PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

# INTEGRACIÓN CON OPENID PARA EL SISTEMA AMADEUS

**DANIEL MAURICIO OJEDA ESCOBAR**

INFORME FINAL DEL PROYECTO
PARA OPTAR AL TÍTULO PROFESIONAL DE
INGENIERO DE EJECUCIÓN EN INFORMÁTICA

JULIO 2009

Pontificia Universidad Católica de Valparaíso
Facultad de Ingeniería
Escuela de Ingeniería Informática

# INTEGRACIÓN CON OPENID PARA EL SISTEMA AMADEUS

**DANIEL MAURICIO OJEDA ESCOBAR**

Profesor Guía: **Alexandru Cristian Rusu**

Profesor Co-referente: **Iván Mercado Bermúdez**

Carrera: **Ingeniería de Ejecución en Informática**

Julio 2009

# Dedicatoria

*A mis padres, a mi hermana, a mis dos abuelas y a mis dos abuelos, en especial a Juan Ojeda, quién no ha podido tener en sus manos ninguna memoria escrita por sus descendientes.*

# Índice

# Glosario de términos

**Sistema de Gestión de Aprendizaje:** o LMS por su sigla en ingles *Learning Managment System* es una aplicación software instalado en un servidor, que se emplea para distribuir y controlar las actividades de formación presencial o e-Learning de una institución u organización.

**Framework:** Es una estructura conceptual y tecnológica de soporte definida con base a la cual un proyecto de software puede ser organizado y desarrollado.

**e-Learning:** Traducido como Aprendizaje electrónico, constituye una propuesta de formación que contempla su implementación predominante mediante Internet, haciendo uso de servicios y herramientas que esta tecnología provee.

**GNU Public Licence:** Es una licencia orientada principalmente a proteger la libre distribución, modificación y uso de software.

# Lista de abreviaturas o siglas

**LMS**: *Learning Managment System* o Sistemas de Gestión de Aprendizaje.
**URL**: *Uniform Resource Locator* o Localizador de Recurso Uniforme.
**J2EE**: *Java Enterprise Edition.*
**TIC**: Tecnologías de la información y comunicación
**JSP**: *Java Server Pages* o Páginas de Servidor Java.
**AMADeUs**: Agentes Micromundos y Análisis de Desarrollo en el Uso de Instrumentos.
**SMS**: *Short Messages Services* o servicio de Mensajes Cortos.
**DAO**: *Data Access Object* u Objeto de Acceso a datos.
**XML**: *Extensible Markup Language* o Lenguaje Extensible de Marcas.
**MVC**: Modelo Vista Controlador.
**UML**: *Unified Modeling Language* o Lenguaje Unificado de Modelado.
**XRI**: *Extensible Resource Identifier* o Identificador de Recurso Extensible.
**IDE**: *Integrated Development Enviroment o Ambiente de Desarrollo Integrado.*
**V&V:** Validación y Verificación**.**

# Lista de figuras

# Lista de tablas

# Resumen

Los Sistemas de Gestión de Aprendizaje o LMS (*Learning Managment System*) son la principal herramienta usada como apoyo a la educación, en el contexto del e-Learning. AMADeUs, un LMS basado en el concepto de aprendizaje flexible ha sido desarrollado como un proyecto de software libre. La presente memoria documenta el trabajo realizado para integrar AMADeUs con el sistema de autentificación digital OpenID. La ventaja de OpenID es que permite a sus usuarios el tener una identidad digital y poder hacer uso de ella a potencialmente en toda la Web.

Las tecnologías utilizadas en el desarrollo de AMADeUs le permiten ser un software independiente de la plataforma y fácilmente extensible. En el presente trabajo se han complementado las tecnologías de AMADeUs, los frameworks Struts e Hibernate con la tecnología de OpenID, puntualmente la librería OpenIDforjava, empleada aquí para llevar a cabo la integración.

**PALABRAS CLAVES:** OpenID, AMADeUs, e-Learning, Sistema de gestión de aprendizaje.

# Abstract

Learning Managment Systems or LMS are the ultimate tool within the context of the e-Learning. It is in this context that AMADeUs has been developed; it is a LMS based in the concept of blended learning and is an open source software project. In this document is presented the work of the integration of AMADeUs with the digital authentication system OpenID, The advantage of OpenID is that it allows its users to have a digital identity and be able to utilize it potentially throughout the entire Web.

AMADeUs has been developed with technologies that make it a platform independent and easily extensible software. In the present work has been complemented the AMADeUs technologies, Struts and Hibernate with the OpenID technology, being this the OpenIDforjava library for J2EE applications needing OpenID support.

**KEY WORDS:** Openid, Amadeus, e-Learning, Learning managment system.

# 1. Introducción

## 1.1. Origen del tema

El presente proyecto se halla enmarcado dentro del Proyecto AMADeUs, desarrollado por el grupo Ciencias Cognitivas y Tecnología Educacional y el Centro de Informática de la Universidad Federal de Pernambuco, Brasil. AMADeUs, esta pensado como un sistema de gestión de aprendizaje que integra la enseñanza a distancia con el uso de diferentes recursos multimedia. Como parte de los módulos a desarrollar en el proyecto, se ha sugerido el de la integración con el sistema de autentificación digital OpenID, el cual permite a sus usuarios ocupar su identidad para registrarse e ingresar en todos los sistemas que tengan soporte con esta tecnología.

A través de la colaboración entre la Pontificia Universidad Católica de Valparaíso y la Universidad Federal de Pernambuco ha surgido la oportunidad de participar en AMADeUs, a lo que se le suma el interés por parte del alumno de participar en un proyecto de código abierto de esta índole ha dado origen al presente trabajo.

## 1.2. Descripción del problema

El sistema de autentificación digital OpenID es una herramienta que permite a quienes tengan una identidad de OpenID usar esta identidad, para ingresar a todos los sistemas Web que soporten este tipo de identidades. Al integrar cualquier sistema con OpenID se obtienen el beneficio, principalmente para el usuario de reducción de la frustración asociada con mantener múltiples nombres de usuario y contraseña, ya que suelen ser fáciles de olvidar y el proceso de recuperación de contraseñas puede llegar a ser tedioso además de lograr un mayor control de los perfiles de usuarios en la Web.

La Fundación OpenID es una organización sin fines de lucro dedicada a promover y proteger el uso de las tecnologías OpenID, entre otras cosas, mantiene un repositorio de librerías para distintos lenguajes y establece un protocolo para la comunicación de las diversas aplicaciones con OpenID, a fin de proporcionar a usuarios y desarrolladores un marco común para el trabajo en dicha tecnología. El protocolo de OpenID establece la forma de llevar a cabo la comunicación entre los usuarios y los proveedores de OpenID. Delineando el paso de información desde los usuarios pasando por las aplicaciones, los navegadores y los proveedores de OpenID.

AMADeUs busca lograr una mayor difusión, no sólo en Brasil, su país de origen, sino también en Latinoamérica, por lo que este sistema de autentificación podrá enriquecer el sistema y abrirle nuevas puertas entre sus potenciales usuarios.

## 1.3. Objetivos

**Objetivo general:**

Integrar al sistema AMADeUs con el sistema de identificación digital OpenID.

**Objetivos específicos:**

- Comprender los conceptos de e-Learning y sistema de gestión de aprendizaje.
- Comprender la arquitectura de AMADeUs así como las tecnologías involucradas en su desarrollo.
- Comprender el sistema de autentificación OpenID y los recursos existentes para la integración.
- Desarrollar el módulo encargado de la integración.
- Validar el resultado del trabajo.

## 1.4. Alcances del proyecto

El presente proyecto contempla la realización de un módulo para el sistema de gestión de aprendizaje Amadeus, para ello, realiza un estudio previo de las posibles alternativas a desarrollar, para luego proceder al análisis, diseño, implementación y prueba del mismo.

Como se verá más adelante se ha optado por el desarrollo del módulo de integración del sistema con OpenID, lo que contemplará el diseño de las interfaces nuevas correspondientes, del modelado propio del módulo, su codificación, implantación en el sistema y correspondientes pruebas.

## 1.5. Metodología de trabajo

La amplia variedad presente en la naturaleza de los proyectos de desarrollo de software que actualmente tienen lugar, ha llevado a la formulación y empleo de diversas metodologías dentro de la industria. Y la elección de la metodología que resulte más apropiada y eficiente para un determinado proyecto se ha transformado en una de las decisiones más importantes que el analista debe tomar en el marco de un proyecto de software.

Debe tenerse presente que en el contexto de la participación en el proyecto Amadeus, además de juzgar la metodología que mejor se adecué al contexto del proyecto en la asignatura, es necesario señalar que el proyecto Amadeus ya cuenta con el uso de la metodología de Análisis y Diseño Orientado a Objetos, por lo que es necesaria su adopción como parte de la integración en el mismo.

Actualmente la tecnología orientada a objetos no se aplica exclusivamente a lenguajes de programación, sino que como se ha señalado en el párrafo anterior, involucra también el análisis y el diseño de un software.

La creciente tendencia a crear software de mayor envergadura llevó a los desarrolladores a crear una nueva metodología de desarrollo que hiciera posible

sistemas informáticos de niveles empresariales y con reglas de negocio más complejas, siendo esta la orientación a objetos.

Conjuntamente con esto podemos señalar las características y ventajas propias de la orientación a objetos y que la hacen la adecuada para el presente proyecto, podemos decir que la orientación a objetos:

- Fomenta la reutilización y extensión del código fuente.
- Permite crear sistemas más complejos.
- Representa de manera más exacta elementos del mundo real.
- Facilita la creación de interfaces gráficas.
- Propicia la creación de prototipos.

Agiliza el desarrollo y mantención del software.

## 1.6. Paradigma de trabajo

Para resolver los problemas reales de una industria, un ingeniero del software (o un equipo de ingenieros) debe incorporar una estrategia de desarrollo que acompañe al proceso métodos y capas de herramientas y fases genéricas. Esta estrategia a menudo se llama modelo de proceso o paradigma de ingeniería del software. Se selecciona un modelo de proceso para la ingeniería del software según la naturaleza del proyecto y de la aplicación, los métodos y las herramientas a utilizarse, y los controles y entregas que se requieren.

Un modelo de proceso del software es una representación abstracta de un proceso del software. Cada modelo de proceso representa un proceso desde una perspectiva particular, y así proporciona sólo información parcial sobre ese proceso.

Estos modelos generales no son descripciones definitivas de los procesos del software. Más bien, son abstracciones de los procesos que se pueden utilizar para explicar diferentes enfoques para el desarrollo del software. Puede pensarse que ellos son como marcos de trabajo del proceso que pueden ser extendidos y adaptados para crear procesos más específicos de ingeniería del software.

- **Modelo en cascada:** Considera las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución, y los representa como fases separadas del proceso, tales como la especificación de requerimientos, el diseño del software, la implementación las pruebas, etc.

- **Desarrollo evolutivo:** Este enfoque entrelaza las actividades de especificación, desarrollo y validación. Un sistema inicial se desarrolla rápidamente a partir de especificaciones abstractas. Este se refina basándose en las peticiones del cliente para producir un sistema que satisfaga sus necesidades.

- **Ingeniería de software basada en componentes:** Este enfoque se basa en la existencia de un número significativo de componentes reutilizables. El proceso de desarrollo del sistema se enfoca en integrar estos componentes en el sistema más que en desarrollarlos desde cero.

Estos tres modelos de procesos genéricos se utilizan ampliamente en la práctica actual de la ingeniería del software. No se excluyen mutuamente y a menudo se utilizan

juntos, especialmente para el desarrollo de sistemas grandes (de hecho el Proceso Unificado de Racional combina elementos de todos estos modelos). Los subsistemas dentro de un sistema más grande pueden ser desarrollados usando enfoques diferentes. Por tanto, aunque es conveniente estudiar estos modelos separadamente, debe entenderse que, en la práctica, a menudo se combinan.

Teniendo en cuenta el hecho de que los requerimientos de este proyecto no estuvieron completamente claros desde el principio, sino que la comprensión de los mismos ha evolucionado a lo largo del trabajo es que se ha escogido el **Modelo Evolutivo**, también conocido como **Paradigma de Desarrollo Basado en Prototipos**, ya que entre las ventajas que presenta dicho paradigma y que ya se han señalado, es posible señalar el que permite ir comprendiendo de mejor manera los requerimientos a medida que se avanza en el proyecto y que el cliente puede visualizar, en etapas tempranas, las características principales que tendrá el software. A medida que se avance en el trabajo, se irán entregando (ver Planificación del Proyecto) prototipos desechables del módulo a construir con el objetivo de que sirva de retroalimentación para lograr ir refinando los requerimientos. Sin embargo, también se han tomado elementos de la **Ingeniería del Software Basado en Componentes**, ya que al trabajar sobre un sistema ya desarrollado y en crecimiento como es AMADeUs, existen varios componentes del sistema con los que el módulo de integración con OpenID deberá interactuar y usar en otros casos.

## 1.7. Herramientas y tecnologías utilizadas

Al estar AMADeUs ya desarrollado como un software, se han definido con anterioridad tanto la metodología y el paradigma de desarrollo como las tecnologías y herramientas que se deben utilizar, también ha sido definida la arquitectura del sistema, todo esto con el fin de obtener un sistema de código abierto, independiente de la plataforma, de alta cohesión y bajo acoplamiento.

A través de la arquitectura y el funcionamiento de AMADeUs está el modelo MVC, este es un patrón de arquitectura de software que separa los componentes del sistema en datos de la aplicación, interfaz de usuario y lógica de control, en tres elementos distintos, llamados: Modelo, Vista y Controlador.

A continuación, una lista de las tecnologías y herramientas involucradas en el proyecto:

- **J2EE:** Conocida también como *Java Enterprise Edition* es un entorno de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java con arquitectura de N niveles y que se complementen con componentes software modulares ejecutándose en un servidor de aplicaciones.

- **Apache Struts:** Más conocida simplemente como Struts es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC bajo la plataforma J2EE, de hecho fue el primer framework para MVC que apareció. Ha sido desarrollada como parte del proyecto Jakarta de la *Apache Software Fundation* aunque actualmente es un proyecto independiente. Struts permite reducir el tiempo de desarrollo al ordenar tareas comunes a muchos proyectos de desarrollo Web, además al ser un software libre y su compatibilidad con las plataformas J2EE la hacen muy disponible.

- **Hibernate:** Es una herramienta de mapeo objeto-relacional para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones. En otras palabras Hibernate busca solucionar el problema de la diferencia entre los modelos de datos coexistentes en una aplicación, el usado en la memoria de la computadora y el usado en la base de datos.

- **IDE Eclipse:** Eclipse es un ambiente de desarrollo integrado o IDE (por su denominación en inglés *Integrated Development Enviroment*) para desarrollo de aplicaciones en java, aunque también puede ampliarse para otros lenguajes de programación. Eclipse cuenta con un sistema extensible de *plug-ins* o complementos para diversas funcionalidades y es un producto de software libre. Una de las características más apreciadas de Eclipse es la posibilidad de manejar perspectivas, es decir, de configurar rápidamente la interfaz de trabajo respecto de la labor puntual que se esta llevando a cabo, ya sea escribir código, depurar el código escrito o administrar las bases de datos.

- **OpenIDforJava:** Es una librería de código abierto desarrollada para habilitar la integración con OpenID en aplicaciones Web en java.

Si bien estas tecnologías y herramientas estaban ya definidas y se debieron seguir al momento de realizar la integración con OpenID, existen también otras que podrían haberse utilizado y servido también a los objetivos del equipo de desarrollo, de las cuales, algunas se mencionan a continuación:

- **PHP:** Acrónimo de *PHP Hypertext Pre-processor*, es un lenguaje interpretado de propósito general ampliamente usado, diseñado especialmente para desarrollo Web, puede ser desplegado en la mayoría de los servidores Web y en casi todos los sistemas operativos. Varios sistemas de gestión de aprendizaje, como Moodle y Claroline por ejemplo están desarrollados e PHP.

- **Spring Framework:** Al igual que Struts es una framework que permite implementar el modelo MVC en aplicaciones Web J2EE, aunque es más reciente que éste, es considerado por muchos, como más flexible que Struts y ofrece mejor interacción con tecnologías distintas a JSP.

- **IDE NetBeans:** Partió como un proyecto universitario en la República Checa y actualmente es uno de los IDE para Java no privativos más utilizados en el mundo, una de las principales desventajas que tiene ante Eclipse es la carencia del concepto de perspectivas.

# 2. Plan de trabajo

**Tabla 2.1 Plan de trabajo.**

| Fecha | Actividad | Estado |
|---|---|---|
| 10 de Septiembre 2008 | Estudio del Estado del Arte. | Completado |
| 3 de Octubre 2008 | Estudio de los posibles módulos a desarrollar. | Completado |
| 6 de Noviembre 2008 | Determinación de los módulos y comienzo del análisis y diseño respectivos. | Completado |
| 1 de Diciembre 2008 | Documentación del análisis y diseño. | Completado |
| 17 de Abril 2009 | Entrega primer prototipo. | Completado |
| 5 de Junio 2009 | Entrega segundo prototipo. | Completado |
| 1 de Julio 2009 | Pruebas y entrega del software (módulo) funcionando. | Completado |
| 1 de Junio 2010 | Adición de la integración con Hibernate. | Completado |
| 15 de Agosto 2010 | Adición del registro con parámetros de OpenID | Completado |

La primera actividad que figura en la tabla el Estudio del Estado del Arte consistió en la investigación de los conceptos de e-Learning y AMADeUs.

En un principio no se tuvo claro la contribución específica al proyecto AMADeUs que se emprendería, se estudió la lista de todos los módulos planteados y el estado de avance en el que se encontraban, se determinó realizar la Integración con OpenID.

Se comenzó luego a estudiar lo relativo a la Integración con OpenID siendo para esto, la principal fuente la página oficial de la OpenID Fundation [OpenID, 09], así como otros sitios que ya contaban con el soporte.

En un principio se contempló incluir en el proyecto un estudio de usabilidad, siendo una parte primordial del enfoque del trabajo, pero posteriormente se descartó debido a las dimensiones del software a construir.

En el capítulo 8, Diseño e Implementación se detalla el trabajo con los prototipos, los cuales se fueron entregando en el transcurso de las instancias de evaluación formales en el contexto de las asignaturas de Proyecto 1 y Proyecto 2.

Posterior al término de las asignaturas se envió la primera versión a los administradores del proyecto AMADeUs, los cuales se manifestaron conformes con lo realizado. El último trabajo hasta el momento consistió en la utilización del framework Hibernate.

# 3. Estudio de Factibilidad

Para todos los sistemas nuevos a desarrollar se debe llevar a cabo previo a su realización un estudio de factibilidad del proyecto, siendo el resultado de dicho estudio un informe que indica si es conveniente llevar a cabo el proceso de desarrollo del sistema.

**Factibilidad operativa:**

El sistema de identificación digital OpenID es una plataforma abierta y descentralizada, compatible con sistemas tales como AOL, Blogger, LiveJournal, WordPress, y una serie de proveedores del servicio (claimID, myID, entre otros). OpenID elimina la necesidad de múltiples nombres de usuario a través de diferentes sitios Web, simplificando así la experiencia del usuario. Tomando todo esto en cuenta es posible afirmar que el módulo (la integración con OpenID) es **operativamente factible** al ayudar a la difusión del sistema Amadeus.

**Factibilidad técnica:**

Las herramientas necesarias para poder realizar el presente proyecto figuran en la siguiente lista:

- Eclipse.
- El Framework Struts.
- El Framework Hibernate
- El motor de bases de datos PostGreSQL 8.3.
- Microsoft Visio (para el modelado con UML).
- El navegador Mozilla Firefox o bien Internet Explorer.
- El sistema operativo Windows XP.

Teniendo en cuenta de que se dispone de estas herramientas, es posible afirmar que el sistema es **técnicamente factible**.

**Factibilidad legal:**

Con respecto a la factibilidad legal se puede decir que no existen trabas legales que impidan el buen desempeño y funcionamiento del software, puesto que no se incurren en infracciones a las leyes vigentes, específicamente:

Ley Nº 19.223 la cual Tipifica figuras penales relativas a la informática, específicamente los artículos 1, 2, 3 y 4 pues el software no daña, altera, etc., ningún sistema de información existente.

Ley Nº 17.336 que dice relación con la propiedad intelectual, específicamente el artículo 41 el cual dice relación con copias o adaptaciones, pues en este proyecto no se

realizan copias de código fuente, interfaces, etc., de algún otro software que pudiera tratar la misma materia.

Teniendo en cuenta que AMADeUs es un software desarrollado en Brasil, esta sujeto también a leyes brasileñas, sin embargo es un proyecto de software libre incluido en el Portal de Software Público Brasileño.

