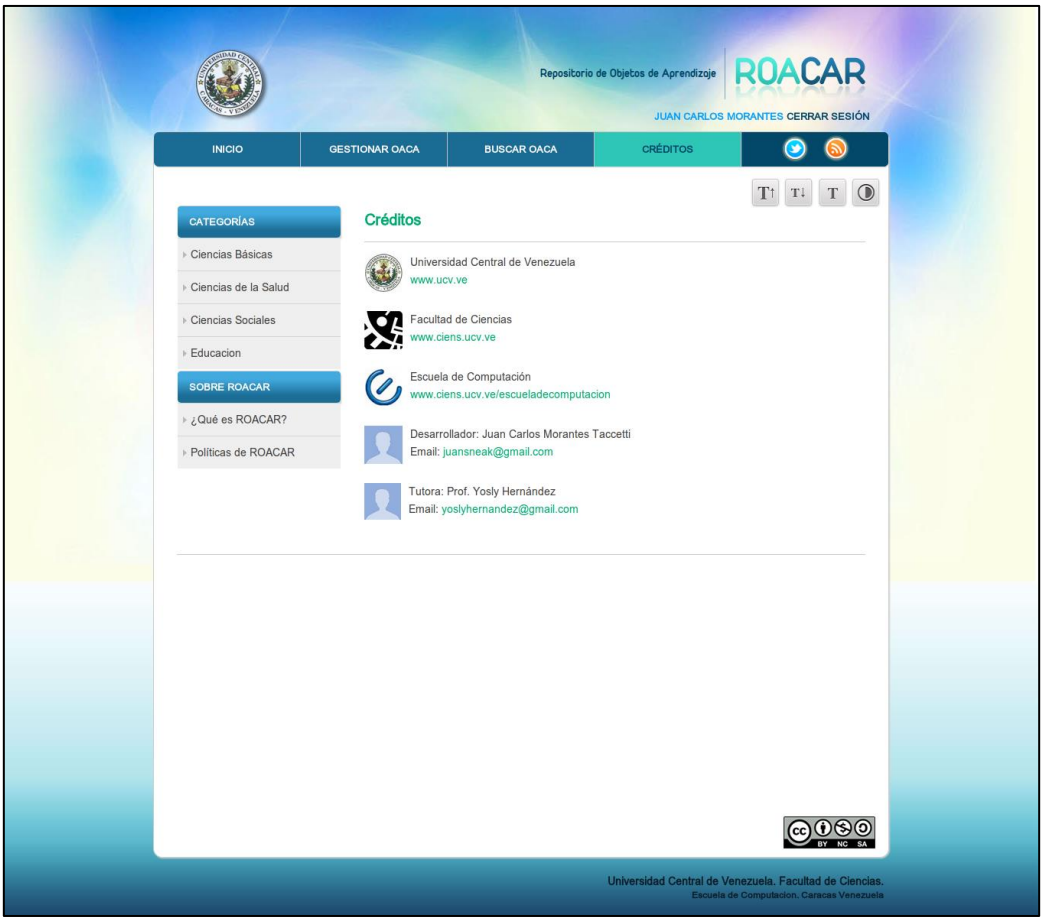


Figura 88. Vista de los créditos del Repositorio

2. Módulo de Administración

A través del Módulo de Administración el usuario Administrador podrá gestionar los OACA, los comentarios y los usuarios del Repositorio. A continuación se describen cada una de las secciones de este Módulo.

2.1 Página de Inicio

En la Figura 89 se puede observar la página de inicio del Módulo de Administración. Esta página contiene un menú horizontal con las principales funcionalidades de este Módulo: Usuarios, Contenidos y Solicitudes.

Figura 89. Vista de la página de inicio del Módulo de Administración

2.2 Usuarios

Esta sección permite gestionar todos los usuarios del Repositorio. Las funciones permitidas son las siguientes:

a) **Crear Nuevo Usuario:** permite crear un nuevo usuario. En la Figura 90 se muestra el formulario para insertar un nuevo usuario.
b) **Ver:** permite ver toda la información del usuario.
c) **Editar:** permite editar la información de un usuario.
d) **Eliminar:** permite borrar un usuario del Repositorio.

