Proyecto Final de Carrera

Aplicación web para el diseño y creación de una asignatura dentro del entorno de enseñanza AdVisor

Ingeniería Técnica de Telecomunicaciones, especialidad Telemática

Daniel Álvarez López

Semestre 2011-1

Universitat Oberta de Catalunya Consultor: David García Solórzano

ABSTRACT

La evolución de la tecnología hace que evolucionen también los métodos de aprendizaje, es por ello que se populariza el método e-Learning, el cual es un modelo de formación a distancia que utiliza Internet como herramienta de aprendizaje. Este modelo permite al alumno realizar el curso desde cualquier parte del mundo y a cualquier hora.

Con un ordenador y una conexión a Internet, el alumno puede realizar las actividades interactivas planteadas, acceder a toda la información necesaria para adquirir el conocimiento, recibir ayuda del profesor; comunicarse con su tutor y sus compañeros, evaluar su progreso, etc.

La idea de este proyecto es la de implementar este concepto creando una herramienta de software específicamente diseñada para adecuarse al entorno de AdVisor, ayudando de esta manera a crear desde cero y con la mayor simplicidad y brevedad posible una asignatura. También se va a poder modificar las asignaturas ya creadas, dando así un plus al valor de ésta aplicación.

Todo esto se consigue mejorando la creación de los diferentes elementos de la asignatura, así como sus definiciones e interrelaciones entre ellos, reduciendo de manera drástica la complejidad y el duro esfuerzo que suponía antes para un profesor.

Palabras claves

- AdVisor, Diseño y creación de una asignatura, Knowledge Visualization, EEES (Espacio Europeo de Enseñanza Superior), Aplicación e-Learning

ABSTRACT

The evolution of technology makes also the evolution of learning methods. This has been the reason of the popularization of the e-Learning method, which is a distance learning model based on Internet applications as a learning tool. This model allows the student to study the subjects from anywhere in the world, at any time.

With a computer and Internet connection, the student can perform interactive activities proposed. He also can access to all the information necessary to acquire the knowledge required, receive help from the teacher, communicate with the tutor and his mates and assess his own progress.

The idea of this project is to implement this concept by creating a software tool specifically designed to adapt to the environment of AdVisor. It will help the user to create a subject from the easiest and soonest way possible. This will be also able to modify the subjects already created, giving an extra value to this application.

We will get the success of this application improving the creation of several elements of the subject, as well as their definitions and interrelations among them. It means that the complexity found before for the teacher will disappear.

Keyword

- AdVisor, Design and creation of a subject, Knowledge Visualization, EEES (Espacio Europeo de Enseñanza Superior), Aplicación e-Learning

Resumen

La digitalización de la información y a la aparición de Internet ha supuesto una transformación de la sociedad. Así, acciones cotidianas como comprar o comunicarse se han trasladado al mundo online. En este sentido, la enseñanza-aprendizaje no ha sido una excepción, ya que se ha creado la necesidad de formarse a lo largo de la vida (lifelong learning), p.ej. cursos de reciclaje. Así, la educación se ha trasladado a Internet, originando un modelo educativo denominado e-Learning.

El modelo e-Learning se caracteriza por el uso de los ordenadores e Internet para enseñar a los estudiantes, lo que permite adaptarse a las necesidades de los alumnos en términos de tiempo y espacio. Asimismo, debido a problemas de aislamiento que suelen tener los estudiantes online, actualmente se proponen plataformas que ayuden a la tutorización y al seguimiento de éstos. En este sentido, AdVisor es una plataforma de la UOC que ayuda a organizar la información de las asignaturas, teniendo como una de sus características la planificación de la actividad del estudiante.

La estructura de AdVisor está centrada en las actividades, lo cual hace que todos los elementos restantes del curso estén vinculados a éstas, proporcionando información de valor añadido al estudiante para poder mejorar su rendimiento y guiarlo hacia un mejor seguimiento y planteamiento de la asignatura. Cada asignatura está compuesta por 4 espacios de información relacionadas entre sí a través de las actividades del curso (ver Ilustración 1). En el esquema se puede observar que las Actividades serán el centro de la asignatura, cada espacio estará relacionado con una o más actividades. El espacio de información llamado Planificación contiene un número de Sesiones donde cada Sesión tiene aproximadamente una hora y media de carga de trabajo. Por otro lado, la Evaluación es un espacio de información que contiene todas las Prácticas y Exámenes de la asignatura, es decir, todos los elementos puntuables de la materia. El espacio de información Temario está compuesto por el índice de la asignatura con sus apartados, sub-apartados, etc. Por último, el espacio de información llamado Competencias y Objetivos es una estructuración sobre un conjunto de conocimientos que el estudiante debe adquirir al final de ese proceso. Así que cada Actividad está relacionada con estos conocimientos y gracias a ello el estudiante puede saber los objetivos del curso y la competencia de la titulación que está desarrollando al hacer dicha Actividad.

La propuesta de este proyecto es crear una herramienta de software que permita al profesor diseñar, crear y modificar una asignatura dentro de la plataforma AdVisor. La aplicación debe servir para construir una asignatura de la UOC desde cero y con la mayor simplicidad y brevedad posible. Por eso, la herramienta planteada, debe mejorar la creación de los diferentes elementos de la asignatura, así como la definición de las interrelaciones existentes entre ellos, reduciendo de este modo la realización de todas estas tareas de manera manual (p.ej. la creación del fichero XML de definición de elementos y relaciones).

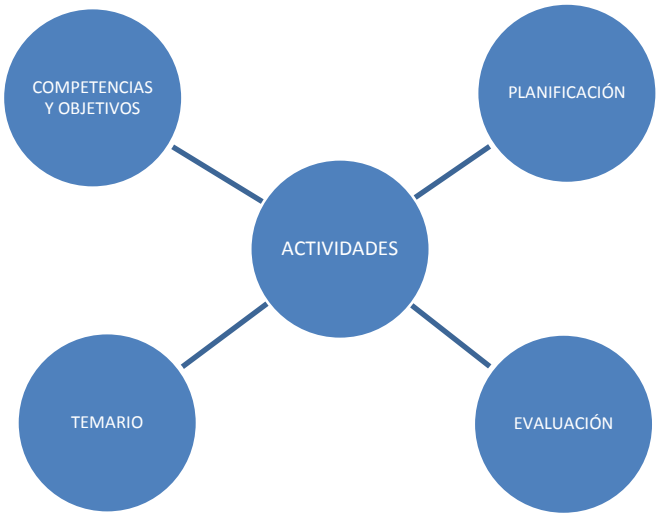

Ilustración 1

Con la ayuda de esta aplicación el profesor sólo tendrá que cargar todos los elementos de la asignatura, estructurarla visualmente y guardarla. Posteriormente el programa comprimirá los componentes de la asignatura y creará un archivo XML basado en la estructura de la materia definida por el docente.

Abstract

Digitization of information and the emergence of the Internet have led to a transformation of society. As well, daily actions how to buy or communicate have moved to the online world. In this sense, the teaching-learning has not been an exception, since it has created a need for form along the life (lifelong learning), e.g. refresher courses. As well, education has been transferred to the Internet, causing a educational model called the e-learning.

The teaching-learning model based on e-Learning is characterized by adapting to the needs of students in terms of time and space. Also, platforms are proposed to assist the mentoring and monitoring of this method due to problems of isolation that online students tend to have in their diary learning. In this regard, Advisor is a platform for the UOC that helps the subjects to organize the information, having as one of its planning features student activity.

The structure of Advisor is focused on the activities, which makes all the remaining elements of the course are linked to them, providing information of value to the students in order to improve their performance and guide them to better track and approach there. Each course consists of 4 areas of information related to each other through the course activities (see **¡Error! No se encuentra el origen de la referencia.**). In the diagram we see that the activities will be the focus of the course, each space will be related to one or more activities. The call information space contains a number of planning sessions where each session is approximately one and half hour workload. On the other hand, the assessment is an information space that contains all the practices and examinations of the subject, that is, all scoring elements of matter. Agenda information space consists of the index subject with its sections, subsections, etc.. Finally, the information space called Skills and Goals is a structure on a set of skills that students should acquire by the end of that process. So every activity is linked to this knowledge and through that the student can learn the course objectives and competence of the qualification being developed to make such activity.

The proposed project is to create a software tool that allows the professor design, create, and modify a subject within the platform AdVisor. The application must be used to build a course at the UOC from scratch and with the greatest possible simplicity and brevity. Therefore, the tool presented, should enhance the creation of the different elements of the subject, as well as defining the interrelationships between them, thereby reducing the performance of all these tasks manually (for example creating XML file to define elements and relationships).

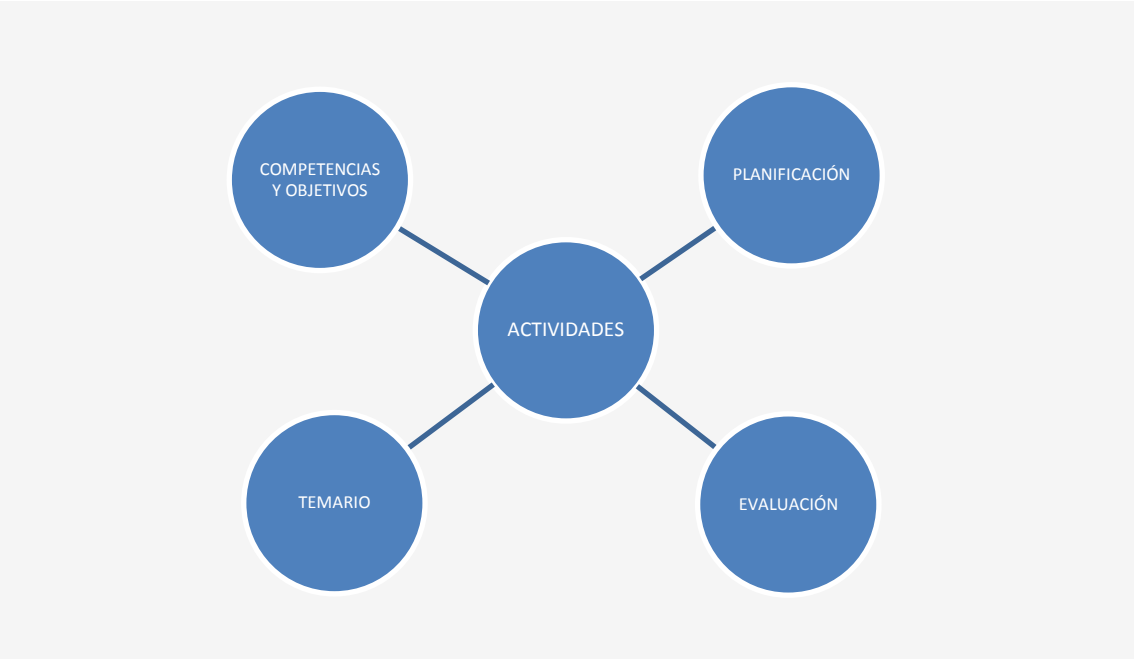

Ilustración 2

With the help of this application, the teacher will only have to load all the elements of the subject, visual structure and save it. Then the program will compress the components of the subject and create an XML file based on the structure of matter defined by the teacher.

Agradecimientos

Mi primer agradecimiento debe ser para mi tutor de TFC David García Solórzano que con su constante apoyo, ánimo y guía fue una persona clave para que el desarrollo de este trabajo se llevara a cabo. Su participación fue esencial tanto para mantener una línea razonable y sistemática de trabajo, enriquecida con útiles sugerencias y propuestas.

Deseo agradecer también a mis amigos por los apoyos y ánimos que me han estado dando continuamente durante todo este tiempo y por la comprensión que han tenido cuando no he podido quedar con ellos.

Por último, un muy especial agradecimiento a mi madre y a mis hermanos que ellos han tenido que sufrir mi falta de disponibilidad encerrado en la habitación por las horas dedicadas a este trabajo.

Índice de Contenidos

Índice de Figuras

Índice de Tablas

Índice de Ilustraciones

Definiciones

AIC	(Aviation Industry Computed Based-Training Comitee) Es una asociación internacional de capacitación de profesionales basada en tecnología. El AICC desarrolla especificaciones para la industria de la aviación para el desarrollo, la entrega, y la evaluación del CBT (Computed Based-Training) y de tecnologías de capacitación relacionadas.
API	Interfaz de programación de aplicaciones o API (Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
IMS	IMS Global Learning Consortium es un Consorcio formado por miembros provenientes de organizaciones educativas, así como de empresas públicas y privadas, cuya misión es desarrollar y promover especificaciones abiertas para facilitar las actividades del aprendizaje online.
JSP	JavaServer Pages es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.
LD	La especificación *Learning Design* parte del lenguaje *Educational Modelling Language* (Lenguaje de Modelado Educativo) desarrollado originalmente en la *Open University of the Netherlands* a partir de la identificación de los principios fundamentales de distintas aproximaciones pedagógicas y de la búsqueda de un equilibrio entre generalidad y expresividad pedagógica (Koper y Manderveld 2004).
LMS	Un sistema de gestión de aprendizaje (Learning Management System) es un software instalado en un servidor web que se emplea para administrar, distribuir y controlar las actividades de formación no presencial de una institución u organización.
MVC	Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos.
MySQL	Es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.
SCORM	SCORM (Sharable Content Object Reference Model) es un conjunto de estándares y especificaciones que permite crear objetos pedagógicos estructurados.
SQL	El lenguaje de consulta estructurada o SQL (Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en estas.
TIC	Las tecnologías de la información y la comunicación (TIC) agrupan los elementos y las técnicas usadas en el tratamiento y la transmisión de las informaciones, principalmente de informática, internet y telecomunicaciones.
UML	Lenguaje Unificado de Modelado (LUM o UML, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.
XML	(eXtensible Markup Language) Es un metalenguaje extensible de etiquetas desarrollado por el Worl Wide Web Consortium (W3C).

Capítulo 1. Introducción

1.1 Objetivos principales

Este Trabajo Final de Carrera (TFC) tiene como principal finalidad crear una herramienta que, de manera sencilla, usable y con buen diseño, se pueda crear una asignatura dentro de la plataforma de aprendizaje de AdVisor. Asimismo, otros subobjetivos son establecidos vinculados al anterior. A continuación se puede ver los objetivos que cumple la herramienta desarrollada.

- En base a la funcionalidad de la aplicación:

  - Escalabilidad: Facilidad a la hora de ampliar y modificar la herramienta.
  - Consistencia: La aplicación funciona siguiendo siempre el mismo comportamiento y acorde a lo que el usuario espera de ésta.
  - Robustez: Ya que la aplicación garantiza, en todo momento, que el funcionamiento sea el correcto.

- En base a la experiencia del usuario:

  - Sencillez: Es un objetivo clave para que el usuario cree una asignatura en pocos pasos.
  - Intuitiva e independiente: El uso de la herramienta debe ser fácil de aprender. Es decir, la curva de aprendizaje debe ser mínima. Así mismo, conseguir que el usuario, sin tener conocimientos previos de la plataforma AdVisor, pueda utilizar la aplicación sin problemas.
  - Para todo tipo de usuarios: La aplicación debe es útil tanto a usuarios expertos como noveles en términos de diseño de asignaturas.
  - Navegable: El usuario debe de tener facilidad para poder desplazarse por todas las páginas que componen un sitio web. Para lograr este objetivo, la aplicación debe proporcionar un conjunto de recursos y estrategias de navegación diseñados para conseguir un resultado óptimo en la localización de la información y en la orientación para el usuario.

1.2 Motivación

La principal motivación por la que opté por este trabajo fue que éste me brindaba la posibilidad de contribuir en la mejora de herramientas para el aprendizaje online. En el caso del presente TFC, su aportación fue la implementación de una herramienta para el profesor que facilite el diseño y creación de asignaturas dentro del entorno AdVisor.

Este TFC ayuda a mejorar las técnicas aprendidas a lo largo de toda la titulación e incluso aprender nuevas, de eso trata un TFC, de ir más allá de los conocimientos

adquiridos y obligar a hacer un esfuerzo, para aprender y atravesar metas que hasta ahora se creen lejanas. Este proyecto sirve para poner a prueba todos los estudios conseguidos y de alguna manera evaluarlos, ayudar a ser consciente de los límites y te impulsa a saber organizar todas las habilidades y entendimientos para llevar a cabo un buen TFC.

Por otro lado, este TFC me permitía practicar y profundizar varios conocimientos adquiridos a lo largo de la titulación, así como adquirir nuevos.

Finalmente, me parecía interesante el resto de adentrarme en un proyecto en el que necesitaba obtener conocimientos nuevos y ser autodidáctico para poder alcanzar los objetivos establecidos.

1.3 Beneficios

Actualmente, sin una herramienta como la que se propone en este trabajo, crear una asignatura para la plataforma AdVisor es una tarea ardua. Cada profesor tiene que diseñar las asignaturas en papel para luego crear manualmente un XML que sirve para introducir la información de la asignatura en la base de datos de AdVisor, esto supone definir todos los elementos, espacios de información e interrelaciones. Evidentemente, generar manualmente dicho XML es una tarea ardua y no exenta de errores. Con la herramienta presentada en este TFC, se facilita esta tarea a los docentes.

Así pues, el gran beneficio y el objetivo principal de este proyecto es la de hacer más fácil la tarea de diseño y creación de la asignatura en AdVisor por parte del profesor.

1.4 Productos obtenidos

Los productos obtenidos por la elaboración de este TFC son:

- La presente memoria técnica que recoge todo el trabajo de análisis, diseño e implementación de la aplicación.
- El código fuente de la aplicación.
- Una presentación que recapitula y enseña los aspectos más relevantes de este TFC.

Capítulo 2. E-Learning

2.1 Metodología e-Learning

El aprendizaje a distancia online, también conocido como e-Learning, está cada vez más presente dentro de las universidades, gracias a las posibilidades que ofrecen las TIC, p.ej. eliminación de las barreras físicas y temporales. Esto permite que muchas personas puedan estudiar cuando más les convenga.

El concepto de e-Learning viene dado por la forma de impartir educación mediante recursos informáticos de una manera íntegra y con una metodología de desarrollo del proceso de enseñanza, la cual tiene como medio de transmisión Internet e Intranet.

Existen principalmente dos tipos de e-Learning según la sincronía de la interacción:

- Síncrono. Es una modalidad en la que el profesor y alumno interactúan al mismo tiempo aunque se encuentren en espacios físicos diferentes, como por ejemplo una clase en videoconferencia
- Asíncronos. En los que el profesor y el alumno no tienen por qué interactuar al mismo tiempo, como por ejemplo un curso de ofimática.

Éste nuevo modelo de aprendizaje ha experimentado un aumento considerable en su uso y todo por la gran expansión de internet junto con nuevas y mejores herramientas que ayudan a facilitar el aprendizaje. Hoy en día, la educación ya no se limita en un lugar concreto, sino que se puede aprender desde dónde sea y cuando sea, ya que no es tan restrictivo como la educación presencial.

En el monográfico *Uso de contenidos digitales: tecnologías de la información, sociedad del conocimiento y universidad* (Álvarez)**,** trata de como se establece una nueva configuración de las estructuras organizativas debido a la introducción de nuevas tecnologías en la educación. La perspectiva que se plantea es la de intentar una mejor eficiencia y eficacia para la actividad educativa ya que se busca la optimización del aprendizaje.

Es muy importante gestionar bien el conocimiento ya que genera valor para la organización. Una persona que adquiera una capacidad de analizar, sintetizar y criticar hace que salga mejor preparada para la educación superior y la vida laboral, así que la gestión del conocimiento es importante.

En global, los sistemas de e-Learning intentan ayudar a los estudiantes a impulsar su conocimiento proporcionando un contenido de aprendizaje estructurado y unas instalaciones comunicativas para todos los ámbitos.

Todos estos sistemas proporcionan información valiosa de cada estudiante que junto con el análisis del progreso del alumno mediante pruebas y exámenes se puede gestionar una planificación del curso con una mayor eficiencia.

Estas evoluciones tanto en tecnología como en metodología han hecho que se cree un nuevo conocimiento útil y que éste se pueda distribuir eficientemente. Es el

concepto de e-Learning colaborativo. De ésta manera, éste nuevo conocimiento se puede fraccionar en cuatro fases: Comunicación, Compartir conocimientos, Distribuir conocimientos y Crear conocimientos.

2.1.1 Ventajas e inconvenientes

El e-Learning presenta una serie de ventajas, entre las cuales destacan (Ramírez, 2007):

- Minimiza los costes de ejecución.
- Ayudas estatales.
- Elimina las barreras de espacio y tiempo.
- Posibilidad de anonimato.
- Mejora la rápida respuesta frente a cambios.
- Puede crear una comunidad de aprendizaje.
- El ritmo del trabajo se puede individualizar más fácilmente que en presencial.
- Los horarios educativos son más flexibles, 24horas al día, 7 días a la semana.
- Estimula la formación de hábitos.
- Estimula el trabajo colaborativo y la participación activa.
- Facilita la actualización de contenidos durante el curso.
- El profesor puede acceder a información sobre acciones hechas por los estudiantes (p.ej. número de veces que ha consultado un recurso) gracias a los ficheros de log y otra información de la base de datos de la plataforma de aprendizaje.
- Puede soportar un gran número de alumnos.
- El alumno desarrolla competencias TIC.