El Portal de Software Público Brasileño inauguró una nueva etapa en el desarrollo de la política del software libre en Brasil. Esta política presenta un nuevo modelo de licencia, de gestión de reglas de disponibilidad de las soluciones desarrolladas para la administración pública y la sociedad. Durante muchos años, la comunidad brasileña de Software Libre pidió que el Gobierno Federal compartieses con todos su inteligencia en el área de desarrollo y dejase de ser un mero usuario de las herramientas automatizadas. Con el Portal de Software Público Brasileño, el sector público se convirtió en protagonista del desarrollo de soluciones informáticas.

La iniciativa fue lanzada en abril de 2007 durante el Foro internacional de Software Libre. En ese ambiente fueron compartidas soluciones que pueden ser útiles a los diferentes organismos públicos y también a la sociedad. Su objetivo es el de reducir costos, mejorar las aplicaciones disponibles y mejorar la atención a la población, además de crear espacios de colaboración en la sociedad.

Las soluciones desarrolladas por Organismos Públicos, ya sea ejecutivo, legislativo o judicial, además de empresas y universidades en este portal, están disponibles gratuitamente a la sociedad. Estados, empresas, organismos públicos, cetros de investigación y cualquier persona que estén interesados en poder obtener el código fuente de estas soluciones mediante el registro en el portal. [Portal, 10]

Considerando lo anteriormente mencionado, es posible decir que el proyecto es **legalmente factible**.

## **Factibilidad económica:**

El objetivo de realizar el estudio de factibilidad económica es determinar si se cuenta con los medios o recursos económicos necesarios para llevar a cabo el proyecto.

En el presente proyecto, los gastos propios del desarrollo del mismo será asumido por parte de el alumno, ya que no se cuenta con una empresa que corra con ellos y el proyecto Amadeus se trata de un proyecto de software libre. Los costos están mencionados en la siguiente lista:

**Tabla 3.1 Costos del proyecto.**

| Concepto | Costo |
|---|---|
| 2 resmas de papel tamaño carta. | $ 4.000 |
| 1 cartucho de impresión tinta negra. | $ 18.500 |
| 1 cartucho de impresión tinta color. | $ 6.250 |
| 5 anillados | $ 3.000 |
| Movilización para reuniones. | $ 30.000 |
| Internet. | $ 115.500 |
| Personal | $ 400.000 |

Ya que se cuenta con los recursos monetarios para la realización del proyecto, es posible decir que el proyecto es **económicamente factible**.

# 4. e-Learning

Desde los albores de la civilización, el hombre ha buscado y valorado conocimiento de distinta índole, poniendo énfasis en su preservación y distribución. La sociedad contemporánea no es la excepción, y el aumento del conocimiento en las ciencias, artes y otras ramas ha traído consigo un incremento en la demanda de educación a distintos niveles. Esta mayor demanda tiene como contra parte una mayor oferta de **educación**, la cual se halla inmersa en un mundo complejo, debiendo por lo mismo hacer frente a obstáculos tales como la disponibilidad de tiempo, ubicación geográfica y sector socioeconómico de las personas que finalmente requieren la enseñanza.

Con el objetivo de satisfacer estas necesidades y con el avance en el desarrollo de las tecnologías de la información y comunicación, ha surgido el concepto de e-Learning, cuya traducción literal sería aprendizaje electrónico, vale decir, la utilización de medios y recursos electrónicos como fuente o medio de aprendizaje en alguna materia, sin embargo, una definición más formal sería la dada por la *American Society of Training and Developing* (Sociedad Americana de Capacitación y Desarrollo) y que dice:

***ñe-Learning es un término que cubre un amplio grupo de aplicaciones y procesos, tales como aprendizaje basado en ordenadores, aulas virtuales y colaboración digital. Incluye entrega de contenidos vía Internet, intranet/extranet, audio y video grabaciones, transmisiones satelitales, tv interactiva, CD-ROM y más.ò***.

Al considerar esta definición que deja el espectro del e-Learning abierto a cualquier proceso en el que se vean involucrados educación y tecnología, como una definición terminante y definitiva, dejaríamos de considerar, por otro lado, otras visiones que acotan el universo del e-Learning a sólo el ámbito de la Internet y las tecnologías asociadas, definiéndola de la siguiente manera:

ñEl uso de tecnologías Internet para la entrega de un amplio rango de soluciones que mejoran el conocimiento y el rendimiento.ò.

Dichas herramientas, con posibilidades tanto síncronas como asíncronas, cuando son incorporadas a la educación, proveen de nuevas formas de interacción y retroalimentación, lo que se traduce en modificaciones en torno al tiempo y el espacio de la participación, tanto de formadores como de estudiantes dentro del proceso de enseñanza y aprendizaje.

Existen tres conceptos o criterios fundamentales en los cuales el e-Learning está basado, los cuales son:

1.- El e-Learning trabaja en **red**, lo que lo hace capaz de ser instantáneamente actualizado, almacenado, recuperado, distribuido y permite compartir instrucción o información.

2.- El e-Learning es entregado al usuario final a través del uso de **ordenadores** utilizando tecnología estándar de internet.

3.- El e-Learning se enfoca en la **visión** más amplia del aprendizaje que van más allá de los paradigmas tradicionales de capacitación. [García, 09]

## 4.1. Evolución del e-Learning

Ya en la década de los ochenta, con la llegada del computador personal, las tecnologías informáticas comenzaron a ser usadas en el ámbito de la educación. Sin embargo, dada la poca versatilidad con las que estas tecnologías contaban en esos tiempos, sólo permitían la realización de unas pocas tareas. Este tipo de software pasó así a ser más que nada en un **elemento de apoyo** a la educación, de ahí surgió lo que llegó a denominarse como educación asistida por computador.

El uso de estas herramientas con fines didácticos comenzó a adquirir un rol más protagónico a comienzos de los noventa con la aparición de los multimedia y de la Internet. En esta época fue posible elaborar materiales con contenidos enlazados, debido al aumento de la capacidad de los computadores y las ventajas del uso de CD-rom interactivos y la capacidad de integrar audiovisuales. Estábamos ya en el período conocido como **multimedia educativa**, el cual se prolongó hasta mediados de los noventa.

Con la consolidación de la Internet a mediados de los noventa, especialmente en el último tercio de esta década, comienza el período de la **Teleinformación**. En él resalta el apoyo de páginas Web educativas, proveyendo interacción y retroalimentación entre profesores y alumnos y entre alumnos y alumnos a través de correo electrónico, foros de discusión y salas de Chat. Todo esto provocó que nuevas opciones se hicieran disponibles, no sólo un aumento bastante significativo en la cobertura, sino también una mayor autonomía en la cobertura, Así como una mayor autonomía del estudiante por medio del estudio independiente y la posibilidad de interacción síncrona y asíncrona.

El desarrollo de tecnologías cada vez más sofisticadas ha permitido entre otras cosas:

- El aumento de las posibilidades de interacción y retroalimentación a través de las distintas herramientas proporcionadas a los usuarios.
- Un acceso más fácil a una amplia gama de contenidos, además,
- El surgimiento de estándares de calidad educativa y técnica, los cuales han facilitado el intercambio, tanto de contenidos como de información entre plataformas pertenecientes a distintas instituciones.

## 4.2. Modelos del e-Learning

En la actualidad, todo programa de e-Learning, en función de las necesidades y características del proyecto formativo dentro de cada institución, debería estar conformado por la definición de tres modelos, a saber, el Modelo Organizacional, el Modelo Pedagógico y el Modelo Tecnológico, a continuación una breve descripción de cada uno de ellos:

- **Modelo Organizacional:** La implantación de procesos de e-Learning conlleva cambios en la organización educativa, determinando cuáles son aquellas necesidades que se desean satisfacer con la aplicación de proyectos e-Learning, así como la creación de una política de comunicación interna para dar a conocer

los cambios que se vayan adoptando. Luego, a partir del diagnóstico, tomar las decisiones referentes al diseño de contenidos y materiales de la acción formativa, a continuación se procedería a la definición de los tipos de estructura y los medios de interacción, para terminar con una etapa de seguimiento y control del proyecto.

- **Modelo Pedagógico:** Se debe tener claro cuales son las expectativas del aprendizaje y proponerse estrategias individuales y colectivas que propicien la búsqueda de información, la interacción, la retroalimentación y el aprendizaje colaborativo a través de las tecnologías. Luego se deben definir los conocimientos, habilidades y competencias que alcanzarán los estudiantes, esto último debe estar relacionado con los contenidos que satisfagan los objetivos ya definidos, así como las actividades que darán cumplimiento a esos objetivos. Finalmente se debe considerar la manera en que se evaluará el aprendizaje y la definición de las dinámicas de atención, interacción y retroalimentación que se le entregará al estudiante sobre motivación al estudio, contenido y metodología de trabajo.
- **Modelo Tecnológico:** Su formulación consiste en la selección de las tecnologías de la información y comunicaciones que, en función del perfil de los formadores, profesores y estudiantes, las características de los contenidos y los objetivos que se pretende alcanzar, mejor se ajusten a las necesidades institucionales. Algunos conceptos importantes dentro de este enfoque son, por ejemplo: la definición del proceso administrativo automatizado adecuado para generar un sistema de control y seguimiento de los usuarios, la relevancia de crear un sistema que garantice la seguridad, privacidad e integridad de la información y el diseño de estándares de acceso de acuerdo con las características de los contenidos y actividades e-Learning. Facilitar el acceso a través de una navegación intuitiva y sencilla para los usuarios, y reducir, dentro de lo posible, el tiempo descarga de la información. También se toma en cuenta la disposición de los elementos de ayuda a la navegación y una organización lógica de los contenidos [Landeta, 09].

## 4.3. Situación actual de los proyectos de e-Learning

A partir de mediados de la década pasada, en diversos países, ha aumentado, junto con los avances en tecnologías de la información y telecomunicaciones, el interés por plantear nuevos proyectos de e-Learning, tanto en el ámbito de la entidades sin fines de lucro, administración pública, en empresas y por supuesto, en instituciones educativas.

En entidades sin fines de lucro están tomando medidas para fomentar en sus colaboradores, la familiarización con las nuevas tecnologías y las nuevas metodologías de enseñanza y aprendizaje. Todo esto con el fin de obtener los beneficios que reporta el uso de herramientas e-Learning en proyectos de cooperación con otros organismos y en la proyección de sus acciones y servicios a la sociedad, ya que amplía el abanico de oportunidades de acercamiento y participación de la ciudadanía en proyectos sociales.

Así mismo, en los servicios de administración pública, sus funcionarios también han comenzado a familiarizarse con estas tecnologías, esta tendencia ha contribuido a la incorporación de prácticas de e-Learning teniendo en mente la meta de proporcionar una formación constantemente flexible para sus funcionarios.

Son tres los efectos que el e-Learning está teniendo a nivel de empresas. Primeramente, dado el mayor valor que se le ha venido dando a los recursos humanos dentro de una empresa en los últimos tiempos, es precisamente en esta área donde se ha ido concentrando la incorporación de plataformas formativas, facilitando así la gestión del conocimiento organizacional.

También, teniendo en cuenta el impacto de factores tales como la globalización, ha aumentado la complejidad y especificación de los diversos perfiles laborales, haciendo de las herramientas de e-Learning, una muy atractiva alternativa de educación y formación constante entre los empresarios y otras personas o profesionales relacionados con ellos. Cabe destacar en este ámbito, que se han creado nuevas líneas de negocios en empresas privadas gracias al e-Learning y su efecto en las relaciones entre profesionales y empresarios de distintas nacionalidades.

El e-Learning en las instituciones de educación superior también ha tenido fuertes impactos en el proceso de enseñanza, contribuyendo a la generación de nuevos modelos de educación relacionados con el proceso de enseñanza y aprendizaje en herramientas tecnológicas.

También dentro de la gestión académico administrativa, donde han surgido nuevos mecanismos o maneras de gestionar la docencia, al contar con el apoyo de las tecnologías de la información y telecomunicaciones, posibilitando la adaptación a las diferentes instituciones, tomando en cuenta sus necesidades y características propias. Por otro lado, en la investigación, han aparecido nuevas estrategias de apoyo, desarrollo y difusión de las distintas investigaciones.

Los cambios en la educación superior, prevén impactos también en las universidades, potenciando el papel de la universidad en las exigencias de la economía basada en una permanente innovación, por medio del fortalecimiento de estructuras y estrategias que hagan posible el acercamiento de conocimientos y resultados de investigaciones a la comunidad en general. La incorporación de herramientas e-Learning al ámbito universitario ha ayudado en la incubación de nuevos negocios y en el cultivo de enlaces con empresas. Potenciando en el camino de la transición de la educación con herramientas didácticas tradicionales al desarrollo de actividades e-Learning la creación de verdaderos campus o comunidades universitarias virtuales o, en línea, contando con una amplia oferta educativa en diversos niveles.

## 4.4. Sistemas de gestión de aprendizaje

Un sistema de Gestión de Aprendizaje o **LMS**, como son comúnmente llamados por su denominación en inglés *Learning Managment System*, esta entre las herramientas más ampliamente utilizadas para los ambientes de e-Learning y son conocidos también como plataformas de aprendizaje. Podemos decir que un LMS es un sistema software basado en un servidor que provee módulos para llevar a cabo los procesos administrativos y de seguimiento que son necesarios para un sistema de enseñanza, haciendo más simple el control de este tipo de tareas. Por lo general estos módulos permiten configurar cursos, matricular alumnos, registrar profesores, asignar cursos a un alumno, llevar informes de progreso y calificaciones. También facilitan el aprendizaje distribuido y colaborativo a partir de actividades y contenidos preelaborados, de forma síncrona o asíncrona, utilizando los servicios de comunicación de Internet como el correo, los foros, las videoconferencias o salas de Chat. [García, 09]

El alumno interactúa con la plataforma a través de una interfaz Web que le permite seguir las lecciones del curso, realizar las actividades programadas, comunicarse con el profesor y con otros alumnos, así como dar seguimiento a su propio progreso con datos estadísticos y calificaciones. La complejidad y las capacidades de las plataformas varían de un sistema a otro, pero en general todas cuentan con funciones básicas como las que se han mencionado.

Los Sistemas de Administración de Contenidos de Aprendizaje o LCMS tienen su origen en los CMS (*Content Management System*) cuyo objetivo es simplificar la creación y la administración de los contenidos en línea, y han sido utilizados principalmente en publicaciones periódicas (artículos, informes, fotografías). En la mayoría de los casos lo que hacen los CMS es separar los contenidos de su presentación y también facilitar un mecanismo de trabajo para la gestión de una publicación Web. Los LCMS siguen el concepto básico de los CMS, que es la administración de contenidos, pero enfocados al ámbito educativo, administrando y concentrando únicamente recursos educativos y no todo tipo de información.

En esencia, se define entonces un LCMS como un sistema basado en Web que es utilizado para crear, aprobar, publicar, administrar y almacenar recursos educativos y cursos en línea. Los principales usuarios son los diseñadores instruccionales que utilizan los contenidos para estructurar los cursos, los profesores que utilizan los contenidos para complementar su material de clase e incluso los alumnos en algún momento pueden acceder a la herramienta para desarrollar sus tareas o completar sus conocimientos.

Los contenidos usualmente se almacenan como objetos descritos e identificables de forma única. En un LCMS se tienen contenedores o repositorios para almacenar los recursos, que pueden ser utilizados de manera independiente o directamente asociados a la creación de cursos dentro del mismo sistema. Es decir que el repositorio puede estar disponible para que los profesores organicen los cursos o también pueden estar abiertos para que cualquier usuario recupere recursos no vinculados a ningún curso en particular, pero que les pueden ser de utilidad para reforzar los aprendido sobre algún tema.

El proceso de trabajo dentro de un LCMS requiere de control en cada fase del contenido, esto conlleva un proceso editorial para controlar la calidad de los contenidos creados, así como para permitir y organizar su publicación.

Como conclusión, se puede afirmar que tanto los LMS como los LCMS se pueden generalizar como sistemas de gestión de aprendizaje ya que los primeros gestionan la parte administrativa de los cursos, así como el seguimiento de actividades y avance del alumno; mientras que los segundos gestionan el desarrollo de contenidos, su acceso y almacenamiento. En el mercado, los más comunes son los LMS ya que la complejidad de los LCMS los ha llevado a un desarrollo más lento.

Cabe destacar en esta distinción, que el término LMS es a menudo usado para referirse indistintamente a un LMS o a un LCMS, aunque en estricto rigor un LCMS es un desarrollo posterior de un LMS.

### 4.4.1. Principales sistemas de gestión de aprendizaje

Existe una amplia variedad de plataformas de gestión de aprendizaje, tanto de código abierto como privativas, a continuación se presenta una breve descripción de algunas de ellas:

**Moodle:**

El proyecto Moodle fue iniciado en 1999 por Martin Dougiamas, de la Universidad Tecnológica de Curtin, Perth, Australia. Su nombre es un acrónimo de *Modular Object-Oriented Dynamic Learning Environment* (Ambiente de Aprendizaje Dinámico Orientado a Objetos Modular). Se distribuye gratuitamente como software libre bajo la Licencia Pública GNU y puede funcionar en Unix, GNU/Linux, OpenSolaris, FreeBSD, Windows, MacOS X, NetWare y otros que soportan PHP, lenguaje en el cual esta desarrollado y soporta varios tipos de bases de datos, en especial MySQL y PostgreSQL.

Tiene una interfaz de navegador de tecnología sencilla, ligera, eficiente y compatible, además promueve una pedagogía constructiva social: colaboración, actividades y reflexión crítica. El sitio es administrado por un usuario administrador, definido durante la instalación, el cual controla la creación de cursos y determina los profesores asignando usuarios a los cursos, además, los temas o *templates* permiten al administrador personalizar los colores del sitio, fuentes, presentación y otros ajustes para ajustarse a las necesidades del caso.

También pueden añadirse nuevos módulos de actividades a los ya instalados, incluso existen paquetes de idiomas, los cuales permiten una localización completa en casi cualquier idioma, actualmente hay disponibles paquetes para 43 idiomas incluyendo español y portugués.

Con respecto a la administración por parte de los usuarios, los estudiantes pueden crear su propio perfil en línea de si mismos, incluyendo fotos y otros detalles y cada usuario puede especificar su zona horaria y todas las fechas marcadas en Moodle se traducirán a esa zona horaria. Estas características, junto a otras más han convertido a Moodle en la plataforma de código abierto de este tipo más ampliamente usada en la actualidad

**Claroline:**

El proyecto Claroline fue iniciado en el año 2000 en el Instituto Pedagógico Universitario de Multimedia de la Universidad Católica de Lovain en Bélgica, por Thomas De Praetere, Hugues Peeters y Christophe Gesché, aunque desde 2004, el Centro de Investigación y Desarrollo del instituto Superior de Ingeniería Belga participa en el desarrollo del proyecto con apoyo externo.

Claroline es compatible con GNU/Linux, Mac OS y Windows y esta basado en tecnologías de código abierto como PHP y MySQL. Se distribuye bajo la GNU General Public License.

Claroline ha sido desarrollada siguiendo las necesidades y experiencias de profesores, ofreciendo una interfaz intuitiva y clara de administración de espacios. La gestión o manejo diario de la plataforma no requiere ninguna habilidad técnica especial. Se instala rápidamente y el uso de cualquier navegador Web permite manejar las distintas partes del curso y la admisión de usuarios con fluidez.

Esta herramienta esta organizada alrededor del concepto de espacios relacionados con un curso o actividad pedagógica. Cada espacio provee una lista de

herramientas que permite crear contenidos de aprendizaje y gestión de actividades de formación.

Entre sus características funcionales podemos señalar que Claroline permite la creación de directorios y subdirectorios para almacenar documentos y archivos accesibles a los usuarios, la creación completa de secuencias de actividades de aprendizaje, facilitar la colaboración de los usuarios durante el trabajo en grupo, herramientas online para el debate, así como calendarios y anuncios.

**Blackboard:**

La compañía de software Blackboard Inc. Con sede en Washington, DC, E.E.U.U. fue fundada en 1997, después de fusionarse a través de los años con otras empresas de desarrollo de cursos en línea pero reteniendo su nombre, para el 2005, Blackboard desarrolló y licenció aplicaciones de programas empresariales y servicios relacionados a varias instituciones educativas en muchos países. Estas instituciones usan el software de Blackboard para administrar e-Learning, procesamiento de transacciones, comercio electrónico (e-commerce) y manejo de comunidades en línea.

En la línea de productos Blackboard figuran la Blackboard Academia Suite y la Blackboard Commerce Suite (con fines más comerciales). La primera de estas consiste en:

- ***Blackboard Learning System:*** Un entorno de gestión de cursos, usada actualmente a nivel mundial por diversas instituciones relacionadas con la educación, en países como Colombia y México.
- ***Blackboard Community System:*** Para comunidades en línea y sistemas de portales.
- ***Blackboard Content System:*** Un sistema para la gestión de contenido.

