UNIVERSIDAD AUTONOMA
DE BAJA CALIFORNIA

INSTITUTO DE INGENIERIA

GESTION DE PERSISTENCIA PARA OBJETOS DE APRENDIZAJE

TESIS

P R E S E N T A:
LSC. José de Jesús Esparza Flores

DIRECTOR DE TESIS
Dr. Gabriel Alejandro López Morteo

Mexicali, Baja California a 10 de Diciembre de 2008.

Agradecimientos

Agradezco primeramente a Dios por llenarme de bendiciones y permitirme llegar hasta este momento de mi vida. Un profundo agradecimiento a mi padre José Manuel Esparza Flores y a mi madre María Esthela Flores Venegas por todo su cariño, apoyo y esfuerzo dedicado a lograr mis metas. Agradezco a mis hermanos Gaby, Jesy y Lalo que han estado conmigo apoyándome durante todo este tiempo. Gracias mi futura esposa Melissa Cervantes por permitirme compartir con ella este logro y estar a mi lado.

Agradezco a mis maestros Brenda Flores, Marcela Rodríguez, Larisa Burtseva, Jorge Ibarra y Martín Díaz por sus valiosas enseñanzas. Quiero agradecer especialmente a mi director de tesis, Gabriel López Morteo, por su valioso tiempo dedicado a este trabajo, por sus consejos y por todos los conocimientos que compartió conmigo. Gracias a mis compañeros Emmanuelle, Rene, Rodolfo y en especial a Lorena Castro por haber compartido conmigo los momentos mas difíciles y satisfactorios durante este proceso y por ser una excelente amiga y compañera. Agradezco también a mis compañeros de la Dirección General de Informática, en especial a Lourdes Baltazar, Mauro Gil y Salvador Cabello por haberme dado la oportunidad de seguir creciendo en mi carrera profesional.

Por último quiero agradecer a todas aquellas personas que de alguna manera hicieron posible la terminación de este trabajo de tesis y que no las mencione, gracias a todos.

José de Jesús Esparza Flores

Dedicatoria

Este trabajo de tesis esta dedicado a los pilares de mi vida: A mis padres José Manuel Esparza Flores y María Esthela Flores Venegas, por estar siempre a mi lado cuando más los necesité y porque mi esfuerzo es el suyo también, los quiero mucho.

A mi futura esposa Melissa Cervantes, por sus consejos día a día, por apoyarme en mis proyectos y por darle un nuevo horizonte a mi vida. Aun nos queda mucho camino por recorrer. Te amo.

Resumen

Uno de los principales requerimientos con los que deben cumplir los objetos de aprendizaje tiene que ver con la persistencia de estado. Existen varias propuestas que pueden satisfacer este requerimiento, sin embargo es importante tomar en cuenta a uno de los principales usuarios de los objetos de aprendizaje, el diseñador instruccional, por lo que resulta conveniente ofrecer una solución a los problemas de persistencia que no requiera un conocimiento amplio sobre sistemas de almacenamiento, parámetros de configuración o cualquier otro elemento que complique la labor del diseñador al momento de seleccionar un esquema de persistencia para el estado de los objetos de aprendizaje. De esta manera se propone una solución con base en los estándares de SCORM, siendo la principal característica de la solución descrita en este articulo la utilización de tanto el modelo de datos y el API de comunicaciones de SCORM, sin que esto signifique una extensión de ambos componentes, permitiendo así cumplir con el estándar y ampliando las posibilidades de su utilización con cursos ya elaborados.

Abstract

One of the main requirements to be satisfied with the objects of learning has to do with the persistence of state. There are several proposals that can meet this requirement, however it is important to take into account one of the main users of the objects of learning, instructional designer, which makes it desirable to offer a solution to the problems of persistence that does not require knowledge comprehensive storage systems, settings or any other element that complicates the work of designer at the time of selecting a scheme of persistence for the state of learning objects. Thus it is proposed a solution based on the standards of SCORM, being the main feature of the solution described in this article using both the data model and API communications SCORM, but this means an extension of both components, thus enabling meet the standard and expanding the possibilities for use with courses already developed.

Índice general

Lista de código fuente

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Los Instructores Interactivos de Diversiones Matemáticas (IIDM)[1] se definen como un componente de software educativo especializado en conceptos matemáticos, representado a través de diversiones matemáticas con el cual uno o varios individuos pueden interactuar con él y entre ellos. Los IIDM tienen el propósito fundamental de generar cierto conocimiento matemático y apoyar la transmisión de conceptos matemáticos de una manera atractiva. Una de las características principales de los IIDM es que se pueden utilizar independientemente del nivel académico del usuario, esto es, el contexto en el que se utiliza es el que define el tipo de usuarios o grupos de usuarios al que va dirigido. [López & Ibarra, 2001]

Cada IIDM es un elemento independiente, por lo que puede existir por si mismo sin el soporte de un sistema computacional especializado o conciencia de ambiente. El IIDM puede ser presentado como una página Web que contiene texto, imágenes, vídeo elementos dinámicos interactivos o aplicaciones de java incrustadas. Por lo tanto pueden ser distribuidos en línea, cd-rom o en cualquier otro medio de almacenamiento o estar disponibles en repositorios Web.

Los IIDM son objetos complejos que necesitan la intervención de diferentes áreas de conocimiento como soporte y con el objetivo de fomentar la interacción humano-máquina y humano-humano que auxilien el proceso de aprendizaje. Un IIDM presentan ciertos elementos computacionales característicos.

1. Acceso simple y rápido al sistema.
2. Capacidad de ser utilizado aisladamente o con conectividad a sitios remotos.
3. Interfaces sencillas y comprensibles.
4. Interacción entre el individuo y el IIDM.
5. Persistencia del estado de la interacción entre el individuo y el IIDM.
6. Soporte de la interacción multiusuarios.
7. Soporte de aprendizaje colaborativo.

El modelo está diseñado para que en una sola entidad se incluyan elementos de diferentes áreas de las ciencias de la computación y del diseño instruccional. Así, el contenido resultante, incorpora características que lo hacen más atractivo para los estudiantes, comparado con sus contrapartes estáticas.

[1] http://dx.doi.org/10.1016/j.compedu.2005.04.014

El Ambiente de aprendizaje basado en Instructores Interactivos de Diversiones Matemáticas (AIIDM) se conceptualiza como un espacio colaborativo de acción para la transmisión de conocimiento matemático. El escenario de uso lo conforma un conjunto de IIDM en un entorno atractivo que fomenta el uso de los mismos. La arquitectura del ambiente basada en módulos independientes entre si permite dar soporte a diferentes aspectos del aprendizaje colaborativo.

El primer prototipo contenedor de AIIDM, es un ambiente virtual que ha sido denominado "Los Supersabios"[2], en honor al comic mexicano de ciencia ficción del mismo nombre. El AIIDM es un portal Web que funge como repositorio o contenedor de IIDM el cual ofrece una serie de servicios que contribuyen a proporcionar una experiencia de lo más provechosa para el usuario al momento de adquirir conocimiento matemático a través de los objetos contenidos en el repositorio. El control de identidad de cada uno de los usuarios dentro del portal es fundamental para lograr los objetivos de adaptar o configurar el ambiente de acuerdo a criterios específicos de preferencias y necesidades por parte del usuario. Cada uno de los objetos contenidos en el ambiente, posee funcionalidad propia e independiente, permitiendo al usuario a través de interfaces bien definidas especificar ciertas características propias del IIDM tales como maximizarlo, minimizarlo, regresarlo a su tamaño original e incluso eliminarlo así como también seleccionar dentro de una lista la combinación de colores para la barra de titulo y el contenido. Personalizando completamente el aspecto del ambiente de acuerdo a las preferencias de un usuario en particular. En la figura 1.1 se muestra la interfaz de personalización que el usuario puede configurar según sus preferencias.

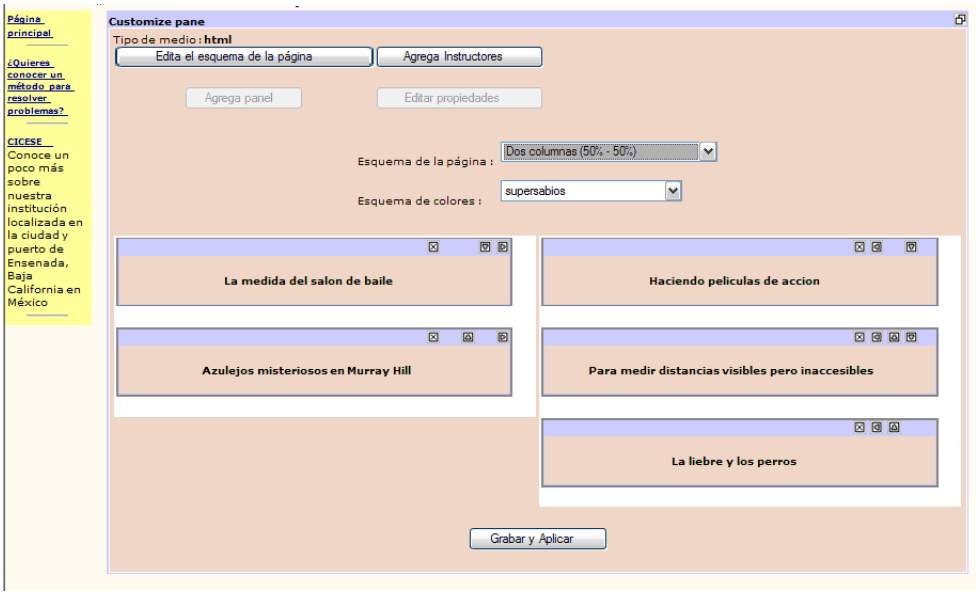


Figura 1.1: Panel de Personalización del AIIDM

Puesto que el AIIDM esta conformado por un conjunto de IIDM, es posible extender la funcionalidad del ambiente agregando o registrando nuevos IIDM ubicados dentro de un catálogo el cual muestra el titulo y una breve descripción donde el usuario selecciona aquellos IIDM que desea sean mostrados en la página principal del ambiente. En la figura 1.2 podemos ver el catálogo de IIDM. Los objetos relacionados con la interacción con otros usuarios tales como mensajería instantánea también se encuentran inmersos dentro del ambiente logrando así conjugar todos los elementos necesarios para adquirir un conocimiento colaborativo dentro del AIIDM. Cada elemento del catálogo es un Portlet (contenedor), el cual lleva inmerso un IIDM, por lo tanto el AIIDM puede cargar diferentes tipos de portlets y por lo tanto de IIDM que pueden ser paginas Web, archivos XML y aplicaciones Web.

[2]http://azul.cicese.mx/supersabios/portal

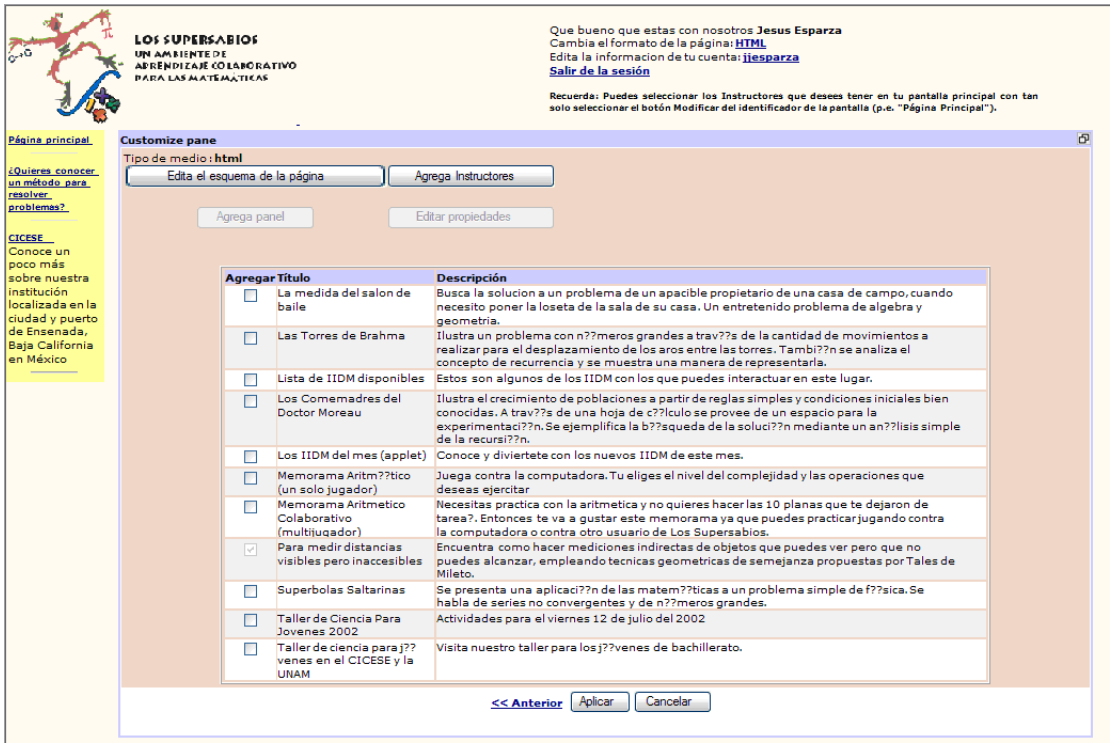


Figura 1.2: Catálogo de Instructores Interactivos de Diversiones Matemáticas.

La arquitectura del AIIDM[3] proporciona los siguientes servicios:

- *Servicio de Membresía*: Administra la información relacionada con el usuario y la actividad de la sesión, la información de sesión es utilizada por las herramientas de interacción y los objetos de aprendizaje interactivos durante su ejecución. Este servicio proporciona al ambiente un enfoque orientado hacia los usuarios permitiendo agregar nuevas características funcionales al AIIDM encaminadas a enriquecer el aprovechamiento de los IIDM llegando a instancias en las que podríamos extender este servicio a los IIDM agregando comportamiento o características específicas a cada IIDM que el usuario utilice dentro del AIIDM.
- *Servicio de Personalización*: Permite personalizar el espacio de trabajo del usuario de acuerdo a sus preferencias.
- *Servicio de Interacción*: Permite la interacción de forma síncrona (mensajería instantánea) y asíncrona (correo electrónico) que permiten la interacción de los usuarios dentro del espacio de trabajo.
- *Servicio de Catálogo*: provee una interfaz para navegar y recuperar objetos de aprendizaje a través de una lista.
- *Servicios de Interoperabilidad*: Permite incorporar contenido externo ya sea local o remoto al sistema, estos contenidos son incorporados al catálogo y a su contexto de ejecución para que puedan estar disponibles localmente.

Si bien la arquitectura del AIIDM formada por los servicios mencionados anteriormente cumple con el objetivo de proporcionar dentro de un contexto de colaboración los artefactos para la transmisión de conocimiento matemático,

[3] http://dx.doi.org/10.1016/j.compedu.2005.04.014

no ocurre lo mismo al momento de contemplar aquellas características que permiten a los IIDM comportarse de forma dinámica y atractiva para el usuario, tal es el caso de la persistencia del estado de la interacción entre el usuario y el IIDM. La falta de un servicio de persistencia por parte del AIIDM en casos determinados podría reducir sustancialmente el aprovechamiento del usuario al utilizar el IIDM o bien tener un servicio de persistencia, permitiría agregar nuevas características a los IIDM enriqueciendo aun mas la experiencia del usuario al utilizarlos dentro del AIIDM.

1.1 Definición del Problema

La necesidad de implementar un servicio de persistencia se hace visible en los múltiples casos en los que un IIDM requiere la captura de una cantidad significativa de datos por parte del usuario, tal es el caso del IIDM “Haciendo Películas de Acción” en donde el usuario realiza cálculos específicos para determinar la altura que alcanza un objeto en un tiempo y velocidad determinados.

En la figura 1.3 se muestra una tabla que forma parte del IIDM ”Haciendo Películas de Acción” en la que el usuario especificó el tiempo la altura aplicando una formula y los parámetros dados. En esta y otras tablas o campos del IIDM se almacenan datos provistos por el usuario. Si el usuario retomara el uso del IIDM en una sesión posterior, se vería en la necesidad de volver a realizar los cálculos o capturar nuevamente la información que ya había obtenido anteriormente. Esto ocurre ya que la implementación actual de los IIDM no contiene un esquema de persistencia de los datos introducidos por el usuario.

Si bien la cantidad de datos introducidos por el usuario puede ser significativa, el verdadero problema seria la imposibilidad por parte del AIIDM de conservar los datos introducidos en el IIDM por el usuario de una sesión de trabajo a otra.

Actualmente, existen modelos o mecanismos que serian de gran utilidad para la resolución de este problema, tal es el caso de implementar una comunicación directa desde el IIDM hacia un sistema de almacenamiento de información, esta posible solución implicaría poner en riesgo la seguridad y consistencia de la información al permitir accesos remotos desde y hacia el sistema de almacenamiento de información y debido a la naturaleza de la arquitectura en la cual se encuentran embebidos los IIDM en la que solamente se pueden comunicar por la red al mismo sistema servidor del cual son provistos hace imposible la comunicación con algún otro servidor, como podría ser el propio sistema de almacenamiento.

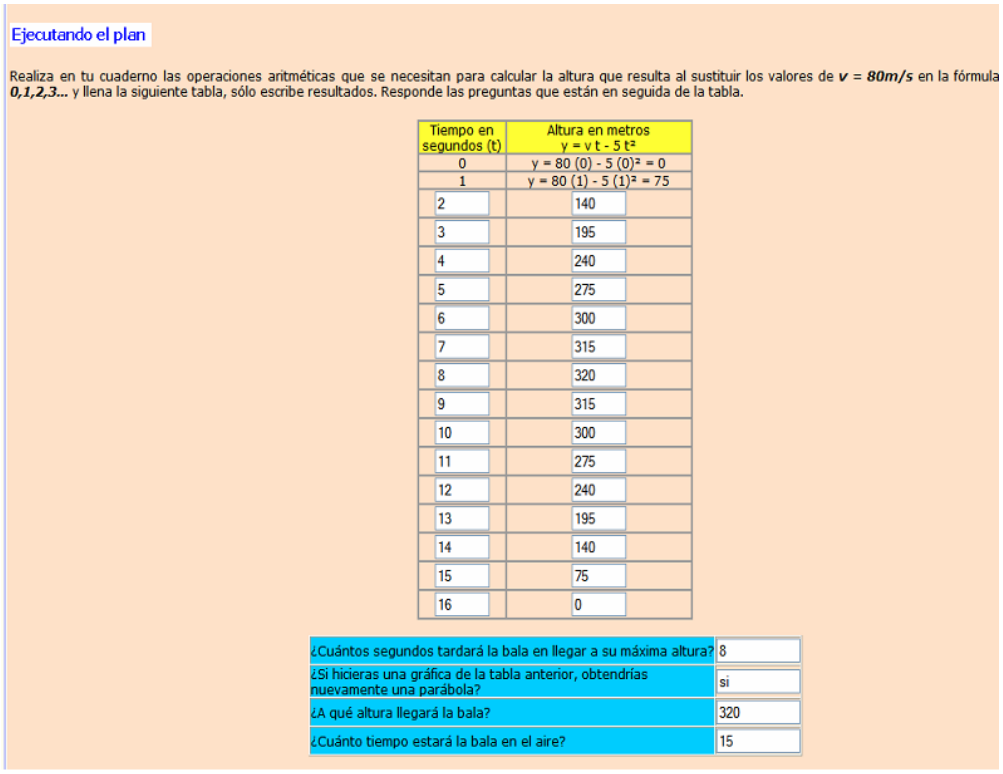


Figura 1.3: IIDM "Haciendo películas de acción", Cajas de entrada de datos

Esto nos lleva a pensar en la posibilidad de utilizar un sistema middleware en el que se deleguen los mecanismos de comunicación de envío y recepción de información desde y hacia los IIDM y el sistema de almacenamiento. Sin embargo, para implementar esta solución, se tendrían que contemplar una serie de cambios en la arquitectura del AIIDM. Cualquiera de las propuestas planteadas requiere codificar dentro de cada IIDM el esquema de persistencia que será utilizado, lo que limita al diseñador instruccional a utilizar solo un esquema de persistencia específico. Por lo tanto, para implementar cualquier otro esquema de persistencia dentro del IIDM implicaría necesariamente el trabajo de un desarrollador de software.

1.2 Justificación

En un estudio realizado por Marcos Galaviz Férman [FERMAN, 2006], en el cual se busca diseñar un modelo que permita adoptar los Instructores Interactivos de Diversiones Matemáticas a estudiantes en el aula de clases de nivel secundaria, se realizó un análisis para conocer los diferentes factores que pueden intervenir en determinada adopción, llegando a diferentes conclusiones entre las cuales podemos destacar las restricciones de tiempo.

A nivel secundaria solo se toman sesiones de 50 minutos para impartir la clase y cubrir todas las actividades de aprendizaje contenidas en el IIDM. En este tiempo los usuarios capturan en campos o tablas las respuestas a las preguntas plantadas o los resultados de los ejercicios propuestos dentro del IIDM.

En muchas ocasiones una sesión de trabajo no es suficiente para terminar todas las actividades de un determinado IIDM. Esto conlleva a tomar o invertir una parte considerable de sesiones futuras tratando de retomar el tema, volviendo a capturar o generar la información previamente obtenida, con el fin de recuperar el estado en el que se encontraba el IIDM y situarse de nuevo en el contexto del problema para concluir su desarrollo.

La falta de mecanismos de persistencia en el estado de las interacciones IIDM-usuario, se traduce en perdida de tiempo y retrabajo, volviendo al AIIDM y a los IIDM en una alternativa de material de trabajo poco atractiva para los profesores y alumnos.

1.3 Escenario de Uso

Para ilustrar de una mejor manera los requerimientos para el manejo de persistencia de datos entre una sesión a otra de los IIDM, se presenta el siguiente escenario de uso hipotético en donde se ejemplifica como los IIDM implementarían el servicio de persistencia ofertado por el AIIDM.

1. El diseñador instruccional debe especificar los atributos del IIDM que persistirán, dicha especificación de atributos no debe ser más complicada que nombrar a los atributos de cierta forma, colocarles un identificador o utilizar una nomenclatura dada.

2. Durante el registro del IIDM dentro del AIIDM, el diseñador instruccional o el profesor especificaran si el IIDM implementará el servicio de persistencia y de ser así deberá seleccionar el esquema de persistencia mas adecuado para el IIDM.

3. Una vez que el IIDM ha sido registrado y asociado con un esquema de persistencia, el administrador del AIIDM lo agregará dentro del servicio de catálogos de IIDM para que los usuarios puedan utilizarlo dentro de su propio ambiente de trabajo.

4. Cuando los usuarios agreguen el IIDM persistente, el AIIDM identificara que el IIDM implementa el servicio de persistencia y cargará en tiempo de ejecución los mecanismos necesarios que le permitan comunicarse con el sistema de almacenamiento para recuperar o guardar el estado del IIDM, es importante reiterar que dicha comunicación con el sistema de almacenamiento se realizara a través del AIIDM y no por el IIDM.

5. Una vez que el usuario deje de utilizar el IIDM persistente ya sea cerrándolo o cerrando la sesión dentro del AIIDM, los mecanismos de persistencia se comunicarán con el sistema de almacenamiento para guardar el estado actual en el que se encuentra el IIDM.

6. Posteriormente cuando el usuario vuelva a iniciar una sesión de trabajo en el AIIDM y utilice el IIDM persistente, los mecanismos de persistencia recuperarán el estado del IIDM cargándolo en memoria, permitiendo al usuario continuar trabajando con él y los datos introducidos en sesiones anteriores.

1.4 Objetivos

General

Diseñar e Implementar un mecanismo que permita la persistencia en tiempo de ejecución, considerando los datos introducidos en los Instructores Interactivos de Diversiones Matemáticas en cada sesión del usuario, los cuales sean registrados bajo algún esquema de persistencia en su contenedor.

Particulares

- Definir los esquemas de persistencia de datos que correspondan a los requerimientos didácticos asociados al uso de los IIDM.
- Desarrollar e implementar los mecanismos que permitan administrar el servicio de persistencia para los IIDM.
- Desarrollar e Implementar las interfaces necesarias para que un IIDM pueda incorporar un esquema de persistencia de datos provisto por su contenedor.

Preguntas de Investigación

Con el objetivo de identificar cuales son las necesidades de información requeridas para definir el alcance del problema actual, surgen las siguientes preguntas:

- ¿Existen modelos de persistencia relacionados con el ámbito educativo?
- ¿Cuales son las propuestas existentes para persistir el estado de un objeto de aprendizaje?
- ¿Cuales son los estandares actuales para el control de persistencia de estado en objetos de aprendizaje?
- ¿Cuáles son los requerimientos funcionales necesarios para implementar un modelo de persistencia en ambientes de aprendizaje electronicos?

1.5 Metodología

A continuación, se describen cada una de las etapas involucradas que serán llevadas a cabo para lograr los objetivos de la investigación, dichos objetivos han sido descritos anteriormente, conforme se desarrollen las etapas de la investigación se cubrirán cada uno de los objetivos particulares. En la figura 1.4 se muestran cada una de las etapas de la metodología de trabajo.

1.5.1 Analizar el Contexto del Problema

En esta etapa se obtendrá un panorama general, a partir de aquellos trabajos elaborados que se apegan a las características del problema planteado con la finalidad de identificar el alcance que tendrá la solución. El análisis de los trabajos relacionados también permitirá adquirir el conocimiento necesario relacionado con las metodologías actuales de los servicios de persistencia, en el capitulo primero y segundo se cubren y se desarrollan dichos puntos.

1.5.2 Analizar la Especificación Run-Time Environment SCORM

En esta etapa se hace una descripción de la especificación o estándar Sharable Content Reference Model (SCORM) para conocer su estructura y los estándares que incluye para el desarrollo, despliegue, utilización y navegación de contenido educativo dentro del Sistema Administrador de Aprendizaje (LMS por sus siglas en inglés). Es importante mencionar la diferencia que existe entre el contenedor de los IIDM persistentes, en este caso el LMS y el AIIDM, la cual radica en que el AIIDM posteriormente puede incorporar el LMS como portlets.

Posteriormente se hace un análisis del Ambiente de Ejecución (RTE por sus siglas en inglés) de SCORM para conocer sus características y las restricciones, reglas y políticas para desarrollar un Sistema de Administración de Contenido Educativo con el propósito de adoptar un modelo estandarizado para la ejecución de objetos de aprendizaje y extenderlo para implementar los mecanismos de persistencia sobre los IIDM. En el capítulo tercero cubre los aspectos relacionados con esta etapa de la metodología.

1.5.3 Realizar las Adecuaciones al LMS de SCORM para adaptarlo al AIIDM

En esta etapa de la metodología se realizan las adecuaciones necesarias al LMS de SCORM, las cuales permiten agregarle ciertas características de portabilidad con respecto al almacenamiento de la información relacionada con los usuarios y los cursos registrados dentro del LMS.

Al trabajar con un sistema manejador de base de datos DBMS embebido en el LMS eliminamos la dependencia o instalación de la base de datos por separado, lo que nos da la ventaja de cargar el LMS en cualquier servidor de aplicaciones Web para Java Enterprise Edition tal como tomcat, glassfish, appserver, etc.

Este LMS funcional tiene las capacidades mínimas para registrar, cargar y realizar la navegación de contenido educativo con base en el estándar de SCORM. La adecuación del LMS permitirá tener un acercamiento con todos aquellos problemas de implementación para cargar objetos de aprendizaje. Posteriormente, con el LMS y las adecuaciones implementadas, el siguiente paso es realizar pruebas con objetos de aprendizaje sobre SCORM, para evaluar su funcionamiento y el buen cumplimiento de los estándares que permitan cargar objetos de aprendizaje desarrollados por terceros sin ningún problema en el LMS. Estos temas son abarcados en el tercer capítulo.

1.5.4 Definir, Diseñar, Desarrollar e Implementar los Esquemas de Persistencia

En esta etapa se definen los esquemas que respondan a las necesidades de persistencia detectadas en el análisis del contexto del problema. Posteriormente a través de herramientas de modelado como UML, poder ver las diferentes perspectivas de los esquemas y encontrar todas aquellas restricciones, limitaciones y características que sea necesario tomar en cuenta para implementar los esquemas dentro del LMS. Así mismo definir las características que deben poseer tanto los objetos de contenido educativo como el LMS para explotar los esquemas de persistencia que permitan al LMS identificar que se trata de un objeto persistente y realizar los procesos adecuados de acuerdo al esquema de persistencia relacionado con el objeto de contenido educativo. El siguiente paso es desarrollar el software que de soporte a los esquemas de persistencia con base en el diseño previo, aplicando la metodología de desarrollo ágil de software Extreme Programming (XP). Esta metodología se basa en la retroalimentación continua entre el equipo de desarrollo y el cliente, la simplicidad en las soluciones implementadas y se especializa en proyectos con requisitos técnicos de alto riesgo, como es el caso de este proyecto, puesto que no hay muchas aplicaciones de persistencia de estado para LMS.

Posteriormente se implementara el software dentro del LMS para probar su buen funcionamiento, lo que implica adecuar el LMS para soportar los esquemas de persistencia. Antes de iniciar cualquier actividad ya sea de diseño, desarrollo o implementación es importante documentarse acerca de las características de la metodología ágil

para aplicarla en el proyecto de una manera satisfactoria, por lo tanto es una actividad importante que debemos considerar. Esta etapa se desarrolla dentro de los capítulos cuarto, quinto y sexto del documento.

1.5.5 Análisis y Metodología de Uso de los Esquemas de Persistencia

En el séptimo capítulo tiene el objetivo de dar a conocer desde un punto de vista critico las características de una posible solución a la persistencia de estado de los objetos de aprendizaje, la cual se desarrolla dentro de este documento de tesis, realizando una comparativa entre una propuesta descrita en [Kassahun et al., 2006], la propuesta presentada por SCORM y los Esquemas de persistencia, resaltando las ventajas y desventajas de cada una de las propuestas con el fin de conocer un mejor panorama de la situación actual referente a la persistencia de estado de los objetos de aprendizaje. Así como también explicar los procedimientos realizados para desplegar los cursos que tengan asignado algún esquema de persistencia, con la finalidad de proporcionar una guía de como utilizar los esquemas de persistencia propuestos e implementados y ampliar la visión acerca de los elementos que intervienen en cada una de las etapas del proceso para desplegar el curso.

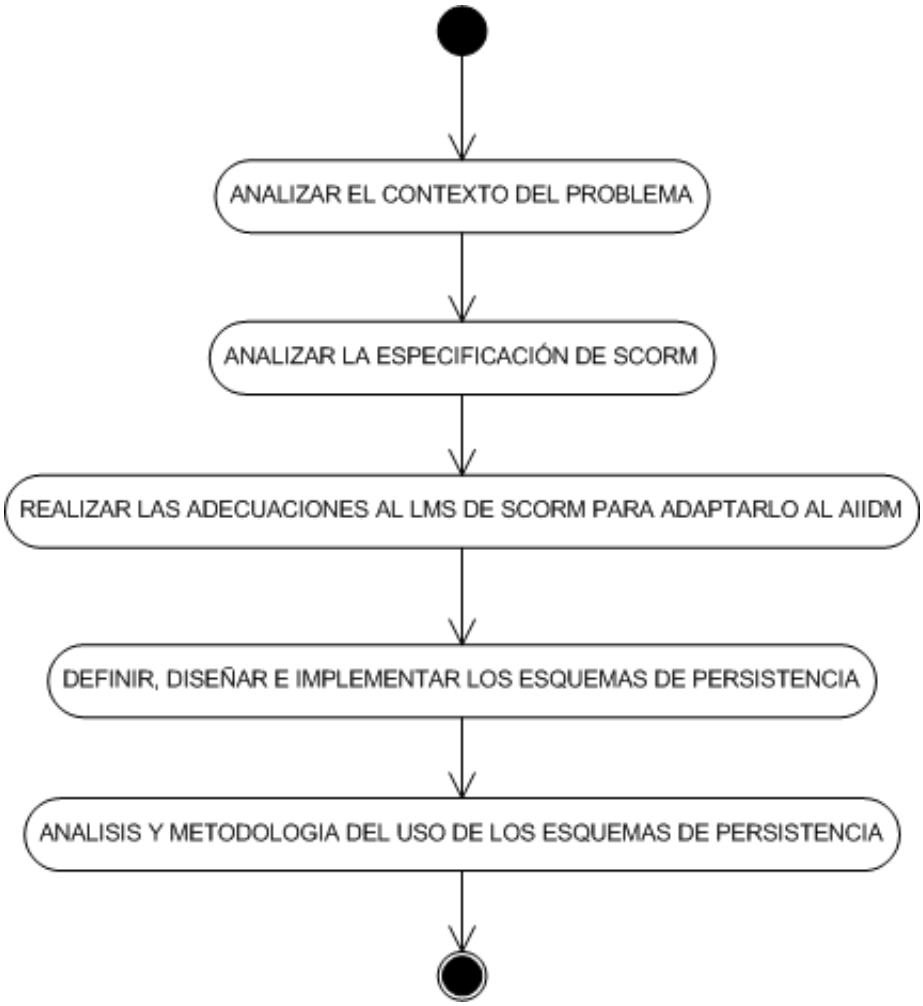


Figura 1.4: Etapas de la Metodología

1.6 Estructura del Documento

En esta sección se realiza una breve descripción de la estructura del presente documento de tesis, de acuerdo a los objetivos citados anteriormente. El documento esta dividido en siete capítulos y cada uno corresponden a las actividades planteadas en la metodología de trabajo.

En este capítulo se realiza una descripción de los objetos de aprendizaje especializados en conceptos matemáticos denominados Instructores Interactivos de Diversiones Matemáticas (IIDM), donde se hace un análisis de su comportamiento dentro del ambiente de aprendizaje que los contiene (AIIDM) con la finalidad de ofrecer un panorama del contexto en el cual se sitúa la problemática. Posteriormente dentro del mismo capítulo se aterriza la definición del problema con base en el análisis previo del AIIDM y las necesidades que tiene para ofrecer un mejor servicio respecto a la persistencia del contenido capturado en los IIDM por parte de los usuarios. Se presenta la justificación del problema que plantea y da a conocer las razones por las cuales el servicio de persistencia es una necesidad que debe ser atacada para mejorar la calidad de los cursos dentro del AIIDM. Seguidamente, con el propósito de entender cuales son las metas u objetivos del trabajo de tesis se presenta un escenario de uso que describe las actividades secuenciales que se deben llevar a cabo para ofrecer un servicio de persistencia y que son necesarias tener en cuenta al momento de definir los objetivos, por último dentro del mismo capítulo se plantea el objetivo general y los objetivos particulares a los cuales se pretender llegar al concluir el trabajo de tesis donde todas las actividades que se realizan se llevaran a cabo con el propósito de lograr dichos objetivos.

En este mismo capítulo se hace una descripción de las etapas que contempla la metodología de trabajo propuesta, la cual incluye primeramente trabajos de investigación del contexto del problema, para pasar luego a analizar una de los especificaciones de e-learning más destacada: SCORM. El análisis de esta especificación se realiza con el propósito de implementar y adecuar el sistema de administración de contenido educativo (LMS) que propone la especificación para desarrollar los servicios de persistencia de acuerdo con SCORM. Posteriormente de acuerdo con la metodología se realiza la definición y el diseño de los esquemas de persistencia que cumplan con ciertas necesidades pedagógicas. Después de esta etapa, se realizan los trabajos de desarrollo para implementar los esquemas de persistencia dentro de un LMS. Finalmente, como última etapa de la metodología se desarrollará un prototipo para implementar algunos esquemas de persistencia que permitan probar la solución desarrollada. Dentro de la descripción de cada una de las etapas se hace referencia a los capítulos involucrados en llevar a cabo las actividades propias de cada etapa.

En el segundo capítulo se realiza un análisis de todas aquellas tecnologías que están relacionadas con la persistencia de la información y que pueden ofrecer una posible solución que permita cubrir los objetivos planteados anteriormente. Después de cada análisis de estas tecnologías se mencionan algunas características que pudieran llegar a ser un problema o una restricción que impide tomar en cuenta dicha tecnología como solución para el problema de persistencia.

En el tercer capítulo se realiza un análisis del Modelo de Referencia para Objetos con contenido Educativo (SCORM). En este capítulo se describen todos aquellos conceptos relacionados con el modelo, su organización y los componentes por los cuales se conforma esta especificación. Continuando con un análisis a fondo de uno de los componentes del modelo, el Run-Time Environment (RTE) donde se describen de igual forma sus componentes, especificaciones y el tratamiento que se le da al contenido educativo que se apega al estándar SCORM. Este análisis se realiza con la finalidad de conocer mas a fondo las características del sistema que se basa en el RTE, llamado LMS, el cual nos permitirá desplegar el contenido educativo que posteriormente sera provisto con esquemas de persistencia de los datos introducidos por el usuario.