Por el contrario, algunos inconvenientes del uso de ésta metodología de aprendizaje son (Ventajas y desventajas del e-Learning, 2008) (Desventajas de e-Learning) (Tatum, 2003):

- El estudiante necesita unos conocimientos previos en el uso de las TICs al igual que los profesores.
- Mantener motivado al alumno para que asuma su propio aprendizaje y así no opte por la renuncia.
- Falta de contacto personal.
- Desorientación general en el acceso a la información.
- Aislamiento (Galusha, 1997)
- Frustración y ansiedad (Kling, 2000)

A pesar de estas desventajas que presenta la metodología e-Learning, actualmente se está llevando a cabo este tipo de aprendizaje.

2.1.2 Investigaciones que reduzcan su impacto en el aprendizaje.

La mayoría de plataformas e-Learning son accesibles mediante una multitud de dispositivos, ordenadores, teléfonos móviles, etc. Es por ello, que son necesarias normas, especificaciones y estándares para poder regular el entorno y así poder ser compatible con la mayoría de las tecnologías posibles.

Por ahora no existe un estándar plenamente consolidado de e-Learning, pero se están haciendo esfuerzos. Mientras tanto, por la gran aceptación y su gran número de herramientas tecnológicas de fácil desarrollo, todos estos intentos de estándares se basan en la interpretación del documento XML (eXtensible Markup Language).

Hay varias iniciativas de definiciones de estándares, entre los que destacan: IMS (Institutional Management System), AICC (Aviation Industry Computed Based-Training Comitee) y SCORM (Sharable Content Object Reference Model).

Las especificaciones IMS cubren un amplio rango de características que se persiguen hacer interoperables entre plataformas, que van desde los metadatos, la interoperabilidad de intercambiar el diseño instruccional entre plataformas, hasta la creación de cursos online para alumnos que tengan alguna discapacidad visual, auditiva u otra.

La guía de AICC CMI001 entrega recomendaciones y lineamientos a seguir para lograr la interoperabilidad entre LMS. Siguiendo estas especificaciones AICC pueden desarrollarse contenidos que serán intercambiables entre distintos LMS que soporten y apliquen la Guía AICC CMI001.

La especificación SCORM logra combinar de excelente forma los elementos de IEEE (Institute for Electrical and Electronic Engineers), AICC e IMS en un único documento consolidado de fácil implementación.

Como se puede apreciar en el informe de investigación (Ramírez, 2007) y en la Figura 1, el proceso de estandarización es iterativo e intervienen desde investigadores y usuarios hasta grupos y cuerpos de estandarización y especificación. Cuando una especificación es aceptada por la mayoría de la comunidad, sigue evolucionando para poder conseguir un amplio consenso. Para esto hay dos vías: la aceptación rápida, la cual se convierte en un estándar de *facto*, o la aceptación lenta en la que no consensuan y cada uno desarrolla a su conveniencia.

Mientras los participantes desarrollan herramientas para poder hacer uso de las especificaciones, se intenta crear un mercado alrededor de la especificación y así intentar lograr su estandarización con el consentimiento del 75% de la comunidad. Una vez ya sea estándar, los organismos, nacionales o internacionales, son los encargados de mantenerlo y actualizarlo mientras dure.

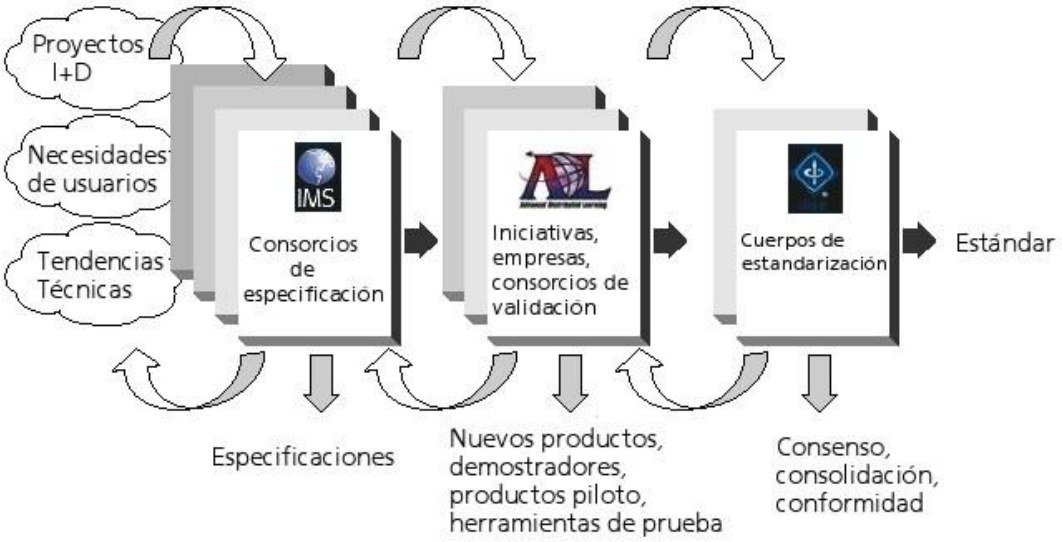

Figura 1: Proceso de Estandarización

2.1.3 e-Learning en la actualidad y su evolución

El desarrollo de e-Learning, enfocado más a la parte tecnológica y como se especifica en el proyecto (Cejudo., 2003), se puede definir en general por tres tipos de herramientas:

- CMS: Content Management System (Sistema Gestor de Contenidos). Este sistema se centra en la gestión y creación de contenidos, principalmente para proyectos pequeños, aunque pueden ser grandes, en los que se necesita que dentro del sistema esté integrado el contenido. Algunas de las herramientas de CMS que se pueden encontrar son Drupal, CoreMedia CMS y PHPNuke.
- LMS: Learning Management System (Sistema Gestor de Aprendizaje). Este sistema está dirigido especialmente para el área educativa, permitiendo la gestión de los contenidos y de los usuarios. Los contenidos de este sistema son creados de manera externa mediante herramientas como Dreamweawer, Golive, Frontpage, Word, etc.
- LCMS: Learning Content Management System. Este sistema integra las utilidades de los anteriores sistemas. Prácticamente es un LMS con un módulo para poder crear contenido. Ejemplos como Blackboard o Saba integran este sistema.

A lo largo del tiempo, estas tres áreas (i.e. CMS, LMS y LCMS) han ido evolucionando. Los primeros trabajos se centraron en los tipos de archivos soportados, luego vino la funcionalidad de las plataformas para poder distribuir contenidos e interaccionar con el usuario, terminando a día de hoy, donde se centran en aspectos abstractos como la metodología.

Evolución de especificaciones relacionadas con el área de contenidos

En la Figura 2, se muestra cómo diferentes iniciativas y proyectos han colaborado para conseguir fomentar al área de contenidos.

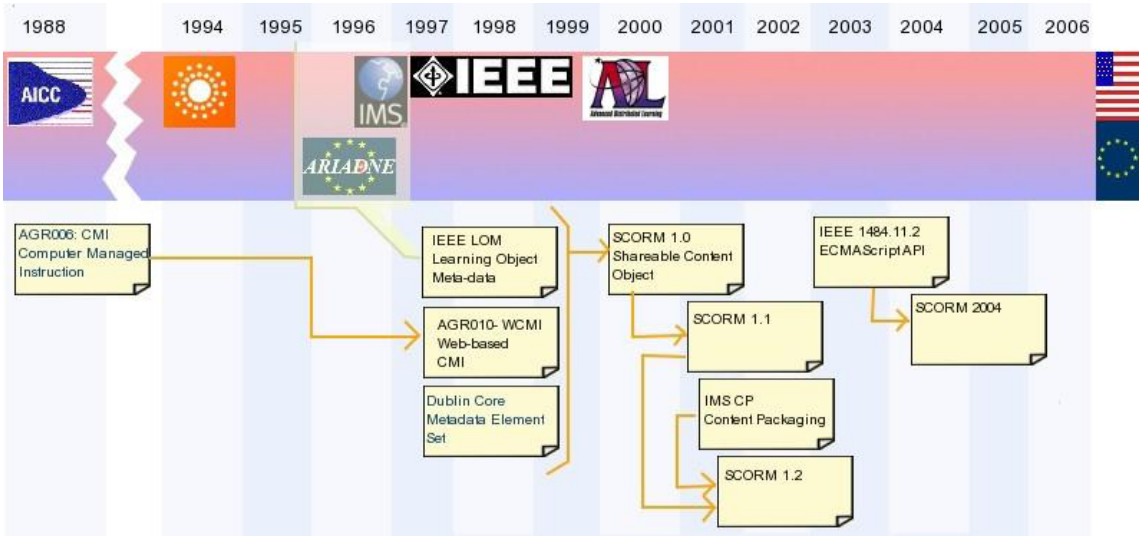

Figura 2: Evolución de especificaciones sobre los contenidos de e-Learning

Evolución de especificaciones relacionadas con el área de sistema gestor de e-Learning

La funcionalidad del sistema e-Learning depende sobre todo del LMS. Esta funcionalidad se consigue con la integración de herramientas que aunque no estén diseñadas para e-Learning, sí son fáciles de integrar en el sistema.
El artículo de Khan (Khan, 2005), hace referencia a una lista de herramientas que cumplen con estos requisitos:

- Herramientas de comunicación.
- Asíncronas: Correo electrónico, listas de distribución, grupos de noticias, etc.
- Síncronas: basadas en texto (chat, IRC, mensajería), videoconferencia, audioconferencia, etc.
- Herramientas de acceso remoto: Telnet, SSH, FTP...
- Navegadores Web.
- Herramientas de búsqueda tanto en el servidor como en la web.
- Aplicaciones cliente-servidor: diccionarios on-line, encuestas, tablones de anuncios y sugerencias, etc.

Inicialmente las especificaciones para esta área se localizaron en los meta-datos, los cuales incluyen el concepto de valor educativo como por ejemplo el nivel de dificultad o la edad de los alumnos, aunque aún insuficiente ya que no permiten actividades a desarrollar en el contenido.

Hay diferentes especificaciones, tales como IMS Simple Sequencing, IMS QTI, SCORM… pero en casi todas hay la contradicción, (y por lo cual hay algunos autores (Griffiths, Blat, Garcia, & Sayago, 2004) que lo advierten) de dos conceptos diferentes de aprendizaje, el conductismo y el constructivismo. SCORM, es el único con una supuesta neutralidad, aunque se decanta por el conductismo, y es por ello que va en auge en cuanto a su utilización por las diferentes organizaciones.

No obstante, es necesario definir nuevas especificaciones para aportar mayor libertad pedagógica y así poder definir una buena metodología en el curso.

Teorías de aprendizaje usadas en e-Learning

Existen diversas teorías del aprendizaje, pero, se pueden fusionar en tres grandes movimientos, como son, conductismo, cognitivismo y constructivismo. Lo que define la diferencia principal entre ellas es la forma como ocurre el proceso de aprendizaje. Por un lado, los modelos conductistas concretan que se logra el aprendizaje a través de un desarrollo estimulo-respuesta-reforzamiento, esto consiste en que un individuo (ej.: el docente) crea una serie de estímulos para lograr que el alumno de una respuesta adecuada a ese estimulo, de ser así, se fortalecerá de manera positiva para que la conducta asimilada sea consolidada y en caso contrario se reforzaría de manera negativa para que desvanezca.

Según los modelos cognoscitivos, las personas actúan en base a sus actitudes, creencias y experiencias y no por los estímulos externos, algunas de ellas plantean que aprender consiste en captar nuevos aprendizajes a la memoria, para que después puedan ser recuperados y usados cuando sea requerido. Mientras que el constructivismo admite que el aprendizaje es un proceso integral donde el individuo se educa en función del entorno y en base a sus propias expectativas y necesidades, para ello el docente debe ayudar que el alumno descubra por si mismo los diferentes aspectos que él considere oportunos (Gros, 1997).

<u>Estilos de Aprendizaje</u>

Los estilos de aprendizaje están distinguidos en varios planteamientos, con propiedad se puede decir que el concepto en si está conectado con las siguientes preguntas: ¿Cómo se instruyen los alumnos?, ¿Por qué asimilan?, ¿Cuando aprenden? Pero, por otro lado, están las posibles respuestas a esta pregunta, J. Cabrera y otros (Juan Silvio Cabrera Albert, 2006) exponen una taxonomía para agrandar la concepción cognitivista de los estilos de aprendizaje, al integrar el componente social y afectivo en nuestro trabajo utilizaremos esta taxonomía.

Taxonomía de procedimientos de aprendizaje (Juan Silvio Cabrera Albert, 2006):

A: Estilos de aprendizaje relacionados con las formas preferidas de los estudiantes de percibir la información (Canales de aprendizaje):

Estilo visual, Estilo verbal-auditivo

B: Estilos de aprendizaje relacionados con las formas preferidas de los estudiantes de procesar la información:

Estilo global, Estilo analítico

C: Estilos de aprendizaje relacionados con las formas preferidas de los estudiantes de planificar su tiempo en el cumplimiento de sus metas como aprendiz:

Estilo planificado, Estilo espontáneo

D: Estilos de aprendizaje relacionados con las formas preferidas de los estudiantes de orientarse hacia la comunicación y sus relaciones interpersonales en el aprendizaje:

Estilo cooperativo, Estilo independiente o individual

Objetos de Aprendizaje

La tecnología en e-Learning o aprendizaje en la red, ha ocasionado una serie de propuestas entre las cuales descubrimos la de Objetos de Aprendizaje o Learning Objects, este concepto cubre una gran cantidad de áreas de trabajo que van desde la composición de objetos de aprendizaje como objetos con UML, hasta la especificación de la estrategia de enseñanza a través de esos objetos. De manera formal existen varias ideas de Learning Objects, nos remitiremos a tres de ellos que considero los más característicos de la diversidad de planteamientos, en ellos se definen los objetos de aprendizaje como:

"...recursos modulares digitales, con identificador único que pueden ser utilizados para el soporte de aprendizaje..." (National Learning Infrastructure Initiative)

"...cualquier recurso digital que puede ser utilizado o reutilizado para dar soporte al aprendizaje..." (Wiley, 2002)

"... cada entidad digital, que pueda ser utilizada, reutilizada o referenciada durante un proceso de aprendizaje soportado por tecnología..." (IEEE Learning Technology Standards committee)

Es evidente que para crear estos contenidos se hace necesario establecer una relación entre el factor pedagógico y el técnico-informático, precisamente es allí donde se halla un gran esfuerzo para lograr que esta mezcla funcione correctamente y posibilite lograr el objetivo de alcanzar el mejor ambiente para los diferentes actores que intervienen del proceso.

Es por todo lo anterior que a partir de algunos ánimos iniciales comienza a desplegarse una tecnología que permitiese abaratar costos y esfuerzos, algunos de estos esfuerzos han derivado en estándares y planteamientos metodológicos, como por ejemplo, IEEE-LOM, SCORM, IMS-LD, entre otros. Para ello se han empleado los avances en aspectos como, tecnología Orientada a Objetos, XML, UML, Ingeniería de Software, Web Semántica, Adaptative Hypermedia Systems, y otros.

2.1.4 Tendencias

Viendo los dos apartados anteriores, se puede observar que tanto en las escuelas como en organizaciones se está enfocando hacia el e-Learning.
Según el informe "El estado del arte de la formación" (El estado del arte de la formación, 2008), las empresas españolas invirtieron 2.095 millones de euros en formación, tanto e-Learning como en presencial y cada año va en aumento. Esto en parte es debido a un porcentaje de bonificación sobre la cuantía ingresada en concepto de formación profesional, dependiendo del tamaño de las organizaciones el porcentaje es mayor. Esto hace que se localice, a grandes rasgos, intereses comunes, uno por parte del trabajador ya que aumenta sus conocimientos y gana valor su currículum, y otro por parte de la empresa, que consigue tener trabajadores mejor preparados y obtiene subvenciones del Estado. Así que, éste informe, nos da una visión general de hacia dónde está el punto de mira de las empresas y que metodología utilizan.

Capítulo 3. Metodología de desarrollo del software

Cuando se va a desarrollar un sistema software es necesario conocer un lenguaje de programación, pero con eso no basta. Si se quiere que el sistema sea robusto y que se pueda mantener es necesario que el problema sea analizado y que sea cuidadosamente diseñada la solución. Se debe seguir un proceso robusto, que incluya las actividades principales. Si se sigue un proceso de desarrollo que se ocupa de plantear cómo se crea el análisis y el diseño, y cómo se interconectan los productos de ambos, entonces la realización de sistemas software va a poder ser planificable y repetible, y la probabilidad de tener un sistema de mejor calidad al final del proceso aumenta considerablemente, y mejor cuando se trata de un equipo de desarrollo formado por varias personas.

3.1 Objetivos

El sistema de la plataforma AdVisor es un LMS puesto que sus contenidos no son creados en la misma plataforma sino que se aportan mediante otras herramientas.

Viendo el entorno de e-Learning y para poder ayudar en el manejo de éste método de aprendizaje, la aplicación que se propone intenta facilitar el trabajo en cuanto a creación de asignaturas. Esta herramienta se crea con criterios innovadores para su funcionamiento reemplazando a la rudimentaria predecesora (VLab), aportando puntos positivos a la plataforma AdVisor.

Conociendo esto, podemos decir que la aplicación que se propone en el Trabajo Final de Carrera es una herramienta de autor.

La definición de una herramienta de autor, trata de la posibilidad de enlazar contenidos multimedia mediante aplicaciones informáticas para usuarios no expertos informáticos, simplemente conexionando objetos, habiendo definido la relación entre estos previamente y ordenándolos en una secuencia determinada. En este caso, para la creación de una asignatura de un curso.

También cabe comentar que AdVisor no cumple del todo con el estándar SCORM ni IMS, ya que estos estándares no albergan todo lo que necesita la plataforma. Al usar algún software actual compatible con SCORM o IMS (como se detallan en el apartado '3.4.1 Herramientas de autor'), sólo se podría formar por Temario-Actividades-Recursos por lo que el resto recursos como Competencias-Objetivos-Sesiones-Evaluaciones-PACs quedarían fuera del estándar, esto supondría que no se podrían relacionar entre ellos. Es por ello que se crea esta aplicación y así poder relacionar todos los recursos necesarios para la creación de una asignatura creando un XML propio para la plataforma AdVisor.

3.2 Metodología Clásica o de Cascada (Waterfall)

En este apartado se expone la metodología clásica de desarrollo, generosamente usada desde el nacimiento de la ingeniería de software hasta la actualidad. Principalmente se describirá por encima la filosofía asociada a esta metodología, los pasos que se siguen con el objetivo de llevar el proyecto a buen puerto y las ventajas e inconvenientes que plantea esta metodología, ya que esta tecnología es trivial y no es tan compleja como otras.

3.2.1 Filosofía

Tal y como su nombre indica, la metodología waterfall está basada en una filosofía lineal (waterfall, en inglés, significa cascada). En ella se observa el proyecto como una serie de pasos que hay que seguir en un orden fijo e invariable. La principal preocupación del equipo de desarrollo es que el proyecto sostenga una estructura fija y muy bien definida, lo cual se alcanza documentando cada paso que se da. De esta forma cada decisión tomada se debe aprobar y, salvo en casos excepcionales, se mantiene permanente en todas las etapas posteriores del proyecto.

Este tipo de pensamiento viene originado de las metodologías empleadas en el desarrollo de proyectos de ingeniería industrial (a causa de que los primeros proyectos de ingeniería de software se hicieron en entornos industriales), en los cuales la retroalimentación entre las diversas etapas era prácticamente inexistente y el proceso necesitaba una estructura rígida e inalterable.

Por todo lo dicho, se puede resumir los principios de esta metodología en tres conceptos: linealidad, rigidez y orden.

3.2.2 Proceso

En este apartado, se definen los distintos pasos (o distintas etapas) que se deben seguir para la finalización de un proyecto mediante el método cascada.

Hay que tener en cuenta que el esquema definido se corresponde con un esquema genérico y que está totalmente abierto a modificaciones. Las distintas etapas por las que pasa un desarrollo siguiendo esta metodología son las siguientes:

- Análisis y definición de requerimientos: Los servicios, restricciones y metas del sistema se definen a partir de las consultas con los usuarios. Entonces, se definen en detalle y sirven como una especificación del sistema. Habitualmente esta especificación se mantiene invariable hasta el final del desarrollo.
- Diseño del sistema y del software: El proceso de diseño del sistema divide los requerimientos en sistemas hardware o software. Establece una arquitectura completa del sistema. El diseño del software identifica y describe las abstracciones fundamentales del sistema software y sus relaciones.
- Implementación y prueba de unidades: Durante esta etapa, el diseño del software se programa como un conjunto de subsistemas de software. Cada

subsistema se debe testear por separado para asegurar que cumpla su especificación.