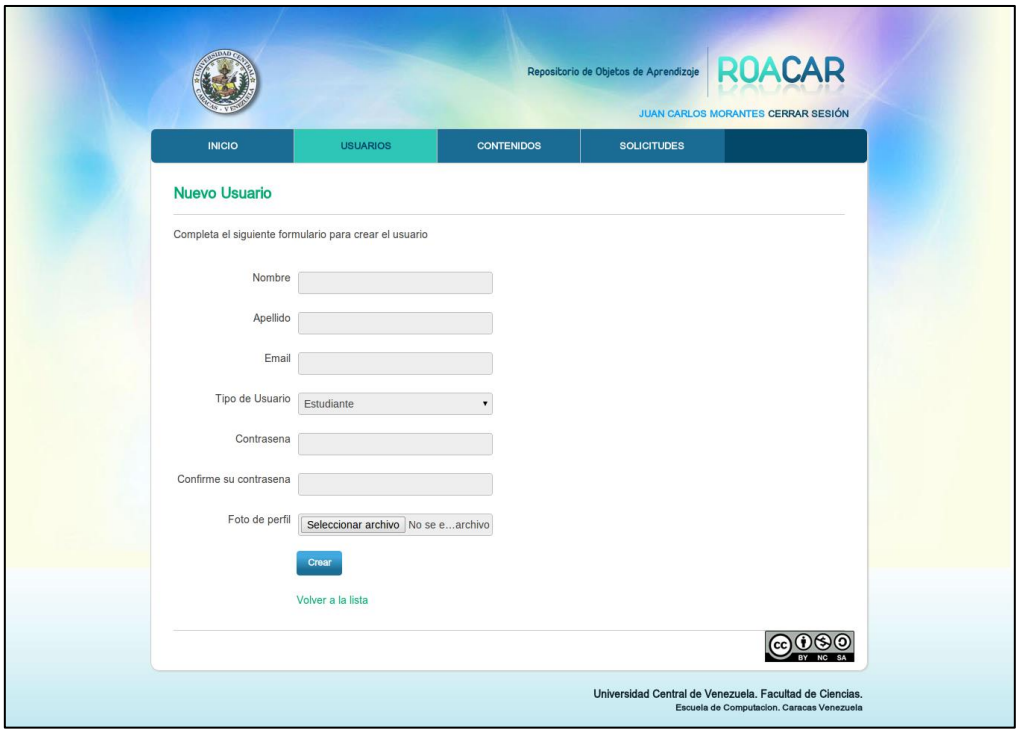


Figura 90. Vista del formulario para insertar un nuevo usuario (Módulo de Administración)

2.3 Contenidos

El objetivo de esta sección es tener un Manejador de Contenidos que permita gestionar los diferentes elementos que aparecen en el Módulo de Usuario, como por ejemplo, el mensaje de bienvenida, las categorías de los OACA, el texto de la página principal, las políticas del Repositorio, entre otros. Sin embargo, este Trabajo Especial de Grado solo abarcó la sección de Categorías, a través de la cual se gestionan las Categorías y Subcategorías de los OACA. Las funciones permitidas para esta sección son las siguientes:

e) **Crear Nueva Categoría:** permite crear una nueva categoría. Al momento de crear una categoría se pueden crear también la cantidad de subcategorías que se desee a través de un formulario anidado. En la Figura 91 se muestra el formulario para crear una nueva categoría.
f) **Ver:** permite ver el nombre de una categoría y el conjunto de subcategorías asociadas.
g) **Editar:** permite editar la información de una categoría. Al momento de editar una categoría se pueden editar o eliminar las subcategorías asociadas. **Eliminar:** permite borrar una categoría. Al eliminar una categoría se borran también todas las subcategorías pertenecientes a la categoría padre.

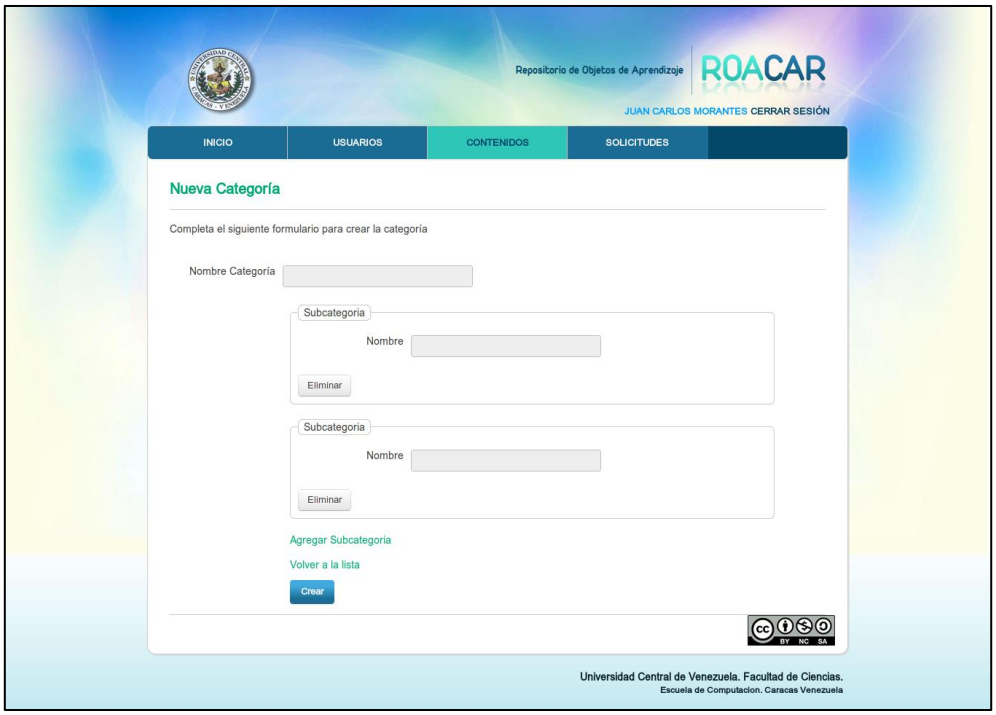


Figura 91. Vista del formulario para crear una nueva Categoría

2.4 Solicitudes

A través de esta sección el usuario Administrador puede gestionar las solicitudes que hagan los usuarios para publicar OACA y comentarios en el Repositorio. Esta sección se divide en: Comentarios y OACA.

