ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE INGENIERÍA

DESARROLLO DE NUEVAS FUNCIONALIDADES EN UN SISTEMA OPEN SOURCE DE AULAS VIRTUALES (LMS)

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

LLUGSHA PINOS WILLIAM JAVIER

DIRECTOR: ING. CÉSAR ESQUETINI CÁCERES

Quito, marzo del 2007

DECLARACIÓN

Yo William Javier Llugsha Pinos, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

William Javier Llugsha Pinos

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por William Javier Llugsha Pinos, bajo mi supervisión.

Ing. César Esquetini Cáceres
DIRECTOR DE PROYECTO

AGRADECIMIENTOS

A todas las personas que confiaron en mí, en especial a Dios por darme la vida y las fuerzas de seguir adelante.

Al Ing. César Esquetini por su guía, profesionalismo, su don de ser y su colaboración para la realización de este proyecto.

A Ramiro Boada por su amistad, conocimientos y apoyo en la realización de esta tesis.

A mis compañeros de la carrera: Francisco, Sandy, Pol, Amy, Ricardo, Mony, Ramiro y muchos más por brindarme su amistad y compañerismo durante toda la carrera.

William

DEDICATORIA

A mi esposa, Yoly

Intentando expresarle mi amor y gratitud por su apoyo incondicional, su comprensión generosa y su gran tolerancia a mis metas profesionales.

A mis padres

Que me han dado la oportunidad de existir y guiarme por el camino del bien.

A mi hermano

Que con su ejemplo supo inculcar en mí el sentimiento de amistad y responsabilidad.

William

INDICE GENERAL

INDICE DE TABLAS

INDICE DE FIGURAS

INTRODUCCION

El trabajo expuesto a continuación engloba el desarrollo y la incorporación de nuevas funcionalidades en un Sistema Open Source de aulas virtuales (LMS – Learning Management System) que apoye el proceso de enseñanza-aprendizaje en programas de pregrado y postgrado de educación superior. Estos nuevos módulos o componentes interactuarán con el LMS seleccionado con el objeto de que dicha plataforma se adapte a una institución de educación superior en las diferentes modalidades: presencial, semi-presencial y a distancia.

Para cumplir con este objetivo, la tesis se ha dividido por capítulos de la siguiente manera:

En el capítulo I se seleccionará al LMS Open Source a través del establecimiento de criterios de selección y su comparación entre ellos. Además, se efectuará un análisis de la arquitectura, metodología, componentes o módulos, etc. del LMS seleccionado.

En el capítulo II se tratará sobre la selección de la metodología de desarrollo, para ello se efectuará una descripción, análisis y comparación de dos metodologías más utilizadas, en base a la descripción del proyecto que se pretende resolver. Este mismo proceso se seguirá para las herramientas de desarrollo a utilizarse en el proyecto.

En el capítulo III se identifica y detalla el problema en base a su definición y se introduce a la fase del análisis, en el cual se definen las historias de usuario, número de iteraciones, la planificación de entregas en base a las iteraciones, etc. Para la implementación se utiliza el desarrollo iterativo, por tal razón para cada iteración contempla el diseño, codificación y pruebas, obteniendo así el producto final.

En el capítulo IV se presentan las conclusiones y recomendaciones que han sido elaboradas una vez terminado el sistema.

CAPITULO 1. SELECCIÓN DEL SISTEMA OPEN SOURCE – LMS (LEARNING MANAGEMENT SYSTEM)

En este capítulo se seleccionará al LMS Open Source que servirá como base de para desarrollo de nuevas funcionalidades que es el objetivo principal de esta tesis.

1.1 ESTABLECIMIENTO DE CRITERIOS DE SELECCIÓN

En esta sección se determinarán los criterios de selección para seleccionar un LMS Open Source.

La selección de un LMS no es una decisión sencilla. Y aunque los LMS comparten un gran número de funcionalidades su filosofía de uso y posibilidades de integración son muy variadas. En este sentido es preciso mencionar que la selección de un LMS dependerá de los requerimientos de la Institución de Educación Superior. Es por ello que al momento de seleccionar uno, se aplican instrumentos de evaluación para determinar el más apropiado según las necesidades de la Institución que la adoptará.

Pero debido a que esta tesis no se basará en alguna Institución de Educación Superior específica y tomando en cuenta que el objetivo principal de esta tesis es "*desarrollar e incorporar nuevas funcionalidades en un Sistema LMS Open Source que apoye el proceso de enseñanza aprendizaje en programas de pregrado y postgrado de educación superior*", se seleccionará al LMS de acuerdo a la evaluación técnica efectuada por el proyecto JOIN[1]

1.1.1 CRITERIOS DE SELECCIÓN

La metodología de evaluación desarrollada por el equipo de JOIN e inspirada en un estudio de Peter Baumgartner y en la recomendación ISO/IEC 9126, establece la comprobación inicial de que el sistema examinado responde a la siguiente "Definición funcional mínima de LMS":

- El sistema debe ser de **código abierto**.

[1] **JOIN** - Proyecto europeo financiado por la iniciativa eLearning de la Comisión Europea, que evalúa la calidad de LMS open source para poder ofrecer información y apoyo a toda la comunidad que desee adoptar alguno de estos sistemas. http://www.ossite.org/join/sp/

- Debe ser **accesible a través de un navegador web estándar**.
- Las opciones de Autoría (producción) así como el resto de funciones del sistema deben poder ser utilizadas **sin** la necesidad de comprar ningún **plug-in** o visualizador adicional.
- Deben haber funciones básicas para la **administración de usuarios**.
- Se debe ofrecer una función de **autentificación**.
- El sistema debe ofrecer gestión de **permisos**.
- El sistema debe estar abierto a la **comunidad**.
- El alumno debe poder interaccionar a través del navegador con el profesor, el sistema y otros alumnos. La **comunicación** debe poder ser electrónica.
- Deben haber funciones básicas para la **evaluación** y progreso de los alumnos y funciones básicas para al menos la Autoría (producción) de tests y evaluaciones.
- Deben haber funciones para la **gestión de cursos**.
- Deben haber funciones para la **gestión de contenidos**.

Si el sistema supera dicha comprobación, es instalado siguiendo su documentación técnica y analizado en base a una 'Lista de criterios' que se traducen en un "**Cuestionario de evaluación**" (ver en el CD adjunto, documento: *JOIN – Cuestionario de evaluacion.doc*) compuesto por 120 preguntas de respuesta cerrada. Dicho cuestionario está destinado al equipo de evaluación de JOIN y al equipo de desarrollo del producto. La información contenida en este cuestionario, una vez completado, se reelabora en forma de tabla y es lo que se conoce como "Ficha técnica". Esta ficha, junto con una "**Ficha del producto**", de carácter más generalista, forma el paquete informativo de un LMS.

La "**Lista de criterios"** en la que se basa la metodología de evaluación de JOIN contempla los cinco aspectos siguientes:

1.1.1.1 Requisitos funcionales

Las características funcionales se identifican mediante un cuestionario sobre las características de acceso y seguridad así como la implementación de permisos, didáctica, tests y evaluación, Autoría (producción), monitorización del curso, comunicación y administración.

1.1.1.2 Mantenibilidad

La calidad se ve también afectada por características no funcionales del sistema de tele-enseñanza. El sistema tiene que poder soportar cambios futuros y esta capacidad se ve afectada por aspectos distintos: la arquitectura del sistema, la conformidad a los estándares y la documentación.

1.1.1.3 Facilidad de uso

La facilidad de uso también es significativa a la hora de elegir un sistema de tele-enseñanza. El sistema debería ser fácil de usar para toda la gente involucrada: tutores, administradores y alumnos. Todos ellos necesitarán algún tipo de formación sobre el sistema antes de usarlo. El sistema debe ser igualmente de fácil navegación.

1.1.1.4 Calidad del soporte

El soporte es también de gran importancia, dado que los productos de software libre no tienen garantía. El soporte disponible se evalúa desde el punto de vista de la operatibilidad diaria y en los momentos críticos de problemas.

1.1.1.5 Coste total

Hay aspectos de calidad que surgen de la licencia de software libre del sistema. Si el desarrollador pierde el entusiasmo o la financiación, el proyecto de desarrollo puede venirse abajo. La organización que emplee su software puede verse enfrentada entonces a un serio problema. Puede ser que tenga que cambiar de sistema, lo que ocasionará costes. En este apartado también se estudia la comunidad de usuarios ya que es una clave de garantía respecto a la evolución, vitalidad y supervivencia del software. Otro aspecto importante a considerar es el

coste inicial de establecimiento del sistema, en términos de hardware necesario, software y dedicación humana.

Las respuestas a las preguntas relacionadas con la funcionalidad, mantenibilidad y facilidad de uso del sistema están cuantificadas en una escala de 0 a 199.

Las respuestas a las preguntas sobre la calidad del soporte y el coste total del producto aportan información cualitativa no cuantificada.

1.2 SELECCIÓN DEL LMS

Para la selección del LMS es indispensable tomar en cuenta los criterios enunciados anteriormente, anteponiendo ciertos criterios académicos y de la información de ciertas organizaciones que efectúan comparaciones entre los diferentes LMS existentes en el mercado, como son los enlaces siguientes:

- **JOIN[2]:** Proyecto europeo financiado por la iniciativa eLearning de la Comisión Europea, que evalúa la calidad de LMS open source para poder ofrecer información y apoyo a toda la comunidad que desee adoptar alguno de estos sistemas. http://www.ossite.org

- **Open Source CMS[3]**: Aquí es posible acceder a instalaciones y trastear en diferentes plataformas de gestión de contenidos entre las que se encuentra Dokeos, Moodle y algunas más no tan extendidas.

- **CMS Matrix[4]**: En este sitio web pueden realizarse comparaciones entre diferentes plataformas de gestión de contenidos. Muy útil para analizar pormenorizadamente todos los aspectos de los LMSs.

- **Edutools[5]**: Otro sitio web en la que se pueden realizar comparaciones de LMSs.

En la actualidad existe una diversidad de LMSs, por esta razón, para la preselección de LMSs se tuvieron que se destacar 3 tendencias bien definidas en

[2] JOIN - http://www.ossite.org
[3] Open Source CMS - www.opensourcecms.com
[4] CMS Matriz - www.cmsmatrix.org
[5] Edutools - www.edutools.info

el panorama actual y se escogieron los 3 sistemas más prometedores y representativos de cada una de dichas tendencias:

1. Entornos centrados en la administración de contenidos de cursos y también una herramienta de colaboración para la modalidad de auto-estudio. El LMS seleccionado fue **Dokeos**.

2. Entornos centrados en la comunicación y las actividades de enseñanza/aprendizaje que incluyen, también, herramientas para gestionar materiales. El sistema escogido fue **Moodle**.

3. Entornos centrados en la creación, gestión y distribución de contenidos, con algunas herramientas de comunicación añadidas, pero en segundo plano. El software seleccionado fue **ATutor**.

Según la metodología de Evaluación de JOIN descrita anteriormente, sugiere que las respuestas al Cuestionario de Evaluación se reelabora en forma de tabla y es lo que se conoce como “Ficha técnica”.

Estas fichas técnicas se las obtuvieron de la evaluación efectuada por JOIN y se las complementará con la información proporcionada por organizaciones como Edutools, CMS Matriz u Open Source CMS.

1.2.1 COMPARACIÓN DE LOS LMS

Para las plataformas preseleccionadas se presentará las fichas del producto:

1.2.1.1 Producto: Dokeos 1.6.4

Breve descripción: El proyecto empezó desde una versión previa de Claroline y se ha convertido en un producto por sí mismo. El objetivo es ayudar al docente a crear contenido pedagógico, a estructurar las actividades en caminos de aprendizaje (itinerarios formativos), a interaccionar con los estudiantes y a seguir su evolución mediante un sistema de informes. Dokeos ha conseguido en poco tiempo el entusiasmo de sus usuarios.

Sitio oficial del proyecto: http://www.dokeos.com

1.2.1.1.1 Ficha del Producto

Tabla 1-1: Ficha del Producto: Dokeos 1.6.4

Ficha del producto	
Nombre LMS	Dokeos
Equipo de desarrollo	El equipo está liderado por *Thomas De Praetere* y colaboran con él más de diez desarrolladores más, además también contribuyen algunas universidades.
Servicio de soporte	Es posible entrar en contacto con el equipo de desarrollo y soporte escribiendo a ***Info Dokeos*** (info@dokeos.com)
Número de la versión	1.6.4
Tecnología utilizada	PHP, Apache, MySQL
Sistemas Operativos soportados	Todos los Sistemas Operativos en los que se pueda instalar PHP, Apache y MySQL (e.g. Windows, Linux, UNix, etc)
Requisitos de Bases de Datos/Software	Apache, PHP 4.x o posterior (con soporte habilitado para mysql, zlib, preg y xml) y MySQL 3.23.6 o posterior.
Tipo de licencia	GPL
Estandares de E-Learning soportados	SCORM 1.2
Lenguas disponibles	En la actualidad están disponibles más de veinte lenguas (árabe, croata, holandés, inglés, finlandés, francés, alemán, italiano, japonés, portugués, ruso, español y muchas más)
Características principales	Dokeos es un fork bastante reciente de Claroline. Ambas herramientas son similares, pero Dokeos muestra su propia personalidad ahora. La aproximación diferente para los caminos de aprendizaje, la compatibilidad SCORM runtime, la organización distinta de algunas herramientas, permiten decir que Dokeos es más que una operación estética de Claroline. La nueva versión examinada también ofrece tecnología plug-in para la extensión de la plataforma.
Posibilidad de gestionar los contenidos de los cursos	Se pueden crear y gestionar cursos desde dentro de Dokeos y es posible realizar contenidos (páginas) y tests.
Aspectos positivos	• Alta facilidad de uso y fiabilidad • Soporte a caminos de aprendizaje • Chat, forum, Video/audio conferencing y muchas otras herramientas de comunicación • Test y evaluación • Módulos de importación de paquetes SCORM y soporte SCORM runtime • Soporte multilingüe • Alta modularidad con tecnología plug-in
Aspectos a mejorar	• Las herramientas de autor pueden mejorar

	• No hay motor de búsqueda • La documentación no está completa y existe sólo en inglés
Principales instalaciones en el mundo	Hay varias instalaciones en todo el mundo, la lista está en el website.
Instalaciones en Italia	• Campus Imedia[6] • Istituto Bergese[7] • Università di Parma[8] • Laser Virtual Campus[9]
Notas	Dokeos es una plataforma muy interesante, la comunidad de desarrollo es amplia y su **Dream map**, que es la lista de desarrollos deseados, es realmente apasionante.
Enlace a la versión demo	http://www.dokeos.com/campus/dokeos/1234bcc0/

Fuente: JOIN (www.ossite.org)
Elaborado por: LLUGSHA, William

1.2.1.1.2 Ficha Técnica

Tabla 1-2: Ficha Técnica: Dokeos 1.6.4

Requisitos funcionales	
Didáctica / Administración del aprendizaje	Dokeos permite crear itinerarios formativos (caminos de aprendizaje) para el curso. Cada itinerario puede ser creado de uno o más módulos, cada uno de estos módulos contiene un conjunto de recursos disponibles en el curso. Una vez que los módulos son escogidos, podemos configurar el orden y las dependencias entre módulos y entre los ítems incluidos.
Pruebas y Evaluaciones	Las pruebas pueden ser creadas dentro del sistema, almacenando las preguntas en un banco de preguntas. En esta versión es posible importar paquetes HotPotatoes. Están disponibles los siguientes tipos de preguntas: selección simple o múltiple, concordancia de respuestas (Matching) y completar en el blanco. Es posible realizar un seguimiento y ver las estadísticas de la pruebas. Estas estadísticas están disponibles para cada usuario.
Formato de Medios / Autoría (producción)	El contenido del curso puede ser generado dentro del sistema (visto como archivos HTML), e importado en cualquier formato (PDF, Doc, txt, HTML, etc). Los módulos SCORM también son soportados dentro de la plataforma, incluso los resultados pueden ser vistos en forma de estadísticas. En esta versión es posible exportar itinerarios de aprendizaje en formato SCORM. El contenido del curso puede ser organizado en carpetas o directorios.
Monitoreo del curso	Los usuarios pueden ser inscritos en un curso por el administrador o ellos pueden inscribirse por ellos mismos. Es posible monitorear mucha información (acceso a herramientas, documentos, accesibilidad, etc.)

[6] Campus Imedia - http://portal.dokeos.com/user_67/
[7] Istituto Bergese - http://www.istitutobergese.it/dokeos/
[8] Università di Parma - https://corsi.unipr.it/index.php
[9] Laser Virtual Campus - http://corsi.laserbs.net/dokeos/index.php

Comunicación	Para la comunicación sincronía están disponibles el Chat y la conferencia de audio y video, en tanto que para la comunicación asíncrona, el mail interno (baúl de tareas), foros y archivos compartidos (publicación de los estudiantes). En esta versión y gracias a la tecnología Plug-in esta disponible **RSS feed**[10]
Look and feel del sistema	Dokeos hace uso de archivos CSS y es posible personalizarlo casi todo, pero estas personalizaciones son a nivel de la plataforma (los usuarios no pueden escoger un diferente estilo o cambiarlo este archivo)
Implementación de Permisos	Los permisos son implementados en base a roles. Existen 4 roles predefinidos: administrador, profesor, usuario e invitado. No es posible modificar estos roles o crear un nuevo rol. Los usuarios pueden ser organizados dentro de grupos.
Otras Características	Mas de 30 lenguajes son soportados por Dokeos (Brazilian, Croatian, Dutch, English, Esperanto, French, German, Italian, Persian, Portuguese, Serbian, Slovenian, Spanish, Swedish, y mucho más). También es soportado el módulo de autenticación LDAP. Gracias al módulo de publicaciones de los estudiantes, los usuarios pueden compartir notas o archivos, como una forma de crear un nuevo ambiente de trabajo.
Mantenibilidad	
La calidad de documentación técnica	En la actualidad los documentos para el desarrollador están disponibles en español, inglés y francés. Este documento está disponible en sección denominada *Development Guidelines* del sitio Web de dokeos, que es una clase de manual que contiene información acerca de la arquitectura, estructura de la base de datos, metodología de desarrollo, convenciones de código, etc. El API esta disponible en la url: www.phpxref.com/xref/dokeos/nav.html.gz?_functions/index.html.gz.
Escalabilidad	Existen muchas instituciones y universidades que están usando Dokeos. Se estima que por cada instalación más de cinco mil usuarios y 150 cursos pueden ser soportados, pero no es claro si el sistema crece horizontalmente. El sistema es muy confiable.
Extensibilidad	El código es claro y las convenciones de código son normalmente respetadas. En estos momentos la documentación del API está completada como se enunció anteriormente, y la oportunidad de ampliar las características con plug-ins es realmente importante, actualmente existen algunos plug-ins disponibles. Los requerimientos del usuario y problemas son manejados dentro del foro de Dokeos y existe un sistema CVS público. Existen 4 versiones estables liberadas.
Adaptabilidad / Cumplimiento de estándares	El sistema Dokeos cumple con el estandar SCORM 1.2 y es posible importar y exportar contenidos SCORM. También soporta el estándar IEEE LOM pero solamente para las herramientas documentos y grupos.
Facilidad de uso	

[10] WIKIPEDIA, Glosario, http://es.wikipedia.org/wiki/RSS, 2006. [**RSS** es parte de la familia de los formatos XML desarrollado específicamente para todo tipo de sitios que se actualicen con frecuencia y por medio del cual se puede compartir la información y usarla en otros sitios web o programas.]

Documentación de Usuario	La documentación esta disponible para los usuarios para su edición en diferentes lenguas. Para el usuario existe el manual del administrador, profesor y estudiante.
Generación de contenido	El sistema es realmente fácil de usar y es posible crear pruebas, contenidos, entre otros dentro de la plataforma, con la opción de modificar lo que nosotros hayamos creado.
Nivel de experticia necesaria	Todas las operaciones del sistema, en cualquier nivel, no requieren de una especial experticia para el manejo de Dokeos. Cualquier operación puede ser ejecutada en forma simple sin ningún tipo de entrenamiento.
Otras características	El sistema es fácil de navegar, el idioma puede ser seleccionado por los usuarios y las personalizaciones son posibles a nivel de plataforma. Las modificaciones gráficas serán disponibles únicamente un plug-in. Los contenidos y datos disponibles en Internet como los módulos SCORM, pueden ser usados en cualquier tipo de documento (formato) y páginas HTML. Los permisos del sistema realmente no son flexibles pero es fácil de entender.
Calidad del soporte	
Soporte	Es posible acceder a muchos soportes para entrenamiento, consulta, hosting, etc. Para cualquier consulta o información es posible escribir a l mail: info@dokeos.com. El equipo de Dokeos está siempre atento ante cualquier requerimiento y por otro lado existe mucha información útil en el foro de Dokeos.
Coste total	
Tipo de usuarios	Colegios y universidades son los usuarios predominantes, pero también las compañías o instituciones públicas. Un listado en el sitio Web de Dokeos incluye solamente organizaciones que están usando los servicios comerciales de Dokeos y el número de usuarios es bastante grande.
Estabilidad financiera	El proyecto inició en el 2003, y el equipo núcleo está compuesto de alrededor de 7 miembros, pero hay pocos desarrolladores alrededor del mundo. El equipo se funda por personas comprometidas por colaborar y por servicios proporcionados.
Coste inicial de establecimiento del sistema	No existen especiales requerimientos de hardware. Los requerimientos de software son: Apache, MySQL, PHP y algunos módulos PHP (mysql, zlib, preg, xml). La puesta en marcha del sistema toma menos de un día.
Costos recurrentes	Actividades periódicas son las mismas para todo sistema Web: parches en el sistema operativo o instalación de herramientas.

Fuentes: JOIN (www.ossite.org); LLUGSHA, William
Elaborado por: LLUGSHA, William

1.2.1.2 Producto: Moodle 1.4.1

Breve descripción: Moodle es uno de los LMS más populares y está actualmente viviendo una fase explosiva de expansión. Su comunidad de usuarios

y desarrolladores es muy numerosa y se caracteriza por su entusiasmo respecto al sistema. Moodle es un proyecto inspirado en la pedagogía del constructivismo social.

Sitio oficial del proyecto: http://www.moodle.org

1.2.1.2.1 Ficha del Producto

Tabla 1-3: Ficha del Producto: Moodle 1.4.1

Ficha del producto	
Nombre LMS	Moodle
Equipo de desarrollo	En 1999 Martin Dougiamas (Australia) inició el proyecto Moodle. Actualmente colaboran en él alrededor de 100 personas entre desarrolladores (cerca de 50), traductores (otros 40), beta-testers ...
Servicio de soporte	Desde hace un año en http://www.moodle.com los Moodle Partners (grupo de empresas de servicios) ofrecen un amplia oferta de servicios comerciales para usuarios, entre los cuales hosting Moodle completo, contratos de soporte remoto, desarrollos a medida y consultoría.
Número de la versión evaluada	1.4.1
Tecnología utilizada	PHP
Sistemas Operativos soportados	Unix, Linux, Windows, Mac OS X, Netware y cualquier otro sistema que soporte PHP.
Requisitos de Bases de Datos/Software	PHP version 4.1.0. o mayor. Cualquier servidor web que soporte PHP (la mayoría de las instalaciones usan Apache). Una base de datos, MySQL y PostgreSQL especialmente, pero también puede usarse Oracle, Access, Interbase, ODBC y otras.
Tipo de licencia	GPL
Estandares de E-Learning soportados	Importación SCORM 1.2 y IMS-QTI
Lenguas disponible	40 lenguas incluidas Arabe, Catalán, Chino (simplificado y tradicional), Checo, Danés, Holandés, Inglés (versiones UK y US), Finlandés, Francés (versiones Francia y Canada), Alemán, Griego, Húngaro, Indonesio, Italiano, Japonés, Noruego, Polaco, Portugués (Portugal y Brasil), Rumanés, Ruso, Slovaco, Español (versiones España, Méjico, Argentina y Caribe), Sueco, Tailandés y Turco.
Características principales	La principal característica de Moodle, aparte de su fundamento en la pedagogía del constructivismo social, es su gran y continuamente creciente comunidad de usuarios que le da al sistema una enorme vitalidad.
Posibilidad de gestionar los contenidos de los cursos	Moodle permite crear y gestionar contenidos de cursos y tests.