- Integración y prueba del sistema: Los subsistemas de software se integran y se prueban como un sistema completo para asegurar que se cumplan los requerimientos del software. Después de las pruebas, el sistema software se entrega al cliente.
- Funcionamiento y mantenimiento: Por lo general (aunque no necesariamente), ésta es la fase más larga del ciclo de vida. El sistema se instala y se pone en funcionamiento. El mantenimiento implica corregir errores no descubiertos en las etapas anteriores del ciclo de vida, mejorar la implementación de las unidades del sistema e implementar nuevas características una vez que se descubren nuevos requerimientos.

En la siguiente Figura 3 se puede ver un esquema del proceso descrito.

Primeramente, el resultado de cada fase es uno o más documentos aprobados, y la fase siguiente no debe empezar hasta que la fase previa haya terminado. En la práctica, estas etapas se sobreponen y proporcionan información a las otras, por ejemplo, durante el diseño se encuentran problemas con los requerimientos; durante la implementación se descubren errores de diseño, y así sucesivamente.

La marcha del desarrollo de un proyecto de ingeniería de software no es un modelo lineal simple, sino que mezcla una serie de iteraciones de las diferentes etapas. Incluso, debido a los costos de producción y aceptación de documentos, las iteraciones son costosas e supone rehacer trabajo. Por ello, lo normal es que tras unas pocas iteraciones se paralicen ciertas etapas del desarrollo (como puede ser la especificación de requerimientos) para poder continuar con las siguientes. Esto último induce que se deban pasar por alto algunos problemas que puedan surgir en etapas posteriores, con lo cual, al final, podríamos obtener un software mal estructurado (debido a errores de diseño solucionados mediante trucos de implementación) o que no hace lo que el cliente quiere que haga (como consecuencia de la omisión de requerimientos).

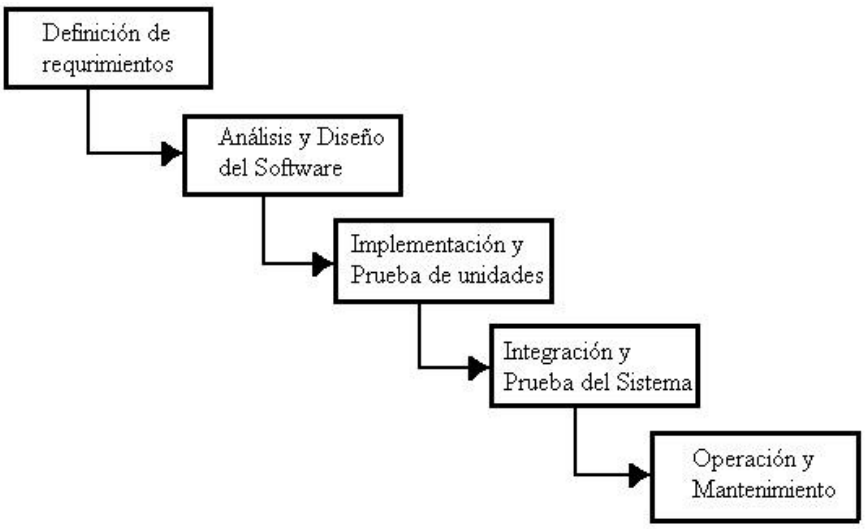

Figura 3: Esquema de la metodología Cascada secuencial

3.2.3 Ventajas e inconvenientes

La metodología de Cascada presenta diversas ventajas como:

- La planificación es sencilla.
- Si se conocen el total de los requerimientos desde el principio, la calidad del programa resultante es alta.
- Permite trabajar con personal poco calificado.
- Se tiene todo bien organizado.
- No se mezclan las fases.
- Es perfecto para aquellos programas que son rígidos donde se especifiquen muy bien los requerimientos y se conozcan muy bien las herramientas a utilizar.

También hay desventajas y problemas que surgen a la hora de implementarlas:

- La duración de todo el ciclo es muy larga.
- Probabilidad alta de fracaso dado que existe poca comunicación con el usuario final.
- El mantenimiento se realiza en el código fuente.
- Las revisiones de programas de gran complejidad son muy difíciles.
- Impone una estructura de gestión de proyectos.
- Para que el programa tenga éxito deben desarrollarse todas las iteraciones.
- Si se cambia el orden de las fases el programa final será de menor calidad.
- Se retrasa la localización y corrección de errores.
- Puede producir programas poco llamativos para los usuarios ya que no se le pueden hacer muchas modificaciones según la marcha.
- Inflexibilidad del modelo: dificultad para responder a cambios en los requerimientos.

3.3 Determinación de Metodología

En el proyecto realizado se usará la metodología de cascada. Dadas las características del proyecto existen otras metodologías que se adaptarían mejor a las características del proyecto pero en este caso particular por motivos relacionados con la organización de la empresa (la Universidad) la metodología que se ha optado al final ha sido por Waterfall.

Las decisiones de esta selección han sido claras y concisas:

- Por tiempo. Tener un tiempo ajustado para la realización del proyecto y desarrollar la aplicación hace que su planificación tenga que ser más sencilla de lo establecido.
- Por requerimientos. Al conocer todos los requerimientos desde un principio, la calidad del programa obtenido será mejor.
- Por personal poco calificado. Al adentrarme en lenguajes de programación y tecnologías que no conozco o conozco levemente hace que Waterfall perfecto para mí.

- Por organizado. Al ser una metodología lineal hace que todo esté bien organizado.

3.4 Análisis de la aplicación

El análisis de la aplicación se realiza con el fin de obtener una visión inicial de las necesidades que deberán ser satisfechas, que servirán de base para el diseño y posterior construcción de la herramienta.

3.4.1 Herramientas similares

En nuestro caso, el profesor no tiene que generar material en formato electrónico porque ya suponemos que lo tiene creado, sólo tenemos que ensamblar la asignatura a partir de éstos, por eso debemos empaquetarlos y estructurarlos bajo el lenguaje XML. Teniendo clara esta premisa, se analiza una serie de herramientas de autor que empaquetan recursos de aprendizaje. Esto permitirá extraer algunas conclusiones que servirán para definir mejor el alcance de este TFC.

Reload Tools

Descripción

Viene compuesto por un editor y por un pre-visualizador SCORM, siendo todo ello gratuito y de código abierto para facilitar el desarrollo de nuevas herramientas.
Con Scorm Player se puede probar los contenidos empaquetados por el Reload Editor como si estuviéramos en un ambiente de ejecución.

Por la parte del editor, asegura que los contenidos cumplan con el estándar, tanto SCORM como para IMS-LD, pudiendo crear, editar, importar y exportar paquetes de contenido.

El programa en sí puede crear cuatro tipos de archivos, los cuales son:

- El archivo “IMS Metadata”.
- El paquete de contenido IMS.
- El paquete de contenido SCORM 1.2 de la ADL.
- El Diseño de Aprendizaje IMS.

Para empaquetar los objetos con el estándar que hemos mencionado anteriormente, se utiliza la tercera opción, la cual te genera el archivo *imsmanifest.xml*, el cual describe el contenido del paquete e incluso la estructura del contenido, este archivo debe estar en la raíz y es obligatorio.

Una vez hecha la organización, creado los recursos, dotarlos de contenidos y hecha la referencia se procede a exportar el paquete y ahora es cuando se genera el paquete .zip con todo el contenido necesario.

Más abajo, como se ve en la Figura 4 se puede apreciar la estructura interna de un paquete.

Valoración

- De fácil uso y de libertad creativa.
  Gracias a esto, es uno de los validadores de paquetes más usado de por las organizaciones, pudiendo crear un paquete 100% sin problemas de compatibilidad.

- Interoperabilidad y estándares.
  Todavía Reload no puede pasar de un formato SCORM a uno Learning Design pero aún y así es de agradecer su versatilidad y su compatibilidad con éstos dos estándares.

- Adaptable a distintos estilos de aprendizaje.
  Múltiples organizaciones, tanto LD como SCORM.

- Ficheros multimedia.
  No presenta restricciones

- Extensibilidad.
  Al ser un proyecto de código abierto, gratuito y aún activo.

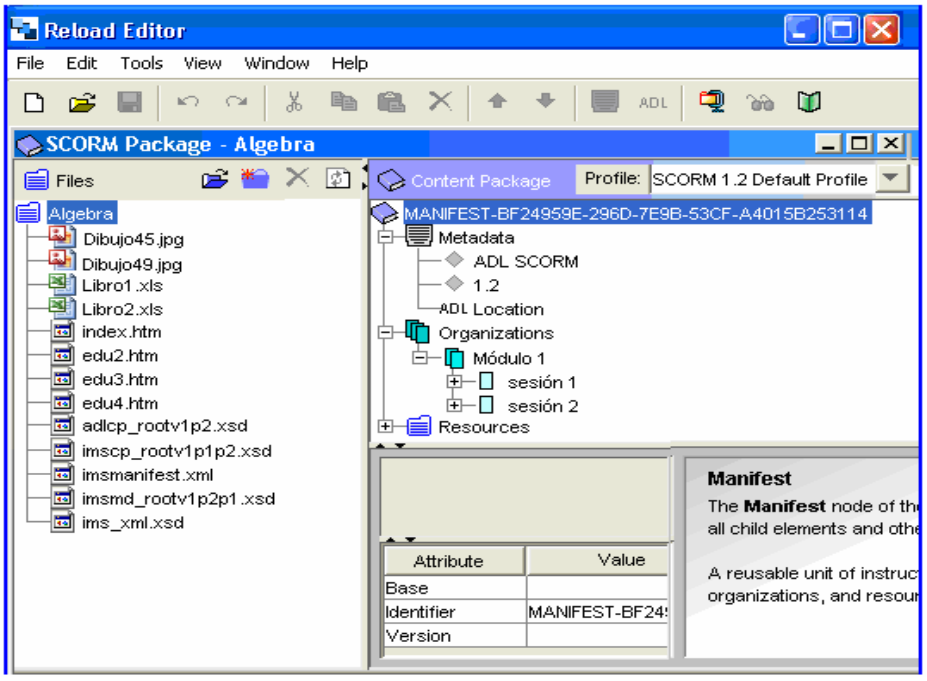

Figura 4: Pantalla principal de Reload Editor

Delta Learn

Descripción

Esta herramienta se encuentra algo limitada en cuanto a la creación de contenido y evaluación. Permite crear la estructura fácilmente, todo lo contrario al de los contenidos.

Como Reload, también posee un reproductor de SCORM. Además, Deltalearn goza de una gran cantidad de opciones como deshacer acciones, añadir manifiestos, borrar referencias e incluso tiene un editor propio de HTML y RTF.

Valoración

- De fácil uso y de libertad creativa.
  Al igual que Reload, es fácil de usar pero se necesita conocimiento del estándar para que el contenido se rija como las definiciones preestablecen.
  No formatea contenido por lo que hay que ver mediante pruebas si estos son presentados de una forma idónea.

- Interoperabilidad y estándares.
  Sólo es compatible con SCORM 1.2, 1.3 y 2004 pero se puede editar ampliamente el manifiesto y los metadatos. Además, permite añadir metadatos a través del asistente.

- Adaptable a distintos estilos de aprendizaje.
  Permite agregar diferentes manifiestos y es compatible con SCORM 1.2, 1.3 y 2004.

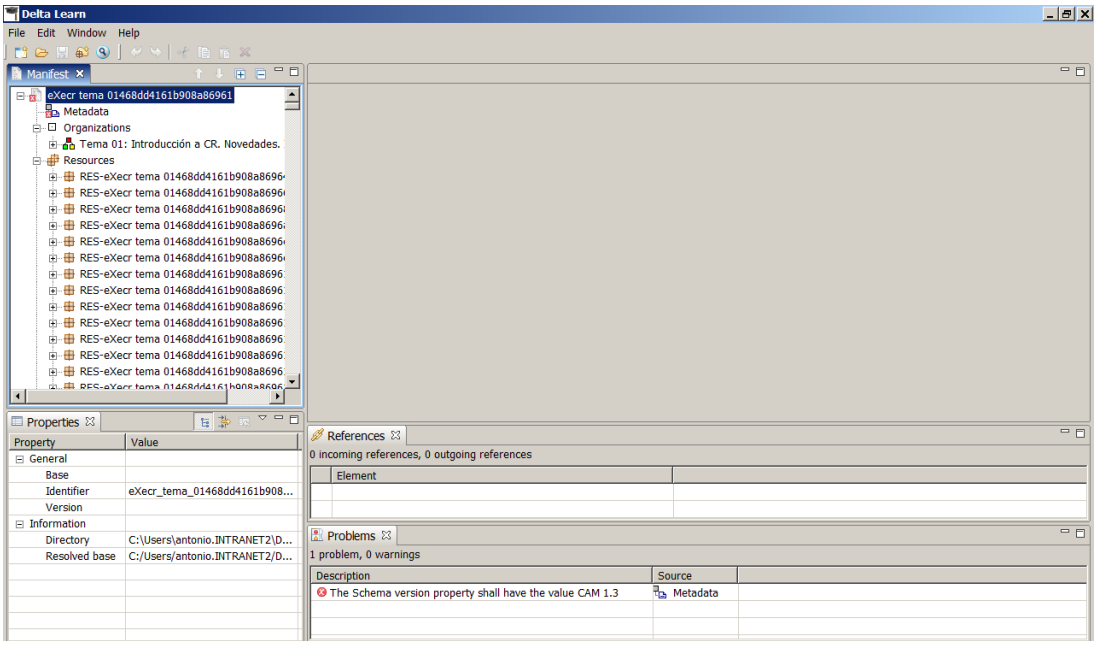

Figura 5: Pantalla principal de Delta Learn

Trident IDE

Descripción

Podría decirse que Trident es una mejora de Delta Learn ya que presenta todas las ventajas de éste y además ofrece un buscador de recursos, una conformidad total con SCORM 2004 v3 y un reproductor de objetos mejorado.

A pesar de todas estas mejoras conforme a Delta Learn, no justifica el precio respecto con Reload, que sigue siendo un referente por todas sus opciones con el aditivo de ser gratuito y de código abierto.

Valoración

Es una mejora de Delta Learn pero no aumenta sustancialmente como para añadir alguna valoración diferente al mencionado.

Thesis Pro

Descripción

Esta aplicación puede diseñar y distribuir contenidos formativos cumpliendo con el estándar SCORM, así los desarrolladores de contenido no tiene por qué conocer la normativa SCORM para crear un contenido compatible con cualquier plataforma LMS.

Su interface es gráfica y contiene 3 módulos, uno para Microsoft Word, otro para Microsoft PowerPoint y el último para Microsoft Producer 2003.

Como se observa en la Figura 6 , se puede ver como se integra en Office y ofrece unas funcionalidades muy fáciles de utilizar para un usuario de este programa.

Valoración

- Simple e integrada.
  Herramienta de autor más familiar, ya que se integra con Microsoft Office.

- Ahorro en costos.
  El periodo de formación disminuye en cuanto su uso, así como también se reduce los costes de formación.

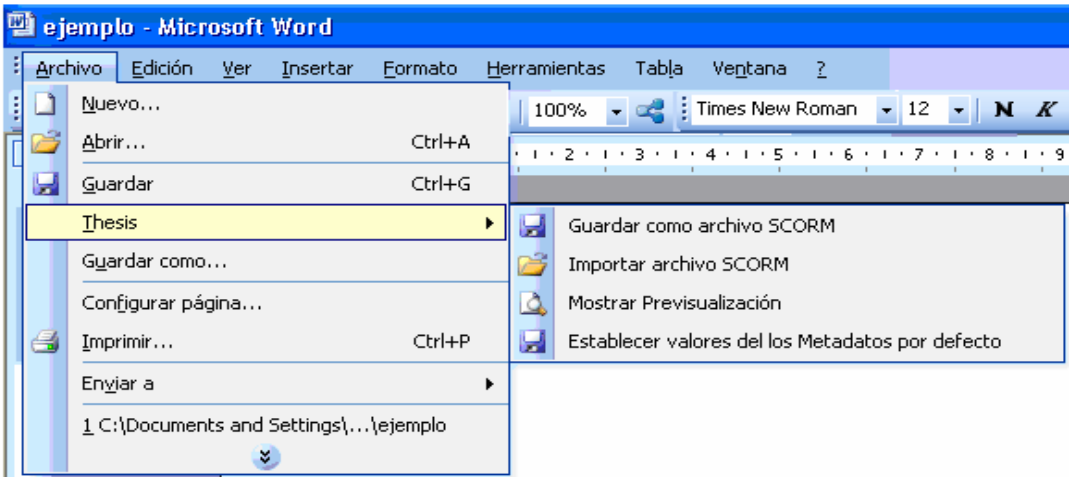

Figura 6: Integración de Thesis Pro con Microsoft Word

3.4.2 Ideas a partir de una visión general de las Herramientas de autor expuestas

Viendo en qué dirección van las Herramientas de autor explicadas nos aportar ideas generales de cómo tiene que desarrollarse la aplicación.

- Para empezar, la herramienta tiene que ser muy usable, es decir de fácil manejo, pero a la vez lo suficientemente compleja para no limitar al usuario en cuanto a la creación de la asignatura.

- Asimismo, se ha de poder mover, eliminar, crear, buscar elementos, con la idea de en un futuro poder crear algún tipo de asistente para la creación de una asignatura a partir de una ya existente.

- Se podrán crear tantos elementos como se quiera, no tienen por qué tener un número limitado.

- El profesor tiene que tener una visión de las relaciones, del trabajo hecho, para evitar que se pierda y que se equivoque en el intento de crear una asignatura.

- El formato de la aplicación debe ser claro y bien organizado ya que en un futuro se podría intentar estandarizar la aplicación y de esta manera ahorraría costos en tiempo, esfuerzo y dinero.

- Los metadatos se deben generar automáticamente y de manera transparente al profesor, ya que no tiene por qué saber cómo funciona este tipo de tecnología, ni para qué se utiliza.

- La herramienta debe de ser extensible.

3.4.3 Tipos de usuario

Para esta aplicación los tipos de usuario se simplifican, ya que al uso, solamente existe un tipo de usuario y son los profesores.

Cada profesor debe de tener su identificación y su clave de acceso para poder utilizar la aplicación. Así que una vez identificados, el profesor podrá acceder a ésta y se le mostrará las asignaturas, si las tuviera, relacionadas con su usuario.

Por último, todos los profesores podrán utilizar la aplicación por completo, es decir, la aplicación no tiene partes restringidas dependiendo del usuario conectado.

3.4.4 Funciones de la aplicación

Como se ha mencionado en el punto anterior, todos los usuarios tendrán acceso a todas las funciones de esta herramienta. Estas funciones son:

- Creación/Modificación de una asignatura.
- Introducción de Actividades, Recursos, Temario, Sesiones, Objetivos y Evaluaciones.
- Borrado de Actividades, Recursos, Temario, Sesiones, Objetivos y Evaluaciones.
- Modificación de datos de Actividades, Recursos, Temario, Sesiones, Objetivos y Evaluaciones.
- Creación de archivo comprimido con los datos de la asignatura creada/modificada.

3.4.5 Técnicas de modelado para la aplicación

Haremos una breve descripción de las tres opciones principales para el análisis y diseño de sistemas.

Descomposición Funcional

Según esta metodología, el diseñador debe centrar su atención en los procedimientos y algoritmos necesarios para que el sistema lleve a cabo su tarea. Inicialmente, el sistema es visto en términos de lo que debe hacer y luego en la etapa de diseño se determina como lo va a hacer. Las herramientas de diseño que dan soporte a esta metodología están basadas en el flujo de datos de la aplicación, e incluyen los Diagramas de Flujo de Datos (DFDs), los Diccionarios de Datos (DDs) y los Diagramas de Estructuras.

Existen varios lenguajes de programación que dan soporte a la descomposición funcional: Java, Cobol, Pascal, C, etc. La tendencia en estos lenguajes de colocar los datos compartidos en áreas globales produce los llamados 'efectos colaterales' (stack of dominoes), tan conocidos por aquellos que trabajan en el desarrollo y mantenimiento de programas, donde los cambios hechos en una parte del sistema es fuente de problemas en otras áreas aparentemente disociadas. Por otra parte, un sistema diseñado con una única función en mente, difícilmente podrá asimilar los cambios evolutivos causados por modificaciones o extensiones.

Desarrollo Estructurado de Jackson

La Programación Estructurada de Jackson (Jackson Structured Programming) y el Desarrollo Estructurado de Jackson (JSD, Jackson Structured Development) [Jackson] son técnicas que modelan el mundo real con un enfoque parcialmente orientado por objetos, donde se utilizan métodos basados en las estructuras de datos. Sin embargo, son esencialmente metodologías de descomposición funcional [Sommerville, p.185] en donde las estructuras de datos son utilizadas como soporte a esta descomposición. A diferencia de la descomposición funcional en donde las estructuras de datos están fundamentalmente determinadas por la estructura funcional, se puede decir que JSD es una metodología parcialmente orientada por objetos o de centro-afuera (middle out), que hace menos énfasis en la parte funcional y comienza el análisis del sistema como un proceso de moldeado.