En el cuarto capítulo se describe el proceso de desarrollo de los mecanismos de persistencia, el cual está basado en la metodología de desarrollo ágil XP [Letelier & Penadés, 2006]. Los mecanismos de persistencia se implementan en el RTE de SCORM con el objetivo de proporcionar las bases que permitirán desarrollar especializaciones del comportamiento de dichos mecanismos con la finalidad de ofrecer diferentes alternativas de persistencia de estado.

En el quinto capítulo se explica la clasificación, los tipos y esquemas de persistencia obtenidos en los trabajos de desarrollo de la solución, con el fin de dar a conocer el propósito de cada uno y las formas en las que se pueden combinar para obtener soluciones mas robustas.

En el sexto capítulo se realiza una comparativa con otras alternativas de persistencia de estado para los objetos de aprendizaje y la solución propuesta en esta tesis. De igual forma, se da a conocer la metodología utilizada para el registro de los objetos de aprendizaje con características de persistencia en su estado y el despliegue de estos en clientes LMS ligeros.

En el séptimo capítulo se presentan las conclusiones, así como también las contribuciones obtenidas de este trabajo de investigación y se presentan los posibles trabajos que pudieran realizarse a futuro con el bien de contribuir a la mejora de los esquemas de persistencia.

Capítulo 2

Marco Teórico

Puesto que la mayoría de los ambientes de aprendizaje basados en computación solo mantienen persistente aquella información relacionada con datos generales de usuarios, información del aprovechamiento de acuerdo a criterios de evaluación específicos o información relacionada con los puntos de interrupción de un curso; no existen mecanismos o propuestas que permitan mantener el estado de la interacción del usuario con los objetos de aprendizaje, en particular con los IIDM. Por tal motivo, resulta evidente la necesidad de integrar aquellos mecanismos disponibles que permitan mantener el estado de la interacción entre el usuario y los objetos de aprendizaje entre una sesión de trabajo y otra.

Con el propósito de tener un panorama mas amplio de las técnicas o mecanismos de persistencia, a continuación se presenta un esquema taxonómico realizado por [Miranda et al., 2006] en donde el autor propone una clasificación de mecanismos bien definidos y orientados al desarrollador de sistemas que ayudan a seleccionar el mecanismo o la integración de mecanismos mas apropiados de acuerdo a las necesidades o requerimientos del sistema, la cual se muestra en la figura 2.1.

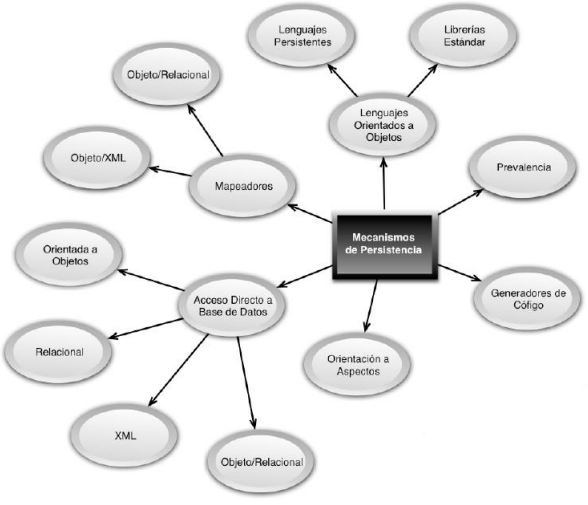


Figura 2.1: Clasificación taxonómica de mecanismos de persistencia para sistemas orientados a objetos.

A continuación, se presentan algunos de los trabajos relacionados con esta propuesta de tesis mas significativos de acuerdo a las necesidades de persistencia del estado de la interacción del usuario con los IIDM, estos trabajos entran en algunas de las clasificaciones planteadas en [Miranda et al., 2006], de acuerdo a sus características y propósitos.

2.1 Lenguajes Orientados a Objetos -Biblioteca Estándar

Proveen un conjunto de librerías e Interfazes de Programación de Aplicaciones (API por sus siglas en inglés) que proporcionan mecanismos para el control de la persistencia de los objetos de una aplicación orientada a objetos.

2.1.1 Framework Flexible de Persistencia para Middleware Orientado a Objetos

En este trabajo se resalta la capacidad que deben tener los ambientes middleware para almacenar el estado de los objetos persistente de una forma eficaz y flexible [Weske & Kuropka, 2001]. Los autores proponen un framework flexible, extensible e independiente del lenguaje que permita implementar servicios de persistencia para resaltar la confiabilidad y tolerancia a fallos de middlewares orientados a objetos.

En caso de perder el estado de los objetos persistentes por fallas en los servidores, el servicio de persistencia es el responsable de recuperar la consistencia del estado de los objetos después de que el sistema sea restaurado. La recuperación de fallos es transparente para el usuario permitiendo al desarrollador preocuparse solo por los requerimientos de la aplicación y dejando al servicio de persistencia los asuntos relacionados con el almacenaje del estado de los objetos persistentes.

El framework propuesto esta compuesto por una serie de interfaces las cuales son definidas por el Interface Definition Lenguage (IDL), es importante señalar que se trata de un framework que permite definir servicios de persistencia, no es una implementación como tal. Su independencia del lenguaje permite que todos los lenguajes soportados por el middleware puedan ser utilizados para implementar servicios de persistencia. Soporta diferentes tipos de repositorios (ej. sistemas de archivos, sistemas relacional de base de datos y sistema orientado a objetos de base de datos.) y permite representar los datos como flujos de binarios o esquemas complejos de mapeo.

Podemos mencionar que la propuesta de este framework se enfoca en la implementación de servicios de persistencia en sistemas distribuidos, con la restricción de solo asegurar la tolerancia a fallos y la confiabilidad en caso de pérdida de datos volátiles causada por fallos en los servidores. Si bien libera al desarrollador de los detalles de persistencia, éste tiene que tiene que heredar los objetos de las clases que el framework maneja para el servicio de persistencia.

2.1.2 Enterprise Java Beans

Enterprise Java Beans (EJB) es el componente principal del grupo de especificaciones JEE[1] definidas por Sun, a través de EJB se define una estructura (“Framework”) en la cual es posible manipular e interactuar con procesos e información que residen en diversos ambientes computacionales desde “MainFrames” hasta Bases de Datos. Estos componentes (EJB) contemplan diversas características esperadas de una aplicación empresarial como: Control de Transacciones, Seguridad, Threading, Propagación de Transacciones, y otras funcionalidades más.

Dichos componentes son ejecutados dentro de un Application Server, y actualmente existen tres tipos principales de EJB.

- *Session EJB*.- Un Session EJB permite realizar cierta lógica solicitada por un cliente ya sea un JSP/Servlet, Applet e inclusive otro EJB. Existen dos tipos de Session EJB’s:

[1] http://java.sun.com/javaee/technologies/javaee5.jsp

  - *Stateless EJB*.- Este tipo de EJB como su nombre lo indica no mantiene estado (Stateless) en el “EJB Container”, estos EJB son utilizados para realizar tareas rutinarias que no requieren identificar o rastrear al cliente (JSP/Servlet) como: operaciones matemáticas complejas, búsquedas generales u otra tarea.
  - *Stateful EJB*.- A diferencia de Stateless (Session) EJB este tipo de EJB permiten mantener la sesión del cliente en el “EJB Container”, de esta manera el cliente (JSP/Servlet/Applet) puede trabajar con cierto juego de datos especifico administrado por el "EJB Container", la aplicación ideal para este tipo de EJB es un componente de compra ("Shopping Cart") el cual puede identificar artículos e información personal del cliente a través de un lapso de tiempo extenso (Session).

- *Entity EJB*.- Un Entity Bean a diferencia de un Session Bean trabaja en conjunción con un depósito de información (generalmente una Base de Datos), esto permite que el EJB manipule información residente en sistemas ajenos al "EJB Container"; en un Stateful (Session) EJB si ocurre una falla en el “EJB Container” se pierde toda información, mientras si se utiliza un Entity EJB aún permanecerá esta información en el sistema aledaño. En otras palabras, un Entity EJB manipula una copia de información que reside en otro sistema. Su objetivo es encapsular los objetos del lado del servidor que almacena los datos. Los EJBs de entidad presentan la característica fundamental de la persistencia:

  - *Bean Managed Persistence.*- Este tipo de Entity Bean requiere que la lógica necesaria para accesar el sistema de información (Base de datos) sea definida manualmente, por lo general esta lógica se encuentra en JDBC y define como y cuando debe ser accesada, actualizada, guardada la información entre el EJB y la Base de Datos.
  - *Container Managed Persistence.*- Este Entity Bean como su nombre lo indica es manejado por el "EJB Container", a diferencia de un Bean Managed EJB donde se requiere definir lógica de acceso manualmente, en un Container Managed EJB el "EJB Container" genera toda lógica de acceso para el sistema de información (Base de Datos). Aparentemente el Bean Managed EJB mencionado anteriormente, no tiene mucha razón de ser al existir un Container Managed EJB, sin embargo, hay casos donde es empleada lógica de acceso sumamente compleja la cual no es posible generar a través del "EJB Container".

Estructura de un EJB

Un EJB esta compuesto de 4 partes (con la excepción de Messaging EJBs) las cuales son:

- Enterprise Bean Class
- Home Interface
- Remote Interface
- Deployment Descriptor

Enterprise Bean Class

Esta clase es el componente medular de un EJB, en esta clase se encuentran definidas toda las funciones utilizadas por un EJB, ya sean procedimientos rutinarios (operaciones matemáticas) o con lógica hacia bases de datos (JDBC). En esta clase reside todo el código funcional que realiza operaciones en Java, desde la activación del EJB hasta su destrucción incluyendo las funciones de negocios para el que éste fue diseñado.

Home Interface

Esta Interfaz (como cualquier otra en Java) solo define un esqueleto para funciones utilizadas en el “Enterprise Bean Class”, las funciones que deben ser declaradas en un “Home Interface” son aquellas necesarias para la creación-activación de un EJB, algunas de estas son: create,passivate, activate. Una de las razones de esta interfaz se debe al diseño distribuido de EJB (RMI), cuando un cliente(JSP/Servlet/Applet) requiere interactuar con un EJB éste no se comunica directamente con el “Enterprise Bean Class”, sino con el “Home Interface” del EJB en cuestión.

Es de suma importancia recordar que esta interfaz (como cualquier otra en Java) no define ningún tipo de lógica o código fuente, solo es una declaración o “esqueleto” de funciones, la lógica o código se encuentra dentro del “Enterprise Bean Class”. Finalmente cabe enfatizar que el “Home Interfaz” solo contiene las declaraciones para funciones necesarias en la creación-activación de EJBs, las declaraciones de funciones de negocios se encuentran en otra interfaz mencionada a continuación.

Remote Interface

Esta interfaz contiene las declaraciones para funciones de negocios definidas en el “Enterprise Bean Class”, y al igual que toda interfaz, solo contiene el “esqueleto” de las funciones. El número de declaraciones dependerá de las funciones declaradas en el “Enterprise Bean Class”.

Deployment Descriptor

Un “Deployment Descriptor” es un archivo en XML que cumple varias funciones. La primera es parametrizar el código Java del “Enterprise Bean Class”, esto es, definir parámetros que varían dependiendo del ambiente; por lo general todo EJB contiene algún tipo de lógica que dependerá del ambiente de ejecución como: nombres de Bases de Datos, Servidores de Paginas, Usuarios privilegiados u otros detalles. Es a través del “Deployment Descriptor” que estos parámetros pueden ser modificados sin la necesidad de modificar el código fuente, inclusive para aquellos EJB adquiridos de terceros los cuales no distribuyen su código fuente es la única manera de “ajustar” este tipo de parámetros.

Además de lo anterior, el “Deployment Descriptor” también indica al “EJB Container”: el tipo de EJB (“Session,Entity,Messaging”), el esquema de seguridad que posee el EJB, en caso de ser un "Container Managed Entity EJB" las funciones para las que se generará lógica, y otras funcionalidades más.

La especificación Enterprise Java Beans (EBJ) permite al desarrollador de los objetos IIDM abstraerse de los detalles técnicos de persistencia para los campos que contiene el objeto y concentrarse en el objetivo educativo del IIDM que elabora, pero no deslinda al desarrollador tener conocimientos técnicos computacionales para utilizar el servicio de persistencia en sus objetos. Esto no siempre será posible en la medida en que los desarrolladores tengan un perfil académico en pedagogía y conocimientos insuficientes en computación para realizar la tarea de aplicar persistencia a sus objetos aplicando Enterprise Java Beans.

2.1.3 Un framework con base en el diseño de patrones para proveer persistencia a lenguajes de programación orientados a objetos

Este framework describe un enfoque para proveer persistencia en lenguajes de programación orientados a objetos sin tener que modificar el sistema run-time o el propio lenguaje [Kienzle & Romanovsky, 2000]. Aplicando patrones de diseño, como Serializer, Factory Method y Strategy. El framework claramente separa el control de la persistencia del control de almacenaje. Una jerarquía de diferentes tipos de almacenaje, usualmente en diferentes dominios de aplicaciones, es utilizada. El framework no depende de ningún tipo de requerimiento especial del lenguaje de programación por lo que puede ser implementado sobre cualquier lenguaje orientado a objetos.

Investigaciones sobre lenguajes de programación persistente y sistemas en los años recientes han mostrado que esta tecnología es utilizada para desarrollar software complejo en muchos dominios de problemas. La persistencia

es utilizada para que los datos de un programa en ejecución sean salvados y puedan ser utilizados en posteriores ejecuciones. La durabilidad en transacciones usualmente se logra usando técnicas de persistencia. Cómo los datos son guardados y qué tipo de medios de almacenaje son utilizados para estos propósitos depende de las demandas de la aplicación y pueden variar de un sistema a otro. Desafortunadamente los lenguajes de programación orientados a objetos mas utilizados aun no ofertan soporte para la persistencia.

Existen muchos esquemas que pueden ser utilizados para proveer persistencia. La mas sofisticada y anhelada forma de persistencia es "Persistencia Ortogonal". Esto es el soporte de persistencia para todos los datos independientemente del tipo de datos del que se trate. En un lenguaje de programación que provee persistencia ortogonal, los datos persistentes son creados y utilizados de la misma manera que los datos no persistentes, recuperar y guardar los valores de estos datos no altera su semántica; y el proceso es transparente para el programador de la aplicación. Si los datos son o no persistentes es determinado usando la técnica llamada persistencia por accesibilidad. El soporte de persistencia designa un objeto a un manejador de persistencia y provee de funcionalidad para localizarlo. Cualquier objeto es accesible por el manejador de persistencia, para instancias siguientes del objeto automáticamente son persistentes.

La persistencia ortogonal es extremadamente difícil de alcanzar, todas sus implementaciones han tenido que modificar levemente el lenguaje de programación y/o modificar el run-time del sistema.

Los autores de esta propuesta identifican los siguientes problemas de la persistencia ortogonal:

- Requiere que ambos, datos y tipos de datos puedan tener un ciclo de vida indefinido. Si se desarrolla una aplicación persistente, serán necesarias las equivalencias estructurales y verificación de tipos dinámicos para relacionar el sistema con el medio de almacenamiento. Cuando se introduce la persistencia ortogonal, la compatibilidad de tipos con una ejecución se extiende a la compatibilidad de tipos a través de diferentes ejecuciones. Esto puede ocasionar conflictos al escribir las reglas del lenguaje de programación.
- Lenguajes de programación tales como java, c# permiten el uso de variables estáticas dentro de clases o como variables globales independientes. Es posible que el programador use tales variables estáticas para enlazar objetos, por ejemplo para instanciar una tabla estática donde los valores de las llaves estén enlazados con algún otro valor. Ahora si los valores de las llaves se hacen persistentes, la tabla por tanto también se hace persistente o bien los valores de las llaves se vuelven inútiles. Esto puede implicar que sea difícil de proveer persistencia automática para las variables estáticas sin romper con la persistencia ortogonal.
- La persistencia ortogonal también requiere que sean contemplados tipos tales como task types/threads y valores de punteros a subprogramas, lo cual puede tener severos problemas de implementación.
- Un programa pudo haberse desarrollado y posteriormente haber cambiado la definición de los tipos y clases, pero aun trata de trabajar con los valores anteriormente guardados en previas ejecuciones. Para poder realizar este tipo de cambios en la definición de tipos y clases se debe llevar un control de versiones con revisiones dinámicas. El problema se pude complicar aun más cuando es considerada la herencia.
- Otro problema importante es la gestión del almacenaje, datos persistentes que ya no serán utilizados deben ser eliminados, para el medio de almacenaje dejarlos ahí implica perdida de la capacidad de almacenaje. Esto básicamente requiere alguna forma automática de recolección de basura, por lo menos para todos los datos persistentes.

Los autores proponen que el soporte de persistencia en un lenguaje de programación orientado a objetos debe alcanzar los siguientes objetivos:

- Clara separación de responsabilidades: El objeto persistente no debe tener conocimiento acerca de los dispositivos de almacenamiento o acerca del formato de los datos que son usados cuando se escribe el estado del objeto en un dispositivo de almacenamiento y viceversa.
- Modularidad y Extensibilidad: Debe ser directo al momento de definir un nuevo objeto a ser persistente o agregar un nuevo dispositivo de almacenamiento.
- Gestión Segura del Almacenamiento: La perdida de espacio en el dispositivo de almacenamiento debe ser prevenido.

Para ayudar a los programadores a alcanzar estos objetivos, los autores del documento proponen un framework general para proveer persistencia de los objetos. El framework puede ser utilizado por dos tipos de programadores: programadores de soporte para la persistencia y programadores de aplicaciones.

Los programadores de soporte agregarán soporte para nuevos dispositivos de almacenamiento para el framework usando técnicas de programación orientadas a objetos.

Los programadores de aplicaciones podrían utilizar el framework para declarar objetos persistentes, cuando declaren un nuevo objeto persistente, el programador debe especificar donde será almacenado el estado del objeto eligiendo entre las implementaciones existentes de los dispositivos de almacenamiento. Las operaciones definidas para el objeto persistente permiten que el programador de aplicaciones grabe o recupere el estado del objeto en cualquier momento.

Mucha de la estructura del framework se basa en los bien documentados patrones de diseño.

Aplicación de Patrones de Diseño:

- Clasificación de Tipos de Almacenaje: Una jerarquia de clases abstractas es la forma más natural de representar la estructura de los tipos de almacenaje.
- Serialización de Objetos: La serialización provee mecanismos eficientes para pasar de objetos en forma de flujos de bytes a estructuras de datos de cualquier forma, así como crear objetos a partir de tales estructuras.
- Creación de Objetos Persistentes: Al momento de crear una instancia del objeto persistente, el programador de aplicaciones debe especificar sobre que tipo de almacenaje quiere almacenar el estado del objeto. El objeto entonces crea una instancia de la clase correspondiente y esta establece la conexión con el dispositivo de almacenamiento.
- Identificación de Objetos Persistentes Debe haber una única manera de identificar los objetos persistentes. Parámetros de almacenamiento en las clases abstractas de los tipos de almacenamiento permiten identificar la localización del dispositivo de almacenamiento y pueden ser utilizados para identificar los objetos.
- Gestión de Almacenamiento: Una vez que el estado del objeto se almacena en un medio no volátil, este permanece ahí por tiempo indefinido, la única manera de liberar el espacio asociado con ese objeto es señalar de forma explicita la eliminación del objeto. Esta situación pude dejar objetos permanentemente si el programador de aplicaciones olvida especificar los parámetros que le permiten identificar al objeto en subsecuentes ejecuciones del sistema. Una solución simple a este problema es proporcionar una cierta clase de directorio persistente confiable. Los parámetros de cada uno de los objetos persistentes creados automáticamente se almacenarían en el directorio, por tanto el programador de aplicaciones puede consultar la lista de objetos persistentes existentes y determinar cuales de ellos necesita de los que quiere eliminar.

Este framework podría ser de utilidad para lograr los objetivos propuestos, puesto que permite desarrollar servicios de persistencia sobre patrones de diseño, los cuales están ampliamente documentados permitiendo tener un mejor entendimiento de la solución que el framework propone y aprovechar las ventajas de utilizar patrones de diseño como la flexibilidad o modularidad. El framework al estar basado sobre patrones de diseño utiliza la serialización para almacenar objetos persistentes, este método es muy lento, poco seguro y ocupa mucho espacio en disco, lo que podría representar problemas de rendimiento al aplicarlo.

2.2 Lenguajes Orientados a Objetos -Lenguajes Persistentes

Son lenguajes de programación extendidos con construcciones para tratar con datos persistentes.

2.2.1 Programación de Objetos con Persistencia por Contrato

Contracted Persistent Object Programming (*CPOP*) esta basado en la persistencia ortogonal y Diseño por Contrato (DBC). CPOP tiene el propósito de unificar el manejo de objetos transitorios y persistentes proporcionando el mismo mecanismo para ambos [Balzer, 2005].

El diseño por contrato es una metodología que permite la especificación mutua de obligaciones entre el cliente y la clase proveedora. Precondiciones son condiciones que el cliente tiene que satisfacer para que la clase proveedora realice las operaciones requeridas satisfactoriamente. Poscondiciones son garantías de la clase proveedora en la calidad de la ejecución de la operación.

Adicionalmente los contratos capturan condiciones de consistencia. En Eiffel un lenguaje de programación orientado a objetos basado en DBC, los contratos son monitoreados en tiempo de ejecución, cualquier violación de contrato señala una violación de restricción consecuentemente resulta en el lanzamiento de una excepción.

Un lenguaje de programación persistente cumple con los siguientes principios de persistencia ortogonal:

- *Ortogonalidad*: Todos los datos deben tener el mismo principio de persistencia independientemente del tamaño, tipo o alguna otra propiedad.
- *Transitividad*: Siempre que un dato es almacenado cada vez que es utilizado los datos tienen que ser almacenados correctamente también. Por otra parte esto asegura que los objetos almacenados puedan ser correctamente interpretados al ser recuperados, puesto que el principio se aplica a los objetos y a sus clases también.
- *Independencia de Persistencia*: La semántica de los lenguajes de programación no debe cambiar cuando se introduce la persistencia a excepción del hecho de que los datos pueden sobrevivir a una sola ejecución de programa. Promueve la reutilización de código; código escrito para objetos transitorios puede ser utilizado para objetos persistentes y viceversa.

Definición de CPOP: Cumple con los principios de persistencia ortogonal y utiliza Diseño por Contrato como su fuerza impulsora. CPOP además unifica el manejo de objetos transitorios y persistentes proporcionando el mismo mecanismo para ambos.

Este esquema permite implementar persistencia en los objetos basados una modalidad del enfoque de persistencia ortogonal, lo cual no permite especificar que atributos serán persistentes ya que hace al objeto totalmente persistente, esto provoca un costo considerable en términos de rendimiento.

2.3 Acceso Directo a Bases de Datos -Objeto/Relacional

Permiten hacer representaciones para recuperar y almacenar el estado de los objetos en esquemas de tablas y registros en bases de datos relacionales,

2.3.1 Un Framework como Middleware para la Persistencia y Consulta de Objetos de Java

El framework propuesto se compone de dos subframeworks: de persistencia y de consultas [Alia et al., 2004].

Framework de persistencia

Considera dos tipos de objetos:

- Memory Instances (MI): representan los objetos de Java y contienen los datos a ser persistentes.
- Data Store Instances (DSI): representan los datos almacenados en el datastore como tablas, registros, etc.

Este framework se concentra en el enlace entre los DSI y MIs provee un manejo estructural entre los objetos persistentes y un datastore particularmente cuando se realizan operaciones de entrada/salida sobre los datos.

Framework de consultas

Permite realizar, optimizar y evaluar consultas sobre datastores heterogéneos y particularmente sobre el framework de persistencia de objetos. Las expresiones de consultas son independientes de cualquier lenguaje de consulta y pueden ser mapeadas sobre varios estándares de lenguajes de consulta.

Existen dos tipos de persistencia, transparente y ortogonal; la persistencia transparente crea una menor distinción entre los objetos persistentes y los transitorios, esto significa que los programadores tienen cierto grado de control sobre la persistencia como abrir o cerrar transacciones; la persistencia ortogonal supone que las propiedades de persistencia son independientes del tipo de objeto: el programador no especifica el objeto que será persistente y contempla que todos los objetos son potencialmente persistentes.

Implementar la persistencia ortogonal se ha definido como una tarea difícil, la persistencia transparente se ha implementado y utilizado ampliamente en la industria, esta tendencia ha sido ocasionada por el hecho de que la mayoría de los datos de las empresas están almacenados en bases de datos relacionales.

Las técnicas complejas de mapeo objeto-relacional están implementadas en el diseño del servidor de la aplicación, en este contexto la persistencia transparente es mas apropiada, muchos estándares para la persistencia transparente han sido desarrollados como por ejemplo CORBA Persistence State Service (COS PSS), Java Data Object (JDO) y EJB-CMP Entity Beans para acceder a objetos persistentes.

Este framework propone una solución para aplicaciones que implementan persistencia de objetos, independientemente si estas aplicaciones siguen el enfoque de persistencia transparente u ortogonal. Programar estas aplicaciones implica manejar instancias de almacenaje como por ejemplo tablas, registros de la base de datos e instancias de memoria como por ejemplo objetos de java, muchas veces los programadores manejan estos dos tipos de instancia, esto es, organizan las transferencias entre las instancias de memoria y las instancias de almacenaje acomodando formatos y tipos de datos que usualmente difieren entre si.

Las instancias de almacenaje pueden residir en bases de datos, sistemas de archivos o mainframes, estos dispositivos de almacenamiento son referidos como DataStores (DS), los cuales soportan transacciones y pueden potencialmente ser federados.

Los estándares de persistencia mencionados anteriormente son referidos como Memory Instances Manager (MIM).

Un requerimiento común es la consulta de objetos persistentes, esto implica proveer acceso asociado a estos objetos usualmente con expresiones de consulta como una forma opuesta al acceso directo de los objetos a partir de referencias.

Los autores proponen una solución para atacar el problema de persistencia independientemente de los estándares existentes de MIMs y enfoques de persistencia, con el objetivo de asegurar que este framework tenga usabilidad en el contexto de estos estándares y enfoques. El middleware debe ser adaptable y flexible en orden de seguir las implementaciones de los DS en los MIMs, desde el punto de vista de los DS es mayormente posible extender el middleware para incorporar diferentes tipos de DS, desde el punto de vista de los MIMs la interfaz expuesta por el middleware debe permitir la implementación de diferentes MIMs.

En particular el middleware framework propuesto permite que sea posible implementar ambas persistencias transparente y ortogonal.

Las propiedades de persistencia son transpuestas en los siguientes objetivos para el middleware presentado:

- Independencia del tipo de datastore (RDBMS, OODBMS, Directorios, flat files).
- Independencia del ciclo de vida en memoria del objeto.
- Transparencia en otros aspectos no funcionales tales como el control de concurrencia y control de consistencia.
- Independencia del lenguaje de consulta.

El framework se define como "un concepto reutilizable de un sistema o parte de un sistema que esta representado por un conjunto de clases abstractas y la forma en como sus instancias interactúan".

El enfoque del framework resulta en la definición de APIs para la persistencia de objetos en un DS, la recuperación de objetos desde un DS y la interacción con la capa MIM o la realización o evaluación de consultas de objetos.

Un aspecto benéfico del enfoque del framework es la reusabilidad del software, los subframeworks de persistencia y consultas pueden ser reutilizados para implementar diferentes estándares como EJB y JDO, reduciendo la producción de código redundante.

La principal característica del framework es la flexibilidad que proporciona para trabajar con diferentes estándares para el control de persistencia, así como también manejar diferentes tipos de sistemas de almacenaje, lo que permite que entre sistemas heterogéneos que implementan este framework pueda haber una interacción entre los objetos propios de cada sistema y controlar de manera transparente para el usuario la persistencia o enfoques de persistencia que se estén utilizando.

2.3.2 Freeze el Motor de las Comunicaciones en Internet (Ice)

Ice fue desarrollado por ZeroC y tiene como metas principales:

- Proveer una plataforma middleware orientada a objetos apropiada para ser usado en ambientes heterogéneos.
- Provee un conjunto completo de características que permiten desarrollar aplicaciones distribuidas para una amplia variedad de dominios.
- Elimina la complejidad innecesaria, haciendo que la plataforma sea fácil de aprender y usar.
- Provee una implementación que es eficiente en redes de banda ancha, uso de memoria y CPU.
- Provee una implementación que tiene seguridad incorporada, permitiendo su uso en redes públicas inseguras.

The Internet Communicaction Engine (Ice)

Ice es un plataforma middleware orientada a objetos [Michi Henning, 2006], fundamentalmente esto significa que Ice provee herramientas, APIs y bibliotecas para desarrollar aplicaciones cliente-servidor orientadas a objetos.

Las aplicaciones Ice son adecuadas para ser utilizadas en ambientes heterogéneos: clientes y servidores pueden ser desarrollados en diferentes lenguajes de programación, pueden correr en diferentes sistemas operativos, en diferente arquitectura y se pueden comunicar entre ellos usando una variedad de tecnologías de red. El código fuente de estas aplicaciones es portable e independiente del ambiente de desarrollo.

Ice incorpora un objeto para el servicio de persistencia, conocido como Freeze. Freeze permite que sea fácil almacenar el estado de un objeto en una base de datos, pudiendo definir el estado del objeto que será almacenado en Slice. El compilador de Freeze genera el código que permite almacenar y recuperar el estado del objeto de y hacia la base de datos. Por defecto Freeze utiliza Berkeley DB como base de dato. Sin embargo es posible utilizar otra base de datos diferente.

Slice (Specification Language for Ice)

Los objetos Ice son interfazes con un numero de operaciones. Las interfazes, operaciones y tipos de datos son intercambiados entre el cliente y el servidor, los cuales son definidos usando el lenguaje Slice.

Slice permite definir contratos cliente-servidor en el sentido de que es independiente de algún lenguaje de programación especifico tal como C++, Java o C#. La especificación de Slice es compilada para un lenguaje de programación específico.

- Servants: Los objetos Ice son entidades conceptuales que tienen un tipo de identidad e información para su localización. Sin embargo las peticiones de clientes deben terminar ejecutando código del lado del servidor, con este código escrito en un lenguaje de programación especifico y ejecutado en un procesador especifico.
- Servant locator: Es un objeto local que el desarrollador implementa para localizar servant para peticiones entrantes. La petición se redirección para ser atendida por el servant correspondiente.

El artefacto del lado del servidor que provee comportamiento para las operaciones invocadas es conocido como servant. Un servant esta fundamentado por uno o mas objetos Ice. En la practica, un servant es una simple instancia de una clase que es escrita por los desarrolladores del servidor y es registrada en tiempo de ejecución del lado del servidor como servant para uno o mas objetos Ice. Los métodos de la clase corresponden con las operaciones sobre las interfazes de objetos Ice y provee el comportamiento para las operaciones.

Freeze representa un conjunto de servicios de persistencia:

- *The Freeze Evictor:* Es una implementación altamente escalable de un Ice servant locator que provee persistencia automática y desalojo de los objetos Ice con un mínimo de código.
- *The Freeze Map:* Es un contenedor genérico asociativo. Las aplicaciones interactúan con un Freeze Map igual que cualquier otro contenedor asociativo, excepto por que las llaves y valores del Freeze map son persistentes. Una vez que se han definido los datos persistentes en Slice, Freeze administra los detalles relacionados con la persistencia.

El servicio de persistencia que ofrece Freeze se apega en buena medida a los requerimientos de persistencia que los objetos IIDM necesitan implementar. Debido a la facilidad para especificar el estado o los atributos de un objeto, el cual debe permanecer después de finalizar la ejecución de la aplicación que lo instancia con el menor código posible. Así como también, deslinda al desarrollador del objeto de los detalles para manejar la persistencia, una desventaja para implementar Freeze podría ser la dependencia hacia con el lenguaje de especificación de Ice, ya que Freeze solo reconoce este lenguaje para especificar los atributos o estados del objeto que persistirán.

2.3.3 Almacenamiento para Estado de Objetos de Aprendizaje definido por el Autor

En el trabajo descrito en [Kassahun et al., 2006]se presenta una propuesta para el almacenamiento, recuperación y compartición del estado de un objeto de aprendizaje definidor por el autor del mismo. Se presenta un LMS con base en SCORM, donde los objetos de aprendizaje directamente realizan transacciones al sistema de almacenamiento a través de lenguje de consulta estructurado (SQL por sus sigla en inglés).

El intercambio de información involucra una extención en el modelo de datos de SCORM, así como un componente que interprete las peticiones y regrese los resultados al objeto de aprendizaje (DBLink).

2.3.4 Author-Defined Storage in the Next Generation Learning Management Systems

En el articulo [Sessink et al., 2003] se mencionan algunas de las características que debe poseer un LMS como respuesta a las tendencias en el aprendizaje electrónico, algunas de estas cualidades son la adaptabilidad, recuperación del estado e información histórica, comparación de resultados contra investigaciones pedagógicas, compartición de bases de datos entre otros. Varias de estas propiedades han sido estandarizadas por SCORM con limitantes significativas en relación con la persistencia de estado para los objeto de aprendizaje como la cantidad de datos que pueden ser almacenados y el tipo de almacenamiento que implementa.

Capítulo 3

Modelo de Referencia para Objetos de Aprendizaje (SCORM)

3.1 Introducción

Con el propósito de proveer mecanismos necesarios para obtener un aprendizaje de buena calidad, que permitan solventar las necesidades de aprendizaje de tal forma que sea rentable y pueda ser utilizado en cualquier momento y en cualquier lugar, el Departamento de Defensa de los Estados Unidos (DOD) y el Departamento de Ciencia y Tecnología Política de la Casa Blanca (White House Office of Science and Technology Policy OSTP) lanzan la iniciativa Advance Distributed Learning (ADL) a finales de 1997, para acelerar el desarrollo a gran escala del software de aprendizaje de una forma dinámica y rentable con el objetivo de estimular el mercado para estos productos.

Para alcanzar estos objetivos, era importante fomentar el desarrollo de contenido de aprendizaje que pudiera ser reutilizable "Objetos Instruccionales" dentro de un esquema de trabajo común basado en la computación y mayormente en un marco de aprendizaje basado en Web. Por lo que se definió el Modelo de Referencia para Objetos de Aprendizaje (Sharable Content Reference Model, SCORM por sus siglas en ingles) que permite al ADL contar con este marco de trabajo. Dentro de SCORM se describe un conjunto de reglas, especificaciones y estándares basados en diferentes esquemas de e-learning, proponiendo diferentes clases de servicios necesarios para solventar un problema de aprendizaje en particular, mostrando cómo estos servicios son agrupados y cómo los estándares que aplica son utilizados para crear contenido de aprendizaje accesible, ínteroperable, durable y reutilizable.

SCORM es un modelo que hace uso de las ventajas de la Web, debido a que la Web proporciona la mejor oportunidad para la reutilización y el acceso de contenido de aprendizaje, posee una infraestructura que constantemente esta en expansión y al trabajar sobre ambientes basados en Web, es posible tener Sistemas de Administración de Aprendizaje (Learning Management System LMS) que puedan cargar contenido creado por diferentes autores en diferentes herramientas e intercambiar información del contenido de aprendizaje, la posibilidad de que diferentes LMS creados por diferentes organizaciones puedan utilizar el mismo contenido de aprendizaje e intercambiar datos con este durante su ejecución y también permitir que diferentes LMSs puedan tener acceso a un repositorio de contenido de aprendizaje para su utilización.

3.2 Organización

SCORM esta conformado como se muestra en la figura 3.1 por un Modelo para Agrupación de Contenido (Content Aggregation Model, CAM), un Ambiente de Ejecución para objetos instruccionales (Run-Time Environment RTE), así como un Modelo de Secuencia y Navegación (Sequencing and Navigation SN) para las presentaciones dinámicas de contenido enfocado a las necesidades del estudiante. SCORM integra tecnologías desarrolladas por grupos tales como IMS[1], AICC[2], ARIADNE[3] e IEEE TSLC[4].

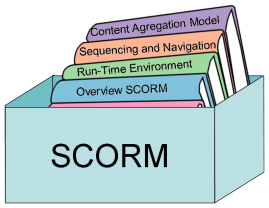


Figura 3.1: Conjunto de Libros de SCORM

3.2.1 Modelo de Agrupación de Contenido (CAM) de SCORM

El modelo de agrupación de contenido[5] permite especificar los tipos de contenido que son utilizados para formar la agrupación de contenido, especifica como se debe realizar el empaquetado de los contenidos para proveer un intercambio satisfactorio de información entre un sistema y otro, también permite describir el contenido usando metadatos. Estos metadatos además permiten o facilitan la búsqueda y descubrimiento de estos paquetes de aprendizaje y también este modelo permite especificar las reglas de secuencia del contenido para apoyar los requerimientos de diseño de la experiencia de aprendizaje. Un paquete de Contenido es un conjunto organizado de objetos de aprendizaje, el cual representa un curso, lección, modulo o simplemente una colección de objetos de aprendizaje relacionados entre si. Este paquete de contenido además incluye información adicional que describe como el LMS debe procesar el paquete y su contenido.