Aspectos positivos	• El sistema es muy intuitivo y fácil de usar • Está traducido a más de 40 lenguas • Se apoya en una gran comunidad de usuarios y desarrolladores
Aspectos a mejorar	• Está en marcha el proyecto de mejora de la documentación Moodle Documentation Project basado en Wiki. • Se está trabajando en un repositorio de módulos desarrollados. • Se está trabajando en el nivel de personalización de la apariencia del sistema mediante plantillas. • Moodle no soporta las especificaciones de accesibilidad (ej. W3C WAI) • Moodle no soporta la exportación SCORM 1.2. y otros estándares de e-learning
Principales instalaciones en el mundo	Miles de instalaciones por todo el mundo, listadas en http://www.moodle.org/sites
Instalaciones en España	En España un centenar de instalaciones, listadas en http://www.moodle.org/sites
Notas	El desarrollo e implantación que Moodle está teniendo es realmente espectacular. Su gran comunidad de usuarios y desarrolladores le confieren un sello especial de calidad y continuidad al proyecto. Es una opción a tener seriamente en cuenta a la hora de escoger un LMS de software libre.
Enlace a la versión demo	http://www.moodle.org

Fuente: JOIN (www.ossite.org)
Elaborado por: LLUGSHA, William

1.2.1.2.2 Ficha Técnica

Tabla 1-4: Ficha Técnica: Moodle 1.4.1

Requisitos funcionales	
Didáctica / Administración del aprendizaje	El sistema ofrece algunos esquemas de cursos pre-estructurados. Los tutores no pueden definir itinerarios formativos condicionales o alternativos. El sistema no soporta planes de aprendizaje. El sistema parcialmente soporta el portafolio del estudiante (es posible seguir una secuencia de todas la actividades del estudiante dentro de un curso, pero el concepto global de portafolio del estudiante no existe). El estudiante recibe retroalimentación de diferentes maneras dependiendo del tipo de actividad que el estudiante este realizando. En cualquier momento el estudiante puede ver el estado de su trabajo y de muchas actividades, el sistema envía emails de alerta para informar al estudiante acerca de sus principales problemas.
Pruebas y Evaluaciones	El sistema ofrece la funcionalidad de Autoría (producción) para la generación y modificación de pruebas. Las pruebas pueden ser importadas de varios formatos. Diferentes tipos de preguntas puede ser empleados para las pruebas: selección simple o múltiple, concordancia de respuestas de términos y figuras (Matching), entre otras. No hay funcionalidad de servicios. El sistema no puede generar certificados después de finalizar una carga de trabajo pero

	un módulo de certificados está siendo desarrollado.
Formato de Medios / Autoría (producción)	El contenido puede ser creado dentro del sistema. Cualquier contenido Web es soportado y puede ser importado. Los cursos de Moodle y los paquetes SCORM 1.2 pueden ser importados. También los cursos pueden ser exportados (en formato propietario XML) con asociados metadatos. Los datos de los usuarios pueden ser importados o exportados desde Moodle (formato XML, texto, DB, LDAP).
Monitoreo del curso	En el sistema, los cursos pueden ser programados (período de registro o bloqueo de cursos). Los estudiantes pueden inscribirse por si mismo en un curso o los instructores ratifican la inscripción. Existe la funcionalidad para el seguimiento de los usuarios: los datos son presentados en tablas y pueden ser exportados. Los datos que pueden ser monitoreados son: resultados de pruebas, accesibilidad a la plataforma del usuario.
Comunicación	La comunicación síncrona es soportado por Moodle a través del Chat y la comunicación asíncrona a través del e-mail y foro. Herramientas externas para comunicación (pizarras, salas de chat) pueden ser integrados al sistema como programas externos. El sistema soporta RSS/blogging.
Look and feel del sistema	El diseño del sistema puede ser personalizado en pequeños aspectos. Los usuarios no pueden personalizar el diseño del sistema. Es posible personalizar la página de login.
Implementación de Permisos	Los permisos son manejados a través de la implementación de roles. Los roles identificados son: administradores, profesores, tutores, estudiantes e invitados. Los usuarios pueden ser organizados in grupos muy fácilmente.
Otras Características	El sistema hace uso de la encripción en la fase de login. No existe alguna funcionalidad para manejar pagos pero el sistema soporta Paypal. Los usuarios pueden registrar el contenido de aprendizaje pero no pueden compartir anotaciones con otros usuarios. Más de 40 idiomas son soportados. El sistema provee de mecanismos de autenticación LDAP.
Mantenibilidad	
La calidad de documentación técnica	Existe documentación para los desarrolladores en más de 4 idiomas. La calidad de la documentación es buena pero podría ser más explícita. Actualmente se está trabajando sobre la documentación del código con phpdoc.
Escalabilidad	El máximo número de usuarios activos de una instalación es de alrededor de 5000 usuarios, de igual manera, el número máximo de cursos dentro una instalación es de alrededor 150 cursos. Es posible administrar diferentes clientes en una instalación. El sistema puede crecer horizontalmente (agrupación de webservers con sesiones almacenadas en la base de datos). El sistema es muy confiable
Extensibilidad	Existen mecanismos documentados para la extensión de características en el proceso de desarrollo, en este sentido existe el concepto de plug-in. Existe parcialmente disponible una API definida. El sistema es desarrollado (en partes) con una aproximación

	orientada a objetos. Los desarrolladores se rigen a ciertas convenciones de código definidas para el LMS. La calidad del código fuente del proyecto es buena. No existe una metodología para la gestión de requerimientos de usuario. Existe un sistema de reporte de errores o fallas empleado por el equipo de desarrollo. El sistema ha tenido 6 releases de importancia y 15 releases de menor importancia.
Adaptabilidad / Cumplimiento de estándares	El sistema es modularizado en partes. Moodle cumple con el estándar SCORM 1.2 (reproducción de paquetes) y el estándar IMS-QTI (importación).
Facilidad de uso	
Documentación de Usuario	Existe ayuda en línea en el sistema y además la plataforma cuenta con documentación de calidad en varios idiomas (mas de 40) para usuarios, profesores y administradores.
Generación de contenido	El contenido de aprendizaje puede ser generado y modificado empleado las funcionalidades del sistema pero esto toma algún tiempo. Las pruebas pueden ser creadas y modificadas dentro del sistema.
Nivel de experticia necesaria	Poca experticia es requerida para usar la plataforma como un tutor o usuario. Para los administradores y profesores más experticia es necesaria debido a que el sistema es complejo y rico en características.
Otras características	Algún material de entrenamiento, además de la ayuda normal puede ser obtenida de manera libre de http://www.moodle.org. El sistema es fácil de manejar. El diseño del sistema puede ser configurado para cualquier organización. Los permisos del sistema es simple y muy fácil de entender. El sistema no cumple con ninguna especificación de accesibilidad.
Calidad del soporte	
Soporte	Las instituciones pueden adquirir algún soporte comercial en Moodle en http://www.moodle.com sobre las bases del contrato y garantía. Moodle provee de conjunto de servicios comerciales para los usuarios, incluyendo el servicio de hosting, contratos de apoyo remoto, desarrollo de código personalizado y consulta. El equipo de Moodle está atento para brindar soporte. Una respuesta a una inquietud en el foro toma unas pocas horas.
Coste total	
Tipo de usuarios	Los usuarios predominantes son las escuelas (25%), pequeñas y medianas empresas (25%), universidades (15%) y la industria (15%). Actualmente hay más de 2000 sitios registrados usando Moodle. Los clientes puede ser vistos en la dirección: http://www.moodle.org/sites
Estabilidad financiera	El proyecto inició en 1999 con el desarrollo de Matin Dougiamas. Actualmente el equipo de desarrollo tiene cerca de 100 miembros (desarrolladores, traductores, testers, etc.).
Coste inicial de establecimiento del sistema	La memoria y el disco necesaria depende del número de usuarios, nivel de concurrencia en el sistema y principalmente la características de los cursos. Los requerimientos de software son un servidor Web, una base de datos (principalmente MySQL) y PHP con

	algunas librerías opensource compiladas, que están incluidas en la distribución de Moodle. El sistema también necesita hardware y software adicional para copias de seguridad, conectividad, etc. Moodle puede ser configurado en menos de 30 minutos. Dependiendo sobre los enlaces a sistemas externos (LDAP, otra base de datos) y otros factores, la instalaci´n de Moodle debería operar sin problemas.
Costos recurrentes	Horas de entrenamiento recomendado para un administrador 20-30 h., para un profesor: 50-100 h., para un tutor: 20-30 h., para un usuario: 5-10 h.

Fuentes: JOIN (www.ossite.org); LLUGSHA, William
Elaborado por: LLUGSHA, William

1.2.1.3 Producto: Atutor 1.4.2

Nombre LMS: ATutor

Breve descripción: Este proyecto empezó en 2002 en colaboración con el Adaptive Technology Resource Centre (ATRC) de la Toronto Universiy. Este centro es un lider internacionalmente reconocido en el desarrollo de tecnologías y estándares que permitan a la gente con discapacidades el acceso a las oportunidades elearning y esta misión ha influenciado profundamente el desarrollo de la plataforma. El desarrollo ha prestado especial interés a la accesibilidad: ATutor es la única plataforma LMS que cumple las especificaciones de accesibilidad W3C WCAG 1.0 de nivel AA+.

Sitio oficial del proyecto: http://www.atutor.ca

1.2.1.3.1 Ficha del Producto

Tabla 1-5: Ficha del Producto: Atutor 1.4.2

Ficha del producto	
Nombre LMS	ATutor
Equipo de desarrollo	Greg Gay (lider del proyecto) y unos cuantos desarrolladores (la lista completa está en el website) en colaboración con el Adaptive Technology Resource Centre (Toronto University).
Servicio de soporte	Es posible entrar en contacto con el equipo de desarrollo y soporte rellenando los campos de la página *Contact ATutor*[11]
Número de la versión evaluada	1.4.2
Tecnología utilizada	PHP, Apache, MySQL

[11] Contact ATutor - http://www.atutor.ca/contact.php?subject=ATutor+Service+Request

Sistemas Operativos soportados	Todos los sistemas operativos en los cuales es posible instalar PHP, Apache y My SQL (ej. Windows, Linux, UNix, etc)
Requisitos de Bases de Datos/Software	Apache 1.2.x, PHP 4.2 o superior (con soporte habilitado para zlib y mysql) y MySQL 3.23.x o superior.
Tipo de licencia	GPL
Estandares de E-Learning soportados	SCORM 1.2, IMS packaging y las especificaciones de accesibilidad W3C WCAG 1.0 de nivel AA+
Lenguas disponibles	Más de dieciséis (danés, holandés, francés, alemán, griego, italiano, portugués, ruso, español y muchos más)
Características principales	Soporte de estándares, facilidades de importación/exportación de contenidos y usuarios, herramientas de seguimiento
Posibilidad de gestionar los contenidos de los cursos	Se pueden gestionar cursos, es fácil crear contenidos y tests dentro de la plataforma e importar paquetes SCORM 1.2 e IMS
Aspectos positivos	• Entorno integrado de autor para crear y gestionar cursos • Soporte de palabras clave para ayudar al motor de búsqueda • Búsqueda posible para todos los cursos del catálogo • Herramientas de importación/exportación de usuarios • Soporte de paquetes SCORM y IMS • Herramientas de accesibilidad • Disponibilidad de herramientas de test y evaluación
Aspectos a mejorar	• SCORM runtime no soportado • Documentación sólo en inglés • Faltan herramientas síncronas
Principales instalaciones en el mundo	Hay varias instalaciones por todo el mundo: la lista completa está en el website
Instalaciones en Italia	• ISIT Bassi Burgatti[12] • Università di Bologna - Facoltà di Economia[13] • Università di Firenze - Facoltà di Medicina[14] • Università di Cassino - Facoltà di Giurisprudenza[15] • Conlabora.net[16]
Notas	Una plataforma muy bonita que permite hacer muchas cosas de un modo simple y donde la facilidad de uso, el perfil educacional y el aprendizaje humano son particularmente tenidos en cuenta.
Enlace a la versión demo	http://www.atutor.ca/atutor/demo.php

Fuente: JOIN (www.ossite.org)
Elaborado por: LLUGSHA, William

[12] ISIT Bassi Burgatti - http://www.isit100.fe.it/atutor/browse.php
[13] Università di Bologna - Facoltà di Economia - http://levi.ei.unibo.it/ATutor/browse.php
[14] Università di Firenze - Facoltà di Medicina - http://internet4.med.unifi.it:8080/atutor/browse.php
[15] Università di Cassino - Facoltà di Giurisprudenza - http://ius.unicas.it/ATutor/login.php
[16] Conlabora.net - http://www.conlabora.net/corsi/login.php

1.2.1.3.2 Ficha Técnica

Tabla 1-6: Ficha Técnica: Atutor 1.4.2

Requisitos funcionales	
Didáctica / Administración del aprendizaje	Actualmente, los itinerarios formativos nos son disponibles y no es posible implementar actividades de aprendizaje.
Pruebas y Evaluaciones	Las herramientas son realmente fáciles de usar y las páginas HTML pueden ser creadas sin ninguna experticia. Contenidos como los paquetes SCORM o IMS pueden ser importados, además páginas HTML o cualquier formato de archivos pueden ser importados y organizados dentro de la plataforma. Las pruebas (selección múltiple, verdadero/falso, preguntas abiertas) pueden ser creadas dentro del sistema y podemos seleccionar si la calificación debe ser asignada automáticamente o por el tutor.
Formato de Medios / Autoría (producción)	Paquetes de contenidos SCORM e IMS pueden ser importados, pero también archivos de cualquier formato. Una característica importante es la posibilidad de exportar un curso entero (o parte de este) en formato del estándar ISM. Los cursos pueden ser clasificados en una simple carpeta y los contenidos pueden ser escritos con palabras claves que serán usadas para el motor de búsqueda. Los datos de los usuarios pueden ser exportados e importados en un formato CSV y todos los archivos multimedia son soportados.
Monitoreo del curso	Un curso puede ser publicado después de una fecha predefinida. Los usuarios pueden inscribirse por ellos mismos y los instructores aprueban la inscripción. Es posible monitorear toda la información (herramientas, estudiantes, etc.) y estos datos pueden ser exportados.
Comunicación	La comunicación síncrona es posible usando Chat, en tanto que la comunicación asíncrona está disponible a través de los foros, email e intercambio de archivos. Toda estas herramientas están en el sistema Acollab, un add-on para ATutor 1.4+.
Look and feel del sistema	El sistema hace uso de hojas de estilo y cada usuario puede escoger el diseño (posición) de los objetos dentro de la página, ocultando algunas cosas o simplemente incorporar uno de los disponibles plantillas. Los instructores pueden también establecer una plantilla para un conjunto de cursos.
Implementación de Permisos	Los permisos del sistema es bastante bueno y no es posible definir nuevos roles. Los roles predefinidos son: administrador, instructor, usuario e invitado, y estos son suficientes para satisfacer cualquier escenario. Es posible crear grupos y todas las herramientas de colaboración son basadas en grupos.
Otras Características	Los usuarios pueden compartir notas pero no pueden registrar un particular contenido. En estos momentos más de 16 idiomas están disponibles y es posible agregar una nota en el final de las páginas del contenido con el objeto de añadir el concepto de propiedad intelectual. Soporte LDAP no es disponible desde esta versión.
Mantenibilidad	
La calidad de documentación técnica	Existe un sitio Web donde las directrices de los desarrolladores está bien descrito (existe solamente la versión en inglés).

Escalabilidad	El sistema no es muy popular, pero existen muchos demos e instalaciones oficiales alrededor del mundo. El sistema es también confiable.
Extensibilidad	El código es bastante bueno y las convenciones de código son casi respetadas. No existe el concepto de plug-in y además el API del sistema no está definido. Los requerimientos del usuario y problemas son manejados dentro del foro. Subversion es una herramienta que ha sido adoptada para el repositorio y el versionamiento de la plataforma. Actualmente existen 6 versiones estables.
Adaptabilidad / Cumplimiento de estándares	El sistema soporta empaquetamiento IMS para importar y exportar funcionalidades y SCORM 1.2 solamente para importar. El sistema es modular en el sentido que las funciones son agrupadas.
Facilidad de uso	
Documentación de Usuario	La documentación esta solamente disponible en inglés y para cada actor involucrado en el escenario (administradores, instructores y estudiantes). También existe ayuda en línea.
Generación de contenido	Los instructores pueden fácilmente crear y modificar contenido, incluso es posible modificar contenido SCORM dentro de un editor. De igual manera es para las pruebas.
Nivel de experticia necesaria	El sistema es fácil de usar y no se requiere un alto nivel de experticia. El único inconveniente es que toma algún tiempo para ver y entender todas las funcionalidades disponibles en la plataforma.
Otras características	Los materiales a ser usados en el sistema pueden ser encontrados en formato SCORM LO, IMS o cualquier archivo con cualquier formato. El sistema es fácil de usar, personalizable y los permisos del sistema es simple de entender. Además, la plataforma cumple con las especificaciones de accesibilidad (W3C WAI).
Calidad del soporte	
Soporte	Es posible contactar al equipo de Atutor en su sitio Web (www.atutor.ca/contact.php?subject=ATutor+Service+Request) para obtener servicios, sobre las bases de un contrato, con respecto a un nuevo desarrollo, hosting, consultas y soporte. El equipo esta siempre presto a responder cualquier inquietud.
Coste total	
Tipo de usuarios	Los tipos predominantes de usuarios son los colegios y universidades, pero también existen pequeñas empresas y organizaciones. El número de usuarios puede ser estimado en algunos miles en todo el mundo y hay una completa lista de instituciones en su sitio Web si nosotros queremos contactarnos con ellos.
Estabilidad financiera	Este proyecto inició en el 2002 y el equipo puede ser estimado eb menos de diez personas. El equipo núcleo está consolidado y hay poco personal que brinda soporte que colabora en el proyecto.
Coste inicial de establecimiento del sistema	No existen requerimientos especiales de hardware. Los requerimientos de software son: Apache, MySQL, PHP (las versiones sugeridas se encuentran descritas en el sitio Web)

Costos recurrentes	Actividades periódicas son las mismas para todo sistema Web: parches en el sistema operativo o instalación de herramientas.

Fuentes: JOIN (www.ossite.org); LLUGSHA, William
Elaborado por: LLUGSHA, William

De la descripción técnica se deriva que **Dokeos**, **Moodle y Atutor** no son los únicos LMS con licencia Open Source, pero **Dokeos** y **Moodle** son probablemente los más difundidos.

La comparación de Moodle y Dokeos puede dar una idea de la variedad de enfoques que los LMS pueden tener.

Moodle basa su modelo pedagógico en el constructivismo social, esto es, en el establecimiento de comunidades alrededor de un tema que realizan actividades, reflexión crítica, etc. Esto marca profundamente su organización e interfaz, construida alrededor de 3 modelos de interacción on-line:

- Weekly, en la que toda la interfaz gira alrededor de la asignación de actividades semanales.
- Topics, en la que queda organizada en base a los temas propuestos en el curso.
- Social, en la que el eje central del curso pasa a ser un foro de discusión.

Por otra parte, el modelo de Dokeos es algo distinto. La interfaz se organiza en base al concepto de curso como agrupación de distintos tipos de recursos: contenido, foro, auto-evaluaciones, etc.

Y aunque las funcionalidades son casi las mismas en ambos sistemas, dependiendo del estilo pedagógico del curso será más fácil impartirlo usando una plataforma y otra. Simplificando mucho podríamos decir que:

- Moodle se adapta mejor a los cursos más basados en la interacción entre los participantes, mientras que con Dokeos poner en marcha un curso en modalidad auto-estudio con elementos de colaboración y comunicación como apoyo será más sencillo.

- Las herramientas ofrecidas por Moodle son mejores que las herramientas ofrecidas por Dokeos pero estas últimas están mejor aprovechadas con lo que Dokeos resulta ser en ese sentido mejor que Moodle.

- Dokeos tiene características de LCMS (Learning Content Management System) con propiedades de LMS (learning management system), Moodle es simplemente LMS. Se observan diferencias en cuanto a la capacidad del programa de ser evolucionado o permitir desarrollar un sistema particular a partir de él.

- Una de las características que no se encuentra en las fichas técnicas de las plataformas, pero que es necesario citar es el estilo y estructura de código. El código se encuentra mejor estructurado en Dokeos y resulta mucho más entendible que la presentada por Moodle. Por otro lado, Dokeos puede ser instalado usando una única base de datos o bien varias; creando para cada curso un grupo de tablas o una base de datos, respectivamente. Garantizando de esta manera la independencia y gestión de cada curso a nivel de base de datos, en tanto que Moodle es instalado usando una única base de datos relacional. Moodle no proporciona documentación relacionada con la base de datos, mientras que en Dokeos existe documentación acerca de la estructura de la base datos.

Todo lo anteriormente señalado permite determinar que Dokeos en su versión 1.6.4, es la plataforma seleccionada.

1.3 ANÁLISIS DE LA ARQUITECTURA Y METODOLOGÍA DEL LMS SELECCIONADO

1.3.1 DESCRIPCIÓN DE DOKEOS

Dokeos es un Learning Management Systems, es decir una plataforma de e-learning, que permite a los docentes y alumnos las funciones administrativas y académicas de la capacitación. Dokeos reúne e integra todos los componentes necesarios para permitir la gestión, administración, comunicación, evaluación y seguimiento de las actividades de enseñanza y aprendizaje en el espacio virtual.

El sistema Dokeos es desarrollado por un equipo internacional de profesores e

informáticos esparcidos por todo el mundo. La Université Catholique de Louvain alentó al Institut de Pédagogie universitaire et des Multimédias para desarrollar y distribuir este programa. El sistema por lo tanto, cuenta con innumerables implementaciones en todo el mundo y miles de alumnos que usan de sus funcionalidades. Una de sus características más notables en su versión 1.6.4 es la posibilidad de poder operar en hasta 31 idiomas (Francés, ingles, castellano, etc.) y es constantemente actualizado por la comunidad internacional la que ha efectuado sustanciales mejores en su traducción al castellano, como en sus capacidades operativas internas.

1.3.2 ESTRUCTURA DEL CÓDIGO DE DOKEOS

Dokeos consiste en diferentes piezas de funcionalidad que reciben el nombre de módulos o herramientas: por ejemplo la agenda, los documentos, los enlaces, el foro, etc. El código de cada uno de estos elementos esta situado en una carpeta separada del resto. El código de Dokeos es mayormente procedural. Lentamente está evolucionando y convirtiéndose en un código más maduro. Las funciones están siendo usadas más y más en el nuevo código, se piensa en encapsulación y en el diseño por capas, no obstante, no se utilizan objetos muy a menudo.

Todo el código que puede ser compartido entre las diferentes herramientas se encuentra recopilado en la carpeta “inc”. En el directorio inc/conf se tiene los ficheros de configuración para algunas herramientas. Fíjate que algunos de estos ficheros son tentativos, es decir, aparecen pero no interactúan con ninguna de las herramientas. Todas estas carpetas están actualmente en progreso y evolución.

A partir de la versión 1.4 de Claroline/Dokeos, la creación de nuevos módulos es un poco más fácil: solamente hay que usar tres sentencias de inclusión, y obtenemos el encabezado, el pie y las variables globales de Dokeos.

Es recomendable mantener el código nuevo tan modular como sea posible. Para ello es necesario dividirlo en diferentes secciones. Si se desea ir un paso más lejos, se puede usar programación orientada a objetos, PHP tiene soporte para ella (y está mejorado en PHP 5) y en lo posible, comentar una o dos líneas al principio de cada fichero de script explicando su propósito y finalidad.

1.3.3 API DE DOKEOS

Hay varias funciones que pueden ser útiles para los desarrolladores con el objeto de reutilización de código. Existe un Sitio Web con la API[17] del LMS Open Source Dokeos en su versión 1.6.4, en la que se explica y detalla las funcionalidades de la plataforma y se puede recurrir a ella para obtener cualquier referencia. Las funciones de librería se encuentran en la carpeta "inc" (abreviatura de include), por lo que es recomendable examinar el contenido de esta carpeta, y de las subcarpeta *inc/lib* y *inc/conf* para hacerse de una idea global de la plataforma.

Dentro del contenido del directorio *inc/lib/* existe un fichero *main_api.lib.php* que tiene unas cuantas funciones de carácter general. Hay también otras librerías por ejemplo para manipulación de texto, para el trabajo con ficheros.

1.3.4 DESCRIPCIÓN DE LA BASE DE DATOS DE DOKEOS

Dokeos puede ser instalado usando una única base de datos o bien varias. Normalmente la opción de base de datos múltiple es la opción utilizada, por lo que se crean las siguientes bases de datos para almacenar:

- La información de carácter general como usuarios y cursos.
- Las estadísticas (seguimiento) de los cursos y/o usuarios.
- El contenido SCORM[18]
- La información del usuario como la agenda personal y la categorización de los cursos del usuario.
- Base de datos para cada curso creado, es decir para cada curso se crea una base de datos con 50 tablas, bajo el siguiente esquema: crs_XXXX_table name; donde XXXX = código del curso

Si Dokeos es instalado usando una única base de datos, las bases de datos mencionados anteriormente se integran en una sola base de datos pero con tablas independientes para cada curso.