Desarrollo Orientado por Objetos

El paradigma Orientado por Objetos (OO) utiliza los mismos componentes de los sistemas de software: datos y procedimientos. Sin embargo, atenúa los procedimientos y hace énfasis en el encapsulamiento de datos y procedimientos en entidades con una interfaz claramente definida. En la descomposición de sistemas basada en un enfoque OO, el sistema es visto como una colección de entidades (objetos) que encapsulan cierto conocimiento (datos) y cierto comportamiento (procedimientos), y que interactúan para realizar la tarea o tareas del caso. Las etapas de análisis y diseño se llevan a cabo con base en estos objetos y en los servicios que éstos prestan, utilizando un modelo cliente-servidor donde los objetos interactúan por medio de mensajes. El uso de este modelo cliente-servidor hace que el sistema sea descrito como 'orientado a responsabilidades' (responsibilty-driven). El diseño detallado, incluyendo la implementación de procedimientos y definición de estructuras de datos, se difiere hasta una etapa más avanzada del proceso de desarrollo y es privada a cada clase de objetos, adhiriendo estrictamente al concepto de ocultamiento de información. De esta forma los procedimientos y estructuras de datos no quedan 'congelados' desde las primeras etapas del diseño, cuando todavía no se tiene un conocimiento profundo del sistema. Un sistema representado en función de objetos es por lo tanto más flexible y la vez menos sensible a los cambios estructurales. Es importante que las estructuras de datos no sean especificadas muy temprano en el proceso de diseño. En consecuencia el desarrollo orientado por objetos se centra en la abstracción de datos en vez de fijar una estructura determinada en la especificación de los objetos que componen el sistema.

En contraste con el enfoque de descomposición funcional, el enfoque OO si bien tiene atributos y está caracterizado por una metodología, donde el fin principal está en generar clases de objetos que puedan ser incluidas en bibliotecas de uso general. Un enfoque que involucra una técnica de análisis y una técnica de diseño es probable que dé como resultado sistemas más robustos. En esta metodología la línea divisoria entre las diferentes etapas del ciclo de vida del software (A/D/I) desaparecen, tornándose un proceso continuo e iterativo donde analistas, diseñadores y programadores trabajan con entidades comunes: objetos.

Por último, una de las características que diferencian radicalmente la metodología OO de la metodología tradicional basada en la descomposición funcional está en los objetivos de cada una. Mientras que en el enfoque tradicional el fin es construir un sistema que de solución a un problema determinado (y este sistema se construye desde cero), en el enfoque OO el fin principal está en generar clases de objetos, que en conjunto den la solución requerida, pero que además puedan ser incluidas en bibliotecas para su futura reutilización. La reutilización de componentes probados y optimizados es lo que marca la verdadera diferencia entre las metodologías y se traduce en un aumento considerable de la productividad.

3.4.6 Determinación de Modelado

Para esta aplicación se ha escogido el último modelado expuesto. En base al desarrollo orientado a objetos, representa una gran ventaja para el desarrollo de aplicaciones, especialmente por su mantenimiento y adaptabilidad de los cambios de procesos, es por eso que se logra una mayor robustez del programa y puede representar mejor lo deseado por el cliente. Además que ayuda a la reutilización y extensión del código permitiendo crear una aplicación más compleja y acercando la herramienta al mundo real.

3.5 Diseño de la aplicación

> El propósito del diseño es el de crear una arquitectura para la naciente implementación, [...] el diseño arquitectural sólo puede comenzar una vez que el equipo tenga un entendimiento razonable de los requerimientos del sistema. [...] El diseño, como el análisis, nunca termina realmente hasta que el sistema final es entregado. Durante esta fase, se alcanza un cierto grado de culminación al poner en su lugar la mayoría de las decisiones estratégicas de diseño y al establecer políticas para diversos problemas tácticos. [...] El diseño se enfoca en la estructura, estática y dinámica, su propósito principal es de crear el esqueleto' concreto del sistema sobre del cual todo el resto de la implementación se basa. (Gomma, 1995)

Estas palabras definen perfectamente qué es el diseño. La creación de la estructura básica del sistema es la tarea clave, aunque también se buscan otras cosas, en particular patrones que simplifiquen el diseño y posibilidades de rehúso entre otras.

3.5.1 Patrón de diseño

Se ha determinado usar, para la aplicación, el patrón de diseño Modelo Vista Controlador (MVC), el cual, desacopla los conceptos para estructurar la aplicación de una mejor manera, con la intención de facilitar la reutilización de ésta. Es decir, intenta que las modificaciones futuras impacten en menor medida a la lógica de los datos.
El MVC en aplicaciones web viene predefinido por 3 capas:

- Vista. Esta parte es la que presenta el modelo en un formato adecuado para interactuar. En nuestro caso, por ejemplo, al ser una aplicación web la "Vista" es la página HTML.

- Controlador. Es la capa que responde a los eventos, normalmente del usuario, y estas acciones hacen peticiones a la capa "Modelo" y a la "Vista" cuando sea necesario. Es el código que recoge los datos dinámicamente y genera contenido HTML si es necesario, en base a ellos.

- Modelo. Ésta última parte es la que se encarga de la base de datos como SQL o XML, donde se define la funcionalidad del sistema y administra la lógica de la aplicación.

  Para una mejor representación visual del MVC se puede observar la Figura 7.

Figura 7: Modelo MVC

La estructuración mediante este modelo agrega una serie de ventajas y desventajas a la aplicación que ahora se detallaran.

- Ventajas
  - Sencillez en representar los datos en diferentes diseños.
  - Separación de los datos de la representación visual de éstos.
  - Creación de independencia de funcionamiento.
  - Simplifica la detección de errores localizándolos y ubicándolos antes.
  - Mejora la escalabilidad.
  - Facilidad de agregar nuevos datos al ser independientes de las otras capas.

- Desventajas
  - Complejidad del sistema al separar los conceptos.

- Aumento de archivos a mantener y desarrollar.

Como se ve, hay bastantes ventajas interesantes como para escoger este modelo, y es por ello que se va a utilizar para esta aplicación, no obstante, este tema puede ser muy subjetivo y dependiendo de la persona puede ser más útil o no.
Con el convencimiento de usar MVC se separará estas capas utilizando JSF, HTML y CSS para la "Vista", la parte de "Controlador" se utilizará JAVA y por último, la capa de "Modelo" se usará MySQL.

Así que ésta aplicación se va a estructurar lo máximo posible mediante el patrón de diseño MVC, se ha intentará separar las funciones y definirlas bien para que sean lo más dinámicas posibles y modulables, así en un futuro, si hubiera que hacer una modificación, que fuese lo más viable y sin que se cambiase gran parte de la aplicación.

Aunque no siempre puede ser así, ya que el MVC es un patrón general y cada uno tiene que adaptarlo a su aplicación en la mejor medida de lo posible.

3.5.2 Diseño MVC de la aplicación

La muestra del diseño incluye una visión general del sistema para posteriormente detallar los puntos fundamentales del mismo. Como en el caso de los requisitos, se muestra el diseño alcanzado tras todas las iteraciones realizadas.

En la Figura 8 se puede ver una visión general por capas de la aplicación. En la parte superior se encuentra la capa de presentación (capa Vista del modelo MVC), es decir, la interfaz. Esta capa se subdivide a su vez en dos capas más específicas; el HTML y JSF. Mediante estas capas, se generan las distintas páginas web.

Una capa por debajo se encuentra la lógica de la aplicación (capa Controlador del modelo MVC). En esta capa se incluye toda la funcionalidad de todas las operaciones que soportará el sistema.

En el caso de la última capa, está fuertemente vinculada a la capa de la lógica ya que la aplicación usa directamente las sentencias (o funciones) que provee el propio lenguaje SQL (capa Modelo del modelo MVC).

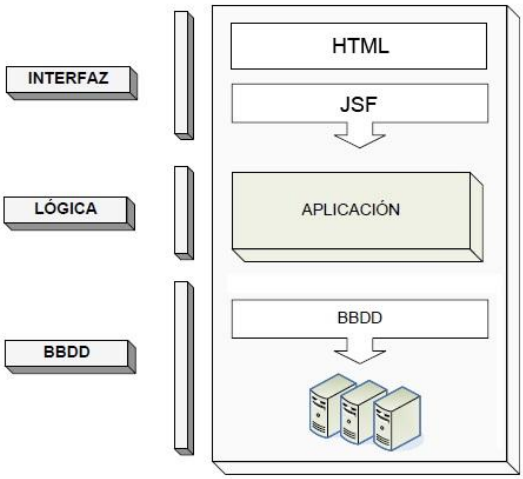

Figura 8: Visión general del sistema

3.5.3 Reglas de funcionamiento

Una vez hecho el análisis previo, se ha de modelar la aplicación para un funcionamiento correcto y poder llevar al usuario por dónde realmente se le quiere llevar, para esto, se necesita un modelo de navegación.

Modelo de Navegación

En este punto, se deben recoger los requisitos de navegación de la aplicación. Para ello, se introduce el modelo de navegación que nos ayuda a dar una vista navegacional (mapa navegacional) para cada tipo de usuario que pueda interactuar con el sistema, que en nuestro caso y como ya se ha mencionado anteriormente, sólo habrá un tipo de usuario, así que habrá navegación completa para todos los usuarios, sin restricciones.

Esta vista nos dará una estructura de accesibilidad (definiendo el conjunto posible de caminos de navegación). Así la semántica navegacional de la aplicación web se captura en función requisitos preestablecidos y del usuario, aceptando una personalización del acceso en función de las necesidades de cada tipo de usuario si fuese necesario. Para crear este modelo, se realizan 2 tareas:

1. Clasificación e Identificación de usuario. Se crea un estudio del tipo de usuario que puede interactuar con el sistema y sus interrelaciones.

2. Construcción de los mapas navegacionales. Para el tipo de usuario detectado, se ha creado una vista navegacional del sistema completa.

Identificación de usuario

En esta etapa se construye un diagrama de agentes (usuarios), donde se especifican qué tipos de usuarios van a poder interactuar con el sistema, qué interrelaciones existen entre ellos y cuál va a ser su modo de acceso al sistema.

En primer lugar se debe estudiar los potenciales de los tipos de usuarios interactivos con el sistema y crear un agente por cada uno de ellos. Una vez identificados, se debe realizar un estudio para detectar interrelaciones entre ellos buscando propiedades comunes (todos los usuarios tendrán el mismo privilegio) creando una taxonomía de usuarios, que permitirán reutilizar especificaciones navegacionales.

En nuestro caso, como bien se sabe, los usuarios de la aplicación serán los docentes que tengan que actualizar o crear una asignatura.

Construcción de los Mapas Navegacionales

Esta fase consiste en crear los mapas navegacionales para cada usuario. Para ésta aplicación será trivial ya que las funciones son para todos equivalentes, es decir, la única restricción que puede haber vendrá representada por la relación que exista en la base de datos sobre los usuarios, así que un profesor sólo va a poder crear/modificar asignaturas que tenga asignadas previamente.

Reglas de funcionamiento preestablecidas sobre la aplicación

Una vez construido el mapa navegacional y el tipo de usuario se empezará a establecer las reglas internas de la aplicación que vienen marcadas por los requisitos previos marcados por el cliente. Estas reglas serán:

- Al crear una sesión, las fechas deben seguir esta regla: F. inicio <=F. fin
- Para una PEC, las fechas deben seguir esta regla: F.inicio<F.entrega<=F.solución<= F.nota
- Una Actividad sólo puede tener una Sesión.
- Una Sesión puede estar en más de una Actividad.
- Un Recurso, Temario, PEC, Objetivo o Sesión no se pueden repetir para una misma Actividad.

3.5.4 Diseño de la interfaz de usuario

La interfaz de usuario es la manera en que los usuarios pueden comunicarse con una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo.

El diseño de la interfaz es uno de los elementos principales en que se basa la realización del programa. Esta importancia toma un relieve especial en las aplicaciones web. Esto puede definirse como:

> Conjunto de trabajos y pasos que seguirá el usuario, durante todo el tiempo que se relacione con el programa, detallando lo que verá y escuchará en cada momento, y las acciones que realizará, así como las respuestas que el sistema le dará (Perea, 1996).

Modelo de Presentación (Interfaz)

Una vez creado el modelo de navegación es necesario declarar las características de presentación del sistema. Para este propósito se adentra el modelo de presentación, que une la información capturada en el modelo de navegación. Este modelo utiliza los nodos o contextos navegacionales como entidades básicas para definir las propiedades de presentación de información.

El modelo de navegación condiciona las características de presentación del sistema. Éste modelo usa los nodos o contextos navegacionales como entidades básicas para definir las propiedades de presentación de información.

Segmentando el modelo de navegación, es posible obtener de forma sistemática un esqueleto de una aplicación web formado por un conjunto de páginas web interconectadas entre sí. Este conjunto de páginas web define la interfaz de

usuario de la aplicación web para la navegación, la visualización y el acceso a su funcionalidad e información.

Incorporando a esta información la especificación construida en el modelo de presentación podrá definirse un modo de visualización de esta información intentando dar la máxima flexibilidad para poder cambiar características estéticas de las páginas web una vez construidas.

Originalmente se diseña la página de inicio o login mostrada en la Figura 9. Posteriormente por cada contexto navegacional o subsistema definido en el mapa de navegación se crea una página web, como se puede observar en la Figura 10 y Figura 11.

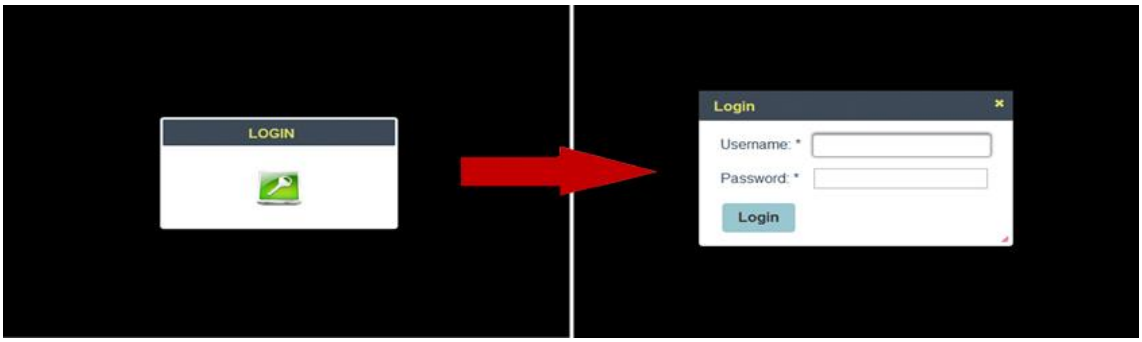

Figura 9: Páginas de inicio de la aplicación

Figura 10: Selección de asignatura existente o creación de una nueva

Figura 11: Última página de la aplicación

En la Figura 12 se puede ver cómo se diferencian los bloques de una asignatura en 6 áreas:

- Actividades
- Competencias y Objetivos
- Planificación
- Recursos
- Temario
- Evaluación

A la vez en la cabecera de la página principal se puede ver las opciones de Guardar, Cargar y Desconexión de una asignatura como también un pequeño desplegable para el cambio de piel de la propia interfaz

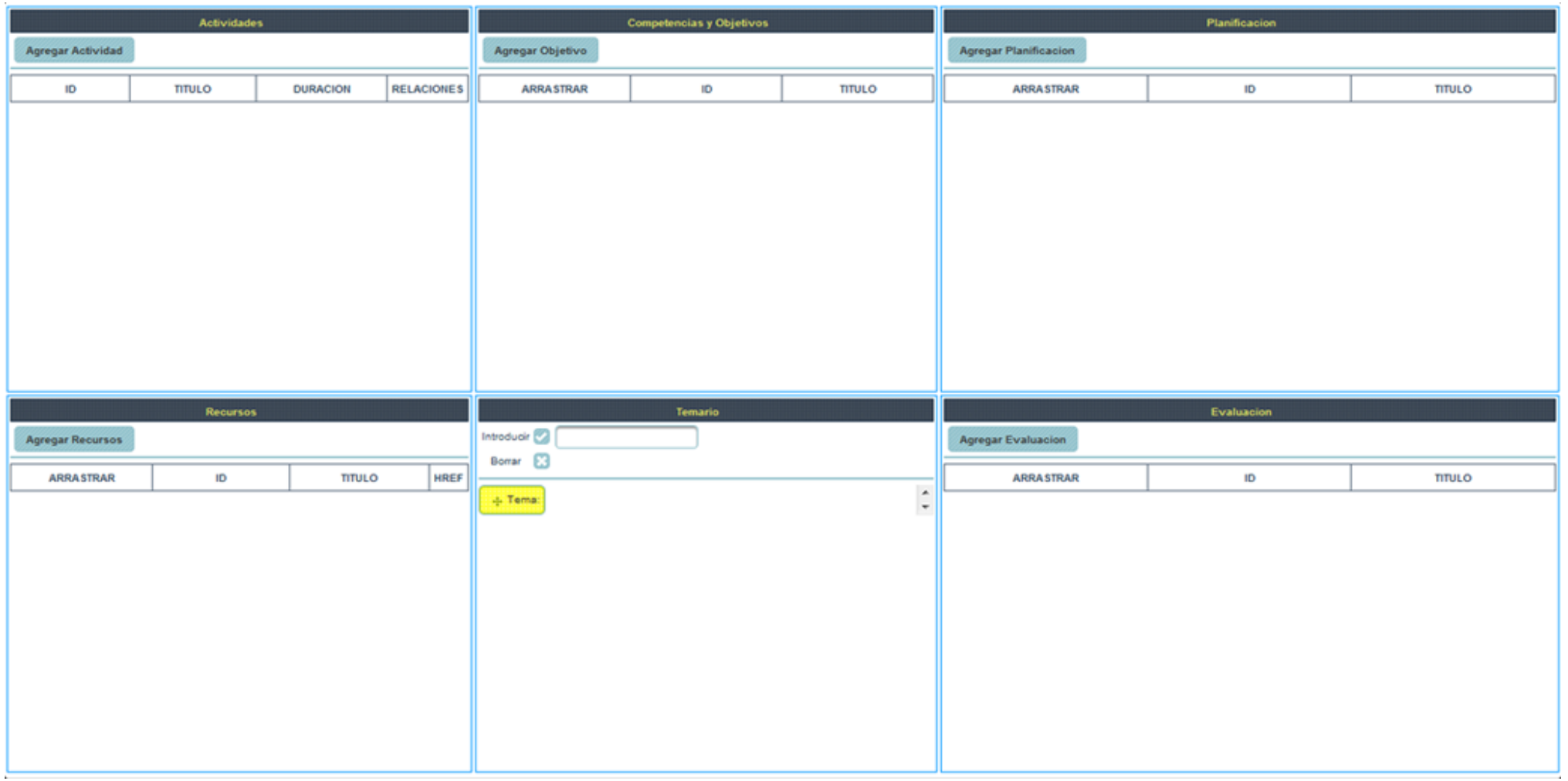

Figura 12: Página Principal

3.6 Desarrollo de la aplicación

En este apartado se va a detallar los lenguajes usados para el desarrollo de la aplicación, así como lenguajes descartados y los detalles de la base de datos (BD).

3.6.1 Lenguajes usados

Para la creación de aplicaciones web se emplean múltiples lenguajes. En este trabajo, se han utilizado diferentes lenguajes de programación web como pueden ser HTML, Java, CSS o JavaScript.

- **Hyper Text Markup Language (HTML)**

El Hyper Text Markup Language (HTML) [Specification, 2004], es un lenguaje de marcado, diseñado para definir textos y definir su presentación en forma de hipertexto, que es el formato estándar de las páginas web. Gracias a Internet y a los navegadores del tipo Mozilla, Firefox, Netscape o Explorer, el HTML se ha convertido en uno de los formatos más populares que existen para la construcción de documentos.

Contrariamente a otros lenguajes de programación, el HTML utiliza etiquetas o marcas, que radican en breves instrucciones de comienzo y final, mediante las cuales se determina la forma con la que deben aparecer el texto, así como las imágenes y los demás elementos, en la pantalla del ordenador.

- **Cascading Style Sheets (CSS)**

Las hojas CSS son un lenguaje formal usado para determinar la presentación de un documento estructurado escrito en HTML. El World Wide Web Consortium (W3C) (Consortium, 2004), es el encargado de formular la especificación de las hojas de estilo que servirá de estándar para los navegadores.

La idea que se reune detrás del desarrollo de CSS es separar la estructura de un documento de su presentación. La información de presentación se proporciona separada en una hoja de estilo con la que se especifica como se ha de mostrar: color, fuente, alineación del texto y tamaño, además puede ser adjuntada tanto como un documento separado o en el mismo documento HTML.

Las ventajas de utilizar CSS son:

- Control centralizado de la presentación de un sitio web completo, con lo que se agiliza de forma considerable la actualización del mismo.
- Los navegadores dejan a los usuarios especificar su propia hoja de estilo local, que será aplicada a un sitio web remoto, con lo que aumenta considerablemente la accesibilidad. Por ejemplo, personas con deficiencias visuales podrían configurar su propia hoja de estilo para aumentar el tamaño del texto o remarcar más los enlaces.
- Una página puede tener diferentes hojas de estilo según el dispositivo que la muestre o incluso a elección del usuario. Por ejemplo, para ser impresa o mostrada en un dispositivo móvil.

- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño.

Hay varias versiones: CSS1 y CSS2, con CSS3 en desarrollo por el W3C. Los navegadores modernos recogen CSS1 de forma completa, aunque existen pequeñas diferencias de implementación dependiendo del navegador.