2.4.1 Comentarios

Esta opción permite listar todos los comentarios que han hecho los usuarios. En esta lista se muestra el nombre del usuario que ha hecho el comentario, y el nombre del OACA que ha sido comentado. Adicionalmente, se muestra la opción de editar, a través de la cual el usuario Administrador puede leer el contenido completo del comentario y permitir o no su publicación. En las figuras 92 y 93 se puede apreciar tanto la lista de comentarios como el formulario de edición de los mismos.

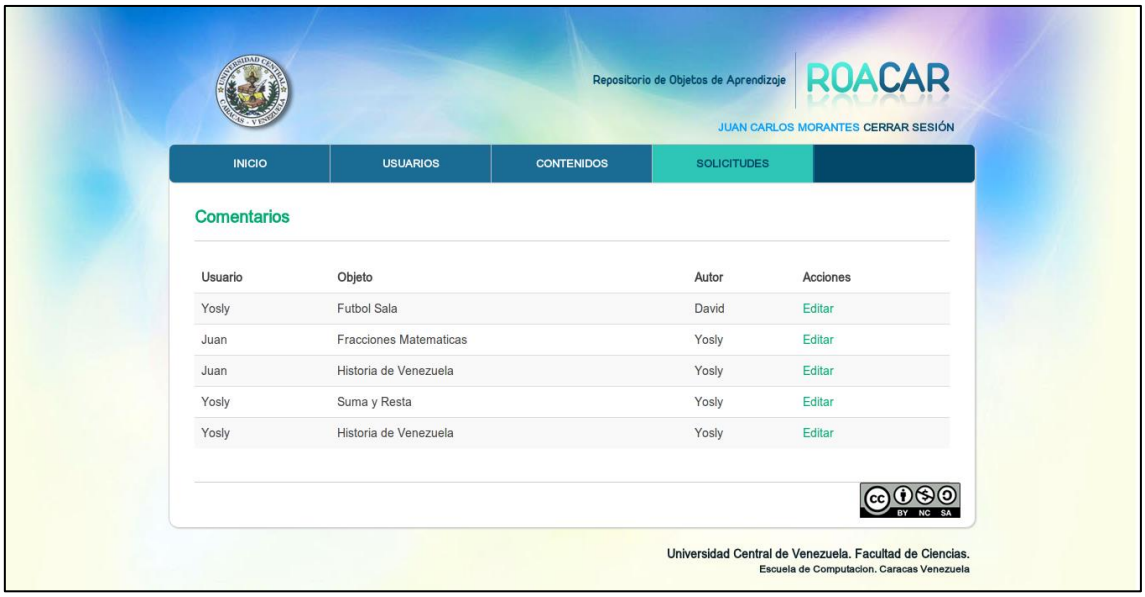


Figura 92. Vista de la lista de Comentarios

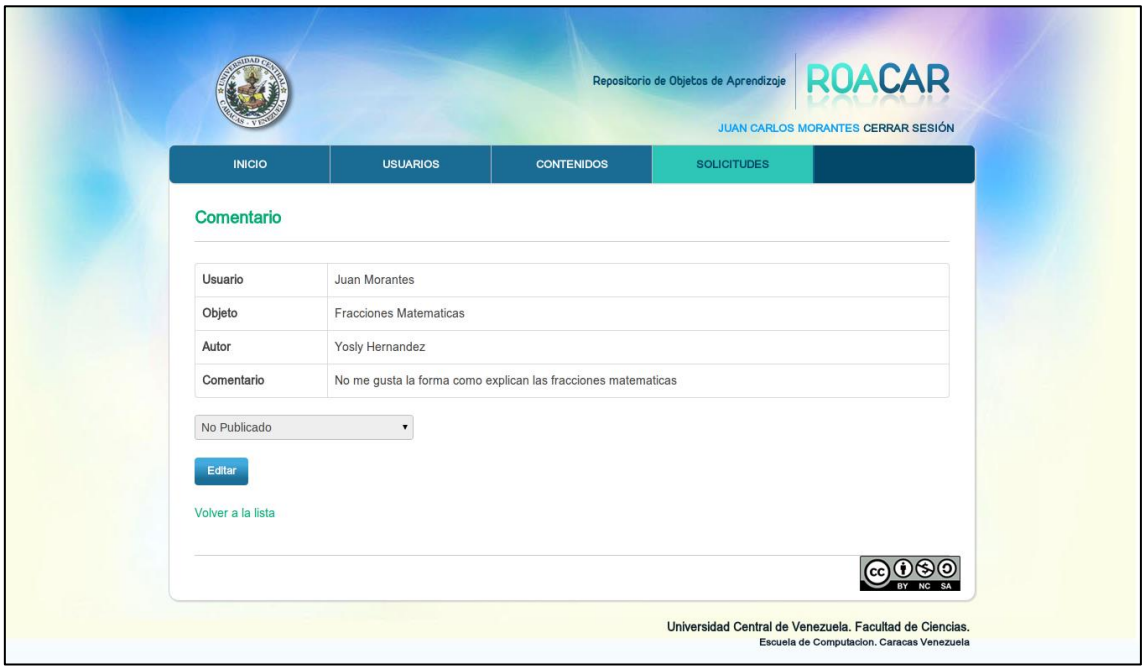


Figura 93. Formulario de edición de los comentarios

2.4.2 OACA

A Través de esta sección el usuario Administrador puede gestionar todos los OACA del Repositorio. Las funciones permitidas son:

a) **Ver:** permite visualizar el OACA directamente en el navegador.
b) **Descargar:** permite descargar un paquete comprimido con el OACA.
c) **Editar:** permite editar tanto la valoración del OACA como su estado (No Autorizado, En Revisión, Certificado). Los únicos OACA que podrán verse públicamente en el Módulo Usuario serán aquellos cuyo estado sea Certificado. En la Figura 94 se puede apreciar el formulario para editar un OACA.
d) **Eliminar:** permite eliminar un OACA del Repositorio.

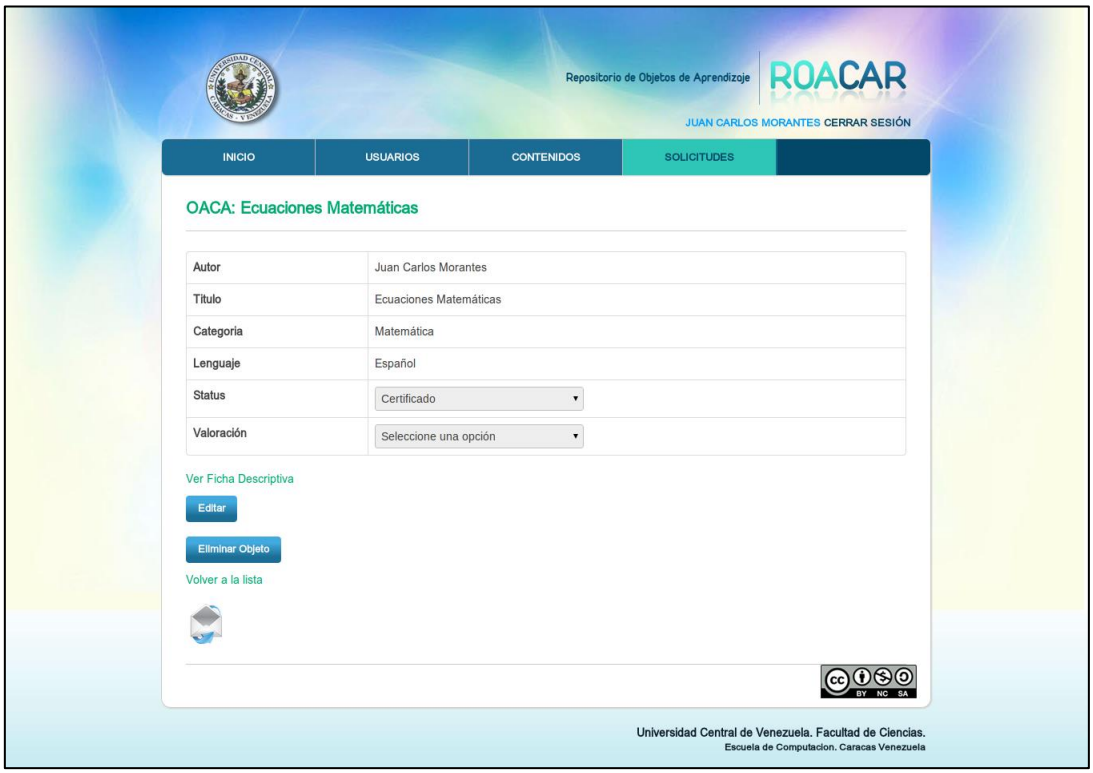


Figura 94. Formulario para editar OACA (Módulo de Administración)

Conclusiones

El Repositorio desarrollado provee un mecanismo seguro y usable para la gestión de los OACA, el cual permite la accesibilidad de los recursos educativos desde cualquier lugar que disponga de una conexión a Internet. Además, los OACA almacenados contienen todos los metadatos propuestos por el estándar LOM, lo que permite la ubicación y reutilización en otros sistemas o en otros entornos educativos. Por otra parte, ROACAR permite la interacción entre sus usuarios a través de la publicación de comentarios, siendo ésta una característica poco común en comparación con otros sistemas del mismo tipo. Adicionalmente, los usuarios no tienen que preocuparse por instalar plugins, programas o herramientas especiales para correr localmente los archivos que conforman a los OACA, ya que el sistema cuenta con la funcionalidad de visualizar directamente en el navegador Web cualquiera de los recursos sin importar su formato.