3.2.2 Ambiente de Ejecución (RTE) de SCORM

La especificación para el ambiente de ejecución de SCORM[6] describe los requerimientos que son impuestos por el LMS para asegurar las condiciones que permitan la interoperabilidad de contenido entre diferentes LMSs, como es, la estandarización del proceso de lanzamiento de contenido, la estandarización de métodos para una efectiva comunicación entre el contenido y los LMSs y una estandarización del modelo de datos utilizado para intercambiar información relacionada con la interacción del estudiante con el contenido. El RTE contempla los requerimientos necesarios para que los Sharable Content Objects (SCO) puedan utilizar un estándar de comunicación para que la información pueda ser transferida desde y hasta el LMS que se está utilizando. Es importante mencionar que un

[1] http://www.imsglobal.org/learningdesign/
[2] http://www.aicc.org/
[3] http://www.ariadne-eu.org
[4] http://ieeeltsc.org/
[5] SCORM 2004 3rd Edition Content Aggregation Model (CAM) Version 1.0, http://www.adlnet.gov/Downloads/DownloadPage.aspx?ID=237
[6] SCORM 2004 3rd Edition Run-Time Environment (RTE) Version 1.0, http://www.adlnet.gov/Downloads/DownloadPage.aspx?ID=237

Sharable Content Object (SCO) es una colección de uno o mas Assets que representan un objeto de aprendizaje capaz de ser lanzado por un LMS y tiene la capacidad para comunicarse e intercambiar información con el LMS que lo lanza utilizando el RTE de SCORM, y un Assets puede ser definido como una representación electrónica de medios, tal como puede ser, texto, imágenes, sonidos, páginas Web o cualquier otro componente de información que pueda ser cargado desde la Web, como un applet o una animación de Flash. Otra diferencia del Asset con el SCO, es que el Asset no requiere de comunicación con el LMS, como lo hace el SCO.

Para que el RTE pueda proveer interoperabilidad entre SCOs y LMSs, debe existir una forma común para lanzar objetos de contenido, una forma común para la comunicación entre SCOs y LMSs cuando el SCO esté cargado y un modelo de datos predefinido que permita el intercambio de información entre el LMS y el SCO durante la ejecución. Para estos propósitos el RTE define tres elementos: Launch, API y Data Model.

3.2.3 Secuencia y Navegación de SCORM

Los mecanismos de secuencia y navegación de SCORM[7] definen métodos para representar el comportamiento requerido para la experiencia de aprendizaje de tal forma que cualquier LMS pueda ordenar las actividades de una forma consistente, mas específicamente estos mecanismos describen las ramificaciones y el flujo de las actividades de aprendizaje en términos del Árbol de Actividades, con base en el resultado de la interacción del estudiante con el contenido de aprendizaje y la estrategia de secuencia propuesta por el autor del objeto de aprendizaje. Un Árbol de Actividades como se muestra en la figura 3.2 es una estructura conceptual de actividades de aprendizaje manejadas por el LMS para cada estudiante.

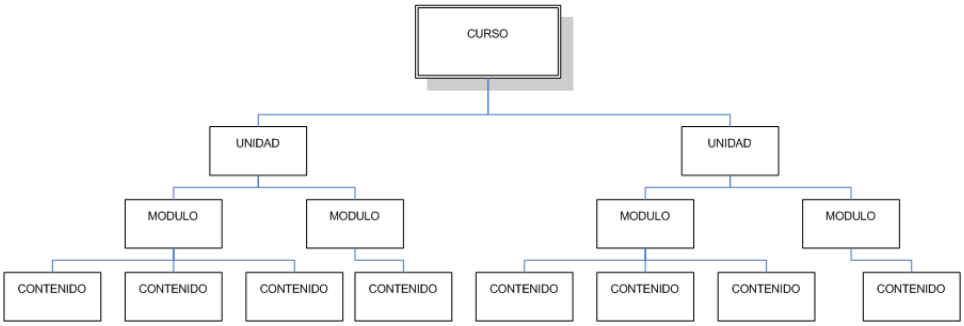


Figura 3.2: Árbol de Actividades

La secuencia y navegación describen cómo los eventos de navegación son disparados y procesados, resultando en la identificación de actividades de aprendizaje para ser cargadas. Cada actividad de aprendizaje identificada para ser cargada debe tener asociado un objeto de contenido de aprendizaje. Varios conceptos de CAM están relacionados con la secuencia y Navegación. EL CAM describe como construir las reglas de secuencia y estas reglas se representan dentro de un manifiesto en XML.

3.3 Ambiente de Ejecución de SCORM

3.3.1 Introducción

SCORM esta definido como un conjunto de libros. El Run-Time Environment es parte de este conjunto de libros, donde se describen los requerimientos del Learning Management System (LMS) para administrar el ambiente en

[7] SCORM 2004 3rd Edition Sequencing and Navigation (SN) Version 1.0, http://www.adlnet.gov/Downloads/DownloadPage.aspx?ID=237

tiempo de ejecución, como puede ser el proceso para lanzar contenido, la comunicación estandarizada entre el contenido y el LMS, así como un modelo estandarizado de datos, que se utilice para pasar información importante de la experiencia de usuario con el contenido. El RTE también cubre los requerimientos de la utilización de un Application Programming Interface (API) para los SCO y el Modelo de Dato del SCORM RTE[8].

3.3.2 Componentes

SCORM RTE cubre los aspectos esenciales que tratan sobre las responsabilidades del LMS para la secuencia de un SCO o Asset en tiempo de ejecución y permita a los SCO indicar o realizar peticiones de navegación. Estos aspectos los podemos clasificar de la siguiente forma:

- Administración del RTE: Incluye el lanzamiento de objetos de contenido (SCO y Assets), administración de la comunicación con el SCO y la administración del modelo de datos del RTE.
- Application Programming Interface (API): Incluye los requerimientos del API por parte del LMS, los requerimientos de comunicación de SCORM y la comunicación de errores.
- SCORM Run-Time Environment Data Model: Incluye la administración del modelo de datos y los requerimientos de comportamiento, así como los tipos de datos requeridos.

3.3.3 Descripción del Run-Time Environment

El RTE define un modelo que toma el control y realiza determinadas tareas cuando un objeto de contenido ha sido identificado para ser lanzado. Las responsabilidades o actividades que administra el RTE son:

- La entrega o presentación del objeto de contenido a los estudiantes por medio del navegador Web (launch)
- Si se trata de un SCO, es responsabilidad del RTE la forma en que se comunica este objeto de contenido con el LMS.
- El seguimiento de la información de un objeto de contenido y como el LMS debe manejar esta información.

Como se mencionaba anteriormente, SCORM fue desarrollado para habilitar el desarrollo de objetos de contenido que fueran reutilizables, interoperables a través de múltiples LMS. Para hacer esto posible, es necesario tener o contar con estándares o formas comunes para lanzar y administrar objetos de contenido, para comunicar estos objetos de contenido con el LMS y además contar con un lenguaje o vocabulario formando la base de la comunicación de una forma estandarizada. En la figura 3.3 podemos identificar estos tres aspectos y como están relacionados entre si.

[8] SCORM 2004 3rd Edition Run-Time Environment (RTE) Version 1.0, http://www.adlnet.gov/Downloads/DownloadPage.aspx?ID=237

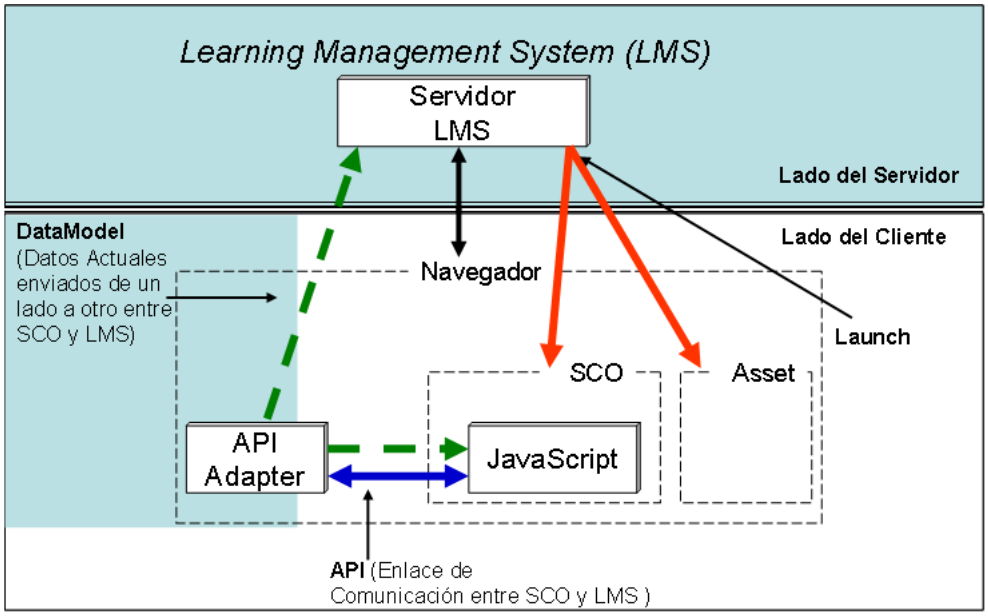


Figura 3.3: Componentes del SCORM Run-Time Environment

En la figura 3.3 además de ver como los componentes que conforman el Run-Time Environment están relacionados, también podemos ver en donde está ubicados estos componentes, como interactúan o se relacionan con los componentes definidos en SCORM, las flechas de color rojo indican que el LMS ubicado del lado del servidor lanza (launch) los objetos de contenido (SCOSs y Assets) en un navegador Web ubicado del lado del cliente. También podremos observar que por medio de sentencias en JavaScript y utilizando el API, representado con la flecha azul, el SCO se comunica con un Adaptador provisto por el LMS, el cual es descargado al acceder al Ambiente de Ejecución (RTE) y tiene la función de servir como intermediario entre un SCO y el LMS y facilitar el intercambio de información, donde se representa con la flecha verde los elementos del modelo de datos de SCORM.

3.3.4 Gestión

Cuando el estudiante interactúa con los objetos de contenido, el LMS evalúa el aprovechamiento del estudiante y las peticiones de navegación. Cuando el LMS identifica una actividad para ser presentada al estudiante, el LMS lanza el objeto de contenido asociado con esta actividad. En la figura 3.4 se muestra la estructura de contenido y cómo puede ser interpretada como un árbol de actividades, esta estructura está definida en el archivo de manifiesto imsmanifest.xml.

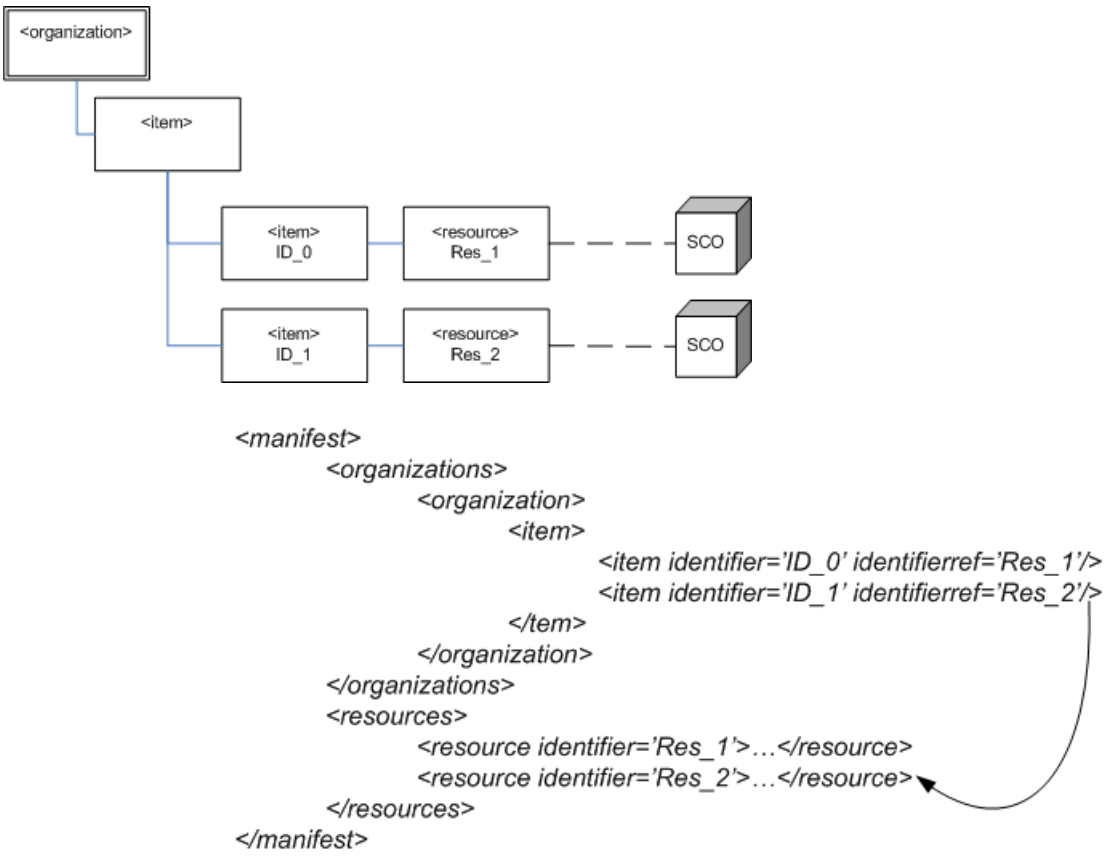


Figura 3.4: Estructura de Contenido

3.3.5 Modelo Temporal del Run-Time Environment

Existe una serie de términos que son necesarios tomar en cuenta para entender como funciona el traspaso de información durante la experiencia de aprendizaje del estudiante. Estos términos están definidos por el Institute of Electrical and Electronics Engineers (IEEE) 1484.11.1 Standard for Learning Technology - Data Model for Content Object Communication[9]:

- *Learner Attempt:* Se define como el intento o esfuerzo aplicado por un estudiante para lograr o completar los requerimientos de una actividad de aprendizaje dentro de un objeto de contenido. Este intento puede abarcar una o más sesiones por parte del estudiante y es capaz de suspender la actividad entre una sesión y otra.
- *Learner Session:* Es un periodo ininterrumpido de tiempo durante el cual un estudiante está accediendo a un objeto de contenido.
- *Communication Session:* Es una comunicación activa entre el objeto de contenido (SCO) y un API.
- *Login Session:* Es un periodo de tiempo durante el cual un estudiante inicia una sesión con el sistema hasta que termina la sesión con el sistema.

Estos términos son de importancia cuando se trata de administrar SCOs por parte del RTE.

[9] http://www.ieeeltsc.org/standards/1484-11-1-2004/

- Cuando se trata de Assets, el RTE se constituye solo de un learner attempts y un learner session para cada Assets lanzado.
- El learner Attempt inicia cuando una actividad ha sido identificada para ser presentada al estudiante, en este punto el estudiante es relacionado con un objeto de contenido, una vez que exista esta relación y el objeto de contenido ha sido lanzado en el navegador Web del estudiante, la Learner Session inicia.
- Si se trata de un SCO se iniciará la comunicación con el LMS y la Communication Session se iniciara y esta termina cuando el SCO termina la comunicación con el LMS.
- Un Learner Session termina cuando el SCO se deja en estado suspendido que significa que no termina todas las actividades, o en estado normal, que significa que cumple con todas las actividades definidas en el SCO.
- Para un SCO el Learner Attempt finaliza cuando una Learner Session finaliza en estado normal y para un Assets el Learner Attempt finaliza cuando el Asset es dejado por el estudiante.

En la figura 3.5 se muestra como estos cuatro conceptos están relacionados y como se comportan para un SCO.

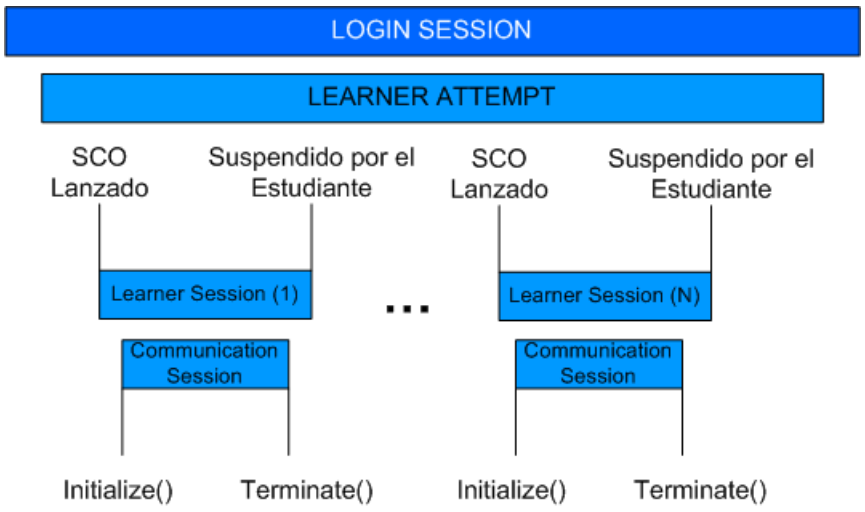


Figura 3.5: Relaciones del Modelo Temporal para un SCO

3.3.5.1 Manejo de Learner Attempts y Learner Sessions

Un Learner Attempts esta relacionado con requerimientos del LMS que establecen en tiempo de ejecución el modelo de datos que el SCO puede comunicar al LMS. Cuando inicia un Learner Attempts para un SCO, el LMS necesita crear e inicializar un nuevo conjunto de datos en tiempo de ejecución para que el SCO los pueda utilizar.

En dado caso de que el LMS inicie con un Learner Attempt previamente cargado y este contenga información que esté fuera del alcance de SCORM, el LMS puede actuar de las siguientes formas:

- Almacenar estos datos, para propósitos posteriores como puede ser reportes, auditorías, diagnósticos o estadísticas.
- Deshacerse de cualquier información relacionada con el Learner Attempt previamente iniciado, el LMS solo deberá guardar información de un Learner Attempt previo solo si fue suspendido.

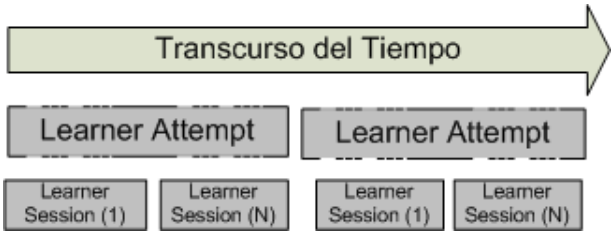


Figura 3.8: Sucesión de Learner Attempts, extendidos sobre varios Learner Sessions

Cabe mencionar que el LMS no especifica ningún requerimiento para la persistencia y el acceso a datos relacionados con un Learner Attempt previamente iniciado, pero en el caso de que ocurra una suspensión del Learner Session provoca que el Learner Attempt quede incompleto y el LMS debe asegurar los datos previos a la suspensión, para que en la siguiente Learner Session se puedan cargar las actividades en las cuales se quedó el estudiante. Las figuras (3.6,3.7,3.8) muestran diferentes relaciones entre un Learner Attempt and Learners Sessions. La figura 3.6 muestra una relación simple entre un Learner Attempt y un Learner Session.

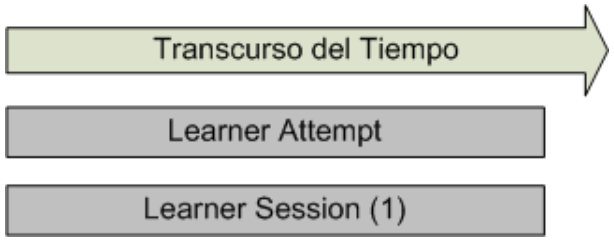


Figura 3.6: Relación Simple entre un Learner Attempt y un Learner Session

La figura 3.7 muestra un Learner Attempt que ha sido extendido sobre algunas Learner Sessions, las cuales han sido suspendidas, por lo tanto el Learner Attempt ha sido subsecuentemente inicializado hasta que una Learner Session termina normalmente.

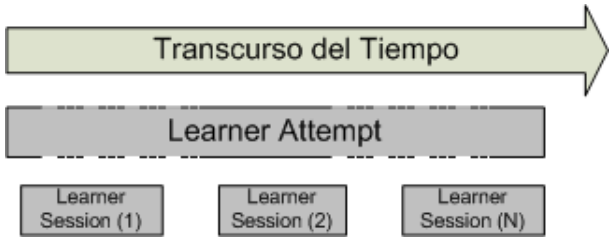


Figura 3.7: Learner Attrmpt Extendido sobre varios Learner Sessions

La figura 3.8 muestra una sucesión de Learner Attempts, y dentro de cada uno de estas Learner Attempts pude haber un numero de Learner Sessions.

3.3.6 Lanzamiento de Objetos de Aprendizaje

SCORM define en su Modelo de Contenido tres tipos de componentes:

- Assets

- SCOs
- Content Organizations

SCO y Assets están definidos como componentes que pueden ser lanzados. Los mecanismos de lanzamiento pueden variar dependiendo del tipo de contenido. El proceso de lanzamiento define la forma común o estandarizada por el LMS para lanzar objetos de aprendizaje en los navegadores Web de los estudiantes. Los procedimientos y responsabilidades para establecer la comunicación entre el objeto lanzado y el LMS varía dependiendo del tipo de objeto.

Es responsabilidad del LMS manejar la secuencia entre las actividades de aprendizaje basadas en comportamientos bien definidos y la evaluación de la información definida para la secuencia aplicada a las actividades, también es responsabilidad del LMS determinar cual actividad de aprendizaje debe ser presentada, dependiendo de los eventos de navegación, ya sea identificando la siguiente actividad de aprendizaje basándose en la secuencia definida en la estructura de contenido, identificando la selección por parte del usuario o basada en el rendimiento de aprendizaje obtenido de experiencias previa.

Una actividad de aprendizaje identificada para ser presentada, debe tener siempre asociado un objeto de aprendizaje y el LMS tiene la responsabilidad de lanzar este objeto. LMS utiliza el URL para localizar el objeto el cual esta definido en el Content Package, para la navegación o para reemplazar el objeto actualmente desplegado con el nuevo objeto de especificado en el URL de localización. En la figura 3.9 podemos ver un ejemplo de como esta definido dentro del manifiesto.

```
<manifest>
    <organizations>
        <organization>
            <item>
                <item identifier='ID_0' identifierref='Res_1'/>
                <item identifier='ID_1' identifierref='Res_2'/>
            </tem>
        </organization>
    </organizations>
    <resources>
        <resource identifier='Res_1'
            type='webcontent'
            adlcp:scormType='sco'
            href='Lesson1/Module1/sco.htm'
        </resource>
    </resources>
</manifest>
```

Figura 3.9: URL utilizado para el lanzamiento

El lanzamiento debe realizarse usando el protocolo de transferencia de hipertexto (HTTP) y presentado en un navegador Web. Para los Assets se requiere que sean lanzados de igual forma, utilizando el protocolo HTTP y puesto que se trata de un Asset, no necesita comunicación con el LMS, vía API y el modelo de datos.

Para un SCO se requiere que sean lanzados uno a la vez por cada estudiante y en una ventana dependiente, ya sea ventana emergente (pop-up window) o un frame hijo (child frame) de la ventana que provee la instancia del API como un Document Object Model (DOM)[10]. Esta instancia del API debe ser provista por el LMS.

[10]http://www.w3.org/TR/DOM-Level-3-Core/

Un SCO tiene la responsabilidad de buscar recursivamente en la ventana padre y/o en la jerarquía de la ventana que abre (opener window) hasta encontrar la instancia del API. Una vez que se haya encontrado la instancia del API, el SCO esta en posibilidades de inicializar la comunicación con el LMS. Para que el SCO pueda comunicarse utilizando la Instancia del API, este debe de comunicarse a través de las funciones que son requeridas como interfaz.

3.3.6.1 Retirar Objetos de Aprendizaje

Cuando un Learner Session termina, el último objeto lanzado es retirado, ya sea para lanzar el siguiente objeto de aprendizaje o para terminar el curso. Para retirar los objetos de aprendizaje es necesario que se genere un evento por parte del LMS o por parte del estudiante, una vez que el objeto de aprendizaje ha sido retirado, el LMS obtiene la información mas actual generada por la interacción del estudiante con el objeto de aprendizaje, con el fin de evaluar esta información para obtener la secuencia o el rumbo que tomara el curso. Si el objeto que se descarga es un Asset, el LMS asume cual fue la interacción que tuvo el estudiante con el objeto de aprendizaje, por el contrario si se trata de un SCO necesita comunicar esta información a través del modelo de datos de SCORM, este paso de información se realiza justo antes de finalizar el Learner Session.

Existen casos en los que la interacción con el SCO no es permitida cuando ya se ha retirado, para estos casos se sigue un comportamiento dependiendo del tipo de ventana en el cual el SCO ha sido lanzado.

1. Si la ventana en la cual se ha lanzado el SCO es una ventana de nivel alto (top-level window), entonces la ventana debe cerrarse después de invocar el método Terminate.
2. Si la ventana en la cual se lanza el SCO no es de nivel alto (top-level window), es decir, que tiene ventana padre, el SCO no puede actuar sobre la ventana padre o cualquier ventana en la jerarquía, por ejemplo un SCO no puede cerrar la ventana principal (top-window) a menos que esta sea la ventana en la cual se lanzó, en estos casos se puede optar por cargar contenido neutral mientras el LMS retira el objeto de aprendizaje.

3.3.7 Interfaz de Programación de Aplicaciones

3.3.7.1 Descripción

SCORM describe el IEEE 1484.11.2-2003 Standard for Learning Technology - ECMAScript Application Programming Interface for Content to Runtime Services Communication[11]. La API habilita la comunicación de datos entre los objetos de aprendizaje y el LMS a través de un conjunto de servicios del API utilizando JavaScript[12]. En este caso los objetos de aprendizaje son los SCO, puesto que son los objetos que requieren tener comunicación con el LMS. El utilizar el estándar del API de comunicación de SCORM, permite que los objetos de aprendizaje puedan trabajar en múltiples sistemas de administración o LMS al contar con una forma de comunicación común, lo que deriva en tener objetos de aprendizaje ínteroperables y por lo tanto reutilizarlos sin ningún problema de comunicación con el sistema que los administra. Los detalles de implementación son propios de cada LMS, es decir, SCORM, no menciona ningún requerimiento en especial para la comunicación de componentes del lado del servidor. El principal objetivo del API es proveer un mecanismo de comunicación que permita al SCO comunicarse con el LMS, todas las comunicaciones son iniciadas por el SCO.

Es importante mencionar los diferentes términos que SCORM maneja para el API y explicar las diferencias que existen entre estos términos. La figura 3.10 muestra la relación que existe entre estos términos.

[11] http://www.ieeeltsc.org/standards/1484-11-2-2003
[12] http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=33835

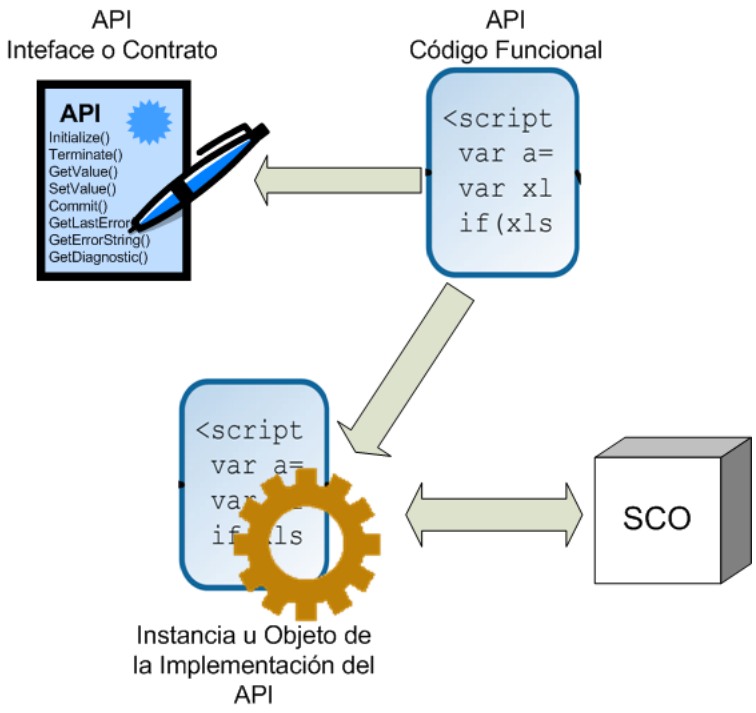


Figura 3.10: API, Instancia del API, Implementación del API

El termino API se define como el conjunto de funciones que el SCO confía en que estén disponibles para su utilización. Las funciones que define el API son las siguientes:

- Initialize()
- Terminate()
- GetValue()
- SetValue()
- Commit()
- GetLastError()
- GetErrorString()
- GetDiagnostic()

Una Implementación del API es una pieza funcional de software que implementa y expone las funciones del API. Para el SCO no importa como estas funciones han sido implementadas, puesto que solo se provee de semántica a las interfaces publicas que define el API. El LMS necesita proveer esta implementación del API y exponer la interfaz publica al cliente, que en este caso se trata del SCO.

Una instancia del API es una ejecución individual dentro de un contexto y mantiene un estado de la implementación del API. Representa la pieza de software con la que el SCO interactúa al realizar sus operaciones.

En la tabla 3.1 se muestran las tres categorías en las cuales se dividen las funciones de la Implementación del API.

Función	Descripción
Funciones de Sesión	Se utilizan para marcar el inicio y el fin de la sesión de comunicación (Communication Session) entre el SCO y el LMS a través de la Instancia del API.
Funciones para la transferencia de datos	Son utilizadas para el intercambio de la información contenida en los elementos del modelo de datos entre el SCO y el LMS a través de la Instancia del API.
Funciones de Soporte	Se utilizan para auxiliar en la comunicación para el manejo de errores entre el SCO y el LMS a través de la Instancia del API.

Tabla 3.1: Categorías de las funciones del API

Es importante mencionar que el API puede ser implementado como un Applet de Java, como se muestra en la figura 3.11, o también puede ser implementado en otros lenguajes de programación que tenga la posibilidad de ser cargado como un plug-in del navegador.

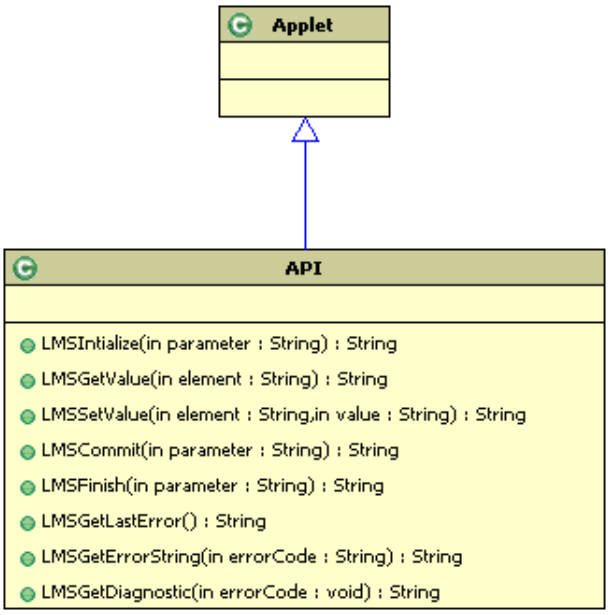


Figura 3.11: Implementación del API en un Applet

3.3.7.2 Funciones del API y su Sintaxis

Para que el SCO pueda iniciar comunicaciones con el LMS, primeramente debe encontrar en la instancia del API provista por el LMS, la forma de localizar la Instancia del API puede variar dependiendo de las necesidades de cada LMS y verse en la necesidad de que los desarrolladores de objetos de aprendizaje tengan que adaptar o cambiar sus objetos para trabajar con un determinado LMS, es por esta razón que se establecieron restricciones para ubicar la

Instancia del API en la jerarquía del Document Object Model (DOM) y el LMS además de proveer una Instancia del API, esta debe tener un nombre común para estandarizar la búsqueda por parte del SCO. A continuación se mencionan las reglas que el API debe cumplir:

- Todos los nombres de las funciones son sensibles a minúsculas y mayúsculas y deben estar escritos exactamente como se muestra.
- Los argumentos de las funciones también son sensibles a minúsculas y mayúsculas.
- Cada una de las llamadas a las funciones del API, con excepción de las Funciones de Soporte, establecen un código de error.
- Todos los parámetros y valores de retorno son pasados como cadenas de texto y deben ser compatibles con los tipos de datos y formatos descritos en el modelo de datos que utiliza el API para la comunicación.

3.3.7.3 Funciones de Sesión

El SCO utiliza las funciones de sesión para iniciar y finalizar la comunicación de información entre este y la Instancia del API.

Función: Initialize

Sintaxis: *return_value = Initialize(parameter)*

Descripción: Se utiliza para iniciar la sesión de comunicación.

Parámetros: Cadena vacía (“”)

Valor de Retorno: “true”, si la sesión de comunicación fue iniciada satisfactoriamente o “false”, si la sesión de comunicación no fue iniciada satisfactoriamente. Si encuentra algún error la API debe establecer el código de error especifico y el SCO puede llamar el método GetLastError() para determinar el tipo de error.

Función: Terminate

Sintaxis: *return_value = Terminate(parameter)*

Descripción: Se utiliza para finalizar la sesión de comunicación, cuando el SCO determina que ya no necesita comunicar con con el LMS. Esta función provoca que se guarden los datos que fueron generados a partir de la ultima llamada del método *Initialize(“”)* o *Commit(“”)*.

Parámetros: Cadena vacía (“”)

Valor de Retorno: “true”, si la sesión de comunicación fue finalizada satisfactoriamente o “false”, si la sesión de comunicación no fue finalizada satisfactoriamente. Si encuentra algún error la API debe establecer el código de error especifico y el SCO puede llamar el método GetLastError() para determinar el tipo de error.

3.3.7.4 Funciones para la Transferencia de Información

Un SCO utiliza estas funciones para almacenar y recuperar información dentro de la sesión de comunicación actual y transferirla en tiempo de ejecución desde y hacia el LMS, por lo regular con la finalidad de tomar decisiones de secuencia y navegación.

Función: getValue

Sintaxis: *return_value = getValue(parameter)*

Descripción: Esta función solicita o hace una petición de información al LMS. La información que el SCO puede solicitar al LMS puede ser:

- Valores de los elementos del modelo de datos soportado por el LMS.
- Versión del modelo de datos soportado por el LMS.
- Puede o no acceder a elementos específicos soportados por el modelo de datos.

Parámetros: Acepta un único parámetro el cual es la identificación completa de un elemento del modelo de datos.

Valor de Retorno: Este método puede retornar uno de dos valores, el cual debe ser de tipo cadena de texto.

- Una cadena de texto que contenga el valor asociado con el parámetro enviado.
- Si ocurre algún error, entonces la Instancia del API debe establecer un código de error con el valor especifico del error y retornar una cadena de texto vacía (“”), el SCO puede llamar al método GetLastError() para determinar el tipo de error. Esta función no puede retornar cadenas de texto vacías como datos validos, el SCO debe verificar si ha ocurrido algún error, si no ha ocurrido algún error, entonces si se acepta como dato valido retornado.

Función: setValue

Sintaxis: *return_value = setValue(parameter_1,parameter_2)*

Descripción: Esta función se utiliza para hacer una petición que transfiera el valor establecido en *parameter_2* y lo asigne al elemento del modelo de datos especificado en *parameter_1*, permitiendo al SCO enviar información al LMS para que la almacene. La Instancia del API se puede comportar de tal forma que cuando se llame esta función inmediatamente haga los datos persistentes lo cual se realiza del lado del servidor o puede almacenarnos un memoria temporal del lado del cliente.

Parámetros:

- *parameter_1*: Es la identificación completa del elemento del modelo de datos a ser establecido.
- *parameter_2*: Es el valor que se establecerá en elemento identificado por *parameter_1*. El valor de parameter_2 debe ser una cadena de texto convertible al tipo de datos definido por el modelo de datos identificado por *parameter_1*.

Valor de Retorno: Este método puede retornar uno de dos valores, el cual debe ser de tipo cadena de texto.

- "true", si se ha establecido el valor de *parameter_2* en el elemento de dato identificado por *parameter_1*.
- "false", regresa false si se encuentra un error al establecer el valor al elemento de dato identificado por *parameter_1*. El SCO puede llamar al método GetLastError() para determinar el tipo de error.

Función: Commit

Sintaxis: *return_value = Commit(parameter)*

Descripción: Este método realiza la petición para almacenar cualquier información que el SCO haya almacenado en la memoria temporal (cache) del API Instance desde la ultima invocación del método *Initialize("")* o *Commit("")*, lo que haya ocurrido mas recientemente. El LMS después de almacenar la información debe establecer el código de error a 0 (Sin Errores) y regresas *"true"*.

Si la Instancia del API no almacena temporalmente información, la invocación a la función *commit("")* debe regresar *"true"* y poner el código de error a 0 sin realizar ningún otro proceso. Este método es utilizado para garantizar que los elementos establecidos por la función *setValue()* serán persistentes con el propósito de reducir las probabilidades que la información se pierda por una interrupción en la sesión de comunicación, que finaliza anormalmente o que termine prematuramente antes de llamar a la función *Terminate("")*.