- **JavaScript**

JavaScript (Eich, 2001), es un lenguaje que analiza y está orientado a las páginas web basado en el paradigma prototipo, con una sintaxis semejante a la del lenguaje Java.

El lenguaje JavaScript se integra dentro del código HTML de las páginas web y actúa en cuanto un evento (un click en un botón, por ejemplo) es ejecutado.

El lenguaje que fue inventado por Brendan Eich en la empresa Netscape Communications, apareció por primera vez en el Netscape Navigator 2.0. Tradicionalmente, se venía utilizando en páginas web HTML, para realizar tareas y operaciones en el marco de la aplicación cliente / servidor.

Los autores inicialmente lo llamaron Mocha y, más tarde, LiveScript, pero fue rebautizado como JavaScript en un anuncio conjunto entre Sun Microsystems y Netscape, en 1995.

En 1997, los autores propusieron JavaScript para que fuera adoptado como estándar internacional the European Computer Manufacturers' Association. (ECMA). En junio de 1997, fue adoptado como un estándar ECMA, con el nombre de ECMAScript. Poco después también lo fue como un estándar ISO.

Para evitar estas incompatibilidades, el World Wide Web Consortium (W3C) diseñó el estándar Document Object Model (DOM), que incorporan los navegadores actuales.

- **Java**

Java (Gosling & Microsystems, 1995) es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrolladas por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre. Entre diciembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL,

de acuerdo con las especificaciones del Java Community Process, de tal forma que prácticamente todo el Java de Sun es ahora software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java aún no lo es).

JavaServer Faces (JSF)

La tecnología Java Server Faces (JSF) es un marco de trabajo de interfaces de usuario (Frameworks) del lado de servidor para aplicaciones Web basadas en tecnología Java, la cual pretende normalizar y estandarizar el desarrollo de aplicaciones web.
Los dos componentes principales son:

- Una API para representar componentes y manejar su estado, manejar eventos, validadores, conversores, internacionalización, etc.
- Etiquetas JSP para expresar los componentes y enlazarlos con objetos del lado servidor.

El framework JSF perfecciona el patrón de diseño MVC (Modelo Vista Controlador), permitiendo una separación clara entre el código de interfaz y el de lógica de negocio, esto hace que el modelo de trabajo sea basado en componentes UI (user interface), definidos por medio de etiquetas y XML.

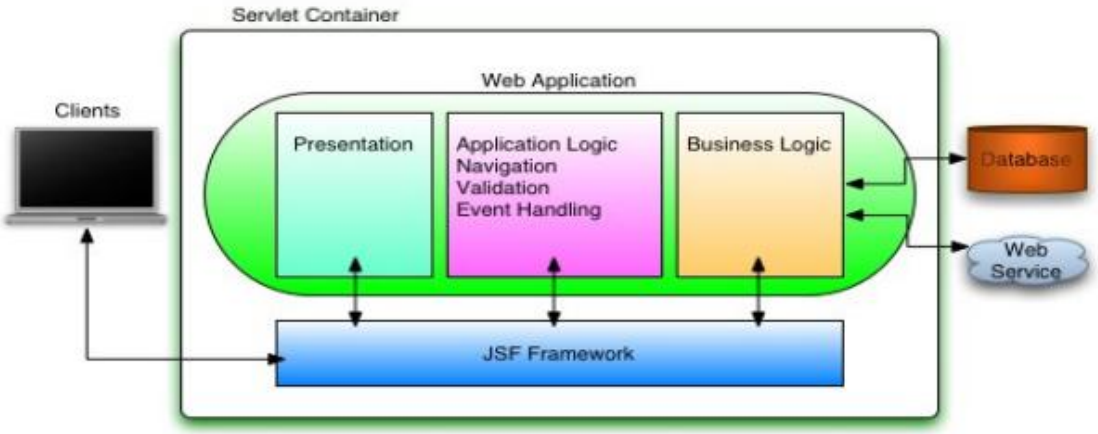

Figura 13: Estructura JSF

Asocia (de forma modular) cada componente gráfico con los datos (parte lógica del programa) y a la vez es muy flexible. Por ejemplo, nos permite crear nuestros propios componentes, o crear nuestras propias plantillas para pintar los componentes según nos convenga.

JSF usa JavaServer Pages (JSP) como la tecnología que permite hacer el despliegue de las páginas, pero también se puede acomodar a otras tecnologías como por ejemplo Facelets o incluso a varias, ya que JSF es un estándar Java, definido en un conjunto de especificaciones (Java Community Process, Especificacion JSR 314).

La implementación de referencia JSF, es el proyecto Mojarra de SUN aunque existen otras implementaciones: MyFaces, Rich Faces, ICEfaces y la utilizada en este proyecto PrimeFaces.

PrimeFaces

PrimeFaces es una librería JavaScript de componentes visuales open source y para su uso, únicamente se ha de adjuntar esta librería al proyecto y utilizar las anotaciones que ofrece ya que son beneficiosas y aporta simplicidad al programador.

Esta librería es desarrollada y mantenida por Prime Technology, una compañía Turca de IT especializada en consultoría ágil, JSF, Java EE y Outsourcing. El proyecto es liderado por Çağatay Çivici, un miembro del "JSF Expert Group".

Las principales características de Primefaces son:

- soporte nativo de Ajax, incluyendo Push/Comet.
- kit para crear aplicaciones web para móviles.
- es compatible con otras librerías de componentes, como JBoss RichFaces.
- uso de JavaScript no intrusivo (no aparece en línea dentro de los elementos, sino dentro de un bloque <script>).
- es un proyecto open source, activo y bastante estable entre versiones.

3.6.2 Lenguajes descartados

Para poder escoger los lenguajes usados para el desarrollo de la aplicación, se ha tenido que hacer una valoración de los diferentes lenguajes potenciales para la creación de la herramienta.

JAVA vs PHP

Características	**Java**	**PHP**
Multiplataforma	Si	Si
Programación Orientada a Objetos	Si	Si
Base de Datos	Conexión múltiple	Conexión múltiple
Licencia	GNU	PHP License (gratis)
Frameworks	JSP, JSF, Quickdb, etc.	CakePHP, Yii, Zend, etc.
Coste desarrollo	Mayor	Menor
Crecimiento del sistema	Más sencillo	Más dificultoso
Mantenimiento	Al mantener una estructura clara será más fácil su mantenimiento	Al no mantener una estructura su mantenimiento se hace más complicado
Integración externa	Mayor número de módulos que facilita la tarea a los programadores	Al ser menos estructurado es más complejo encontrar un módulo competo

		integrable con facilidad
Escalabilidad	Mejor	Empieza a mejorar con PHP5
Modulación	Sí, por ejemplo MVC (Model-Vista-Controlador)	Sí, pero no tan estructurada como Java, al tener todas las capas lógicas en el mismo archivo .php
Formación	Mayor coste al ser una tecnología más amplia y desarrollada	Menor coste
Seguridad	Mayor eficacia y transparencia hacia usuarios y programadores	Controla la seguridad de una forma más manual
Rendimiento	Menor	Mayor

Tabla 1: Java vs PHP

Viendo la Tabla 1 se puede observar que más o menos están igualados estos dos lenguajes de programación para desarrollar la aplicación, pero los puntos más fuertes por el que me he decantado por escoger Java han sido la mejor escalabilidad, su modularidad y sus Frameworks. También, cabe destacar que Advisor está hecho en Java y esto sería una pieza clave para la integración de la herramienta en ésta plataforma.

También a modo resumen se ha podido plasmar en una gráfica a modo estimativo la comparación entre las dos tecnologías tratadas.

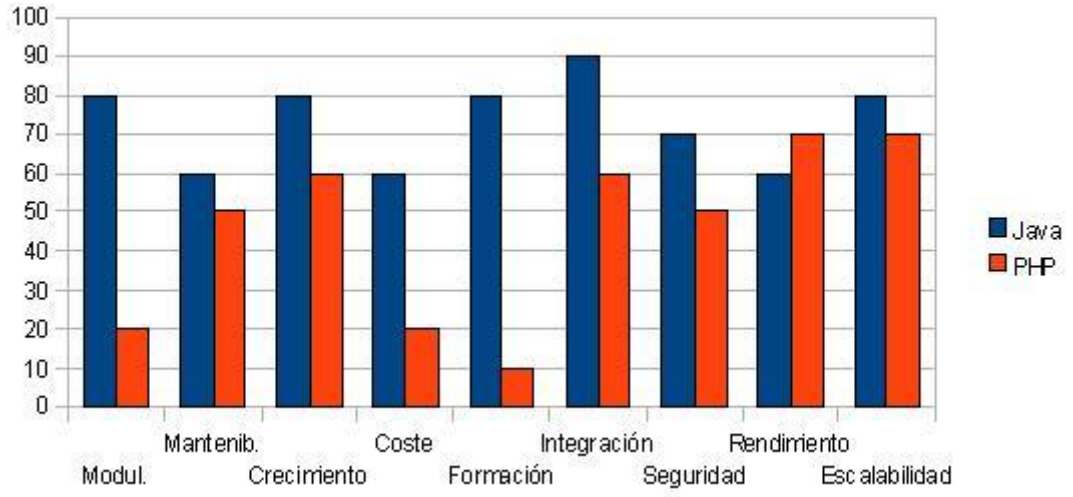

Tabla 2: Gráfica Java vs PHP

3.6.3 Detalles de la Base de Datos (BD)

En la BD de la aplicación, se ha utilizado las tablas ya existentes de la Advisor para así lograr una mejor integración. En la Tabla 3 se puede observar cómo queda.

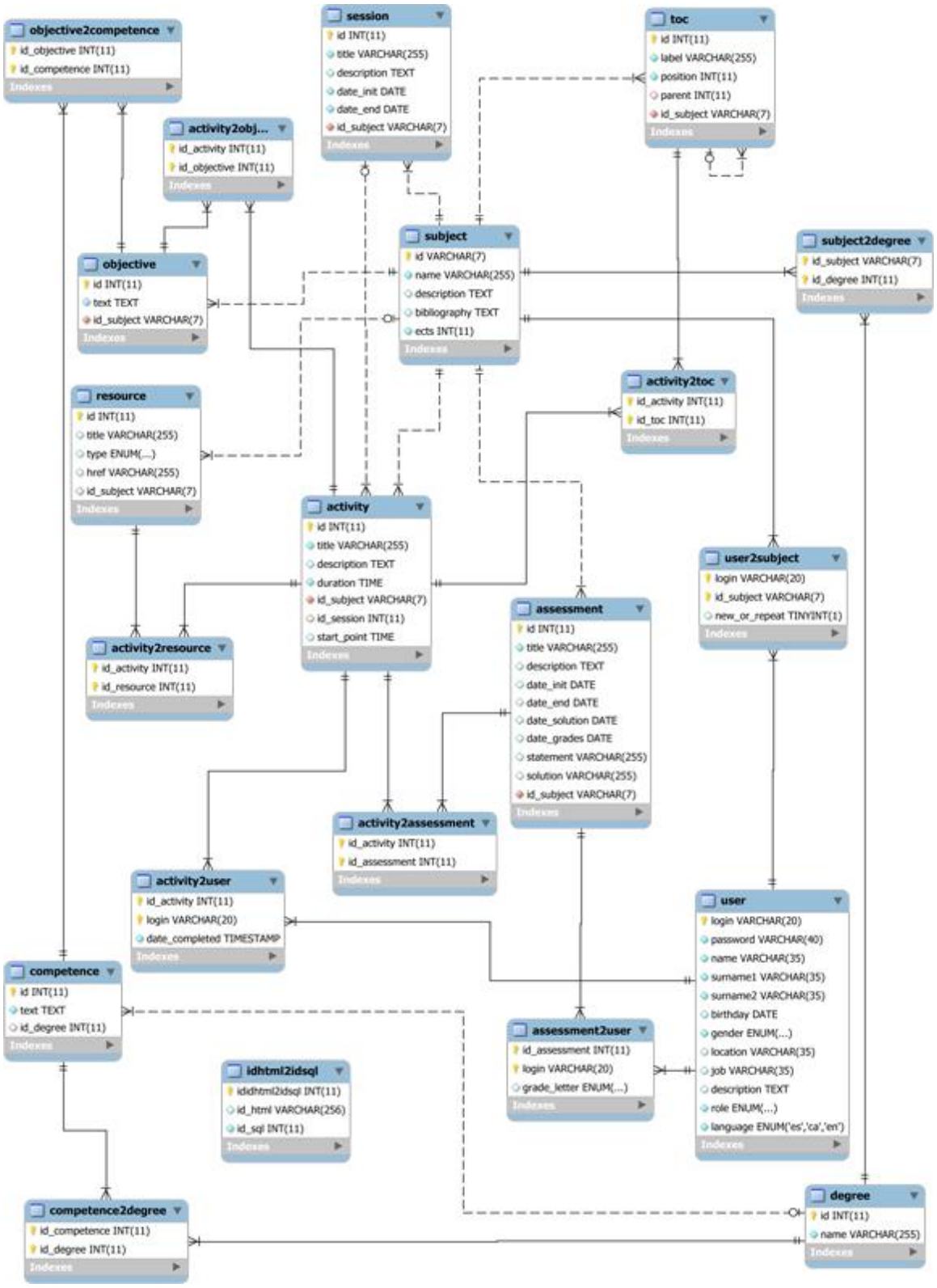

Tabla 3: Diagrama de la BD

3.6.4 Seguridad

En los últimos años, se han desarrollado multitud de aplicaciones web que acceden a bases de datos, estas aplicaciones, al estar disponibles a través de la web, son más propensas a recibir ataques que permitan acceder o modificar la base de datos, es por ello que el tema de seguridad es de especial interés.

Autenticación

Un sistema de autenticación es un modulo de seguridad para asegurar que el usuario que visita las páginas es quien dice ser. Así, conociendo a ese usuario, se le podrá dar acceso a más aspectos de la página que si fuese un usuario desconocido o anónimo. En la Figura 14 se muestra el proceso de autenticación, donde se pide un usuario y una contraseña para acceder a la aplicación de acceso restringido.

Una vez que se obtienen los datos de autenticación (usuario y contraseña escritos en la página inicial), se hace una comprobación de los mismos, y se redirecciona al navegador a la página de la aplicación restringida, en caso de que coincidan con los que están guardados en la base de datos. De lo contrario, se vuelve a mostrar la página de autenticación pidiendo de nuevo el usuario y la contraseña.

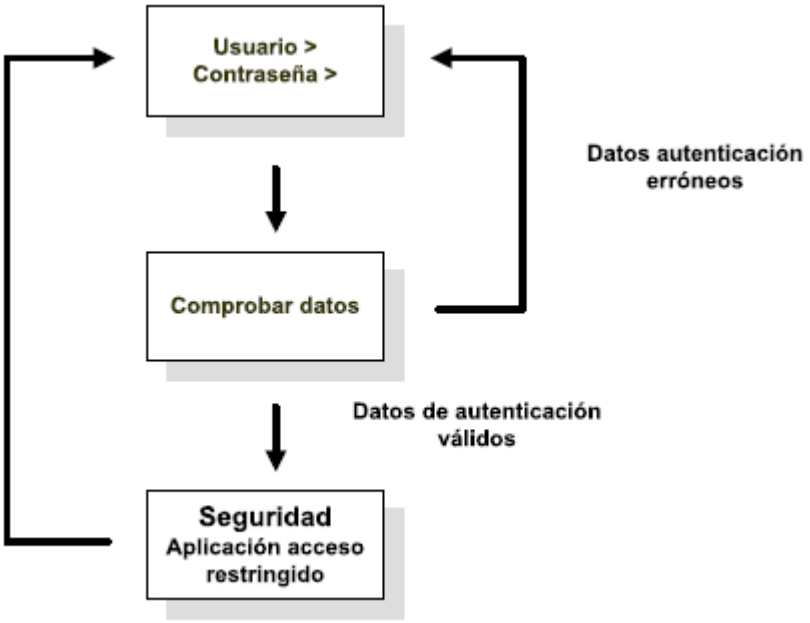

Figura 14: Autenticación del usuario

La aplicación de acceso prohibido, aparte de mostrar las funcionalidades que se quieren proteger con usuario y contraseña, debe realizar unas comprobaciones de seguridad para saber si se ha pasado con éxito el proceso de autenticación o si se está intentando acceder de manera no permitida a esa página.

Para la aplicación se ha utilizado un cifrado SHA-1 que es un estándar en la industria del algoritmo hash y es usado por muchas aplicaciones para guardar contraseñas.

Capítulo 4. Ejemplos de uso

En este capítulo se describen 2 ejemplos de uso de la herramienta desarrollada en este TFC. Éstos pueden servir como tutoriales.

4.1 Login

Al arrancar la aplicación web, la primera pantalla que se le muestra al usuario es la que se puede observar en la Figura 15. Para poder introducir el login y password del usuario clica sobre el dibujo de la llave, ésta nos llevará a la siguiente ventana, Figura 16, que será donde se pondrá el 'username' y el password' que cada usuario debería de poseer.

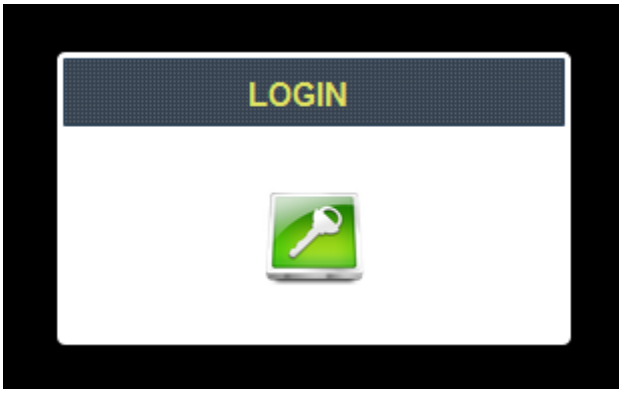

Figura 15: Login

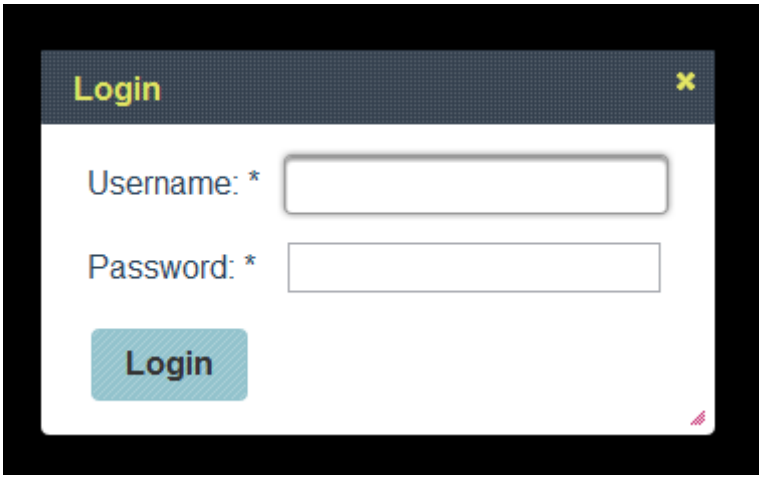

Figura 16: Login 2

4.2 Creación de una asignatura nueva

Una vez introducido correctamente el 'username' y el 'password' el usuario visionará un mensaje de bienvenida y seguidamente podrá escoger entre: (1) modificar una asignatura o (2) crear una nueva (ver Figura 17).

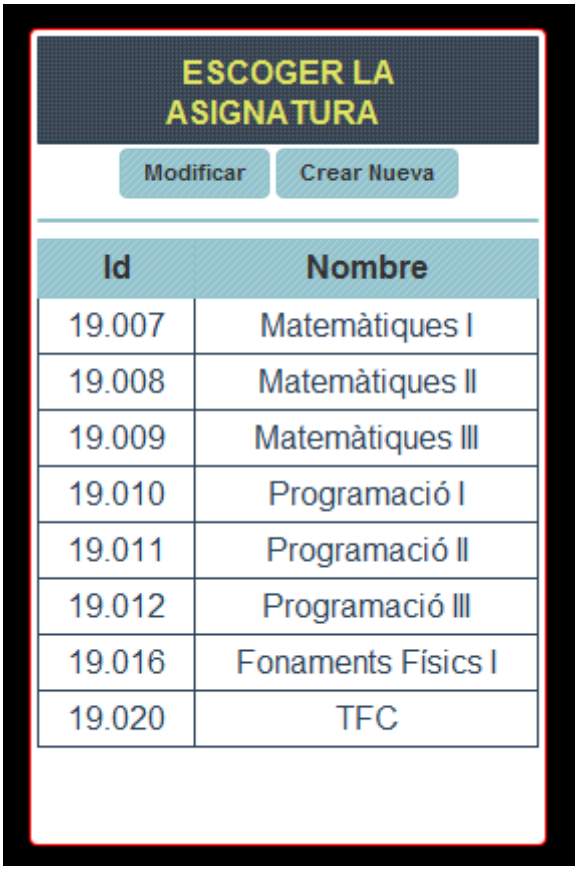

Figura 17: Escoger asignatura

En la Figura 18, se puede apreciar los campos de datos que debe ser introducidos: id de la asignatura, el nombre, la descripción, la bibliografía y el número de créditos. En la parte derecha, se puede observar que hay una tabla con las carreras de la universidad, el profesor deberá seleccionar la carrera a la que pertenece la asignatura nueva, como se muestra en la Figura 19 y pulsar el botón 'Aceptar'.