Aunque el software de Blackboard es de código cerrado, la compañía proporciona una arquitectura abierta, llamada Building Blocas, que puede ser usada para extender la funcionalidad de sus productos.

**Atutor:**

Este proceso comenzó en 2002 en colaboración con el Adaptive Thecnology Resource Center de la Universidad de Toronto. Este centro es un líder mundialmente reconocido en el desarrollo de tecnologías y estándares que permitan a la gente con discapacidades el acceso a las oportunidades de e-Learning y esta misión ha influenciado profundamente el desarrollo de la plataforma, prestando especial interés a la accesibilidad

Atutor es un software de código abierto, desarrollado en PHP y que funciona con MySQL.

**Tabla 4.1 Comparación entre LMS.**

| LMS | Licencia | Especificaciones Técnicas | Origen |
|---|---|---|---|
| AMADeUs | Licencia Pública GNU | J2EE<br>PostgreSQL<br>Apahe Tomcat | Universidad Federal de Pernambuco. |
| Moodle | Licencia Pública GNU | PHP<br>MySQL<br>Unix<br>Windows<br>MacOS | Universidad Tecnológica De Curtin, Perth, Australia. |
| Claroline | Licencia Pública GNU | PHP<br>MySQL<br>Unix<br>Windows<br>MacOS | Universidad Católica de Lovain, Bélgica. |
| Blackboard | Licencia Pública GNU | De código cerrado | Washington, EE.UU. |
| Atutor | Licencia Pública GNU | PHP<br>MySQL<br>Apache | Universidad de Toronto, Canadá. |

La Tabla 4.1 muestra una comparación entre los LMS antes mencionados tomando en cuenta aspectos tanto técnicos como de su origen.

**Diferencias con AMADeUs:**

- Paquetes de idiomas: A diferencia de otros LMS, AMADeUs no cuenta con paquetes de idiomas, extensiones que permitan contar con las interfaces en mas de un idioma, a fin de llegar a más usuario y hacer posible su utilización en países o entre usuarios de otra lengua. Sin embargo, AMADeUs cuenta con la tecnología para lograr esto de manera rápida cambiando un único archivo.

- Personalización (templates): AMADeUs cuenta con una única interfaz, o más bien dicho no es posible personalizar la interfaz par cada usuario como en otros LMS que ofrecen una variedad de *templates* o temas en sus interfaces.

- Trayectoria: AMADeUs es un LMS relativamente nuevo, y que no es tan conocido como otros de distribución y uso más masivo (Moodle o Blackboard por ejemplo), aunque ha tenido un rápido crecimiento y esta posicionándose en más instituciones.

# 5. El sistema AMADeUs

## 5.1. Origen del sistema AMADeUs

AMADeUs es un **Sistema de Gestión de Aprendizaje** o LMS (por su sigla en inglés *Learning Managment System*) de código abierto desarrollado por el grupo **Ciencias Cognitivas y Tecnología Educacional** y el Centro de Informática de la **Universidad Federal de Pernambuco**, Brasil. AMADeUs es un acrónimo de Agentes Micromundos y Análisis de Desarrollo en el Uso de Instrumentos.

AMADeUs Está basado en el concepto de *blended learning* (aprendizaje mixto o flexible), el cual consiste en un proceso docente semipresencial, lo cual significa que un curso dictado en este formato incluirá tanto clases presenciales como actividades de e-Learning.

Esta plataforma es un LMS de segunda generación, es decir orientado a la integración de diversos servicios multimedia, entregando enseñanza a distancia a través de medios diversos y variados como juegos, vídeos, texto, audio, imágenes, simulaciones, estimulando formas innovadoras de interacción entre sus usuarios y entre éstos y el sistema. Todo esto se logra a través del acceso Web al sistema, que permite la visualización de información sobre la participación de los usuarios en los diferentes contextos de uso, llamadas percepciones.

## 5.2. Funcionalidades de AMADeUs

AMADeUs incorpora diversos modos de interacción entre sus usuarios con otros usuarios, con el sistema y con el contenido, en otras palabras esta caracterizado por ser un ambiente de enseñanza colaborativo, donde los profesores y alumnos pueden integrarse con el ambiente entre sí, siendo capaces de percibir las actividades y acciones de los participantes.

El sistema fue implementado utilizando métricas de diseño, percepción, evaluación y usabilidad, siendo éstos identificados a través de requerimientos con el objetivo de hacer de la utilización del ambiente algo más amigable.

Los profesores pueden gestionar el aprendizaje de sus alumnos difundiendo información a través del ambiente y posteriormente evaluándolos incluso de manera cualitativa, observando su capacidad de iniciativa, de interacción con los demás alumnos, de investigación de temas fuera de la clase, de trabajo en equipo y otros aspectos tratados en pruebas y actividades tradicionales, promoviendo y facilitando así la educación a distancia.

La plataforma Web de AMADeUs es considerada el corazón mismo de la aplicación, cuyo propósito principal es simplificar la conectividad entre los componentes del sistema y también dar soporte a otros aspectos importantes de la interacción y percepción de los usuarios, la prestación de servicios de sesiones, además de la distribución y descentralización. [Lopes *et al.*, 08]

La plataforma Web de AMADeUs está constituida por cuatro módulos principales los cuales son responsables de toda la base de apoyo al aprendizaje, siendo éstos:

- **Módulo de Registro:** Coordina los servicios de registro de usuarios y cursos en el ambiente, por lo tanto se ocupa de tareas para el registro de nuevos usuarios, en la cual son necesarias actividades de registro, actualización de datos, login de usuario, cambio de contraseña, solicitud de docencia, lista de usuarios. En cuanto a cursos se ocupa de tareas como registrar curso, buscar curso y validar curso.
- **Módulo de Gestión de Contenido:** Responsable de entregar los materiales que sea necesarios asociar a cada curso, es la parte encargada de la gestión de contenidos y componentes de aprendizaje en los diversos formatos que se presenten, además que permite la integración de las funcionalidades de evaluación.
- **Módulo de Evaluación:** Da soporte para que los profesores puedan evaluar a los alumnos en base a actividades realizadas, permite además verificar que éstas se han llevado ya a cabo o si están aún pendientes, por lo tanto permite un seguimiento de los alumnos dentro del sistema.

- **Módulo de Percepción Social**: Permite la aplicación del concepto de transparencia y percepción social. Su principal función es facilitar la comprensión y reducir las dificultades asociadas a las diferencias espacio-temporales de los usuarios. Básicamente proporciona información que rápidamente indica cuáles son los recursos disponibles y las actividades pendientes en cada uno de los contextos, por ejemplo, brinda información sobre los mensajes personales sin leer, la lista de usuarios de incurso en que se participa, etc.

Como ya se ha mencionado, AMADeUs no se restringe únicamente al entorno Web, sino que busca extender la experiencia del usuario ofreciendo varios recursos multimedia adicionales, desde mecanismos de comunicación tales como foros y salas de chat, hasta simuladores para la enseñanza de ciencias exactas como matemáticas o física. También se ofrecen herramientas para la interacción lúdica entre estudiantes y profesores, lo que contribuye a ampliar la experiencia de ambos, de entre estos componentes adicionales, podemos mencionar [Lopes *et al*., 08]:

- **Móviles:** Es la extensión para dispositivos móviles de AMADeUs, considera la gestión de contenidos poniendo a disposición de los usuarios una percepción casi total de los contenidos del entorno AMADeUs, como también información sobre cursos y la posibilidad de mantenerse actualizados de cambios generales en el sistema y de los cursos en que el usuario esta inscrito. La expansión a dispositivos móviles se transforma en una gran ventaja, posibilitando también la distribución de servicio de mensajes cortos (SMS).

- **Servidor de Juegos Multiusuario:** Su principal función es el desarrollo de juegos en un entorno en el que puedan participar tanto los alumnos como los profesores, este ambiente puede soportar múltiples usuarios, los cuales a través de la cooperación y sana competencia pueden ampliar las oportunidades de

interacción. De esta manera se contribuye a que la situación de la enseñanza sea más agradable y motivadora.

- **TV Digital:** Integra las tecnologías ya desarrolladas en una plataforma de aprendizaje que permita acceso a contenidos y formas interactivas de interacción por medio de TV digital interactiva. De acuerdo a los modelos del Sistema Brasilero de Televisión Digital (SBTVD). Con esto se pretende dar mayor nivel de interactividad, así como mayores facilidades de acceso a la información e interacción social, logrando que el alumno esté cada vez más cercano al aprendizaje.

- **Video Colaborativo:** Este módulo comprende el desarrollo de un entorno que permita que el reparto de videos por parte del profesor ocurra de forma intuitiva de modo que acompañen las acciones de los alumnos que previa codificación y análisis, puedan dar información al profesor respecto de la manera como los alumnos organizan sus acciones y como se sucede el aprendizaje de manera colaborativa.

- **EriMont:** Es otra de las extensiones de AMADeUs que consta de un entorno colaborativo síncrono, específicamente diseñado para la enseñanza de conceptos de física. Consiste básicamente en la monitorización de experimentos de física a distancia y en tiempo real, en el cual los usuarios colaboran en el montaje y realización de experimentos físicos. Gracias a su estructura, EriMont permite tanto a profesores y alumnos realizar experimentos físicos en los cuales otros actores del entorno puedan interactuar, ya sea como ejecutores del experimento o colaborando en la obtención de resultados.

## 5.3. Arquitectura de AMADeUs y tecnologías

AMADeUs implementa el **Modelo Vista Controlador** o MVC, un patrón de arquitectura de software que separa los componentes del sistema en datos de la aplicación, interfaz de usuario y lógica de control, en tres elementos distintos, llamados:

- **Modelo:** La representación específica de la información con la cual el sistema opera, es la representación del mundo real definida por medio de clases.
- **Vista:** Presenta el modelo en un formato adecuado para interactuar con el usuario.
- **Controlador:** responde a eventos, usualmente acciones del usuario invocando peticiones al modelo y a la vista.

Además**,** AMADeUs ha sido planteado y desarrollado como un sistema independiente tanto de la plataforma, como del sistema operativo y de otras tecnologías, teniendo este objetivo en mente se han elegido las tecnologías más adecuadas para su desarrollo, las cuales se procede a nombrar y a dar una breve explicación:

- **J2EE:** Conocida también como *Java Enterprise Edition* es un entorno de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java con arquitectura de N niveles y que se complementen con componentes software modulares ejecutándose en un servidor de aplicaciones.

- **Apache Struts:** Más conocida simplemente como Struts es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC y bajo la plataforma J2EE, ha sido desarrollada como parte del proyecto Jakarta de la *Apache Software Fundation* aunque actualmente es un proyecto independiente. Struts permite reducir el tiempo de desarrollo al ordenar tareas comunes a muchos proyectos de desarrollo Web, además, el hecho de que sea un software libre y su compatibilidad con las plataformas J2EE la hacen muy disponible.

- **Hibernate:** Es una herramienta de mapeo objeto-relacional para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones. En otras palabras Hibernate busca solucionar el problema de la diferencia entre los modelos de datos coexistentes en una aplicación, el usado en la memoria de la computadora y el usado en la base de datos.

- **IDE Eclipse:** Eclipse es un ambiente de desarrollo integrado o IDE (por su denominación en inglés *Integrated Development Enviroment*) para desarrollo de aplicaciones en java, aunque también puede ampliarse para otros lenguajes de programación. Eclipse cuenta con un sistema extensible de *plug-ins* o complementos para diversas funcionalidades y es un producto de software libre.

Teniendo en cuenta el modelo MVC y las tecnologías y herramientas descritas anteriormente, es posible lograr una mayor y mejor comprensión de la arquitectura del sistema AMADeUs. A continuación se describen las capas del sistema, con los elementos que las componen [Lopes, 08].

### 5.3.1. La capa de presentación

Esta es percibida por el usuario principalmente a través de las interfaces del sistema, presentadas por medio de páginas Web, cuyos elementos principales son:

- **Actions o Acciones:** Son elementos encapsulados por Struts, es un flujo de ejecución de una entrada de información a través de una página Web, consiguiendo con esto una programación más sencilla de las acciones en el sistema.
- **Forms o Formularios:** Es una representación de información que viaja entre la página y el sistema.
- **Filtros:** Con esto, se hace un pre-procesamiento de la información que proviene de la interfaz Web antes que sean ejecutadas las acciones del sistema.
- **Taglib o Librería de Etiquetas:** Facilitan la creación de las interfaces Web, con las etiquetas se pueden planificar y mantener de manera más eficiente las Java Server Pages (Páginas de Servidor de Java) o JSP, páginas dinámicas que soportan la arquitectura J2EE al poder contener código java.

### 5.3.2. La capa de negocio

Todo sistema esta compuesto por un conjunto de reglas de negocio, o sea, un flujo lógico que debe ser procesado para que obtengamos el resultado deseado. Una regla del negocio es representada en UML a través de un caso de uso. En el marco del análisis y diseño orientado a objeto existe un preocupación con respecto a la separación real de las reglas de negocio de las demás funcionalidades del sistema es esencial para tener un sistema de bajo acoplamiento y de fácil mantención y extensión.

Por ese motivo fue creado un elemento llamado Controlador dentro de la arquitectura. Un controlador es responsable de la ejecución de uno o más flujos de ejecución que so modelados en un caso de uso, o sea, podemos decir que el controlador es en si una implementación de una regla de negocio.

Cuando se elabora una arquitectura para el desarrollo de sistemas, principalmente orientados a objetos, es importante lograr una separación real de cada una de las capas que componen la arquitectura, en esto juega un importante papel el Registro. El registro es una clase perteneciente a la capa de negocio, responsable de tratar con las reglas referentes a la manipulación de datos con algún mecanismo de persistencia, manipulando esto a través de un DAO, concepto explicado más adelante. De modo que las principales tareas añadir, modificar, insertar y eliminar.

Con la creación del registro, separamos las reglas de negocio del acceso a los elementos de persistencia, un registro puede implementar reglas para el control de la última modificación de un dato, de esta manera el registro accede a la capa de persistencia a través de una interfaz, conocida como interfaz, que en AMADeUs corresponden a las clases de tipo InterfaceDAO (como AccessInfoDAO u OpenIDDataDAO), esta interfaz representa una separación total entre las capas de negocio y los datos, ya que el registro no sabe en realidad, que DAO lo esta utilizando.

### 5.3.3. La capa de datos

Como se ha dicho, un controlador necesita obtener información para su procesamiento y la obtiene a través de un registro, sin embargo el registro, no conoce qué mecanismo de persistencia está siendo utilizado y no tiene necesidad de saberlo, puesto que la información que es devuelta al registro y son recuperados a través de un componente llamado DAO (*Data Access Object*) u Objeto de Acceso a Datos.

Un DAO es un patrón responsable de manipular directamente cualquier mecanismo de persistencia necesario para el sistema. Además, un mecanismo de persistencia no necesita necesariamente ser un motor de base de datos, ya que también puede acceder y manipular archivos XML, archivos de texto o un servidor LDAP, entre otras formas de obtención de datos [Lopes, 08].

Esto se ha diseñado así, a fin de lograr que AMADeUs sea un sistema completamente independiente del mecanismo de persistencia. Por ejemplo, si un controlador necesita una determinada información, éste deberá hacer un pedido para el registro que, a su vez, buscará los datos a través de una interfaz DAO, pudiendo esta información venir de cualquier mecanismo de persistencia, sea éste base de datos, archivos XML, archivos de texto, etc.

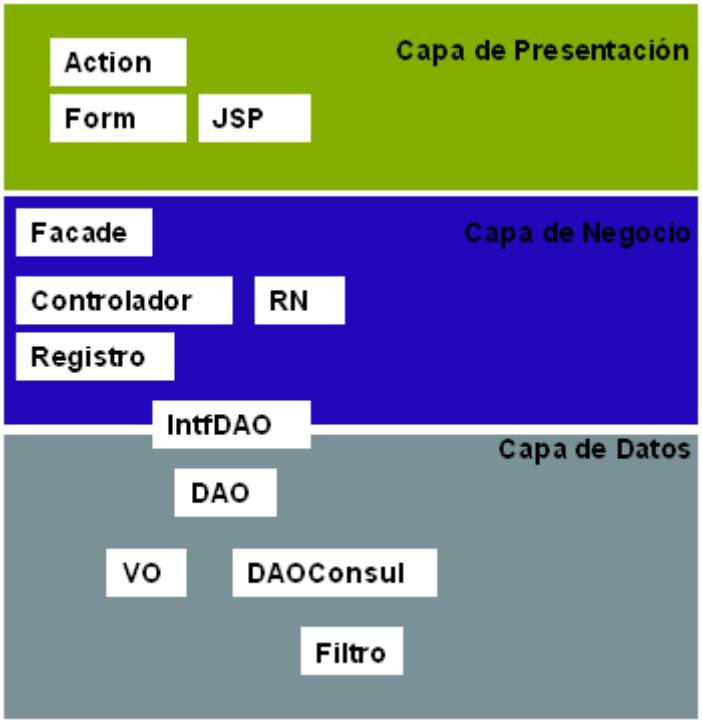
**VO DTO y Filtros:**

Con respecto a las reglas de negocio y el cómo estas son accesadas, ya se han señalado los principales elementos que componen esa capa dentro de la arquitectura, sin embargo, existen otros elementos que dan apoyo a la arquitectura como un todo. Si bien estos elementos son independientes de la capas, aparecen tanto en la capa de negocio, como en la capa de persistencia y también en la capa de presentación.

Dentro de estos elementos de apoyo podemos destacar los tres más utilizados y de mayor importancia para conseguir la independencia entre las capas, estos son:

**VO**: *Value Objects* u Objetos de Valor, son de esencial importancia para la arquitectura. Es a través de los VO que los datos viajan desde la capa de presentación a la capa de negocio y viceversa. Los VO son clases que encapsulan datos y métodos para manipularlos. Dentro de los VO podemos señalar dos principales, VOGenérico y VOEntidad, un VOGenérico es una implementación genérica de un VO, donde podemos guardar cualquier tipo de información, mientras que un VOEntidad es una representación de una tupla de una tabla de una base de datos. Con eso conseguimos facilitar la manipulación de datos contenidos dentro de una tabla de una base de datos en uso. En este caso, los VOEntidad son representados conforme a las especificaciones de VO de Hibernate.
**ODT**: DataTransfer Object u Objetos de Transferencia de Datos, son elementos que encapsulan diversos objetos para facilitar el transporte de los mismos dentro de la arquitectura, estos se usan por ejemplo, en la creación de tablas de visualización de datos y también para facilitar la paginación de datos.
**Filtros:** Son elementos que encapsulan datos que serán utilizados por los DAO para que estos pueda utilizarlos en una determinada consulta con criterios de selección.

**Figura 5.1 Arquitectura de AMADeUs**




En la Figura 5.1, puede apreciarse la arquitectura del sistema AMADeUs, con las partes y elementos antes descritos y explicados en el texto.

# 6. Autentificación con OpenID

OpenID es un sistema de autentificación digital descentralizado, con el que el usuario puede identificarse en un sitio Web a través de una URL, y puede ser verificado por cualquier servidor que soporte el protocolo.

OpenID es una manera fácil y rápida de obtener autentificación a través de la Web y de manejar la identidad virtual del usuario. Para obtener una cuenta de OpenID, basta con que el usuario cree una, lo cual es gratuito, en alguno de los muchos proveedores de OpenID existentes actualmente, una vez que esta cuenta existe, el usuario obtiene una identidad en la forma de una URL, la cual es una combinación del nombre de usuario escogido y el nombre del sitio proveedor, por ejemplo: ñhttp://cristóbal.lopez.myid.netò.

El usuario puede utilizar su nueva identidad de OpenID en cualquier sitio o sistema que cuente con soporte para esta tecnología, de esta manera, puede usar un único par de nombre de usuario y contraseña para identificarse en dos o más sitios, reduciendo el stress ocasionado por la necesidad de recordar varios pares de nombre de usuario y contraseña a continuación, se explicarán los elementos que componen el mundo de OpenID:

- **Proveedores de OpenID:** Son sitios o sistemas Web especializados en brindar autentificación OpenID, sus labores principales son el crear la identidad, almacenarla y proveer la autentificación a los sitios de los usuarios, los cuales son redirigidos a la página de su proveedor de OpenID al momento de solicitar la autentificación para llevar a cabo este proceso.

  Dado que un usuario puede escoger el proveedor de su preferencia, e incluso puede tener múltiples cuentas en múltiples proveedores, esto hace que OpenID como sistema de autentificación logre ser descentralizado, e independiente.

- **Fundación OpenID:** La Fundación OpenID es una organización internacional sin fines de lucro formada por individuos y compañías dedicadas a habilitar, promover y proteger las tecnologías OpenID. Fue formada en junio de 2007, la organización sirve como una organización de confianza pública que representa la comunidad abierta de desarrolladores, vendedores y usuarios. La fundación asiste a la comunidad proporcionando la infraestructura necesaria y ayuda en la promoción y da soporte a la expandida adopción de OpenID. Esto incluye la gestión de propiedad intelectual así como el crecimiento viral y la participación global en la proliferación de OpenID.