---

[17] API Dokeos v1.6.4 - www.phpxref.com/xref/dokeos/nav.html.gz?_functions/index.html.gz

[18] WIKIPEDIA, Glosario, http://es.wikipedia.org/wiki/SCORM, 2006. [**SCORM** (Sharable Content Object Reference Model) es una especificación que permite crear objetos pedagógicos estructurados.]

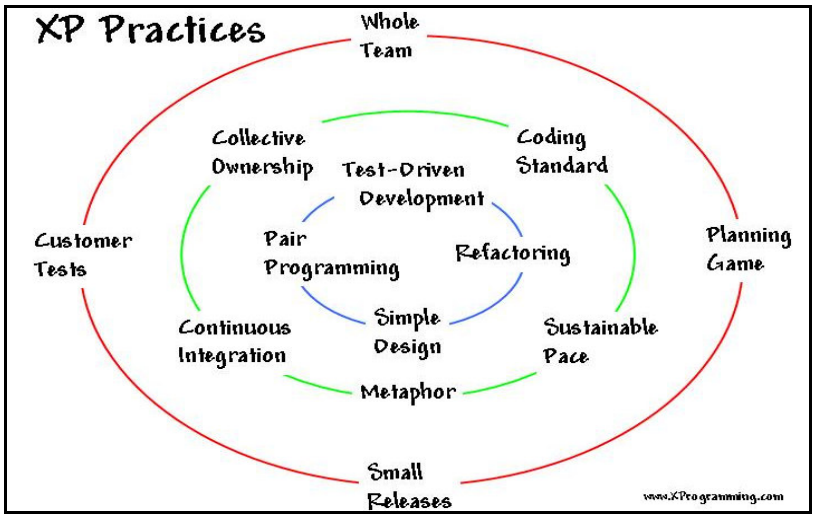
Dokeos cuenta con un documento en el que se explica la estructura de la base de datos (ver en el CD adjunto, documento: *Dokeos – Estructura BDD.pdf*).

1.3.5 MÉTODO DE TRABAJO Y METODOLOGÍA DE DISEÑO DE DOKEOS

El software se desarrolla acorde a una determinada forma de trabajar que se denomina metodología. Una metodología puede respetar una cierta filosofía o conjunto de ideas acerca del desarrollo del software. Algunas metodologías son muy estrictas, las cuales pueden ser necesarias por ejemplo en el desarrollo del software de las naves espaciales, otras sin embargo, son muy simples y flexibles. Dokeos es desarrollado siguiendo la metodología de Programación Extrema (Extreme Programming o XP). Esta metodología es simple y flexible.

Dokeos es un programa de software libre. El software libre y el software Open Source funciona tan bien porque anima a todo el mundo a participar en él. Para ello, es necesario comunicarse a menudo; y liberar las modificaciones tan pronto como sea posible: corrección de errores, nuevas piezas de código, una nueva herramienta, nueva documentación. Liberar incluso cuando algo no está completo del todo todavía, puede resumirse en: liberar pronto, liberar a menudo. La comunidad de usuarios ayudará a resolver los problemas. Con el tiempo se puede crear una pieza perfecta de código, por lo que no intentes hacerlo todo solo, con la ayuda de la comunidad conseguiremos este objetivo. Esta aproximación además elimina los errores más rápidamente que otras aproximaciones.

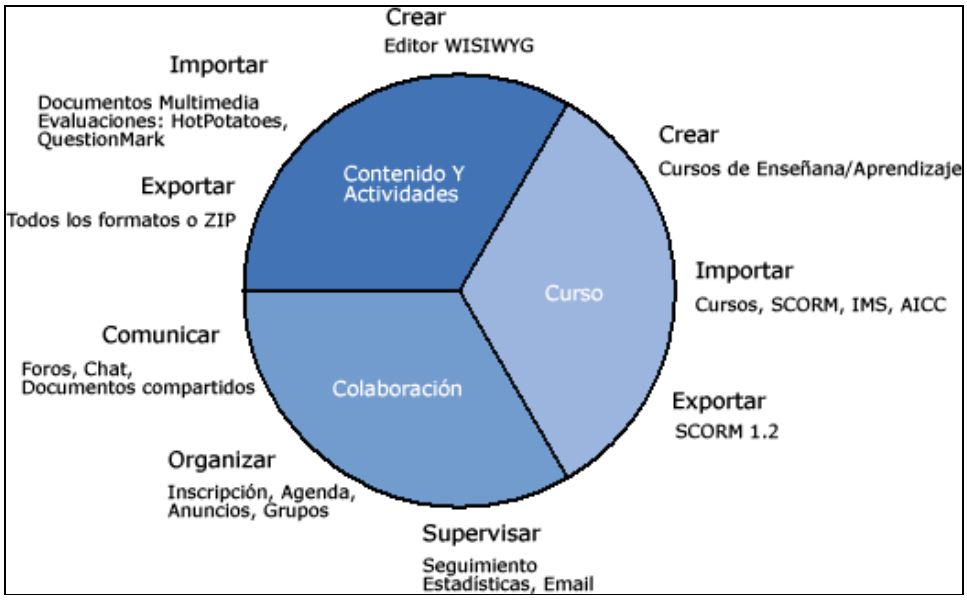
Figura 1-1: Mejores Prácticas de Extreme Programming



Fuente: Dokeos
Elaborado por: LLUGSHA, William

1.4 IDENTIFICACIÓN Y DESCRIPCIÓN DE LOS MÓDULOS EXISTENTES

Figura 1-2: Principales Módulos o funcionalidades de Dokeos v1.6.4



Fuente: Dokeos.com
Elaborado por: LLUGSHA, William

1.4.1 MODULOS DE ADMINISTRACION

En estos módulos incluyen la gestión de usuarios, cursos, carreras, etc., por los cual serán gestionados únicamente por el Administrador de la Plataforma.

1.4.1.1 Usuarios

Esta herramienta permite gestionar usuarios:

- *Búsqueda de usuarios*: Búsqueda de usuarios a través del ingreso de parámetros de búsqueda (nombre, apellido, nombre de usuario, etc.)
- *Lista de usuarios*: Despliegue de una lista de todos los usuarios registrados, con las opciones siguientes:
    - Acceso a los cursos a los que pertenece un usuario.
    - Modificación de la información del usuario
    - Eliminación de usuarios
    - Adición de usuarios: Registro individual de usuarios nuevos.

- *Exportación de usuarios a un fichero XML/CSV*: Exportación de todos los usuarios registrados en el sistema, ó los que pertenecen a un curso específico.

- *Importación usuarios desde un fichero XML/CSV*: Importación de nuevos usuarios en bloque a través de un archivo en formato XML ó CSV.

1.4.1.2 Cursos

Este módulo permite gestionar cursos:

- *Búsqueda de cursos*: Búsqueda de cursos a través del ingreso de parámetros de búsqueda (título, categoría, código, etc.)

- *Lista de cursos*: Despliegue de una lista de todos los cursos registrados, con las opciones siguientes:

    - Acceso a los usuarios que posee el curso.

    - Acceso a la página principal de un curso.

    - Modificación de la información del curso.

    - Eliminación de cursos.

- *Categorías de cursos*: Gestión de las categorías de los cursos. Estas categorías pueden ser relacionadas con las áreas del conocimiento.

- *Creación de cursos*: Permite la creación de nuevos cursos.

1.4.1.3 Plataforma

Este módulo permite gestionar aspectos relacionados con la plataforma:

- *Parámetros de configuración de la plataforma*: Esta opción permite registrar la configuración de la plataforma a nivel de los siguientes aspectos: Parámetros de la Plataforma, Curso, Idiomas, Usuarios, Herramientas, etc.

- *Anuncios del sistema*: Permite al Administrador registrar anuncios. Estos anuncios serán vistos en forma global por todos los usuarios del sistema.

1.4.2 COMPONENTES DE UN CURSO

1.4.2.1 Agenda: Manejo de cronograma de clases y eventos

Existen una serie de eventos que las instituciones educativas deben coordinar: el inicio y fin de las clases, las fechas de evaluaciones parciales y finales, entregas de trabajos, conferencias, avisos, cobranzas, etc. Los profesores a su vez establecen sus cronogramas de clase debiendo respetar el marco institucional. El sistema cuenta con una función que provee la posibilidad de administrar todas estas cuestiones.

1.4.2.2 Documentos

Esta sección mantiene un mecanismo comprensivo para organizar los archivos tal como le gustaría hacerlos disponible a los estudiantes. Es posible hacer muchos directorios, y tener directorios dentro de los directorios y en ellos agrupar los archivos. Es posible: Borrar, Desplazar (Mover), Renombrar, Agregar un comentario, Impedir que los estudiantes vean los archivos (Visible / Invisible), Subir los archivos a un directorio

1.4.2.3 Anuncios

La herramienta de Anuncios permite enviar un anuncio y/o correo electrónico a todos los alumnos, a algunos de ellos o algún grupo. Puede ser una manera eficiente de conducir a los alumnos de vuelta al sitio web si ellos no lo visitan muy a menudo.

1.4.2.4 Chat

Como los alumnos no están en contacto con el profesor presencialmente, el Chat le permite comunicarse con él y mantener una conversación en línea y realizar consultas y aclaraciones.

1.4.2.5 Foro

Esta opción permite que se realicen consultas entre alumnos y profesores en forma asincrónica. Los foros permiten intercambiar opiniones dándose tiempo para contestar. Si un email permite un dialogo uno a uno, los foros permiten un dialogo público o semipúblico.

1.4.2.6 Baúl de tareas (dropbox)

El ciclo de intercambios de archivos entre docentes y alumnos tiene en esta herramienta todo los que necesita. Esto hará innecesario el uso de emails con adjuntos. En este sentido el dropbox es una herramienta de administración de contenidos dedicada al intercambio de datos punto a punto. Cualquier tipo de fichero es aceptado: Word, Excel, PDF etc. Se mantienen versiones de documentos para evitar la destrucción documentos que tienen el mismo nombre.

1.4.2.7 Conferencia on-line

Este sistema permite enseñar hasta 100 personas juntas de una manera simple.

- **Audio**: Las voz del profesor es llevada a los participantes mediante un streaming de mp3
- **Contenido**: Los participantes siguen una presentación de Power Point o de cualquier otro documento
- **Interacción**: los participantes hacen las preguntas al profesor por medio de web chat

1.4.2.8 Publicaciones de los estudiantes (Trabajos de los estudiantes)

Permite a todos los usuarios docentes y estudiantes subir documentos para ser compartidos con el resto. Dependiendo del escenario del curso, se tiene que decidir si se quiere que todos los alumnos vean todos los documentos o ser tú el único que pueda verlos.

1.4.2.9 Usuarios

En esta sección está la lista de los cursantes, donde puede controlar quien está registrado en su curso. Usted no puede modificar la información del estudiante como el nombre o contraseña porque depende de ellos actualizar su propio perfil.

La herramienta de los usuarios se conecta a la herramienta de los Grupos por las razones obvias (los grupos son grupos de los usuarios). Sin embargo, le permitirán crear grupos vacíos si sus estudiantes no están todavía registrados. De esta manera, puede organizar la estructura de su curso antes de empezarlo.

1.4.2.10 Grupos

Usted puede crear grupos de estudiantes. Un grupo es una colección de estudiantes que comparten el mismo foro y/o la misma área de documentos.

Como docente, se le permiten entrar en cada foro y área de documento del grupo. Los tutores entrarán en cualquier grupo también, pero la lista de los foros mencionará a qué grupos ellos supervisan.

1.4.2.11 Ejercicios: Generador de cuestionarios

Esta herramienta es una función del sistema usado para construir exámenes y encuestas. El generador incluye las herramientas que crean preguntas, asignan valores a las preguntas, y permiten el feedback para cada pregunta.

Permitiendo la generación de pruebas de opción múltiples (Multiple Choice), de una respuesta correcta o de varias respuestas correctas, completar en el blanco y también concordancias (Matching). El sistema permite definir el peso apropiado para número de respuestas correctas por pregunta. Por ejemplo, si se tiene una pregunta con 4 respuestas (posiblemente más de una correcta) y el estudiante tiene dos mal y dos correctas, usted puede darle la mitad de los puntos, pero también puede decidir que esto no es satisfactorio y dar, por ejemplo, el máximo apunta (20) si todo es correcto y cero punto en cualquier otra combinación.

El sistema permite incorporar imágenes y sonidos en la evaluación.

1.4.2.12 Estadísticas (Tracking)

Sólo los administradores y profesores pueden ver las estadísticas y pueden pulsar el botón en este enlace que les permite ver cuánto uso ha tenido el curso, junto con otra información.

El seguimiento es uno de los ejes principales para un aprendizaje exitoso. En Dokeos esto significa que:

- **Para el estudiante:** un conocimiento más profundo de su rendimiento;
- **Para el profesor:** una interacción más precisa con el estudiante,

- **Para el supervisor:** una supervisión cualitativa y cuantitativa del proceso de aprendizaje.

1.4.2.13 Descripción del Curso

El sistema cuenta con un asistente para elaborar el programa del curso o materia. El sistema da la opción de agregar información sobre el curso, objetivos, programa de estudios, etc. Mediante una serie de preguntas y recomendaciones por pantalla se va guiando al profesor para permitirle hacer o perfeccionar la comunicación de su programa de estudio a la comunidad educativa.

1.4.2.14 Enlaces

La herramienta de enlace permite crear una librería de recursos para los alumnos. Especialmente recursos que no han sido creados por el profesor.

Cuando la lista de enlaces crece, puede ser interesante organizarlos en categorías para ayudar a los alumnos a encontrar la información correcta en el lugar adecuado. Es posible editar cada enlace para reasignarlo a una nueva categoría (aunque antes es necesario crear la categoría primero).

El campo descripción puede ser usado para dar información sobre el destino de la página web (link) pero también puede usarse para describir que es lo que esperas que tus alumnos hagan con el enlace. Si, por ejemplo, si el sitio web contiene información sobre Aristóteles, se le puede indicar al alumno que estudien la diferencia entre la síntesis y el análisis.

1.4.2.15 Itinerario de Aprendizaje

La herramienta “Itinerarios de Aprendizaje” tiene dos funciones: la primera, crear un Itinerario de aprendizaje y la otra subir un Itinerario de aprendizaje en formato Scorm o IMS. El Itinerario de aprendizaje es una sucesión de pasos de aprendizaje incluidos dentro de un módulo. Pueden sustentarse en contenidos (con una apariencia similar a una tabla de contenidos) o fundamentarse en actividades.

CAPITULO 2. SELECCIÓN DE LA METODOLOGÍA Y HERRAMIENTAS DE DESARROLLO

2.1 SELECCIÓN DE LA METODOLOGÍA

En este capítulo se efectuará una descripción del problema en forma global, y con esta descripción o caracterización, iniciar con la descripción de dos metodologías de desarrollo posibles a seleccionar. En esta sección se analizarán: Rational Unified Process (RUP)[19], y Extreme Programming (XP)[20].

Es preciso mencionar que, la selección de la metodología dependerá del tipo y requerimientos del proyecto. La metodología seleccionada será la que más se ajuste a dichos requerimientos y en la que se basará el desarrollo del proyecto (ciclo de vida del proyecto).

2.1.1 DESCRIPCIÓN DEL PROBLEMA

En la sección 1.3 y 1.4 se realizó un análisis sobre el LMS seleccionado. En la sección 1.3 el análisis se basó en la arquitectura y metodología de desarrollo de Dokeos en su versión 1.6.4, y por otro lado en la sección 1.4, se identificaron los módulos con las que cuenta actualmente la plataforma. Es preciso mencionar que la versión 1.6.4 de Dokeos es la versión más estable.

Además, tomando en cuenta que este proyecto tiene como objetivo desarrollar e incorporar nuevos módulo o funcionalidades en el LMS seleccionado, con la finalidad de que apoye el proceso de enseñanza aprendizaje en programas de pregrado y postgrado de educación superior. Estos nuevos módulos o componentes interactuarán con la plataforma Dokeos v1.6.4 con el objetos de que dicha plataforma se adapte a una institución de educación superior tanto en la modalidad presencial como la modalidad a distancia. En este sentido, se ve necesario que dentro la plataforma Dokeos v1.6.4, la creación de módulos de gestión de carreras o programas académicos de pregrado y postgrado.

Debido a que se administrará carreras o programas académicos, es necesario la gestión de períodos académicos y mallas curriculares, es decir la gestión de pre-

[19] Proceso Racional Unificado – Metodología de desarrollo creada por IBM
[20] Programación Extrema – Metodología de desarrollo Ágil. Metodología creada por Kent Beck

requisitos y co-requisitos para un curso o asignatura perteneciente a la carrera o programa académico.

Por otro lado, como herramienta o componente de un curso es necesario la incorporación de un módulo de publicación de calificaciones en diferentes categorías (pruebas, exámenes, exposiciones, etc.) por parte del profesor.

Con todas estas incorporaciones dentro del LMS Dokeos, y teniendo en cuenta los requerimientos del cliente, es necesario la implementación de un módulo de Servicios en Línea, que permita al usuario matricularse en una carrera o programa académico, y con ello tener la opción de ver el estado de sus matrículas, acceder a su currículum o expediente académico en dicha carrera.

Auspiciante, cliente o interesado:

- Miembros del grupo de tesis (Director y graduando).
- Los requerimientos están definidos por el grupo de desarrollo del proyecto.

Recursos:

- Humano: 2 personas para el proyecto (Director y graduando).
- Tiempo de duración: 3 meses.
- Económico: limitado
- Tecnológico:
    - Hardware: Se cuenta con 2 computadores para el desarrollo. El un computado actuará como servidor de aplicaciones y de base de datos; y el otro como equipo de desarrollo de los módulos.
    - Software: En el mercado existen herramientas open source que pueden ayudar al desarrollo del proyecto.
- Información: Para el desarrollo del proyecto, DOKEOS[21] cuenta con un portal en la que se incluye la documentación necesaria, foros, un API[22] y una comunidad de usuarios. Por otro lado, en la biblioteca de la EPN cuenta tesis de grado con temas relacionados con el tema.

[21] Dokeos – LMS. www.dokeos.org
[22] API de Dokeos v1.6.4 - www.phpxref.com/xref/dokeos/nav.html.gz?_functions/index.html.gz

Estructura tecnológica del proyecto

- Equipos:
  - Servidor: sistema operativo multiplataforma (Linux ó Windows)
  - Equipo de desarrollo bajo la Plataforma Windows.
- Lenguaje de desarrollo: PHP
- Es necesaria la utilización de servidores Web, Aplicación, Base de datos.
- Disponibilidad del sistema 24 x 7.
- Permitirá la comunicación a través de los módulos existentes en la plataforma.
- Sistema multicapa (servidor de aplicaciones, base de datos, capa de negocio).
- Debe soportar un gran número de usuarios.
- Es necesario la entrega de documentación y guías del proyecto.
- Programación orientada a objetos.

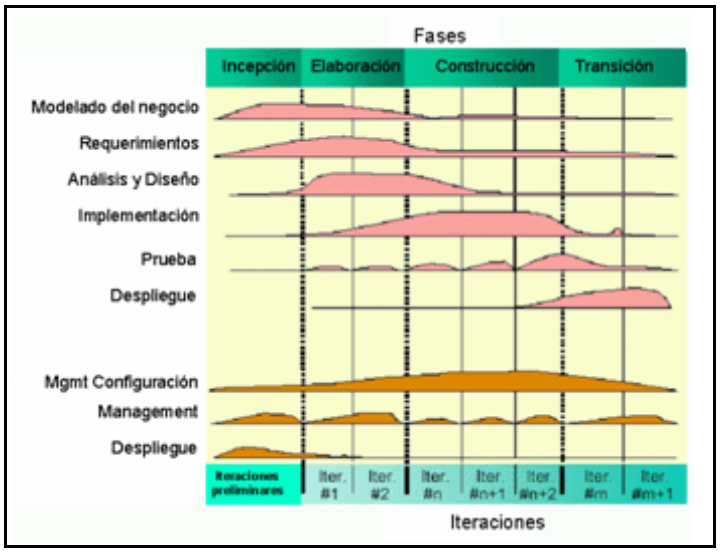
2.1.2 DESCRIPCIÓN DE LA METODOLOGIA DE DESARROLLO: RATIONAL UNIFIED PROCESS – RUP

El RUP es un producto de Rational (IBM). Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el codigo fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

2.1.2.1 Fases

El proceso de ciclo de vida de RUP se divide en cuatro fases bien conocidas llamadas Incepción, Elaboración, Construcción y Transición. Esas fases se dividen en iteraciones, cada una de las cuales produce una pieza de software demostrable. La duración de cada iteración puede extenderse desde dos semanas hasta seis meses. Las fases son:

Figura 2-1: Fases y workflows de RUP, basado en [BMP98]



Fuente: Jacobsob Ivar, Booch Grady, Rumbaugh James, El proceso Unificado de Desarrollo de software
Elaborado por: LLUGSHA, William

2.1.2.1.1 Incepción

Significa “comienzo”, pero la palabra original (de origen latino y casi en desuso como sustantivo) es sugestiva y por ello la traducimos así. Se especifican los objetivos del ciclo de vida del proyecto y las necesidades de cada participante. Esto entraña establecer el alcance y las condiciones de límite y los criterios de aceptabilidad. Se identifican los casos de uso que orientarán la funcionalidad. Se diseñan las arquitecturas candidatas y se estima la agenda y el presupuesto de todo el proyecto, en particular para la siguiente fase de elaboración. Típicamente es una fase breve que puede durar unos pocos días o unas pocas semanas.

2.1.2.1.2 Elaboración.

Se analiza el dominio del problema y se define el plan del proyecto. RUP presupone que la fase de elaboración brinda una arquitectura suficientemente sólida junto con requerimientos y planes bastante estables. Se describen en detalle la infraestructura y el ambiente de desarrollo, así como el soporte de herramientas de automatización. Al cabo de esta fase, debe estar identificada la mayoría de los casos de uso y los actores, debe quedar descripta la arquitectura de software y se debe crear un prototipo de ella. Al final de la fase se realiza un análisis para determinar los riesgos y se evalúan los gastos hechos contra los originalmente planeados.

2.1.2.1.3 Construcción.

Se desarrollan, integran y verifican todos los componentes y rasgos de la aplicación. RUP considera que esta fase es un proceso de manufactura, en el que se debe poner énfasis en la administración de los recursos y el control de costos, agenda y calidad. Los resultados de esta fase (las versiones alfa, beta y otras versiones de prueba) se crean tan rápido como sea posible. Se debe compilar también una versión de entrega. Es la fase más prolongada de todas.

2.1.2.1.4 Transición.

Comienza cuando el producto está suficientemente maduro para ser entregado. Se corrigen los últimos errores y se agregan los rasgos pospuestos. La fase consiste en prueba beta, piloto, entrenamiento a usuarios y despacho del producto a mercadeo, distribución y ventas. Se produce también la documentación. Se llama transición porque se transfiere a las manos del usuario, pasando del entorno de desarrollo al de producción.

2.1.2.2 Prácticas comunes de RUP

A través de las fases se desarrollan en paralelo nueve workflows o disciplinas: Modelado de Negocios, Requerimientos, Análisis & Diseño, Implementación, Prueba, Gestión de Configuración & Cambio, Gestión del Proyecto y Entorno. Además de estos workflows, RUP define algunas prácticas comunes:

2.1.2.2.1 Desarrollo iteractivo de software.

Las iteraciones deben ser breves y proceder por incrementos pequeños. Esto permite identificar riesgos y problemas tempranamente y reaccionar frente a ellos en consecuencia.

2.1.2.2.2 Administración de requerimientos.

Identifica requerimientos cambiantes y postula una estrategia disciplinada para administrarlos.

2.1.2.2.3 Uso de arquitecturas basadas en componentes.

La reutilización de componentes permite asimismo ahorros sustanciales en tiempo, recursos y esfuerzo.

2.1.2.2.4 Modelado visual del software.

Se deben construir modelos visuales, porque los sistemas complejos no podrían comprenderse de otra manera. Utilizando una herramienta como UML, la arquitectura y el diseño se pueden especificar sin ambigüedad y comunicar a todas las partes involucradas.

2.1.2.2.5 Prueba de calidad del software.

RUP pone bastante énfasis en la calidad del producto entregado.