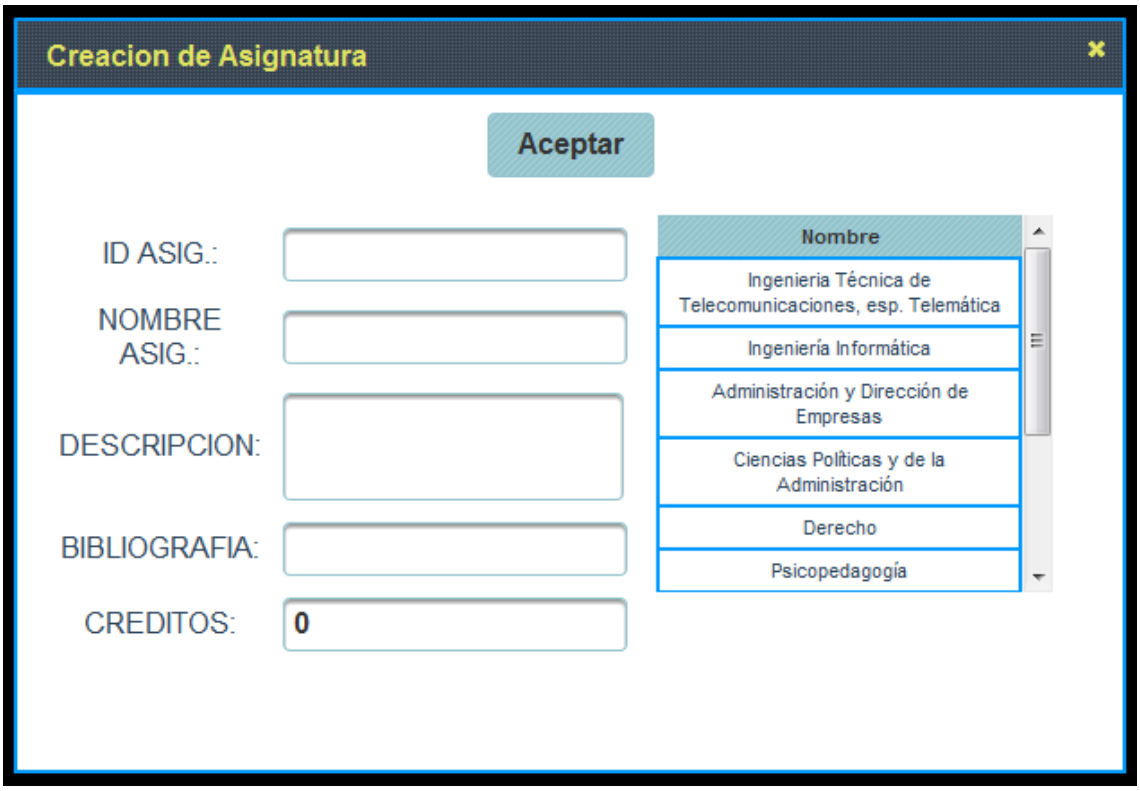

Figura 18: Introducción de datos para la asignatura nueva

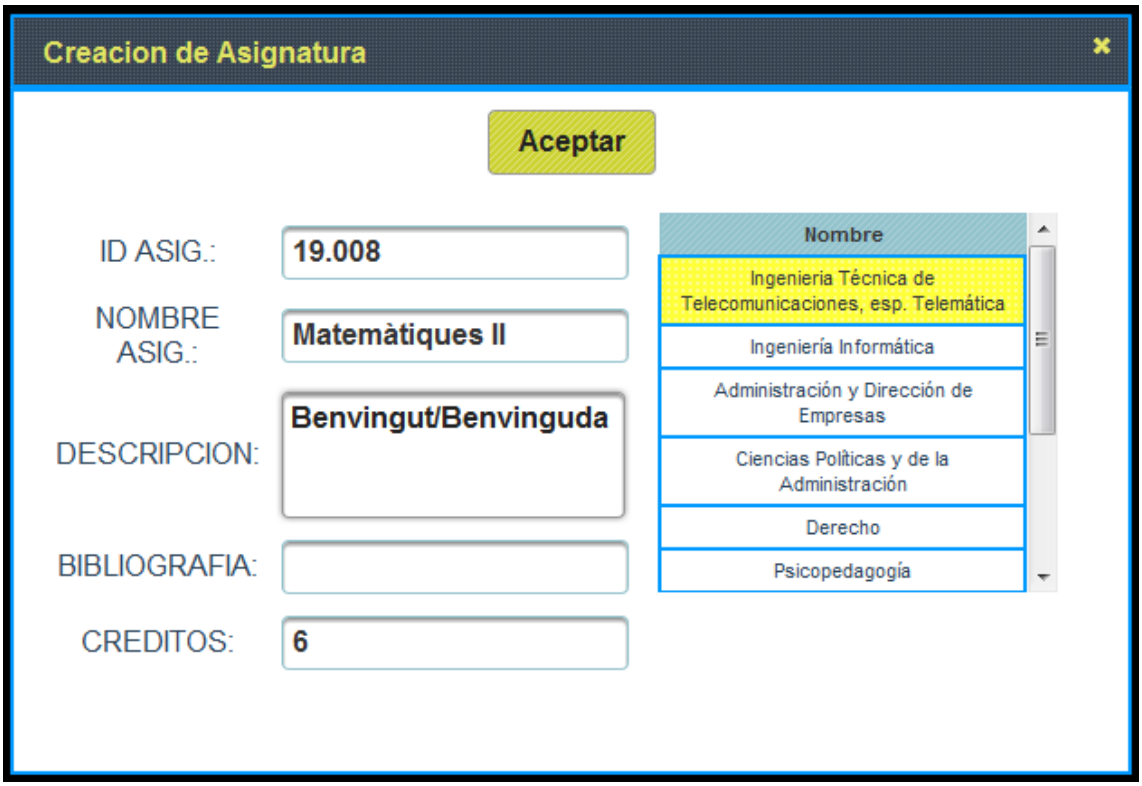

Figura 19: Introducción de datos para la asignatura nueva

Una vez, hecho esto aparecerá un mensaje informativo el cual indica que se procede a ir a la pantalla principal de la aplicación.

Una vez en ella (Figura 20), se procederá a insertar los la datos necesario y que el usuario crea necesario.

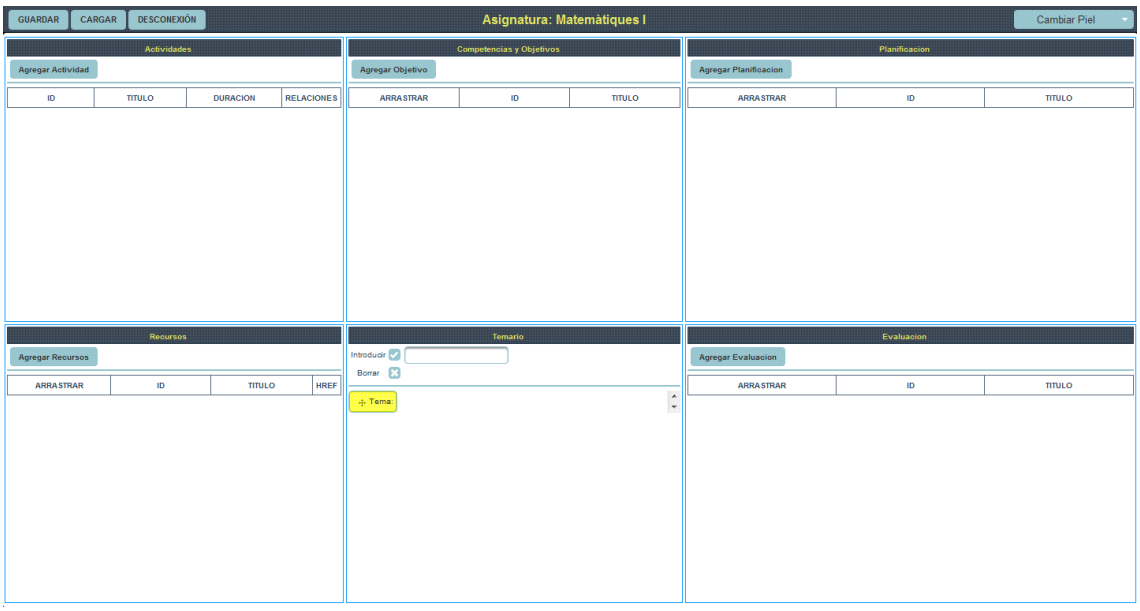

Figura 20: Pantalla principal

4.3 Modificar una asignatura

Para la modificación de una asignatura el usuario entrará con su usuario y password y una vez salga el mensaje de Bienvenida, se le mostrará en la pantalla las asignaturas relacionadas con el profesor (ver Figura 21). Se seleccionará la asignatura que se quiera modificar y se hará clic en 'Modificar'.

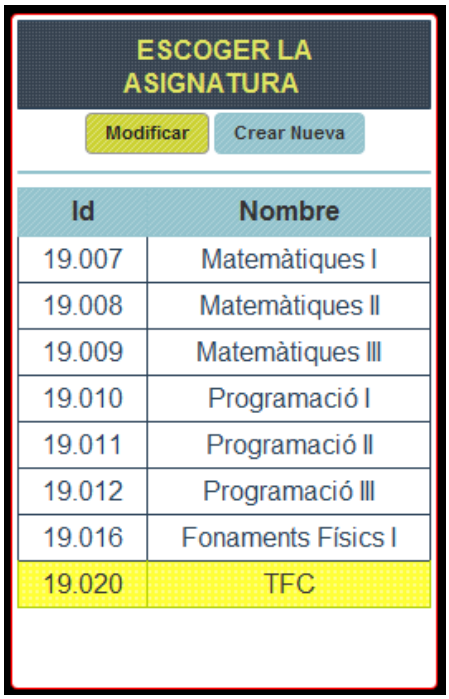

Figura 21: Selección de asignatura

Una vez hecho esto, aparecerá la página principal y para cargar los datos de la asignatura se hará clic en el botón de 'CARGAR' que está ubicado arriba a la izquierda. Entonces se procederá a la carga de la base de datos de la aplicación (ver Figura 22).

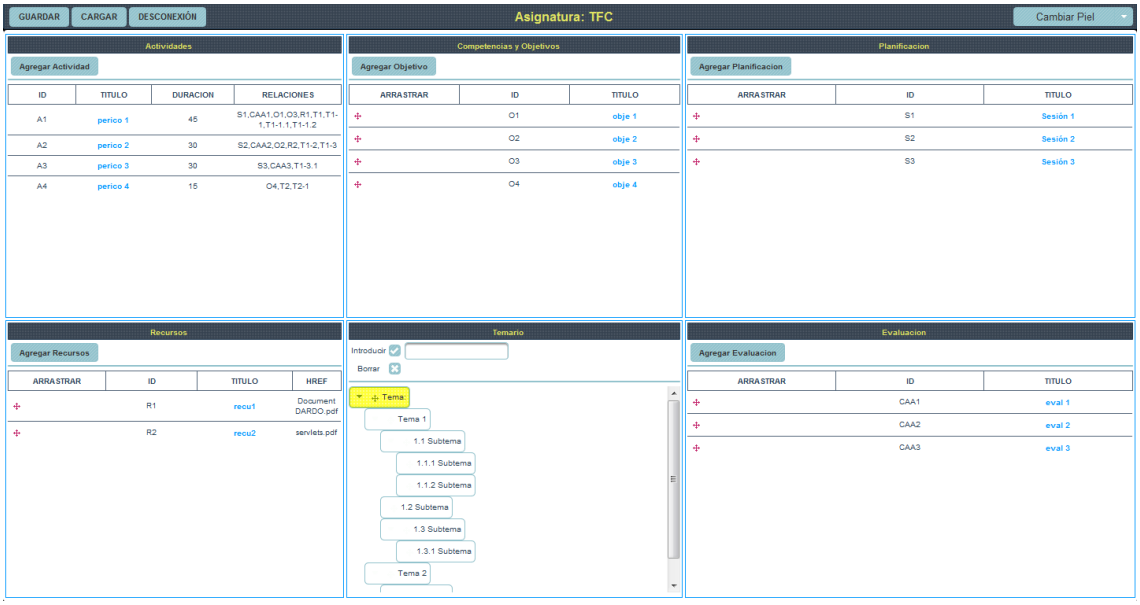

Figura 22: Asignatura cargada

Cuando los datos ya se hayan cargado, se podrán cambiar, ya sea modificando los existentes o creando nuevos.

4.4 Agregar/Modificar/Borrar Planificación

En el cuadro Planificación, si se pulsa 'Agregar Planificación' aparecerá una ventana en la que habrá que rellenar los siguientes datos: título, descripción, fecha inicio y fecha entrega. En la siguiente Figura 23 se muestra un ejemplo.

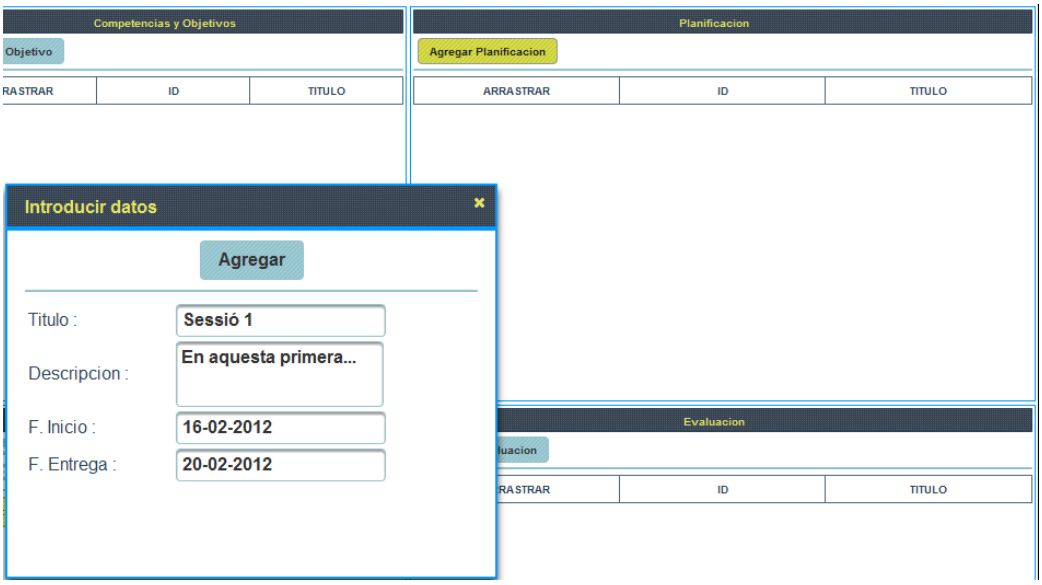

Figura 23: Agregar Sesión

Vemos que una vez pulsado el botón de Aceptar el cuadro de Planificación queda cómo en la Figura 24, donde hay tres columnas, la primera 'Arrastrar' (cuyo uso se explicará más adelante), la columna Id de la sesión y el título de la sesión. Se puede observar que el nombre de la sesión está en azul, pues bien, si hacemos clic en este enlace, aparecerá el un Dialogo como el de la Figura 25, en el cual se puede modificar todos los datos. Para guardar los cambios, damos en 'Aceptar' o si por el contrario se quiere eliminar la sesión seleccionada hacemos clic en el botón 'Borrar'. El botón 'Cancelar' sale del Dialogo sin hacer ninguna acción.

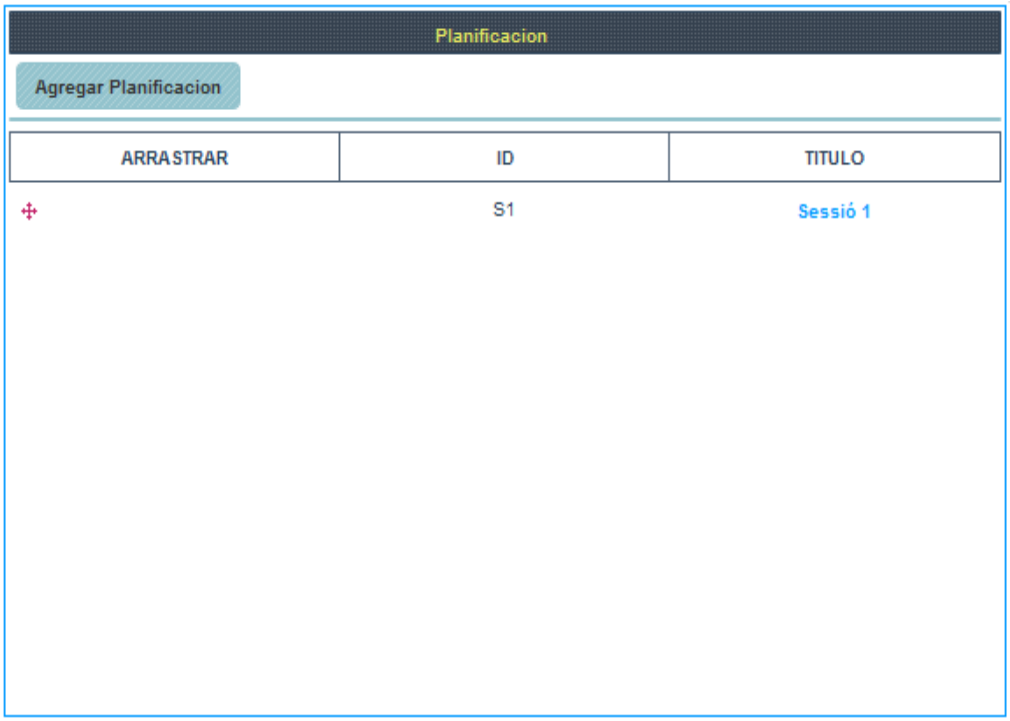

Figura 24: Sesión introducida

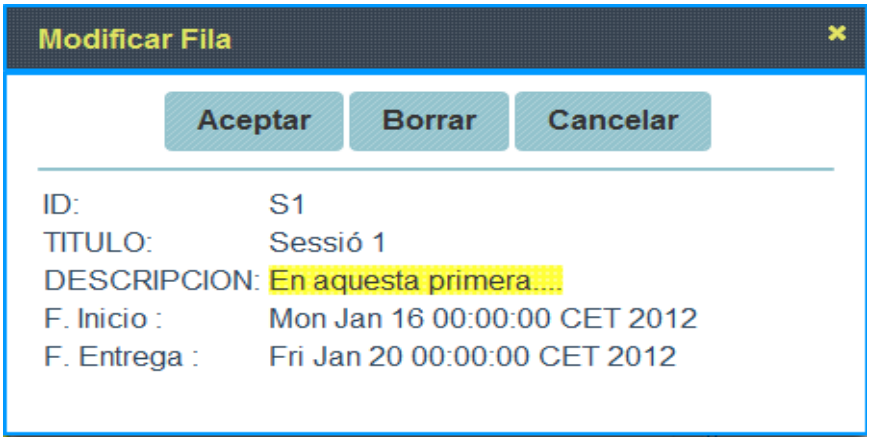

Figura 25: Modificar/Borrar Sesión

4.5 Agregar/Modificar/Borrar Evaluación

En el cuadro Evaluación, el procedimiento es similar al visto ya para la Planificación. Haciendo clic en el botón 'Agregar Evaluación', aparecerá un Dialogo parecido al de Planificación pero adaptado para los datos requeridos de Evaluación, se inserta los datos y agregamos la evaluación. Una vez esto, se actualizará la tabla de Evaluación en la cual también aparecerá las 3 columnas dichas anteriormente y en la que también, si hacemos clic en el Título se podrá modificar/borrar la evaluación seleccionada (Figura 26, Figura 27 y Figura 28).

Figura 26: Agregar Evaluación

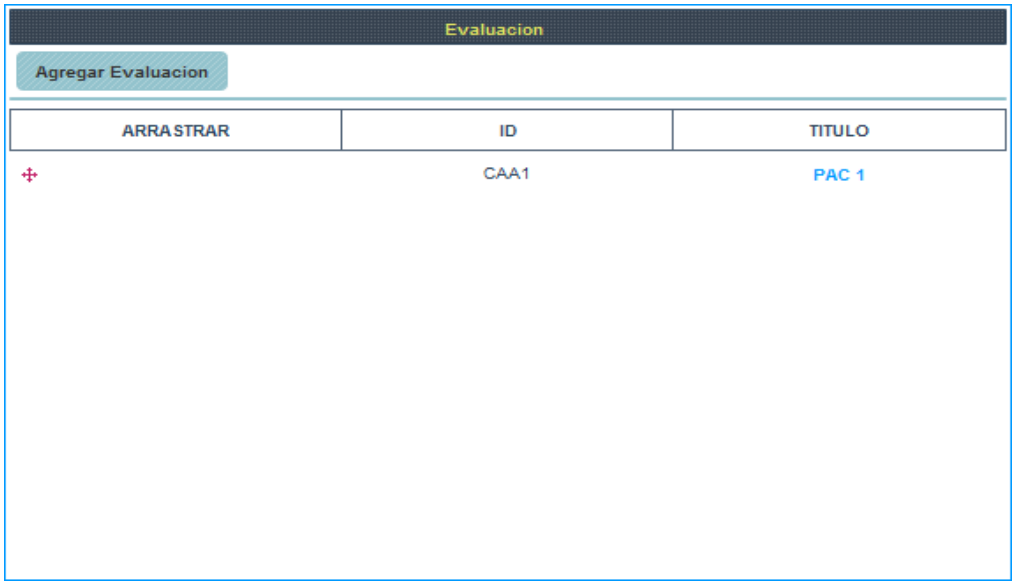

Figura 27: Evaluación introducida

Figura 28: Modificar/Borrar Evaluación

4.6 Agregar /Borrar Temario

En el área de Temario se puede observar en la Figura 29 como es un poco diferente a las demás secciones. Hay un campo en el cual se puede introducir texto y dos botones, el 'Introducir' y el 'Borrar'. Además, por defecto, está seleccionado el nodo 'Tema'.