Otro aspecto importante, es que se fomenta la calidad de los OACA, ya que son sometidos a un proceso de revisión a través del cual se determina si el recurso cumple con las normas mínimas de calidad para poder ser publicado. Todas las funcionalidades mencionadas anteriormente hubiesen sido imposibles de llevar a cabo sin la utilización de una Repositorio como ROACAR.

El Repositorio fue desarrollado utilizando tecnologías actuales entre las que destaca el framework Symfony2, lo que significó un reto personal en vista de que fue necesario llevar a cabo un proceso de aprendizaje, además de adaptarse a las normas y principios establecidos por dicho framework para poder sacarle provecho a las bondades que ofrece.

En cuanto a la metodología de desarrollo, se implementó la Programación Extrema, la cual facilitó el trabajo de forma organizada, agrupando los requerimientos en un conjunto de iteraciones que se fueron desarrollando progresivamente. Además, la constante comunicación con los actores involucrados en el proceso permitió realizar frecuentes revisiones de los avances del Repositorio. Se puede afirmar que la utilización de ésta exitosa metodología sirvió de aprendizaje y experiencia para futuros proyectos.

Finalmente, debido a la elaboración de este Trabajo Especial de Grado, se logró un aporte significativo a la Universidad Central de Venezuela, ya que no se contaba con un espacio accesible, robusto, seguro y confiable que permitiera almacenar de forma organizada todos los OACA que se desarrollan en esta área. De ahora en adelante, será mucho más práctico gestionar todos estos recursos a través de ROACAR, y adicionalmente, existe la posibilidad de comunicación con otros Repositorios. Por lo tanto, se puede afirmar que los resultados obtenidos fueron altamente satisfactorios ya que se logró cubrir con todos los objetivos planteados.

Recomendaciones

A pesar del gran aporte que ha significado ROACAR para la gestión de los OACA, se pueden considerar algunas mejoras y nuevas funcionalidades que permitan el constante crecimiento del Repositorio, como las siguientes:

a) Reconstruir todas las interfaces del Repositorio empleando la técnica de desarrollo Web conocida como Diseño Web Adaptable (en inglés, Responsive Web Design), con el objetivo de ofrecer al usuario un diseño que se adapte a cualquier resolución de pantalla. De esta forma, el Repositorio estaría optimizado para todo tipo de dispositivos como computadores de escritorio, tabletas, teléfonos móviles, entre otros.

b) Completar la sección de “Contenidos” del Módulo de Administración para permitir que el Administrador pueda editar otros contenidos además de las categorías de los OACA, como por ejemplo el texto de bienvenida, las políticas, los créditos, los mensajes de error de los formularios, entro otros.

c) Sacar provecho de la arquitectura orientada a servicios de ROACAR para establecer una conexión con otras aplicaciones y/o crear una federación de Repositorios de la Universidad Central de Venezuela.

d) Incorporar en el Repositorio un sistema de evaluación de OACA que permita automatizar los procesos referentes al control de calidad. Esto reduciría notablemente la carga de trabajo del Administrador, el cual debe revisar y asignar una valoración a cada uno de los recursos que son cargados en el Repositorio.

Referencias Bibliográficas

Instituto Nacional de Estadísticas, Geografía e Informática de México. (2003). Recuperado el 2 de Octubre de 2013

Repositorio de Objetos de Aprendizaje para la Universidad Simón Bolívar. (2011). Recuperado el 23 de Mayo de 2012, de http://esopo.usb.ve

ADL. (2002). *Emerging And Enabling Technologies for the design of Learning Object Repositories.* Recuperado el 10 de Mayo de 2012, de http://xml.coverpages.org/ADLRepositoryTIR.pdf

APROA. (2005). *Aprendiendo con Objetos de Aprendizaje*. Recuperado el 15 de Mayo de 2012, de http:/www.aproa.cl

Blanco, S. (2004). *Biblioteca Semántica de Webquest.* Recuperado el 25 de Mayo de 2012, de http://webquest.xtec.cat/articles/diversos_autors/Blanco_tesis.pdf

Burgos, J. M., Galve, J., García, J., & Sutil, M. (2002). *Modelo Conceptual para la Organización del Aprendizaje.* Recuperado el 14 de Abril de 2012, de http://www.lsi.us.es/iberamia2002/confman/SUBMISSIONS/374-oss-ortuel.PDF

Cabezas, L. (2004). *Manuel Imprescindible de PHP5.* Recuperado el 12 de Mayo de 2013, de http://alvareitor.programasfull.com/php-php5-manual-descargar-espanol.html

De la Cueva, J. (2007). *Propiedad intelectual, nuevas tecnologías y libre acceso a la cultura.* Recuperado el 16 de Julio de 2013, de http://www.ccemx.org/procomun/?p=157

Downes, S. (2004). Recuperado el 20 de Mayo de 2012, de http://www.downes.ca/files/book3.pdf

Fernadez Nogales, A. (2004). *Investigación y Técnicas de Mercado.* ESIC Editorial.