2.1.2.2.6 Control de cambios y trazabilidad.

La madurez del software se puede medir por la frecuencia y tipos de cambios realizados.

2.1.3 DESCRIPCIÓN DE LA METODOLOGÍA DE DESARROLLO: EXTREME PROGRAMMING – XP

2.1.3.1 Valores promovidos por XP

2.1.3.1.1 Comunicación

XP se nutre del ancho de banda más grande que se puede obtener cuando existe algún tipo de comunicación: la comunicación directa entre personas. Es muy importante entender cuales son las ventajas de este medio. Cuando dos (o más) personas se comunican directamente pueden no solo consumir las palabras formuladas por la otra persona, sino que también aprecian los gestos, miradas, etc. que hace su compañero. Sin embargo, en una conversación mediante el correo electrónico, hay muchos factores que hacen de esta una comunicación, por así decirlo, mucho menos efectiva.

2.1.3.1.2 Coraje

El coraje es un valor muy importante dentro de la programación extrema. Un miembro de un equipo de desarrollo extremo debe de tener el coraje de exponer sus dudas, miedos, experiencias sin "embellecer" éstas de ninguna de las maneras. Esto es muy importante ya que un equipo de desarrollo extremo se basa en la confianza para con sus miembros. Faltar a esta confianza es una falta más que grave.

2.1.3.1.3 Simplicidad

Dado que no se puede predecir como va a ser en el futuro, el software que se esta desarrollando; un equipo de programación extrema intenta mantener el software lo más sencillo posible. Esto quiere decir que no se va a invertir ningún esfuerzo en hacer un desarrollo que en un futuro pueda llegar a tener valor. En el XP frases como "...en un futuro vamos a necesitar..." o "Haz un sistema genérico de..." no tienen ningún sentido ya que no aportan ningún valor en el momento.

2.1.3.1.4 Feedback

La agilidad se define (entre otras cosas) por la capacidad de respuesta ante los cambios que se van haciendo necesarios a lo largo del camino. Por este motivo uno de los valore que nos hace más ágiles es el continuo seguimiento o feedback que recibimos a la hora de desarrollar en un entorno ágil de desarrollo. Este feedback se toma del cliente, de los miembros del equipo, en cuestión de todo el entorno en el que se mueve un equipo de desarrollo ágil.

2.1.3.2 Prácticas de XP

Según los valores promovidos por XP se fundamenta en las siguientes doce prácticas:

- Planificación incremental
- Testing
- Programación en parejas
- Refactorización
- Diseño simple
- Propiedad colectiva del código
- Integración continua
- Cliente en el equipo
- Releases pequeñas
- Semanas de 40 horas
- Estándares de codificación
- Uso de Metáforas

2.1.4 COMPARACIÓN DE LAS METODOLOGÍAS: XP y RUP

La comparación entre XP y RUP han sido tomadas de Javahispano [8] y se citan a continuación:

A diferencia de XP, RUP tiene carga excesiva de documentación que en una buena parte de los casos nunca se utiliza y que hace que el desarrollador no se enfoque en el código que es lo más importante del proyecto.

Sin embargo, RUP y XP también trabajan por medio de ciclos o iteraciones de desarrollo, en el caso de XP corresponden a un conjunto de historias de usuario.

Un ciclo o una iteración en un período de tiempo en el que se realiza un conjunto de funcionalidades determinadas que en el caso de XP corresponden a un conjunto de historias de usuario. Los ciclos aquí son particularmente cortos y que se piensa que entre más rápido se le entreguen desarrollos al cliente, más retroalimentación se va a tener y esto va a representar una mayor calidad del producto a largo plazo. Existe una fase de análisis inicial encaminada a programar los ciclos de desarrollo y cada ciclo contiene diseño codificación y pruebas, con la diferencia que aquí no son etapas separadas sino que están superpuestas de tal manera que no es correcto separar un proyecto XP en estas fases porque el esquema de trabajo no se divide con el tiempo de esta forma, cosa que si sucede con RUP.

A continuación se expondrán diferencias entre RUP y XP (ver en detalle en el CD adjunto, documento: *Comparación RUP - XP.doc*):

Tabla 2-1: Comparación entre RUP y XP

	Extreme Programming (XP)	**Racional Unified Process (RUP)**
Tamaño de los equipos	Proyectos cortos y equipos más pequeños	Proyectos y equipos grandes, en cuanto a tamaño y duración
Obtención de requisitos	Se basa en UserStories	RUP crean como base UseCases.
Carga de trabajo	XP es un proceso ligero. En el desarrollo de un proyecto con XP es más importante la entrega al cliente del software que necesita	RUP es un proceso pesado. RUP define en cada momento del ciclo de vida del proyecto.

Relación con el cliente	Fuerte comunicación con el cliente. Feedback del cliente.	Se presentarán al cliente los artefactos del final de una fase y solo después de que el cliente acepte los artefactos generados se pasará a la siguiente fase.
Desarrollo	Las iteraciones de XP tienen por lo general una duración menor que en RUP, puesto que la carga a llevar por los programadores a parte del desarrollo del propio software es menor.	Proceso Iterativo. RUP genera también releases basados en los artefactos después de cada fase, pero en su caso no se limitan solo al código, si no que las releases viene acompañada de todo lo que traería el producto final, es decir, notas de la versión, instrucciones de instalación, ayuda de uso, etc.
Código fuente	XP es el único que presenta la compartición del código.	RUP opta por la propiedad del código
Conocimiento sobre la arquitectura	Programación en parejas. Discusiones que lleven a mejores estructuras y algoritmos y que este proceso aumente la calidad del software.	En RUP se intentará reducir la complejidad del software a producir a través de una planificación intensiva.
Evaluación del estado del proyecto	XP define esos componentes pequeños para hacer un seguimiento de las mismas, pero la tarea del reporting recae solo en los jefes de proyecto.	RUP por su parte, es tan grande y complejo en este sentido como en el resto, por lo que manejar el volumen de información que puede generar requiere mucho tiempo.
Puntos flacos	Pobre en cuanto a documentación debido a que XP es un proceso muy orientado a la implementación.	Mucha documentación y en ocasiones para el desarrollo de software por medio de equipos pequeños es RUP muy grande y prácticamente inalcanzable

Fuente: http://www.javahispano.org/download/articulos/metodos_desarrollo.pdf, 2002
Elaborado por: LLUGSHA, William

En relación a las mejores prácticas de cada una de las metodologías descritas anteriormente, se puede concluir y garantizar la calidad del software. RUP tienen un modelo de desarrollo más elaborado, siendo XP especializado en proyectos pequeños y medianos, y que están conformados por un equipo de desarrollo pequeño.

Por otro lado, Dokeos es desarrollado siguiendo la metodología de Programación Extrema (Extreme Programming o XP) ya que esta metodología es simple y flexible. La tabla 2-2, servirá como referencia para la selección de la metodología de desarrollo del proyecto.

Tabla 2-2: Comparación de RUP y XP en relación a Dokeos 1.6.4

	RUP	XP
Metodología de desarrollo de Dokeos v1.6.4		✓
Estándares de codificación	✓	✓
Propiedad colectiva del código		✓
Desarrollo iteractivo de software	✓	✓
Implementación Sencilla y Diseño simple		✓
Verificación continua de la calidad	✓	✓
Requerimientos del cliente	✓	✓
Enfocado en equipo	✓	✓
Programación en parejas		✓
Control de cambios y trazabilidad	✓	
Gestión del riesgo	✓	
Experiencia de los desarrolladores	✓	✓

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

Por las características que tiene este proyecto y por las características enunciadas en la Tabla 2-2 se puede optar por desarrollar XP para seguir con la misma metodología de desarrollo en la que se basa Dokeos como LMS seleccionado. Además, Dokeos es un LMS de software libre. El software libre y el software Open Source funciona tan bien porque anima a todo el mundo a participar en él. Para ello, es necesario comunicarse a menudo; y liberar las modificaciones tan pronto como sea posible: corrección de errores, nuevas piezas de código, una nueva herramienta, nueva documentación. Liberar incluso cuando algo no está completo del todo todavía, puede resumirse en: liberar pronto, liberar a menudo. La comunidad de usuarios ayudará como soporte.

Una vez seleccionado la metodología de desarrollo, en el siguiente subcapítulo se procederá a explicar en detalle a Extreme Programing

2.1.5 EXPLICACIÓN DE LA METODOLOGÍA DE DESARROLLO SELECCIONADA: EXTREME PROGRAMMING – XP

2.1.5.1 Definición

XP (*eXtreme Programming*) es una nueva disciplina para el desarrollo de software, que ha irrumpido recientemente con gran revuelo en los métodos, técnicas y metodologías existentes. Concretando más, se trata de una

metodología *«ligera»*, en contraposición a las metodologías «pesadas» como Métrica.

La Programación Extrema es una metodología de desarrollo de software que se basa en la simplicidad, la comunicación y la retroalimentación o reutilización del código desarrollado (reciclado de código). Cuenta con tan solo 6 años de vida, pero con un gran respaldo por parte de grandes empresas cómo la Ford, DaimlerChrysler, First Union National Bank -USA-, etc., que lo que buscan en definitiva es la reducción de costes.

2.1.5.1.1 Un cambio en la manera de programar

El software diseñado para ser simple y elegante no es menos valioso que aquel que es complejo y difícil de mantener. Un proyecto típico consume del orden de 20 veces más en recursos humanos que en hardware. Pongamos un ejemplo, si un proyecto cuesta 2 millones de dólares (o euros) al año en programadores, costará en equipos (mantenimiento, etc.) en torno a los 100.000 dólares (o euros). Supongamos por un momento que somos programadores listos y encontramos la forma de ahorrar a la compañía un 20 % de los gastos en los equipos, explotando nuestros conocimientos acerca de la arquitectura, redes, etc. Pero esto, hará que el código sea complejo y difícil de mantener. Sin embargo, supongamos ahora que en lugar de esto, construimos nuestros programas de forma que sean sencillos de entender y actualizar; consiguiendo un ahorro de personal en torno a un 10%. Esta segunda opción gustará, sin duda, mucho más al cliente (concepto básico de proporciones).Otro aspecto a tener en cuenta es la localización de los temidos bugs, o fallos en el programa. XP pone especial hincapié en el testeo de nuestros programas. Crearemos test antes de la implementación, durante y cuando hayamos concluido. Cuando se detecte un fallo, se crearán nuevos test. No puede aparecer dos veces el mismo fallo. Una buena noticia para los usuarios será la apertura de los programadores ante los cambios en los requerimientos (XP es una mentalidad de trabajo). Los usuarios nos notificarán los cambios que sean necesarios para que nuestro sistema se adapte perfectamente a sus necesidades. Lo más importante es la calidad del software, por mucho que le mostremos a un usuario lo bonito de nuestros bucles y tabulaciones en el código fuente, jamás llegará a comprender el esfuerzo realizado en la construcción del mismo.

Planteemos el siguiente problema. Deseamos construir un puente sobre un río. Dicho puente comienza a construirse desde ambas orillas, con el fin de encontrarse en el punto medio. Pero cuando este hecho se produce, nos damos cuenta de que existen unos 60 cm de diferencia, por ejemplo. De la misma manera aplicamos esto al software, no pueden producirse estos problemas a la hora de la integración.

XP surge como solución a estos problemas. Se basa en observar qué es lo que hace que el desarrollo de un programa sea rápido o lento. XP es una metodología importante por dos razones. Primero y principal, porque constituye un método de control para las actividades de desarrollo de software que se han convertido en métodos operativos estándar. Y segundo, es una de las pocas y nuevas metodologías ligeras desarrolladas para reducir el coste del software. XP va un paso más allá, definir un proceso simple y satisfactorio para la implementación de software.

2.1.5.2 Introducción a la metodología XP

Podríamos decir que XP nace «oficialmente» hace cinco años en un proyecto desarrollado por Kent Beck en DaimlerChrysler, después de haber trabajado varios años con Ward Cunningham en busca de una nueva aproximación al problema del desarrollo de software que hiciera las cosas más simples de lo que nos tenían acostumbrados los métodos existentes. Para muchos, XP no es más que sentido común.

Kent definió cuatro grandes tareas a realizar en el desarrollo de todo proyecto: *planificación, diseño, desarrollo* y *pruebas*; teniendo siempre presente las cuatro características básicas que debe reunir un programador XP: simplicidad en el desarrollo, comunicación entre las partes implicadas, realimentación para poder reutilizar y coraje y que fue definido en la sección.

2.1.5.2.1 Objetivos

El objetivo principal que persigue XP es la satisfacción del cliente. Esta metodología fue diseñada para proporcionar el software que el cliente necesita cuando lo necesite. Deberemos responder de forma rápida a los cambios en las

necesidades del cliente, incluso cuando estos cambios se produzcan al final del ciclo de vida de dicho software (simplicidad y realimentación).

El segundo objetivo es potenciar al máximo el trabajo en equipo. Tanto los Jefes de Proyecto, como los clientes y desarrolladores, son parte del equipo encargado de la implementación de software de calidad (comunicación). Esto implicará que los diseños deberán ser claros y sencillos. Y los clientes deberán disponer de versiones operativas cuanto antes para poder participar en el proceso creativo mediante sus sugerencias y aportaciones.

- El código será revisado continuamente, mediante la programación en parejas (dos personas por máquina).

- Se harán pruebas todo el tiempo, no sólo de cada nueva clase (pruebas unitarias) sino que también los clientes comprobarán que el proyecto va satisfaciendo los requisitos (**pruebas funcionales**).

- Las pruebas de integración se efectuarán siempre, antes de añadir cualquier nueva clase al proyecto, o después de modificar cualquiera existente (**integración continua**), para lo que nos serviremos de f*rameworks de testing*, como el JUnit[23] ó PhpUnit[24].

- Se (re)diseñará todo el tiempo (refactoring), dejando el código siempre en el estado más simple posible.

- Las iteraciones serán radicalmente más cortas de lo que es usual en otros métodos, de manera que nos podamos beneficiar de la retroalimentación tan a menudo como sea posible.

«Todo en el software cambia. Los requisitos cambian. El diseño cambia. El negocio cambia. La tecnología cambia. El equipo cambia. Los miembros del equipo cambian. El problema no es el cambio en sí mismo, puesto que sabemos que el cambio va a suceder; el problema es la incapacidad de adaptarnos a dicho

[23] WIKIPEDIA, Glosario, http://es.wikipedia.org/wiki/JUnit, 2006. [**JUnit** es un conjunto de librerías creadas por Erich Gamma y Kent Beck que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java]
[24] SOURCEFORGE, http://phpunit.sourceforge.net. [**PhpUnit** es un framework para ejecutar pruebas de unidad para aplicaciones PHP basado en el framework **JUnit** para Java.

cambio cuando éste tiene lugar» Kent Beck.

2.1.5.2.2 Variables

XP define cuatro variables para cualquier proyecto software: *coste, tiempo, calidad y alcance*.

Además, especifica que, de estas cuatro variables, sólo tres de ellas podrán ser fijadas por las fuerzas externas al proyecto (clientes y jefes de proyecto), mientras que el valor de la variable libre será establecido por el equipo de desarrollo en función de los valores de las otras tres.

Normalmente los clientes y jefes de proyecto creen ser capaces de fijar de antemano el valor de todas las variables. Cuando esto ocurre, la calidad es lo primero que se esfuma de la ecuación. Y esto por una sencilla razón, que frecuentemente se ignora: nadie es capaz de trabajar bien cuando está sometido a mucha presión. XP hace a las cuatro variables visibles para todo el mundo (programadores, clientes y jefes de proyecto), de manera que se pueda jugar con los valores de la entrada hasta que la cuarta variable tenga un valor que satisfaga a todos (pudiendo escoger otras variables diferentes a controlar, por supuesto). Ocurre además que las cuatro variables no guardan entre sí una relación tan obvia como a menudo se quiere ver.

XP hace especial énfasis en equipos de desarrollo pequeños (diez o doce personas como mucho) que, naturalmente, se podrán ir incrementando a medida que sea necesario, pero no antes, o los resultados serán generalmente contrarios a lo esperado.

Con la **calidad** también sucede otro fenómeno extraño: frecuentemente, *aumentar la calidad conduce a que el proyecto pueda realizarse en menos tiempo*. En efecto, en cuanto el equipo de desarrollo se habitúe a realizar pruebas intensivas y se sigan estándares de codificación, poco apoco comenzará a avanzar mucho más rápido de lo que lo hacía antes, mientras la calidad del proyecto se mantiene asegurada por las pruebas al 100%, lo que conlleva mayor confianza en el código y, por tanto, mayor facilidad para adaptarse al cambio, sin estrés, lo que hace que se programe más rápido.

Frente a esto, está la tentación de sacrificar la calidad interna del proyecto la que es apreciada por los programadores para reducir el tiempo de entrega del proyecto, en la confianza de que la calidad externa no se vea demasiado afectada. Sin embargo, ésta es una apuesta a muy corto plazo, que suele ser una invitación al desastre, pues obvia el hecho fundamental de que *todo el mundo trabaja mucho mejor cuando le dejan hacer trabajo de calidad*.

No tener esto en cuenta conduce a la desmoralización del equipo y, con ello, a la ralentización del proyecto mucho más allá del tiempo que hubiera podido ganarse al principio con esta reducción de calidad. En cuanto al **alcance** del proyecto, es una buena idea dejar que sea esta la variable libre, de manera que una vez fijadas las otras tres, el equipo de desarrollo determinaría el alcance mediante:

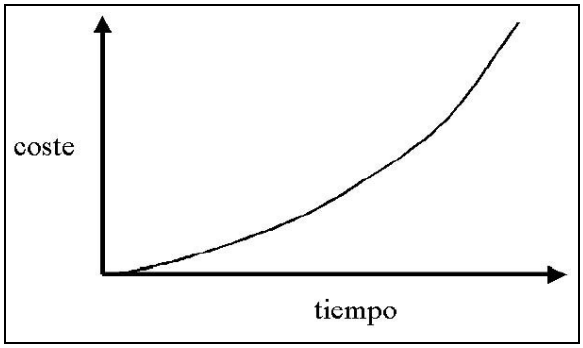
- La estimación de las tareas a realizar para satisfacer los requisitos del cliente.

- La implementación de los requisitos más importantes primero, de manera que el proyecto tenga en cada instante tanta funcionalidad como sea posible.

2.1.5.2.3 El coste del cambio

Es importante al menos reseñar una de las asunciones más importantes e innovadoras que hace XP frente a la mayoría de los métodos conocidos, y es la referida al coste del cambio. En efecto, siempre ha sido una verdad universal el hecho de que el coste del cambio en el desarrollo de un proyecto se incrementaba exponencialmente en el tiempo:

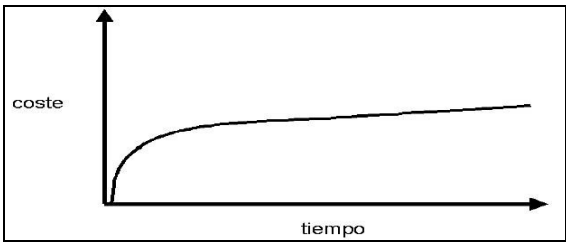
Figura 2-2: Coste del cambio - metodologías tradicionales



Fuente: www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-XP.pdf
Elaborado por: LLUGSHA, William

Lo que XP propugna es que esta curva ha perdido validez y que con una combinación de buenas prácticas de programación y tecnología es posible lograr que la curva sea la contraria. Con la metodología XP, se pretende conseguir:

Figura 2-3: Coste del cambio - metodologías XP



Fuente: www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-XP.pdf
Elaborado por: LLUGSHA, William

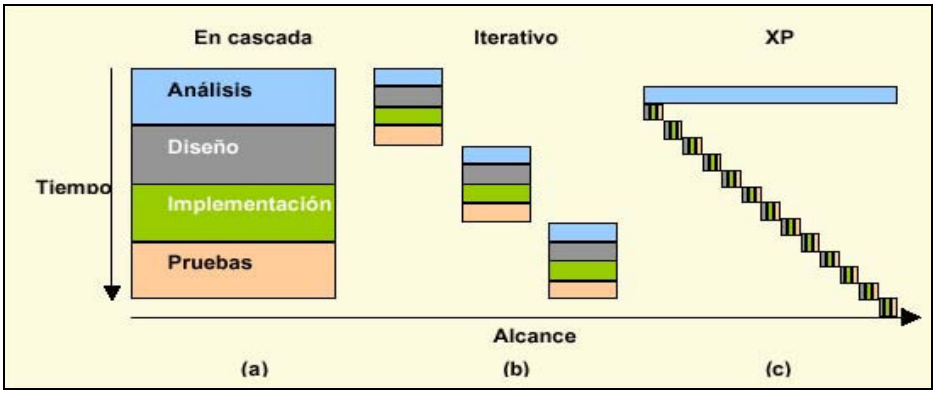
Si decidimos emplear XP como proceso de desarrollo de software, deberemos adoptarlo basándonos en dicha curva.

La idea fundamental aquí es que, en vez de diseñar para el cambio, diseñaremos tan sencillo como sea posible, para hacer sólo lo que sea imprescindible en un momento dado, pues la propia simplicidad del código, y, sobretodo, los tests y la integración continua, hacen posible que los cambios puedan ser llevados a cabo tan a menudo como sea necesario.

2.1.5.2.4 Ciclo de vida

Como se ha demostrado, los largos ciclos de desarrollo de los métodos tradicionales son incapaces de adaptarse al cambio, tal vez lo que haya que hacer sea ciclos de desarrollo más cortos. Esta es una de las ideas centrales de XP (Figura 2-4).

Figura 2-4: Evolución de largos ciclos de desarrollo en cascada (a) a ciclos más cortos (b) y a la mezcla que hace XP (c)



Fuente: www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-XP.pdf
Elaborado por: LLUGSHA, William

2.1.5.3 Prácticas de XP

Según los valores promovidos por XP se fundamenta en las siguientes doce prácticas:

2.1.5.3.1 Planificación incremental

La Programación Extrema asume que la planificación nunca será perfecta, y que variará en función de cómo varíen las necesidades del negocio. Por tanto, el valor real reside en obtener rápidamente un plan inicial, y contar con mecanismos de feedback que permitan conocer con precisión dónde estamos. Como es lógico, la planificación es iterativa: un representante del negocio decide al comienzo de cada iteración qué características concretas se van a implementar.

El objetivo de la XP es generar versiones de la aplicación tan pequeñas como sea posible, pero que proporcionen un valor adicional claro, desde el punto de vista del negocio. A estas versiones se las denomina releases.

Una release cuenta con un cierto número de historias. La historia es la unidad de funcionalidad en un proyecto XP, y corresponde a la mínima funcionalidad posible que tiene valor desde el punto de vista del negocio. Durante cada iteración se cierran varias historias, lo que hace que toda iteración añada un valor tangible para el cliente.

Es fundamental en toda esta planificación la presencia de un representante del cliente, que forma parte del equipo y que decide cuáles son las historias más valiosas. Estas historias son las que se desarrollarán en la iteración actual.

La obtención de feedback que permita llevar a cabo estimaciones precisas es fundamental. Se hacen estimaciones para cada historia, de modo que en cuanto se comienzan a tener datos históricos, éstos se utilizan para hacer que las siguientes estimaciones sean más precisas.

Como se puede ver, y como siempre ocurre con la Programación Extrema, el enfoque utilizado para llevar a cabo la planificación es eminentemente pragmático. Gran parte de la eficacia de este modelo de planificación deriva de una división clara de responsabilidades, que tiene en cuenta las necesidades del

negocio en todo momento. Dentro de esta división, el representante del cliente tiene las siguientes responsabilidades:

- Decidir qué se implementa en cada release o iteración.
- Fijar las fechas de fin de la release, recortando unas características o añadiendo otras.
- Priorizar el orden de implementación, en función del valor de negocio.

Las responsabilidades del equipo de desarrollo son las siguientes:

- Estimar cuánto tiempo llevará una historia: este feedback es fundamental para el cliente, y puede llevarle a reconsiderar qué historias se deben incluir en una iteración.
- Proporcionar información sobre el coste de utilizar distintas opciones tecnológicas.
- Organizar el equipo.
- Estimar el riesgo de cada historia.
- Decidir el orden de desarrollo de historias dentro de la iteración.

2.1.5.3.2 Testing

La ejecución automatizada de tests es un elemento clave de la XP. Existen tanto tests internos (o tests de unidad), para garantizar que el mismo es correcto, como tests de aceptación, para garantizar que el código hace lo que debe hacer. El cliente es el responsable de definir los tests de aceptación, no necesariamente de implementarlos. Él es la persona mejor cualificada para decidir cuál es la funcionalidad más valiosa.