Para introducir un título, lo que se ha de hacer es seleccionar el nodo 'padre', escribir el titulo que se desea y pulsar 'Introducir'. Así, el nodo introducido quedará como hijo del nodo padre (ver la Figura 30). Así mismo, para borrar algún tema se selecciona el nodo que se quiera borrar y hacemos clic en el botón Borrar, como se muestra en la Figura 31.

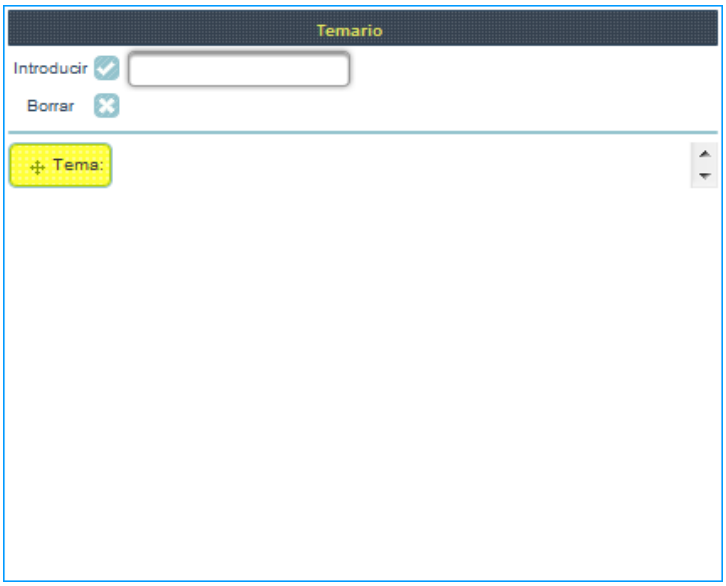

Figura 29: Cuadro de Temario

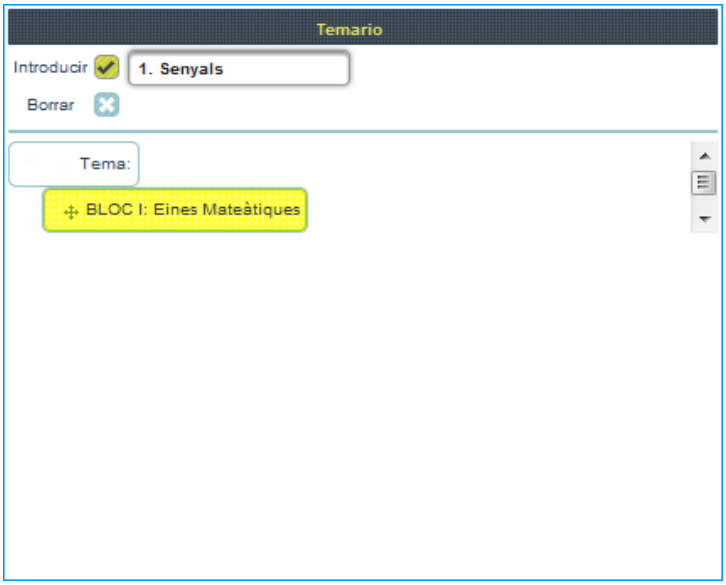

Figura 30: Insertar Tema

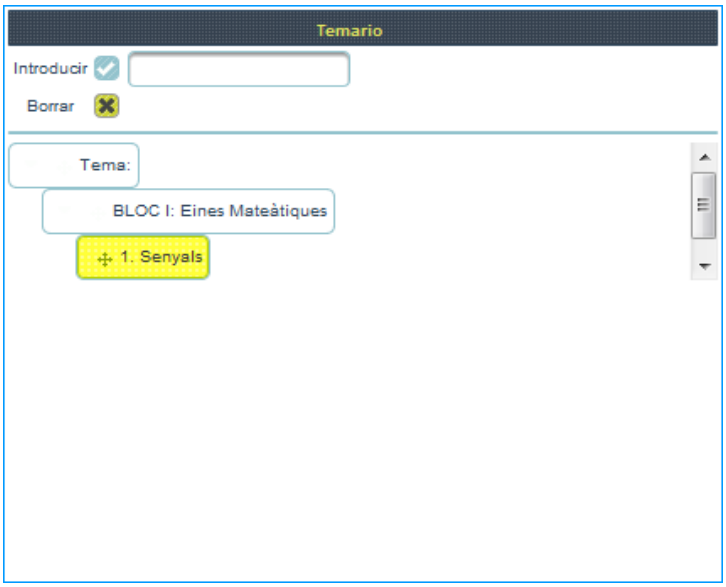

Figura 31: Borrar Tema

4.7 Agregar/Modificar/Borrar Recursos

En el área de Recursos, semblante al de Planificación y Evaluación, con la diferencia en las columnas, en la que se ha añadido la de href. Esta columna marcará el tipo de fichero que se ha adjuntado, .pdf, .doc, etc.

Para agregar un Recurso hacemos clic al botón 'Agregar Recursos', nos aparecerá un nuevo Dialogo, como en la Figura 32, habrá para insertar el título del recurso y luego se podrá seleccionar el archivo a adjuntar. Con el botón 'Choose' escogeremos el archivo que deseemos y nos aparecerá en el cuadro como en la Figura

33, una vez hecho esto, sólo nos quedará subirlo al servidor mediante el botón 'Upload' que hay tanto arriba del cuadro, como a la derecha del archivo seleccionado.

También se puede observar que si queremos eliminar el archivo antes de subirlo al servidor, tenemos el botón de la derecha con el símbolo de Cancelar, con esté cancelaremos este archivo para la subida al servidor.

Ahora sólo queda pulsar el botón 'Agregar' y nos aparecerá el recurso en la tabla de Recursos como en la Figura 34.

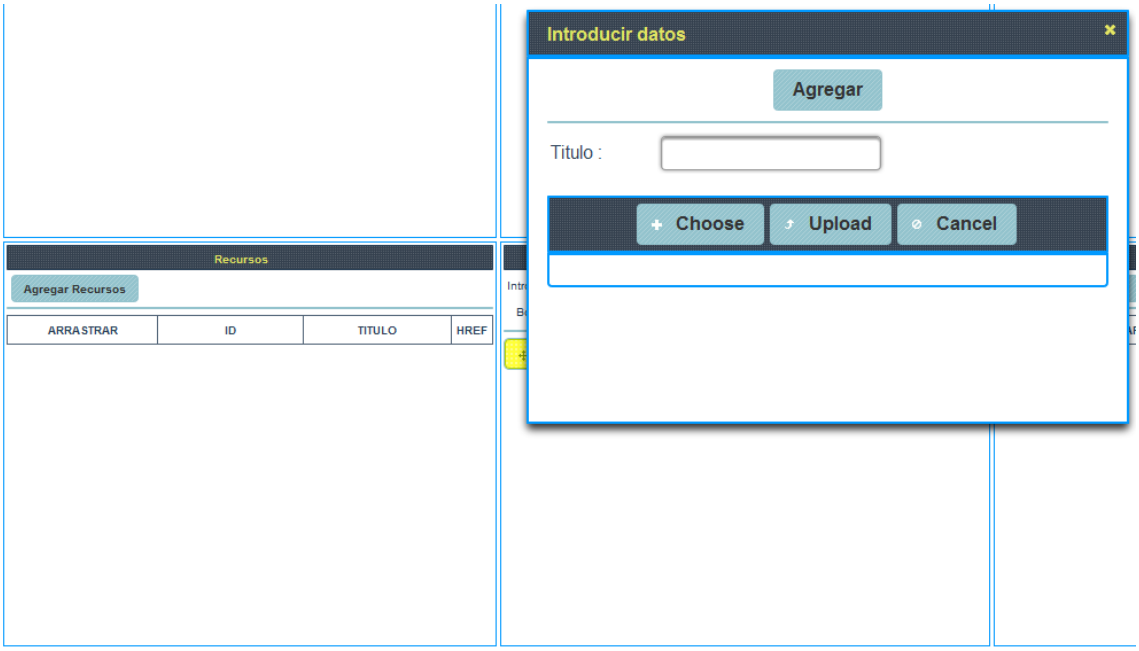

Figura 32: Introducir Recurso

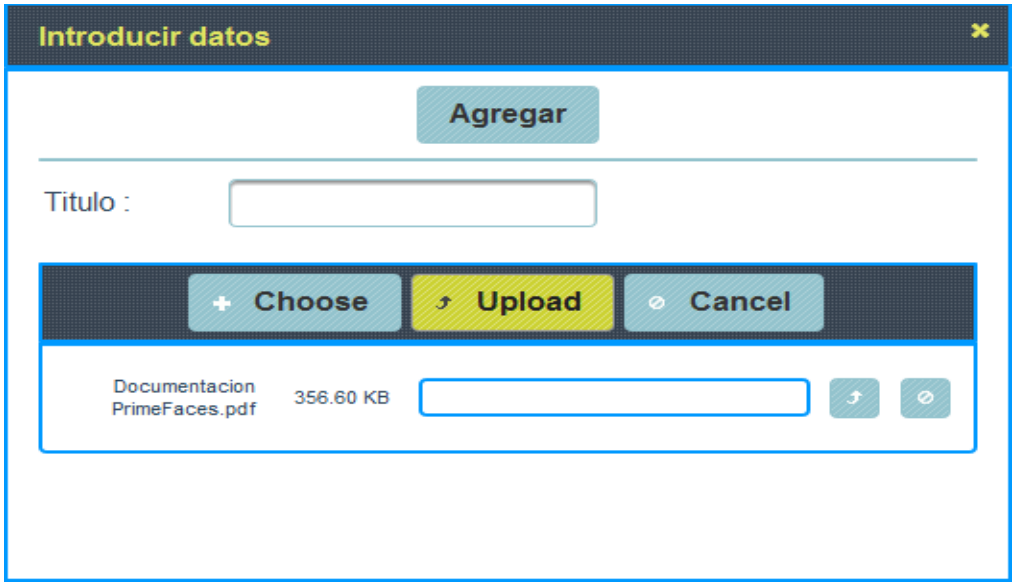

Figura 33: Subir Fichero seleccionado

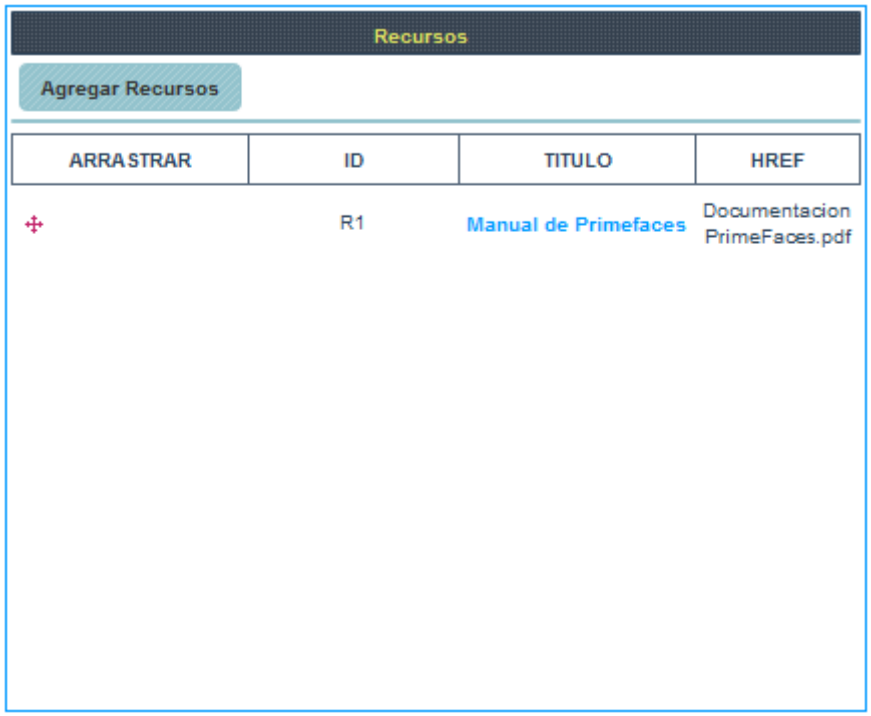

Figura 34: Recurso introducido

4.8 Agregar/Modificar/Borrar Competencias y Objetivos

El cuadro de Competencias y Objetivos a simple vista es similar a los otros pero veremos que para introducir y borrar también es diferente.

Para introducir hay que hacer clic en el botón 'Agregar Objetivo', se abrirá un Dialogo como el de la Figura 35, allí tendremos que introducir el título del objetivo y las competencias de titulación con las que está relacionada. Para seleccionar las competencias vinculadas con el objetivo se tiene que seleccionar con el ratón mediante un clic, si se quiere seleccionar más de una competencia hay que mantener pulsado la tecla 'Control' (crtl) y mientras tanto hacer clic a las competencias deseadas, una a una. Hay que añadir que para deseleccionar, también se puede hacer manteniendo la tecla 'Control' y haciendo clic en las competencias no deseadas.

Para modificar/borrar un objetivo lo seleccionamos como en los casos anteriores, haciendo clic el titulo, y vemos como en la Figura 36 que se nos aparece un Dialogo parecido al de agregar. En este Dialogo se puede modificar el Titulo y las competencias. Para modificar las competencias simplemente habrá que volver a seleccionar las que se quiera mediante la tecla 'Control' y haciendo clic con el ratón. Una vez hecho esto, procedemos a aceptar el cambio.

Para borrar el objetivo solamente hay que hace clic al botón de 'Borrar'.

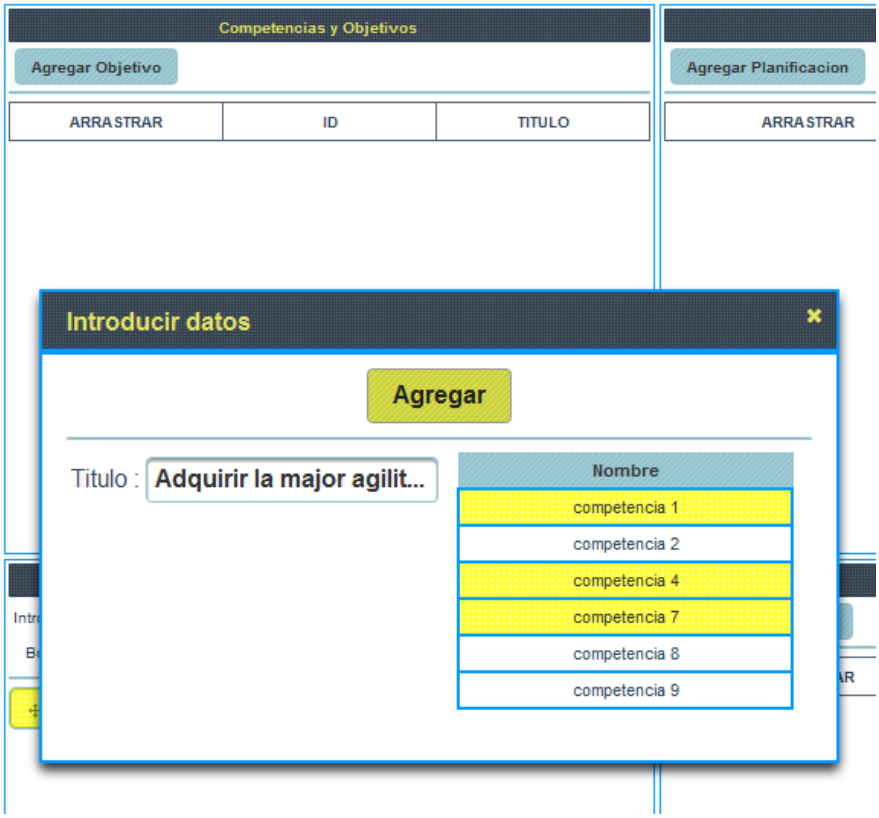

Figura 35: Introducir Objetivo

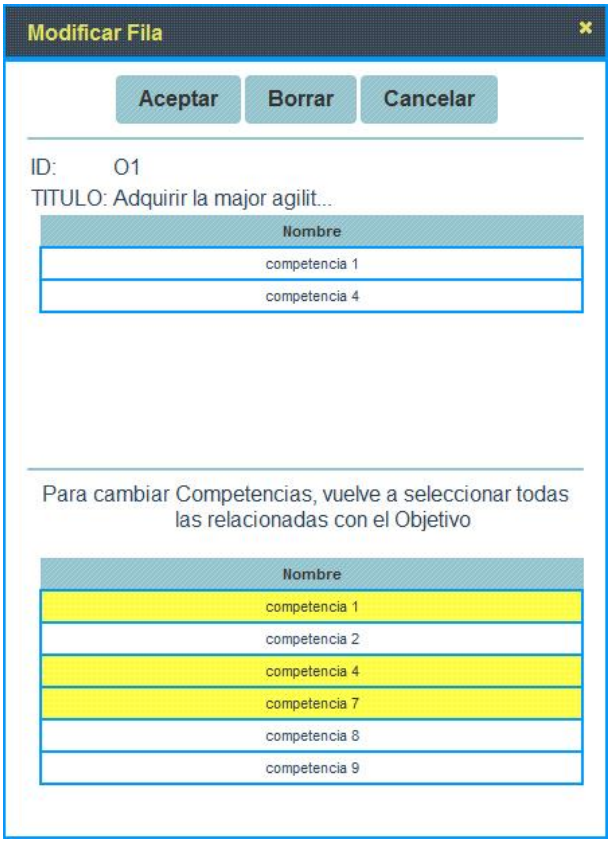

Figura 36: Modificar/borrar Objetivo

4.9 Agregar/Modificar/Borrar Actividades

En el cuadro de Actividades para añadir actividades (ver Figura 37) hay que hacer clic al botón 'Agregar Actividad', con el cual nos aparecerá un Diálogo en el que podremos insertar el Título, Descripción y la Duración en minutos de la Actividad mediante un spinner.

Se puede observar que este cuadro tiene 4 columnas. La última columna nos indica las relaciones que hay de los demás elementos con esa actividad.

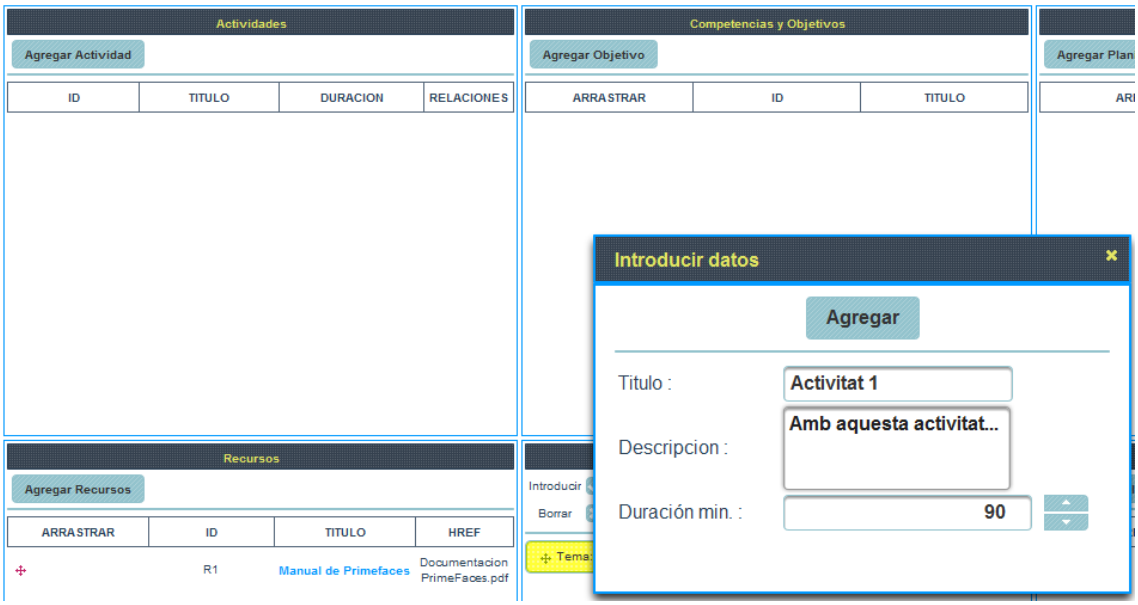

Figura 37: Agregar Actividad

Si lo que se desea es modificar algún dato de la Actividad procederemos a seleccionar la actividad, se nos abrirá de nuevo otro Dialogo el cual nos mostrará la información de esa actividad. (Figura 38). Allí se podrá cambiar los datos pertinentes y además las relaciones que deseemos. Para ello, como en el mismo caso de Competencias y Objetivo, seleccionaremos las relaciones que queremos que se guarden para esa actividad y las que queremos que se borren no las seleccionaremos, aceptamos el cambio y la actividad quedará guardada y modificada a nuestro gusto.