Parámetros: Cadena vacía ("")

Valor de Retorno: Este método puede retornar uno de dos valores, el cual debe ser de tipo cadena de texto.

- "true", si la información se ha almacenado satisfactoriamente.
- "false", si la información no se ha almacenado satisfactoriamente. El SCO puede llamar al método GetLastError() para determinar el tipo de error.

3.3.7.5 Funciones de Soporte

Estas funciones de soporte del API permiten al SCO determinar si un error fue encontrado y como manejar las condiciones de error encontradas. Con cada una de las funciones y solo esas funciones del API descritas anteriormente, puede ocurrir o presentarse una condición de error. Una vez que estas condiciones de error son encontradas el código de error es cambiado para indicar el error encontrado. Las invocaciones a las funciones de soporte no alteran el estado de error.

Función: getLastError

Sintaxis: *return_value = getLastError()*

Descripción: Este método hace una petición para obtener el código de error del estado de error actual de la Instancia del API.

Parámetros: Esta función no acepta ningún parámetro.

Valor de Retorno: Regresa el código de error del estado de error actual de la Instancia del API. El valor debe ser una cadena de texto convertible a un valor entero en el rango de 0 a 65536.

Función: GetErrorString

Sintaxis: *return_value = GetErrorString(parameter)*

Descripción: Esta función se utiliza para obtener información una descripción textual del estado actual de error.

Parámetros:

- *parameter*: Representación en cadena de texto del código de error (valor entero) correspondiente con el mensaje de error.

Valor de Retorno: Este método regresa un mensaje textual que contiene una descripción del código de error especificado en *parameter.* La descripción del Error tiene las siguientes características:

- El valor de retorno debe ser una cadena de texto con un máximo de longitud de 255 caracteres.
- SCORM no hace ninguna restricción sobre el contenido que debe tener el mensaje de error. La descripción textual del error es una especificación del LMS.
- Si el código de error enviado es desconocido por el LMS, se debe regresar una cadena de texto vacía (“”). Este es el único caso en que se debe regresar una cadena vacía.

Función: GetDiagnostic

Sintaxis: *return_value = GetDiagnostic(parameter)*

Descripción: Esta función permite al LMS definir información adicional sobre el diagnostico de la Instancia del API. Esta función tampoco altera el estado de error actual.

Parámetros:

- *parameter*: Un valor especifico implementado para el diagnostico. La longitud máxima del parámetro debe ser 255 caracteres. Este parámetro puede ser un código de error, aunque no esta limitado a aceptar solo códigos de errores. Si el parámetro es una cadena vacía (“”), se recomienda que se regrese información de diagnostico del ultimo error encontrado.

Valor de Retorno: Regresa una cadena de texto representando información de diagnostico. La longitud máxima de la cadena de texto es de 255 caracteres. Si el parámetro es desconocido para el LMS, debe regresar una cadena vacía(“”).

La IEEE define una categoría y rangos numéricos para los códigos de error:

Categoría del Código de Error	Rango del Código de Error
No Error	0
General Errors	100-199
Syntax Errors	200-299
RTS Errors	300-399
Data Model Errors	400-499
Implementation-defined Errors	1000-65535

Tabla 3.2: Categoría y rangos numéricos del Código de Errores

3.3.7.6 Modelo de Estados para la Sesión de Comunicación

LA IEEE define un modelo de estados conceptual por los cuales la Instancia del API transcurre durante su existencia. En la figura 3.12 se describen los estados que una Instancia del API transcurre para un SCO dado en tiempo de ejecución, los estados cambian como respuesta a eventos específicos. Cada uno de los estados definidos de la Instancia del API define cuales funciones puede invocar el SCO. Los estados por los cuales pasa la Instancia del API son:

- Not Initialized
- Running
- Terminated

Este modelo de estados es solo un modelo conceptual, no requiere implementación y es utilizado para ilustrar el comportamiento esperado de las funciones del API durante una típica sesión de comunicación.

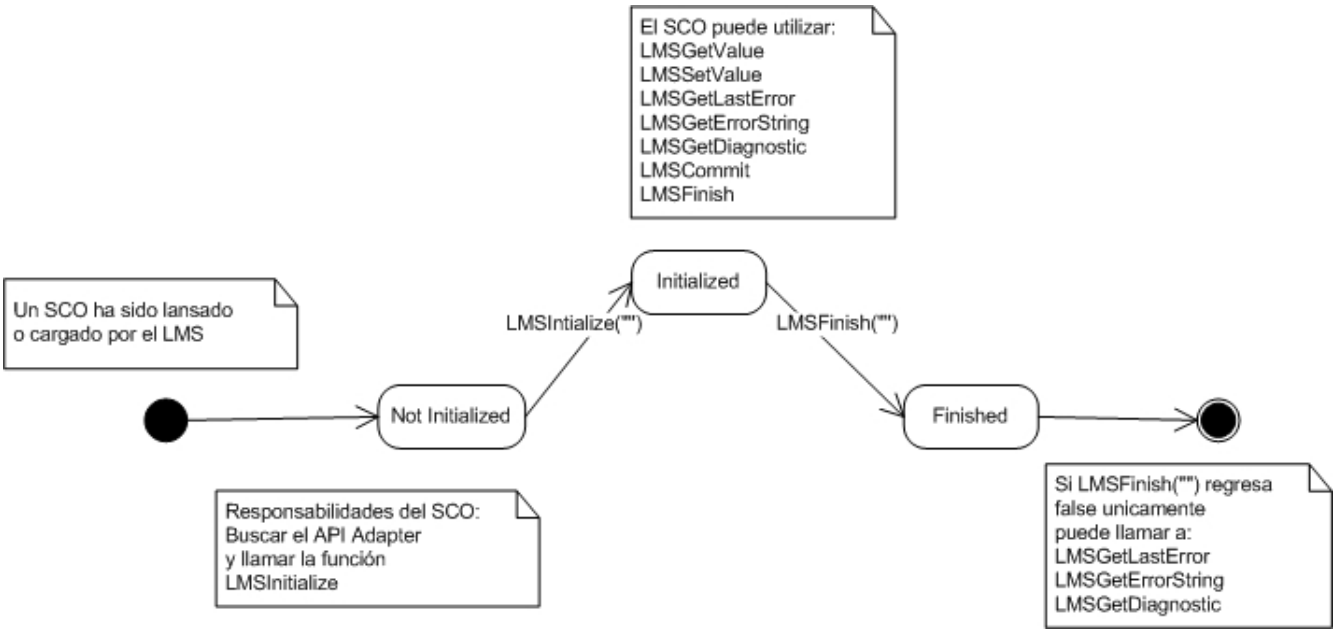


Figura 3.12: Transición Conceptual de los Estados de la Instancia del API

Not Initialized: Describe el estado actual de la comunicación entre el SCO lanzado y antes de que la función *Initialize("")* invocada por el SCO termine satisfactoriamente su procesamiento. En este estado, el SCO debe buscar la Instancia del API provista por el LMS. El SCO pude llamar a alguna de estas funciones:

- Initialize()
- GetLastError()
- GetErrorString()
- GetDiagnostic()

Running: Describe el estado actual de la comunicación una vez que la función *Initialize("")* invocada por el SCO termine satisfactoriamente su procesamiento y antes de llamar a la función *Terminate("")* invocada por el SCO. El SCO pude llamar a alguna de estas funciones:

- Terminate()
- GetValue()
- SetValue()
- Commit()
- GetLastError()
- GetErrorString()
- GetDiagnostic()

Terminated: Describe el estado actual de la comunicación una vez que la función *Terminate("")* ha finalizado satisfactoriamente su procesamiento. El SCO puede llamar a alguna de estas funciones:

- GetLastError()
- GetErrorString()
- GetDiagnostic()

3.3.8 Responsabilidades del LMS

SCORM requiere que el LMS provea una instancia del API como se define en el estándar de la IEEE y SCORM. La Instancia del API de ocultar al SCO los detalles de implementación del API. SCORM no pone ninguna restricción acerca de la infraestructura de comunicación de la Instancia del API.

3.3.8.1 Instancia del API

Como se menciona anteriormente, SCORM requiere que el LMS provea de una Instancia del API que implemente la funcionalidad requerida. Para que el SCO pueda utilizar la Instancia del API, el LMS tiene ciertos requerimientos que establecen donde y como proveer el acceso a la Instancia del API. Para proveer medios ínter operables para localizar la Instancia del API, esta debe ser accesible vía DOM como un objeto llamado *API_1484_11*. El LMS debe proveer la capacidad para que el SCO pueda acceder a la Instancia del API vía ECMAScript (JavaScript).

Para que el SCO pueda buscar la Instancia del API provista por el LMS, el LMS debe lanzar el SCO dentro de la jerarquía DOM del navegador. El LMS debe lanzar el SCO en una ventana del navegador ya sea ventana hijo o marco hijo del la ventana del LMS que contiene la Instancia del API.

Jerarquía de ventanas padre: En este caso, los SCO y Assets son lanzados en una estructura donde la instancia del API se encuentra en un conjunto de marcos HTML. En este caso, el LMS puede proveer la Instancia del API en alguna parte de la jerarquía de parents frames dentro del frameset como se muestra en la figura 3.13. El algoritmo

descrito por el estándar de la IEEE buscará en “jerarquía de ventanas padre”, hasta que la Instancia del API sea encontrada o hasta que se alcance el punto donde ya no haya otros parents frames.

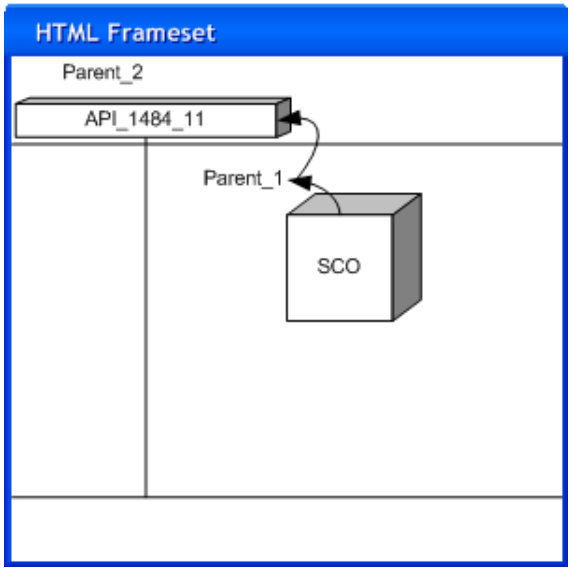


Figura 3.13: Localización del API: Chain of Parents

Ventana que abre: En este caso, los SCO y Assets son lanzados en una ventana nueva, por lo regular referida como ventana emergente. En este caso el LMS puede proveer la Instancia del API en la ventana que abrió la nueva ventana (ventana que abre). Como se muestra en la figura 3.14.El algoritmo descrito por el estándar de la IEEE determinara si el SCO fue lanzado por una ventana que abre, si es así, se buscara dentro de esta la Instancia del API.

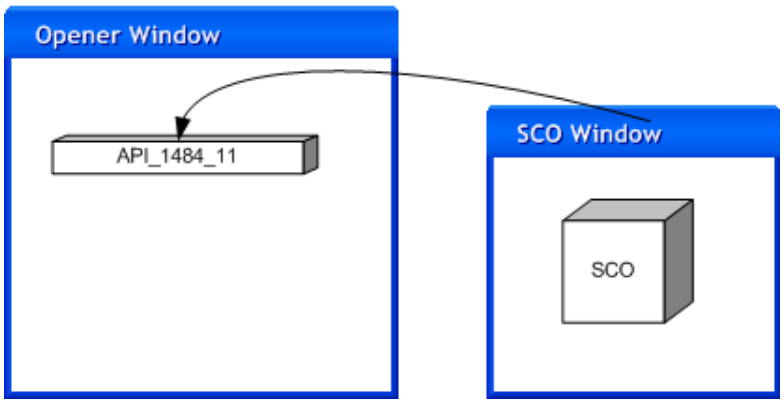


Figura 3.14: Localización del API: Opener

Cadena de Ventanas padre que abren: En este caso, los SCO y Assets son lanzados en una nueva ventana y el LMS a puesto la Instancia del API en un marco padre de la ventana que abre. Si el SCO ha sido lanzado desde una ventana que abre, será incluida para hacer la búsqueda de la Instancia del API, si no se encuentra la instancia del API en esta ventana, el algoritmo de búsqueda determina si la ventana que abre tiene padres. El algoritmo está diseñado para buscar en la jerarquía de ventanas padre hasta que se encuentre la Instancia del API o determine que no hay más ventanas padre.

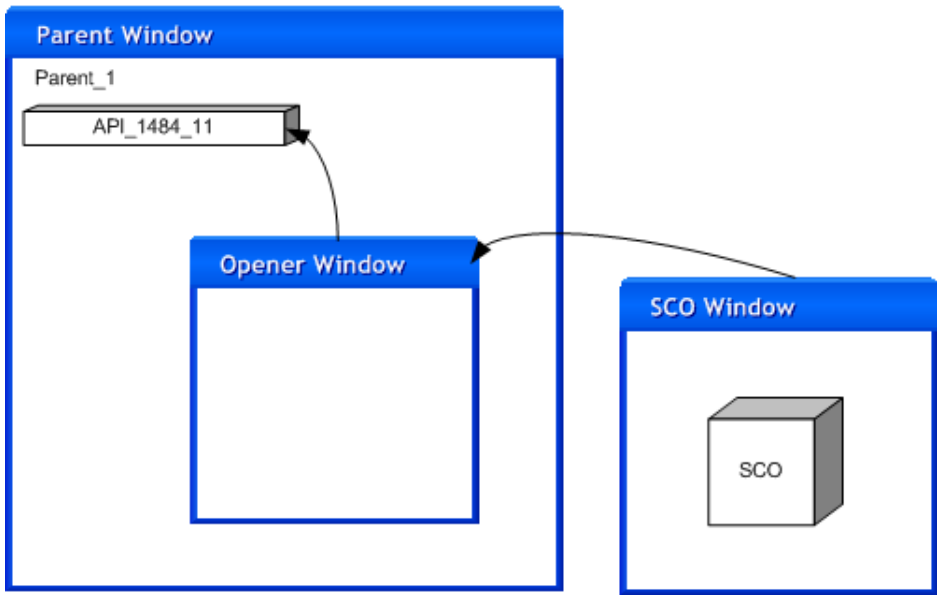


Figura 3.15: Localización del API: Chain of Parents of the Opener

3.3.9 Responsabilidades de SCO

Todos los SCO deben tener ciertas responsabilidades cuando se comunica a través del API. Los SCO deben buscar el API de una forma consiente, es por eso que existen las restricciones de en donde el LMS debe localizar la Instancia del API y el porque debe tener el nombre impuesto por el estándar, ya que los SCO buscaran en la jerarquía DOM el objeto con el nombre API_1484_11. De otra forma sería muy difícil proveer un mecanismo consistente de comunicación dentro del ambiente de ejecución.

3.3.9.1 Buscando la Instancia del API

Como mencionamos anteriormente, el SCO debe buscar la instancia del API provista por el LMS dentro de la jerarquía DOM del navegador Web. El SCO debe hacer la búsqueda en las siguientes localizaciones y en el orden en que aparecen, estas restricciones son impuestas por el estándar de la IEEE:

1. En la cadena de parents de la ventan actual, si existe algún parent, se debe buscar hasta que el tope de la cadena de ventanas parent haya sido alcanzado.
2. En la ventana opener, si existe alguna.
3. En la cadena de parents de la ventana opener, si existe alguna, hasta que el tope de la cadena de ventanas parent haya sido alcanzado.

Una vez que se ha encontrado la Instancia del API, debe haber una interacción mínima haciendo llamadas a las funciones *Initialize("")* y *Terminate("")*. El estándar de la IEEE provee un algoritmo en JavaScript para localizar la Instancia del API, de una forma consistente:

Código 1 Búsqueda en JavaScript de la Implementación del API

```
var nFindAPITries = 0;
var API = null;
var maxTries = 500;
var APIVersion = "";

//La función ScanForAPI() busca un objeto llamado API_1484_11 en la ventana que es pasada como argumento, si el
objeto es encontrado la función regresa su referencia. Esta función busca en un número máximo de parents de la
ventana actual. Si no se encuentra el objeto,la función regresa una valor nulo.

function ScanForAPI(win)
{
   while ((win.API_1484_11 == null) && (win.parent != null) && (win.parent != win))
   {
      nFindAPITries++;
      if (nFindAPITries >maxTries)
      {
         return null;
      }
      win = win.parent;
   }
   return win.API_1484_11;
}

//La función GetAPI() inicia el proceso de búsqueda de la Instancia del API, toma como parámetro una variable que
representa la ventana actual. Empieza a hacer la búsqueda en un orden especifico y se detiene cuando la Instancia
del API se ha encontrado, comienza a buscar en los parents de la ventana actual, si los tuviera, si no encuentra la
Instancia del API, después verifica si existen ventanas opener, si existen realiza la búsqueda en estas ventanas.

function GetAPI(win)
{
   if ((win.parent != null) && (win.parent != win))
   {
      API = ScanForAPI(win.parent);
   }
   if ((API == null) && (win.opener != null))
   {
      API = ScanForAPI(win.opener);
   }
   if (API != null)
   {
      APIVersion = API.version;
   }
}
```

3.3.10 Descripción del Modelo de Datos

El propósito de establecer un modelo común de datos para asegurarse que el conjunto de información definida para el SCO, pueda ser utilizada o se pueda seguir en otros LMS y este sepa como recibir la información, almacenarla y procesarla. El modelo de datos de SCORM esta basado en el estándar 1484.11.1 del Learner Technology - Data Model for Content Object Communication producido por el IEEE LTS CMI. 1484.11.1 es un estándar que define un modelo de datos utilizado para comunicar información de un SCO hacia un LMS. El modelo de datos contempla elementos que permiten guardar información relacionada con el estudiante, la interacción del estudiante con el SCO, información de los objetivos del curso, información relacionada con el estado del curso, si fue terminado satisfactoriamente y también sirve para tomar decisiones relacionadas con la secuencia del curso o hacer reportes sobre la interacción total del estudiante con el SCO.

3.3.10.1 Elementos del Modelo de Datos

Para identificar el modelo de datos que estamos utilizando, los nombres de los elementos descritos en el modelo de datos de SCORM deben empezar con las siglas *cmi*. Esto permite al LMS identificar que los elementos del modelo de datos son parte del estándar IEEE 1484.11.1 Data Model for Content Object Communication. Para cualquier otro modelo de datos desarrollado debe empezar con otras siglas diferentes a las mencionadas anteriormente.

Todos los elementos del modelo de datos descritos por SCORM deben ser implementados y soportados por el LMS, por otra parte para el SCO el uso de todos los elementos del modelo de datos es opcional, puesto que la interacción mínima del SCO para comunicarse con el LMS se realiza a través de las funciones *Initialize("")* y *Terminate("")* y no requiere el uso de las funciones *SetValue()* o *GetValue().*

El nombre de todos los elementos del modelo de datos están limitados para manejarse en ECMAScript como cadenas de texto usando la notación de punto (*eje*. *cmi.success_status*). En las llamadas a la función *SetValue()*, los valores utilizados para ser asignados a los elementos del modelo de datos, deben estar también limitados para manejarse como cadenas de texto.

3.3.10.2 Efectos del modelo de datos en la secuencia

La secuencia de SCORM describe cómo una serie de objetos de aprendizaje son identificados para ser lanzados con base en la información de secuencia, comportamiento de secuencia y los resultados de la interacción del estudiante con el objeto de aprendizaje lanzado. SCORM no pone ninguna restricción en cuando y como la información de un SCO en ejecución es utilizada para la secuencia, SCORM solo requiere que esta información sea la más reciente.

3.3.10.3 Reglas generales del modelo de datos de SCORM

A continuación se mencionan las reglas y características que son importantes para utilizar el modelo de datos de SCORM de una forma adecuada.

- El primer símbolo identifica el modelo de datos.
- Hay tres palabras reservadas. Deben estar en minúsculas e iniciar con guión bajo.
  - _version: Se utiliza para determinar la versión del modelo de datos soportado por el LMS.
  - _children: Se utiliza para determinar que elementos del modelo de datos son soportados por el LMS.
  - _count: Determina el número de elementos que existen actualmente en la lista.
- Los arreglos comienzan en 0, los elementos son puestos en el arreglo de forma secuencial.
- El modelo de datos es sensitivo a mayúsculas y minúsculas.
- El modelo de datos es implementado sobre un SCO y otro SCO no puede acceder al modelo de datos de otro SCO.

3.3.11 Estructura de archivos del SCORM Run-Time Environment

EL Run-Time Environment (RTE) está conformado por distintos componentes y separados de acuerdo a su propósito, es importante mencionar que esta organización de archivos o estructura responde a una implementación del RTE desarrollada bajo Java, por lo que cualquier otra compañía que desarrolle un RTE que se apegue al estándar de SCORM, puede tener una estructura distinta y estar desarrollado en otro lenguaje de programación.

Los componentes del RTE están conformados de la siguiente manera:

1. Run-Time Server Component: Estos componentes están implementados utilizando aplicaciones basadas en Java, tales como Servlets de Java para procesamiento del lado del servidor.
2. Run-Time Environment Client Component: Estos componentes están implementados utilizando JSP, Java Applets, HTML y JavaScript y se utilizan para desplegar información y realizar procesamiento del lado del cliente.

3.3.11.1 Run-Time Server Component

Los servlets responden a peticiones realizadas por los componentes que se encuentran del lado del cliente y son responsables de la persistencia del modelo de datos, así como de la secuencia y el análisis lógico. Los módulos que conforman los Server Component son los siguientes:

1. *ADL Validator*: Este módulo es el responsable de validar el archivo de manifiesto *imsmanifest.xml* el cual es requerido para la agregación de contenido de paquetes SCO. ADL Validator también analiza el archivo XML y regresa una representación en DOM al RTE Server Component.
2. *Utilities*. Este módulo contiene utilidades de propósito general en archivos de Java.
3. *ADL Sequencer*: Este modulo es responsable de manejar la secuencia de navegación. Contiene clases que crean e inicializan el árbol de actividades, mantiene el árbol de actividades a través del proceso de secuencia y manejan toda la lógica de dicha secuencia.
4. *RTE*: EL paquete RTE es la raíz de todos los paquetes Server Component. Contiene la lógica para utilizar otros paquetes cuando sea necesario. También contiene clases de java que interactúan con los Java Server Pages para implementar la funcionalidad del lado del servidor.
5. *DataModels*: Este paquete contiene las clases que mantienen el modelo de datos de SCORM, así como las clases utilizadas para obtener información del modelo de datos referente al modo de navegación.
6. *Shareable State Persistence*: Este paquete contiene las clases que son utilizadas para proveer la implementación de IMS Shareable State Persistence.

3.3.11.2 Run-Time Environment Client Component (Java Files)

Consisten en una serie de interfaces implementadas en HTML, JavaScript y una Instancia del API, el cual es implementado como una instancia de un Java Applet. El Applet es descargado al cliente cuando el usuario entra a la página del Run-Time Environment a través de un navegador Web. Este Applet provee la comunicación a los SCO con el LMS del lado del servidor para la persistencia de los elementos del Modelo de Datos.

3.3.11.3 Run-Time Environment Client Component (Web Files)

Consiste en un conjunto de paquetes que son utilizados para las tareas administrativas de RTE, las cuales se dividen en la siguientes categorías:

1. *Administrative*: Este módulo contiene los archivos que permiten cambiar las contraseñas, eliminar cursos, manejar los objetivos globales, crear o borrar usuarios y limpiar la base de datos. Los archivos que componen el modulo administrativo se localizan en la carpeta "admin".
2. *Help*: Este módulo contiene información sobre como utilizar el RTE. Los archivos que componen el modulo de ayuda se localizan en la carpeta "Help".
3. *Import*: Este módulo es responsable de manejar el proceso de importación de un curso. Estos archivos utilizan los componentes del lado del servidor para manejar el análisis, validación y el acceso a la base de datos. Estos archivos están localizados en la carpeta "Import".
4. *Includes*: EL Run-Time Environment utiliza hojas de estilo (*.css) que manejan la apariencia de las interfaces. Estos archivos se encuentran localizados en la carpeta "Includes" bajo el Web root del Run-Time Environment.
5. *Run-Time*: Este módulo contiene los archivos usados para entregar un Paquete de SCORM. Adicionalmente se incluye el API de SCORM. Todas las imágenes utilizadas en el Run-Time Environment menú están localizadas en la carpeta "menu-images" la cual esta localizada en este mismo modulo. Todos estos archivos están localizados en la carpeta "runtime".
6. *Special State*: Este modulo contiene páginas de información que describen estados especiales de secuencias. Es posible que el contenido no sea entregado como resultado de una petición de secuencia. En ese evento, estas páginas informan al usuario sobre el estado en el que se encuentra el secuenciador. Estas páginas son incluidas para ayudar a depurar las secuencias en el contenido. Este no es un requerimiento de SCORM que debe estar presente. EL LMS puede manejar el estado de las secuencia de una manera personalizada. Los archivos que componen el modulo Special State están localizados en la carpeta "specialstate" bajo el Web root del Run-Time Environment.

3.3.12 Descripción del manifiesto del Content Package

Una vez que se ha hecho el diseño del contenido de aprendizaje, es necesario ponerlo a disponibilidad de los estudiantes, repositorios o LMS. La IMS Content Packaging Specificaction[13] se diseño para proveer una manera estandarizada de estructurar e intercambiar contenido de aprendizaje entre diferentes sistemas y herramientas, también tienen le propósito de proporcionar un lugar para describir la estructura, como está organizado y definir un comportamiento requerido sobre una colección de contenido de aprendizaje. SCORM Content Packaging es un conjunto de específico de reglas del IMS Content Packaging Specification extendido para implementar reglas sobre Assets, SCO y Content Organization.

3.3.12.1 Componentes del Content Package

A continuación se describirá la nomenclatura utilizada para definir paquetes de contenido (content package) y crear dichos paquetes para proveer interoperabilidad de contenido basado en Internet.

Un Content Package contiene dos grandes componentes:

[13] http://www.imsglobal.org/content/packaging/cpv1p1p4/imscp_infov1p1p4.html

- Un documento XML[14] que describe la estructura del contenido y los recursos asociados del paquete, el archivo de manifiesto se llama (*imsmanifest.xml*). Es necesario que este archivo se encuentre en el archivo raíz del content package.
- El contenido (archivos físicos) que conforman el content package.

La figura 3.16 muestra un diagrama conceptual de los componentes del Content Package

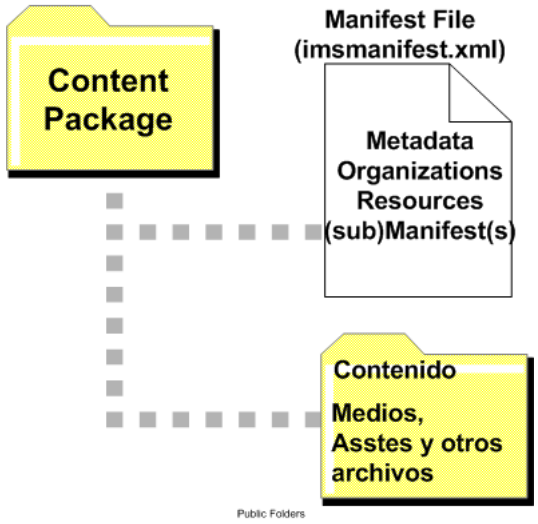


Figura 3.16: Diagrama Conceptual del Content Package

3.3.12.1.1 Package Un paquete representa una unidad de aprendizaje, esta unidad de aprendizaje puede ser parte de un curso el cual tiene capacidades instruccionales independientes, es decir, puede ser aplicado como parte de un curso mas grande o puede ser aplicado de forma separada, también el paquete puede ser una porción del curso, el curso entero o una colección de cursos. Una vez que el paquete es recibido por el sistema destinatario, este debe poder ser desempaquetado por si mismo y contener toda la información necesaria para utilizar de forma apropiada el contenido del paquete.

3.3.12.1.2 Manifest El manifiesto es un documento XML que contienen elementos estructurados del contenido de un paquete. Si el content package tiene como finalidad ser presentado al usuario final, el manifiesto debe contener información que describa como esta organizado el contenido. Un manifiesto pude describir parte de un curso, el cual puede definirse como un objeto de aprendizaje, o bien un curso entero, una colección de cursos, o solo una colección de contenido que puede ser enviado entre un sistema y otro.

Una regla general es que el paquete debe contener siempre un manifiesto de alto nivel que contenga uno o mas (sub)manifiestos. El manifiesto de alto nivel debe describir el paquete y los (sub)manifiestos deben describir el contenido en el nivel que estos (sub)manifiestos alcancen, como puede ser un curso u objeto de aprendizaje.

El manifiesto debe seguir las siguientes reglas definidas por el IMS Content Packaging Specification:

- El manifiesto debe ser nombrado en todos los casos como: *imsmanifest.xml*
- El archivo *imsmanifest.xml* y cualquier otro archivo de control tales como (DTD,XSD) deben estar ubicados en el directorio raíz del Content Package.

[14] http://www.w3.org/XML/

3.3.12.2 Componentes del Manifiesto

Como se mencionaba anteriormente el archivo de manifiesto representa la información necesaria para describir el contenido del paquete. La figura muestra los componentes del manifiesto.

Figura 3.17: Componentes del Manifiesto

El Manifiesto esta compuesto por las secciones siguientes:

- Metadata: Información que describe el Content Package en su totalidad.
- Organizations: Contiene la estructura del contenido o también puede describir otras organizaciones de recursos creando unidades independientes o un conjunto de unidades.
- Resources: Define los recursos de aprendizaje empaquetados en el Content Package.
- (sub)Manifest(s): Describe cualquier otra unidad lógica, como puede ser un objeto de aprendizaje, contenido en la estructura jerárquica del manifiesto principal.

3.3.12.2.1 Metadata Como se menciona anteriormente, los metadatos son utilizados para describir el content package en su totalidad, estos metadatos permiten realizar búsquedas y descubrimientos de content packages en repositorios. Tienen la capacidad de describir las características con las que cuenta un Content Package.

3.3.12.2.2 Organizations Este componente es utilizado para describir como el contenido esta organizado en el content package. Este puede contener uno o mas componentes Organization, donde cada uno de los cuales describe una estructura particular para el contenido del paquete.

3.3.12.2.3 Recursos Este componente describe los recurso externos así como su localización en el content package. Estos archivos pueden contener video, texto o cualquier otra pieza de información electrónica. Las relaciones conceptuales entre los archivos también deben representarse dentro de los recursos. La combinación de recursos se puede categorizar como contenido.

3.3.12.2.4 Contenido El contenido representa los archivos referenciados en los componentes resources. Pueden ser archivos locales contenidos en el Content Package, o pueden ser archivos externos que están referenciados por un (Uniform Resource Indicator) URI. Todos los archivos físicos incluidos en el Content Package deben ser declarados y referenciados en el manifiesto para hacer posible el intercambio de content package.

3.3.13 Adecuaciones Realizadas al Sample Run-Time Environment de SCORM 2004 3ra Edición

Con el objetivo de proporcionar una implementación que integre cada uno de los libros de estándares y especificaciones de SCORM, ADL proporciona una implementación de muestra el Sample Run-Time Environemnt[15], distribuyéndose únicamente para que sea empleada con propósitos de investigación. Esta implementación provee una demostración funcional del Run-Time Environment descrito en SCORM 2004 3ra Edición, el cual no es un LMS cien por ciento funcional, sino mas bien un ejemplo del RTE el cual puede estar dentro de un LMS más robusto con un conjunto de propiedades mas amplia.

La demostración incluye los componentes mas significativos que se encargan de la importación y entrega de Content Aggregation Package, la comunicación estandarizada entre el contenido y el LMS, la utilización de un modelo de datos estandarizado para pasar información importante de la experiencia del estudiante con el contenido e información relacionada con la secuencia y navegación del contenido. Además, de la parte funcional del LMS ADL proporciona el código fuente de la implementación.

Puesto que el framework de SCORM, implementa la persistencia de la información relacionada con el usuario y el objeto de aprendizaje, de acuerdo a un modelo de datos preestablecido que permite el intercambio de información entre el contenido educativo y diferentes LMS.

Al contar con un lenguaje o vocabulario en común, esta cualidad de SCORM tiene el objetivo de lograr que el contenido educativo sea reutilizable e interoperable entre múltiples Learning Management System (LMS).

Por lo que resulta de gran utilidad para los objetivos planteados anteriormente, aunque con la desventaja de tener forzosamente un vocabulario predefinido para el intercambio de datos, lo que se contrapone a las cualidades que el sistema de persistencia debe ofrecer.

Tomando en cuenta que en la mayoría de las ocasiones el contenido desplegado en los objetos de aprendizaje no siempre manejaran el mismo lenguaje y resultaría inapropiado extender el modelo de datos, ya que el modelo debería contemplar las estructuras y el formato de la información a ser persistente para almacenarla y recuperarla posteriormente.

Tomando en cuenta las cualidades del Framework de SCORM para administrar y desplegar contenido educativo y los objetivos particulares para el control de persistencia del contenido de los objetos de aprendizaje, los cuales fueron planteados en la introducción. Se tomó la decisión de contemplar el Run-Time Environment de SCORM como un mecanismo de comunicación y despliegue de contenido educativo dentro del ambiente de aprendizaje AIIDM, con las adecuaciones necesarias para recuperar de un sistema de almacenamiento la información de la interacción con el objeto de aprendizaje y el usuario, así como desplegar esta información posteriormente.

Para lograr que la implementación de muestra del RTE de SCORM que ofrece ADL pudiera ser integrada al AIIDM como mecanismo principal para desplegar los IIDM con cualidades de persistencia en su contenido, fue necesario realizar las adecuaciones necesarias para este propósito. ADL hace la distribución del Sample RTE solo para plataformas con base en sistemas operativos Windows, la aplicación esta desarrollada como una aplicación Web en J2EE, instala en el sistema una estructura de directorios en la carpeta raíz del disco duro (Unidad C), el servidor de aplicaciones Apache Tomcat y como sistema de almacenamiento utiliza una base de datos de Access[16] y configura los DSN (Data Source Name) en el Administrador de Orígenes de Datos ODBC (Open DataBase Conectivity) en el sistema operativo, así como un sistema de carpetas para almacenar la información del usuario con respecto a la interacción con el contenido de alguno de los cursos, como se muestra en la figura 3.18.

[15] disponible en: SCORM Sample Run-Time Environment `http://www.adlnet.gov/downloads/downloadpage.aspx?ID=280`

[16] http://office.microsoft.com/es-es/access/default.aspx

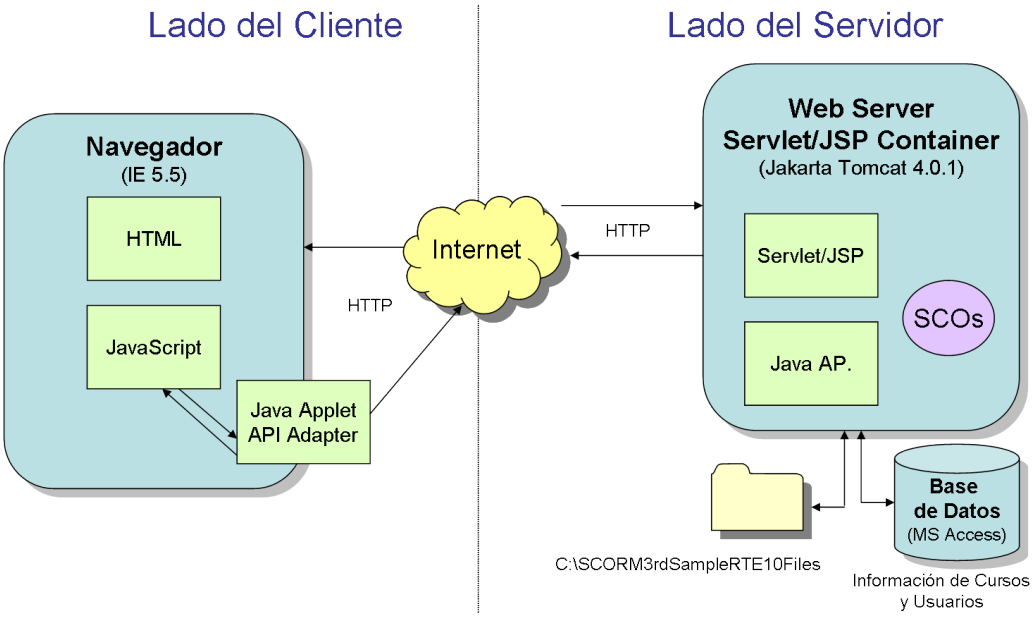


Figura 3.18: Arquitectura del Sistema Sample RTE

Una de las primeras razones para realizar las adecuaciones al Sample RTE, fue eliminar o quitar la necesidad de realizar un despliegue de la aplicación sobre un sistema operativo en particular, migrar a un sistema de almacenamiento más robusto y flexible y de igual forma evitar el almacenamiento de los archivos requeridos por la aplicación en sistemas de carpetas con rutas fijas en la raíz del disco duro, todo esto con el objetivo de obtener una aplicación más portable sin restricciones de plataforma, bases de datos propietarias o sistemas de archivos con rutas fijas. El diagrama de paquetes de la figura 3.19 muestra la distribución de las clases que sufrieron alguna modificación para lograr los objetivos de portabilidad mencionados anteriormente.