Galeana, L. (2012 de Abril de 2004). *Objetos de Aprendizaje.* Recuperado el 18 de Julio de 2012, de http://www.cudi.edu.mx/primavera_2004/presentaciones/Lourdes_Galeana.pdf

García, F. (2000). *Modelo de Reutilización Soportado por Estructuras Complejas de Reutilización.* Recuperado el 19 de Mayo de 2012, de http://gredos.usal.es/jspui/bitstream/10366/21860/1/978-84-7800-920-6.pdf

Ginestà, M., & Perez, O. (2007). *Bases de Datos en PostgreSQL.* Recuperado el 10 de Marzo de 2012, de http://materials.cv.uoc.edu/cdocent/_8VJL129J4BABEVPA35V.pdf

Herández, Y. (2009). *Conceptualización de Objetos de Aprendizaje.* Recuperado el 3 de Octubre de 2013

Hilera, J. (2006). *Tecnologías de implementación de Repositorios de Objetos de Aprendizaje.* Recuperado el 6 de Mayo de 2013, de http://www.cc.uah.es/hilera/

IEEE. (2002). *Draft Standard For Learning Object Metadata.* Recuperado el 16 de Abril de 2012, de http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf

Iniciativa Colaborativa de Objetos de Aprendizaje Utilizables y Reutilizables. (s.f.). Recuperado el 15 de Mayo de 2012, de http://roa.mppeu.gob.ve/

L'Allier, J. (1997). *Frame of Reference: NETg's Map to Its Products, Their Structures and Core Beliefs.* Obtenido de http://web.archive.org/web/20020615192443/www.netg.com/research/whitepapers/frameref.asp

Laguna, M. (2011). *Introducción al Modelo de Referencia SCORM.* Recuperado el 15 de Abril de 2012, de http://unizar.es/innovacion/ecoleccion1/archivos/PDF/49_SCORM_CURSO.pdf

Leslie, S., Landond, B., Lamb, B., & Poulin, R. (2011). *Learning Object Repository Software.* Recuperado el 18 de Mayo de 2012, de http://web.archive.org/web/20020615192443/www.netg.com/research/whitepapers/frameref.asp

López, C. (2005). *Los Repositorios de Objetos de Aprendizaje como soporte a un entorno E-Learning.* Recuperado el 20 de Mayo de 2012, de http://gredos.usal.es/jspui/bitstream/10366/56649/1/DIA_Repositoriosobjetos.pdf.pdf

Los Santos, A. (2009). *Revisión de los Servicios Web SOA/REST: Características y Rendimiento.* Recuperado el 17 de Mayo de 2012, de http://www.albertolsa.com/wp-content/uploads/2009/07/mdsw-revision-de-los-servicios-web-soap_rest-alberto-los-santos.pdf

Márquez, S. (2007). *La Web Semántica.* Recuperado el 20 de Enero de 2013, de http://books.google.co.ve/books/about/La_Web_Sem%C3%A1ntica.html?hl=es&id=afuncWknStoC&redir_esc=y

Navarro, R. (2006). *Modelo, Diseño e implementación de Servicios Web.* Recuperado el 7 de Abril de 2013, de http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf

Pacheco, N. (2011). *Manual de Symfony2 en Español.* Recuperado el 28 de Marzo de 2012

Polsani, P. (2003). *Use and Abuse of Reusable Learning Objects.* Recuperado el 15 de Abril de 2013, de http://www.et.iitb.ac.in/jclub/doku/lib/exe/fetch.php?media=polsani_1_.pdf

Recursos Educacionales Multimedia Para El aprendizaje y Enseñanza Online. (s.f.). Recuperado el 4 de Mayo de 2012, de http://www.merlotchile.cl/

Rehak, D., & Manson, R. (2003). *Keeping the Learning in Learning Objects.* Recuperado el 27 de Mayo de 2012, de http://oro.open.ac.uk/800/

Rodriguez, J. (2001). *Revisión de Metadatos Educativos y su Utilización en Sistemas de Intermediación.* Recuperado el 3 de Abril de 2012, de http://www.esev.ipv.pt/3siie/actas/actas/doc43.pdf

Rouyet, J. (2004). *A Comparative Study of the Metadata SCORM and Dublin Core.* Recuperado el 5 de Abril de 2013, de http://spdece.uah.es/papers/Rouyet_Final.pdf

Singh, R. (2004). *A Modelfor Maintaining Interoperability of Coarse XML Sharable Learning Objects after ReAuthoring in a Standards-Based Editor.* Recuperado el 10 de Julio de 2013

Tamayo Yero, M. J., Lemes Báez, J. J., & Naranjo Ortiz, T. (2011). *Sistema Integral para el Control de Cuentas de Dominio Provincial.* Recuperado el 14 de Mayo de 2012, de http://jornada2011.sld.cu/index.php/jornada/2011/paper/viewFile/105/39

Tramullas, J. (2002). *Propuestas de Concepto y Definición de la Biblioteca Digital.* Recuperado el 5 de Octubre de 2013, de http://eprints.rclis.org/15118/1/04_2002.pdf.upv.es/jbidi/jbidi2002/Camera-ready/Sesion1/S1-1.pdf

Wiley, D. (2000). *Connecting Learning Objects to Instructional Desig Theory: a Definition, a Metaphor, and Taxonomy.* Recuperado el 25 de Julio de 2013, de http://wesrac.usc.edu/wired/bldg-7_file/wiley.pdf