Para borrar un actividad, lo mismo que los casos anteriores, tan simple como seleccionar la Actividad deseada y hacer clic en el botón 'Borrar'.

Figura 38: Modificar/Borrar Actividad

4.10 Agregar relaciones en Actividades

Para agregar las relaciones es muy sencillo, simplemente hay que arrastrar la crucecita roja que hay en cada fila de las tablas hacia la actividad que se quiere relacionar, como en la Figura 39. Para arrastrar hay que hacer clic en la cruz y mantener pulsado mientras arrastras la cruz hasta la actividad deseada. El lugar donde hay que soltar la cruz es el que se enciende el foco amarillo de la columna Relaciones de Actividades. El resultado se puede ver en la Figura 40.

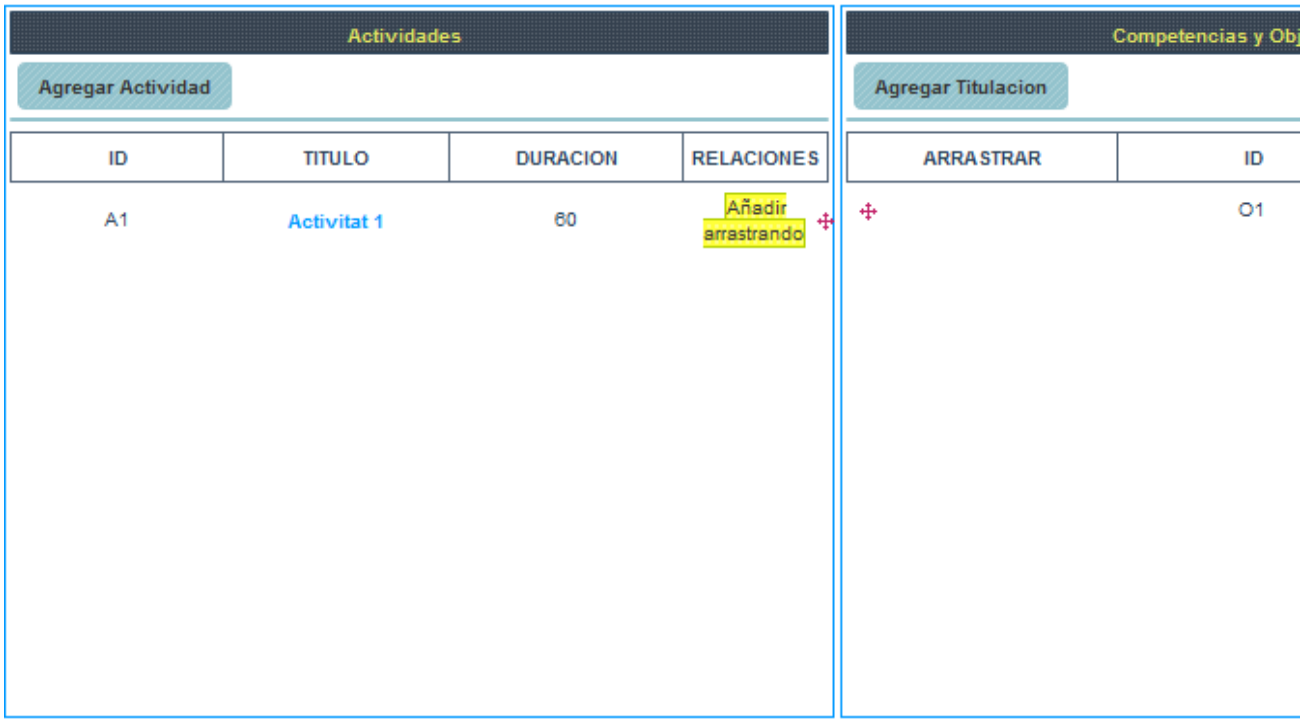

Figura 39: Añadir relación

Figura 40: Relación insertada

4.11 Guardar asignatura y Finalizar Sesión

Por último, para guardar la asignatura, se procederá a hacer clic en el botón de Guardar, éste hará los procesos pertinentes para hacer el archivo comprimido necesario y nos redireccionará automáticamente a la última página, en la cual podremos: (1) o finalizar la sesión, o (2) crear/modificar otra asignatura (ver Figura 41).

Figura 41: Finalizar o Seguir

Capítulo 4. Conclusiones y líneas futuras

Tras finalizar este trabajo, se dispone de la aplicación vía web que permite la creación y modificación de asignaturas de la facultad.

Como se ha visto, se ha realizado un análisis del desarrollo para lograr su máximo aprovechamiento. Fruto de este análisis se sintetizan los puntos clave, las dificultades, conocimientos adquiridos y la valoración de la experiencia.

Al comienzo de este trabajo se plantearon una serie de objetivos, como por ejemplo, que el sistema fuera fácilmente ampliable, de los cuales, la gran mayoría se han satisfecho, dejando algunas líneas abiertas para futuras implementaciones.

4.1 Conclusiones personales

Estos últimos 4-5 meses, que ha sido más o menos lo que me ha llevado de tiempo este proyecto, solo veo cosas positivas. En primer lugar decir, que me ha resultado una experiencia muy gratificante de la que he aprendido muchísimo. Muchas cosas que desconocía por completo y que después de todo este tiempo han despertado en mi mucha curiosidad, como son las tecnologías web y su problemática.

A día de hoy, las aplicaciones web tienen una importancia especial y es por ello por lo que pienso que me puede ayudar mucho durante mi vida profesional. Sin más, decir que han sido unos meses bastantes aprovechados y de los que estoy muy contento por todo los conocimientos que he adquirido.

4.2 Dificultades en el desarrollo

Algunas de las dificultades encontradas en el desarrollo de esta aplicación han sido:

- Elección del lenguaje de programación. Debido a mi desconocimiento de los numerosos lenguajes existentes para programar aplicaciones web, esta ha sido una de las principales dificultades encontradas. Al final, el lenguaje utilizado para programar la aplicación fue Java con el framework JSF. Los motivos principales de esta elección fueron que el lenguaje Java ya lo conocía por haberlo estudiado en la Universidad y además aportaba una mayor integración con la plataforma Advisor, ya que ésta está desarrollada en Java. Sin embargo, JSF me aventuré a utilizarlo por la buenas críticas hacía este framework que había leído por internet.

- Problemas con la base de datos. Otro punto del desarrollo que también planteó dificultad, fue que la aplicación tuviera que guardar los datos en una base de datos. Por suerte había estudiado SQL hace años aunque nunca lo he utilizado, así que he tenido que reciclar todo lo aprendido.

- Diseño de la web. El gran escollo a superar a sido el diseño de la web e intentar diseñarlo simple e intuitivo. Esto ha supuesto tener que luchar con los escollos de los lenguajes, ya que todos tienen limitaciones tecnológicas y según qué cosa se quiera hacer pues no es posible su implementación. También, es cierto que tampoco soy diseñador web y esto dificulta a la hora de hacer una aplicación bonita a los ojos.

- Problemas en la depuración. Los programas informáticos son a menudo imperfectos y, por lo tanto, el código puede contener diversos tipos de errores. La única forma de detectar los errores lógicos es probar el programa, ya sea manual o automáticamente, y comprobar que el resultado es el esperado. Las pruebas deben ser una parte integrante del proceso de desarrollo del software. Desgraciadamente, aunque las pruebas pueden indicar que el resultado de un programa es incorrecto, normalmente no proporcionarán ninguna pista acerca de que parte del código ha causado realmente el problema. Es aquí donde cobra sentido el proceso de depuración. Este escollo ha llevado mucho tiempo en corregir los errores, tener que analizar el programa y localizar el mal funcionamiento de éste. Es por esto que cobra especial importancia el diseñar una estructura de programa buena y que logre y permita realizar esta ardua pero importante labor de una manera sencilla.

4.3 Conocimientos

Evidentemente en una carrera como Ingeniería Técnica de Telecomunicaciones, no es posible infundir todas las enseñanzas que podría necesitar el alumno en su carrera profesional principalmente por falta de tiempo y de cambio continuo en la tecnología.

Hoy en día, el tema de las aplicaciones web, tecnologías web, aplicaciones cliente / servidor, etc. están siendo muy usadas por mucha gente a la hora de realizar programas, es por ello que me he aventurado a realizar este proyecto y lograr superar esta meta. Durante la realización de este proyecto me encontrado con numerosos problemas, muchos de los cuales nunca los había estudiado durante los años de docencia. Por ejemplo, el cómo diseñar una aplicación vía web para facilitar el uso de la misma, o el conocimiento de lenguajes que se utilizan para desarrollar una aplicación web, como pueden ser, JSF, CSS, JavaScript o HTML.

Actualmente, las aplicaciones y tecnologías web son un área en auge debido al impulso que están sufriendo desde el boom de la web 2.0, con la aparición de servicios sociales on-line. Sin embargo, existe un vacío de asignaturas durante la carrera que traten el desarrollo y problemática de este tipo de aplicaciones.

4.4 Líneas futuras

Al finalizar este trabajo quedan abiertas varias líneas de desarrollo entre las que se pueden destacar las siguientes:

- Mejorar posibles problemas con la codificación ISO-8859-1 y UTF-8. No he visto el caso pero dependiendo del navegador utilizado podría haber problemas con los acentos.
- Mejorar la selección de una asignatura, para esto ahora mismo se tiene que clicar en los espacios en blanco de la celda, esto es debido a las limitaciones de la tecnología PrimeFaces utilizada.
- Cargar los datos de la asignatura de golpe, es decir, no tener que dar aceptar para cada área de la asignatura. Debido también por limitaciones de la tecnología JSF usada.
- Verificar que los redimensionamientos automáticos vayan correctamente. Usando Chrome e Internet Explorer no se ha dado el caso pero podría ocurrir con cualquier otro navegador.
- Mantenimiento y mejora del reciclado del código
- Poder seleccionar varias relaciones o competencias sin tener que mantener pulsado el botón de 'Control', esto es debido también a una limitación de la tecnología PrimeFaces usada.
- Poder cargar una asignatura a partir de un *.zip en vez de como hace ahora mismo de la base de datos.
- Poder cargar las secciones de una asignatura que el profesor escogiera, por ejemplo, sólo Actividades, Temario y Planificaciones.
- Establecer algún tipo de autoayuda en la aplicación

ANEXOS

Jerarquía de ficheros

Carpeta *tema*. Esta es la carpeta principal o raíz de la aplicación.

Carpetas *build y dist*. Todos los objetos creados a partir de NetBeans, los comandos crear y ejecutar se crean en estas carpetas.

Carpeta *nbproject*. Contiene la mayoría de los metadatos de los proyectos de NetBeans, incluidos los recursos que son llamados por los principales script de NetBeans.

Carpeta *src.* Contiene todas las clases para la aplicación.

Carpeta *web.* Contiene las páginas web xhtml, archivos css, templates…

Carpeta *css.* Contiene los archivos css y temas para la aplicación.

Carpeta *jquery.* Contiene la librería jquery.

Carpeta *META-INF.* Contiene información de cuál es la carpeta raíz.

Carpeta PrimeFaces. Contiene la librería de PrimeFaces.

Carpeta templates. Contiene la plantilla para la aplicación.

Carpeta Upload. Contiene los archivos comprimidos de las asignaturas creadas.

Carpeta WEB-INF. Contiene elementos de configuración del WAR como: Página de Inicio, Ubicación ("Mapeo") de Servlets y parámetros para componentes.

Archivo index.html. Página web de inicio de la aplicación.

Archivo crear_modificar_asig.html. Página web donde se escoge si se tiene que crear o modificar una asignatura.
Archivo crear.html. Página web donde está todo el peso de la aplicación, creación/modificación de los datos.

Archivo guardado.html. Página web final de la aplicación.

A continuación en la Figura 42 y Figura 43 Se puede observar la estructura de las dos carpetas más importantes de este proyecto.

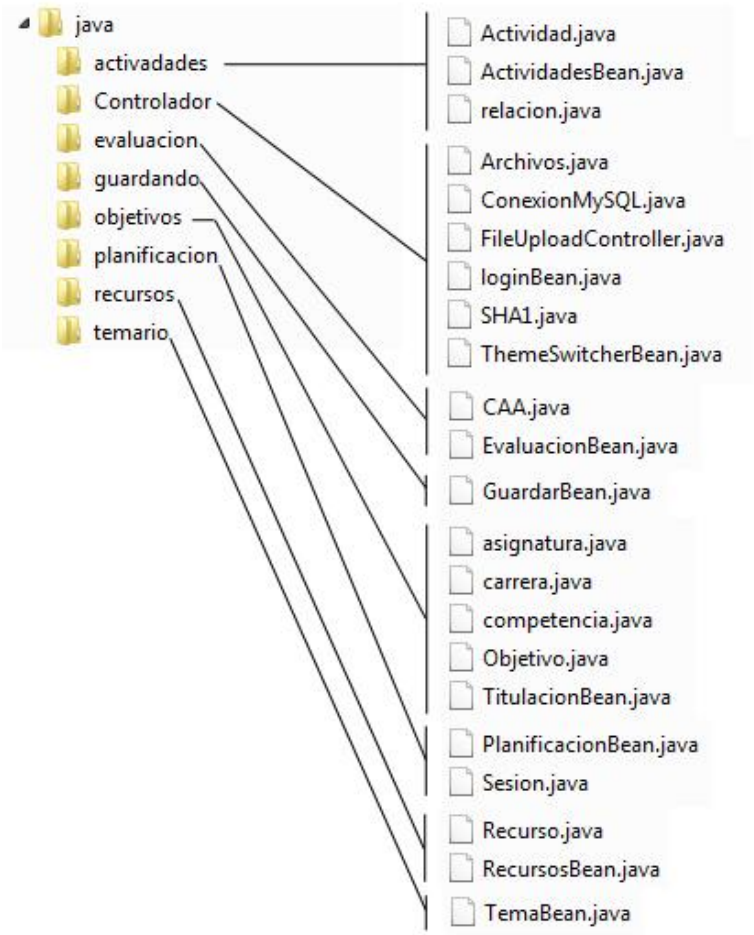

Figura 42: Jerarquía src/java

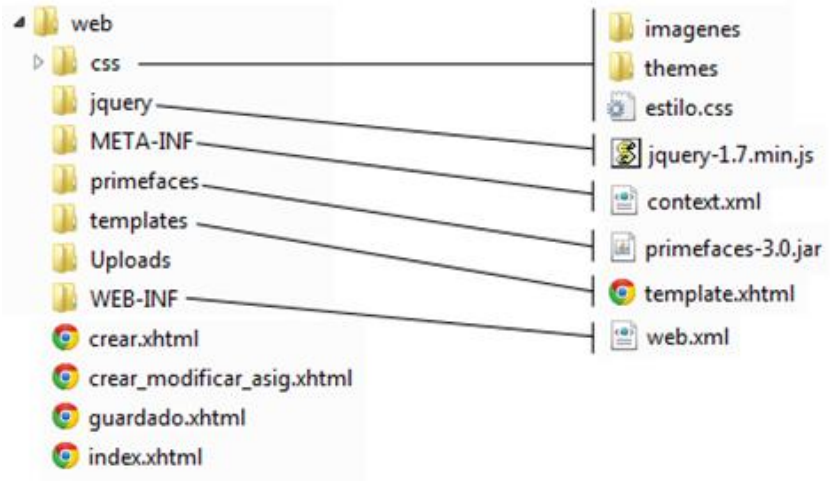

Figura 43: Jerarquía /web/

Bibliografía

Agile. (s.f.). Obtenido de http://www.agilemanifesto.org

Álvarez, J. V. (s.f.). *Uso de estándares e-learning en espacios educativos*. Obtenido de http://huespedes.cica.es/huespedes/revfuentes/campo_02.htm

Beck, K. (1999). *Extreme Programming Explained: Embrace Change.*

Cejudo., S. D. (2003). *E-Learning. Análisis de plataformas gratuitas.*

Consortium, W. W. (2004). *http://www.w3c.org/*.

Desventajas de e-Learning. (s.f.). Obtenido de http://www.sistemaselearning.com/contenidos/Desventajas-del-e-learning.asp

Eich, B. (2001). *JavaScript Bible.*

El estado del arte de la formación. (2008). Obtenido de http://www.elogos.es/Documents/estudios/panel-latam-2010.pdf

Galusha, J. M. (1997). *Barriers to Learning in Distance Education.*

Gomma, H. (1995). *A software design method for real-time systems. Communications of the ACM, volumen 17.*

Gosling, J., & Microsystems, S. (1995). *www.java.com*.

Grau, M. I., & Segura, X. S. (2003). *Desarrollo Orientado a Objetos con UML*.

Griffiths, D., Blat, J., Garcia, R., & Sayago, S. (2004). *La aportación de IMS Learning Design a la creación de recursos pedagógicos reutilizables.* Obtenido de http://www.um.es/ead/red/M5/griffiths16.pdf

Gros, B. y. (1997). *Diseños y Programas Educativos: Pautas Pedagógicas para la Elaboración e Software.* Barcelona: Ariel Educación.

IEEE Learning Technology Standards committee. (s.f.). Obtenido de http://ltsc.ieee.org

Inconvenientes y ventajas del e-Learning. (s.f.). Obtenido de http://www.educaweb.com/esp/servicios/monografico/elearning/opinion2.asp

Java Community Process, Especificacion JSR 314. (s.f.). Obtenido de http://www.jcp.org/en/jsr/detail?id=314

Juan Silvio Cabrera Albert, G. F. (2006). *El estudio de los estilos de aprendizaje desde una perspectiva Vigotskiana: una aproximación conceptual.* Obtenido de www.campus-oei.org/revista/deloslectores/1090Cabrera.pdf

Khan, B. (2005). In *Learning features in an open, flexible, and distributed environment.* (pp. 137-153). AACE Journal.

Kling, H. a. (2000). Students' distress with a web-based distance education course.
Larman, C. (1999). *UML y Patrones.*

Morgado, E. M. (2007). *GESTIÓN DEL CONOCIMIENTO EN SISTEMAS E-LEARNING, BASADOS EN OBJETOS DE APRENDIZAJE, CUALITATIVA Y PEDAGÓGICAMENTE DEFINIDOS.* Salamanca.

National Learning Infrastructure Initiative. (s.f.). Obtenido de http://www.educause.edu/live/live034/

Nonaka, H. T. (1986). The New Product Development Game. *Harvard Business Review*
Perea, C. D. (1996). *http://www.xtec.es/˜cdorado/cdora1/esp/disseny.htm*.

R. Koper, B. O. (2004). *Representing the Learning Design of Units of Learning. Educational Technology & Society.* Obtenido de http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.4.2417&rep=rep1&type=pdf

Ramírez, J. A. (2007). *Estado del arte del eLearning. Ideas para la definición de una plataforma universal,*. Obtenido de http://www.lsi.us.es/docs/doctorado/memorias/Marquez,%20Jose%20M.pdf
Real Academia Española. (s.f.). Obtenido de http://www.rae.es/rae.html

Sommerville, I. *Ingeniería del software.* Madrid: Pearson educación S.A.

Tattersal. (2003). *IMS Learning Design Frequently Asked Question.* Obtenido de http://dspace.learningnetworks.org/bitstream/1820/116/2/IMS%20Learning%20Design%20FAQ%201.0.pdf

Tatum. (2003). *Informe e-Learning*. Obtenido de http://banners.noticiasdot.com/termometro/boletines/docs/consultoras/tatum/2003/tatum_elearning03.pdf

Valdivia, I. M., Pascual, T. G., & Roca, A. E. (2006). *DELIMITACIONES PREVIAS A LA FORMACIÓN PARA EL USO DE LAS TIC EN LA ENSEÑANZA UNIVERSITARIA: FUNCIONES Y COMPETENCIAS DEL DOCENTE EN ENTORNOS VIRTUALES.* Barcelona.

Ventajas e inconvenientes de la formación on-line. Principales retos y oportunidades. (s.f.). Obtenido de http://www.areadeventas.com/articulos%20Oscar3.htm

Ventajas y desventajas del e-Learning. (s.f.). Obtenido de http://www.tress.com.mx/esp/Portals/0/Documentos%20varios/Boletín%20mensual/Junio/E-learning%20esp.pdf

Ventajas y desventajas del e-Learning. (Junio de 2008). *Redes sociales en educación* .

Wiley, D. A. (2002). *Instructional Use of Learning Objects, Connecting learning objects to instructional design theory: A definition, a metaphor, and taxonomy.* Agency for Instructional Technology.