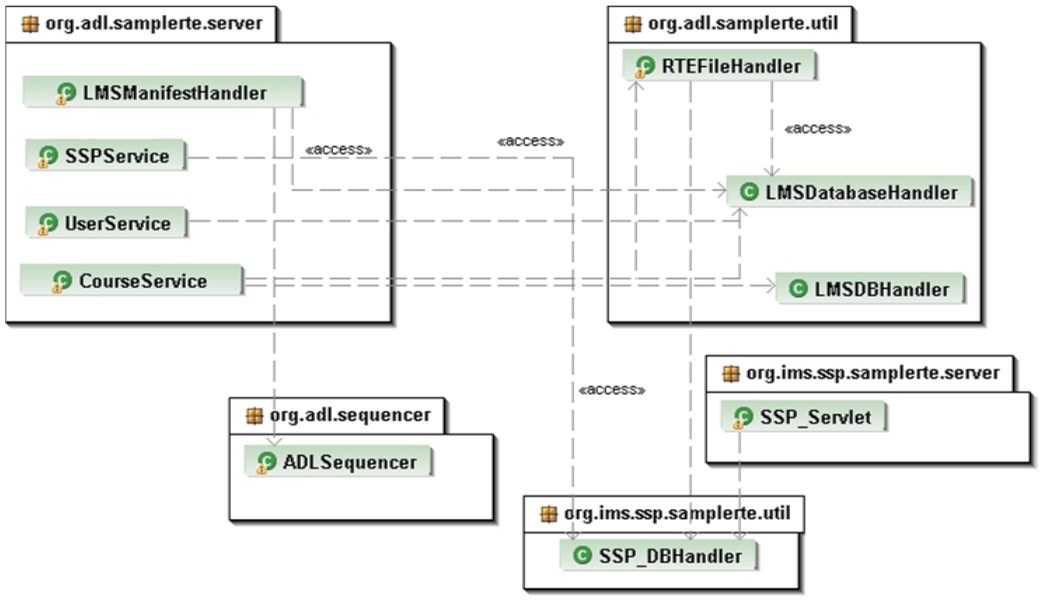


Figura 3.19: Diagrama de Paquetes de las clases con adecuaciones

Otro de los requerimientos por los cuales se realizaron las adecuaciones del Sample RTE fue para obtener una implementación del Run-Time Environment cien por ciento dedicada al despliegue de objetos de aprendizaje y control de la persistencia del contenido de los objetos de aprendizaje lo suficientemente robusto, con un mejor rendimiento y a la vez más ligero. Para lograr este objetivo fue necesario prescindir de todos aquellos componentes relacionados con las opciones administrativas del LMS. En la siguiente lista se muestran dichos componentes así como los archivos JSP que intervienen en la funcionalidad:

- Opciones administrativas de cuentas de usuario
  - newUser.jsp
  - deleteUser.jsp
  - LMSUserAdmin.java
  - UserService.java
- Opciones administrativas de los cursos registrados
  - importCourse.jsp
  - deleteCourse.jsp
  - clearDatabase.jsp
  - LMSCourseAdmin.java
  - CourseAdmin.java

Los servlets y clases de java, no fueron eliminados totalmente del sistema, puesto que además de realizar funciones administrativas, también son utilizados para cargar los cursos de un usuario o para desplegar información relacionada con este, por lo que solo fue necesario dejar dicha funcionalidad. Es importante mencionar que la funcionalidad administrativa, quedará como una aplicación independiente y de igual forma de funcionalidad dedicada solo para dichos propósitos.

Con respecto al sistema de almacenamiento, para reemplazar la base de datos, se creo una base de datos con la misma estructura de la base de datos original, es decir, tablas, columnas, llaves primarias,etc. en Apache Derby. Derby es un proyecto de Apache[17] el cual consiste en un manejador de bases de datos relacional de código abierto completamente implementada en Java y disponible bajo la licencia Apache Versión 2.0[18]. Entre las cualidades que podemos resaltar de Apache Derby, es que es una base de datos basada en Java, JDBC y el estándar SQL, tiene soporte para modo cliente/servidor, es fácil de instalar, desplegar y utilizar, pero sobre todo provee un driver JDBC empotrado el cual permite incluir una base de datos Derby en una aplicación basada en Java, esto nos da la ventaja primeramente de utilizar una base de datos de código abierto y de uso gratuito y posteriormente de agregar la característica de portabilidad a la implementación personalizada del Sample RTE, evitando configuraciones extras para manejar la base de datos. Además si a esto le agregamos la posibilidad de utilizar una ruta dinámica para los archivos y carpetas que el LMS genera y utiliza dentro del sistema de archivos de la plataforma aseguramos un LMS que puede ser desplegado en cualquier servidor de aplicaciones Servlet/JSP.

[17] Apache Derby: http://db.apache.org/derby/
[18] Apache Licencia: http://db.apache.org/derby/license.html

Capítulo 4

Diseño y Desarrollo de los Mecanismos de Persistencia

En este capítulo en primer lugar se presenta el desarrollo de los mecanismos de persistencia que proporcionan la base tecnológica para soportar los esquemas de persistencia. El desarrollo de estos mecanismos se realizó sobre el Sample Run-Time personalizado, aprovechando las características de despliegue de contenido y de comunicación con el LMS.

La diferencia que existe entre el LMS que incorpora los mecanismos de persistencia de estado para los objetos de aprendizaje y el AIIDM, es que eventualmente el AIIDM podrá eventualmente registrar objetos de aprendizaje IIDM envueltos en clientes ligeros del LMS de SCORM, con la finalidad de aprovechar las ventajas de SCORM y los mecanismos de persistencia desarrollados en esta tesis.

El ciclo de desarrollo de los prototipos para los mecanismos de persistencia se realizó bajo la metodología Extreme Programming, la cual consta de varias etapas. En este caso, tomamos las partes más importantes y significativas que se adaptaran a las características del proyecto.

Como se especifica en [Canós et al., 2003] y en [Duartes & Rojas, 2008] el ciclo de vida inicia con la especificación de historias de usuario. Las historias de usuario son fichas donde el usuario describe las características que debe tener el sistema en lenguaje común, así como la funcionalidad que debe proveer. Sobre estas historias de usuario se establece la prioridad, esfuerzo o tiempo requerido y se pone de manifiesto en el plan de entrega. Es en este plan donde se especifican las iteraciones que serán necesarias para desarrollar los requerimientos escritos en las fichas de usuario como se muestra en la figura: 4.1.

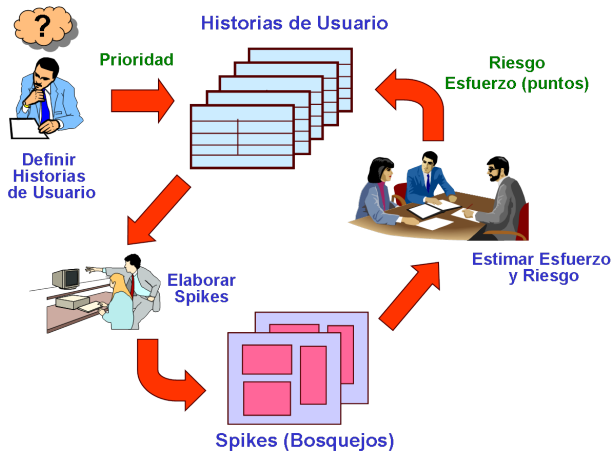


Figura 4.1: Vista de Actividades de un Proyecto XP

Las iteraciones son las etapas en las cuales se asignan cierto numero de historias de usuario de acuerdo a su prioridad, siendo una de las primeras tareas de los desarrolladores descomponer estas historias de usuario en tareas de programación. Las tareas de programación son fichas donde se especifica en lenguaje técnico los requisitos o la funcionalidad que el sistema debe proveer.

Posteriormente y antes de realizar los trabajos de diseño y programación se elaboran las pruebas de aceptación. Dichas pruebas ayudan a enfocar los objetivos del desarrollo y son realizadas por el usuario final. Con las tareas de programación y las pruebas de aceptación elaboradas, se inicia con el diseño, el cual debe ser realizado lo mas sencillo posible y apegado únicamente a las fichas de las tareas de programación, con el propósito de seguir con la programación y la realización de las pruebas de aceptación propias de cada iteración. En la figura 4.2 se muestran las actividades realizadas en cada una de las iteraciones. Para este proyecto se definieron como pruebas de aceptación todas aquellas acciones necesarias para cumplir con las historias de usuario. Si alguna de estas acciones no se cumple, entonces se tiene que volver al diseño y la programación para lograr satisfactoriamente las pruebas de aceptación.

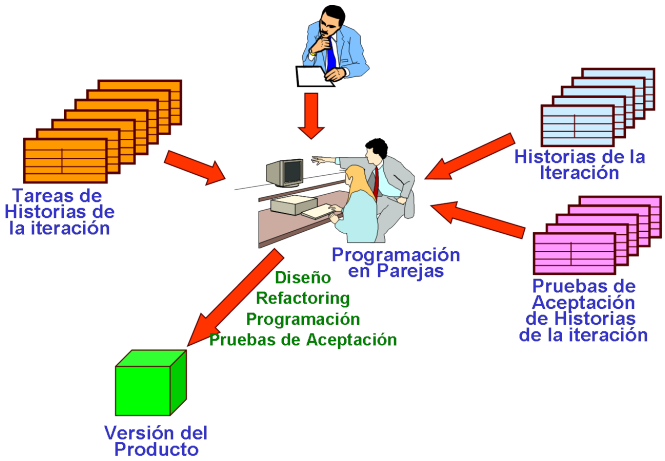


Figura 4.2: Actividades en cada una de las iteraciones

4.1 Historias de Usuario

En esta sección se presentan las historias de usuario que se desarrollaron para implementar los mecanismos de persistencia con base en el comportamiento que el sistema debe tener en aquellos casos en los que el servicio de persistencia intervengan. En las tablas 4.1, 4.2, 4.3, 4.4, 4.5 y 4.6 se describen estas historias de usuario.

Historia de Usuario	
Número: 1	**•Usuario: Diseñador Instruccional**
Nombre historia: Recuperación de Información del Contenido de aprendizaje	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Alto
Puntos estimados: 2	**Iteración asignada: 1**
Programador responsable: José de Jesús Esparza Flores	
Descripción: **Después de retomar las actividades de un objeto de aprendizaje, donde el Learner Session haya sido suspendido previamente, el sistema recupera la información la cual fue introducida en los campos del objeto de aprendizaje.**	
Observaciones: *El Learner Session es el periodo de tiempo ininterrumpido durante el cual un estudiante esta accediendo a un objeto de aprendizaje.* *Los campos son aquellos controles de entrada de datos.* *La información a recuperar debe ser la correspondiente al curso, el usuario y la actividad de aprendizaje que desarrolla.*	

Tabla 4.1: Historia de Usuario: Recuperación de Información del Contenido de Aprendizaje

Historia de Usuario	
Número: 2	**Usuario: Diseñador Instruccional**
Nombre historia: Despliegue de la Información en el objeto de aprendizaje	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada: 1**
Programador responsable: José de Jesús Esparza Flores	
Descripción: **El sistema despliega la información recuperada en los controles de entrada de datos del objeto de aprendizaje correspondientes.**	
Observaciones: *El despliegue de la información se realizara una vez cargado el objeto de aprendizaje de forma transparente para el estudiante.*	

Tabla 4.2: Historia de Usuario: Despliegue de la Información en el objeto de aprendizaje

Historia de Usuario	
Número: 3	**Usuario:** Diseñador Instruccional
Nombre historia: Obtener la Información del Contenido de Aprendizaje	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: José de Jesús Esparza Flores	
Descripción: Antes de que el estudiante salga de la actividad actual, el sistema debe recuperar la información introducida en los controles del objeto de aprendizaje. La actividad actual del estudiante puede ser dejada por la interrupción del Learner Session, la finalizacion del Learner Attempt o para continuar con la siguiente actividad.	
Observaciones: Learner Attempt: Es el intento o esfuerzo aplicado por un estudiante para lograr o completar los requerimientos de una actividad de aprendizaje dentro de un objeto de contenido.	

Tabla 4.3: Historia de Usuario: Obtener la Información del Contenido de Aprendizaje

Historia de Usuario	
Número: 4	**Usuario:** Diseñador Instruccional
Nombre historia: Almacenamiento de la información del Contenido de Aprendizaje	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Alto
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: José de Jesús Esparza Flores	
Descripción: Almacenar los datos introducidos en los controles de entrada del objeto de aprendizaje durante la actividad de aprendizaje realizada por el estudiante.	
Observaciones: *La información a almacenar debe ser la correspondiente al curso, el usuario y la actividad de aprendizaje que desarrolla.*	

Tabla 4.4: Historia de Usuario: Almacenamiento de la información del Contenido de Aprendizaje

Historia de Usuario	
Numero: 5	•Usuario: Administrador del LMS
Nombre Historia: Registrar Cursos con un Esquema de Persistencia Asociado	
Prioridad en Negocio: Alta	Riesgo en Desarrollo: Alto
Puntos Estimados: 4	Iteración asignada: 3
Programador Responsable: José de Jesús Esparza Flores	
Descripción: Después de haber registrado el Curso en LMS con éxito, el administrador podrá asignarle un esquema de persistencia al curso, así como también, especificar las configuraciones necesarias referentes a los tipos de persistencia que componen el esquema.	
Observaciones: El diseñado instruccional deberá especificar el esquema de persistencia que debe tener el curso y los elementos que conformaran el estado del curso.	

Tabla 4.5: Historia de Usuario: Registro de Cursos con Esquema de Persistencia

Historia de Usuario	
Numero: 6	•Usuario: Usuario Final
Nombre Historia: Implementación de los Esquema de Persistencia en el LMS	
Prioridad en Negocio: Alta	Riesgo en Desarrollo: Medio
Puntos Estimados: 2	Iteración asignada: 3
Programador Responsable: José de Jesús Esparza Flores	
Descripción: Después que el usuario cargue en el LMS el curso para su utilización, el LMS deberá identificar si el curso tiene un esquema de persistencia registrado, de ser así, deberá cargar las configuraciones pertinentes para dar soporte al esquema.	
Observaciones: Los esquemas de persistencia no deberán alterar el funcionamiento normal del curso y deberán incluir los elementos necesarios para soportar los mecanismos de persistencia.	

Tabla 4.6: Historia de Usuario: Implementación de Esquema

4.2 Plan de Entrega

Debido a las características del proyecto las entregas se realizarán por alcance y no por fechas, es decir, se realizarán las entregas de las implementaciones funcionales en cada una de las iteraciones una vez que las tareas de desarrollo, así como las pruebas de aceptación hayan sido completadas.

No.	Nombre	Prioridad	Riesgo	Esfuerzo	Iteración
1	Recuperación del Contenido de Aprendizaje	Media	Alto	2	1
2	Despliegue de la información del objeto de aprendizaje	Media	Medio	1	1
3	Obtener la información del contenido de aprendizaje	Media	Medio	2	2
4	Almacenamiento de la información del contenido de aprendizaje	Media	Alto	3	2
5	Registrar cursos con un esquema de persistencia asociado	Alta	Alto	4	3
6	Implementación de los esquema de persistencia en el LMS	Alta	Medo	2	3

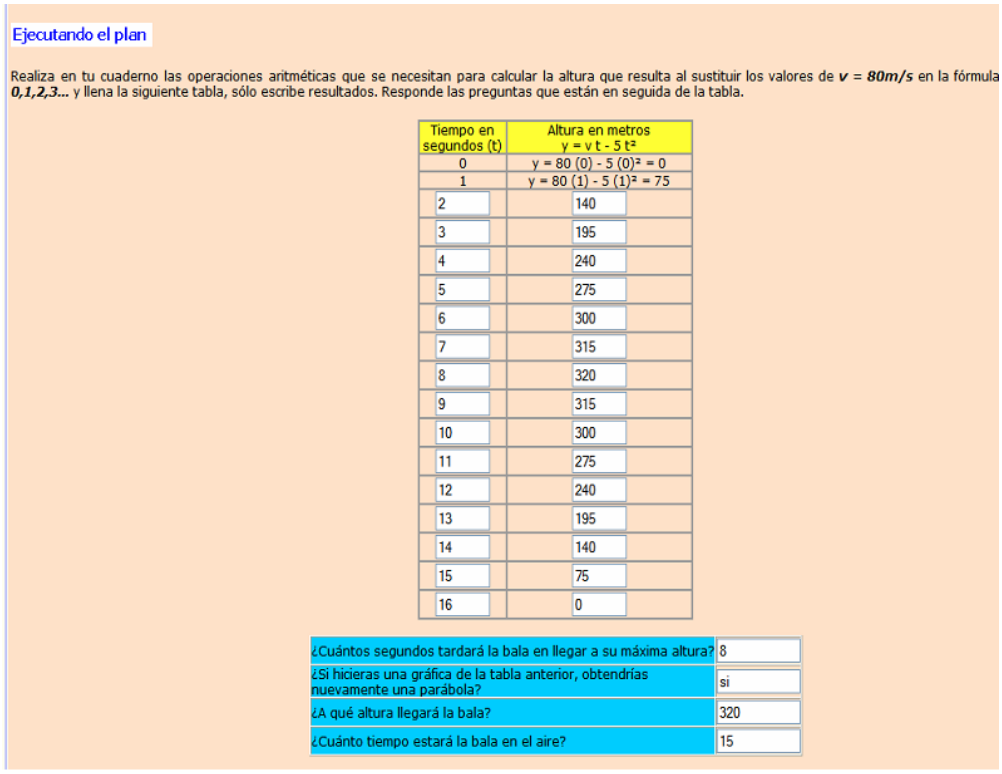
Tabla 4.7: Plan de Entrega

Primera Iteración En la primera iteración se trata de obtener la funcionalidad asociada con la recuperación de la información relacionada con la actividad del objeto de aprendizaje y posteriormente desplegarla en pantalla de acuerdo a los campos de introducción de datos correspondientes. En esta iteración, la información que se recupera es obtenida de un sistema de almacenamiento, es decir, se da por hecho que ya fue recuperada de la base de datos y a partir de esta información de prueba se desarrolla la funcionalidad necesaria para su despliegue.

Segunda Iteración En la segunda iteración se pretende entregar una implementación funcional de la aplicación acuerdo a las historias de usuario, donde la información pueda ser obtenida de los campos de entrada de la actividad del objeto de aprendizaje y guardarla en un sistema de almacenamiento de acuerdo al curso de aprendizaje y la actividad relacionada con el estudiante. Así mismo integrar la funcionalidad obtenida en la primera iteración con la finalidad de desarrollar los mecanismos de almacenaje de la información, con datos obtenidos directamente de la actividad de aprendizaje.

Tercera Iteración En la tercera iteración se realizaran las adecuaciones pertinentes tanto en el LMS como en esquema de la base de datos para dar soporte a la asignación de los esquemas de persistencia a un curso registrado previamente, teniendo en cuenta las características de cada uno de estos esquemas con el fin de desarrollar el software que permita configurarlas e implementarlas. Así como también se desarrollarán los mecanismos necesarios en el LMS para que se carguen las configuraciones necesarias del esquema con el cual este registrado un curso determinado, teniendo en cuenta los mecanismos de comunicación básicos de persistencia para soportar los esquemas. Con el fin de que el usuario final pueda interactuar con los cursos y pueda trabajar con el estado del curso dependiendo del esquema que tenga asignado.

Prototipo En la imagen 4.3 se muestra un primer bosquejo de como debe desplegarse la información en los controles de entrada de datos, con el fin de ilustrar el comportamiento del sistema.



Ejecutando el plan

Realiza en tu cuaderno las operaciones aritméticas que se necesitan para calcular la altura que resulta al sustituir los valores de ***v = 80m/s*** en la fórmula ***0,1,2,3...*** y llena la siguiente tabla, sólo escribe resultados. Responde las preguntas que están en seguida de la tabla.

Tiempo en segundos (t)	Altura en metros $y = v\,t - 5\,t^2$
0	$y = 80\,(0) - 5\,(0)^2 = 0$
1	$y = 80\,(1) - 5\,(1)^2 = 75$
2	140
3	195
4	240
5	275
6	300
7	315
8	320
9	315
10	300
11	275
12	240
13	195
14	140
15	75
16	0

¿Cuántos segundos tardará la bala en llegar a su máxima altura?	8
¿Si hicieras una gráfica de la tabla anterior, obtendrías nuevamente una parábola?	si
¿A qué altura llegará la bala?	320
¿Cuánto tiempo estará la bala en el aire?	15

Figura 4.3: Primer Prototipo: Recuperación y Despliegue de la información almacenada en la base de datos.

4.3 Primera Iteración

4.3.1 Tareas de Desarrollo para la Historia de Usuario: 1

Tarea	
Número tarea: 1	**Número historia:** 1
Nombre tarea: Interfaces de recuperación de la información	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: --/--/----	**Fecha fin:** --/--/----
Programador responsable: José de Jesús Esparza Flores	
Descripción: Desarrollar las interfaces de comunicación del cliente con el servidor para recuperar la información como cadena de texto con formato XML. Los mecanismos de comunicación contemplan llamadas directas al servidor para proveer los datos almacenados a través de Asincronus,JavaScript,XML (AJAX) en un archivo javascript independiente de la implementación del API de comunicación existente, el cual será llamado después de cargar el contenido de la actividad educativa asociada al estudiante.	

Tabla 4.8: Tarea de Desarrollo: Interfaces de recuperación de la Información

Tarea	
Número tarea: 2	**Número historia:** 1
Nombre tarea: Recuperar Identificadores de la información a obtener	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: --/--/----	**Fecha fin:** --/--/----
Programador responsable: José de Jesús Esparza Flores	
Descripción: Extender la funcionalidad de la implementación del API (Applet) de comunicaciones para obtener los identificadores del curso en el cual el estudiante se encuentra, así como la clave del estudiante y el nombre de la actividad que esta realizando.	

Tabla 4.9: Tarea de Desarrollo: Recuperar Identificadores de la información a obtener

4.3.2 Tareas de Desarrollo para la Historia de Usuario: 2

Tarea	
Número tarea: 1	Número historia: 2
Nombre tarea: Obtener la estructura XML de la cadena	
Tipo de tarea : Desarrollo	Puntos estimados: .5
Fecha inicio: --/--/---	Fecha fin: --/--/---
Programador responsable: José de Jesús Esparza Flores	
Descripción: Desarrollar los mecanismos necesarios que permitirán al cliente en este caso al navegador Web analizar la estructura de la cadena de texto y obtener un objeto de tipo Document Object Model (DOM) para manipular los nodos que lo conforman. La funcionalidad para recorrer el objeto DOM, se realizara a traves de un archivo JavaScript independiente de la implementacion del API de comunicación existente, el cual será llamado después de cargar el contenido de la actividad educativa asociada al estudiante.	

Tabla 4.10: Tarea de Desarrollo: Obtener la estructura XML de la cadena

Tarea	
Número tarea: 2	Número historia: 2
Nombre tarea: Inserción de datos a los controles de entrada	
Tipo de tarea : Desarrollo	Puntos estimados: .5
Fecha inicio: --/--/---	Fecha fin: --/--/---
Programador responsable: José de Jesús Esparza Flores	
Descripción: Recorrer los nodos del objeto DOM y de acuerdo los campos de tipo de dato e identificador de dato, asignar los valores correspondientes en los controles de entrada en la actividad de aprendizaje. La asignación de los valores a los campos de entrada, será a través de DHTML, el cual será implementado desde un archivo JavaScript independiente de la implementación del API de comunicación existente, el cual será llamado después de cargar el contenido de la actividad educativa asociada al estudiante.	

Tabla 4.11: Tarea de Desarrollo: Inserción de datos en los controles de entrada

Elementos de Diseño para la Primera Iteración

A continuación se presentan las funciones implementadas bajo JavaScript de las tareas correspondientes a la primera historia de usuario. Este archivo contiene todas las rutinas JavaScript que permiten actualizar dinámicamente la página HTML (mediante DHTML) y las rutinas que permiten comunicarse con el servidor para el envío y recepción de información.

Función	Descripción	Atributos
loadXML	De acuerdo al navegador se analiza la cadena de texto para obtener un objeto DOM, con el fin de manipular posteriormente los nodos	•cadena_xml: Variable tipo String que contiene la información recuperada en formato XML.
setCampos	Asigna valor a cada uno de los inputs del documento html de acuerdo con el id del input recorriendo un documento xml el cual es recuperado desde el servidor	•xmlDoc: Objeto tipo DOM que sera recorrido para recuperar de sus nodos la información correspondiente a los controles de entrada de datos.
createXmlHttpRequest	Crea un objeto de la clase XMLHttpRequest, dependiendo del navegador la sintaxis para crear el objeto varia	•Ninguno.
startRequest	Lanza la petición al servidor llamando la función createXmlHttpRequest para obtener el objeto y establecer los parámetros necesarios de comunicación y envía la petición al servidor.	•cadena: Variable de tipo String que contiene los datos para identificar la información del contenido a recuperar, tales como curso, actividad y usuario. •modo: Variable de tipo booleana que se envía como parte del URL, para especificar al servidor en caso de ser verdadera que se trata de una recuperación de datos y falsa si se trata del almacenamiento de los datos.
handleStateChange	Se utiliza para conocer el estado del objeto XmlHttpRequest después del envio de la petición	•Ninguno

Tabla 4.12: Rutinas JavaScript para la comunicación con AJAX

Para obtener los identificadores fue necesario agregar cierta funcionalidad que permitiera obtener dichos datos con el fin de recuperar los datos del curso que el estudiante esta tomando, así como la actividad que esta desarrollando y el identificador del estudiante desde las rutinas de JavaScript. A continuación se presenta una tabla con los detalles de la funcionalidad agregada al Applet de SCORM.

Función	Descripción	Atributos
`getMUserID`	Regresa el identificador del usuario que previamente se identifico en el sistema.	Ninguno
`getMActivityID`	Regresa el identificador de la actividad presentada al usuario.	Ninguno
`getMCourseID`	Regresa el identificador del curso que esta tomando el usuario actualmente.	Ninguno

Tabla 4.13: Funciones Agregadas a la Implementación del API de SCORM

Para el envío y recepción de datos desde el cliente al servidor y viceversa, se empleo una estructura XML la cual contiene además de los datos a ser persistentes, los datos necesarios para identificarlos dentro del sistema de almacenamiento, tales como la identificación del usuario, la identificación del curso y la identificación de la actividad. La estructura de la cadena de envío se muestra en: 2

Documento XML 2 Estructura XML para la cadena de envío de datos

```
<?xml version='1.0' encoding='UTF-8'?>
<contenido>
    <datos userid='jesparza' courseid='curso-1'
    activityid ='Actividad-1'>
    </datos>
</contenido>
```

Una vez que el servidor procesa la petición de forma exitosa, es decir, encuentra información a ser enviada al cliente, el proceso del lado del servidor regresa una cadena de texto. La estructura 3 muestra los datos referentes a la identificación del usuario, el curso y la actividad, después se especifican los campos así como sus atributos los cuales serán enviados al cliente para ser desplegados en el objeto de aprendizaje.

Documento XML 3 Estructura XML para la cadena de envío de datos

```
<?xml version='1.0' encoding='UTF-8'?>
<contenido>

    <datos userid='jesparza' courseid='curso-1' activityid ='Actividad-1'></datos>
    <campos>
        <campo name='field1' type='text' value='1'></campo>
        <campo name='field2' type='text' value='2'></campo>
        <campo name='field3' type='text' value='3'></campo>
        <campo name='field4' type='text' value='4'></campo>
    </campos>
</contenido>
```

En el diagrama se muestra la secuencia de las interacciones que existen entre los objetos que intervienen en la comunicación del primer prototipo: 4.4

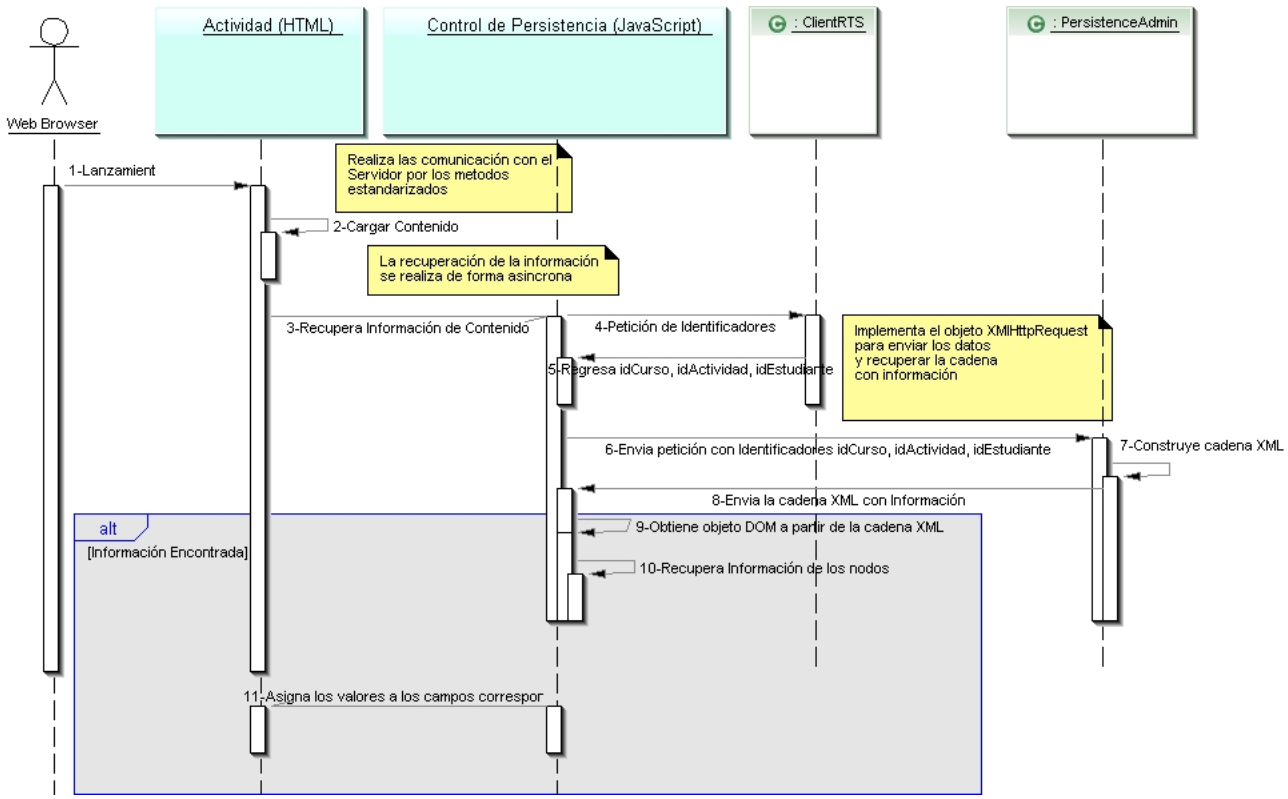


Figura 4.4: Diagrama de Secuencias de Comunicación del Primer Prototipo

Del lado del servidor se implementó un Servlet que recibe las peticiones de datos, en este primer prototipo esta clase no tiene acceso a la base de datos, la única función que tiene es recibir la solicitud, generar una cadena de texto en formato XML simulando un resultado y posteriormente enviarla al cliente. El diagrama se muestra en la figura: 4.5

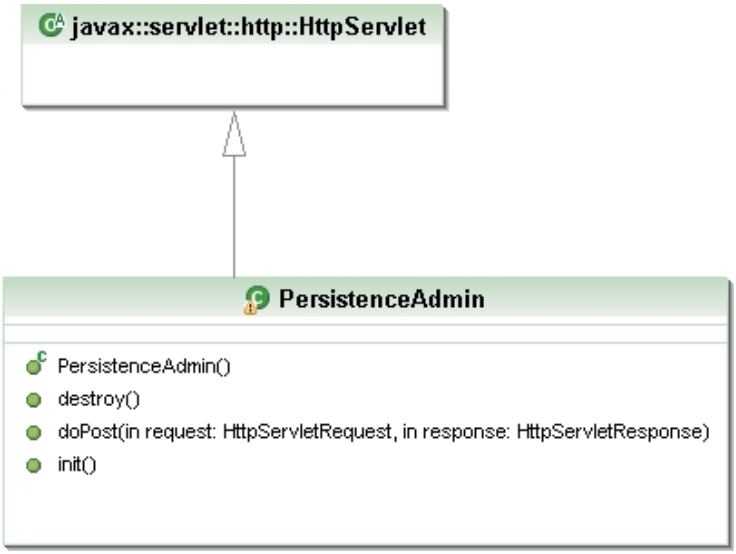


Figura 4.5: Diagrama de la clase: PersistenceAdmin

4.4 Plan de Entrega Inicial (Calendario de Compromiso)

4.4.1 Iteración 1

Versión Inicial Consta de 2 Historias de Usuario

1. Recuperación de Información del Contenido de aprendizaje – 2 puntos
   - 2 Tareas
     (a) Interfaces de recuperación de la información (1 punto)
     (b) Recuperar Identificadores de la información a obtener (1 punto)
2. Despliegue de la Información en el objeto de aprendizaje – 1 punto
   - 2 Tareas
     (a) Obtener la estructura XML de la cadena (.5 puntos)
     (b) Inserción de datos a los controles de entrada (.5 puntos)

Pruebas de Aceptación
Historia 1:

- El usuario accede a una actividad del curso, donde en sesiones previas interactuó con los elementos de la pagina HTML, donde se realizó la captura de datos en los elementos de entrada correspondientes.
  - El sistema identifica el usuario relacionado con el curso y la actividad presentada para recuperar del sistema de almacenamiento los datos capturados en las sesiones previas.

Historia 2:

- Desplegar los datos en los controles correspondientes de entrada dentro de la pagina HTML en los navegadores Internet Explorer y Mozilla Firefox.
  - La prueba no paso satisfactoriamente para los dos navegadores debido a las diferencias con existen entre estas tecnologías especificamente para la función del DOM getElementByID(arg), donde para Internet Explorer el arg puede ser el nombre o identificador de la caja de entrada, en el caso de Mozilla Firefox es solamente el id de la caja de entrada.

Incidencias

- Los mecanismos para recuperar los datos persistentes desde el servidor requieren que se conozca previamente la dirección o URL del servidor, esto resulta inapropiado en términos de interoperabilidad puesto que la ubicación del servidor de aplicaciones que contiene el sistema no siempre será la misma, lo que implica modificar la línea de código en el archivo de JavaScript que hace referencia a la ruta del servidor.
- Puesto que la implementación del API de SCORM incluye la información referente al estado en el que se encuentra el curso, la información del usuario e información de la secuencia de las actividades relacionadas con el curso y el estudiante, se tuvo que agregar las interfaces que permitieran recuperar los datos del curso, actividad y estudiante desde las rutinas de JavaScript para armar la petición de los datos al servidor.
- Al momento de desplegar la información se encontraron restricciones en la forma en que Internet Explorer y Mozilla Firefox implementan dentro de la clase para objetos DOM la función "getElementById" puesto que para Internet Explorer el nombre de un elemento y la id del mismo elemento son términos utilizados indistintamente, mientras que en Mozilla Firefox estos términos no tienen el mismo significado. Esto nos da como resultado que en Internet Explorer podemos utilizar document.geElementById(nombre_elemento) y document.geElementById(id_elemento) indistintamente sin ningún problema, mientras que en Mozilla Firefox, no sucede igual.

4.5 Segunda Iteración

4.5.1 Tareas de Desarrollo para la Historia de Usuario: 3

Tarea	
Número tarea: 1	**Número historia: 3**
Nombre tarea: Crear Estructuras de Comunicación para el servicio de Persistencia	
Tipo de tarea : Diseño/Desarrollo	**Puntos estimados: .5**
Fecha inicio: --/--/----	**Fecha fin: --/--/----**
Programador responsable: José de Jesús Esparza Flores	
Descripción: Diseñar e Implementar las clases LMSPersistenServlet, LMSPesistenRequest y LMSPersistenceResponse las cuales contendrán los mecanismos necesarios que permitirán la comunicación con el servicio de persistencia y la implementación del API (Applet) para las tareas de almacenamiento y recuperación de datos.	

Tabla 4.14: Tarea de Desarrollo: Crear Estructuras de Comunicación para el servicio de Persistencia

Tarea	
Número tarea: 2	**Número historia:** 3
Nombre tarea: Extender el modelo de comunicación existente.	
Tipo de tarea : Diseño/Desarrollo	**Puntos estimados:** 1
Fecha inicio: --/--/----	**Fecha fin:** --/--/----
Programador responsable: José de Jesús Esparza Flores	
Descripción: Integrar al modelo de comunicación existente las extensiones necesarias para contemplar los requerimientos de acceso a los servicios de persistencia sin afectar las funciones ya existentes.	