El hecho de que los tests sean automatizados es el único modo de garantizar que todo funciona: desde el punto de vista de la XP, si no hay tests, las cosas sólo funcionan en apariencia. Aún más, si un test no está automatizado, no se le puede considerar como tal.

El objetivo de los tests no es corregir errores, sino prevenirlos. Por ejemplo, los tests siempre se escriben antes que el código a testear, no después: esto aporta un gran valor adicional, pues fuerza a los desarrolladores a pensar cómo se va a usar el código que escriben, poniéndolos en la posición de consumidores del software. Elaborar los tests exige pensar por adelantado cuáles son los problemas más graves que se pueden presentar, y cuáles son los puntos dudosos. Esto evita muchos problemas y dudas, en lugar de dejar que aparezcan "sobre la marcha".

Un efecto lateral importante de los tests es que dan una gran seguridad a los desarrolladores: es posible llegar a hacer cambios más o menos importantes sin miedo a problemas inesperados, dado que proporcionan una red de seguridad. La existencia de tests hace el código muy maleable.

2.1.5.3.3 Programación en parejas

La XP incluye, como una de sus prácticas estándar, la programación en parejas. Nadie programa en solitario, siempre hay dos personas delante del ordenador. Ésta es una de las características que más se cuestiona al comienzo de la adopción de la XP dentro de un equipo, pero en la práctica se acepta rápidamente y de forma entusiasta.

El principal argumento contra la programación en parejas es que es improductiva. Esta idea se basa en el hecho de que dos programadores "programan el doble por separado". Esto sería así si no fuera por las siguientes razones:

- El hecho de que todas las decisiones las tomen al menos dos personas proporciona un mecanismo de seguridad enormemente valioso.
- Con dos personas responsabilizándose del código en cada momento, es menos probable que se caiga en la tentación de dejar de escribir tests, etc., algo fundamental para mantener el código en buena forma. Es muy difícil que dos personas se salten tareas por descuido o negligencia.
- El hecho de programar en parejas permite la dispersión de know-how por todo el equipo. Este efecto es difícil de conseguir de otro modo, y hace que la incorporación de nuevos miembros al equipo sea mucho más rápida y

eficaz.

- El código siempre está siendo revisado por otra persona. La revisión de código es el método más eficaz de conseguir código de calidad, algo corroborado por numerosos estudios, muchos de los cuáles son anteriores a la Programación Extrema.

- En contra de lo que pueda parecer, los dos desarrolladores no hacen lo mismo: mientras el que tiene el teclado adopta un rol más táctico, el otro adopta un rol más estratégico, preguntándose constantemente si lo que se está haciendo tiene sentido desde un punto de vista global.

- Los datos indican que la programación en parejas es realmente más eficiente. Si bien se sacrifica un poco de velocidad al comienzo, luego se obtiene una velocidad de crucero muy superior. Esto contrasta con lo que ocurre en la mayor parte de los proyectos, en los que se arranca con una velocidad enorme pero rápidamente se llega a un estado muy parecido a la parálisis, en el que progresos cada vez más pequeños consumen cantidades de tiempo cada vez más grandes. Todos conocemos proyectos que se pasan el 50% del tiempo en el estado de "finalizado al 90%".

2.1.5.3.4 Refactorización

A la hora de la verdad, el código de la mayor parte de las aplicaciones empieza en un razonable buen estado, para luego deteriorarse de forma progresiva. El coste desorbitado del mantenimiento, modificación y ampliación de aplicaciones ya existente se debe en gran parte a este hecho.

Uno de los objetivos de la XP es mantener la curva de costes tan plana como sea posible, por lo que existen una serie de mecanismos destinados a mantener el código en buen estado, modificándolo activamente para que conserve claridad y sencillez. A este proceso básico para mantener el código en buena forma se le llama refactorización.

La refactorización no sólo sirve para mantener el código legible y sencillo: también se utiliza cuando resulta conveniente modificar código existente para hacer más

fácil implementar nueva funcionalidad.

2.1.5.3.5 Diseño simple

Otra práctica fundamental de la Programación Extrema es utilizar diseños tan simples como sea posible. El principio es "utilizar el diseño más sencillo que consiga que todo funcione". Se evita diseñar características extra porque a la hora de la verdad la experiencia indica que raramente se puede anticipar qué necesidades se convertirán en reales y cuáles no. La XP nos pide que no vivamos bajo la ilusión de que un diseño puede resolver todas o gran parte de las situaciones futuras: lo que parece necesario cambia con frecuencia, es difícil acertar a priori.

Es obvio que, si no vamos a anticipar futuras necesidades, debemos poder modificar el diseño si alguna de estas se materializa. La XP soporta estas modificaciones gracias a los tests automatizados. Estos permiten hacer cambios importantes gracias a la red de protección que proporcionan. La refactorización, que hace que el código existente sea claro y sencillo, también ayuda a hacer factibles las modificaciones.

La XP define un "diseño tan simple como sea posible" como aquél que:

- Pasa todos los tests.
- No contiene código duplicado.
- Deja clara la intención de los programadores (enfatiza el qué, no el cómo) en cada línea de código.
- Contiene el menor número posible de clases y métodos.

2.1.5.3.6 Propiedad colectiva del código

La XP aboga por la propiedad colectiva del código. En otras palabras, todo el mundo tiene autoridad para hacer cambios a cualquier código, y es responsable de ellos. Esto permite no tener que estar esperando a otros cuando todo lo que hace falta es algún pequeño cambio.

Por supuesto, cada cuál es responsable de las modificaciones que haga. El principio básico es "tú lo rompes, tú lo arreglas, no importa si está en el código propio o en el de otros".

Por último, vale la pena tener en cuenta que la existencia de tests automatizados impide que se produzca un desarrollo anárquico, al ser cada persona responsable de que todos los tests se ejecuten con éxito al incorporar los cambios que ha introducido al programa.

2.1.5.3.7 Integración continua

En muchos casos la integración de código produce efectos laterales imprevistos, y en ocasiones la integración puede llegar a ser realmente traumática, cuando dejan de funcionar cosas por motivos desconocidos. La Programación Extrema hace que la integración sea permanente, con lo que todos los problemas se manifiestan de forma inmediata, en lugar de durante una fase de integración más o menos remota.

La existencia de una fase de integración separada tiene dos efectos laterales indeseables: se empieza a hacer codificación "yo-yo", en la que todo el mundo modifica código "sólo para que funcione, ya lo ajustaremos", y hace que se acumulen defectos. Evitar que se acumulen defectos es muy importante para la XP, como lo es el conseguir que los defectos que cada programador inyecta los elimine él mismo.

2.1.5.3.8 Cliente en el equipo

Algunos de los problemas más graves en el desarrollo son los que se originan cuando el equipo de desarrollo toma decisiones de negocio críticas. Esto no debería ocurrir, pero a la hora de la verdad con frecuencia no se obtiene feedback del cliente con la fluidez necesaria: el resultado es que se ha de optar por detener el avance de los proyectos, o por que desarrollo tome una decisión de negocio. Por otra parte, los representantes del negocio también suelen encontrarse con problemas inesperados debido a que tampoco reciben el feedback adecuado por parte de los desarrolladores.

La XP intenta resolver este tipo de problemas integrando un representante del

negocio dentro del equipo de desarrollo. Ésta persona siempre está disponible para resolver dudas y para decidir qué y qué no se hace en cada momento, en función de los intereses del negocio. Debido a su inmersión dentro del equipo, y a que es él quien decide qué y qué no se hace, junto con los tests que verifican si la funcionalidad es la correcta y deseada, esta persona obtiene un feedback absolutamente realista del estado del proyecto.

2.1.5.3.9 Releases pequeñas

Siguiendo la política de la XP de dar el máximo valor posible en cada momento, se intenta liberar nuevas versiones de las aplicaciones con frecuencia. Éstas deben ser tan pequeñas como sea posible, aunque deben añadir suficiente valor como para que resulten valiosas para el cliente.

2.1.5.3.10 Semanas de 40 horas

La Programación Extrema lleva a un modo de trabajo en que el equipo está siempre al 100%. Una semana de 40 horas en las que se dedica la mayor parte del tiempo a tareas que suponen un avance puede dar mucho de sí, y hace innecesario recurrir a sobreesfuerzos -excepto en casos extremos.

Además, el sobreesfuerzo continuado pronto lleva a un rendimiento menor y a un deterioro de la moral de todo el equipo.

2.1.5.3.11 Estándares de codificación

Para conseguir que el código se encuentre en buen estado y que cualquier persona del equipo pueda modificar cualquier parte del código es imprescindible que el estilo de codificación sea consistente. Un estándar de codificación es necesario para soportar otras prácticas de la XP.

Sin embargo, la XP también es pragmática en esto, y apuesta por establecer un número mínimo de reglas: el resto se irán pactando de-facto. Esto evita un ejercicio inicial más o menos estéril.

2.1.5.3.12 Uso de Metáforas

La comunicación fluida es uno de los valores más importantes de la Programación Extrema: la programación en parejas, el hecho de incorporar al equipo una

persona que represente los intereses del negocio y otras prácticas son valiosas entre otras cosas porque potencian enormemente la comunicación.

Para conseguir que la comunicación sea fluida es imprescindible, entre otras cosas, utilizar el vocabulario del negocio. También es fundamental huir de definiciones abstractas. Dicho de otro modo, XP no pretende seguir la letra de la ley, sino su espíritu. Dentro de este enfoque es fundamental buscar continuamente metáforas que comuniquen intenciones y resulten descriptivas, enfatizando el qué por delante del cómo.

La metodología XP es una metodología ágil:

- Los individuos e interacciones son más importantes que los procesos y herramientas.

  Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. La gente es el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades.

- Software que funcione es más importante que documentación exhaustiva.

  Desarrollar software que funciona más que conseguir una buena documentación. La regla a seguir es “no producir documentos a menos que sean necesarios de forma inmediata para tomar un decisión importante”. Estos documentos deben ser cortos y centrarse en lo fundamental.

- La colaboración con el cliente es más importante que la negociación de contratos.

  La colaboración con el cliente más que la negociación de un contrato. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.

- La respuesta ante el cambio es más importante que el seguimiento de un plan.

  Responder a los cambios más que seguir estrictamente un plan. La habilidad de responder a los cambios que puedan surgir a los largo del proyecto (cambios en los requisitos, tecnología, equipo, etc.) determina el éxito o fracaso del mismo. Por ello, la planificación no debe ser estricta sino flexible y abierta.

2.1.5.4 Fases de la metodología XP

Hay diversas prácticas inherentes a la Programación Extrema, en cada uno de los ciclos de desarrollo del proyecto.

2.1.5.4.1 Planificación

XP plantea la planificación como un permanente diálogo entre la parte empresarial y técnica del proyecto, en la que los primeros decidirán el alcance (qué es lo realmente necesario del proyecto), la prioridad (qué debe ser hecho en primer lugar), la composición de las versiones (qué debería incluir cada una de ellas) y la fecha de las mismas. En cuanto a los técnicos, son los responsables de estimar la duración requerida para implementar las funcionalidades deseadas por el cliente, de informar sobre las consecuencias de determinadas decisiones, de organizar la cultura de trabajo y, finalmente, de realizar la planificación detallada dentro de cada versión. XP no es sólo un método centrado en el código, sino que sobre todo es un método de gestión de proyectos software.

Historias de usuarios

Las historias de usuario tienen el mismo propósito que los casos de uso, pero no son lo mismo. Son escritos por los propios clientes, tal y como ven ellos las necesidades del sistema. Por tanto serán descripciones cortas y escritas en el lenguaje del usuario, sin terminología técnica.

Las historias de usuario son similares al empleo de escenarios, con la excepción de que no se limitan a la descripción de la interfaz de usuario. También conducirán el proceso de creación de los test de aceptación. Uno o más de uno

de estos test se utilizarán para verificar que las historias de usuario han sido implementadas correctamente.

Una de las mayores equivocaciones a cerca del uso de las historias de usuario es la diferencia que existe entre estas y la tradicional especificación de requisitos.

La principal diferencia es el nivel de detalle. Las historias de usuario solamente proporcionaran los detalles sobre la estimación del riesgo y cuánto tiempo conllevará la implementación de dicha historia de usuario. El nivel de detalle de las historias de usuario debe ser el mínimo posible que permita hacerse una ligera idea de cuánto costará implementar el sistema. Cuando se llegue a la fase de implementación, los desarrolladores podrán acudir al cliente para ampliar detalles.

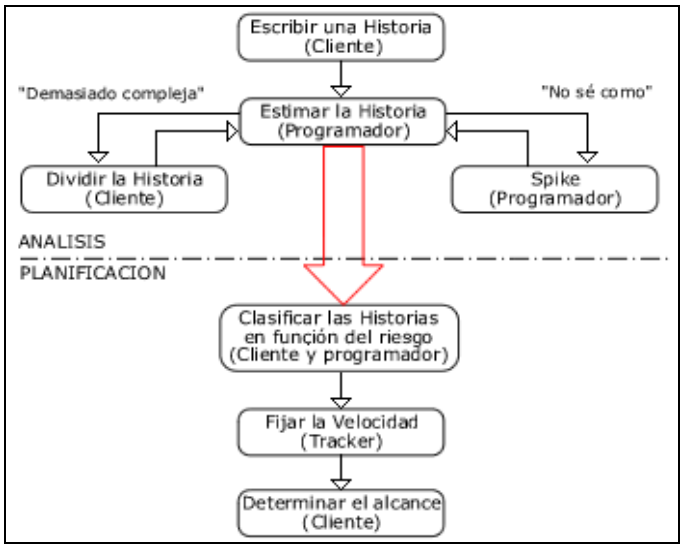
Los desarrolladores deberán hacer una estimación de cuánto tiempo, les llevará implementar cada historia de usuario. Las condiciones ideales son aquellas en las que se codifica la historia de usuario sin otras distracciones y sabiendo exactamente qué es lo que hay que implementar. Como resultado deberíamos obtener un periodo ideal de 1, 2 ó 3 semanas. Más de 3 semanas implica que debemos dividir la historia de usuario en partes. Menos de 1 semana implica que la historia de usuario es demasiado sencilla y tendremos que unir dos o más de ellas.

Se crea un plan de entregas

Las historias de usuario servirán para crear el plan estimado de entrega. Se convocará una reunión para crear el plan de entregas. El plan de entregas se usará para crear los planes de iteración para cada iteración. Es en este momento cuando los técnicos tomarán las decisiones técnicas y los comerciales las decisiones comerciales. En esta reunión estarán presentes tanto desarrolladores como los usuarios. Con cada historia de usuario previamente evaluada en tiempo de desarrollo ideal, el cliente las agrupará en orden de importancia. Una semana ideal es cuánto tiempo costaría implementar dicha historia si no tenemos nada más que hacer, incluyendo la parte de test correspondiente.

De esta forma se puede trazar el plan de entregas en función de estos dos parámetros: tiempo de desarrollo ideal y grado de importancia para el cliente.

Figura 2-5: Esquema Plan de entregas



Fuente: www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-XP.pdf
Elaborado por: LLUGSHA, William

Se controla la velocidad del proyecto

La velocidad del proyecto es una medida de cuan rápido se está desarrollando. Se usa para determinar cuántas historias de usuario pueden ser implementadas antes de una fecha dada (tiempo), o cuánto tiempo es necesario para llevar a cabo un conjunto de historias (alcance). Cuando se realiza una planificación por alcance se divide el numero total de semanas entre la velocidad de proyecto para determinar cuántas iteraciones estarán disponibles.

Se divide el proyecto en iteraciones

Cada iteración corresponde a un periodo de tiempo de desarrollo del proyecto de entre una y tres semanas. De esta forma, un proyecto, se divide en una docena de iteraciones, más o menos. Al principio de cada iteración se debería convocar una reunión para trazar el plan de iteración correspondiente.

Está prohibido intentar adelantarse e implementar cualquier cosa que no esté planeada para la iteración en curso. Habrá suficiente tiempo para añadir la funcionalidad extra cuando sea realmente importante según el plan de entregas.

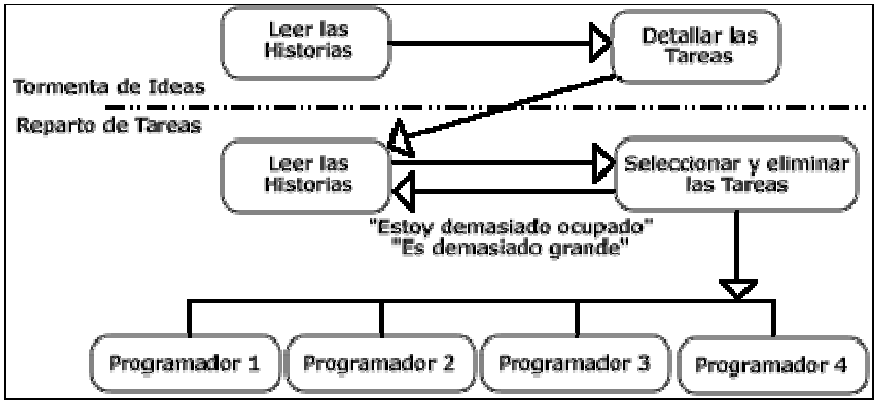
Se usará la velocidad del proyecto para determinar si una iteración está sobrecargada. La suma de los días que costará desarrollar todas las tareas de la

iteración no debería sobrepasar la velocidad del proyecto de la iteración anterior. Si la iteración está sobrecargada, el cliente deberá decidir que historias de usuario retrasar a una iteración posterior.

Al comienzo de cada iteración se traza el plan de iteración

El plan de iteración consiste en seleccionar las historias de usuario que, según el plan de entregas, corresponderían a esta iteración. También se eligen qué pruebas de aceptación fallidas se corregirán. Un plan de iteración puede verse como:

Figura 2-6: Esquema Plan de Iteración

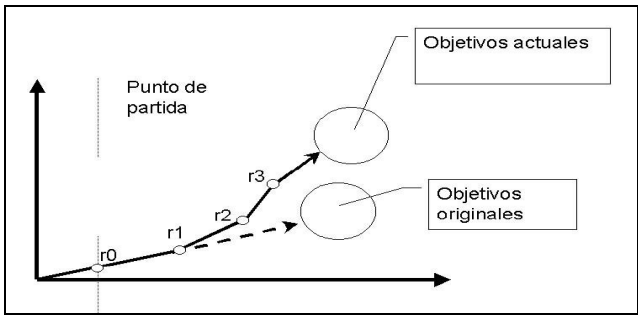


Fuente: www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-XP.pdf
Elaborado por: LLUGSHA, William

Cada historia de usuario se transformará en tareas de desarrollo. Cada tarea de desarrollo corresponderá a un periodo ideal de uno a tres días de desarrollo.

Es necesario mantener vigiladas la velocidad del proyecto y el movimiento de historias de usuario. Puede ser necesario volver a calcular las historias de usuario y negociar el plan de entrega cada de tres a cinco iteraciones. Como estaremos siempre implementando las historias de usuario más importantes para el cliente, estaremos haciendo lo máximo posible por nuestro cliente y la dirección. En cada iteración iremos alcanzado unas metas:

Figura 2-7: Metas de cada iteración



Fuente: www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-XP.pdf
Elaborado por: LLUGSHA, William

Se rota al personal

La rotaciones evitarán que las personas se conviertan en si mismas en un cuello de botella. Si sólo una persona de nuestro equipo es capaz de trabajar en un área concreta, existirá un riesgo enorme si esa persona nos deja por cualquier circunstancia. De esta forma, las rotaciones permitirán que todo el mundo conozca cómo funciona el sistema en general y ayudarán a realizar un reparto más equitativo del trabajo. Además el hecho de que se asignen por parejas nos permitirá entrenar a un nuevo miembro, simplemente dividiendo el grupo original en dos.

Cada día se convoca una reunión de seguimiento

La comunicación entre las diferentes partes que interviene en un proyecto resulta fundamental para su desarrollo. Esto se consigue gracias a las reuniones. Podremos analizar los problemas y las soluciones. Se recomienda que estas sean frecuentes, de poca duración y a ser posible delante de la pantalla del ordenador. Según la metodología XP, se recomienda que sean diarias; recordemos que los usuarios se considerarán parte integrante del equipo de desarrollo del proyecto. La reunión de seguimiento de cada mañana debe usarse para sacar a la luz los problemas, las soluciones y centrar el objetivo del equipo.

Corregir la propia metodología XP cuando falla

Deberemos corregir el proceso cuando éste falle. Cuando comencemos con un proyecto, seguiremos la metodología XP, pero debemos cambiar aquello que no funcione. Además los cambios que se realicen deberán ser comunicados al resto

del equipo, todo el mundo debe estar al corriente de los cambios. Esto no significa que cambiemos lo que no nos guste, sino que cambiemos aquello que no funciona con nuestro problema en particular.

2.1.5.4.2 Diseño

XP establece unas recomendaciones o premisas a la hora de abordad esta etapa.

Simplicidad

La simplicidad es la llave. Siempre costará menos tiempo de implementar un diseño sencillo que uno complejo. Por lo que, trataremos siempre de realizar las cosas de la manera más sencilla posible. Si alguna parte de la implementación resulta especialmente compleja, deberías replantearla (divide y vencerás). Así, cualquier cambio y modificación será mucho más sencillo. En ocasiones, realizar un diseño sencillo puede resultar una tarea especialmente difícil.

Elegir una metáfora para el sistema

Una metáfora para el sistema es una historia que todo el mundo puede contar a cerca de cómo el sistema funciona (Kent Beck).

La tarea de elegir una metáfora para el sistema nos permitirá mantener la coherencia de nombres de todo aquello que se va a implementar. El nombre de los objetos o partes de nuestro sistema es muy importante. La tarea de “poner nombre”, sencilla a simple vista, no lo es tanto. Debemos elegir un sistema de nombres que permita que cualquiera que lo vea adivine la relación entre el objeto y aquello que representa.

Por ejemplo, cuando se representa la segmentación en un procesador con una cadena de montaje de vehículos. Debemos encontrar una imagen de aquello que pretendemos representar en el mundo real. Así podremos tener una imagen mental más clara.

Usar tarjetas CRC (Cargo o clase, Responsabilidad y Colaboración) en las reuniones de diseño

Para poder diseñar el sistema como un equipo deberemos cumplir con tres

principios: Cargo o Clase, Responsabilidad y Colaboración (CRC). Las tarjetas CRC permitirán desprendernos del método de trabajo basado en procedimientos y trabajar con una metodología basada en objetos. Las tarjetas CRC permiten que el equipo completo contribuya en la tarea del diseño. Una tarjeta CRC representa un objeto. El nombre de la clase se coloca a modo de título en la tarjeta, las responsabilidades se colocan a la izquierda, y las clases que se implican en cada responsabilidad a la derecha, en la misma línea que su requerimiento correspondiente.

Crear soluciones puntuales para reducir riesgos

Un programa *Spike*, es un programa muy simple que explora una posible solución al problema. A partir de estos pequeños programas iremos construyendo la solución a nuestro problema.

El sistema se pone por primera vez en producción en, a lo sumo, unos pocos meses, antes de estar completamente terminado. Las sucesivas versiones serán más frecuentes (entre un día y un mes).

No se añadirá funcionalidad en las primeras etapas

Debemos evitar caer en la tentación de ir añadiendo funcionalidades según se nos vayan ocurriendo, aun incluso que sepamos exactamente cómo implementarlas. Es decir, debemos centrarnos en la tarea que se ha fijado para hoy, y hacerla lo mejor posible. Programaremos lo que se ha fijado, y no perderemos el tiempo en desarrollar código que no sabemos si será utilizado.

Reaprovechar cuando sea posible

Cuando eliminamos redundancia, eliminamos funcionalidad inútil, y rejuvenecemos antiguos diseños, estamos reciclando código. El reciclaje, dentro del ciclo de vida de un proyecto, ahorra tiempo e incrementa la calidad. El reciclaje implicará mantener el código limpio y fácil de comprender, modificar y ampliar. Esto puede resultar un poco costoso al principio, pero resulta fundamental a la hora de realizar diseños futuros.

2.1.5.4.3 Desarrollo

Esta etapa debe reunir las siguientes características o cualidades:

El cliente está siempre disponible

Una de las pocas condiciones que impone la metodología XP es tener al usuario siempre disponible. No sólo para ayudar al equipo de desarrollo, sino formando parte de él. Todas las fases que se realizan en un proyecto XP requieren de comunicación con el usuario, preferiblemente cara a cara, en persona, sin intermediarios.