- **Sitios que soportan OpenID:** Hay una cantidad creciente de sitios y sistemas que soportan OpenID, muchos de ellos redes sociales virtuales y servicios de correo electrónico, de hecho, OpenID surgió a partir de comunidades de bloggeros, entre los más destacados, a causa de su gran cantidad de usuarios y popularidad son Google, Yahoo, Blogger.

## 6.1. La integración con OpenID

Para los sitios que buscan la integración con OpenID hay diferentes opciones, una manera de lograrlo es por medio de la utilización de plugins para los diferentes sistemas de gestión de contenido o CMS, muy usados para Web blogs, wikis o foros, existen plugins para Drupal, Wordpress, SPIP, phpBB. También hay servicios de host que ofrecen facilidades para aceptar OpenID en un sitio.

Finalmente, esta la opción usada en AMADeUs, la de utilizar una de las librerías desarrolladas para distintos lenguajes, éstas han sido desarrolladas por miembros de la comunidad OpenID. En el sitio Web oficial de la Fundación OpenID se encuentra un repositorio con varias de estas librerías, que abarcan lenguajes tales como: C#, Coldusion, Haskell, Java, Peral, PHP, Pitón, Ruby, y Squeak/Smalltalk.

La librería utilizada en AMADeUs es **OpenIDforjava**, en términos generales, a fin de realizar la integración sirviéndose de esta librería se debe descargar el archivo comprimido que contiene las clases necesarias e incluir estas en el proyecto que se esté desarrollando. Una vez hecho esto, dichas clases pueden utilizarse en combinación con otras clases añadidas al sistema.

## 6.2. Especificaciones de OpenID

Con el fin de asegurarse que toda integración con OpenID por parte de cualquier sistema esté en armonía con los principios generales de cómo OpenID debe ser implementado, evitando toda complicación innecesaria, de manera modular, libre y a la vez extensible, es que la comunidad de especificación técnica de OpenID, por medio de una lista de correo electrónico ha desarrollado las Especificaciones de OpenID (*OpenID Specifications*, http://openid.net/developers/specs/).

Hasta la fecha se cuenta con un total de seis especificaciones de carácter oficial, las cuales son:

- OpenID Authentication 2.0.
- OpenID Atribute Exchange 1.0
- OpenID Provider Authentication Policy Extensión 1.0.
- OpenID Authetication 1.1.
- OpenID Simple Registration Extensión.
- Yadis Discovery Protocol.

OpenID Authentication 1.0 fue reemplazada en 2007 por **OpenID Authentication 2.0**, la cual define el protocolo de autentificación de OpenID, es básicamente la directriz principal al momento de implementar OpenID, para poder entender el protocolo, es necesario comprender ciertos conceptos empleados en él y su significado dentro del contexto de la autentificación con OpenID:

- **Identificador** (*Identifier*): Se trata de una URI del tipo ôhttpô o ôhttpsô, o una XRI, corresponde a la la identidad de OpenID.
- **Agente-Usuario** (*User-Agent*): Es el navegador Web del usuario final.
- **Parte Confidente** *(Relaying Party)*: Es una aplicación Web que desea probar que el usuario final posee o controla un identificador.
- **Proveedor de OpenID** (*OpenID Provider*): Es un servidor de autentificación a la cual la parte confidente hace referencia para validar la posesión del identificador por parte del usuario final.

- **URL de Punto Final** (*OP Endpoint URL*): La URL que acepta mensajes del protocolo de autentificación, obtenidos al llevar a cabo el descubrimiento en el Identificador Provisto por el Usuario.
- **Identificador del Proveedor** (*OP Identifier*): Un identificador para un Proveedor de OpenID.
- **Identificador Provisto por el Usuario** (*User Proveided Identifier*): Es el identificador que ha sido presentado por el usuario final a la relaying party, o seleccionado por el usuario en su proveedor de OpenID. Durante la fase de inicialización del protocolo, un usuario final ingresa, ya sea su propio identificador o un identificador de su proveedor de OpeniD. Si este último es usado, el proveedor de OpenID debe ayudar al usuario final a seleccionar un identificador para compartir con la relaying party.
- **Identificador Reclamado** (*Claimed Identifier*): Un identificador que el usuario final declara poseer, el objetivo final del protocolo es verificar la posesión efectiva del mismo por parte del usuario final. Este Identificador Reclamado es el identificador obtenido al normalizar el Identificador Provisto por el Usuario, si es que era una URL.
- **Identificador Local del Proveedor de OpenID** (*OP-Local Identifier*): Un identificador alternativo para un usuario final que es local a un proveedor en particular, pero no está necesariamente bajo el control del usuario final.

Una vez explicados estos conceptos se esta en condiciones de explicar el Protocolo de Autentificación de OpenID:

El **Usuario Final** inicializa la autentificación al presentar un **Identificador Provisto por el Usuario** a la **Parte Confidente** a través del **Agente-Usuario**.

Después de normalizar el **Identificador Provisto por el Usuario**, la **Parte Confidente** ejecuta el descubrimiento y establece la **URL de Punto Final** del **Proveedor de OpenID**, la cuál el usuario usa para autentificación.

La **Parte Confidente** y el **Proveedor de OpenID** establecen una asociación usada por el **Proveedor** para firmar mensajes subsecuentes y por la **Parte Confidente** para verificar dichos mensajes; eso elimina la necesidad de futuros requerimientos directos para verificar la firma después de cada autentificación.

La **Parte Confidente** redirige el **Agente-Usuario** del **Usuario Final al Proveedor de OpenID** con un *authentication request de OpenID* (pedido de autentificación).

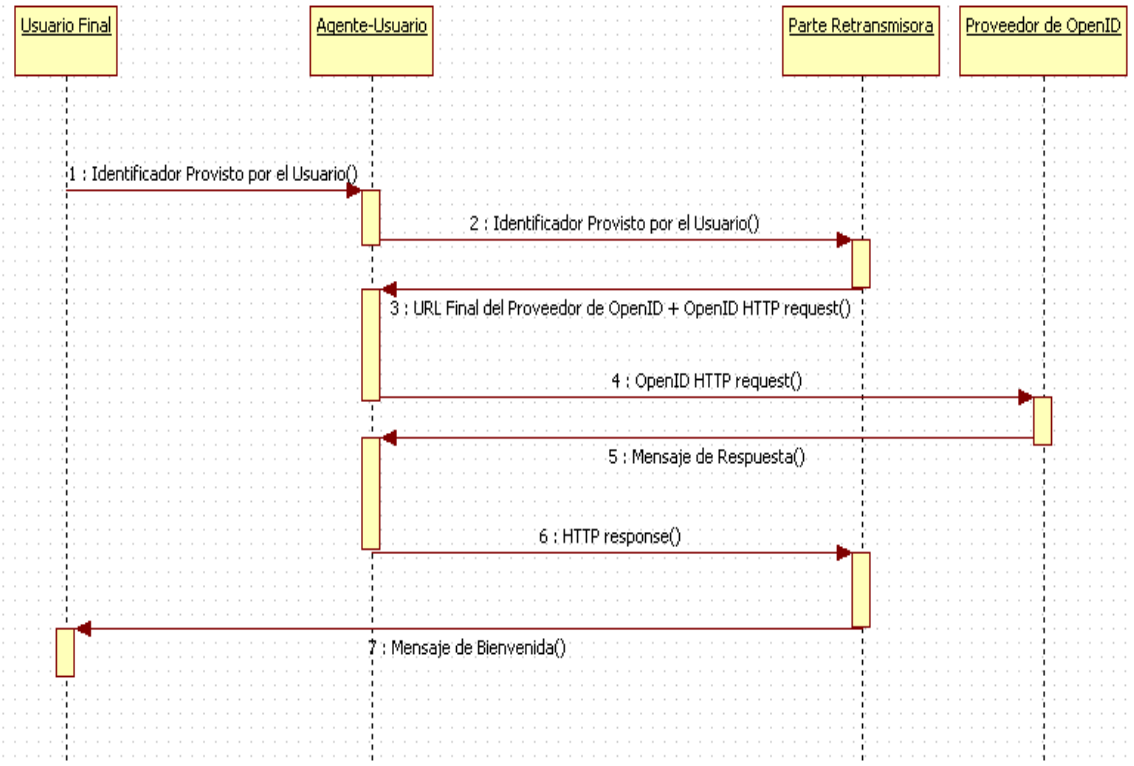
El **OP** establece si el **Usuario Final** esta autorizado para ejecutar autentificación de OpenID y si desea hacerlo. La manera en cual el usuario final se autentifica ante el **OP** y cualquier política referente a dicha autentificación dependen enteramente del proveedor y no forma parte del protocolo.

El **Proveedor de OpenID** redirige al **Agente-Usuario** del **Usuario Final** de regreso a la **Parte Confidente**, ya sea con una confirmación de que la autentificación está aprobada o un mensaje de autentificación fallida.

La **Parte Confidente** verifica la información recibida del **Proveedor de OpenID** incluso verificando la URL retornada, verifica la información de descubrimiento.

A fin de exponer esto de una manera más gráfica se presenta el siguiente diagrama de secuencia:

**Figura 6.1 Diagrama de secuencia del protocolo.**




En el diagrama se puede apreciar la comunicación entre los distintos elementos mencionados en el protocolo.

OpenID Provider Authentication Policy Extensión 1.0 es una extensión al protocolo de autentificación provee un mecanismo por el cual una aplicación Web puede solicitar la aplicación de alguna política de autentificación específica por parte del proveedor de OpenID, también provee un mecanismo por el cual el proveedor informa a la aplicación cuál política de autentificación ha sido usadas.

OpenID Atribute Exchange 1.0 es una extensión para intercambio de información de identidad entre usuario y proveedor, se proveen mensajes para recuperación y almacenamiento de información de identidad. **OpenID Simple Registration Extensión** es una extensión al protocolo de autentificación, para intercambio ligero de información de perfil. Esta diseñada para pasar ocho piezas comunes de información cuando un usuario se registra en un sitio. En cuanto a Yadis Discovery Protocol se desarrolla de manera independiente de OpenID.

# 7. Análisis

## 7.1. Requerimientos del módulo

Los requerimientos de un sistemas consisten en la descripción de los servicios proporcionados por el sistema así como sus restricciones operativas, es decir brindan información sobre lo que éste debe hacer y en algunos casos también sobre lo que no debe hacer, y en que condiciones [Sommerville, 05].

Una clasificación común de los requerimientos de un sistema es la que suele hacerse en requerimientos funcionales y no funcionales:

Requerimientos funcionales: son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que debe reaccionar ante entradas particulares y de cómo se debe comportar en situaciones particulares.

Requerimientos no funcionales: Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares, los requerimientos funcionales a menudo se aplican al sistema en su totalidad.

**Requerimientos funcionales:**

- El módulo debe dar a los usuarios del sistema la opción de ingresar a él usando su cuenta de OpenID.
- El módulo debe dar a los usuarios nuevos la oportunidad de registrarse con su cuenta de OpenID.
- El módulo debe verificar con el proveedor de OpenID correspondiente la validez de la cuenta (o nombre de usuario) ingresada por el usuario.
- El módulo debe brindar a los usuarios la posibilidad de agregar o remover nuevas cuentas de OpenID de su cuenta de Amadeus.

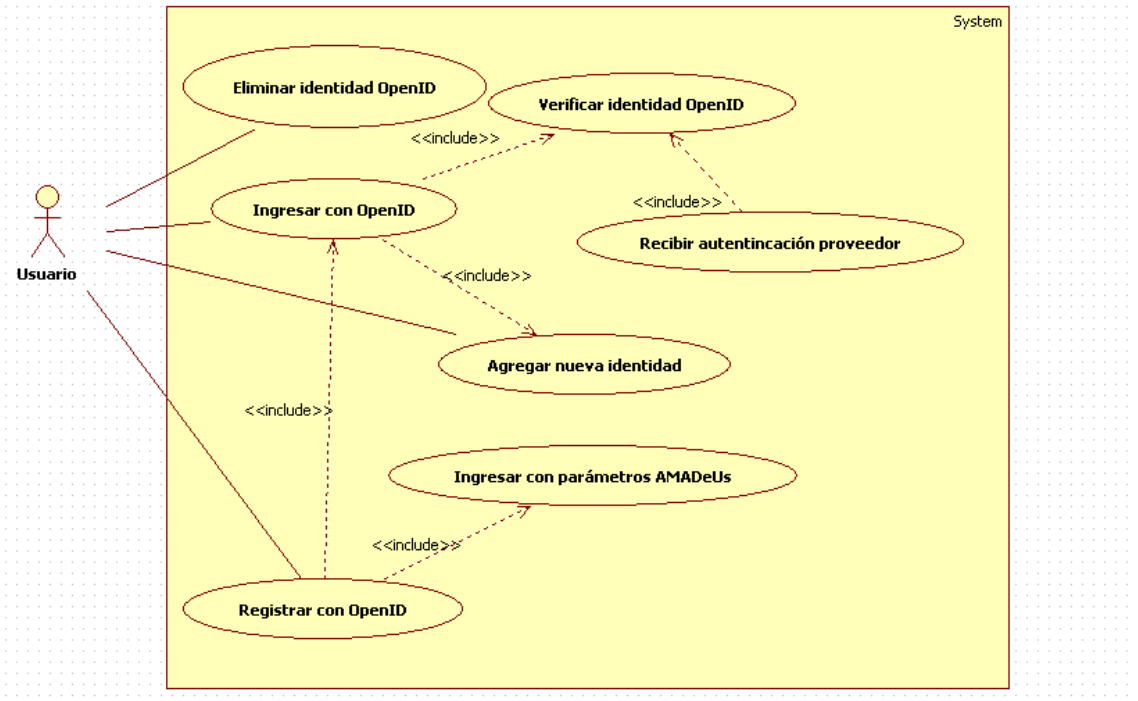
**Requerimientos no funcionales:**

- Cada usuario debe tener una única identidad interna en el sistema.
- Cada usuario puede tener muchas identidades de OpenID.
- Cada identidad de usuario debe corresponder a un único usuario.
- El sistema y en particular, este módulo deberá funcionar con los navegadores Mozilla Firefox e Internet Explorer.

## 7.2. Casos de uso

A continuación figuran una serie de diagramas y otros documentos que representan el funcionamiento del módulo de integración con OpenID.

**Figura 7.1 Diagrama de casos de uso.**




No se detalla el caso de uso Solicita Información de Perfil, ya que no ocurre dentro del sistema, y lo lleva a cabo el proveedor de OpenID con el usuario en el sitio Web del primero.

**Especificación de casos de uso:**

| | |
|---|---|
| Nombre | Ingresar con una cuenta de OpenID |
| Actores | Usuario, Proveedor de OpenID |
| Prioridad | Alta. |
| Entradas | El usuario ingresa una cuenta de OpenID. |
| Precondiciones | El usuario debe existir previamente en el sistema |
| Salidas | El usuario ingresa al sistema Amadeus |
| Flujo principal | 1. El usuario selecciona la opción de ingresar al sistema con su identidad de OpenID.<br>2. El sistema le solicita que ingrese su nombre de usuario de OpenID<br>3. El sistema verifica la identidad con el proveedor de OpenID<br>4. El sistema verifica la existencia de dicha cuenta de OpenID dentro de las de los usuarios del sistema. |
| Flujo secundario | 4.1 Si la cuenta de OpenID no existe previamente en el sistema, va a caso de uso Agregar Nueva Cuenta de OpenID. |

| Nombre | Agregar nueva cuenta de OpenID. |
|---|---|
| Actores | Usuario. |
| Prioridad | Alta. |
| Entradas | El usuario posee una identidad de OpenID no ingresada al sistema. |
| Precondiciones | El usuario ingresa al sistema Amadeus. |
| Salidas | El usuario ingresa al sistema Amadeus. |
| Flujo principal | 1. El sistema da la opción al usuario. de agregar su identidad de OpenID a su cuenta del sistema.<br>2. El usuario selecciona la opción de agregar su identidad de OpenID.<br>3. El sistema solicita al usuario que ingrese su nombre de usuario y contraseña del sistema.<br>4. El usuario ingresa exitosamente sus datos.<br>5. El sistema informa al usuario que la nueva identidad se ha agregado a la información de su cuenta de Amadeus. |
| Flujo secundario | No se contempla flujo secundario. |

| Nombre | Eliminar identidad de OpenID de la cuenta de usuario. |
|---|---|
| Actores | Usuario. |
| Prioridad | Media. |
| Entradas | El método recibe la identidad de OpenID. |
| Precondiciones | El usuario se encuentra previamente logeado en el sistema. |
| Salidas | Se ha eliminado la identidad de OpenId deseada. |
| Flujo principal | 1. El usuario selecciona la opción de administrar su información de sus identidades de OpenID.<br>2. El sistema muestra al usuario la lista de su o sus identidades de OpenID con la información correspondiente.<br>3. El usuario selecciona le eliminación de una de ellas.<br>4.El sistema despliega un mensaje informando que la acción se ha llevado a cabo |
| Flujo secundario | No se contempla flujo secundario. |

| Nombre | Registrarse con una identidad de OpenID en Amadeus. |
|---|---|
| Actores | Usuario, Proveedor de OpenID. |
| Prioridad | Alta. |
| Entradas | Datos de identidad del usuario. |
| Precondiciones | El usuario cuenta con una identidad de OpenID.<br>El usuario no se encuentra registrado en el sistema. |
| Salidas | Se ha registrado un nuevo usuario a partir de una identidad de OpenId. |
| Flujo principal | 1. El usuario se autentifica con una identidad de OpenID.<br>2. El usuario selecciona la opción de registrarse en el sistema con su identidad de OpenID.<br>3. El usuario rellena los campos que el sistema le solicita a fin de registrarse.<br>4. El usuario se halla registrado en el sistema con su identidad de OpenID. |
| Flujo secundario | No se contempla flujo secundario. |

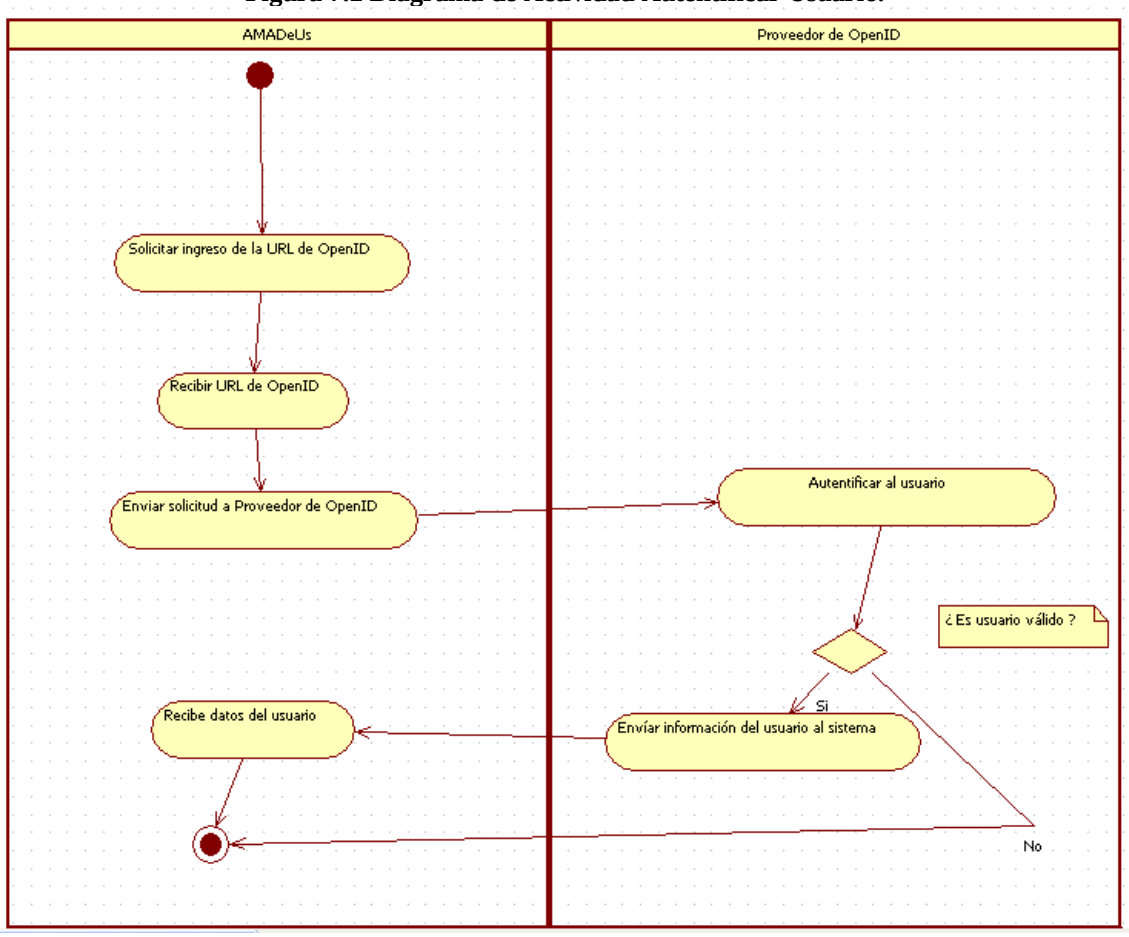
## 7.3. Diagrama de actividades

El propósito del diagrama de actividades es modelar un proceso de flujo de trabajo, servicios proporcionados por un objeto.