Tabla 4.15: Tarea de Desarrollo: Adecuar el modelo de comunicación existente

Tarea	
Número tarea: 3	**Número historia:** 3
Nombre tarea: Actualizar la Implementación del API al modelo de comunicación extendido.	
Tipo de tarea : Diseño/Desarrollo	**Puntos estimados:** .5
Fecha inicio: --/--/----	**Fecha fin:** --/--/----
Programador responsable: José de Jesús Esparza Flores	
Descripción: Implementar la funcionalidad que de soporte a través del API existente a los servicios de persistencia, tomando en cuenta el modelo de comunicación extendido sin afectar el comportamiento principal de la implementación del API. Tomando en cuenta las restricciones del modelo de datos y su comportamiento en casos en los que se hacen invocaciones a elementos del modelo de datos no existentes.	

Tabla 4.16: Tarea de Desarrollo: Actualizar la Implementación del API al modelo de comunicación adecuado

4.5.2 Tareas de Desarrollo para la Historia de Usuario: 4

Tarea	
Número tarea: 1	**Número historia: 4**
Nombre tarea: Diseñar y Crear las tablas de la Base de Datos	
Tipo de tarea : Diseño/Desarrollo	**Puntos estimados: 1**
Fecha inicio: --/--/----	**Fecha fin: --/--/----**
Programador responsable: José de Jesús Esparza Flores	
Descripción: **Diseñar las tablas de la base de datos y posteriormente escribir y ejecutar las sentencias DDL con el diseño de las tablas tomando en cuenta los tipos de datos de tablas que almacenan información de los cursos y actividades, así como estudiantes para las llaves primarias y columnas de las tablas de persistencia. La columna que almacenara los datos persistentes sera de tipo BLOB.**	

Tabla 4.17: Tarea de Desarrollo: Diseñar y Crear las tablas de la Base de Datos

Tarea	
Número tarea: 2	**Número historia: 4**
Nombre tarea: Desarrollar los servicios de Persistencia	
Tipo de tarea : Diseño/Desarrollo	**Puntos estimados: 2**
Fecha inicio: --/--/----	**Fecha fin: --/--/----**
Programador responsable: José de Jesús Esparza Flores	
Descripción: **Diseñar e implementar las funciones necesarias que permitan armar y analizar cadenas de texto con formato XML como resultado de la recuperación de los datos o como entrada para almacenar la información en la base de datos, así como también los mecanismos que ejecuten sentencias SQL para la recuperación, inserción y actualización de los datos persistentes.**	

Tabla 4.18: Tarea de Desarrollo: Desarrollar los servicios de Persistencia

Elementos de Diseño para la Segunda Iteración En la segunda iteración se desarrolló un prototipo con modificaciones en el diseño de comunicación del cliente con el servidor, en la primera iteración se implementó un mecanismo soportado por AJAX con comunicación directa con las funciones de persistencia, en este segundo prototipo además de realizar las tareas de desarrollo propias a la iteración se implementaron mecanismos de comunicación a través de las funciones estandarizadas de SCORM RTE, es decir, se explotaron las características que provee para recuperar y almacenar información referente al modelo de datos preestablecido.

Este cambio en el diseño del mecanismo de comunicación elimina la limitante de especificar de manera explicita el nombre y puerto del servidor de aplicaciones del sistema, así como también elimina la necesidad de tener otro conjunto de rutinas JavaScript, es decir, en las rutinas de JavaScript (API_Wraper) que ya implementa cualquier paquete o curso con base en el estándar SCORM RTE, se agregaron las funciones necesarias para la recuperación

y almacenamiento de los datos persistentes, esto evita tener que modificar el código de las paginas Web para considerar los servicios de persistencia.

Otra de las ventajas que trae el incorporar las llamadas al servicio de persistencia a través de la implementación del API de SCORM RTE, tiene que ver con la seguridad en las llamadas a las funciones de persistencia.

Al contemplar los protocolos de comunicación, existe la seguridad de que cada llamada para recuperar los datos persistentes es seguida de la inicialización de la comunicación o inicio de la sesión de comunicación.

Así como también cada llamada para almacenar los datos persistentes es previa a el termino de la sesión de comunicación, lo que proporciona cierto grado de confiabilidad en el envío y recepción de la información.

Para habilitar la comunicación a través del modelo utilizado por el SCORM RTE, fue necesario extender la clase ServletProxy, la cual se encuentra en el paquete org.adl.samplerte.cliente, esta clase implementa el patrón de diseño Proxy, el cual trabaja como un concentrado de objetos para controlar el acceso a estos. Para aprovechar esta característica se agrego la funcionalidad para que se hicieran las redirecciones o los accesos al objeto que recibe las peticiones para la persistencia, en este caso a un objeto de la clase LMSPersistenceServlet, este objeto recibe un objeto de tipo LMSPersistenceRequest, el cual incluye la información requerida para llevar a cabos los procesos solicitados en la petición y una vez solicitados regresa un objeto de tipo LMSPersistenceResponse que contiene los resultados del procesamiento. En la figura 4.6 se ilustra con un diagrama de secuencias dicho comportamiento, y en la figura 4.7 se muestran las relaciones que existen entre estas clases.

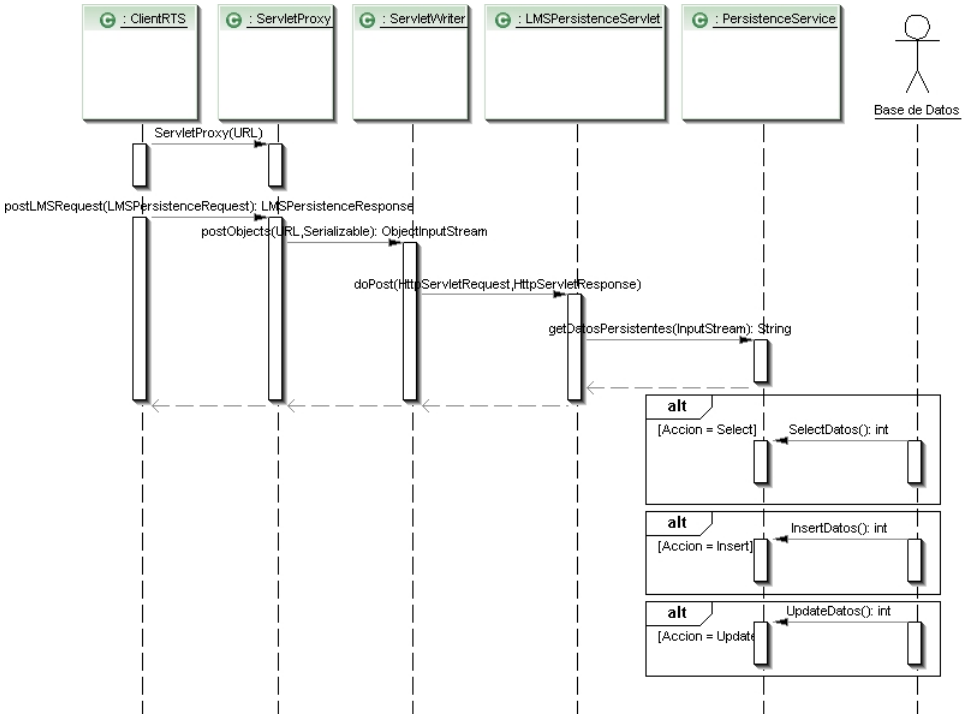


Figura 4.6: Diagrama de Secuencia: Comunicación con el Servicio de Persistencia

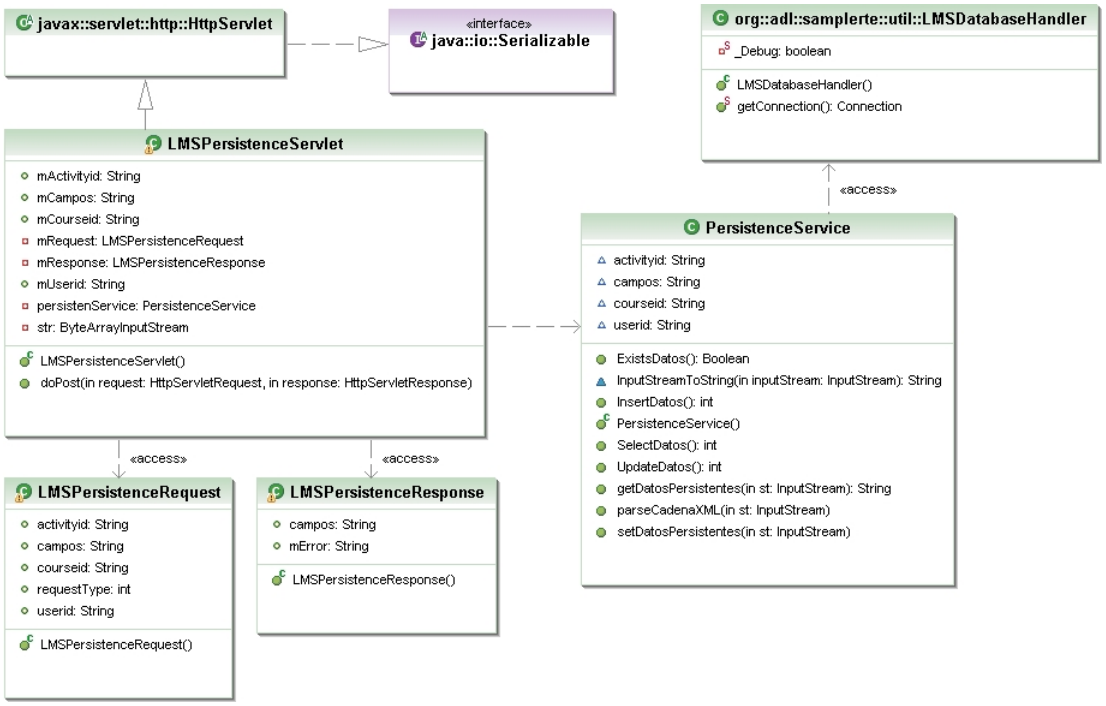


Figura 4.7: Diagrama de Clases: Estructura de clases para el mecanismo de comunicación

Del lado del cliente se realizaron las modificaciones necesarias en primer lugar para dar soporte al modelo adecuado de comunicación el cual incluye los mecanismos necesarios para solicitar los servicios de persistencia del lado del servidor, además del soporte para comunicaciones, se proporcionó de la funcionalidad necesaria para que las funciones del API pudieran identificar cuando se trataba de una petición al servicio de persistencia y no una petición al mecanismo de almacenamiento de información contemplada en el estándar, de tal forma que no se alterara el comportamiento original.

Para lograr este comportamiento se agregó una adecuación al modelo de datos, es decir, un elemento no contemplado en la instancia del API, este nuevo elemento al no identificarse con el modelo de datos provoca un error 401 el cual describe que se ha pasado como parámetro un elemento indefinido y no reconocido por la instancia del API a la función getValue() o en el segundo caso cuando se pasa como primer parámetro de la función setValue(). A partir de que se provoca este error con el elemento no reconocido y que en este caso se trata de la invocación de los servicios de persistencia de los datos, se identifica el tipo de petición, realizando el procesamiento necesario para enviar la petición al servicio de persistencia y enviando la respuesta al API Wraper (JavaScript) para su procesamiento, donde también se identifica que se trata de una petición del servicio de persistencia y ejecuta las rutinas necesarias, ya sea para recuperar información o enviar en información al servidor. En la figura 4.8 podemos ver el comportamiento con un diagrama de secuencias y en la figura 4.9 se muestran las funciones agregadas a la instancia del API para soportar el modelo de comunicaciones adecuado.

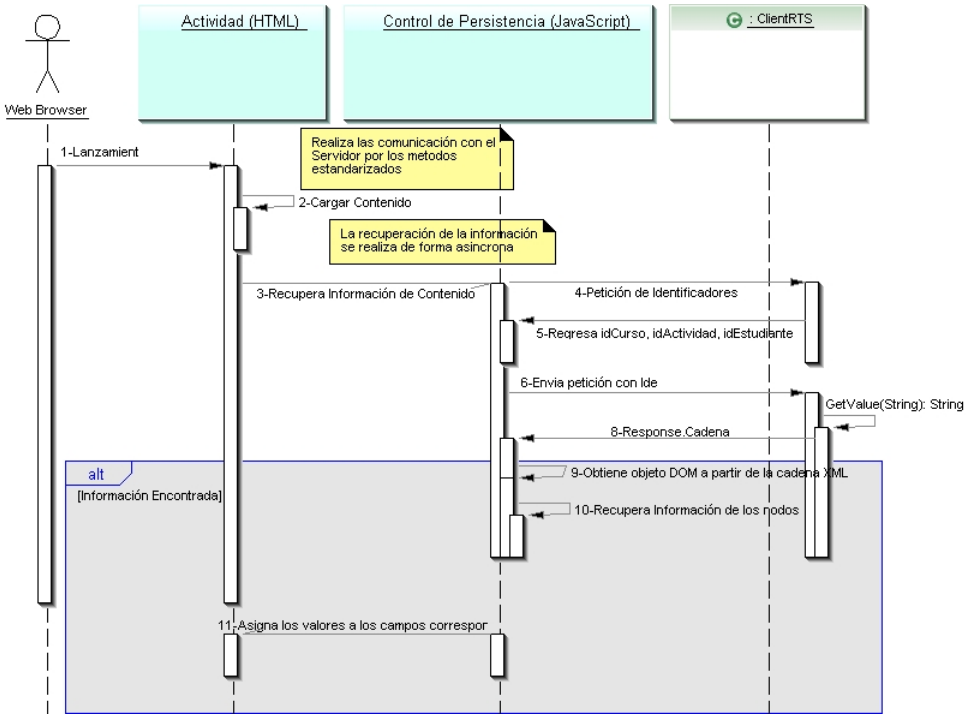


Figura 4.8: Diagrama de Secuencia: Comunicación del lado del cliente

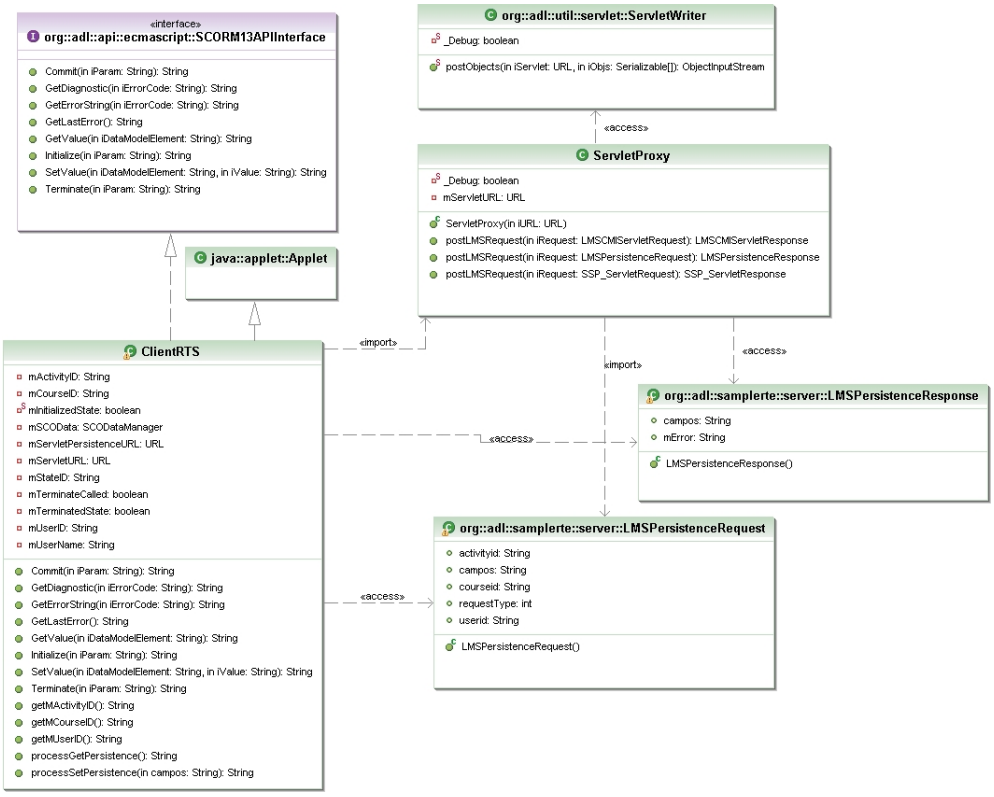


Figura 4.9: Diagrama de Clases: Estructura de Clases para soportar la comunicación del lado del cliente

Una vez que la comunicación se establece y la petición llega a los servicios de Persistencia ésta es procesada. Los servicios de persistencia son implementados por la clase PersistenceService, la cual a través de rutinas SQL y conexiones a la base de datos recupera la información o en su defecto la inserta en la base de datos. En dado caso que ya exista un registro para un curso, actividad y usuario en particular, se ejecuta una rutina para actualizar dicho registro. Esta clase además de encargarse de establecer la comunicación con la base de datos y ejecutar rutinas SQL, convierte los datos persistentes en cadenas de bytes para su almacenamiento o en dado caso de que se tratase de una recuperación convierte la cadena de bytes en cadena de texto, la cual se incorpora al XML de respuesta y se envía de regreso al mecanismo de comunicación para ser procesada por el cliente. En la figura 4.10 se muestra el diagrama de las tablas que conforman la base de datos, donde la tabla PersistenceInfo almacena los datos persistentes.

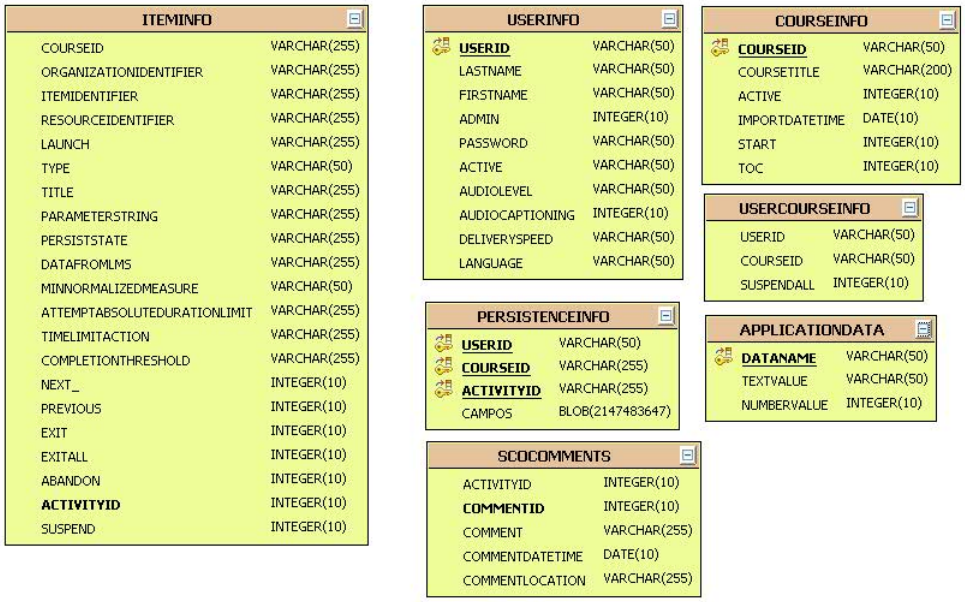


Figura 4.10: Diagrama de las clases de la base de datos

4.6 Plan de Entrega (Segunda Iteración)

4.6.1 Iteración 2

Segunda Versión Consta de 2 historias de Usuario

1. Obtener la Información del Contenido de Aprendizaje – 2 puntos
   - 3 Tareas:
     (a) Crear Estructuras de Comunicación para el servicio de Persistencia (.5 puntos)
     (b) Adecuar el modelo de comunicación existente (1 punto)
     (c) Actualizar la Implementación del API al modelo de comunicación (.5 puntos)
2. Almacenamiento de la información del Contenido de Aprendizaje
   - 2 Tareas:
     (a) Diseñar y Crear las tablas de la Base de Datos (1 punto)
     (b) Desarrollar los servicios de Persistencia (2 puntos)

Pruebas de Aceptación
Historia 3:

- Enviar datos de prueba desde el cliente a través de los mecanismos de comunicación implementados y recibirlos del lado del servidor.
- Enviar información desde el servidor a través de los mecanismos de comunicación implementados y recibirlos del lado del cliente.

Historia 4:

- Almacenar información desde los servicios de persistencia en la base de datos y recuperar dicha información de acuerdo a los tipos de datos y nombres de tablas y columnas especificados en el diseño de la base de datos.

Incidencias

- Hubo cambios en la estructura de la cadena con formato XML, puesto que el esquema que seguía para almacenar los campos, con su nombre, valor y tipo no correspondían para aquellos controles de tipo Checkbox y OptionButton, por lo que se modificó la rutina para crear la estructura de los datos a persistir diferenciando el valor almacenado dependiendo del tipo de control, con la finalidad de que en la recuperación de los datos se identifique el tipo de control, almacenar ya sea el valor del campo, si se tratara de un control tipo text o en caso de tratarse de un CheckBox u OptionButton poner su atributo *enabled* a "*true*" en caso de que así hubiese sido almacenado. Por lo tanto, la rutina para desplegar los datos en la pagina HTML también sufrió modificaciones para apegarse a estas restricciones.
- Para resolver el problema de la incompatibilidad de las funciones GetElementById(), se recomienda especificar a los controles de entrada de datos de la pagina HTML tanto el nombre y el ID, siendo el mismo valor en ambos casos.
- Es probable que en futuras iteraciones se contemple la necesidad de especializar los métodos GetValue() y SetValue() para recuperar y almacenar datos desde medios diferentes a los controles de entrada de una pagina HTML.

4.7 Tercera Iteración

4.7.1 Tareas de Desarrollo para la Historia de Usuario: 5

Tarea	
Numero tarea: 1	•Numero historia: 5
Nombre tarea: Actualización de la base de datos para el soporte de Esquemas de Persistencia	
Tipo de tarea: Diseño/Desarrollo	Puntos Estimados: 1
Fecha inicio: --/--/----	Fecha fin: --/--/----
Programador Responsable: José de Jesús Esparza Flores	
Descripción: Crear las tablas y catálogos necesarios para registrar los esquemas de persistencia.	

Tabla 4.19: Tarea de Desarrollo: Actualización de la base de datos para el soporte de Esquemas de Persistencia

Tarea	
Numero tarea: 2	•Numero historia: 5
Nombre tarea: Actualizar el mantenimiento de los cursos para contemplar la asignación de los Esquemas de Persistencia	
Tipo de tarea: Diseño/Desarrollo	Puntos Estimados: 2
Fecha inicio: --/--/----	Fecha fin: --/--/----
Programador Responsable: José de Jesús Esparza Flores	
Descripción: Modificar el mantenimiento de los cursos para asignar algún de los esquemas registrados y presentar las páginas que permitan especificar la configuración de cada uno de los esquemas de persistencia.	

Tabla 4.20: Tarea de Desarrollo: Actualización de la base de datos para el soporte de Esquemas de Persistencia

4.7.2 Tarea de Desarrollo para la Historia de Usuario: 6

Tarea	
Numero tarea: 1	•Numero historia: 6
Nombre tarea: Adecuar el LMS para dar soporte a los tipos de persistencia que integran cada uno de los Esquemas	
Tipo de tarea: Diseño/Desarrollo	Puntos Estimados: 2
Fecha inicio: --/--/----	Fecha fin: --/--/----
Programador Responsable: José de Jesús Esparza Flores	
Descripción: Identificar y adecuar cada uno de los procesos relacionados con el control de la persistencia, con el fin de dar soporte a los tipos de persistencia que pudiera requerir un curso determinado, las adecuaciones tienen que hacerse tanto del lado del cliente, como del lado del servidor, con la restricción de desarrollarlo lo suficientemente transparente para aquellos cursos que no requerían o implementen algún esquema de persistencia.	

Tabla 4.21: Tarea de Desarrollo: Actualización de la base de datos para el soporte de Esquemas de Persistencia

Elementos de Diseño para la Tercera Iteración
En la Tercera iteración se diseñaron y se crearon las tablas de bases de datos necesarias para soportar los esquemas de persistencia. En primera instancia, se crearon los catálogos de esquemas, es decir, una tabla en la que cada registro representa un esquema de persistencia y cada columna representa los tipos de persistencia que implementa el esquema. Esto se hizo con la finalidad de contar con un catálogo pre-establecido de esquemas para uso genérico. Se creó otro catálogo para describir el tipo de persistencia "Con Vencimiento" donde se especifica si el estado del objeto de aprendizaje se debe eliminar al cerrar la sesión o al finalizar el curso.

Para asociar un Esquema de Persistencia con cada uno de los Assets (paginas Web), se creó una tabla que relaciona id del curso, el id de la actividad y un campo de tipo Entero para especificar si el Esquema esta activo para la actividad de un determinado curso.

En el caso de que el Esquema considere la persistencia parcial, se creó una tabla que almacena el id del curso, el id de la actividad y un campo de tipo BLOB que almacena una cadena de tipo XML, en la cual se especifican los campos que no serán parte del estado persistente del Asset. La estructura del documento XML que se muestra en el código de 4 indica como se almacenan los campos que serán discriminados.

Documento XML 4 Estructura XML para especificar el Estado Parcial de un Objeto de Aprendizaje

```
<?xml version = '1.0' encoding='UTF-8'?>
    <campos>
        <campo>T2</campo>
        <campo>NN1</campo>
        <campo>NN3</campo>
        <campo>NN3</campo>
    </campos>
```

La figura 4.11 muestra el diagrama de la base de datos existente con las tablas correspondientes al registro y la asignación de los Esquemas de Persistencia.

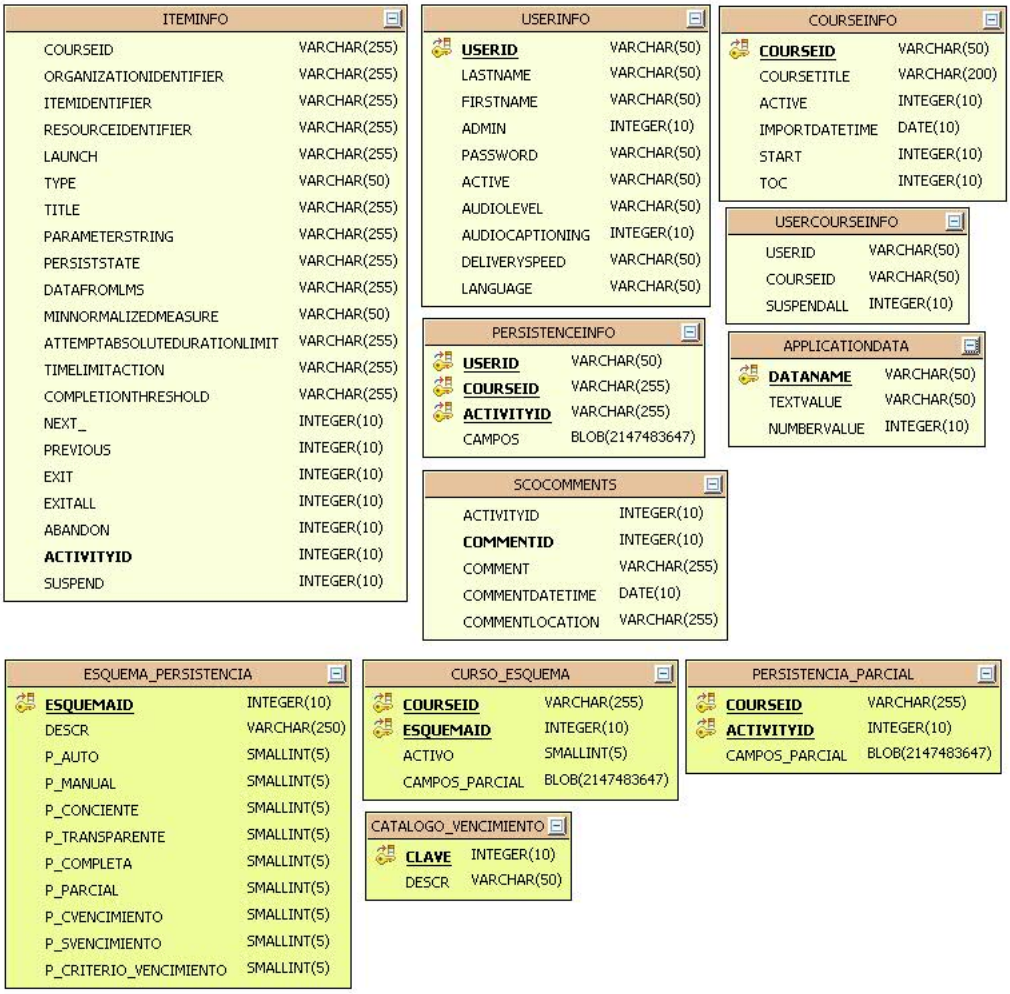


Figura 4.11: Diagrama de la base de datos con las tablas para Esquemas de Persistencia

El LMS cuenta con un modulo para el mantenimiento de los cursos, donde el administrador puede registrarlos, eliminarlos y agregar comentarios a los cursos registrados. Este modulo por motivos de practicidad se dejó fuera del LMS que despliega los cursos al usuario final, ya que solo el administrador tiene acceso a estas opciones. Es en este modulo donde se agregó la funcionalidad para darle el comportamiento de persistencia de estado a los cursos registrados. El diagrama 4.12 de secuencias muestra el comportamiento que se lleva a cabo para registrar algún esquema de persistencia al curso seleccionado.

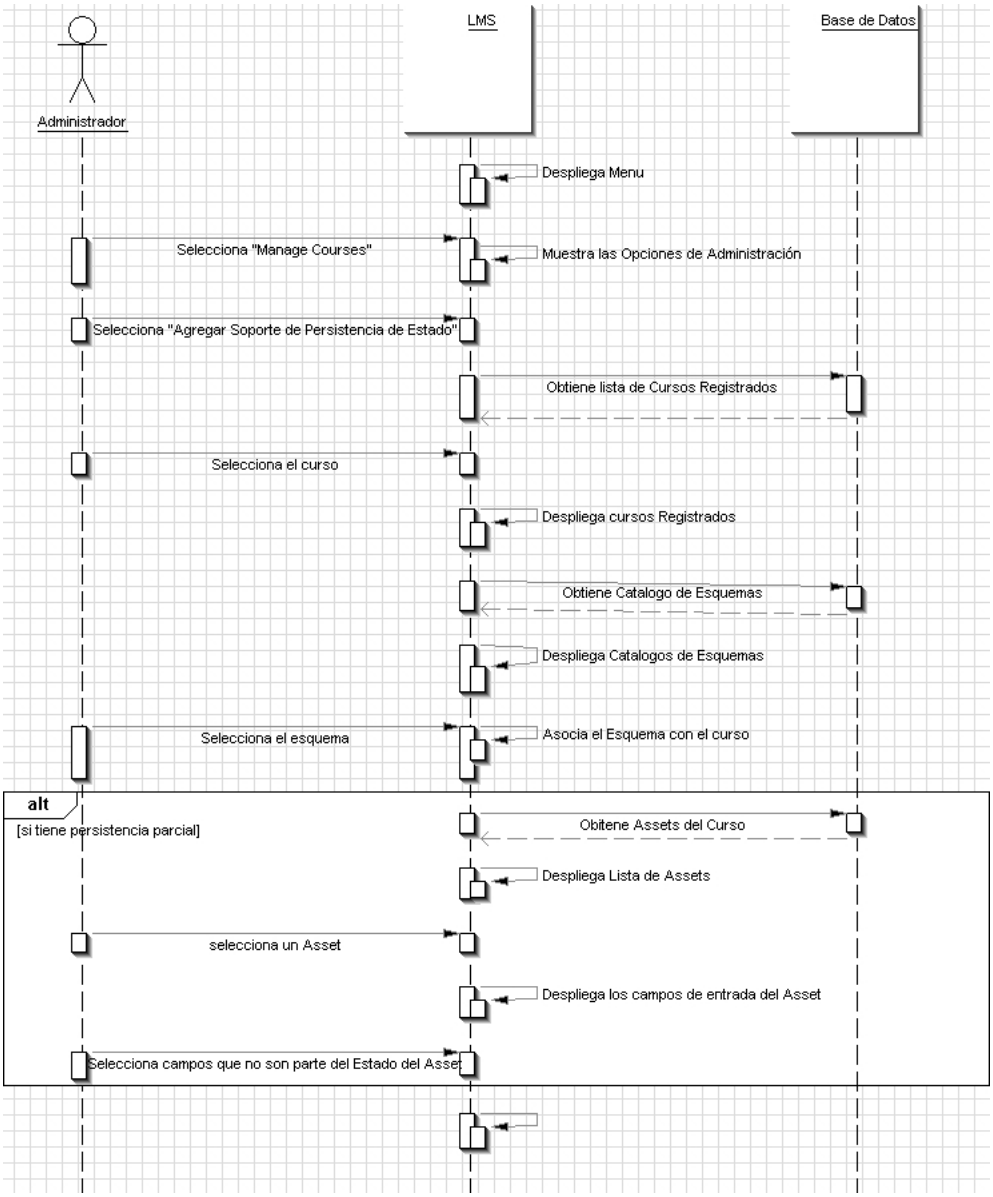


Figura 4.12: Diagrama de Secuencia de Asignación de Esquema de Persistencia

Puesto que el LMS incorpora un sistema de almacenamiento embebido, es decir, solo puede ser accedido desde el JVM que lo arranca, es necesario replicar la base de datos del modulo de administración de cursos, al LMS que despliega los cursos al usuario final.

Los esquemas de persistencia requieren de una serie de modificaciones al comportamiento del LMS, los cuales van desde el despliegue del Asset (Pagina Web), hasta el momento de replegar o descargar el objeto de aprendizaje del LMS, por lo que es importante primeramente identificar o dar a conocer el comportamiento y los elementos que intervienen en el proceso de implementación de los esquemas de persistencia. El diagrama de secuencias de la figura 4.13 muestra de una manera más clara el escenario en el que ocurre este proceso.

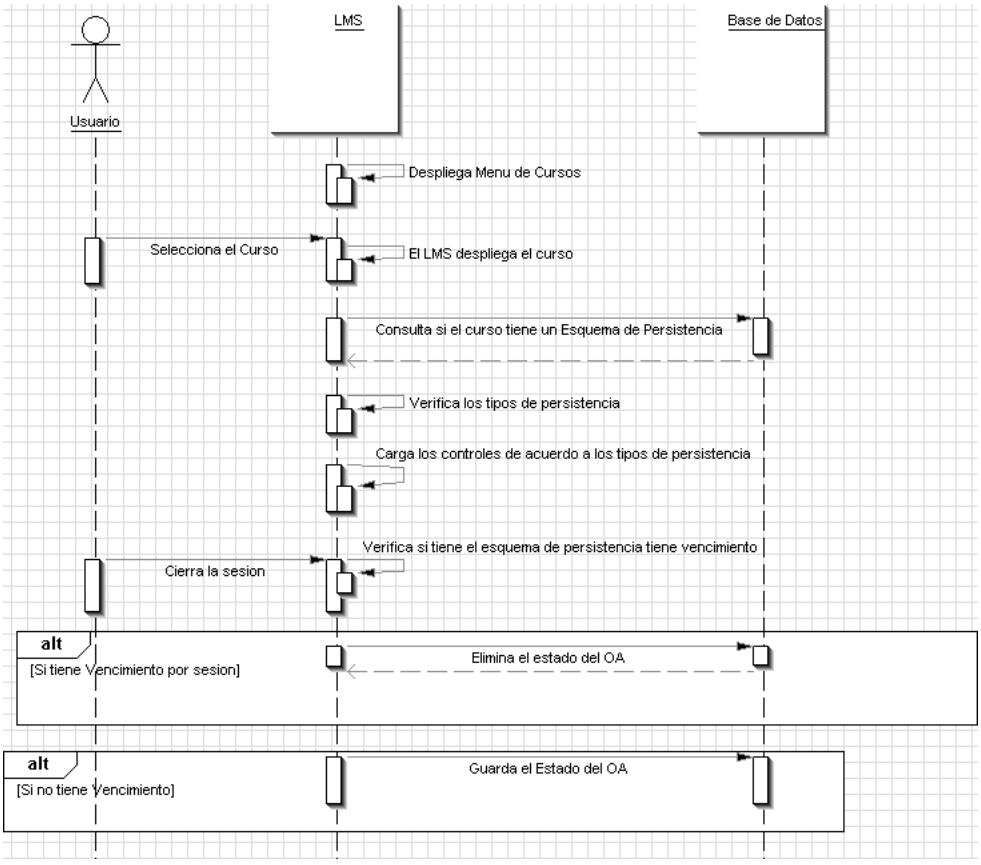


Figura 4.13: Diagrama de Secuencia de Asignación de Esquema de Persistencia

4.8 Plan de Entrega (Tercera Iteración)

4.8.1 Iteración 3

Tercera Versión Consta de 2 historias de Usuario

1. Registrar Cursos con un Esquema de Persistencia Asociado – 4 puntos
   - 2 Tareas:
     (a) Actualización de la base de datos para el soporte de esquemas de persistencia (2 puntos)
     (b) Actualizar el mantenimiento de los cursos para contemplar la asignación de los esquemas de persistencia (2 puntos)
2. Implementación de Esquemas de Persistencia en el LMS
   - 1 Tarea:
     (a) Adecuar el LMS para dar soporte a los tipos de persistencia que integran cada uno de los esquemas (2 puntos)

Pruebas de Aceptación
Historia 5:

- Registrar un curso en LMS con el modulo de Administración
- Asignar un esquema con características de persistencia parcial después de haberse registrado el curso
- Registrar los campos que no formarán parte del estado parcial
- Modificar el esquema de persistencia
- Modificar los campos del estado parcial del curso

Historia 6:

- Desplegar un curso con un esquema de persistencia asignado
- Cargar los controles dentro del LMS, de acuerdo a las características del esquema de persistencia
- Generar el estado de un Asset
- Guardar el estado del Asset
- Recuperar el estado del Asset
- Validar que el estado del Asset corresponda con el estado parcial registrado en el modulo de Administración

Incidencias

- Puesto que el modulo de administración del LMS no es expuesto al usuario final, se creo una versión solo con opciones de mantenimiento para los cursos registrados. Al realizar esta separación estamos hablando de que tenemos dos bases de datos idénticas pero separadas, la primera, es la que tiene el curso registrado y la asignación con el esquema de persistencia, la segunda que utiliza el LMS para desplegar los cursos al usuario final. Por tal motivo se genera un conflicto al empatar estas bases de datos, es decir, al momento de presentarle un curso con características de persistencia al usuario final, es necesario copiar la base de datos del modulo de administración al LMS que despliega los cursos.