Durante la reunión del plan de entregas, el usuario propondrá qué historia de usuario se incluye en cada plan. También se negociarán los plazos de entrega. El usuario o cliente tomará las decisiones que le afecten para alcanzar los objetivos de su negocio. También es necesario que el cliente colabore en la realización de los test. Estos test comprobarán que el sistema está listo para pasar a la fase de producción. El usuario comprobará los resultados obtenidos y tomará decisiones en cuanto a la utilización o no del sistema realizado.

Se debe escribir código de acuerdo a los estándares

El código a de ser desarrollado siguiendo los estándares de desarrollo para facilitar su lectura y modificación por cualquier miembro del equipo de desarrollo.

Es decisiva, para poder plantear con éxito la propiedad colectiva del código. Ésta sería impensable sin una codificación basada en estándares que haga que todo el mundo se sienta cómodo con el código escrito por cualquier otro miembro del equipo.

Desarrollar la unidad de pruebas primero

Cuando los test son creados antes que el código, la implementación del código será mucho más rápida. El tiempo empleado en desarrollar un test y algo de código para probarlo es aproximadamente el mismo tiempo que se emplea en crear exclusivamente dicho código. La creación de las unidades de test ayuda al programador a tener una visión a cerca del cómo, en definitiva, del

comportamiento del programa. Además, aún estamos a tiempo de dar marcha a tras, ya que el programador no a concluido la implementación. Esta manera de trabajar resulta especialmente beneficiosa en el diseño de complicados sistemas software.

«Cualquier característica de un programa para la que no haya un test automatizado, simplemente no existe».

Y es que éste es sin duda el pilar básico sobre el que se sustenta XP. Otros principios son susceptibles de ser adaptados a las características del proyecto, de la organización, del equipo de desarrollo. Aquí no hay discusión posible: si no hacemos tests, no estaremos haciendo XP.

Para ello, deberemos emplear algún framework de testing automático, como JUnit [Junit-www] o cualquiera de sus versiones para diferentes lenguajes. No sólo eso, sino que escribiremos los tests antes incluso que la propia clase a probar.

Todo el código debe programarse por parejas

Todo el código que formará parte del plan, será desarrollado por dos personas que trabajarán de forma conjunta en un ordenador. De esta manera, se incrementará la calidad del software desarrollado sin afectar al tiempo de entrega. Partimos de la base de que este equipo de dos personas posee unos conocimientos similares en cuanto a la tarea que van a realizar, es decir, están aproximadamente al mismo nivel. Mientras uno de ellos se encarga de pensar la táctica con la que se va a abordar el problema, el otro se encargará de pensar las estrategias que permiten llevar dichas tácticas a su máximo exponente. Ambos roles son intercambiables.

Sólo una pareja se encargará de integrar el código

En esta etapa pueden aparecer problemas debidos a la integración de los módulos que se han desarrollado y no han sido testeados todavía. Las unidades de test se encargarán de verificar la corrección de dichos módulos. Estos test deberán ser completos, en el sentido de que, un fallo el los mismo podría derivar que determinados errores pasaran inadvertidos.

Integrar frecuentemente

Los programadores deberán actualizar sus módulos con las versiones más recientes del trabajo realizado tan pronto como les sea posible. De esta manera, todo el mundo trabajará siempre con la última versión. Dicha actualización es responsabilidad de cada pareja de programadores. Esta integración se llevará a cabo cuando el éxito en las pruebas para su test correspondiente sea del 100 %, o cuando se trate de una parte que constituye un todo funcional, en cuanto esté acabada. Esta frecuencia con la que se inserta el nuevo código nos permitirá una rápida detección de los problemas de compatibilidad.

Todo el código es común a todos

Esta filosofía nos permite que cualquiera contribuya al desarrollo de cualquier parte del proyecto. Cualquier programador podrá cambiar una línea de código para añadir funcionalidad o eliminar algún fallo. Las personas dejan de ser un cuello de botella, en cuanto a la programación. Cualquiera puede realizar un cambió en el mismo siempre y cuando beneficie a la arquitectura del mismo. La responsabilidad del funcionamiento recaerá sobre el equipo al completo.

Dejar las optimizaciones para el final

No optimizaremos el código hasta el final. Nunca trataremos de averiguar cuales serán los posibles cuellos de botella del programa. Haz el trabajo, hazlo bien, y entonces hazlo rápido.

No trabajar más de 40 horas semanales

Trabajar horas extras absorbe el espíritu y la motivación del equipo. Aquellos proyectos que requieren horas extras para acabarse a tiempo pueden convertirse en un problema. En lugar de esto, utilizaremos las conocidas reuniones de plan de entregas para cambiar los objetivos del proyecto. También es una mala idea incorporar nueva gente al proyecto, una vez que este ya ha comenzado. Se trabajará un máximo de 40 horas semanales. Nadie es capaz de trabajar 60 horas a la semana y hacerlo con calidad.

2.1.5.4.4 Pruebas

Todo el código debe ir acompañado de su unidad de pruebas

Las unidades de test o pruebas constituyen unos de los pilares básicos de XP.

Uno de los errores que se suele cometer es pensar que podemos dejar la construcción de los test para los últimos meses en la realización de un proyecto. Descubrir todos los errores que pueden aparecer lleva tiempo, y más si dejamos la depuración de todos para el final.

Las unidades de test están directamente relacionadas con el concepto de posesión del código. En cierta manera, una parte del código no será reemplazado si no supera los test que existen para ese código. Después de cada modificación, podremos emplear los test para verificar que un cambio en la estructura no introduce un cambio en la funcionalidad. Sin embargo, si se añade nuevas capacidades a nuestro código, tendremos que rediseñar la unidad de test, para adaptarse a la nueva funcionalidad.

De esta manera, la probabilidad de que exista un fallo en ambos (test y código) es menor. Si creásemos los test después de la creación del código tenderíamos a hacerlos utilizando nuestro código como un generador, trasladado los fallos de uno al otro.

Todo el código debe pasar las unidades de pruebas antes de ser implantado

Las unidades de test serán incluidas junto con el código que verifican dentro del repositorio. El código no se considerará completo si este no consta de su unidad de test correspondiente. El código será implantado cuando supere sus correspondientes unidades de test.

Ante un fallo, una unidad de pruebas

Cuando se detecte un fallo, crearemos una unidad de pruebas para protegernos del mismo. De esta manera, la localización del mismo será mucho más fácil por parte de los programadores. Este nuevo test será empleado para aislar el fallo y depurarlo.

Se deben ejecutar pruebas de aceptación a menudo y publicar los resultados

Las pruebas de aceptación son creadas a partir de las historias de usuario. Durante una iteración la historia de usuario seleccionada en la planificación de iteraciones se convertirá en una prueba de aceptación. El cliente o usuario especifica los aspectos a testear cuando una historia de usuario ha sido correctamente implementada. Una historia de usuario puede tener más de una prueba de aceptación, tantas como sean necesarias para garantizar su correcto funcionamiento. Una prueba de aceptación es como una caja negra. Cada una de ellas representa una salida esperada del sistema. Es responsabilidad del cliente verificar la corrección de las pruebas de aceptación y tomar decisiones a cerca de las mismas. Una historia de usuario no se considera completa hasta que no supera sus pruebas de aceptación.. Esto significa que debe desarrollarse un nuevo test de aceptación para cada iteración o se considerará que el equipo de desarrollo no realiza ningún progreso. Las pruebas de aceptación deberían automatizarse ya que se deben pasar frecuentemente. La puntuación de las pruebas de aceptación se hará pública a todo el equipo. La garantía de calidad (Quality Assurance- QA), es una parte esencial en el proceso de XP. La realización de este tipo de pruebas y la publicación de los resultados debe ser lo más rápido posible, para que los desarrolladores puedan realizar con la mayor rapidez posible los cambios que sean necesarios. A las pruebas de aceptación también se las conoce con el nombre de pruebas de funcionalidad, y constituyen la garantía de que los requerimientos fijados por los usuarios han sido reflejados en el sistema.

2.1.6 APLICACIÓN DE EXTREME PROGRAMMING EN EL PROYECTO

Según el análisis efectuado en la sección 2.1.4, se optará por la programación extrema XP, ya que ésta se ajustará mejor a las características del proyecto.

Más concretamente, se cumplen las recomendaciones para emplear XP en un proyecto [Wells03] [Ferrer03]:

- Interés sincero por todas las partes en que el proyecto tenga éxito.

- El equipo de trabajo es pequeño.
- No existe un contrato fijo previo especificando tiempo, recursos y alcance.
- El equipo dispone de una formación elevada (esa es la finalidad) y capacidad de aprender.
- El proyecto tiene un riesgo alto en cuanto a lo innovador de la tecnología.

También se ha tenido en cuenta el éxito que ha tenido *extreme programming* en proyectos open source como es el caso de Dokeos, cuyas características hacen que se adapte especialmente bien esta metodología.

Como es evidente no todas las prácticas son aplicables al presente trabajo, ya que se trata de un proyecto de titulación; por lo que a continuación se especifica para cada una de ellas su aplicación en este proyecto:

2.1.6.1 Planificación Incremental

Se ha decidido realizar cuatro *releases* con sus respectivas iteraciones. Cada una de ellas proporcionando un valor de negocio claro, a grosso modo serán:

- Carreras: gestión de carreras o programas académicos
- Menciones: gestión de menciones
- Períodos Académicos: gestión de períodos académicos
- Malla Curricular: gestión de mallas curriculares (gestión de pre-requisitos y co-requisitos)
- Matrícula: gestión de matrícula en línea
- Calificaciones: gestión de calificaciones

En el Capítulo 3, se detallará la Planificación de las entregas en cada una de estas iteraciones.

2.1.6.2 Testing

Una de las principales aportaciones de esta metodología es el concepto de desarrollo dirigido por tests (*Test Driven Development*). Los tests son realizados a priori con el fin de prevenir errores, no de solucionarlos. Esto confiere una gran

calidad al software resultante. Los test serán realizados durante el desarrollo del proyecto a través de las pruebas de unidad.

2.1.6.3 Programación en parejas

Dado que la realización de este proyecto es de carácter individual, esta práctica no ha sido aplicada.

2.1.6.4 Refactorización

En las sucesivas iteraciones ha sido necesario refactorizar partes del núcleo del sistema Dokeos y este proceso ha sido realmente sencillo, a lo que ha contribuido en gran parte la cantidad de tests realizados.

2.1.6.5 Diseño simple

El diseño se ha mantenido sencillo, desde luego pasando los tests, sin código duplicado y con un mínimo de código gracias al núcleo del sistema de Dokeos que proporciona un gran dinamismo y evita la necesidad de implementación de operaciones repetitivas.

2.1.6.6 Propiedad colectiva del código

A pesar de que en este proyecto no ha existido un grupo de desarrolladores como tal, y gracias a que Dokeos con su licencia open source, se ha llevado a la práctica esta propiedad colectiva, donde cualquier desarrollador puede examinar el código y realizar las modificaciones que consideren pertinentes, pudiendo revertirse estas contribuciones en el proyecto.

2.1.6.7 Integración continua

Las pruebas de integración se efectuarán siempre, antes de añadir cualquier nueva clase al proyecto, o después de modificar cualquiera existente (**integración continua**), para lo que nos serviremos de f*rameworks de testing*, como PhpUnit.

2.1.6.8 Cliente en el equipo

Aunque esto estrictamente no se realiza dada la naturaleza del proyecto, la experiencia del director de este proyecto en el ámbito de la Educación Superior hace que se tengan en todo momento presente los intereses y visión de negocio.

2.1.6.9 Releases pequeñas

Se ha seguido esta práctica, liberando nuevas versiones según la funcionalidad que se ha ido implementando.

2.1.6.10 Semanas de 40 horas

No es aplicable a este proyecto.

2.1.6.11 Estándares de codificación

Se ha seguido el estándar de codificación Java, definido por Sun en [JavaCodeConventions], y además por las convenciones de código propuestas en el Manual del Desarrollador de Dokeos.

2.1.6.12 Uso de metáforas

Se ha utilizado el vocabulario de negocio en inglés por su mayor alcance a nivel mundial. Para facilitar la comunicación con los posibles usuarios se han establecido mecanismos como listas de correo.

2.2 SELECCIÓN DE LA PLATAFORMA DE DESARROLLO

Debido a que el LMS Open Source seleccionado fue Dokeos en su versión más estable 1.6.4, se adoptará o heredará la plataforma de desarrollo es decir:

- Lenguaje de Programación: PHP
- Servidor Web: Apache
- Base de Datos: MySQL

2.2.1 DESCRIPCION DE PHP

PHP es un lenguaje de programación usado generalmente para la creación de contenido para sitios web. PHP es un acrónimo recurrente que significa "PHP Hypertext Pre-processor" (inicialmente PHP Tools, o, Personal Home Page Tools), y se trata de un lenguaje interpretado usado para la creación de aplicaciones para servidores, o creación de contenido dinámico para sitios web. Últimamente también para la creación de otro tipo de programas incluyendo

aplicaciones con interfaz gráfica usando la biblioteca GTK+[25].

PHP es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas Web dinámicas, similar al ASP de Microsoft o el JSP de Sun, embebido en páginas HTML y ejecutado en el servidor.

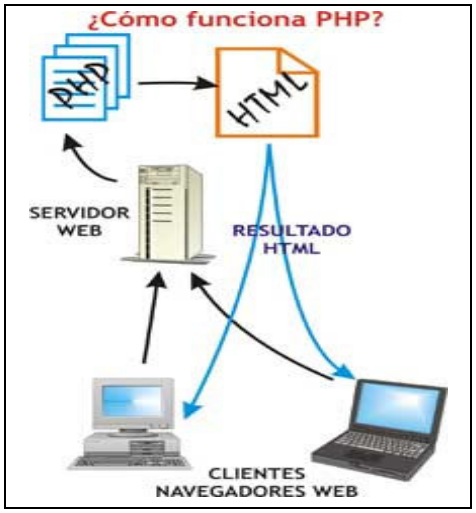
La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas. No es un lenguaje de marcas como podría ser HTML, XML o WML. Está más cercano a JavaScript o a C, para aquellos que conocen estos lenguajes.

2.2.1.1 Funcionamiento de PHP

A diferencia de Java o JavaScript que se ejecutan en el navegador PHP se ejecuta en el servidor por eso nos permite acceder a los recursos que tenga el servidor como por ejemplo podría ser una base de datos. El programa PHP es ejecutado en el servidor y el resultado es enviado al navegador. El resultado es normalmente una página HTML pero también podría ser una página WML (Wap).

Al ser PHP un lenguaje que se ejecuta en el servidor no es necesario que su navegador lo soporte, es independiente del navegador, sin embargo, para que sus páginas PHP funcionen el servidor donde están alojadas debe soportar PHP.

Figura 2-8: Funcionamiento de PHP



Fuente: LinuxCentro.net
Elaborado por: LLUGSHA, William

[25] WIKIPEDIA, Glosario, http://es.wikipedia.org/wiki/GTK%2B, 2006. [**GTK+** es un grupo importante de bibliotecas o rutinas para desarrollar interfaces gráficas de usuario (GUI) para principalmente los entornos gráficos GNOME, XFCE y ROX de sistemas Linux.]

2.2.1.2 Características

Al ser un lenguaje libre dispone de una gran cantidad de características que lo convierten en la herramienta ideal para la creación de páginas web dinámicas:

- Soporte para una gran cantidad de bases de datos: MysSQL, PostgreSQL, Oracle, MS SQL Server, Sybase mSQL, Informix, entre otras.
- Integración con varias bibliotecas externas, permite generar documentos en PDF (documentos de Acrobat Reader) hasta analizar código XML.
- Ofrece una solución simple y universal para las paginaciones dinámicas del Web de fácil programación.
- Perceptiblemente más fácil de mantener y poner al día que el código desarrollado en otros lenguajes.
- Soportado por una gran comunidad de desarrolladores, como producto de código abierto, PHP goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y reparen rápidamente.
- El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP.
- Con PHP se puede hacer cualquier cosa que podemos realizar con un script CGI, como el procesamiento de información en formularios, foros de discusión, manipulación de cookies y páginas dinámicas.

Un sitio con páginas webs dinámicas es aquel que permite interactuar con el visitante, de modo que cada usuario que visita la página vea la información modificada para propósitos particulares.

2.2.1.3 Seguridad

PHP es un potente lenguaje y el intérprete, tanto incluido en el servidor Web como módulo o ejecutado como un binario CGI, puede acceder a ficheros, ejecutar comandos y abrir comunicaciones de red en el servidor. Todas estas

características hacen que lo que se ejecute en el servidor Web sea seguro por defecto.

PHP ha sido diseñado específicamente para ser un lenguaje más seguro para escribir programas CGI, Perl o C y con la correcta selección de las opciones de configuración de tiempo de compilación y ejecución se consigue la exacta combinación de libertad y seguridad que se necesita. Ya que existen diferentes modos de utilizar PHP, existe también una multitud de opciones de configuración que permiten controlar su funcionamiento. Una gran selección de opciones garantiza que se pueda usar PHP para diferentes aplicaciones, pero también significa que existen combinaciones de estas opciones y configuraciones del servidor que producen instalaciones inseguras.

2.2.1.4 Ventajas del PHP

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL
- Leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su página oficial (www.php.net), entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- Permite crear los formularios para la web.
- Biblioteca nativa de funciones sumamente amplia e incluida

- No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

Por lo tanto, la plataforma que será utilizada a lo largo del proceso de desarrollo de nuestro proyecto es PHP.

CAPITULO 3. DESARROLLO DEL SISTEMA

Una vez que se han investigado los lineamientos de la Programación Extrema, en capítulos anteriores se desarrollará y/o implementará nuevas funcionalidades o módulos en un LMS Open Source Dokeos v1.6.4.

De acuerdo con la metodología, XP tiene las siguientes fases: planeación o análisis, diseño, implementación ó codificación, y pruebas, pero estas fases se aplican en cada iteración, es decir al resolver cada una de las historias de usuario, se hace el diseño, se codifica y se prueba hasta que el sistema soporte cada uno de los requerimientos del cliente. Es por este motivo que en este capítulo no se podrá distinguir cada una de las fases como un título por separado ya que la forma de trabajar de XP sugiere realizar los pasos antes mencionados.

En primera instancia se define el problema como tal, estableciendo de esta manera los requerimientos del cliente. Estos requerimientos fueron tomados de una reunión informal. A continuación se detallan cada una de las fases de desarrollo hasta obtener la plataforma funcional.

3.1 DEFINICIÓN DEL PROBLEMA

En la sección 1.3 y 1.4 se realizó un análisis sobre el LMS seleccionado. En la sección 1.3 el análisis se basó en la arquitectura y metodología de desarrollo de Dokeos v1.6.4, y por otro lado en la sección 1.4, se identificaron los módulos con las que cuenta actualmente la plataforma. Es preciso mencionar que la versión 1.6.4 de Dokeos es la versión más estable y que este LMS exclusivamente administra la toma de cursos.

Además tomando en cuenta que este proyecto tiene como objetivo desarrollar e incorporar nuevos módulos o funcionalidades en el LMS Dokeos v1.6.4, con la finalidad de que apoye el proceso de enseñanza aprendizaje en programas de pregrado y postgrado de educación superior. Estos nuevos módulos deben interactuar con la plataforma Dokeos con el objeto de que dicha plataforma se adapte a una institución de educación superior en las diferentes modalidades (presencial, semipresencial o a distancia).

En este sentido, es necesario que dentro la plataforma Dokeos v1.6.4, exista un módulo de gestión de carreras o programas académicos de pregrado y postgrado en las diferentes modalidades. Este módulo manejará el ingreso, modificación, eliminación o búsqueda de dichas carreras. Entre la información académica que se pueda requerir de una carrera o programa académico, están las menciones o especialidades que pueda tener una carrera de una Institución de Educación Superior; por esta razón es necesario la administración de estas menciones o especialidades.

Debido a que se administrará carreras o programas académicos, es necesario la gestión de períodos académicos y mallas curriculares, es decir la gestión de pre-requisitos y co-requisitos para un curso o asignatura perteneciente a la carrera o programa académico.

Por otro lado, en la identificación efectuada de módulos (componentes) que pertenecen a un curso en la plataforma Dokeos v1.6.4 (subcapítulo 1.4.2), se pueden destacar: Foros, Documentos, Chat, Anuncios, Grupos, etc., pero de acuerdo a los requerimientos del cliente, es necesario la incorporación de un módulo de publicación de calificaciones en diferentes categorías (pruebas, exámenes, exposiciones, etc.) por parte del profesor.

Con todas estas incorporaciones dentro del LMS Dokeos, y teniendo en cuenta los requerimientos del cliente, es necesario la implementación de un módulo de Servicios en Línea, que permita al usuario matricularse en una carrera o programa académico, y con ello tener la opción de ver el estado de sus matrículas, acceder a su currículum o expediente académico en dicha carrera.

Es preciso mencionar que con la implementación de estos nuevos módulos, este proyecto no pretende ser un Sistema Administrativo para una Institución de Educación Superior, sino ofrecer al usuario un “herramientas en línea”, que apoye el proceso de enseñanza-aprendizaje en programas de pregrado y postgrado de educación superior.

3.2 ANALISIS

Para proceder con el desarrollo de este proyecto, se tuvo reuniones periódicas con el cliente, en nuestro caso fue el Ing. César Esquetini, director del proyecto, quien a través de conversaciones se pudieron establecer los requerimientos o necesidades que necesitaba la plataforma Dokeos. Esta información se recopiló en un documento el cual se detalló en la Definición del Problema de la sección anterior 3.1.

3.2.1 HISTORIAS DE USUARIOS (USER STORIES)

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software, lo que equivaldría a los casos de uso en el proceso unificado.

Se describen brevemente las características que el sistema debe tener desde la perspectiva del cliente, a través de las historias de usuarios.

Como se explicó en la sección 2.1.5.4.1, las estimaciones de esfuerzo asociado a la implementación de las historias de usuarios se establecen utilizando como medida el punto. Un punto, equivale a una semana ideal de programación, y las historias generalmente valen de 1 a 3 puntos.

Para este proyecto, las historias de usuarios son definidas en base al documento de definición del problema expuesto en la subcapítulo 3.1.

3.2.1.1 Historia 1: Creación de carreras o programas académicos

Tabla 3-1: Historia 1 - Creación de carreras o programas académicos

Historia de Usuario	
Número: 1	**Usuario**: Administrador de la plataforma
Nombre de historia: Creación de carreras o programas académicos	
Prioridad en negocio: Alta (Alta / Media / Baja)	**Riesgo en desarrollo**: Baja (Alta / Media / Baja)
Puntos estimados: 1	**Iteración asignada**: 1
Programador responsable: William Llugsha	
Descripción: Para la creación de una nueva carrera o programa académico el Administrador de la Plataforma	

de Dokeos, debe ingresar el Título de la carrera, costo del crédito, costo de la taza administrativa, y seleccionar el Nivel de Formación (Nivel Técnico Superior, Tercer Nivel o Cuarto Nivel), la modalidad de estudios (presencial, semipresencial o distancia), la duración y el sistema de estudios.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

3.2.1.2 Historia 2: Visualización, modificación, eliminación y búsqueda de carreras o programas académicos

Tabla 3-2: Historia 2 - Visualización, modificación, eliminación y búsqueda de carreras o programas académicos

Historia de Usuario	
Número: 2	**Usuario**: Administrador de la plataforma
Nombre de historia: Visualización, modificación, eliminación y búsqueda de carreras o programas académicos	
Prioridad en negocio: Alta (Alta / Media / Baja)	**Riesgo en desarrollo**: Baja (Alta / Media / Baja)
Puntos estimados: 2	**Iteración asignada**: 1
Programador responsable: William Llugsha	
Descripción: Al usuario se le presenta un listado de carreras o programas académicos con información principal como la modalidad, nivel de formación, duración y el número de cursos o asignaturas que posee esta carrera. El usuario podrá visualizar, modificar, eliminar las carreras registradas, e inclusive buscar carreras o programas académicos a través del ingreso de ciertos criterios (título, modalidad, etc.)	

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

3.2.1.3 Historia 3: Gestión de menciones

Tabla 3-3: Historia 3 - Gestión de menciones

Historia de Usuario	
Número: 3	**Usuario**: Administrador de la plataforma
Nombre de historia: Gestión de menciones	
Prioridad en negocio: Alta (Alta / Media / Baja)	**Riesgo en desarrollo**: Baja (Alta / Media / Baja)
Puntos estimados: 1	**Iteración asignada**: 2
Programador responsable: William Llugsha	
Descripción: El sistema debe permitir el registro y visualización de la información de las menciones de una carrera o programa académico. Esta información puede ser susceptible de modificación o eliminación.	