Cómo se podrá observar, en el diagrama de actividades correspondiente a este módulo, se cuenta con 2 calles o carriles, representando las responsabilidades del usuario y el proveedor de OpenID.

El diagrama de secuencia de la figura 7.2 muestra el flujo del sistema desde que se solicita el ingreso de la URL de OpenID hasta que habiendo consultado con el proveedor se recibe su respuesta, la división se ha hecho para facilitar la comprensión y no complicar la lectura del diagrama.

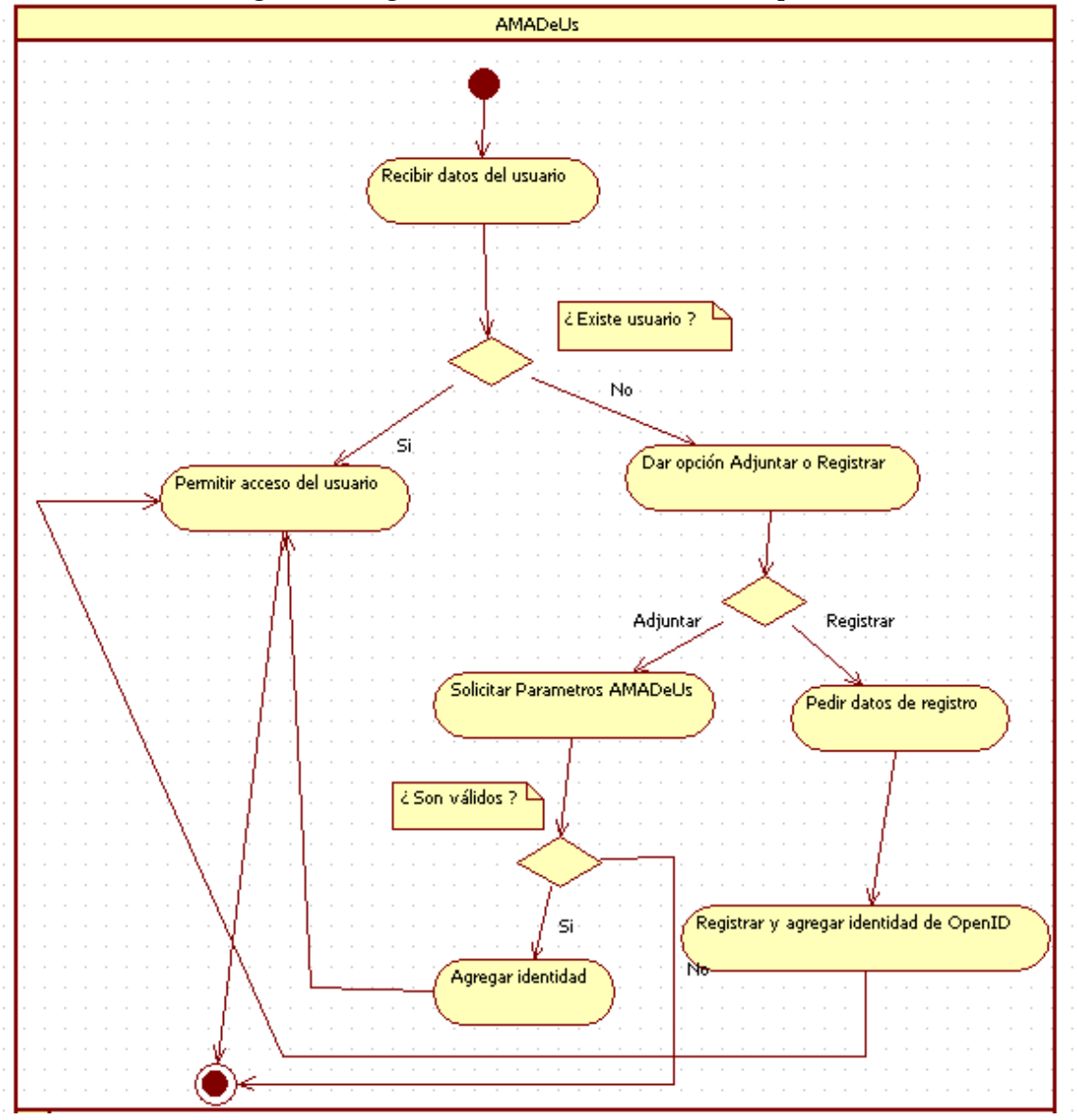
**Figura 7.2 Diagrama de Actividad Autentificar Usuario.**




La figura 7.3 muestra el flujo del sistema desde la recepción de la respuesta del proveedor. Al recibir la respuesta del proveedor el sistema verifica que dicha identidad o URL de OpenID esté asociada a un usuario dentro de AMADeUs, si es así, permitirá que dicho usuario ingrese al sistema. De lo contrario, podrían darse dos situaciones, primero, que el usuario exista en el sistema pero no halla adjuntado esta identidad de OpenID a su cuenta de AMADeUs o segundo, que el usuario de OpenID no exista en AMADeUs. En el primer caso se le solicitarán al usuario que ingrese sus parámetros de

AMADeUs y en el segundo se le dara la opción de registrarse en AMADeUs, incluso con datos de su perfil de OpenID que el halla decidido compartir.

**Figura 7.3 Diagrama de Actividad Administrar respuesta**



# 8. Diseño e implementación

En esta parte del documento se presenta el diseño y la implementación correspondientes a la integración con OpenID, lo que incluye diagramas de secuencia de UML mostrando la interacción de los distintos objetos del módulo, así como muestras del código fuente.

## 8.1. Diseño

Un diseño de software es una descripción de la estructura del software que se va a implementar, los datos que son parte del sistema, las interfaces entre los componentes del sistema y, algunas veces, los algoritmos utilizados. [Sommerville, 05]

En el capítulo 5 ya se ha tratado la arquitectura del sistema AMADeUs. El presente trabajo se halla inmerso en él, por lo que parte importante de la etapa de diseño consistió en la comprensión de la arquitectura de dicho sistema, la comprensión de las capas y las tecnologías (como Struts e Hibernate) implicadas. A continuación se presentan los diagramas de secuencia que explican la interacción de los distintos objetos dentro del módulo de integración con OpenID. Es importante señalar que se muestra su interacción con objetos ya pertenecientes al sistema, por ejemplo la clase UserAction a la que se le han agregado 3 métodos y se han intervenido otros 2.

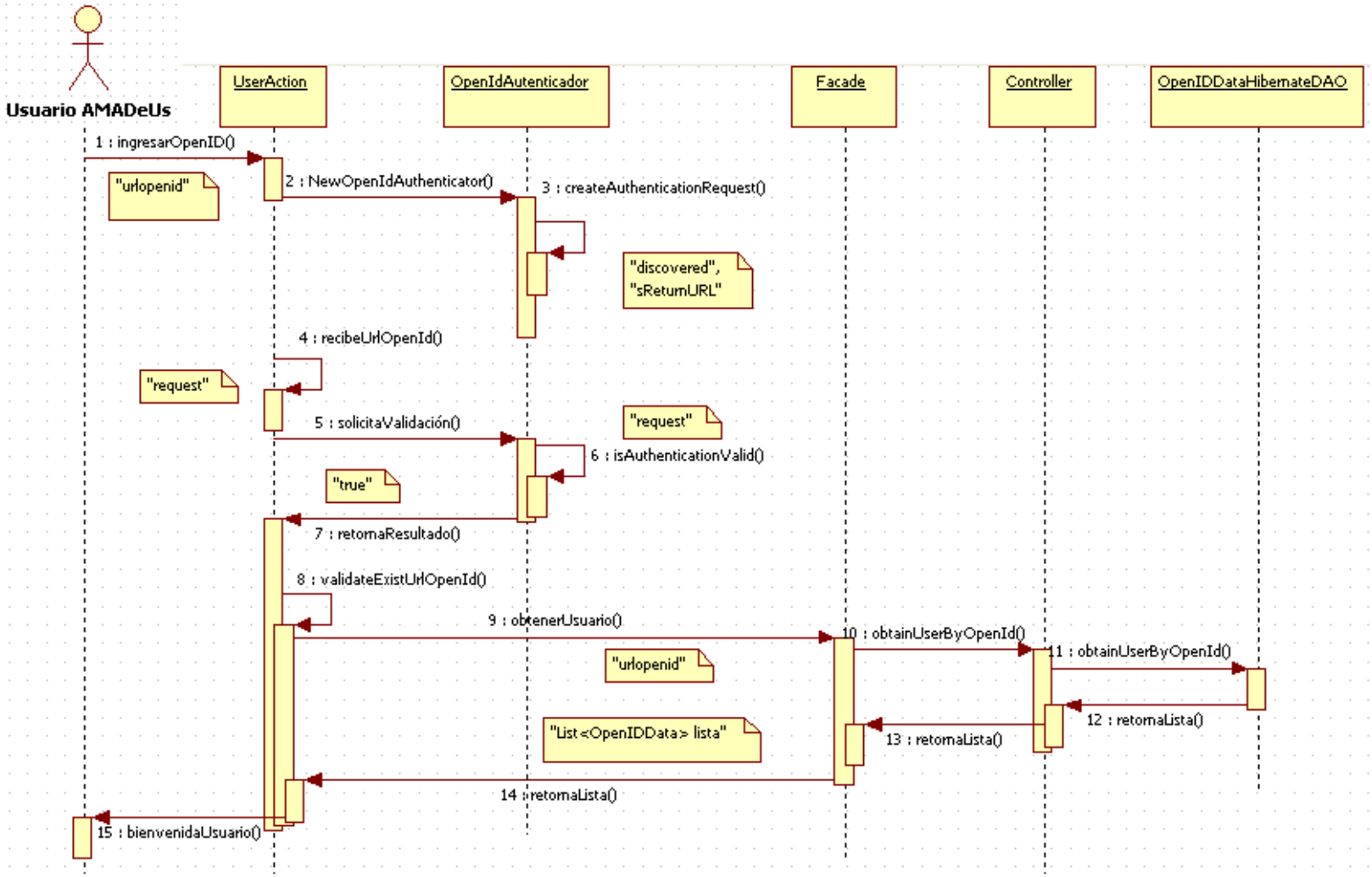
Se ha prescindido de la utilización de otros diagramas aparte de los que no figuran en este documento, por considerar que no eran necesarios o que, en el caso por ejemplo, de los diagramas de colaboración, repetirían información aportada por los diagramas de secuencia

En la figura 8.1 se puede apreciar el diagrama de secuencia correspondiente al proceso de ingresar a AMADeU con OpenID, dicho diagrama es similar a la figura 6.1, la del protocolo de OpenID.

Este diagrama representa como el usuario introduce su URL de OpenID, la cual llega a la clase UserAction, la que crea un objeto del tipo OpenIdAutenticador para que en base a dicho parámetro realice el descubrimiento necesario y pueda crear la solicitud de autentificación.

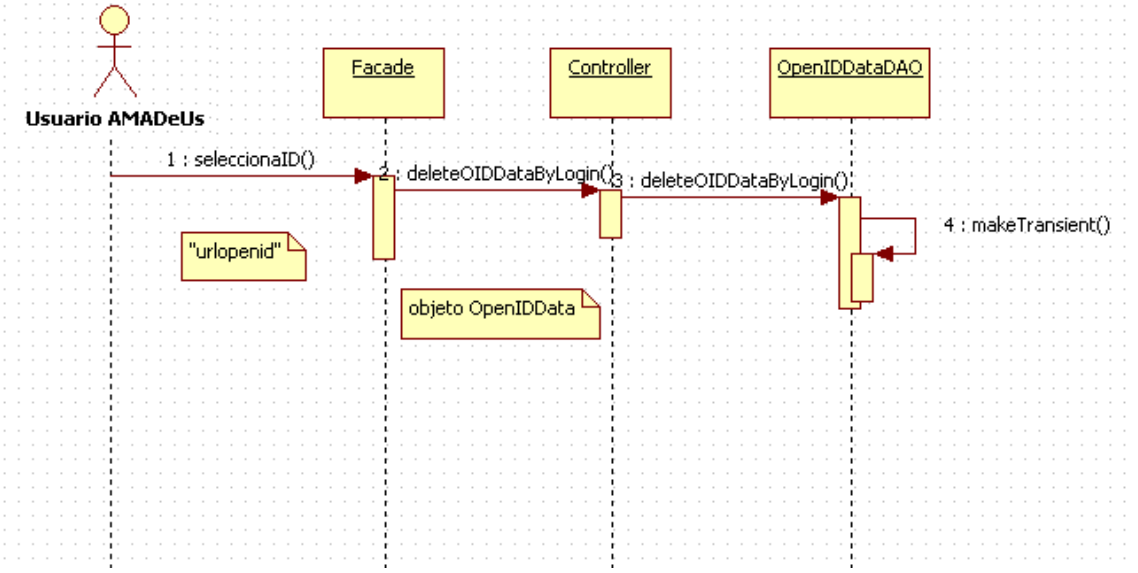
Cuando se analiza la respuesta del proveedor respectivo y esta es afirmativa, se valida que la URL de OpenID corresponda a un usuario de OpenID por medio del método validateExistUrlOpenId, la cual hace uso del método obtainUserByOpenId para obtener el par de URL de OpenID y login correspondiente al usuario, a fin de permitirle el ingreso al sistema.

**Figura 8.1Diagrama de Secuencia Ingresar con OpenID.**



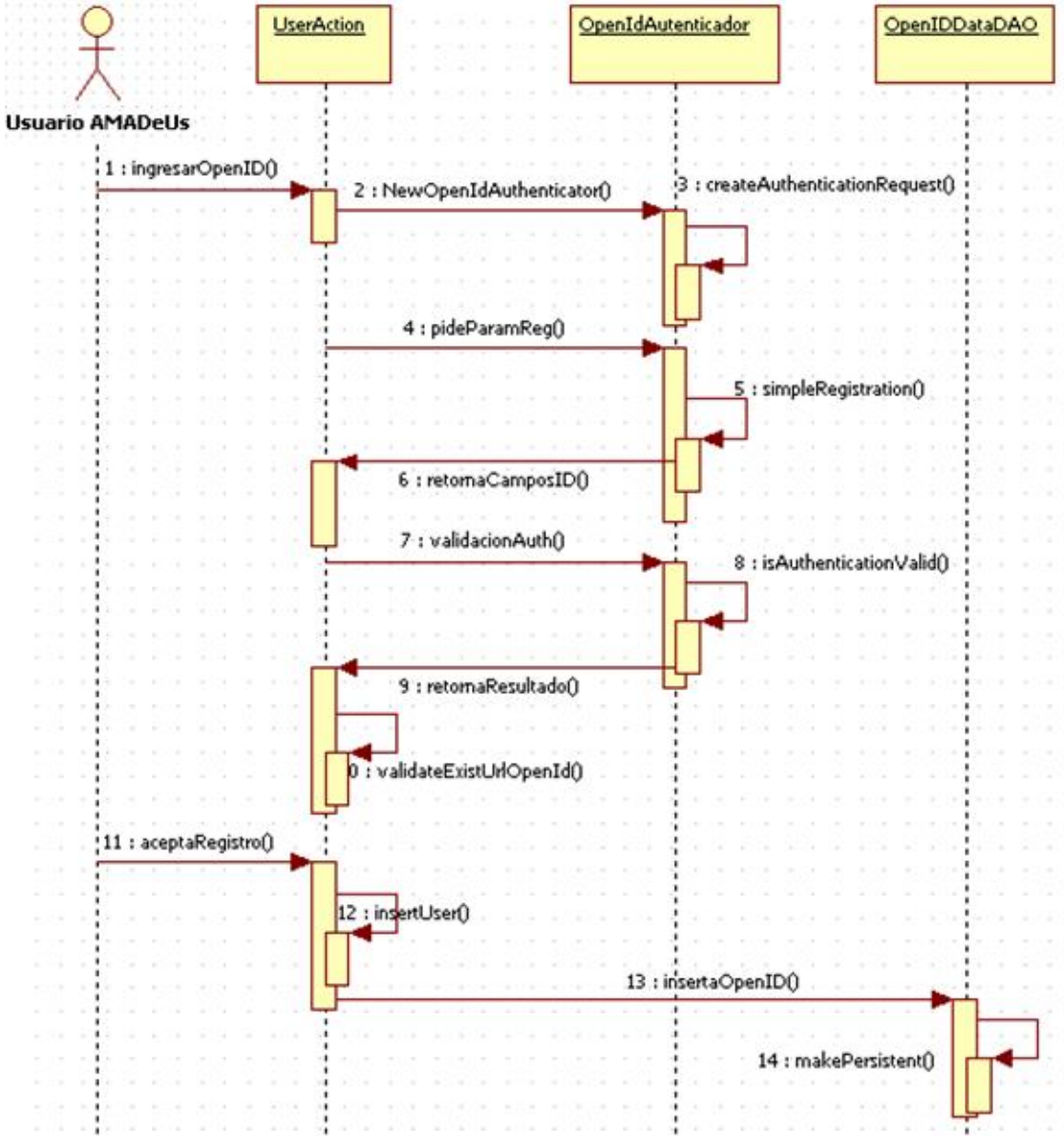
En la figura 8.2 puede verse el diagrama de secuencia correspondiente al proceso de eliminar una identidad de OpenID de la cuenta de AMADeUs de un usuario, la URL seleccionada para su eliminación o desvinculación de la cuenta es recibida a través de un formulario en una página JSP, y esta solicita a la clase facade que use el método deleteOIDDataByLogin para eliminar la fila de la tabla correspondiente a ese par de URL y login del usuario(es decir un objeto del tipo OpenIDData), esto es llevado a cabo finalmente por el método makeTransient

**Figura 8.2 Diagrama de Secuencia Eliminar Identidad de OpenID.**



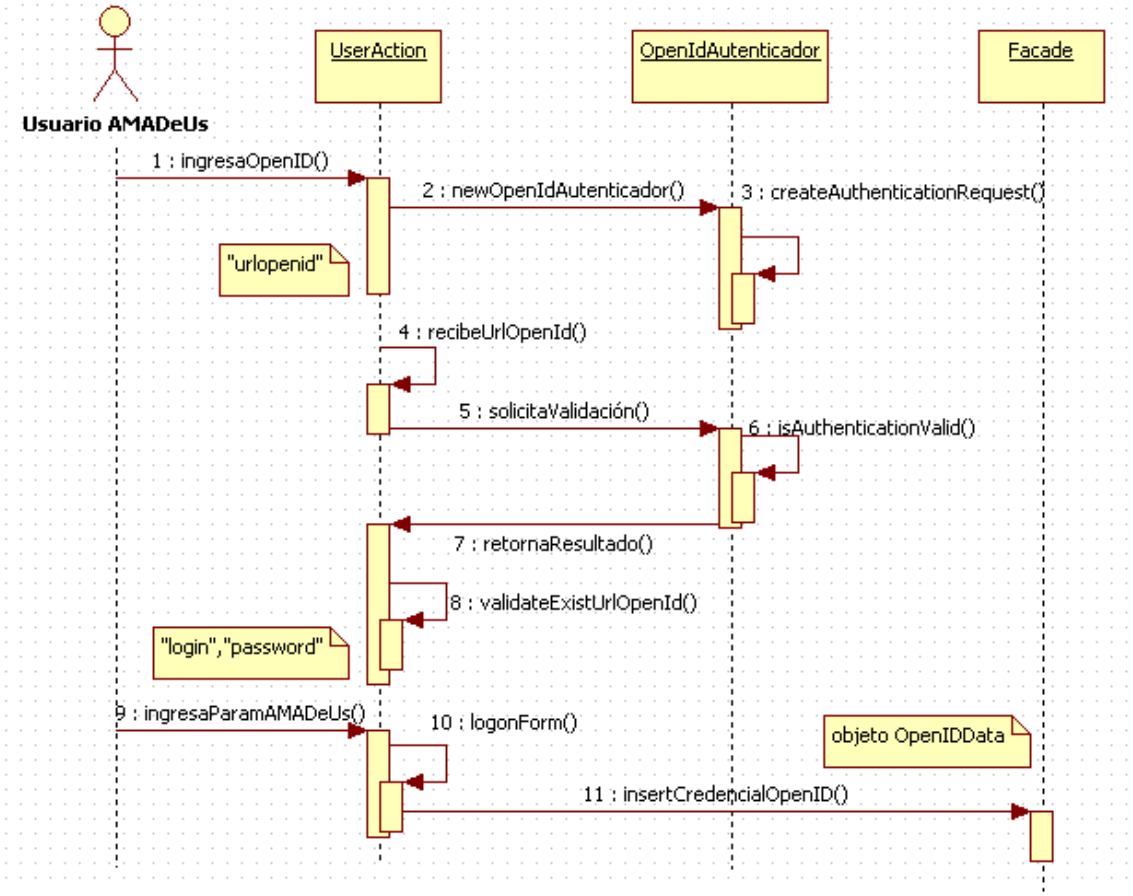

En la figura 8.3 puede verse el diagrama de secuencia correspondiente al proceso de registrarse con OpenID. La particularidad que tiene este proceso es que el método createAuthenticationRequest, aparte de generar la solicitud de autentificación se encarga de especificar cuales son los parámetros de identidad que el sistema AMADeUs requiere de sus usuarios (ninguno de los cuales es obligatorio), los cuales son el nombre completo, el correo electrónico y el nombre de usuario o *nickname*. Más tarde se ve que el método simpleRegistration es usado para extraer los parámetros recibidos del proveedor de OpenID a fin de almacenarlos en la sesión del usuario para que le sean mostrados en el formulario de registro de AMADeUs.