Wiley, D. (2007). *The Learning Objects Literature.* Recuperado el 28 de Agosto de 2013, de http://www.opencontent.org/docs/wiley-lo-review-final.pdf

Wiley, D. (2009). *Creating Open Educational Resources.* Recuperado el 15 de Noviembre de 2013

Wolter, R. (2001). *XML Web Services Basicis.* Obtenido de http://msdn.microsoft.com/en-us/library/ms996507.aspx

Zambrano, S., & León, J. C. (Febrero de 2010). *Programación Extrema - Informe.* Recuperado el 26 de Marzo de 2010, de http://www.scribd.com/doc/26495149/Programacion-extrema-Informe

ANEXO

Historias de Usuario

Número: 1	**Prioridad:** MEDIA	**Estimación:** 5 días
Nombre: Diseñar el prototipo de interfaz del Módulo de Usuario		
Descripción: Diseño de las principales pantallas del Módulo de Usuario en código HTML.		

Número: 2	**Prioridad:** MEDIA	**Estimación:** 2 días
Nombre: Diseñar logo del Repositorio		
Descripción: Diseñar el logo del Repositorio usando la herramienta Photoshop.		

Número: 3	**Prioridad:** ALTA	**Estimación:** 1 día
Nombre: Realizar la configuración inicial de Symfony2		
Descripción: Chequear los requisitos para la correcta instalación de Symfony2 y realizar la configuración para el acceso a la Base de Datos		

Número: 4	**Prioridad:** ALTA	**Estimación:** 2 días
Nombre: Desarrollar la Clase Usuario		
Descripción: Crear la Clase Usuario con todos sus métodos y atributos		

Número: 5	**Prioridad:** ALTA	**Estimación:** 3 días
Nombre: Desarrollar el controlador UsuarioController		
Descripción: Crear el controlador UsuarioController con sus métodos básicos		

Número: 6	**Prioridad:** ALTA	**Estimación:** 6 días
Nombre: Configurar el archivo de seguridad de Symfony2 y crear las funciones necesarias para el inicio de sesión.		
Descripción: Configurar el archivo de seguridad de Symfony2 con todos los parámetros necesarios para manejar la autenticación y el control de acceso a las diferentes secciones del Repositorio, y crear los métodos necesarios para el manejo de las sesiones.		

Número: 7	**Prioridad:** MEDIA	**Estimación:** 1 día
Nombre: Desarrollar las vistas de la Clase Usuario		
Descripción: Crear todas las vistas de la Clase Usuario en código HTML con sus respectivas hojas de estilo.		

Número: 8	**Prioridad:** BAJA	**Estimación:** 1 día
Nombre: Desarrollar validaciones del formulario de Registro de Usuario		
Descripción: Añadir los parámetros necesarios a la Clase Usuario para realizar las validaciones de los formularios correspondientes a dicha Clase.		

Número: 9	**Prioridad:** ALTA	**Estimación:** 5 días
Nombre: Crear la Clase OA		
Descripción: Crear la Clase OA con todos sus métodos, atributos y relaciones con otros modelos.		

Número: 10	**Prioridad:** ALTA	**Estimación:** 5 días
Nombre: Crear todos los Modelos correspondientes a los metadatos		
Descripción: Crear todos los Modelos correspondientes a los metadatos junto con todos sus métodos y atributos.		

Número: 11	**Prioridad:** ALTA	**Estimación:** 2 días
Nombre: Crear el controlador OAController		
Descripción: Crear el controlador OAController con sus métodos básicos.		

Número: 12	**Prioridad:** ALTA	**Estimación:** 8 días
Nombre: Desarrollar las funciones CRUD dentro del OAController		
Descripción: Crear todas las funciones necesarias dentro del controlador OAController para listar, crear, actualizar y eliminar OA pertenecientes a la Clase OA.		

Número: 13	**Prioridad:** MEDIA	**Estimación:** 1 día
Nombre: Configurar las rutas de la Clase OA		
Descripción: Actualizar el archivo de rutas para permitir el acceso a todas las		

funciones correspondientes a la Clase OA.

Número: 14	**Prioridad:** BAJA	**Estimación:** 3 días
Nombre: Desarrollar las vistas correspondientes a la Clase OA		
Descripción: Diseñar las vistas HTML correspondientes a la Clase OA junto con sus respectivas hojas de estilo.		

Número: 15	**Prioridad:** ALTA	**Estimación:** 2 día
Nombre: Crear la Clase Comentario		
Descripción: Crear la Clase Comentario con todos sus métodos y atributos.		

Número: 16	**Prioridad:** ALTA	**Estimación:** 2 día
Nombre: Crear el controlador ComentarioController		
Descripción: Crear el controlador ComentarioController con sus métodos básicos.		

Número: 17	**Prioridad:** ALTA	**Estimación:** 1 día
Nombre: Crear una función dentro del controlador ComentarioController que permita la publicación de comentarios		
Descripción: Crear una función dentro del controlador ComentarioController que le permita a los usuarios realizar comentarios sobre cualquier OA publicado.		