Capítulo 5

Clasificación de tipos de persistencia y esquemas de persistencia

Con el objetivo de proporcionar a los diseñadores instruccionales un marco de referencia que muestre el alcance y facilite el entendimiento de las distintas opciones de persistencia y por consiguiente, ayude a la toma de decisiones al momento de seleccionar el conjunto de elementos que permitan satisfacer las necesidades de persistencia de estado en los objetos de aprendizaje (OA) de acuerdo a su diseño; a continuación se presenta cada uno de los tipos de persistencia, su clasificación de acuerdo a su comportamiento y los esquemas de persistencia como una implementación del conjunto de funcionalidades que ofrecen cada uno de los tipos de persistencia.

5.1 Tipos de Persistencia

Para definir los tipos de persistencia, es necesario mencionar que estos tienen como base los mecanismos de persistencia desarrollados en el capitulo cinco, los cuales cubren todas las especificaciones técnicas que deben ser implementadas tanto en el OA como en el LMS que lo contiene, con el fin de establecer los medios de comunicación que permiten el intercambio y almacenamiento de información relacionada con el estado del OA. Entonces se puede definir los tipos de persistencia como una especialización del comportamiento de los mecanismos básicos de persistencia donde se toman en cuenta ciertos criterios para recuperar y almacenar los datos que conforman el estado de un OA.

Cabe mencionar que de acuerdo al estándar de SCORM, un objeto de aprendizaje (Sharable Content Object, SCO) esta conformado por uno o mas recursos de aprendizaje (Assets), los cuales pueden ser paginas html, Applets, etc. En este caso los recursos de aprendizaje implementan los mecanismos de persistencia.

A continuación se listan cada uno de los tipos de persistencia propuestos y que fueron desarrollados de acuerdo a las necesidades más comunes para el almacenamiento del estado de un SCO. Posteriormente en este mismo capitulo se describen las categorías en las cuales se agrupan los siguientes tipos de persistencia.

Persistencia automatizada.- La persistencia automatizada realiza los procesos de recuperación y almacenamiento de información de acuerdo a la ocurrencia de eventos disparados por el propio Asset. Una vez que el LMS carga el SCO y se presentan los Assets al usuario a través del navegador Web, el Asset dispara un evento que realiza las acciones necesarias para establecer la comunicación con el LMS y recuperar su estado el cual fue previamente

almacenado o bien, antes de descargar el Asset con el cual el usuario esta trabajando, se dispara un evento que realiza las acciones necesarias para establecer la comunicación con el LMS el cual envía el estado del Asset a la base de datos para su almacenamiento; la principal característica de este tipo de persistencia es que el usuario no interviene en la ocurrencia de dichos eventos.

Persistencia manual.- Las acciones para recuperar el estado persistente de un Asset también pueden ser ejecutadas tras un evento disparado por parte del usuario, donde este puede o no estar consiente de la ocurrencia de las acciones relacionadas con los procesos de recuperación o almacenamiento del estado de un Asset. El usuario debe disparar el evento que ejecute las acciones necesarias para hacer persistente el estado de un Asset, el cual solo podrá ser ejecutado durante el tiempo en el que el SCO este cargado dentro del LMS.

Persistencia conciente por el usuario.- Con el fin de que el usuario sea conciente de los procesos de recuperación y almacenamiento del estado del Asset con el cual esta trabajando en su navegador Web, el LMS realiza notificaciones para darle a conocer al usuario que se ha recuperado el estado del Asset desplegado, o en su defecto, las causas por las cuales no ha sido posible realizar con éxito dicho proceso; de igual forma se le notifica al usuario que el estado del Asset ha sido almacenado en la base de datos o las causas por las cuales dicho proceso no se realizo con éxito.

Persistencia transparente para el Usuario.- Los procesos de recuperación o almacenamiento del estado persistente de un Asset, se realizan de forma transparente para el usuario. Independientemente si la carga del estado del Asset fue automática o manual, al usuario no se le notifica de ningún evento ocurrido tras este proceso, salvo la ocurrencia de algún error relacionado con la persistencia fallida del estado del Asset.

Persistencia Completa.- El Asset no tiene especificados ningún tipo de restricción que determine cuales de sus elementos serán tomados en cuenta para su estado, por lo que considera o incluye todos sus elementos para hacerlos persistentes.

Persistencia Parcial.- El Asset tiene configurada una carga o recuperación parcial de su estado, como resultado de la especificación de sus elementos que serán considerados para la persistencia de su estado, es decir, no se recuperan o se almacenan los datos que no este señalados para determinado propósito.

Persistencia con Vencimiento.- El Asset puede incluir parámetros de configuración que permitan especificar mediante condiciones cuando la persistencia de su estado puede ser recuperado o almacenado y cuando no, pueden ser condiciones temporales o relacionadas con propósitos establecidos por el autor de la actividad de aprendizaje.

Persistencia Sin Vencimiento.- El Asset no tiene especificado ningún parámetro de vencimiento para la persistencia del estado de la actividad, por lo tanto recupera y almacena la información siempre que sea posible.

5.2 Clasificación de los Tipos de Persistencia

Con el fin de ofrecer una mayor cantidad de opciones en la recuperación y almacenamiento de datos y así mismo agrupar y generalizar las características de comportamiento comunes entre los tipos de persistencia, se realizó una clasificación, la cual permite a los diseñadores instruccionales identificar de manera mas clara y amplia la naturaleza de los tipos de persistencia y los escenarios en los cuales puedan ser de utilidad, creando así combinaciones mas ricas y flexibles.

5.2.1 Por el control del proceso de persistencia

Los tipos de persistencia incluidos en esta clasificación se caracterizan por controlar los procesos relacionados con la recuperación y almacenamiento del estado de un Asset. Dicho de otra manera, se puede especificar en un objeto de aprendizaje Asset la forma en la cual se van a disparar los eventos que den pie a la persistencia de su estado.

La ocurrencia de mencionados eventos se puede realizar de forma automatizada, donde el usuario no se hace partícipe en el proceso de persistencia. También los eventos pueden ocurrir por el disparo producido por el usuario. En este caso no solo se involucra el usuario, sino que, se hace responsable del control de la persistencia de estado del Asset cargado en su navegador Web.

Tipos de Persistencia:

- Persistencia Automatizada
- Persistencia Manual

5.2.2 Por conciencia de uso

En esta clasificación se agrupan los tipos de persistencia que están relacionados con la conciencia de uso por parte del usuario, donde los procesos de almacenamiento y recuperación por los que pasa la información generada dentro de un Asset en su sesión de trabajo le pueden ser notificados.

Hacer conciente al usuario de la persistencia de estado de los objetos de aprendizaje, contribuye al aumento de las sensaciones de seguridad y confianza en el uso de los medios electrónicos para su aprendizaje.

Por otro lado, la conciencia de uso también esta relacionada con la transparencia, donde los procesos de persistencia de la información que se genera en una sesión de trabajo, no son notificados al usuario y por tal motivo no es conciente de los estados por los que pasa el proceso de recuperación y almacenamiento del estado del Asset con el cual se encuentra trabajando.

Tipos de Persistencia:

- Persistencia conciente para el usuario
- Persistencia transparente para el usuario

5.2.3 Por el estado del Objeto de Aprendizaje

Un objeto de aprendizaje esta conformado por un conjunto de atributos, los cuales obtienen determinado valor como producto de la interacción con el usuario dentro de una sesión de trabajo, produciendo así, el estado de un OA. Son los valores del conjunto de atributos los que son almacenados y recuperados de la base de datos para interactuar con estos en sesiones de trabajo posteriores del usuario.

El estado del objeto de aprendizaje puede estar conformado por los valores para el conjunto total de sus atributos, o bien, dependiendo de las necesidades o requerimientos instruccionales que obedecen a determinado comportamiento, el diseñador instruccional puede especificar cuales atributos conformaran el estado del objeto de aprendizaje.

Tipos de Persistencia:

- Persistencia total de estado
- Persistencia parcial de estado

5.2.4 Por Temporalidad

La temporalidad se refiere a la característica de los tipos de persistencia que controlan el acceso al estado de un objeto de aprendizaje entre una sesión y otra, es decir, permite especificar las condiciones que deben cumplirse en determinado momento para acceder a los datos almacenados, estas condiciones pueden obedecer a restricciones de diseño de acuerdo con los requerimientos del objeto de aprendizaje.

Tipos de Persistencia:

- Persistencia con Vencimiento
- Persistencia sin Vencimiento

En la figura 5.1, se presenta un diagrama que muestra a los mecanismos de persistencia como la base de la cual se derivan las distintas clasificaciones de tipos de persistencia de acuerdo a su comportamiento.

Figura 5.1: Diagrama de la Clasificación de Tipos de Persistencia.
Los rectángulos de color gris representan los tipos de persistencias incluidos en cada una de las categorías.

5.3 Esquemas de Persistencia

Podemos definir como esquema de persistencia a la implementación de uno o mas tipos de persistencia compatibles entre si y de diferentes clases, los cuales conforman un conjunto de funcionalidades aplicadas a objetos de aprendizaje con características de persistencia en su estado.

Los esquemas al estar integrados por los tipos de persistencia, pueden ofrecer soluciones de persistencia a la medida de las necesidades de los OA, o bien, el LMS puede ofrecer un catálogo de esquemas de persistencia donde se combinan los distintos tipos de persistencia explicados anteriormente.

A continuación se presenta en la tabla 5.1 una propuesta base del catalogo para esquemas de persistencia que se implementarán dentro del LMS que posee los medios o mecanismos necesarios para establecer la comunicación que auxilia al proceso de persistencia.

ESQUEMAS DE PERSISTENCIA								
CATEGORÍAS	CONTROL DE PROCESO DE PERSISTENCIA		CONCIENCIA DE USO		ELEMENTOS DE ESTADO		TEMPORALIDAD	
TIPOS DE PERSISTENCIA	AUTOMATIZADA	MANUAL	CONCIENTE	TRANSPARENTE	PARCIAL	TOTAL	CON VENCIMIENTO	SIN VENCIMIENTO
ESQUEMAS								
PROTOTIPO 1 (IMPLEMENTADO)	X			X		X		X
PROTOTIPO 2		X	X			X		X
PROTOTIPO 3	X			X	X		X	
PROTOTIPO 4		X		X		X	X	

Tabla 5.1: Esquemas de Persistencia

Capítulo 6

Análisis y Metodología de Uso de los Esquemas de Persistencia

6.1 Análisis Comparativo con Propuestas Similares

Esta sección tiene el objetivo de dar a conocer desde un punto de vista critico las características de una posible solución a la persistencia de estado de los objetos de aprendizaje, la cual se desarrolla dentro de este documento de tesis, realizando una comparativa entre una propuesta descrita en [Kassahun et al., 2006], la propuesta presentada por SCORM y los Esquemas de persistencia, resaltando las ventajas y desventajas de cada una de las propuestas con el fin de conocer un mejor panorama de la situación actual referente a la persistencia de estado de los objetos de aprendizaje.

Como se menciona en [Sessink et al., 2003], en la actualidad los sistemas de administración de aprendizaje aun tienen muchas limitantes, estándares como SCORM han venido a cubrir o satisfacer una gran mayoría de estas necesidades, sin embargo es posible identificar algunos requerimientos adicionales de funcionalidad, entre estos esta "La Recuperación del historial y del estado"; el alcance de este concepto involucra la posibilidad de obtener en un momento dado las sesiones de trabajo históricas, así como el estado de cada una de estas sesiones. Actualmente el estándar de SCORM (ADL, 2004) ofrece elementos dentro de su modelo de datos tales como *cmi.launch_data*, *cmi_suspend_data* para el almacenamiento de información, pero el uso de estos elementos es muy limitado por varias razones; entre las limitaciones mas importantes esta el tamaño de los datos que pueden ser almacenados, el cual es muy reducido.

Una de las pocas propuestas que atacan la problemática de la recuperación del estado de los objetos de aprendizaje es la que se menciona y explica en [Kassahun et al., 2006], donde ofrecen un componente que encapsula toda la funcionalidad referente a la recuperación y almacenamiento de los datos relacionados con el objeto de aprendizaje; es una propuesta bastante robusta ya que puede ser implementada para sistema de administración de aprendizaje (LMS) basado en el estándar SCORM, el cual es mas recomendable, pero también ofrece mecanismos de comunicación para aquellos LMS que no implementen el estándar antes mencionado. La propuesta llamada DBLink esta conformada por un plug-in que recibe las peticiones por parte del LMS para las transacciones de información, ofrece un conjunto de interfaces que tienen que ser configuradas para trabajar con un servidor de base de datos, por lo que resulta necesario especificar el URL del servidor, puerto y credenciales de identificación; otra de las características de esta propuesta es la exposición de un conjunto de elementos que extienden el modelo de datos de SCORM, esta extensión se utiliza para hacer las configuraciones de comunicación con el servidor de base de datos y para realizar transacciones a través de llamadas a sentencias SQL desde los objetos de aprendizaje para recuperar o almacenar la información de su estado.

Desde un punto de vista crítico, esta propuesta tiene varias ventajas, la primera es que es mas robusto que la propuesta que ofrece el estándar de SCORM, tiene la posibilidad de delegar las tareas de DBMS a un servidor de base de datos y no propiamente al LMS, puede trabajar con varios DBMS para almacenar diferentes tipos de información, así como también, utiliza el API de SCORM para realizar las tareas de recuperación y almacenamiento de datos.

Por otro lado, también podemos mencionar algunas características que limitan la funcionalidad de esta solución, entre las cuales resalta la necesidad de extender el modelo de datos del estándar, esto obliga al autor o diseñador instruccional a conocer las restricciones de cada uno de los elementos extendidos del modelo.

Cabe destacar que el perfil de un autor o diseñador instruccional no es propiamente el de un tecnólogo, entonces resulta evidente que algunos conceptos no serán del todo claros. Otra de las características que limitan el funcionamiento de la propuesta es la necesidad de escribir sentencias de código SQL dentro de los objetos de aprendizaje donde el autor tiene que modificar sus objetos ya existentes para incluir estas sentencias. Nuevamente nos vemos en la situación donde el autor o diseñador instruccional necesita tener conocimientos mas avanzados de computación, en especial con temas relacionados con las bases de datos.

Los Esquemas de Persistencia presentados en este documento, tienen como propósito principal el ofrecer al autor o diseñador instruccional un conjunto de opciones de persistencia que puedan ser integrados en un solo esquema de acuerdo a sus necesidades instruccionales. Tratando de delegar al LMS las tareas de recuperación y almacenamiento de la información, con el objetivo de que el autor o diseñador instruccional no se vea en la necesidad de conocer detalles ticónicos para proveer a sus objetos de aprendizaje las características necesarias para mantener y recuperar su estado en un momento determinado.

De igual forma se busco de la mejor manera evitar extender el modelo de datos de SCORM [Dodds, 2006a], lo que significa que el autor o diseñador instruccional no tiene que aprender o tratar de conocer mas elementos del estándar, esto permite que el autor o diseñador instruccional no tenga que incluir sentencias complejas dentro de sus recursos o paginas HTML para almacenar la información surgida de la interacción entre el usuario y los elementos del objeto de aprendizaje. Obteniendo de esta forma un método sencillo y simple de implementar la persistencia de estado sin tener que lidiar con problemas ajenos a los relacionados con los objetos de aprendizaje.

Actualmente los esquemas de persistencia implementados en el LMS solo recuperan el estado de aquellos objetos de aprendizaje con características de paginas HTML, ya sea desplegada como tal o como un JSP, sin embargo, se puede extender en un momento dado hacia elementos tipo Applets o Flash según las necesidades del objeto de aprendizaje.

Esta solución viene dada dentro de un LMS implementado bajo los estándar es de SCORM, por consecuente cualquier curso con base en este estándar puede ser desplegado con características de persistencia dentro de sus recursos HTML. Uno de los enfoques que se le dio al LMS va dirigido hacia la portabilidad de los cursos registrados, haciendo posible llevar los cursos de una forma desconectada, es decir, a pesar de que LMS esta bajo el esquema de aplicación Web, es posible desplegar un curso sin la necesidad de conectarse a un servidor remoto ya sea un servidor de aplicación o un servidor de base de datos, debido a que el LMS incorpora una base de datos embebida y un servicio que levanta los competentes necesarios para el despliegue del curso, permitiendo llevar el LMS a cualquier plataforma que permita ejecutar una maquina virtual Java, o bien, distribuirlo por diferentes medios, ya sea discos magnéticos, como una aplicación descargada de Internet, etc.

La solución de los esquemas de persistencia además de ofrecer un LMS con características de portabilidad, ofrece un módulo independiente que permite realizar el registro de los cursos que serán desplegados al usuario final, este modulo permite el registro de los cursos, así como también la asignación y configuración de los esquemas de persistencia a los cursos.

Una de las características notorias del LMS es la incorporación de una base de datos embebida, la cual no tiene que ser previamente configurada y evita cualquier intervención del usuario, esta característica por su naturaleza tiene la limitante de establecer una sola instancia por maquina virtual que la ejecuta, por lo que resulta necesario replicar las bases de datos a través del sistema de archivo con el fin de incorporarlas en el LMS para el usuario final.

Es importante mencionar que los esquemas de persistencia tienen cierto grado de flexibilidad que permite al autor o diseñador instruccional especificar las características de persistencia que requieren sus objetos de aprendizaje.

Si bien la propuesta de los Esquemas de Persistencia dentro de un LMS que incorpora características de portabilidad no atacan de lleno la problemática de recuperar el historial y estado de los objetos de aprendizaje que se menciona en [Sessink et al., 2003], si ofrecen una solución útil a los actuales cursos desarrollados bajo el estándar de SCORM que carecen de persistencia de estado.

6.2 Metodología de Uso para los Esquemas de Persistencia

Esta sección tiene el propósito de dar a conocer paso a paso las acciones a seguir para desplegar un curso con éxito dentro del LMS transportable, actualmente este proceso requiere de mucha labor por parte del administrador, varios de los siguientes procesos pueden ser automatizados en futuras versiones del LMS, para fines de demostración y sobre todo para percibir todo lo que involucra la puesta en marcha del LMS transportable se realizaran los procesos de forma manual.

6.2.1 Armar el Paquete del Curso

Si contamos con un curso basado en el estándar de SCORM, o bien, se esta en el proceso de composición del mismo, es necesario especificar en cada uno de los archivos HTML la cabecera especificada con el fin de que el navegador aplique las reglas correspondientes a archivos de tipo XHTML como se muestra en el código 5.

Documento HTML 5 Cabecera HTML

```
<!DOCTYPE html PUBLIC ‘‘-//W3C//DTD XHTML 1.0 Transitional//EN’’
‘‘http://wwww.w3.org./TR/xhtml1/DTD/xhtml1-transitional.dtd’’>
<html xmlns:’’http://www.w3.org/1999/xhtml’’>

    ...

</html>
```

Si nuestro Assets incluye elementos de entrada de datos tales como: inputs, checkbox, radio, etc. es necesario incluir como un recurso el archivo: APIWrapper.js, este archivo además de incluir todas las llamadas al API que el estándar especifica, incorpora los controles necesarios para el manejo de los mecanismos de persistencia. Además de incluir este archivo como un recurso del cual dependen las paginas HTML, es necesario declarar el uso del archivo APIWrapper.js en la sección de cabecera de cada una de las páginas que lo necesiten, tal como se muestra en el código 6.

Documento HTML 6 Especificación del APIWraper.js en la cabecera de los archivos

```
<html xmlns=''http://www.w3.org/1999/xhtml''>
    <head>

        <meta http-equiv=''Content-Type'' content=''text/html; charset=UTF-8''/>
        <script src=''../util/APIWrapper.js'' type=''text/javascript''></script>
        <title>División de fracciones</title>
    </head>
    <body onload=''loadPage()'' onunload=''return unloadPage()''>
        ...
    </body>
</html>
```

El uso de la sentencias “loadPage()” inicia la comunicación con el implementación del API de SCORM por lo que es importante ejecutar esta función en el evento Load de la pagina, así como también mandar llamar la función “unLoadPage()” en el evento unload de la pagina con el fin de terminar las comunicación es con la implementación del API al descargar la pagina.

Es importante también mencionar que todos los elementos de entrada antes mencionados tales como inputs, checkbox, radio, etc. tienen que tener su respectiva etiqueta de cierre, ya que son analizados por un parser para recuperar y asignar los datos contenidos. Por motivos de compatibilidad entre los navegadores mas populares Internet Explorer y Mozilla Firefox, es necesario agregar además del nombre a los tag input, el atributo id, ya que puede no funcionar adecuadamente con navegadores Mozilla Firefox, el código 7 muestra un ejemplo de este requerimiento.

Documento HTML 7 Requerimientos para las etiquetas input

```
<html xmlns=''http://www.w3.org/1999/xhtml''>
    <head>

        <meta http-equiv=''Content-Type'' content=''text/html; charset=UTF-8''/>
        <script src=''../util/APIWrapper.js'' type=''text/javascript''></script>
        <title>División de fracciones</title>
    </head>
    <body onload=''loadPage()'' onunload=''return unloadPage()''>
        <form>
            <input type="text" size="9" name="T2" id="T2"/>
        </form>
    </body>
</html>
```

Después de haber realizado las adecuaciones necesarias a cada uno de los recursos que conforman el paquete del curso, se procede a generar el archivo del curso en formato zip, de acuerdo al estándar de SCORM [Dodds, 2006b]. Esto con la finalidad de registrarlo en el modulo de mantenimiento de cursos del LMS, el cual tiene que ser previamente desplegado dentro de un servidor de aplicaciones J2EE tal como tomcat.

Para registrar el curso en el modulo de administración del LMS, es necesario acceder al sistema con credenciales de administrador tal y como se muestra en la figura 6.1.

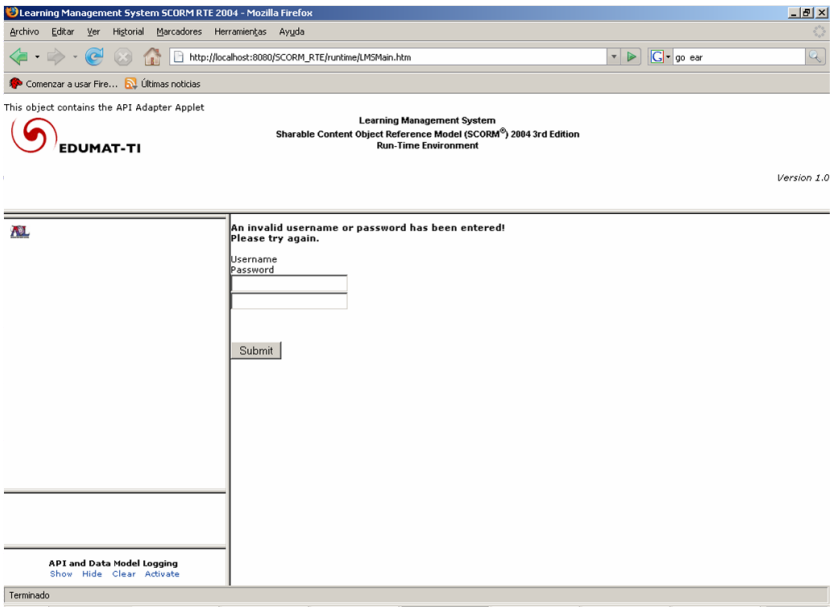


Figura 6.1: Login LMS Modulo de Administración

En el menú principal seleccionamos la opción de *"Importar Curso"*, donde posteriormente nos mostrara una pantalla en donde necesitamos especificar la ubicación del paquete zip de nuestro curso, así como también necesitamos especificar que el LMS valide que el curso cumple con el estándar para ser registrado, esta opción se recomienda para evitar problemas posteriores, la imagen 6.2 muestra un ejemplo de este caso.

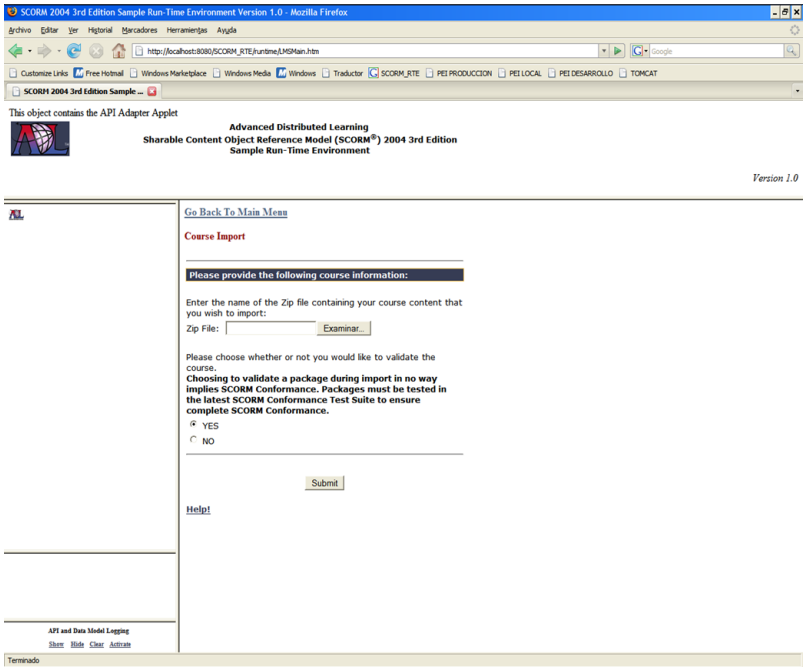


Figura 6.2: Registro de Cursos LMS Modulo de Administración

Una vez registrado el curso satisfactoriamente, procedemos a asignarle un esquema de persistencia, los esquemas de persistencia están previamente configurados con un comportamiento especifico dentro de los catálogos de la base de datos, la modificación del comportamiento de alguno de los esquemas implica, ya sea generar un esquema nuevo o bien modificar los ya existentes, cabe mencionar que estas tareas se realizan a nivel de bases de datos. Para asignarle un esquema de persistencia al curso seleccionamos del menú principal la opción de *"Administrar Curso"* dentro de este menú seleccionamos la opción que dice "Agregar Soporte para la Persistencia de Estado" como se muestra en la imagen 6.3.

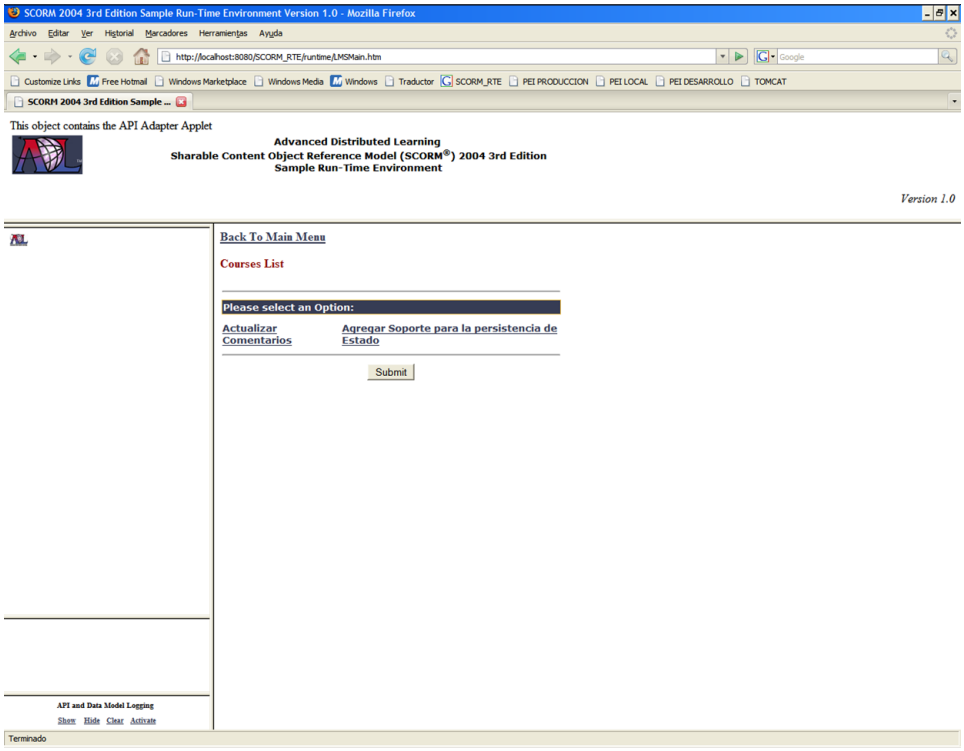


Figura 6.3: Menú de Administración de Cursos Registrados

Posteriormente, se selecciona de un menú desplegable uno de los cursos registrados a los cuales les asignaremos el esquema de persistencia, si el esquema de persistencia involucra la especificación de ciertas configuraciones para cada Asset de tipo HTML, aparecerá después de la asignación un menú desplegable que muestra dichos Assets para aplicarles las configuraciones pertinentes a cada uno, dependiendo del esquema que se haya seleccionado. La imagen 6.4 muestra la asignación del esquema de persistencia y la imagen 6.5 muestra las configuraciones que deben especificarse para aplicar satisfactoriamente un tipo de persistencia que compone el esquema, llamado persistencia parcial.

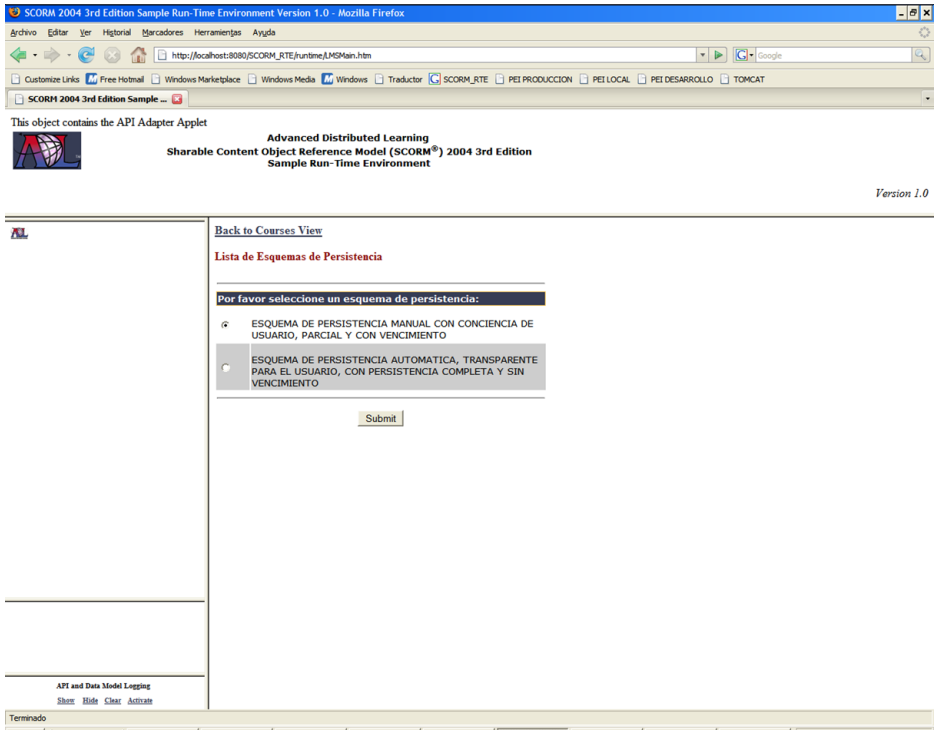


Figura 6.4: Menú de asignación de Esquemas de Persistencia

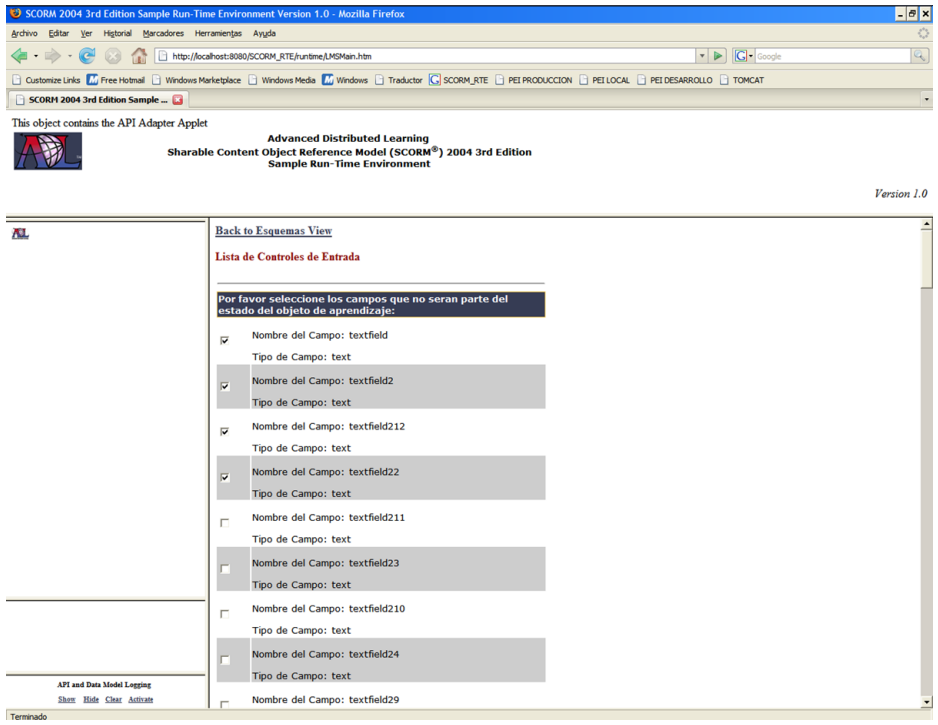


Figura 6.5: Configuración para el tipo de persistencia parcial

Una vez que tenemos el curso registrado y asignado a un esquema de persistencia, es necesario replicar un conjunto de archivos al LMS que desplegara el curso al usuario final; el LMS transportable se compone de la estructura de archivos y carpetas que se muestran en la imagen 6.6.

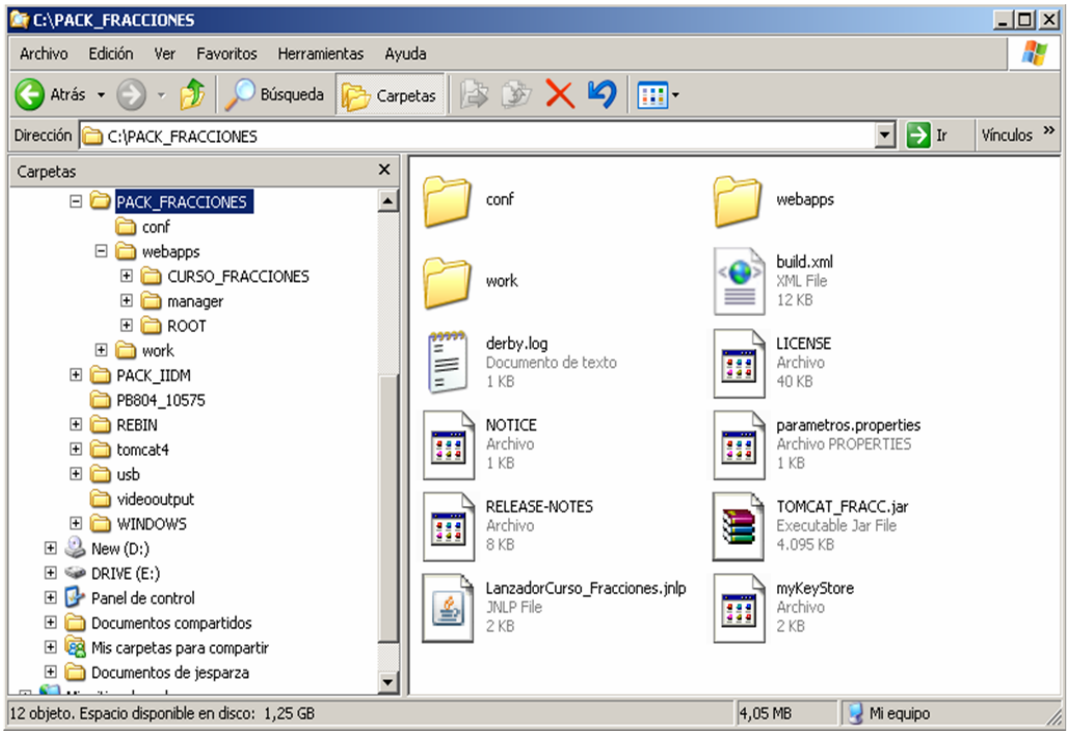


Figura 6.6: Estructura de Archivos y Carpetas LMS Transportable

La carpeta webapps contiene las carpetas de contexto del curso a desplegar, dentro de esta carpeta encontramos las siguientes carpetas:

1. CourseImport.- Contiene los archivos del curso registrado.
2. database.- Contiene los archivos de la base de datos embebida en el LMS.
3. SCORM3rdSampleRTE10Files.- Contiene información relacionada con el historial de navegación de cada uno de los usuarios registrados en el curso.