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

3.2.1.4 Historia 4: Gestión de Períodos Académicos

Tabla 3-4: Historia 4 - Gestión de Períodos Académicos

Historia de Usuario	
Número: 4	**Usuario**: Administrador de la plataforma
Nombre de historia: Gestión de Períodos Académicos	
Prioridad en negocio: Alta (Alta / Media / Baja)	**Riesgo en desarrollo**: Baja (Alta / Media / Baja)
Puntos estimados: 1	**Iteración asignada**: 2
Programador responsable: William Llugsha	
Descripción: El sistema debe permitir el registro y visualización de la información de los períodos académicos de una carrera o programa académico. Esta información puede ser susceptible de modificación o eliminación. Cada período académico puede tener 3 estados: "*actual*", "*histórico*" o "*futuro*", con la particularidad de que solamente un período académico debe estar con estado "*actual o activo*". Esta información puede ser susceptible de modificación o eliminación.	

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

3.2.1.5 Historia 5: Gestión de pre-requisitos y co-requisitos (Malla Curricular)

Tabla 3-5: Historia 5 - Gestión de pre-requisitos y co-requisitos (Malla Curricular)

Historia de Usuario	
Número: 5	**Usuario**: Administrador de la plataforma
Nombre de historia: Gestión de Pre-requisitos y Co-requisitos (Malla Curricular)	
Prioridad en negocio: Alta (Alta / Media / Baja)	**Riesgo en desarrollo**: Baja (Alta / Media / Baja)
Puntos estimados: 1	**Iteración asignada**: 2
Programador responsable: William Llugsha	
Descripción: El sistema debe permitir el registro y visualización de los cursos que son pre-requisitos o co-requisitos de un curso que pertenecen a una carrera o programa académico. Los cursos que mantengan dependencia con otros cursos pueden ser susceptibles de dar de baja dicha relación de dependencia.	

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

3.2.1.6 Historia 6: Gestión de Calificaciones

Tabla 3-6: Historia 6 - Gestión de Calificaciones

Historia de Usuario	
Número: 6	**Usuario**: Profesor

Nombre de historia: Gestión de Calificaciones	
Prioridad en negocio: Alta (Alta / Media / Baja)	**Riesgo en desarrollo**: Baja (Alta / Media / Baja)
Puntos estimados: 2	**Iteración asignada**: 3
Programador responsable: William Llugsha	
Descripción: El sistema debe permitir el registro y consulta de las calificaciones parciales de los estudiantes que pertenecen a un curso dentro de un período académico determinado. Estas calificaciones parciales serán registradas seleccionando la categoría (Tareas, evaluación, pruebas, exámenes, etc), junto con el nombre de dicha calificación. También debe permitir el registro y consulta de las calificaciones finales (calificación de aprobación del curso del estudiante) y el tipo de aprobación (exonerado, aprobado, fallido, etc)	

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

3.2.1.7 Historia 7: Gestión de Matrícula

Tabla 3-7: Historia 7 - Gestión de Matrícula

Historia de Usuario	
Número: 7	**Usuario**: Administrador, Estudiantes
Nombre de historia: Gestión Matrícula	
Prioridad en negocio: Media (Alta / Media / Baja)	**Riesgo en desarrollo**: Baja (Alta / Media / Baja)
Puntos estimados: 2.4	**Iteración asignada**: 4
Programador responsable: William Llugsha	
Descripción: El usuario podrá matricularse en una carrera o programa académico dentro de un período académico activo. Para ello, el usuario debe seleccionar las materias o cursos que desee y los materiales (libros, folletos, guías, etc.) relacionados con el curso. Al final del proceso de matriculación, al usuario se le presenta el comprobante de matrícula. El usuario Administrador tendrá la opción de matricular a otros estudiantes (nuevos estudiantes o estudiantes de la carrera), modificar o eliminar las matrículas de los estudiantes.	

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

3.2.1.8 Historia 8: Visualización de Currículum Académico

Tabla 3-8: Historia 8 - Visualización de Currículum Académico

Historia de Usuario	
Número: 9	**Usuario**: Estudiantes
Nombre de historia: Visualización de Currículum Académico	
Prioridad en negocio: Media (Alta / Media / Baja)	**Riesgo en desarrollo**: Baja (Alta / Media / Baja)

Puntos estimados: 1	**Iteración asignada**: 3
Programador responsable: William Llugsha	
Descripción: El usuario debe tener la posibilidad de visualizar su currículum o expediente académico.	

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

3.2.2 PLANIFICACIÓN DE ENTREGAS (RELEASE PLANNING)

Las historias de usuario servirán para crear el plan estimado de entregas. Para ello se convocó a una reunión con el cliente para crear el plan de entregas. El plan de entregas se usará para crear los planes de iteración para cada iteración.

Con cada historia de usuario descrita y evaluada en tiempo de desarrollo ideal, el cliente las agrupará en orden de importancia.

De esta manera, se procederá a trazar el plan de entregas en función de dos parámetros: Tiempo de desarrollo ideal (Esfuerzo) y Grado de importancia para el cliente (prioridad), para ello es necesario realizar una sinopsis de las historias de usuarios en base a la prioridad o esfuerzo. Ver tabla 3-10.

Tabla 3-9: Sinopsis de las Historias de usuarios

Nombre Historia	Prioridad	Esfuerzo
HISTORIA 1: Creación de carreras o programas académicos	Alta	1
HISTORIA 2: Visualización, modificación, eliminación y búsqueda de carreras o programas académicos	Alta	2
HISTORIA 3: Gestión de menciones	Alta	1
HISTORIA 4: Gestión de Períodos Académicos	Alta	1
HISTORIA 5: Gestión de pre-requisitos y co-requisitos (Malla Curricular)	Alta	1
HISTORIA 6: Gestión de Calificaciones	Alta	2
HISTORIA 7: Gestión de Matrícula	Media	2.4
HISTORIA 8: Visualización de Currículum Académico	Media	1
Total semanas		**11.4**

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

En base a la tabla 3-10, se efectuó un resumen de las historias de usuario en relación a la prioridad y esfuerzo de cada una de ellas. En esta tabla se puede

apreciar que el esfuerzo estimado es de 11.4 semanas.

Tabla 3-10: Planificación de Entregas

Iteraciones	Historias	Prioridad	Esfuerzo	Fecha Inicio	Fecha Final
Iteración 1	HISTORIA 1	Alta	1	02/10/2006	20/10/2006
	HISTORIA 2	Alta	2		
Iteración 2	HISTORIA 3	Alta	1	23/10/2006	14/11/2006
	HISTORIA 4	Alta	1		
	HISTORIA 5	Alta	1		
Iteración 3	HISTORIA 6	Alta	2	15/11/2006	05/12/2006
	HISTORIA 8	Media	1		
Iteración 4	HISTORIA 7	Media	2.6	06/12/2006	22/12/2006

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

De acuerdo a esta planificación, se tiene 4 iteraciones, con una duración de aproximadamente 3 semanas cada una.

3.2.3 HERRAMIENTAS

En el desarrollo de este proyecto se han utilizado las siguientes herramientas en cada ámbito. Todo el software utilizado es gratuito y tan sólo la herramienta Dreamweaver y Zend Studio no es open source.

3.2.3.1 Desarrollo

La plataforma AVE, fue desarrollada bajo los siguientes aspectos técnicos:

- ApacheFriends XAMPP (basic package) version 1.5.3a
    - Apache 2.2.2
    - MySQL 5.0.21
    - PHP 5.1.4 + PHP 4.4.2-pl1 + PEAR
    - PHP-Switch win32 1.0 (use the "php-switch.bat")
    - XAMPP Control Version 2.3 from www.nat32.com
    - XAMPP Security 1.0
    - phpMyAdmin 2.8.1

- ADOdb 4.80

- Zend Studio(TM) 3.5, es una completa plataforma de desarrollo para PHP, Zend Studio es un ambiente cliente servidor para la programación, debug, análisis de desarrollo de aplicaciones PHP.
- Macromedia Dreamweaver version 8.0.2, con las siguientes extensiones:
  - PHAkt3, es una mejor alternativa al modelo de servidor PHP_MySQL, que hace uso de la librería ADOdb para la gestión con la base de datos
  - PHP/Java Class Generator, para la generación de classes PHP en el estilo javabeans.

3.2.3.2 Ejecución

El software requerido para la ejecución del sistema es:

1. Un servidor de aplicaciones web
2. Un sistema gestor de base de datos relacional
3. Un navegador web

Este es el software que ha sido utilizado para la ejecución y los tests de este sistema Dokeos v1.6.4 para cada uno de los roles anteriores teniendo en cuenta su interoperabilidad.

1. Servidor de aplicaciones web: Apache 2.2.2
2. Base de datos: MySQL 5.0.21
3. Navegadores web: Firefox 0.8, MS Internet Explorer (5.0+), Netscape (4.7+), Mozilla (1.2+), Safari, Opera.

3.3 DESARROLLO ITERATIVO

Una vez definidas las historias de usuarios en la sección 3.2.1 y la planificación de entregas en la sección 3.2.2 se procederá a detallar cada una de las fases a seguirse en cada una de las iteraciones.

3.3.1 ITERACION 1

3.3.1.1 Diseño

En la fase de diseño se detallan las tarjetas CRC (Class, Responsability and Collaboration) para las Historias 1 y 2, según lo definido en las Historias de Usuarios y la Planificación de Entregas.

En este sentido se detallan las Tarjetas CRC de las historias correspondientes a esta iteración.

a. Clase Carrera (CareerManager)

Tabla 3-11: Iteración 1 - Tarjeta CRC para la Clase CareerManager

Clase	**CareerManager**
Descripción	Clase localizada en la capa del negocio para gestionar la información de las Carreras o Programas Académicos.
Atributos	- career_id (integer) - type_career_id (integer) - modality_id (integer) - system_id (integer) - career_name (string) - career_cost_credit (float) - career_administrative_rate (float) - career_num_quotas (integer) - career_duration (integer) - career_creation_date (date) - courseList (array) - userList (array) - academic_periodList (array) - specialitiesList (array)
Métodos	CareerManager() read(parámetro) get() set(parámetro) save() delete() selectNbrCourses() selectNbrSpecialities() addToCourseList(parámetro) addToAcademicPeriodList(parámetro) addToSpecialitiesList(parámetro) editToSpecialitiesList(parámetro) removeFromCourseList(parámetro) removeFromAcademiPeriodList(parámetro) removeFromSpecialitiesList(parámetro) isInList(parámetros) getCareers() searchCereer(parámetros) getCourses()

	getUsers(parámetros) getAcademicPeriods() getSpecialities() getCurrentAcademicPeriod()
Responsabilidades	- Inicializar el objeto (Constructor de la clase). - Obtener la información de una carrera o programa académico. - Obtener el código de un objeto CareerManager (código de carrera). - Asignar un nuevo código de carrera al código de un objeto CareerManager (código de carrera). - Almacenar la información de una carrera o programa académico (inserción o actualización). - Eliminar la información de una carrera o programa académico. - Obtener el número de cursos pertenecientes a una carrera o programa académico. - Obtener el número de menciones pertenecientes a una carrera o programa académico. - Agregar a la lista de cursos un objeto curso pasado como parámetro. - Agregar a la lista de Períodos Académicos un objeto período académico pasado como parámetro. - Agregar a la lista de menciones un objeto mención pasado como parámetro. - Editar la información de un elemento de la lista de menciones. - Remover de la lista de cursos un objeto curso pasado como parámetro. - Remover de la lista de Períodos Académicos un objeto período académico pasado como parámetro. - Remover de la lista de menciones un objeto mención pasado como parámetro. - Consultar si un objeto determinado se encuentra en un listado determinado (Listado de cursos, períodos académicos o menciones). - Obtener un listado de objetos de carreras registradas. - Buscar carreras dado ciertos parámetros de búsqueda. - Obtener un listado de objetos de cursos que pertenecen a una carrera o programa académico. - Obtener un listado de objetos de usuarios (profesores, tutores o estudiantes) que pertenecen a una carrera o programa académico dado ciertos filtros (curso, período académico o tipo de usuario). - Obtener un listado de objetos de períodos académicos que pertenecen a una carrera o programa académico. - Obtener un listado de objetos de menciones relacionadas a una carrera. - Obtener el Período Académico Actual o Activo de una carrera o programa académico.
Colaboradores	ADOdb CourseManager UserManager AcademicPeriodManager MentionCareerManager

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

b. Clase Course (CourseManager)

Tabla 3-12: Iteración 1 - Tarjeta CRC para la Clase CourseManager

Clase	**CourseManager**
Descripción	Clase localizada en la capa del negocio para gestionar la información de los Cursos.
Atributos	- code (string) - directory (string) - db_name (string) - course_language (string) - title (string) - description (string) - category_code (string) - visibility (integer) - show_score (integer) - tutor_name (string) - visual_code (string) - department_name (string) - department_url (string) - disk_quota (integer) - last_visit (date) - last_edit (date) - creation_date (date) - expiration_date (date) - target_course_code (string) - subscribe (integer) - unsubscribe (integer) - career_id (integer) - type_course_id (integer) - credits (integer) - cycle (integer) - sp_career_id (integer) - tutor_id (integer) - resourceList (array) - userList (array) - cost_course (float)
Métodos	CourseManager() read(parámetro) get() set(parámetro) save() delete() selectNbrUsers() selectNbrResources() addToUserList(parámetro) addToResourceList(parámetro) removeFromUserList(parámetro) removeFromResourceList(parámetro) isInList(parámetros) getTotalResourcesOfCourse(parámetro) getUsersOfCourse(parámetros) getTrainingCourses()
Responsabilidades	- Inicializar el objeto (Constructor de la clase). - Obtener la información del curso. - Obtener el código de un objeto CourseManager (código del curso). - Asignar un nuevo código de curso al código de un objeto CourseManager (código del curso). - Almacenar la información de un curso (inserción o actualización). - Eliminar la información de un curso.

	- Obtener el número de usuarios de un curso. Este número será relativo, es decir dependerá del tipo de usuario y el período académico. - Obtener el número de recursos o materiales de un curso. - Agregar a la lista de usuarios un objeto usuario pasado como parámetro. - Agregar a la lista de recursos o materiales un objeto recurso pasado como parámetro. - Remover de la lista de usuarios un objeto usuario pasado como parámetro. - Remover de la lista de recursos o materiales un objeto recurso pasado como parámetro. - Consultar si un objeto determinado se encuentra en un listado determinado (Listado de usuarios, recursos o materiales). - Obtener el costo total de recursos de un curso. - Obtener los usuarios de un curso dado un cierto período académico y tipo de usuario. - Obtener los cursos de capacitación o entrenamiento. Estos cursos no pertenecerán a una carrera o programa académico.
Colaboradores	ADOdb CareerManager UserManager ResourceManager

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

c. Clase Usuario (UserManager)

Tabla 3-13: Iteración 1 - Tarjeta CRC para la Clase UserManager

Clase	**UserManager**	
Descripción	Clase localizada en la capa del negocio para gestionar la información de los usuarios.	
Atributos	- user_id (integer) - lastname (string) - firstname (string) - username (string) - password (string) - auth_source (string) - email (string) - passport_ci (string) - date_born (date) - civil_state (char) - gender (char) - nationality (char) - status (integer) - official_code (string) - address (string) - phone (string)	- phone_movil (string) - company (string) - address_company (string) - picture_uri (string) - creator_id (integer) - competences (string) - diplomas (string) - openarea (string) - teach (string) - productions (string) - chatcall_user_id (integer) - chatcall_date (date) - chatcall_text (string) - state_id (integer) - support_center_id (integer) - support_center_name (integer)

	- phone_office (string)	- status_inscrip (char) - courseList (array)
Métodos	UserManager() read(parámetro) get() set(parámetro) save() delete() getCareersOfUser() getCoursesOfUser(parámetros) getPotencialUsers(parámetro) getStatusInCareer(parámetro) getEnrollsOfUser(parámetros) isAdmin()	
Responsabilidades	- Inicializar el objeto (Constructor de la clase). - Obtener la información del curso. - Obtener el código de un objeto UserManager (código del usuario). - Asignar un nuevo código de usuario al código de un objeto UserManager (código del usuario). - Almacenar la información de un usuario (inserción o actualización). - Eliminar la información de un usuario. - Obtener las carreras en las que está inscrito el usuario. - Obtener los cursos de un usuario en una carrera y período académico determinado. - Obtener los usuarios que pueden ser matriculados o inscritos en una carrera (Usuarios nuevos que pueden formar parte de una carrera). - Obtener el estatus de un usuario dentro de una carrera (administrador, profesor, estudiante). - Obtener todas las matriculas de un estudiante en una carrera y en un período académico.	
Colaboradores	ADOdb CourseManager CareerManager EnrollManager	

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

d. Clase Tipo de Curso (TypeCourseManager)

Tabla 3-14: Iteración 1 - Tarjeta CRC para la Clase TypeCourseManager

Clase	**TypeCourseManager**
Descripción	Clase localizada en la capa del negocio para gestionar la información de los Tipos de Cursos (cursos que pertenecen a una carrera o simplemente son cursos de capacitación).
Atributos	- type_course_id (integer) - type_course_name (string)

Métodos	TypeCourseManager() getTypeCourses()
Responsabilidades	- Inicializar el objeto (Constructor de la clase). - Obtener los tipos de cursos disponibles.
Colaboradores	ADOdb

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

e. Clase Sistema de Estudios (SystemCareerManager)

Tabla 3-15: Iteración 1 - Tarjeta CRC para la Clase SystemCareerManager

Clase	**SystemCareerManager**
Descripción	Clase localizada en la capa del negocio para gestionar la información de los Sistemas de Estudios de las Carreras o Programas Académicos.
Atributos	- system_id (integer) - system_name (string)
Métodos	SystemCareerManager () getSystems()
Responsabilidades	- Inicializar el objeto (Constructor de la clase). - Obtener los Sistemas de Estudios disponibles (Semestres, años, ciclos, etc.).
Colaboradores	ADOdb

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

f. Clase Modalidades de Estudios (ModalityCareerManager)

Tabla 3-16: Iteración 1 - Tarjeta CRC para la Clase ModalityCareerManager

Clase	**ModalityCareerManager**
Descripción	Clase localizada en la capa del negocio para gestionar la información de los Modalidades de Estudio de las Carreras o Programas Académicos.
Atributos	- modality_id (integer) - modality_name (string)
Métodos	ModalityCareerManager() getModalities()
Responsabilidades	- Inicializar el objeto (Constructor de la clase). - Obtener las Modalidades de Estudios disponibles (Presencial, Semipresencial, Distancia).
Colaboradores	ADOdb

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

g. Clase Niveles de Formación (LevelCareerManager)

Tabla 3-17: Iteración 1 - Tarjeta CRC para la Clase LevelCareerManager

Clase	**LevelCareerManager**
Descripción	Clase localizada en la capa del negocio para gestionar la información de los Modalidades de Estudio de las Carreras o Programas Académicos.
Atributos	- type_career_id (integer) - type_career_name (string)
Métodos	LevelCareerManager() getLevels()
Responsabilidades	- Inicializar el objeto (Constructor de la clase). - Obtener los Niveles de Formación disponibles (Nivel Técnico Superior, Tercer Nivel, Cuarto Nivel).
Colaboradores	ADOdb

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

h. Clase Tipo de recursos o materiales (TypeResourceManager)

Tabla 3-18: Iteración 1 - Tarjeta CRC para la Clase TypeResourceManager

Clase	**TypeResourceManager**
Descripción	Clase localizada en la capa del negocio para gestionar la información de los Tipos de recurso o materiales para un curso.
Atributos	- type_resource_id (integer) - type_resource_name (string)
Métodos	TypeResourceManager() getTypeResources()
Responsabilidades	- Inicializar el objeto (Constructor de la clase). - Obtener los Tipos de Recursos o materiales disponibles para un curso (libro, guía, etc.).
Colaboradores	ADOdb

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

i. Clase Recursos o materiales (ResourceManager)

Tabla 3-19: Iteración 1 - Tarjeta CRC para la Clase ResourceManager

Clase	**ResourceManager**
Descripción	Clase localizada en la capa del negocio para gestionar la información de los Recursos o Materiales que puede poseer un curso.
Atributos	- resource_id (integer) - course_code (string) - type_resource (object) - resource_description (string) - resource_cost (float)

Métodos	ResourceManager() read(parámetro) get() set(parámetro) save() delete()
Responsabilidades	- Inicializar el objeto (Constructor de la clase). - Obtener la información de un recurso o material de un curso. - Obtener el código de un objeto ResourceManager (código del recurso o material). - Asignar un nuevo código de recurso o material al código de un objeto ResourceManager (código de recurso o material). - Almacenar la información de un recurso o material de un curso (inserción o actualización). - Eliminar la información de un recurso o material de un curso.
Colaboradores	ADOdb TypeResourceManager

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

3.3.1.2 Implementación

A continuación se muestra el código fuente de las clases antes diseñadas. Para mayor detalle de la implementación ver en el CD adjunto, documento: *Dokeos for University - Implementación/index.html*.

a. Clase Carrera (CareerManager)