**Figura 8.3 Diagrama de Secuencia registrarse con OpenID.**




En la figura 8.4 se muestra el diagrama de secuencia correspondiente al proceso de adjuntar una nueva identidad de OpenID a la cuenta de usuario de AMADeUs, el proceso es similar al de registrar un nuevo usuario, una vez que el usuario se ha autentificado con su proveedor de OpenID y se comprueba que no existe en el sistema, se le da la opción de adjuntar esta identidad en el caso de que se trate efectivamente de un usuario de AMADeUs, por lo que, luego de ingresar su nombre de usuario y contraseña, se le permite el ingreso al sistema mientras que en paralelo, se adjunta su nueva identidad para que el sistema pueda reconocerla la siguiente vez que el usuario la use.

**Figura 8.4 Diagrama de Secuencia Adjuntar una identidad.**




## 8.2. Prototipos de la integración con OpenID

El desarrollo del módulo de integración con OpenID para el sistema AMADeUs comprende la generación y entrega de prototipos, tal como se menciona en la planificación del proyecto. A continuación figura una descripción de los dos prototipos estables y principales del módulo (hubo otros prototipos no estables que no se considerarán):

### 8.2.1. Primer prototipo

El primer prototipo estable del sistema, contaba ya con la funcionalidad básica de la integración con OpenID, se comunicaba con el proveedor respectivo y reconocía la identidad previamente almacenada.

Entre las falencias o reparos hallados al revisar este prototipo y que se planteo mejorar en la siguiente versión, fueron las siguientes:

- **Conexiones implícitas en el código:** Muchos métodos añadidos a una clase anterior o los métodos de una clase nueva, al momento de establecer conexión con la base de datos, tenían el código fuente necesario ócen duroô, lo cual hacía

engorrosa la presentación del código e iba en contra de las buenas prácticas de codificación.

- **Necesidad de configurar Proxy en una clase:** Dado que el sistema debe comunicarse con el servidor de un proveedor de OpenID a fin de que se realice la autentificación del usuario, era necesario tener en cuenta, la posibilidad de que la conexión a internet empleada por algún usuario precisara de un determinado Proxy para funcionar, si bien estaba contemplada con anticipación dicho escenario, sólo era posible ajustar el Proxy ingresando directamente a uno de los archivos .java, y manualmente modificarlo, lo cual, al igual que en el caso anterior hacia engorrosa la comprensión del código.

- **Ausencia de manual de usuario:** En este punto, no se contaba con un manual de usuario que fuera capaz de explicar de manera clara la forma de instalación y configuración de los distintos parámetros a ajustar en función de lo mencionado en los párrafos anteriores.

### 8.2.2. Segundo prototipo

El segundo prototipo cumplía principalmente con suplir las falencias encontradas en el anterior, en el sentido de hacer más cómoda para el usuario la tarea de instalar y configurar los datos necesarios para que el sistema pueda ejecutarse de manera correcta, e independientemente de la conexión usada por él, procediéndose por lo tanto a:

- **Agrupar código recurrente:** Como se ha dicho anteriormente, las nuevas conexiones a la base de datos y la configuración del Proxy exigían a usuario una tarea bastante engorrosa, esto se ha reparado, escribiendo el código necesario para exportar dichos datos desde un archivo XML donde pueden ser configurados o establecidos por el usuario sin la necesidad de que éste deba modificar las clases y métodos en el código fuente de la aplicación.

- **Redacción manal de usuario:** Con el objetivo de simplificar la instalación y configuración para el usuario, se redactó un manual, el cual explica paso a paso las tareas necesarias para poder ejecutar el sistema y poder usarlo con todas sus funcionalidades.

### 8.2.3. Tercer prototipo

El tercer prototipo fue desarrollado para ajustar de mejor manera la construcción del módulo de integración con OpenID a la estructura de AMADeUS, particularmente en la utilización del framework Hibernate, por lo que se procedió a crear nuevas clases que pudieran hacer esto posible, en el tercer prototipo se ha hecho:

- **Hibernate:** Se procedió a la creación de una clase correspondiente a la tabla que almacena los datos de las identidades de OpenID adjuntadas por cada

usuario, así como de otras clases creadas a partir de ella, necesarias para el funcionamiento con Hibernate y que se detallan más adelante.

- **Modificación de métodos:** Se modificaron algunos métodos ya existentes que realizaban la inserción de usuarios y otras consultas respecto a la verificación de las identidades de OpenID en AMADeUs.
- **Funcionalidad:** Adición de la opción de eliminar identidades que se hallan adjuntado previamente en el sistema AMADeUs.

### 8.2.4. Cuarto prototipo

El cuarto prototipo consistió básicamente en la implementación de la especificación para el registro de los usuarios con las identidades que ellos deseen aportar de su perfil de OpenID, disminuyendo así el tiempo empleado en el registro y llenado de los campos que AMADeUs exige a sus nuevos usuarios

## 8.3. Métodos y clases

En la implementación del módulo de integración con OpenID se ha debido, añadir al código fuente de AMADeUs ciertas clases con sus respectivos métodos para posibilitar la comunicación con los proveedores de OpenID, así como la modificación de ciertos métodos a fin de incorporar ciertas funciones relativas a la integración (tal como al adjuntar nuevas identidades). También se han hecho modificaciones en los archivos de configuración propios de los frameworks usados. Antes de proceder a describir el código fuente, es muy importante comprender ciertos conceptos relativos al funcionamiento del framework Struts:

Struts es, como se ha dicho, un framework que permite implementar el modelo MVC en aplicaciones Web en J2EE, dicho de otra manera, permite separar las capas de negocio, datos y presentación en estas aplicaciones, un elemento muy importante en Struts es el archivo de configuración, que consiste en un archivo XML que contiene información sobre los métodos, sus entradas y la forma en que deben ser tratadas sus salidas, definiendo otros elementos usados recurrentemente en la aplicación, entre los elementos más importantes están los **ActionMapping**, que se utilizan para definir cada una de las posibles acciones a realizar por la aplicación, relacionando el nombre de la solicitud o *path* con su *type*, es decir, la clase Java que será encarada de resolverlo.

Los **ActionForm** son elementos que relacionan un objeto de la clase java *bean* con un formulario HTML. Los **ActionForward** definene los posibles rumbos que puede tomar una acción al completarse relacionando el nombre lógico del forward con el nombre físico de un recurso como una página JSP. Un *action* de Struts es una instancia de una subclase de una clase Action, la cual implementa una porción de una aplicación Web y cuyo método retorna un ActionForward.

De esta manera, a través del archivo de configuración, Struts controla quién hace qué en la aplicación, relacionando las páginas JSP y las entradas y salidas del usuario y hacia el usuario con las clases y métodos que se harán cargo de procesarlas. Por ejemplo, en la figura 6.4 Archivo de Configuración de Struts puede apreciarse un ejemplo de **ActionMapping**:

**Figura 8.5 Archivo de Configuración de Struts.**

```
<action path="/redirectToOpenidProvider" parameter="logonFormByOpenid"
   type="br.ufpe.cin.amadeus.amadeus_web.struts.action.UserActions"
    name="logonFormByOpenid" input="/jsp/user/formlogonopenid.jsp"
                            scope="request">
                                </action>
```

Este extracto indica que: Lo ingresado por el usuario en la página **formlogonopenid. jsp** será recibido y manejado por la clase **UserActions,** por el método correspondiente al parámetro ***ñ/redirectToOpenidProviderò***

A continuación figura una lista de estas adiciones y modificaciones al código fuente, con explicaciones de su funcionamiento.

**La clase OpenIdAuthenticator:**

Esta es la clase que se encarga de implementar los métodos que realizan la comunicación con OpenID, utilizan las funciones proporcionadas por la librería OpenIDforjava, esta clase consta de 2 métodos, los cuales son:

- **El método createAuthenticationRequest:** Este método crea una solicitud (request) de autentificación, recibe un string conteniendo la URL del usuario o identidad de OpenID, obtiene el proveedor correspondiente a esa URL, lo que se denomina *discovery* o descubrimiento y retorna un ActionForward con la información necesaria para el envío de la solicitud al ser utilizado en el método redirectToOpenidProvider de la clase UserActions (explicado más adelante).

- **El método isAuthenticationValid:** Este método es el encargado de validar la autentificación, recibe la respuesta el proveedor de OpenID que viene como un HTTP request. Es invocado por el método recibeUrlOpenId de la clase UserAction (explicado más adelante).

- **El métodosimpleRegistration:** Este método es el encargado de obtener los datos de identificación para el nuevo usuario de AMADeUs. Recibe un HttpServletRequest y retorna un arreglo de String conteniendo los 3 campos que AMADeUs exige a sus nuevos usuarios.

**Figura 8.6 El método createAuthenticationRequest**

```
/**
Crea una nueva sollicitud de Autentificacion.
@param a_sOpenIdUrl La URL de OpenID del usuario.
@param a_oRequest El request a usar.
@return El ActionForward usado para redirigir al usuario a su
proveedor.
@throws OpenIDException
@author zschwenk
*/
public ActionForward createAuthenticationRequest(String a_sOpenIdUrl,
HttpServletRequest a_oRequest)
            throws OpenIDException
      {

ActionForward afOpenId = null;
```

```
      RealmVerifier oRealmVerifier = new RealmVerifier();
      oRealmVerifier.setEnforceRpId(false);
      m_oManager.setRealmVerifier(oRealmVerifier);

//Obtiene los valores del proxy desde el archive de configuration

      AnnotationConfiguration confhib = new AnnotationConfiguration();
      confhib.configure();
      String proxy=
      confhib.getProperties().getProperty("hibernate.proxy").toString(
      );
      String port=
confhib.getProperties().getProperty("hibernate.proxyPort").toString();
      String useproxy=
confhib.getProperties().getProperty("hibernate.Useproxy").toString ();
            int port2 = Integer.parseInt(port);

/**
La URL a la que el proveedor de OpenID deberá enviar su respuesta.
Para configurar dicha URL se debe ir al archivo src/hibernate.cfg.xml
             * */

            AnnotationConfiguration confhib2 = new
            AnnotationConfiguration();
            confhib2.configure();
            String url
      =confhib2.getProperties().get("hibernate.returnurl")
      .toString();
            url=url + "recibeLoginURLOpenId.do?method=recibe";
            String sReturnURL = url;
//"http://localhost:8080/AmadeusLMS404/recibeLoginURLOpenId.do?method=
//recibe";

//El código para el registro de usuarios, el que pide los atributos de
//identificación

            SRegRequest sregReq = SRegRequest.createFetchRequest();
            sregReq.addAttribute("fullname", true);
            sregReq.addAttribute("nickname", false);
            sregReq.addAttribute("email", false);



//Establece los valores del proxy si son necesarios
            if(useproxy == "no")
            {
               ProxyProperties proxyProps = new
               ProxyProperties();
               proxyProps.setProxyHostName(proxy);
               proxyProps.setProxyPort(port2);
               HttpClientFactory.setProxyProperties(proxyProps);
            }

// ejecuta el descubrimiento a partir del Identificador provisto por
el usuario

            List discoveries = m_oManager.discover(a_sOpenIdUrl);
// intenta conectar con el proveedor de OpenID
// y reenvía un punto final para autentificación
            DiscoveryInformation discovered =
```

```
m_oManager.associate(discoveries);

// almacena la información del descubrimiento en la session   //del
usuario

            a_oRequest.getSession().setAttribute("openid-disc",
discovered);
            a_oRequest.getSession().removeAttribute("openid");

a_oRequest.getSession().setAttribute("openid",discovered.getDelegateId
entifier());//



// obtiene un mensaje AuthRequest para ser envíada al proveedor de
//OpenID
       AuthRequest authReq = m_oManager.authenticate(discovered,
sReturnURL);


       authReq.addExtension(sregReq);



            if (! discovered.isVersion2() )
            {
            // Opcion 1: Obtiene la dirección del proveedor de OpenID
            // El único método soportado por OpenID 1.x
            // La URL usualmente esta limitada a ~2048 bytes



                  afOpenId = new
ActionForward(authReq.getDestinationUrl(true), true);
            }
            else
            {
// Opcion 2: HTML FORM Redirection (Allows payloads >2048 bytes)

                  afOpenId = new
ActionForward(authReq.getDestinationUrl(true), true);
             a_oRequest.getSession().setAttribute("parameterMap",
authReq.getParameterMap());
             a_oRequest.getSession().setAttribute("destinationUrl",
authReq.getDestinationUrl(true));
            }

            return afOpenId;
                                    }
```

**Figura 8.7 El método isAuthenticationValid.**

```
public boolean isAuthenticationValid(HttpServletRequest a_oRequest)
            throws OpenIDException
      {

boolean bAuthenticationValid = false;


// extrae los parámetros desde la respuesta (response) de
```

```
autentificación
// (la cual viene como un HTTP request desde el proveedor de OpenID)
ParameterList lstResponse =
new ParameterList(a_oRequest.getParameterMap());

// recupera la información del descubrimiento previamente almacenada

DiscoveryInformation discovered =
(DiscoveryInformation)a_oRequest.getSession().getAttribute("openid-
disc");

a_oRequest.getSession().removeAttribute("openid");

a_oRequest.getSession().setAttribute("openid",discovered.getDelegateId
entifier());


// extrae la URL desde el HTTP request
  StringBuffer receivingURL = a_oRequest.getRequestURL();
  String queryString = a_oRequest.getQueryString();
  if (queryString != null && queryString.length() > 0)
  receivingURL.append("?").append(a_oRequest.getQueryString());



// verifica la respuesta; ConsumerManager debe ser la misma instancia
// (static) usada para colocar la solicitud de autentificación
//(authentication request)

 VerificationResult verification = m_oManager.verify(
 receivingURL.toString(),
 lstResponse, discovered);

// examina el resultado de la verificación y extrae el identificador
//verificado
  Identifier verified = verification.getVerifiedId();

  if (verified != null)
  {
   // AuthSuccess authSuccess =
  //  (AuthSuccess) verification.getAuthResponse();

  bAuthenticationValid = true;
  }

  return true;
}
```

**Figura 8.8El método simpleRegistration.**

```
public String[] simpleRegistration(HttpServletRequest srRequest)
throws OpenIDException{
            String rt="";
            String IdSet[]={"","","",""};

            ParameterList lstResponse = new
ParameterList(srRequest.getParameterMap());

AuthSuccess autsuccess = AuthSuccess.createAuthSuccess(lstResponse);
```

```
  if(autsuccess.hasExtension(SRegMessage.OPENID_NS_SREG)){

      MessageExtension ext =
autsuccess.getExtension(SRegMessage.OPENID_NS_SREG);

      if(ext instanceof SRegResponse)
      {
      SRegResponse sregResp = (SRegResponse) ext;

      String fullName = sregResp.getAttributeValue("fullname");
      String nickName = sregResp.getAttributeValue("nickname");
      String elmail = sregResp.getAttributeValue("email");

      IdSet[0]= fullName;
      IdSet[1]= nickName;
      IdSet[2]= elmail;

      }

  }
  return IdSet;
      }
```

**La clase OpenIDData:**

Esta clase fue creada para administrar (y crear) la tabla del mismo nombre de la base de datos, por lo tanto, existe una correspondencia directa entre sus atributos ñloginò que representa el login de un usuario y ñuseropenidò que representa una identidad de OpenID de un usuario y las columnas de la tabla., sus métodos son lo que se conoce como getters and setters o accesotes y modificadores, y son:

**Figura 8.9 La clase OpenIDData.**

```
public class OpenIDData implements Serializable {

      @Id // establece useropenid como PK
      private String useropenid;
      private String login;


      public OpenIDData() {
      }

  // los métodos getters y setters

      public String getUseropenid() {
            return useropenid;
      }

      public void setUseropenid(String useropenid) {
            this.useropenid = useropenid;
      }

      public String getLogin() {
            return login;
      }

      public void setLogin(String login) {
```

```
            this.login = login;
        }
        }
```

El método getUseropenid: Retorna el valor del atributo ñuseropenidò.

- **El método setUseropenid:** Establece el valor del atributo ñuseropenidò.
- **El método getLogin:** Retorna el valor del atributo ñloginò.
- **El método setLogin:** Establece el valor del atributo ñloginò.

## La clase interfaz OpenIDDataDAO:

Es una interfaz, es decir, una colección de métodos abstractos y propiedades en la que se especifica lo que se debe hacer pero no su implementación, que será llevada a cabo por las clases que la implementan, en este caso, esta interfaz es implementada por la clase OpenIDDataHibernateDAO.

**Figura 8.10 La clase interfaz OpenIDDataDAO.**

```
public interface OpenIDDataDAO extends GenericDAO<OpenIDData, Integer>
{
      public List<OpenIDData> getOpenIDDataByLogin(String login);
      public List<OpenIDData> getOpenIDDataByOpenId(String openid);

}
```

## La clase OpenIDDataHibernateDAO:

Esta clase hereda de la clase hereda de la clase GenericHibernateDAO e implementa los métodos definidos en la interfaz OpenIDDataDAO, estos métodos se encargan de realizar consultas a la base de datos, y son:

- **El método getOpenIDDataByLogin:** que recibe el login de un usuario y devuelve una lista donde cada elemento es del tipo OpenIDData, es decir contiene un login y una identidad de OpenID. Sirve para saber si una identidad de OpenID existe en la base de datos.

**Figura 8.11 El método getOpenIDDataByLogin.**

```
public List<OpenIDData> getOpenIDDataByLogin(String login)
{
List<OpenIDData> results = getSession().createSQLQuery("SELECT * from
openiddata where login = '"+ login +
"'").addEntity(OpenIDData.class).list();
       //Retorna lista de los usuarios con ese login
            return results;
}
```

- **El método getOpenIDDataByOpenId:** Es similar al método anterior, este recibe una identidad de OpenID y retorna una lista donde cada elemento es del tipo OpenIDData, es decir contiene un login y una identidad de OpenID. Sirve para obtener las identidades de un usuario.

**Figura 8.12 El método getOpenIDDataByOpenId.**

```
public List<OpenIDData> getOpenIDDataByOpenId(String openid){

List<OpenIDData> results = getSession().createSQLQuery("SELECT *
from OpenIDData where useropenid = '"+ openid +
"'").addEntity(OpenIDData.class).list();

 //Retorna lista de los usuarios con ese OpenID
      return results;

      }
```

## La clase UserActions:

Esta clase se encarga de realizar todas las operaciones propias de la gestión de los usuarios, esta es una clase previamente existente en AMADeUs, recibe entradas a sus métodos que están especificadas en el archivo de configuración de Struts, se le han añadido 3 nuevos métodos a esta clase:

- **El método redirectToOpenidProvider:** Se encarga de recibir la identidad de OpenID y de llamar al método createAuthenticationRequest de la clase OpenIdAuthenticator para que este realice la solicitud de autentificación ante el proveedor de OpenID correspondiente.

**Figura 8.13 El método redirectToOpenidProvider.**

```
//Redirige la solicitud (request) al proveedor de OpenID
correspondiente, para comprender mejor el método véase en la clase
OpenIdAuthenticator

 public ActionForward redirectToOpenidProvider(ActionMapping mapping,
ActionForm form,HttpServletRequest request, HttpServletResponse
response)
throws Exception
{
            ActionForward aForward = null;
            DynaActionForm dyna = (DynaActionForm) form;
            ActionMessages messages = new ActionMessages();

            String urlopenid = dyna.getString("urlopenid");
            request.getSession().setAttribute("openid2", urlopenid);
            OpenIdAuthenticator autenticadorRequest = new
OpenIdAuthenticator();


            aForward =
```

```
autenticadorRequest.createAuthenticationRequest(urlopenid,request);

            return aForward;

}
```

- **El método recibeUrlOpenId:** Se encarga de recibir la respuesta del proveedor de OpenID y de interpretar su respuesta, retornando valores entendibles para Struts.