Número: 18	**Prioridad:** MEDIA	**Estimación:** 4 días
Nombre: Añadir la función de visualización en el controlador OAController		
Descripción: Añadir una función en el controlador OAController que permita visualizar los OA directamente en el navegador independientemente del formato de los OA.		

Número: 19	**Prioridad:** MEDIA	**Estimación:** 1 día
Nombre: Añadir la función de descarga en el controlador OAController		

Descripción: Añadir una función en el controlador OAController que permita descargar cualquier OA publicado independientemente de su formato.

Número: 20	**Prioridad:** MEDIA	**Estimación:** 1 día
Nombre: Actualizar el archivo de rutas de la Clase OA		
Descripción: Actualizar el archivo de rutas de la Clase OA para permitir el acceso a las funciones de visualización y descarga		

Número: 21	**Prioridad:** BAJA	**Estimación:** 1 día
Nombre: Diseñar la interfaz de búsqueda avanzada		
Descripción: Crear la interfaz de búsqueda avanzada en código HTML con sus respectivas hojas de estilo.		

Número: 22	**Prioridad:** ALTA	**Estimación:** 5 días
Nombre: Programar las funciones necesarias para realizar la búsqueda avanzada		
Descripción: Programar las funciones que permitan realizar consultas a Base de Datos filtrando por los parámetros establecidos en la búsqueda avanzada.		

Número: 23	**Prioridad**: MEDIA	**Estimación:** 1 día
Nombre: Actualizar la Clase OA para manejar el número de descargas		
Descripción: Actualizar la Clase OA para añadir los atributos y las relaciones necesarias para manejar el número de descargas de cada OA.		

Número: 24	**Prioridad:** ALTA	**Estimación:** 1 día
Nombre: Desarrollar método para obtener los OA más descargados		
Descripción: Programar las funciones necesarias para consultar en Base de Datos por los OA más descargados.		

Número: 25	**Prioridad:** ALTA	**Estimación:** 1 día
Nombre: Desarrollar método para obtener los OA más recientes		

Descripción: Programar las funciones necesarias para consultar en Base de Datos por los OA más recientes.

Número: 26	**Prioridad:** BAJA	**Estimación:** 2 días
Nombre: Diseñar interfaz de la ficha descriptiva		
Descripción: Crear la interfaz de la ficha descriptiva en código HTML con su respectiva hoja de estilo.		

Número: 27	**Prioridad:** MEDIA	**Estimación:** 2 días
Nombre: Diseñar la interfaz principal del Módulo de Administración.		
Descripción: Crear la interfaz principal del Módulo de Administración con sus respectivas hojas de estilo.		

Número: 28	**Prioridad:** ALTA	**Estimación:** 2 días
Nombre: Desarrollar la Clase Categoría		
Descripción: Desarrollar la Clase Categoría con todos sus métodos, atributos y relaciones.		

Número: 29	**Prioridad:** ALTA	**Estimación:** 4 días
Nombre: Desarrollar las funciones CRUD de las categorías de los OA		
Descripción: Desarrollar todas las funciones necesarias para listar, crear, editar y eliminar las categorías de los OA.		

Número: 30	**Prioridad:** MEDIA	**Estimación:** 1 día
Nombre: Desarrollar las vistas correspondientes a la Clase Categoría		
Descripción: Diseñar las vistas HTML correspondientes a la Clase Categoría junto con sus respectivas hojas de estilo.		

Número: 31	**Prioridad:** ALTA	**Estimación:** 4 días
Nombre: Desarrollar las funciones del CRUD de usuarios		
Descripción: Desarrollar todas las funciones necesarias para listar, crear, editar y eliminar las usuarios a través del Módulo de Administración.		

Número: 32	**Prioridad**: MEDIA	**Estimación:** 1 día
Nombre: Añadir un nuevo campo en la Clase OA que corresponda con la calificación del OA.		
Descripción: Actualizar la Clase OA y añadir un nuevo atributo correspondiente a la valoración del OA.		

Número: 33	**Prioridad**: MEDIA	**Estimación:** 1 día
Nombre: Añadir un nuevo campo en la Clase OA que corresponda con el estado del OA (en revisión, certificado y no autorizado)		
Descripción: Actualizar la Clase OA y añadir un nuevo atributo correspondiente a su estado, el cual determinará la visibilidad del OA en la lista de OA publicados.		

Número: 34	**Prioridad**: MEDIA	**Estimación:** 1 día
Nombre: Añadir un nuevo campo en la Clase Comentario que corresponda con su estado (publicado o no publicado)		
Descripción: Actualizar la Clase Comentario y añadir un nuevo atributo correspondiente a su estado, el cual determinará la visibilidad del comentario en los metadatos del OA.		

Número: 35	**Prioridad**: MEDIA	**Estimación:** 1 día
Nombre: Actualizar las consultas en Base de Datos referentes a los OA y los comentarios.		
Descripción: Actualizar las consultas en Base de Datos referentes a los OA y los comentarios para poder mostrar a los usuarios únicamente los OA y comentarios que hayan sido aprobados por el Administrador.		