Es necesario copiar estas carpetas a la carpeta de contexto dentro del LMS transportable, así como también configurar el archivo de propiedades llamado: parámetros.properties, donde se debe especificar el nombre que tendrá el contexto del curso para su correcto despliegue en el navegador.

Para desplegar el curso de una manera mas sencilla para el usuario final, se genero un lanzador de cursos, esta aplicación de escritorio tiene la capacidad de iniciar el servidor tomcat, registrar el contexto del curso y cargar el

navegador con la bienvenida del curso, con el fin de que lo pueda utilizar sin ningún problema. Esta aplicación levanta el servidor tomcat embebido en un puerto distinto al 8080 con la finalidad de no interferir con algún otro servidor que lo tenga en uso. Por lo que resulta necesario en esta primera versión de la aplicación especificar el puerto sobre el cual se levantara el servidor, así como también especificar el contexto que contiene el curso a ser registrado en dicho servidor. Una vez generados los ejecutables, importamos los archivos correspondientes a formato JAR para ser ejecutado. La aplicación que lanza los cursos tiene la apariencia mostrada en la imagen 6.7.

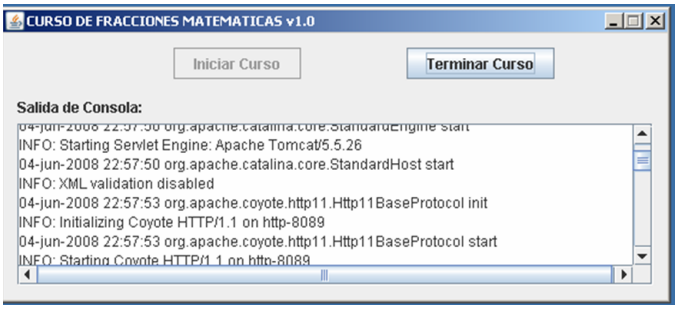


Figura 6.7: Aplicación: Lanzador de cursos

El lanzador incluye todo lo necesario para desplegar el curso en la maquina cliente del estudiante, por lo que una vez adquirido el paquete del lanzador, el estudiante no tendrá que configurar ningún parámetro para poder desplegar el curso, el requerimiento mas importante es tener instalado el JRE 1.4 o superior, un navegador Web Internet Explorer o Mozilla Firefox. Una de las ventajas de la aplicación lanzador, es que puede ser llevada hacia el protocolo JNLP para su distribución con el objetivo de llegar a mas personas y con la ventaja de poder actualizar los cursos manteniéndolos vigentes.

Capítulo 7

Conclusiones y Trabajo a futuro

7.1 Conclusiones

En la actualidad, los sistemas de administración de aprendizaje presentan una gran variedad de soluciones complementarias y opcionales que permiten potencializar la educación en medios electrónicos; sin embargo, aun existen problemas de interoperabilidad al momento de utilizar estas soluciones para compartir sus recursos de infraestructura y contenido. Limitando así, las capacidades que debe cumplir un objeto de aprendizaje para que sea aprovechado de la mejor manera según sus objetivos.

Alternativas como SCORM, han venido a satisfacer algunas de estas necesidades, aplicando un conjunto de estándares y especificaciones para crear objetos de aprendizaje interoperables, accesibles, durables y reutilizables; potencializando los esquemas de trabajo común basados en computación y especialmente sobre Web. Sin embargo, pudimos identificar la recuperación del historial y estado de un objeto de aprendizaje como una necesidad adicional y hasta cierto punto compleja, puesto que involucra todas aquellas posibles formas de interacción con el usuario y requerimientos específicos de cada objeto de aprendizaje.

Definir e implementar los servicios de persistencia necesarios, requiere de un análisis amplio de la problemática que se quiere solucionar, tomando en cuenta criterios tales como el tipo de interacción que se necesita, el sistema de administración de aprendizaje, el sistema manejador de bases de datos, el tipo de objetos de aprendizaje, tales como, simuladores y paginas Web, así como el conocimiento técnico de los diseñadores instruccionales. Conocer bien estos criterios, permitirá evaluar el impacto del uso de diversos esquemas de aprendizaje en OA. Los esquemas de persistencia ofrecen una solución práctica y sencilla que viene a satisfacer la persistencia de estado para objetos de aprendizaje con base en ambientes de aprendizaje sobre Web y el estándar de SCORM.

De acuerdo con el objetivo general para esta tesis, se obtuvo un conjunto de tipos de persistencia de estado para los objetos de aprendizaje con base en el estándar de SCORM. Se logró clasificar estos tipos de persistencia y ubicarlos en un contexto de uso, es decir, generalizar sus características y agruparlos para que los diseñadores instruccionales puedan obtener un panorama más amplio de la aplicación de estos tipos de persistencia.

Con la finalidad de que los objetos de aprendizaje incorporaran más de un tipo de persistencia de estado, se desarrollaron los esquemas de persistencia los cuales integran de forma flexible uno o más tipos de persistencia, dando así, una especialización del comportamiento de persistencia más robusto y apegado a las necesidades de cada objeto de aprendizaje. Se trabajó sobre una implementación sencilla que permitiera a los administradores de los sistemas de aprendizaje o bien a los diseñadores instruccionales registrar los objetos de aprendizaje con algún esquema de persistencia. Bajo este enfoque, se obtuvo una especialización del RTE de SCORM para soportar los esquemas y un

mecanismo de comunicación a base de scripts que puede ser incorporado en los objetos de aprendizaje en tiempo de desarrollo. Esto se realizó sin alterar el estándar para armar los paquetes de contenido de SCORM, conservando así la forma en la que se registran y despliegan los objetos de aprendizaje. Debido a lo anterior, se pudo presentar al usuario final un objeto de aprendizaje que permite conservar el estado de su interacción en sesiones de trabajo.

Los Esquemas de Persistencia, tienen como propósito principal el ofrecer al autor o diseñador instruccional un conjunto de opciones de persistencia que puedan ser integrados en un sólo esquema de acuerdo a sus necesidades instruccionales. Tratando de delegar al LMS las tareas de recuperación y almacenamiento de la información, con el objetivo de que el autor o diseñador instruccional no se vea en la necesidad de conocer detalles técnicos para proveer a sus objetos de aprendizaje las características necesarias para mantener y recuperar su estado en un momento determinado.

De igual forma, se busco de la mejor manera evitar extender el modelo de datos de SCORM, lo que significa que el autor o diseñador instruccional no tiene que aprender o tratar de conocer mas elementos del estándar, evitando que tenga que incluir sentencias complejas dentro de sus recursos o páginas HTML para almacenar la información surgida de la interacción entre el usuario y los elementos del objeto de aprendizaje. Obteniendo de esta forma un método sencillo y simple de implementar la persistencia de estado sin tener que involucrarse con problemas ajenos a los relacionados con los objetos de aprendizaje.

7.2 Contribuciones

Para entender un poco las contribuciones de esta tesis, es necesario tener en cuenta que el punto de inicio fue la adopción del RTE de SCORM, un LMS que proporciona capacidades para desplegar y administrar contenido educativo. Con el fin de ofrecer esquemas con una variedad de funcionalidades para la persistencia se desarrollaron ocho tipos de comportamiento para la persistencia de estado de los objetos de aprendizaje:

1. Persistencia Automática
2. Persistencia Manual
3. Persistencia con conciencia de uso por parte del usuario
4. Persistencia transparente para el usuario
5. Persistencia Total
6. Persistencia Parcial
7. Persistencia con Vencimiento
8. Persistencia sin vencimiento

Los ocho tipos de persistencia se clasificaron según sus características y comportamiento en cuatro categorías las cuales tienen como propósito describir de forma general las cualidades de cada uno de los tipos de persistencia.

1. Control de Proceso de Persistencia
2. Conciencia de Uso

3. Estado del objeto de aprendizaje
4. Temporalidad

Se trabajó sobre el LMS de SCORM para implementar los módulos que permitieran soportar tipos y esquemas de persistencia tanto del lado del servidor como en el cliente. Se incorporaron características al LMS para hacerlo multiplataforma, se agregó una base de datos embebida como repositorio con el fin de eliminar dependencias de plataforma. Los tipos de persistencia se implementaron sobre dos esquemas distintos para probar su funcionalidad. Se agregaron características de persistencia a dos cursos con contenido de aprendizaje distinto, con el fin de validar la metodología que permite incorporar características de persistencia de estado en sus elementos (páginas Web). Cada uno de los cursos tiene características de transportabilidad, es decir, pueden ser distribuidos como un software empaquetado directamente al usuario final. Para el registro de los cursos en el LMS, se desarrolló un módulo que permite asignar algún esquema de persistencia al momento de registrar el paquete del curso.

Se trabajó sobre un artículo sobre la persistencia de estado en objetos de aprendizaje para el LACLO 2008[1], el cual fue aceptado para su presentación en dicho evento, *Anexo 1*.

7.3 Líneas de trabajo a futuro

1. La primera línea para la continuación de este trabajo de investigación es el desarrollo de una metodología que permita obtener los esquemas de persistencia más adecuados a las necesidades de los diseñadores instruccionales, tomando en cuenta tanto factores tecnológicos como pedagógicos.
2. Con el fin de ofrecer los servicios de persistencia en el Ambiente de Instructores Interactivos Educativos de Diversiones Matemáticas el (AIIDM), como en un principio se planteó en esta tesis y aprovechando las características de la implementación del LMS como cliente ligero, se considera como una línea de trabajo a futuro integrar los cursos con base en el LMS de SCORM presentado como portlets en el AIIDM.
3. En cuestión de disponibilidad, se considera la integración de módulos que permitan asegurar contra fallos el buen funcionamiento de los mecanismos de persistencia.
4. Con respecto en al módulo de administración del LMS, se considera desarrollar los mecanismos que permitan importar los objetos registrados a los clientes ligeros de una forma ágil y de fácil uso para los administradores.
5. Se considera apropiado trabajar sobre los mecanismos que permitan crear interfaces con otros tipos de objetos, tales como, simuladores en flash, applets, etc.
6. Implementación de los esquemas de persistencia y el LMS en un contexto educativo.

[1] http://ingsw.ccbas.uaa.mx/laclo2008/

Referencias

[Alia et al., 2004] Alia, M., Chassande-Barrioz, S., Déchamboux, P., Hamon, C., & Lefebvre, A. (2004). A middleware framework for the persistence and querying of java objects.

[Balzer, 2005] Balzer, S. (2005). Contracted persistent object programming.

[Canós et al., 2003] Canós, J. H., Letelier, P., & Penadés, M. C. (2003). Métodologías Ágiles en el desarrollo de software.

[Dodds, 2006a] Dodds, P. (2006a). *SCORM® 2004 3rd Edition Content Aggregation Model (CAM)*, volume Version 1.0. Advanced Distributed Learning (ADL).

[Dodds, 2006b] Dodds, P. (2006b). *SCORM® 2004 3rd Edition Run-Time Environment (RTE)*, volume Version 1.0. Advanced Distributed Learning (ADL).

[Duartes & Rojas, 2008] Duartes, A. O. & Rojas, M. (2008). Las metodologías de desarrollo Ágil como una oportunidad para la ingeniería de software educativo.

[FERMAN, 2006] FERMAN, M. G. (2006). Apoyo a la enseñanza de las matemáticas utilizando un ambiente de aprendizaje basado en instructores interactivos de diversiones matemáticas: Modelo para el maestro. Master's thesis, CENTRO DE INVESTIGACIÓN CIENTÍFICA Y DE EDUCACIÓN SUPERIOR DE ENSENADA.

[Kassahun et al., 2006] Kassahun, A., Beulens, A., & Hartog, R. (2006). Providing author-defined state data storage to learning objects.

[Kienzle & Romanovsky, 2000] Kienzle, J. & Romanovsky, A. (2000). A framework based on design patterns for providing persistence in object-oriented programming languages.

[Letelier & Penadés, 2006] Letelier, P. & Penadés, M. C. (2006). Métodologías ágiles para el desarrollo de software: extreme programming (xp).

[López & Ibarra, 2001] López, G. & Ibarra, J. (2001). Ambiente de aprendizaje basado en instructores interactivos de diversiones matemÁticas.

[Michi Henning, 2006] Michi Henning, M. S. (2006). *Distributed Programming with Ice*. ZeroC, Howard Street, 5th Floor, San Francisco, California, 94105, USA., 3.1.1. edition.

[Miranda et al., 2006] Miranda, P., Prudenza, J., Segurola, A., Calegari, D., & Corral, J. (2006). Arquitectura de acceso a datos en sistemas orientados a objetos: Clasificación y descripción de mecanismos de persistencia.

[Sessink et al., 2003] Sessink, O., Beeftink, R., Tramper, J., & Hartog, R. (2003). Author-defined storage in the next generation learning management systems.

[Weske & Kuropka, 2001] Weske, M. & Kuropka, D. (2001). Flexible persistence framework for object-oriented middleware.

Apéndice A

Artículo presentado en LACLO 2008

Proceedings Latin American Conference on Learning Objects, pp-pp, (Mes, 200x), ISBN: XXXXXXX.

Persistencia de Estado para Objetos de Aprendizaje con base en el estándar SCORM

Gabriel López Morteo
Instituto de Ingeniera
Universidad Autónoma de Baja California
Calle de la Normal S/N Insurgentes Este, Mexicali Baja California, México
galopez@iing.mxl.uabc.mx

José de Jesús Esparza Flores
Instituto de Ingeniera
Universidad Autónoma de Baja California
Calle de la Normal S/N Insurgentes Este, Mexicali Baja California, México
jesusesparza6@yahoo.com

Resumen

Uno de los principales requerimientos con los que deben cumplir los objetos de aprendizaje tiene que ver con la persistencia de estado. Existen varias propuestas que pueden satisfacer este requerimiento, sin embargo es importante tomar en cuenta a uno de los principales usuarios de los objetos de aprendizaje, el diseñador instruccional, por lo que resulta conveniente ofrecer una solución a los problemas de persistencia que no requiera un conocimiento amplio sobre sistemas de almacenamiento, parámetros de configuración o cualquier otro elemento que complique la labor del diseñador al momento de seleccionar un esquema de persistencia para el estado de los objetos de aprendizaje. De esta manera se propone una solución con base en los estándares de SCORM, siendo la principal característica de la solución descrita en este articulo la utilización de tanto el modelo de datos y el API de comunicaciones de SCORM, sin que esto signifique una extensión de ambos componentes, permitiendo así cumplir con el estándar y ampliando las posibilidades de su utilización con cursos ya elaborados.

Palabras Claves: *persistencia de estado, objeto de aprendizaje, SCORM, Learning Management System.*

Abstract

One of the main requirements to be satisfied with the objects of learning has to do with the persistence of state. There are several proposals that can meet this requirement, however it is important to take into account one of the main users of the objects of learning, instructional designer, which makes it desirable to offer a solution to the problems of persistence that does not require knowledge comprehensive storage systems, settings or any other element that complicates the work of designer at the time of selecting a scheme of persistence for the state of learning objects. Thus it is proposed a solution based on the standards of SCORM, being the main feature of the solution described in this article using both the data model and API communications SCORM, but this means an extension of both components, thus enabling meet the standard and expanding the possibilities for use with courses already developed.

Keywords: *retrieve state, learning object, SCORM, learning Management System.*

1. Introducción

Los objetos de aprendizaje, han venido a optimizar el aprovechamiento de contenidos educativos en los salones de clases, proporcionando a los profesores y estudiantes herramientas para apoyar las funciones de docencia y aprendizaje. Sin embargo es necesario que los objetos de aprendizaje cuenten con ciertas características que permitan aprovechar de una mejor manera los contenidos educativos. Tal como se menciona en [1] la generación actual de sistemas administradores de aprendizaje (Learning Management System, LMS) aun tienen ciertas limitantes. Una de estas limitantes es la falta de persistencia de estado, la cual resulta evidente en un estudio realizado por [2] donde se demuestra que en ocasiones una sola sesión no resulta suficiente para completar las actividades dadas por objetos de aprendizaje IIDM descritos en [3]. Por tal motivo es necesario trabajar sobre las actividades del objeto de aprendizaje en sesiones posteriores. La falta de mecanismos de persistencia se traduce en una perdida sustancial de tiempo de una segunda sesión para retomar las actividades y capturar la información que fue previamente obtenida. Esto ocasiona un desinterés tanto por parte del profesor así como por parte del estudiante para trabajar con medios de aprendizaje electrónico.

A partir del análisis realizado sobre los IIDM y la problemática que presentan con el control de persistencia ya no solo se pretende ofrecer una solución propia para este tipo de objeto de aprendizaje, sino una solución que permita implementarse sobre objetos de aprendizaje estandarizados por la especificación SCORM, la cual esta ampliamente documentada y utilizada.

El objetivo principal de este artículo es dar a conocer una propuesta para el control de la persistencia de estado de los objetos de aprendizaje. El cual permita a los diseñadores instruccionales de acuerdo a sus requerimientos, especificar cierto comportamiento en los mecanismos de control de persistencia manteniendo el estándar de SCORM.

2. Mecanismos de Persistencia

La solución de persistencia de estado propuesta en este artículo esta implementada sobre el LMS de SCORM 3ra Edición 2004 [4]. Este estándar ofrece una interfaz de comunicaciones entre el objeto de aprendizaje SCO desplegado en el cliente y el LMS. El SCO esta compuesto por un conjunto de recursos llamados Assets, los cuales pueden ser de tipo (HTML, Applets, archivos JavaScript). Esta característica del LMS de SCORM, sirvió como canal para establecer los mecanismos básicos de comunicación para la persistencia de estado, tratando de no extender el modelo de datos de comunicaciones de SCORM [4] conservando en la medida de lo posible las especificaciones de dicho estándar.

Uno de los propósitos centrales de esta propuesta de persistencia para objetos de aprendizaje con base en SCORM, esta relacionado con la falta de conocimientos avanzados de computación por parte de los diseñadores instruccionales para la implementación de algún esquema de persistencia. Por lo que se buscó diseñar una solución que no requiriera más que la especificación de etiquetas HTML dentro de los recursos de este tipo para implementar algún esquema de persistencia y de esta forma hacer transparente para los diseñadores instruccionales los mecanismos y configuraciones requeridas para la persistencia de estado de los objetos de aprendizaje. Cabe mencionar que esta propuesta no extiende el API de comunicaciones entre el LMS y el SCO, ni el modelo de datos, con la finalidad de apegarse lo más posible al estándar de SCORM. Si bien SCORM, implementa una solución para la persistencia de datos, es muy limitada en cuanto a las opciones que ofrece para hacer persistente los objetos de aprendizaje, basándose solamente en el almacenamiento de datos arbitrarios.

El objetivo principal de los mecanismos de persistencia, es ofrecer las bases para establecer posteriormente especializaciones del comportamiento de la persistencia de estado, a través de distintos tipos de persistencia.

3. Clasificación de los tipos de persistencia

Con el fin de ofrecer una mayor cantidad de opciones en la recuperación y almacenamiento de datos y así mismo agrupar y generalizar las características de comportamiento comunes entre los tipos de persistencia, se realizó una clasificación, la cual permite a los diseñadores instruccionales identificar de manera mas clara y amplia la naturaleza de los tipos de persistencia y los escenarios en los cuales puedan ser de utilidad, creando así combinaciones mas ricas y flexibles. En la figura 1 se muestra tanto la clasificación como los tipos de persistencia que las componen.

Figura 1. Clasificación de los tipos de Persistencia.

3.1. Por el control del proceso de persistencia

Los tipos de persistencia incluidos en esta clasificación se caracterizan por controlar los procesos relacionados con la recuperación y almacenamiento del estado de un recurso HTML del objeto de aprendizaje. Dicho de otra manera, se puede especificar en un objeto de aprendizaje la forma en la cual se van a disparar los eventos que den pie a la persistencia de su estado.

La ocurrencia de mencionados eventos se puede realizar de forma automatizada, donde el usuario no se hace participe en el proceso de persistencia. También los eventos pueden ocurrir por el disparo producido por el usuario. En este caso no solo se involucra el usuario, sino que, se hace responsable del control de la persistencia de estado del objeto cargado en su navegador Web.

Persistencia Automatizada: La persistencia automatizada realiza los procesos de recuperación y almacenamiento de información de acuerdo a la ocurrencia de eventos disparados por el propio Asset. Una vez que el LMS carga el SCO y se presentan los Assets al usuario a través del navegador Web, el Asset dispara un evento que realiza las acciones necesarias para establecer la comunicación con el LMS y recuperar su estado el cual fue previamente almacenado o bien, antes de descargar el Asset con el cual el usuario esta trabajando, se dispara un evento que realiza las acciones necesarias para establecer la comunicación con el LMS el cual envía el estado del Asset a la base de datos para su almacenamiento; la principal característica de este tipo de persistencia es que el usuario no interviene en la ocurrencia de dichos eventos.

Persistencia Manual: Las acciones para recuperar el estado persistente de un Asset también pueden ser ejecutadas tras un evento disparado por parte del usuario, donde este puede o no estar consiente de la ocurrencia de las acciones relacionadas con los procesos de recuperación o almacenamiento del estado de un Asset. El usuario debe disparar el evento que ejecute las acciones necesarias para hacer persistente el estado de un Asset, el cual solo podrá ser ejecutado durante el tiempo en el que el SCO este cargado dentro del LMS.

3.2. Por conciencia de uso

En esta clasificación se agrupan los tipos de persistencia que están relacionados con la conciencia de uso por parte del usuario, donde los procesos de almacenamiento y recuperación por los que pasa la información generada dentro de un Asset en su sesión de trabajo le pueden ser notificados.

Hacer conciente al usuario de la persistencia de estado de los objetos de aprendizaje, contribuye al aumento de las sensaciones de seguridad y confianza en el uso de los medios electrónicos para su aprendizaje.

Por otro lado, la conciencia de uso también esta relacionada con la transparencia, donde los procesos de persistencia de la información que se genera en una sesión de trabajo, no son notificados al usuario y por tal motivo no es conciente de los estados por los que pasa el proceso de recuperación y almacenamiento del estado del Asset con el cual se encuentra trabajando.

Persistencia conciente por el usuario: Con el fin de que el usuario sea conciente de los procesos de recuperación y almacenamiento del estado del Asset con el cual esta trabajando en su navegador Web, el LMS realiza notificaciones para darle a conocer al usuario que se ha recuperado el estado del Asset desplegado, o en su defecto, las causas por las cuales no ha sido posible realizar con éxito dicho proceso; de igual forma se le notifica al usuario que el estado del Asset ha sido almacenado en la base de datos o las causas por las cuales dicho proceso no se realizo con éxito.

Persistencia Transparente para el usuario: Los procesos de recuperación o almacenamiento del estado persistente de un Asset, se realizan de forma transparente para el usuario. Independientemente si la carga del estado del Asset fue automática o manual, al usuario no se le notifica de ningún evento ocurrido tras este proceso, salvo la ocurrencia de algún error relacionado con la persistencia fallida del estado del Asset.

3.3. Por el estado del objeto de aprendizaje

Un objeto de aprendizaje esta conformado por un conjunto de atributos, los cuales obtienen determinado valor como producto de la interacción con el usuario dentro de una sesión de trabajo, produciendo así, el estado de un OA. Son los valores del conjunto de atributos los que son almacenados y recuperados de la base de datos para interactuar con estos en sesiones de trabajo posteriores del usuario.

El estado del objeto de aprendizaje puede estar conformado por los valores para el conjunto total de sus atributos, o bien, dependiendo de las necesidades o requerimientos instruccionales que obedecen a determinado comportamiento, el diseñador instruccional puede especificar cuales atributos conformaran el estado del objeto de aprendizaje.

Persistencia Completa: El Asset no tiene especificados ningún tipo de restricción que determine cuales de sus elementos serán tomados en cuenta para su estado, por lo que considera o incluye todos sus elementos para hacerlos persistentes.

Persistencia Parcial: El Asset tiene configurada una carga o recuperación parcial de su estado, como

resultado de la especificación de sus elementos que serán considerados para la persistencia de su estado, es decir, no se recuperan o se almacenan los datos que no este señalados para determinado propósito.

3.4. Por Temporalidad

La temporalidad se refiere a la característica de los tipos de persistencia que controlan el acceso al estado de un objeto de aprendizaje entre una sesión y otra, es decir, permite especificar las condiciones que deben cumplirse en determinado momento para acceder a los datos almacenados, estas condiciones pueden obedecer a restricciones de diseño de acuerdo con los requerimientos del objeto de aprendizaje.

Persistencia con Vencimiento: El Asset puede incluir parámetros de configuración que permitan especificar mediante condiciones cuando la persistencia de su estado puede ser recuperado o almacenado y cuando no, pueden ser condiciones temporales o relacionadas con propósitos establecidos por el autor de la actividad de aprendizaje.

Persistencia sin Vencimiento: El Asset no tiene especificado ningún parámetro de vencimiento para la persistencia del estado de la actividad, por lo tanto recupera y almacena la información siempre que sea posible.

4. Esquemas de Persistencia

Podemos definir como esquema de persistencia a la implementación de uno o mas tipos de persistencia compatibles entre si y de diferentes clases, los cuales conforman un conjunto de funcionalidades aplicadas a objetos de aprendizaje con características de persistencia en su estado.

Los esquemas al estar integrados por los tipos de persistencia, pueden ofrecer soluciones de persistencia a la medida de las necesidades de los objetos de aprendizaje, o bien, el LMS puede ofrecer un catálogo de esquemas de persistencia donde se combinan los distintos tipos de persistencia explicados anteriormente.

5. Implementación de los esquemas de persistencia

Los Esquemas de Persistencia presentados en este documento, tienen como propósito principal el ofrecer al autor o diseñador instrucción un conjunto de opciones de persistencia que puedan ser integrados en un solo esquema de acuerdo a sus necesidades instruccionales. Tratando de delegar al LMS las tareas de recuperación y almacenamiento de la información, con el objetivo de que el autor o diseñador instruccional no se vea en la necesidad de conocer detalles técnicos para proveer a sus objetos de aprendizaje las características necesarias para mantener y recuperar su estado en un momento determinado.

De igual forma se busco de la mejor manera evitar extender el modelo de datos de SCORM [4], lo que significa que el autor o diseñador instruccional no tiene que aprender o tratar de conocer mas elementos del estándar, esto permite que el autor o diseñador instruccional no tenga que incluir sentencias complejas dentro de sus recursos o paginas HTML para almacenar la información surgida de la interacción entre el usuario y los elementos del objeto de aprendizaje. Obteniendo de esta forma un método sencillo y simple de implementar la persistencia de estado sin tener que lidiar con problemas ajenos a los relacionados con los objetos de aprendizaje.

Actualmente los esquemas de persistencia implementados en el LMS solo recuperan el estado de aquellos objetos de aprendizaje con características de paginas HTML, ya sea desplegada como tal o como un JSP, sin embargo, se puede extender en un momento dado hacia elementos tipo Applets o Flash según las necesidades del objeto de aprendizaje.

Esta solución viene dada dentro de un LMS implementado bajo los estándar es de SCORM, por consecuente cualquier curso con base en este estándar puede ser desplegado con características de persistencia dentro de sus recursos HTML. Uno de los enfoques que se le dio al LMS va dirigido hacia la portabilidad de los cursos registrados, haciendo posible llevar los cursos de una forma desconectada, es decir, a pesar de que LMS esta bajo el esquema de aplicación Web, es posible desplegar un curso sin la necesidad de conectarse a un servidor remoto ya sea un servidor de aplicación o un servidor de base de datos, debido a que el LMS incorpora una base de datos embebida y un servicio que levanta los competentes necesarios para el despliegue del curso, permitiendo llevar el LMS a cualquier plataforma que permita ejecutar una maquina virtual Java, o bien, distribuirlo por diferentes medios, ya sea discos magnéticos, como una aplicación descargada de Internet, etc.

5.1. Requerimientos para Implementar los Esquemas de Persistencia

Con el propósito de dar a conocer paso a paso las acciones a seguir para desplegar un curso con éxito dentro del LMS transportable, actualmente este proceso requiere de mucha labor por parte del administrador, varios de los siguientes procesos pueden ser automatizados en futuras versiones del LMS, para fines de demostración y sobre todo para percibir todo lo que involucra la puesta en marcha del LMS transportable se realizaran los procesos de forma manual.

5.1.2. Armar el paquete del Curso

Si contamos con un curso basado en el estándar de SCORM, o bien, se esta en el proceso de composición del mismo, es necesario especificar en cada uno de los archivos HTML la cabecera especificada en con el fin de que el navegador aplique las reglas correspondientes a archivos de tipo XHTML como se muestra en el fragmento de código 1 de una pagina HTML.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://wwww.w3.org./TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns:"http://www.w3.org/1999/xhtml">

    ...

</html>
```

Código 1. Cabecera HTML

Si el Assets incluye elementos de entrada de datos tales como: inputs, checkbox, radio; es necesario incluir como un recurso el archivo: APIWrapper.js, este archivo además de incluir todas las llamadas al API que el estándar especifica, incorpora los controles necesarios para el manejo de los mecanismos de persistencia. Además de incluir este archivo como un recurso del cual dependen las paginas HTML, es necesario declarar el uso del archivo APIWrapper.js en la sección de cabecera de cada una de las páginas que lo necesiten, tal como se muestra en el código 2.

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>

        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <script src="../util/APIWrapper.js" type="text/javascript"></script>
        <title>División de fracciones</title>
    </head>
    <body onload="loadPage()" onunload="return unloadPage()">
        ...
    </body>
</html>
```

Código 2. Especificación del APIWrapper.js en la cabecera de los archivos.

El uso de la sentencias "loadPage()" inicia la comunicación con el implementación del API de SCORM por lo que es importante ejecutar esta función en el evento Load de la pagina, así como también mandar llamar la función "unLoadPage()" en el evento unload de la pagina con el fin de terminar las comunicación es con la implementación del API al descargar la pagina.

Es importante también mencionar que todos los elementos de entrada antes mencionados tales como inputs, checkbox, radio, etc. tienen que tener su respectiva etiqueta de cierre, ya que son analizados por un parser para recuperar y asignar los datos contenidos. Por motivos de compatibilidad entre los navegadores mas populares Internet Explorer y Mozilla Firefox, es necesario agregar además del nombre a los tag input, el atributo id, ya que puede no funcionar adecuadamente con navegadores Mozilla Firefox, el código 3 muestra un ejemplo de este requerimiento.

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>

        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <script src="../util/APIWrapper.js" type="text/javascript"></script>
        <title>División de fracciones</title>
    </head>
    <body onload="loadPage()" onunload="return unloadPage()">
        <form>
            <input type="text" size="9" name="T2" id="T2"/>
        </form>
    </body>
</html>
```

Código 3. Requerimientos para las etiquetas Input.

5. Asignación del esquema de persistencia al curso

Una vez que el curso ha sido registrado en el LMS por parte del administrador, el siguiente paso es asignar el esquema de persistencia con el cual estará trabajado el curso. Dentro de las opciones administrativas del LMS seleccionamos la opción de Mantenimiento a Cursos, la cual nos mostrara un meno como el que se presenta en la figura 2.

Figura 2. Menú de Administración de Cursos Registrados.

Posteriormente seleccionaremos de un menú desplegable uno de los cursos registrados a los cuales les asignaremos el esquema de persistencia, si el esquema de persistencia involucra la especificación de ciertas configuraciones para cada Asset de tipo HTML, aparecerá después de la asignación un menú desplegable que muestra dichos Assets para aplicarles las configuraciones pertinentes a cada uno, dependiendo del esquema que se haya seleccionado. La figura 3 muestra la asignación del esquema de persistencia.

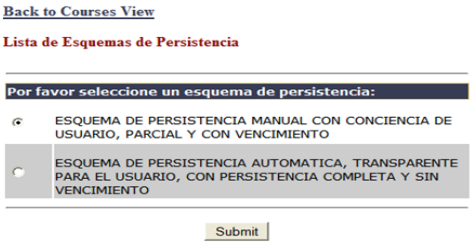


Figura 3. Menú de Asignación de Esquemas de Persistencia

5.2. Despliegue del curso dentro del LMS

Para desplegar el curso de una manera más sencilla para el usuario final, se genero un lanzador de cursos, esta aplicación de escritorio tiene la capacidad de iniciar el servidor tomcat, registrar el contexto del curso y cargar el navegador con la bienvenida del curso, con el fin de que lo pueda utilizar sin ningún problema. Esta aplicación levanta el servidor tomcat embebido en un puerto distinto al 8080 con la finalidad de no interferir con algún otro servidor que lo tenga en uso. Por lo que resulta necesario en esta primera versión de la aplicación especificar el puerto sobre el cual se levantara el servidor, así como también especificar el contexto que contiene el curso a ser registrado en dicho servidor. Una vez generados los ejecutables, importamos los archivos correspondientes a formato JAR para ser ejecutado. La aplicación que lanza los cursos tiene la apariencia mostrada en la figura 4.

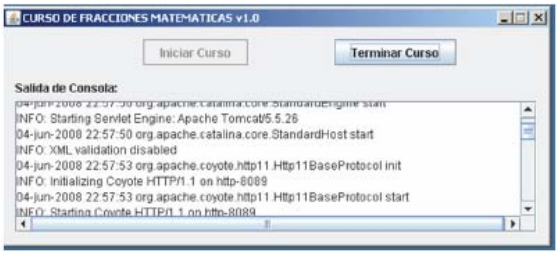


Figura 4. Aplicación: Lanzador de Cursos

El lanzador incluye todo lo necesario para desplegar el curso en la maquina cliente del estudiante, por lo que una vez adquirido el paquete del lanzador, el estudiante no tendrá que configurar ningún parámetro para poder desplegar el curso, el requerimiento mas importante es tener instalado el JRE 1.4 o superior, un navegador Web Internet Explorer o Mozilla Firefox. Una de las ventajas de la aplicación lanzador, es que puede ser llevada hacia el protocolo JNLP para su distribución con el objetivo de llegar a mas personas y con la ventaja de poder actualizar los cursos manteniéndolos vigentes.

6. Conclusiones

En la actualidad los sistemas de administración de aprendizaje aun tienen muchas limitantes, estándares como SCORM han venido a satisfacer algunas de estas necesidades. Sin embargo aun podemos identificar un conjunto de requerimientos adicionales. Entre estas necesidades esta "La Recuperación del historial y estado del objeto de aprendizaje".

La falta de un control en la recuperación del historial y estado para los objetos de aprendizaje puede limitar su aprovechamiento. Es importante contar con una solución que permita incorporar este requerimiento dentro de los estándares actuales para los objetos de aprendizaje, que no implique un impacto en la forma en la que se diseñan los cursos y se despliegan, en especial para los métodos estandarizados como SCORM.

Los esquemas de persistencia ofrecen una solución funcional que viene a satisfacer la persistencia de estado para objetos de aprendizaje basados en Web.

En la actualidad existe una gran variedad de objetos de aprendizaje, por lo que se tiene una gran oportunidad para ampliar los esquemas para hacerlos mas robustos que permitan soportar futuras implementaciones de diferentes objetos de aprendizaje.

7. Referencias

[1] Sessink, O., Beeftink, R., Tramper, J., & Hartog, R. Author-defined storage in the next generation learning management systems. (pp.˜5)

[2] FERMAN, M. G. (2006). Apoyo a la enseñanza de las matemáticas utilizando un ambiente de aprendizaje basado en instructores interactivos de diversiones matemáticas: Centro de Investigación Científica y de Educación Superior de Ensenada.

[3] Instructores Interactivos de Diversiones Matemáticas. http://dx.doi.org/10.1016/j.compedu.2005.04.014

[4] Learning, A. D. (2004b). SCORM 2004 3rd Edition Run-Time Environment Version 1.0.