**Figura 8.9 El método recibeUrlOpenId.**

```
//Recibe la solicitud desde del proveedor de OpenID, retorna un
actionforward que toma valores comprensibles para el framework
struts

 public ActionForward recibeUrlOpenId(ActionMapping mapping,
ActionForm form,
                  HttpServletRequest request, HttpServletResponse
response)
                  throws Exception {

            ActionForward aForward = null;

            OpenIdAuthenticator autenticadorRequest = new
OpenIdAuthenticator();

            if(autenticadorRequest.isAuthenticationValid(request)){
                  aForward = mapping.findForward("success");

            }else{
                  aForward = mapping.findForward("error");
            }

            return aForward;
      }
```

- **El método validateExistUrlOpenId:** Una vez que la respuesta del proveedor ha sido recibida, este método verifica si la URL ingresada por el usuario esta asociad a algún usuario dentro de AMADeUs, retornando si es así o no a través de Struts.

**Figura 8.14 El método validateExistUrlOpenId.**

```
 //Una vez que la respuesta del proveedor de OpenID ha sido recibida,
este método verifica si la URL provista por el usuario esta asociada a
alguna cuenta de Amadeus
 //retorna un action forward, si es 'success',permite al usuario
correspondiente ingresar a Amadeus

 public ActionForward validateExistUrlOpenId(ActionMapping mapping,
ActionForm form,  HttpServletRequest request, HttpServletResponse
response)
                  throws Exception {

      String urlopenid =
(String)request.getSession().getAttribute("openid2");
```

```
 /* busca algún usuario asociado a esa URL de OpenID*/
      List<OpenIDData> lista = facade.obtainUserByOpenId(urlopenid);


      if (lista.size()>0)
      {
            OpenIDData credencial = lista.get(0);

/*Crea un objeto para contener la información de acceso de un usuario
en particular*/

      AccessInfo user=new AccessInfo();
      user.setLogin(credencial.getLogin());
      user= facade.obtainAccessInfoByLogin(user.getLogin());
      AccessInfo accessInfo = facade.logon(user.getLogin(),
user.getPassword());
      request.getSession().removeAttribute("user");
      request.getSession().setAttribute("user", accessInfo);

      return mapping.findForward("success");

      }else{
            return mapping.findForward("newAssociationURLOpenId");
      }
      }
```

Además de los métodos recién descritos, hay ciertos métodos que ya estaban desarrollados en esta clase y que se les añadió algunas líneas de código a fin de servir a funcionalidades de la integración con OpenID, y que son:

- **El método insertUser:** este método recibe valores de un formulario ingresados por un usuario que esta registrándose en AMADeUs, este método hace uso del método insertCredentialOpenID.

**Figura 8.15 Código extra para el método InsertUser.**

```
/* Aqui comienza código adicional para este método
            Establece una identidad de OpeID para este usuario*/

      OpenIDData credencialOpenId = new OpenIDData();
      String useropenid ="";
         try{
           useropenid =
request.getSession().getAttribute("openid2").toString();

            }
            catch (Exception e) {
                  useropenid="";
            }
            String login = myForm.getString("login").toString();

            credencialOpenId.setLogin(login);
            credencialOpenId.setUseropenid(useropenid);



            if (!passConfirmation.equals(accInfo.getPassword()))
                  messages.add("confirmation", new ActionMessage(
                              "errors.confirmation.invalid"));
```

```
            try {
                  facade.insertPerson(p);

                  if(useropenid!=""){

      facade.insertCredencialOpenID(credencialOpenId);

                  }


                  facade.confirmRegistry(p.getAccessInfo());
            } catch (InvalidUserException e) {
                  messages.add("confirmation", new
ActionMessage(e.getMessage()));
            }
```

- **El método logonForm:** Este es el método que permite a los usuarios ingresar al sistema después de ingresar sus parámetros de identificación de AMADeUs, se le ha añadido líneas de código para que un usuario pueda adjuntar una nueva identidad de OpenID cuando ya se ha identificado con un proveedor y ya existe en AMADeUs.

**Figura 8.16 El código añadido para el método logonForm.**

```
// Aqui comienza código adicional para este método
// Adjunta una identidad de OpenID (una URL) para el usuario existente

                  try {
            AccessInfo accessInfo = facade.logon(login, password);

            request.getSession().removeAttribute("user");
            request.getSession().setAttribute("user", accessInfo);

            String useropenid = "";
            OpenIDData credencialOpenId = new OpenIDData();
               try{
                   useropenid =
request.getSession().getAttribute("openid2").toString();

                  }catch (Exception e) {
                  useropenid="";
                  }


/*crea una credencial para inserter o adjuntar una nueva URL de OpenID
*/
            credencialOpenId.setLogin(login);
            credencialOpenId.setUseropenid(useropenid);

            if (useropenid!="")
            {

      credencialOpenId=facade.insertCredencialOpenID(credencialOpenId)
;

                        }
                         request.removeAttribute("openid2");
```

## La clase Facade:

Esta clase recibe su nombre del patrón facade, este objeto (el objeto de esta clase) proporciona una especie de interfaz simplificada para un código fuente más extenso, la clase Facade usada en AMADeUs es una lista de métodos que a su vez llaman a otros métodos los cuales realizan el trabajo requerido y que se hallan en la clase Controller.

## La clase Controller:

Esta clase implementa métodos definidos en la clase Facade, retornándole los valores a sus métodos homólogos en esa clase, los métodos añadidos a esta clase son:

- **El método obtainUserByOpenId:** Recibe una identidad de OpenID y retorna una lista con el login del usuario que posee esa identidad, hace uso del método getOpenIDDataByOpenId previamente explicado.

**Figura 8.17 El método obtainUserByOpenId.**

```
/**
       * Retorna un arreglo conteniendo
       * la lista con el login del usuario correspondiente
       * a una URl de OpenID especifica
       *
       *
       * ***/


public List<OpenIDData> obtainUserByOpenId(String urlopenid) {
OpenIDDataDAO dau = factory.getOpenIDDataDAO();


List<OpenIDData> coleccion = dau.getOpenIDDataByOpenId(urlopenid);

return coleccion;
      }
```

- **El método deleteOIDataByLogin:** Recibe un objeto correspondiente a una par de login/identidad de OpenID, o visto de otra manera una fila de la tabla OpenIDData, con el fin de eliminarla, esto en el caso de que un usuario desee eliminar una identidad de su cuenta de AMADeUs.

**Figura 8.18 El método deleteOIDataByLogin.**

```
public void deleteOIDataByLogin(OpenIDData row) {
            OpenIDDataDAO dao =factory.getOpenIDDataDAO();
            dao.makeTransient(row);
}
```

- **El método insertCredentialOpenID:** Recibe un objeto del tipo OpenIDData y lo hace persistente, es decir, lo inserta en la tabla del mismo nombre, es la antítesis de método anterior.

**Figura 8.19 El método insertCredentialOpenID.**

```
public OpenIDData insertCredencialOpenID(OpenIDData credencialOpenId)
{

       OpenIDDataDAO dao1 =factory.getOpenIDDataDAO();
       return dao1.makePersistent(credencialOpenId);


      }
```

# 9. Validación

La validación del software o, de forma más general, la verificación y validación (V&V) se utiliza para que el sistema se ajusta a su especificación y que cumple con las expectativas de quién lo comprará. La verificación y validación tienen lugar en cada una de las etapas del proceso de software, comenzando con revisiones de requerimientos y continúa con revisiones al diseño, inspecciones de código y prueba del producto [Sommerville, 05].

Existen dos aproximaciones en el proceso de V&V, las cuales se complementan en el análisis y verificación de un sistema:

- Las inspecciones de software: analizan y comprueban la documentación, como documento de requerimientos y diagramas de diseño, así como también el código fuente. Son técnicas estáticas, ya que no requieren la ejecución del software.
- Las pruebas de software: examinan las salidas o respuestas del software en base a datos de prueba. Son una técnica dinámica, ya que requiere la ejecución del software.

A continuación, se explica como se emplearon estas técnicas en el presente desarrollo.

## 9.1. Inspecciones de software

Las inspecciones de software fueron llevándose a cabo en todos los procesos del desarrollo, validando los requerimientos, los diagramas de diseño y el código fuente del módulo.

- **Requerimientos:** Como no se tenía claridad original con respecto a los requerimientos del sistema, estos fueron establecidos por el alumno y aceptados por el personal del Proyecto AMADeUs.

- **Diagramas de diseño:** Se realizó la inspección de los diagramas de diseño (diagramas UML tales como diagramas de casos de uso, de actividades, etc.). Esta fue llevada a cabo por los profesores en instancias de evaluación y por el alumno.

- **Código fuente:** Fue siendo inspeccionado por los profesores en instancias de evaluación y corregido de una instancia a otra.

## 9.2. Pruebas de software

Las pruebas del software tienen como objetivos:

- Demostrar al desarrollador y al cliente que el software satisface sus requerimientos. Por lo tanto se realizó al menos una prueba para cada requerimiento del software construido.
- Descubrir defectos en el software en el que su comportamiento no es deseable o correcto. Muchas veces los casos de pruebas relacionados con este objetivo fueron hechos en base a las caídas observadas del sistema.

La validación del módulo de integración con OpenID se fue realizando a medida de cada prototipo estaba listo o se consideraba que debería estarlo. La manera de validarlo consistió en la realización de pruebas de software, más específicamente pruebas de **caja negra** o funcionales, en las cuales el módulo se prueba de acuerdo a sus funcionalidades analizando sus entradas y sus salidas [Sommerville, 05].

En un principio se encontraron diversos errores en los casos de pruebas, tales como la permanencia de una identidad al seleccionar eliminarla, el no reconocimiento de una identidad previamente almacenada en el sistema.

**Caso de prueba: Ingresar al sistema con una identidad de OpenID:**

**Objetivo:** Encontrar errores en el ingreso al sistema AMADeUs usando una identidad de OpenID previamente almacenada en el sistema.
**Precondiciones:** El usuario ya existe en el sistema y tiene una identidad de OpenID adjuntada a su cuenta.
**Poscondiciones:** El usuario ingresa al sistema, dándole este la bienvenida.
**Datos:** Una identidad de OpenID.
**Observaciones:** En los tres casos se observa finalmente que el usuario logra ingresar con éxito al sistema.

**Tabla 9.1 Caso de Prueba Ingresar al sistema**

| Datos de entrada | Resultados esperados | Resultados obtenidos |
|---|---|---|
| dojeda - http://dmoescobar.myid.net/ | Ingresa al sistema | Ingresa al sistema |
| dojeda - http://jonspenlow.myopenid.com/ | Ingresa al sistema | Ingresa al sistema |
| dojeda - http://danielojeda.openid.es/ | Ingresa al sistema | Ingresa al sistema |

**Caso de prueba: Adjuntar una nueva identidad de OpenID**

**Objetivo:** Encontrar errores en el proceso de adjuntar una nueva identidad de OpenID a su cuenta de AMADeUs
**Precondiciones:** El usuario ya existe en el sistema y posee una identidad de OpenID que no esta adjuntada a su cuenta de AMADeUs.
**Poscondiciones:** El usuario logra adjuntar una nueva identidad de AMADeUs a su cuenta siendo capaz de visualizarla.
**Datos:** Una identidad de OpenID.

**Observaciones:** En los tres casos se observa finalmente que el usuario logra adjuntar con éxito su identidad a su cuenta en el sistema.

**Tabla 9.2 Caso de Prueba Adjuntar una identidad**

| Datos de entrada | Resultados esperados | Resultados obtenidos |
|---|---|---|
| http://dmoescobar.myid.net/ - dojeda | Adjunta la identidad | Adjunta la identidad |
| http://jonspenlow.myopenid.com/- dojeda | Adjunta la identidad | Adjunta la identidad |
| http://danielojeda.openid.es/- dojeda | Adjunta la identidad | Adjunta la identidad |

**Caso de prueba: Registrarse en el sistema AMADeUs con una identidad de OpenID.**

**Objetivo:** Encontrar Errores en el proceso que lleva a un usuario de OpenID a registrarse exitosamente en AMADeUs con sus datos de OpenID.
**Precondiciones:** El usuario de OpenID se halla autentificado por su proveedor pero no por AMADeUs, ya que no existe en él.
**Poscondiciones:** El usuario de OpenID se ha registrado en AMADeUs.
**Observaciones:** Se observa en los tres casos que el usuario logra registrarse usando los datos de su perfil de OpenID que prefiera.

**Tabla 9.3 Caso de prueba Registrarse con OpenID**

| Datos de entrada | Resultados esperados | Resultados obtenidos |
|---|---|---|
| http://dmoescobar.myid.net/ + dojeda | Se registra con la identidad. | Se registra con la identidad. |
| http://jonspenlow.myopenid.com/ + dojeda | Se registra con la identidad. | Se registra con la identidad. |
| http://danielojeda.openid.es/ + dojeda | Se registra con la identidad. | Se registra con la identidad. |

**Caso de prueba: Eliminar una identidad de OpenID del sistema AMADeUs.**

**Objetivo:** Encontrar errores en el proceso de eliminar una identidad de OpenID de su cuenta de AMADeUs.
**Precondiciones:** El usuario se ha logeado en el sistema y posee al menos una identidad de OpenID adjuntada a su cuenta de AMADeUs.
**Poscondiciones:** EL usuario logra aliviar con éxito una identidad de OpenID de su cuenta en el sistema.
**Datos:** La identidad de OpenID que el usuario desee eliminar.
**Observaciones:** En los tres casos el usuario ha logrado eliminar con éxito la identidad que el ha seleccionado.

**Tabla 9.4 Caso de prueba Eliminar una identidad**

| Datos de entrada | Resultados esperados | Resultados obtenidos |
|---|---|---|
| dojeda * http://dmoescobar.myid.net/ | Se elimina la identidad. | Se elimina la identidad. |
| dojeda * http://jonspenlow.myopenid.com/ | Se elimina la identidad. | Se elimina la identidad. |
| dojeda * http://danielojeda.openid.es/ | Se elimina la identidad. | Se elimina la identidad. |

Es importante aclarar que si bien se observan resultados positivos y satisfactorios en las pruebas realizadas al software, estas **no indican que no existan más defectos**, sino más bien, que no se ha logrado probar la existencia de más defectos en el sistema. Además estos resultados se han obtenido después de hallar fallas en el software y de realizar las correcciones necesarias.

# 10. Conclusiones

AMADeUs surge como un LMS de segunda generación desarrollado como un software de código abierto por la Universidad Federal de Pernambuco. AMADeUs comprende la integración de diversos recursos multimedia y esta basado en el concepto de aprendizaje flexible o mixto. Profesores y alumnos de la Pontificia Universidad Católica de Chile han comenzado, desde mediados de 2009 a tomar parte en el proyecto AMADeUs, desarrollando diversos módulos en el y llevando a cabo la localización de la herramienta a la cultura chilena, así como un estudio de usabilidad sobre ella.

AMADeUs en su desarrollo y arquitectura comprende la utilización de tecnologías y herramientas que, además de ser en su mayoría no privativas, son actualmente ampliamente utilizadas, tales como Struts e Hibernate. El trabajo en el presente proyecto ha implicado la investigación de estas dos herramientas previamente mencionadas, así como otras (IDE Eclipse, J2EE), ambas herramientas cuentan con documentación en sus respectivos sitios oficiales, así como con una comunidad de usuarios constantemente en crecimiento.

El empleo del framework Struts ha sido al mismo tiempo una ventaja y un desafío. La herramienta ha ayudado a comprender la manera en la que fue construido AMADeUs y como se comunican los objetos de manera interna. También se ha debido investigar el modelo MVC y comprender como este es implementado por Struts en las aplicaciones de este tipo, si bien puede resultar complejo en un principio, al comprender el framework es posible percatarse de las ventajas que éste proporciona, poniendo un orden al código fuente y separando el sistema en capas para una mejor comprensión y una más fácil extensión.

La utilización de Hibernate fue similar a la descrita en el párrafo anterior, demandó investigación de ella para poder comprenderla, ya que era una nueva manera de manejar el acceso a los datos en una aplicación, abstrayéndose del motor de bases de datos que se pretendía usar. Una vez que se entiende y se emplea Hibernate resultan evidentes las ventajas que se obtienen con él, permitiendo un acceso más ordenado y uniforme a la base de datos y haciendo posible también la reutilización de código, una de las características de la orientación a objetos y que por lo tanto debería cumplirse en todos los desarrollos de software de este tipo. La utilización conjunta de los dos frameworks anteriormente descritos ha permitido efectivamente desarrollar software más fácilmente extensible y mantener el código fuente de una manera más ordenada y clara para el desarrollador.

OpenID es un sistema de autentificación digital de código abierto y descentralizado, que brinda a sus usuarios la posibilidad de crear una identidad en la Web, permitiéndoles usar dicha identidad en todo sistema o página que soporte la identificación con OpenID. A fin de comprender la amanera en cómo OpenID funciona y cómo debe ser implementado ha sido muy útil la información ofrecida por la Fundación OpenID en su página Web oficial en la que además de documentación más sencilla para los potenciales usuarios, es posible encontrar información de interés especial para los desarrolladores.

De especial utilidad han sido las especificaciones de OpenID las cuales describen el protocolo de autentificación y cuáles son las distintas partes que participan en el proceso de autentificar a los usuarios así como su rol en él y como llevan a cabo su colaboración. También ha sido parte primordial del presente trabajo la utilización de la librería OpenIDforjava, La librería se ha complementado de manera satisfactoria con las clases ya existentes en AMADeUs, así como con las clases que se han debido incluir para el funcionamiento pleno del módulo desarrollado.

AMADeUs tiene la ventaja, como se ha mencionado, de que sus tecnologías lo hacen fácilmente extensible, pero por otra parte plantea un desafío al desarrollador ya que hay una falta de documentación con respecto al proceso de desarrollo previo de la aplicación. Si bien hubo siempre disponibilidad de consultar con el equipo que lo desarrolló e incluso se pudo tener instancias de conversar con uno de los profesores encargados del proyecto, fue necesario una mayor investigación que si se hubiera contado desde un principio, por ejemplo, con diagramas de casos de uso (de toda el software), diagramas de clases y diagramas de secuencia o colaboración.

También influyó en el desarrollo del presente proyecto el hecho de que los requerimientos para la integración con OpenID no estuvieron claros desde el principio, por lo que se debió investigar la manera en que otros sistemas se integran con OpenID. Sin embargo esta situación ha brindado la oportunidad de contar con una mayor libertad al momento de diseñar y desarrollar la aplicación permitiendo además la exploración de distintas maneras de documentar el trabajo. Ha sido posible poner en práctica de una manera más real, y por lo mismo, más compleja las distintas etapas del desarrollo del software que se han estudiado en las asignaturas de la carrera y como cada una de estas etapas se suceden una y otra vez colaborando en el proceso de construcción.

Ha sido especialmente esclarecedora la utilización, en este caso, del lenguaje unificado de modelado (UML) para el modelado del software, ya que se ha robustecido la comprensión por parte del alumno de los propósitos y ventajas de cada uno de los diagramas, en particular del diagrama de actividades, clave al momento de comprender el flujo del producto a desarrollar y como se comunicaba la aplicación (AMADeUs) con los proveedores de OpenID y como implementaba el protocolo y también la comprensión de los diagramas de secuencia encargados de modelar la interacción entre los distintos objetos de la aplicación.

Se puede concluir que la experiencia en el presente trabajo ha sido enriquecedora, se conjugó la participación en un proyecto que involucra muchas más personas y a la vez estar relativamente alejado del cuerpo principal y más antiguo de desarrollo, lo que exigió mayor creatividad e investigación. Se ha tenido la oportunidad de conocer y trabajar con herramientas que de otro modo quizás no se hubieran comprendido o podido emplear. Si bien el proyecto no es considerado un trabajo de gran envergadura, si ha incluido todas las etapas del desarrollo de un software, así como el uso de varias herramientas y la investigación de nuevas tecnologías (como OpenID). Además, cada una de las instancias de evaluación ha contribuido a una mayor comprensión de la manera de construir un software y de cómo documentar apropiadamente dicha construcción.

A modo de trabajo futuro, es posible señalar la implementación de un proveedor de OpenID, pudiendo así AMADeUs posicionarse, no sólo como un sistema que soporte OpenID, sino también como un sistema que provea identidades, las almacene y permita la autentificación de usuarios, permitiendo así dotar a los usuarios de AMADeUs de una mayor capacidad de interacción en la Web.

# 11. Referencias


[Fowler, 99] Martin Fowler con Kendall Scout, *UML Gota a Gota 2º edición*, Pearson Addison Weasley; México, 1999.

[García, 09] Francisco José García Peñalvo, *Estado actual de los sistemas e-learning*, Revisado el 20 Septiembre, 2009.
http://www.usal.es/~teoriaeducacion/rev_numero_06_2/n6_02_art_garcia_penalvo.htm

[Hibernate, 10] Sitio del Framework Hibernate, Revisado el 28 de Marzo, 2010.
http://www.hibernate.org/

[Landeta, 09] Ana Landeta, *Libro de Buenas prácticas en e-Learning*, Revisado el 28 Agosto, 2009.
http://www.buenaspracticas-elearning.com/

[Lopes *et al*., 08] Luanna Lopes Lobato, Bruno de Sousa Monteiro, Alex Sandro Gomes, *Amadeus-MM: LMS com integração de serviços multimídia*; Centro de Informática, Universidad Federal de Pernambuco, Recife, Brasil, 2008.

[Lopes, 08] Luanda Lopes Lobato, *AMADEUS Documento de Arquitectura do Software*, Ciências Cognitivas e Tecnologia Educacional, Recife, Brasil, 2008.

[OpenID, 09]Sitio oficial de la OpenID Fundation, Revisado el 15 de octubre 2009.
http://openid.net/

[OpenId4java, 09] Sitio donde esta alojado el código de la librería OpenId4java, Revisado 20 de septiembre 2009.
http://code.google.com/p/openid4java/

[Ponce, 07] Marcela Ponce Martínez, *Memoria de Título*, Escuela de Ingeniería Informática, 2007; Valparaíso, Chile.

[Portal, 10] Sitio Web oficial del Portal de Software Libre Brasileño, Revisado 20 de junio de 2010.
http://softwarelivre.org/

[Sommerville, 05] Ian Sommerville, *Ingeniería del Software, 7º edición*, Pearson Addison Wesley; Madrid, España, 2005.

[Struts, 08] Sitio Web oficial de Apache Struts, Revisado el 29 de noviembre 2008.
http://struts.apache.org/

# Anexo A ï  Manual de usuario

El manual de usuario se redactó originalmente en inglés, y luego se tradujo al español. El presente manual de usuario consta de cinco ítems o partes e indica al usuario de Amadeus como llegar a tener el sistema ejecutándose y a realizar la configuración necesaria de la base de datos y del proxy (en el caso de que sea necesario).

1.- En el primer ítem hay una lista de los archivos que es necesario tener a fin de de instalar y ejecutar Amadeus.

2.- En el segundo ítem se encuentra una explicación de cómo instalar Apache Tomcat.

3.- El tercer ítem consiste en una breve explicación de cómo instalar PostgreSQL.

4.- El cuarto ítem tiene la explicación de cómo importar el archivo .war de Amadeus.

5.- Finalmente en el quinto y último ítem se explica cómo configurar el archivo XML a fin de asignar los valores necesarios a los campos de la base de datos y el Proxy.

## 1 Elementos necesarios:

Antes que nada, es necesario contar con los siguientes elementos:

**Figura A.1 Apache Tomcat.**


**Apache-Tomcat-6.0.18.exe:** El contenedor de servlets requerido.

**Figura A.2 AMADeUs**


**AmadeusLMS404.war:** Este archivo contiene el código fuente del sistema.

**Figura A.3 Postgresql.**


**Postgresql-8.3.3-1.zip**: El archivo de instalación de el sistema de gestión de bases de datos requerido.

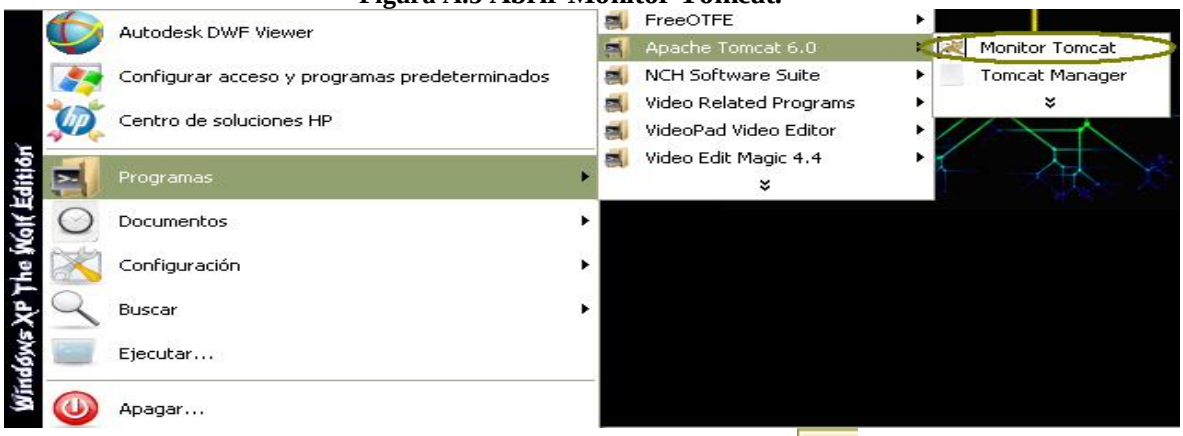
**Figura A.4 Amadeus SQL.**


**Amadeus_web_v0.5_openid.sql:** El archivo .sql necesario para crear la base de datos.

## 2 Instalación de Apache Tomcat:

Al hacer cllick en el archivo apache_tomcat.6.018.exe se abrirá automáticamente el asistente para instalación. Una vez que Apache Tomcat esté instalado se debe ir a Inicio → Programas → Apache Tomcat 6.0 → Monitor Tomcat, tal como se muestra en la siguiente figura.

**Figura A.5 Abrir Monitor Tomcat.**




Entonces debería aparecer un pequeño icono como este  en la esquina inferior derecha del escritorio, al hacer clic derecho en el y seleccionar ó*iniciar servicio*ô, el icono debe cambiar y mostrarse de ésta manera:  , lo cual da a entender que el servicio se ha iniciado.

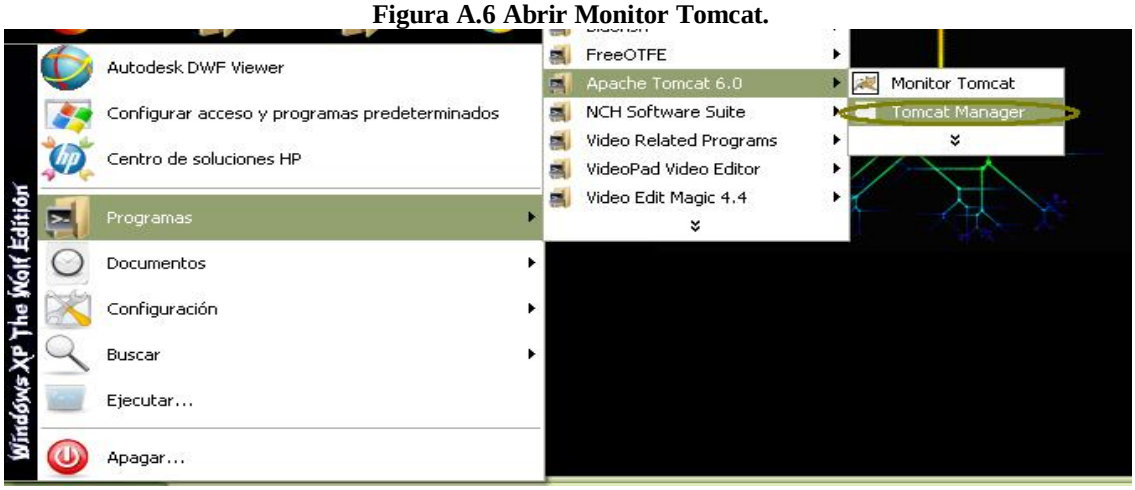
### 3. Instalación de PostgreSQL:

Se debe descomprimir el archivo postgresql-8.3.3-1.zip y seguir las instrucciones proporcionadas por el instalador para instalarlo, durante el proceso de instalación se debe mantener el puerto por defecto ó5432ô. Luego, se debe crear una nueva base de datos y retener el nombre de la base de datos, el del propietario de la base de datos y la contraseña para su posterior uso en el ítem cinco.

### 4. Importación del arhivo .war:

Una vez que Apache Tomcat ha sido instalado, se debe ir a Inicio → Programas → Apache Tomcat 6.0 → Tomcat Manager, tal como se muestra en la siguiente figura:

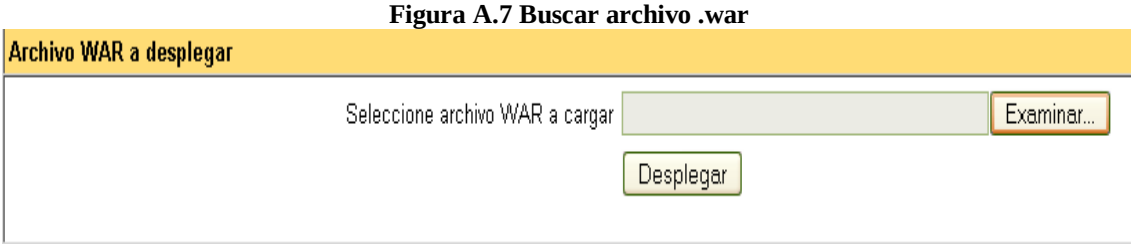
**Figura A.6 Abrir Monitor Tomcat.**




El administrador (manager) de Tomcat se abrirá en el navegador predeterminado del equipo, una vez abierto, se deben seguir los siguientes pasos para importar el archivo .war:

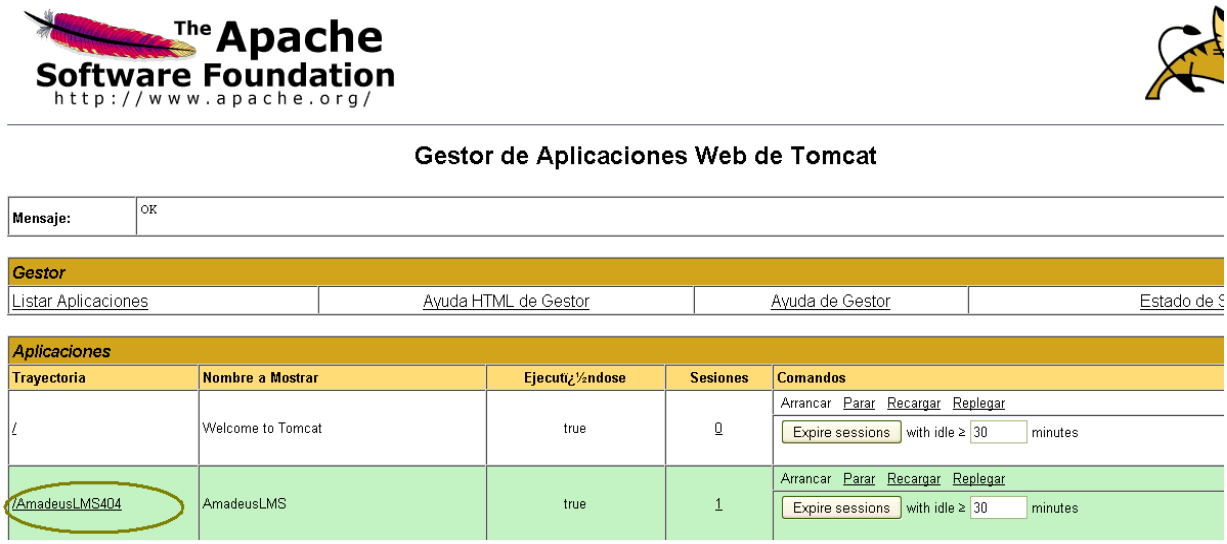
Se debe ir a la sección que se muestra abajo y hacer clic en ó examinarô, a fin de localizar y seleccionar el archivo .war, luego de lo cual se debe hacer clic derecho en la opción ódesplegarô.

**Figura A.7 Buscar archivo .war**




Entonces, la aplicación Amadeus404 aparecerá en la lista de aplicaciones disponibles, se verá de la siguiente manera:

**Figura A.8 Aplicación disponible para ejecutar.**




Después de seleccionar ó recargarô, Amadeus debería ejecutarse inmediatamente (siempre y cuando se haya creado previamente la base de datos).

## 5.- Configuración de la base de datos y del Proxy

Para configurar los parámetros de la base de datos e indicar el uso del proxy (en el caso en que sea necesario), se deben seguir las siguientes instrucciones:

En cualquiera de los casos previamente mencionados, es necesario es necesario abrir el archivo hibernate.cfg.xml, ubicado en un directorio similar al siguiente: C:\Archivos de Programas\Apache Software Fundation\Tomcat6.0\webapps\AmadeusLMS404\WEB-INF\classes\hibernate.cfg.xml.

A continuación, una imagen de dicho archivo abierto:

**Figura A.9 Archivo XML de configuración.**

```
<property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>
<property name="hibernate.connection.driver_class">org.postgresql.Driver</property>
<property name="hibernate.connection.url">jdbc:postgresql://localhost:5432/amadeusII</property>
<property name="hibernate.connection.username">postgres</property>
<property name="hibernate.connection.password">admin</property>
<property name="hibernate.current_session_context_class">thread</property>
<property name="hibernate.proxy">proxya4.ucv.cl</property>
<property name="hibernate.proxyPort">8080</property>
<property name="hibernate.Useproxy">no</property>
<property name="hibernate.returnurl">http://localhost:8080/AmadeusLMS404/</property>
<property name="hibernate.dbase">org.postgresql.Driver</property>
```

Correspondiendo a:

**<property name="hibernate.connection.url">:** Es el nombre de la base de datos.
**<property name="hibernate.connection.username">:** Es el nombre del usuario que es propietario de la base de datos.
**<property name="hibernate.connection.password">:** Corresponde a la contraseña de la base de datos.
**<property name="hibernate.current_session_context_class">:** Es el Proxy, el cuál sólo es necesario configurarlo si la conexión de internet utiliza uno.
**<property name="hibernate.proxy">:** Es el número del puerto del Proxy.
**<property name="hibernate.proxyPort">:** Es ñNOô si no se desea que Amadeus considere un valor para el proxy.
**<property name="hibernate.Useproxy">:** Es la URL a la cual el proveedor de OpenID envía la respuesta.
**<property name="hibernate.returnurl">:** Es otro parámetro necesario para la conexión ala base de datos.

# Anexo B ï Requirements specification

This requirements specification has been written according to the AMADeUs format and only in english for the use of the AMADeUs Project.

**USECASE**: **[UCoid01] ï ENTER THE SYSTEM WITH OPENID IDENTITY.**

**Function**: Let the user log into the system with an OpenID Identity.

**Update history**

| *Date* | *Description* | *Name* |
|---|---|---|
| Nov 6th 2009 | Creation of the Use Case | D. Ojeda |

**Actors**: USER, OPENID PROVIDER.

**Business Priority**: Essential ■ Important Desirable
**Technical Priority**: Essential Important ■ Desirable

**Pre-conditions**:
The user is on the welcome page of the system.

**Pos-conditions**:
The user has entered the system using his OpenID identity.

**Main flow of events**

| *Steps* | *Action* |
|---|---|
| 1 | User select the option login with OpenID |
| 2 | System ask the user to enter his OpenID username |
| 3 | System verifies the identity with the corresponding OpenID provider. |
| 4 | System checks that the OpenID identity is attached to an user |

**Alternative flow**

| *Steps* | *Action* |
|---|---|
| 4.1 | If the OpenID identity doesnôt exist in the system, the system ask the user to login or register in AMADeUs. |
| 4.2 | If the OpenID identity exists in the system, it let the user enter into the system. |

**Visual interface**


**Matrix of Impact**

| *UseCase* | *Description of impact* | *Input* | *Output* |
|---|---|---|---|
| UCoid02 | The user must autentícate with OpenID before attachment | | X |
| UCoid03 | The user must authenticate with OpenID before register with OpenID | | X |

**USECASE: [UCoid02] ï ATTACH AN OPENID IDENTITY.**

**Function**: Let the user attach an OpenID Identity to his AMADeUs account .

**Update history**

| *Date* | *Description* | *Name* |
|---|---|---|
| Nov 7th 2009 | Creation of the Use Case | D. Ojeda |

**Actors**: USER, OPENID PROVIDER.

**Business Priority**: Essential ■ Important Desirable
**Technical Priority**: Essential Important ■ Desirable

**Pre-conditions**:
The user owns an OpenID identity but its not yet associated to his AMADeUs account.
**Pos-conditions**:The user has attached a new OpenID identity to his AMADeUs account.

**Main flow of events**

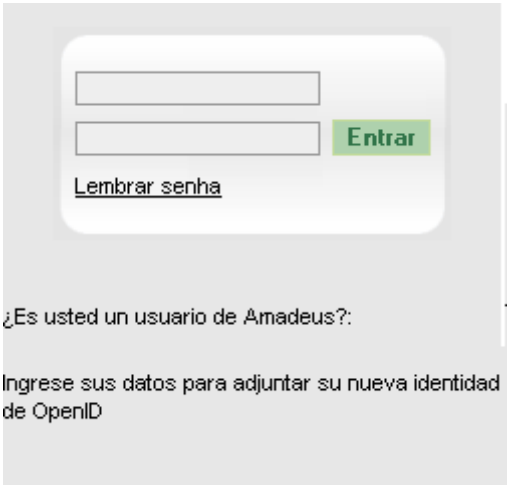
| *Steps* | *Action* |
|---|---|
| 1 | The user select the attach OpenID identity option. |
| 2 | The system asks the user to enter his AMADeUs login and password. |
| 3 | The user enter successfully his AMADeUs login and password. |

| | |
|---|---|
| 4 | The system informs the user that his new OpenID identity has been attached |

**Alternative flow**

| *Steps* | *Action* |
|---|---|
| | Not considered. |

**Visual interface**




**Matrix of Impact**

| *UseCase* | *Description of impact* | *Input* | *Output* |
|---|---|---|---|
| UCoid01 | The user must authenticate with OpenID before attach a new identity | X | |
| UCoid03 | The user must have at least one OpenID identity before manage his OpenID identities | | X |

**USECASE: [UCoid03] ï DELETE AN OPENID IDENTITY OF THE SYSTEM.**

**Function**: Let the user delete an OpenID Identity previously attached to his AMADeUs account.

**Update history**

| *Date* | *Description* | *Name* |
|---|---|---|
| Nov 7th 2009 | Creation of the Use Case | D. Ojeda |

**Actors**: USER.
**Business Priority**: Essential ■ Important Desirable
**Technical Priority**: Essential Important ■ Desirable

**Pre-conditions**:
The user has entered the system.

**Pos-conditions**:
The user has deleted at least one of his OpenID identities from the system.

**Main flow of events**

| *Steps* | *Action* |
|---|---|
| 1 | The user select the manage OpenID identities option. |
| 2 | The system shows the user a list containing his OpenID identities. |
| 3 | The user select the elimination of one of them. |
| 4 | The system informs the user that the action has been executed. |

**Alternative flow**

| *Steps* | *Action* |
|---|---|
| | Not considered. |

**Visual interface**


**Matrix of Impact**

| *UseCase* | *Description of impact* | *Input* | *Output* |
|---|---|---|---|
| UCoid01 | The user must authenticate with OpenID before manage his OpenIDs | X | |
| UCoid02 | The user must have an OpenID identity before manage his OpenIDs | X | |
| UCoid03 | The user must have an OpenID identity before manage his OpenIDs | X | |

**USECASE: [UCoid04] ï REGISTER WITH OPENID.**

**Function**: Let the user register into the system with an OpenID Identity.

**Update history**

| *Date* | *Description* | *Name* |
|---|---|---|
| Nov 7th 2009 | Creation of the Use Case | D. Ojeda |

**Actors**: USER, OPENID PROVIDER.

**Business Priority**: Essential ■ Important Desirable
**Technical Priority**: Essential Important ■ Desirable

**Pre-conditions**:
The user owns an OpenID identity but its not yet associated to his AMADeUs account.

**Pos-conditions**:
The user has created an AMADeUs account using his OpenID identity.

**Main flow of events**

| *Steps* | *Action* |
|---|---|
| 1 | The user selects the register using his OpenID identity. |
| 2 | The system asks the user to fill the fields corresponding to a new user. |
| 3 | If tutor is not on-line the system displays the last three visit dates, in descending order. |
| 4 | The system registers the new user and attaches the OpenID identity. |

**Visual interface**


**Matrix of Impact**

| *UseCase* | *Description of impact* | *Input* | *Output* |
|---|---|---|---|
| UCoid02 | The user must authenticate with OpenID before register with OpenID | X | |
| UCoid03 | The user must have at least one OpoenID identity in AMADeUs to manage his OpenID identities | | X |