Tabla 3-20: Iteración 1 – Implementación de la Clase CareerManager

Clase: CareerManager

```
class CareerManager
{
        var  $career_id;
        var  $type_career_id;
        var  $modality_id;
        var  $system_id;
        var  $career_name;
        var  $career_cost_credit;
        var  $career_administrative_rate;
        var  $career_num_quotas;
        var  $career_duration;
        var  $career_creation_date;
        var  $courseList;
        var  $userList;
        var  $academic_periodList;
        var  $specialitiesList;

        /**
```

```
	* Constructor de la clase CareerManager (Inicializar el objeto)
	*/
	function CareerManager($career_id = null)
	{
		global $conn;
		if($career_id!=null)
		{
			$this->career_id = $career_id;
			// begin Recordset
			$query_rsCareerManager = "SELECT * FROM career WHERE career_id='$this->career_id'";
			$rsCareerManager = $conn->SelectLimit($query_rsCareerManager) or die($conn-
>ErrorMsg());
			$totalRows_rsCareerManager = $rsCareerManager->RecordCount();
			// end Recordset

			if ($totalRows_rsCareerManager > 0)
			{
				$this->type_career_id = $rsCareerManager->Fields('type_career_id');
				$this->modality_id = $rsCareerManager->Fields('modality_id');
				$this->system_id = $rsCareerManager->Fields('system_id');
				$this->career_name = $rsCareerManager->Fields('career_name');
				$this->career_cost_credit = $rsCareerManager->Fields('career_cost_credit');
				$this->career_administrative_rate = $rsCareerManager-
>Fields('career_administrative_rate');
				$this->career_num_quotas = $rsCareerManager->Fields('career_num_quotas');
				$this->career_duration = $rsCareerManager->Fields('career_duration');
				$this->career_creation_date = $rsCareerManager->Fields('career_creation_date');
				// Read Courses
				$this->courseList = $this->getCourses();
				// Read Academic Periods
				$this->academic_periodList = $this->getAcademicPeriods();
				// Read Specialities
				$this->specialitiesList = $this->getSpecialities();
				// Read Student List
				//$this->getUsers($course_code = "%", $academic_period_id = "%", $type_user =
"5");
			}
			else
			{
				$this->career_id = "";
				$this->type_career_id = "";
				$this->modality_id = "";
				$this->system_id = "";
				$this->career_name = "";
				$this->career_cost_credit = "";
				$this->career_administrative_rate = "";
				$this->career_num_quotas = "";
				$this->career_duration = "";
				$this->career_creation_date = "";

				$this->courseList = array();
				$this->userList = array();
				$this->academic_periodList = array();
				$this->specialitiesList = array();
			}
		}
```

```
                else
                {
                        $this->career_id = "";
                        $this->type_career_id = "";
                        $this->modality_id = "";
                        $this->system_id = "";
                        $this->career_name = "";
                        $this->career_cost_credit = "";
                        $this->career_administrative_rate = "";
                        $this->career_num_quotas = "";
                        $this->career_duration = "";
                        $this->career_creation_date = "";
                        $this->courseList = array();
                        $this->userList = array();
                        $this->academic_periodList = array();
                        $this->specialitiesList = array();
                }
        }//end constructor CareerManager

        /**
        * Obtener la información de una carrera  o programa académico
        */
        function read($career_id)
        {
                global $conn;
                $this->career_id = $career_id;
                // begin Recordset
                $query_rsCareerManager = "SELECT * FROM career WHERE career_id='$this->career_id'";
                $rsCareerManager = $conn->SelectLimit($query_rsCareerManager) or die($conn->ErrorMsg());
                $totalRows_rsCareerManager = $rsCareerManager->RecordCount();
                // end Recordset

                if ($totalRows_rsCareerManager > 0)
                {
                        $this->type_career_id = $rsCareerManager->Fields('type_career_id');
                        $this->modality_id = $rsCareerManager->Fields('modality_id');
                        $this->system_id = $rsCareerManager->Fields('system_id');
                        $this->career_name = $rsCareerManager->Fields('career_name');
                        $this->career_cost_credit = $rsCareerManager->Fields('career_cost_credit');
                        $this->career_administrative_rate = $rsCareerManager->Fields('career_administrative_rate');
                        $this->career_num_quotas = $rsCareerManager->Fields('career_num_quotas');
                        $this->career_duration = $rsCareerManager->Fields('career_duration');
                        $this->career_creation_date = $rsCareerManager->Fields('career_creation_date');
                        // Read Courses
                        $this->courseList = $this->getCourses();
                        // Read Academic Period
                        $this->academic_periodList = $this->getAcademicPeriods();
                        // Read Specialities
                        $this->specialitiesList = $this->getSpecialities();
                        // Read Student List
                        //$this->userList = $this->getUsers($course_code = "%", $academic_period_id = "%",
$type_user = "5");
                        return true;
                }
                else
                {
```

```
                $this->career_id = "";
                $this->type_career_id = "";
                $this->modality_id = "";
                $this->system_id = "";
                $this->career_name = "";
                $this->career_cost_credit = "";
                $this->career_administrative_rate = "";
                $this->career_num_quotas = "";
                $this->career_duration = "";
                $this->career_creation_date = "";

                $this->courseList = array();
                $this->userList = array();
                $this->academic_periodList = array();
                $this->specialitiesList = array();
                return false;
        }
    }//end read

    /**
    * Almacenar la información de una carrera o programa académico (inserción o actualización) en la base de datos.
    */
    function save()
    {
        global $conn;
        if ($this->career_id == "" || $this->career_id == 0)
        {
            $insertSQL = sprintf("INSERT INTO career (type_career_id, modality_id, system_id, career_name, career_cost_credit, career_administrative_rate, career_num_quotas, career_duration, career_creation_date) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)",
                GetSQLValueString($this->type_career_id, "int"),
                GetSQLValueString($this->modality_id, "int"),
                GetSQLValueString($this->system_id, "int"),
                GetSQLValueString($this->career_name, "text"),
                GetSQLValueString($this->career_cost_credit, "double"),
                GetSQLValueString($this->career_administrative_rate, "double"),
                GetSQLValueString($this->career_num_quotas, "int"),
                GetSQLValueString($this->career_duration, "int"),
                GetSQLValueString(date("Y-m-d H:i:s"), "date"));
            $result = $conn->Execute($insertSQL) or die($conn->ErrorMsg());
            $this->career_id = mysql_insert_id();
        }
        else
        {
            $updateSQL = sprintf("UPDATE career SET type_career_id=%s, modality_id=%s, system_id=%s, career_name=%s, career_cost_credit=%s, career_administrative_rate=%s, career_num_quotas=%s, career_duration=%s, career_creation_date=%s WHERE career_id=%s",
                GetSQLValueString($this->type_career_id, "int"),
                GetSQLValueString($this->modality_id, "int"),
                GetSQLValueString($this->system_id, "int"),
                GetSQLValueString($this->career_name, "text"),
                GetSQLValueString($this->career_cost_credit, "double"),
                GetSQLValueString($this->career_administrative_rate, "double"),
                GetSQLValueString($this->career_num_quotas, "int"),
                GetSQLValueString($this->career_duration, "int"),
```

```
                        GetSQLValueString($this->career_creation_date, "date"),
                        GetSQLValueString($this->career_id, "int"));
                        $result = $conn->Execute($updateSQL) or die($conn->ErrorMsg());
                }
                return ($conn->Affected_Rows());
}//end save()

/**
* Eliminar la información de una carrera o programa académico de la base de datos.
*/
function delete()
{
                global $conn;
                $deleteSQL = sprintf("DELETE FROM career WHERE career_id=%s",
                GetSQLValueString($this->career_id, "int"));
                $result = $conn->Execute($deleteSQL) or die($conn->ErrorMsg());
                return ($conn->Affected_Rows());
} // end delete()

/**
* Obtener el número de cursos pertenecientes a una carrera o programa académico.
*/
function selectNbrCourses()
{
                return sizeof($this->courseList);
}

/**
* Obtener el número de menciones pertenecientes a una carrera o programa académico.
*/
function selectNbrSpecialities()
{
                return sizeof($this->specialitiesList);
}

/**
* Agregar a la lista de cursos un objeto curso pasado como parámetro.
*/
function addToCourseList($course)
{
                // checks if the course ID is not in the list
                if(!$this->isInList($course->code,$this->courseList))
                {
                        $this->courseList[$course->code] = $course;
                        return true;
                }
                return false;
}

/**
* Agregar a la lista de Períodos Académicos un objeto período académico pasado como parámetro.
*/
function addToAcademicPeriodList($ap)
{
                // checks if the academic period ID is not in the list
                if(!$this->isInList($ap->academic_period_id,$this->academic_periodList))
```

```
		{
			$this->academic_periodList[$ap->academic_period] = $ap;
			return true;
		}
		return false;
	}

	/**
	* Agregar a la lista de menciones un objeto mención pasado como parámetro.
	*/
	function addToSpecialitiesList($speciality)
	{
		// checks if the speciality is not in the list
		if(!$this->isInList($speciality->sp_career_id,$this->specialitiesList))
		{
			$this->specialitiesList[$speciality->sp_career_id] = $speciality;
			return true;
		}
		return false;
	}

	/**
	* Editar la información de un elemento de la lista de menciones.
	*/
	function editToSpecialitiesList($speciality)
	{
		// checks if the speciality is in the list
		if($this->isInList($speciality->sp_career_id,$this->specialitiesList))
		{
			$this->specialitiesList[$speciality->sp_career_id] = $speciality;
			return true;
		}
		return false;
	}

	/**
	* Remover de la lista de cursos un objeto curso pasado como parámetro.
	*/
	function removeFromCourseList($course)
	{
		// checks if the course is in the list
		if($this->isInList($course->code,$this->courseList))
		{
			unset($this->courseList[$course->code]);
			return true;
		}
		return false;
	}

	/**
	* Remover de la lista de Períodos Académicos un objeto período académico pasado como parámetro.
	*/
	function removeFromAcademiPeriodList($ap)
	{
		// checks if the academic period is in the list
		if($this->isInList($ap->academic_period_id,$this->academic_periodList))
```

```
                {
                        unset($this-> academic_periodList[$ap->academic_period_id]);
                        return true;
                }
                return false;
        }

        /**
        * Remover de la lista de menciones un objeto mención pasado como parámetro.
        */
        function removeFromSpecialitiesList($speciality)
        {
                // checks if the speciality is in the list
                if($this->isInList($speciality->sp_career_id,$this->specialitiesList))
                {
                        unset($this->specialitiesList[$speciality->sp_career_id]);
                        return true;
                }
                return false;
        }

        /**
        * Consultar si un objeto determinado se encuentra en un listado de Objetos determinado (Listado de cursos,
períodos académicos o menciones).
        */
        function isInList($id,$list)
        {
                return array_key_exists($id,$list);
        }

        /**
        * Obtener un listado de objetos de carreras registradas.
        */
        function getCareers()
        {
                global $conn;
                $careers = array();
                // begin Recordset
                $query_rsGetCareers = "SELECT * FROM career ORDER BY career_name";
                $rsGetCareers = $conn->SelectLimit($query_rsGetCareers) or die($conn->ErrorMsg());
                $totalRows_rsGetCareers = $rsGetCareers->RecordCount();
                // end Recordset

                while (!$rsGetCareers->EOF) {
                        $careers[] = new CareerManager($rsGetCareers->Fields('career_id'));
                        $rsGetCareers->MoveNext();
                }
                return $careers;
        }

        /**
        * Buscar carreras dado ciertos parámetros de búsqueda.
        */
        function searchCereer($career_name = "",$type_career_id = "",$modality_id = "",$career_duration =
"",$system_id = "")
        {
```

```
		global $conn;
		$careers = array();
		// begin Recordset
		$query_rsSearchCareer = "SELECT * FROM career
WHERE career_name LIKE '%$career_name%' AND type_career_id LIKE '%$type_career_id%' AND modality_id LIKE
'%$modality_id%' AND career_duration LIKE '%$career_duration%' AND system_id LIKE '%$system_id%'
ORDER BY career_name";
		$rsSearchCareer = $conn->SelectLimit($query_rsSearchCareer) or die($conn->ErrorMsg());
		$totalRows_rsSearchCareer = $rsSearchCareer->RecordCount();
		// end Recordset
		while (!$rsSearchCareer->EOF) {
			$careers[] = new CareerManager($rsSearchCareer->Fields('career_id'));
			$rsSearchCareer->MoveNext();
		}
		return $careers;
	}

	/**
	* Obtener un listado de objetos de cursos que pertenecen a una carrera o programa académico.
	*/
	function getCourses()
	{
		global $conn;
		$courses = array();
		// begin Recordset
		$query_rsCoursesList = "SELECT code FROM course c WHERE career_id = '$this->career_id'
ORDER BY title";
		$rsCoursesList = $conn->SelectLimit($query_rsCoursesList) or die($conn->ErrorMsg());
		$totalRows_rsCoursesList = $rsCoursesList->RecordCount();
		// end Recordset
		while (!$rsCoursesList->EOF) {
			$courses[] = $rsCoursesList->Fields('code');
			$rsCoursesList->MoveNext();
		}
		return $courses;
	}

	/**
	* Obtener un listado de objetos de usuarios (profesores, tutores o estudiantes) que pertenecen a una carrera o
programa académico dado ciertos filtros (curso, período académico o tipo de usuario).
	*
	* $type_user (Código del tipo de usuario)
	* - 1: Profesor o Administrador del curso
	* - 5: Estudiante
	* Casos de Obtención de Usuarios
	* - Obtener el Profesor Principal:		$academic_period_id = 0, $type_user = 1;
	* - Obtener Estudiantes:			$academic_period_id = i, $type_user = 5;
	* - Obtener Administradores de Curso:		$academic_period_id = i, $type_user = 1;
	* - Obtener Estudiantes y Administradores de Curso: $academic_period_id = i, $type_user = %;
	* - Obtener todos los usuarios:			$academic_period_id = %, $type_user = %;
	* Donde i = academic period id > 0 and % = Cualquier Código del Período Académico o Tipo de Usuario
	*/
	function getUsers($course_code = "%", $academic_period_id = "%", $type_user = "%")
	{
		global $conn;
		$this->userList = array();
```

```
                        // begin Recordset
                        $query_rsUserList = "SELECT cu.* FROM course_rel_user cu, course c, user u WHERE c.code =
cu.course_code AND u.user_id = cu.user_id AND c.career_id = '$this->career_id' AND cu.status LIKE '$type_user' AND
cu.academic_period_id LIKE '$academic_period_id' AND cu.course_code LIKE '$course_code' GROUP BY cu.user_id";
                        //echo $query_rsUserList;
                        $rsUserList = $conn->SelectLimit($query_rsUserList) or die($conn->ErrorMsg());
                        $totalRows_rsUserList = $rsUserList->RecordCount();
                        // end Recordset
                        while (!$rsUserList->EOF) {
                                $this->userList[] = new UserManager($rsUserList->Fields('user_id'));
                                $rsUserList->MoveNext();
                        }
                }

                /**
                * Obtener un listado de objetos de períodos académicos que pertenecen a una carrera o programa académico.
                */
                function getAcademicPeriods()
                {
                        global $conn;
                        $ap = array();
                        // begin Recordset
                        $query_rsAcademicPeriodList = "SELECT academic_period_id FROM academic_period ac, career ca
WHERE ca.career_id = ac.career_id AND ca.career_id = '$this->career_id' ORDER BY status";
                        $rsAcademicPeriodList = $conn->SelectLimit($query_rsAcademicPeriodList) or die($conn-
>ErrorMsg());
                        $totalRows_rsAcademicPeriodList = $rsAcademicPeriodList->RecordCount();
                        // end Recordset
                        while (!$rsAcademicPeriodList->EOF) {
                                $ap[] = $rsAcademicPeriodList->Fields('academic_period_id');
                                $rsAcademicPeriodList->MoveNext();
                        }
                        return $ap;
                }

                /**
                * Obtener el Período Académico Actual o Activo de una carrera o programa académico.
                */
                function getCurrentAcademicPeriod()
                {
                        global $conn;
                        // begin Recordset
                        $query_rsCurrentAcademicPeriod = "SELECT * FROM career ca, academic_period ap WHERE
ca.career_id='$this->career_id' AND ca.career_id = ap.career_id AND status = 'Actual'";
                        $rsCurrentAcademicPeriod = $conn->SelectLimit($query_rsCurrentAcademicPeriod) or die($conn-
>ErrorMsg());
                        $totalRows_rsCurrentAcademicPeriod = $rsCurrentAcademicPeriod->RecordCount();
                        // end Recordset

                        if($totalRows_rsCurrentAcademicPeriod > 0){
                                return new AcademicPeriodManager($rsCurrentAcademicPeriod-
>Fields('academic_period_id'));
                        }

                        return null;
                }
```

```
	/**
	* Obtener un listado de objetos de menciones relacionadas a una carrera o programa académico.
	*/
	function getSpecialities()
	{
		global $conn;
		$specialities = array();
		// begin Recordset
		$query_rsSpecialities = "SELECT * FROM speciality_career WHERE career_id='$this->career_id' ";
		$rsSpecialities = $conn->SelectLimit($query_rsSpecialities) or die($conn->ErrorMsg());
		$totalRows_rsSpecialities = $rsSpecialities->RecordCount();
		// end Recordset
		while (!$rsSpecialities->EOF) {
			$specialities[$rsSpecialities->Fields('sp_career_id')] = new
SpecialityCareerManager($rsSpecialities->Fields('sp_career_id'));
			$rsSpecialities->MoveNext();
		}
		return $specialities;
	}

} // end class CareerManager
```

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

b. Clase Course (CourseManager)

Tabla 3-21: Iteración 1 – Implementación de la Clase CourseManager

Clase: CourseManager

```
class CourseManager
{
	var  $code;
	var  $directory;
	var  $db_name;
	var  $course_language;
	var  $title;
	var  $description;
	var  $category_code;
	var  $visibility;
	var  $show_score;
	var  $tutor_name;
	var  $visual_code;
	var  $department_name;
	var  $department_url;
	var  $disk_quota;
	var  $last_visit;
	var  $last_edit;
	var  $creation_date;
	var  $expiration_date;
	var  $target_course_code;
	var  $subscribe;
	var  $unsubscribe;
	var  $career_id;
	var  $type_course_id;
	var  $credits;
```

```
var  $cycle;
var  $sp_career_id;

var  $tutor_id;  // integer with user id of teacher of course
var  $resourceList;  // array with the list of course's resources
var  $userList;  // array with the list of course's user
var  $cost_course;

/**
* Constructor de la clase Course (Inicializar el objeto)
*/
function CourseManager($code = null)
{
	global $conn;
	if($code!=null)
	{
		$this->code = $code;
		// begin Recordset
		$query_rsCourseId = "SELECT * FROM course WHERE code='$this->code'";
		$rsCourseId = $conn->SelectLimit($query_rsCourseId) or die($conn->ErrorMsg());
		$totalRows_rsCourseId = $rsCourseId->RecordCount();
		// end Recordset

		if ($totalRows_rsCourseId>0)
		{
			$this->directory = $rsCourseId->Fields('directory');
			$this->db_name = $rsCourseId->Fields('db_name');
			$this->course_language = $rsCourseId->Fields('course_language');
			$this->title = $rsCourseId->Fields('title');
			$this->description = $rsCourseId->Fields('description');
			$this->category_code = $rsCourseId->Fields('category_code');
			$this->visibility = $rsCourseId->Fields('visibility');
			$this->show_score = $rsCourseId->Fields('show_score');
			$this->tutor_name = $rsCourseId->Fields('tutor_name');
			$this->visual_code = $rsCourseId->Fields('visual_code');
			$this->department_name = $rsCourseId->Fields('department_name');
			$this->department_url = $rsCourseId->Fields('department_url');
			$this->disk_quota = $rsCourseId->Fields('disk_quota');
			$this->last_visit = $rsCourseId->Fields('last_visit');
			$this->last_edit = $rsCourseId->Fields('last_edit');
			$this->creation_date = $rsCourseId->Fields('creation_date');
			$this->expiration_date = $rsCourseId->Fields('expiration_date');
			$this->target_course_code = $rsCourseId->Fields('target_course_code');
			$this->subscribe = $rsCourseId->Fields('subscribe');
			$this->unsubscribe = $rsCourseId->Fields('unsubscribe');
			$this->career_id = $rsCourseId->Fields('career_id');
			$this->type_course_id = $rsCourseId->Fields('type_course_id');
			$this->credits = $rsCourseId->Fields('credits');
			$this->cycle = $rsCourseId->Fields('cycle');
			$this->sp_career_id = $rsCourseId->Fields('sp_career_id');
			// Get Teacher id
			$tutor = $this->getUsersOfCourse($academic_period_id = "0", $type_user = "1");
			foreach($tutor as $t)		$this->tutor_id = $t->user_id;

			$career = new CareerManager($this->career_id);
			$this->cost_course = $career->career_cost_credit * $this->credits;
```

```
                                $this->userList = array();
                                // Resource List
                                // begin Recordset
                                $query_rsResourcesCourse = "SELECT resource_id FROM course c, resource r
WHERE c.code = r.course_code AND r.course_code = '$this->code'";
                                $rsResourcesCourse = $conn->SelectLimit($query_rsResourcesCourse) or
die($conn->ErrorMsg());
                                $totalRows_rsResourcesCourse = $rsResourcesCourse->RecordCount();
                                // end Recordset
                                while (!$rsResourcesCourse->EOF) {
                                        $this->resourceList[] = $rsResourcesCourse->Fields('resource_id');
                                        $rsResourcesCourse->MoveNext();
                                }
                        }
                        else
                        {
                                $this->directory = "";                  $this->db_name = "";
                                $this->course_language = "";            $this->title = "";
                                $this->description = "";                $this->category_code = "";
                                $this->visibility = "";                 $this->show_score = "";
                                $this->tutor_name = "";                 $this->visual_code = "";
                                $this->department_name = "";            $this->department_url = "";
                                $this->disk_quota = "";                 $this->last_visit = "";
                                $this->last_edit = "";                  $this->creation_date = "";
                                $this->expiration_date = "";            $this->target_course_code = "";
                                $this->subscribe = "";                  $this->unsubscribe = "";
                                $this->career_id = "";                  $this->type_course_id = "";
                                $this->credits = "";                    $this->cycle = "";
                                $this->tutor_id = "";                   $this->cost_course = 0;
                                $this->sp_career_id = "";
                                $this->resourceList = array();          $this->userList = array();
                        }
                }
                else
                {
                        $this->directory = "";                  $this->db_name = "";
                        $this->course_language = "";            $this->title = "";
                        $this->description = "";                $this->category_code = "";
                        $this->visibility = "";                 $this->show_score = "";
                        $this->tutor_name = "";                 $this->visual_code = "";
                        $this->department_name = "";            $this->department_url = "";
                        $this->disk_quota = "";                 $this->last_visit = "";
                        $this->last_edit = "";                  $this->creation_date = "";
                        $this->expiration_date = "";            $this->target_course_code = "";
                        $this->subscribe = "";                  $this->unsubscribe = "";
                        $this->career_id = "";                  $this->type_course_id = "";
                        $this->credits = "";                    $this->cycle = "";
                        $this->tutor_id = "";                   $this->cost_course = 0;
                        $this->sp_career_id = "";
                        $this->resourceList = array();          $this->userList = array();

                }
        }//end constructor CourseManager

        /**
```

```
	* Obtener la información del curso.
	*/
	function read($code)
	{
		global $conn;
		$this->code = $code;
		// begin Recordset
		$query_rsCourseId = "SELECT * FROM course  WHERE code='$this->code'";
		$rsCourseId = $conn->SelectLimit($query_rsCourseId) or die($conn->ErrorMsg());
		$totalRows_rsCourseId = $rsCourseId->RecordCount();
		// end Recordset

		if ($totalRows_rsCourseId>0)
		{
			$this->directory = $rsCourseId->Fields('directory');
			$this->db_name = $rsCourseId->Fields('db_name');
			$this->course_language = $rsCourseId->Fields('course_language');
			$this->title = $rsCourseId->Fields('title');
			$this->description = $rsCourseId->Fields('description');
			$this->category_code = $rsCourseId->Fields('category_code');
			$this->visibility = $rsCourseId->Fields('visibility');
			$this->show_score = $rsCourseId->Fields('show_score');
			$this->tutor_name = $rsCourseId->Fields('tutor_name');
			$this->visual_code = $rsCourseId->Fields('visual_code');
			$this->department_name = $rsCourseId->Fields('department_name');
			$this->department_url = $rsCourseId->Fields('department_url');
			$this->disk_quota = $rsCourseId->Fields('disk_quota');
			$this->last_visit = $rsCourseId->Fields('last_visit');
			$this->last_edit = $rsCourseId->Fields('last_edit');
			$this->creation_date = $rsCourseId->Fields('creation_date');
			$this->expiration_date = $rsCourseId->Fields('expiration_date');
			$this->target_course_code = $rsCourseId->Fields('target_course_code');
			$this->subscribe = $rsCourseId->Fields('subscribe');
			$this->unsubscribe = $rsCourseId->Fields('unsubscribe');
			$this->career_id = $rsCourseId->Fields('career_id');
			$this->type_course_id = $rsCourseId->Fields('type_course_id');
			$this->credits = $rsCourseId->Fields('credits');
			$this->cycle = $rsCourseId->Fields('cycle');
			$this->sp_career_id = $rsCourseId->Fields('sp_career_id');
			$this->resourceList = array();
			$this->userList = array();
			// Get Teacher id
			$tutor = $this->getUsersOfCourse($academic_period_id = "0", $type_user = "1");
			foreach($tutor as $t)		$this->tutor_id = $t->user_id;

			$career = new CareerManager($this->career_id);
			$this->cost_course = $career->career_cost_credit * $this->credits;

			// begin Recordset
			$query_rsResourcesCourse = "SELECT resource_id FROM course c, resource r WHERE
c.code = r.course_code AND r.course_code = '$this->code'";
			$rsResourcesCourse = $conn->SelectLimit($query_rsResourcesCourse) or die($conn-
>ErrorMsg());
			$totalRows_rsResourcesCourse = $rsResourcesCourse->RecordCount();
			// end Recordset
			while (!$rsResourcesCourse->EOF) {
```

```
                                $this->resourceList[] = $rsResourcesCourse->Fields('resource_id');
                                $rsResourcesCourse->MoveNext();
                        }

                        $career = new CareerManager($this->career_id);
                        $this->cost_course = $career->career_cost_credit * $this->credits;

                        return true;
                }
                else
                {
                        $this->directory = "";                  $this->db_name = "";
                        $this->course_language = "";            $this->title = "";
                        $this->description = "";                $this->category_code = "";
                        $this->visibility = "";                 $this->show_score = "";
                        $this->tutor_name = "";                 $this->visual_code = "";
                        $this->department_name = "";            $this->department_url = "";
                        $this->disk_quota = "";                 $this->last_visit = "";
                        $this->last_edit = "";                  $this->creation_date = "";
                        $this->expiration_date = "";            $this->target_course_code = "";
                        $this->subscribe = "";                  $this->unsubscribe = "";
                        $this->career_id = "";                  $this->type_course_id = "";
                        $this->credits = "";                    $this->cycle = "";
                        $this->tutor_id = "";                   $this->cost_course = 0;
                        $this->sp_career_id = "";
                        $this->resourceList = array();          $this->userList = array();

                        return false;
                }
        }//end read Course

        /**
        * Almacenar la información de un curso (inserción o actualización) en la base de datos.
        */
        function save()
        {
                global $conn;
                if ($this->code == "" || $this->code == null)
                {
                        $insertSQL = sprintf("INSERT INTO course (code, directory, db_name, course_language,
title, `cycle`, `description`, category_code, visibility, show_score, tutor_name, visual_code, department_name,
department_url, disk_quota, last_visit, last_edit, creation_date, expiration_date, target_course_code, subscribe,
unsubscribe, career_id, type_course_id, credits, sp_career_id) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s,
%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)",
                                GetSQLValueString($this->code, "text"),GetSQLValueString($this->directory, "text"),
                                GetSQLValueString($this->db_name, "text"),GetSQLValueString($this->course_language, "text"),
                                GetSQLValueString($this->title, "text"),GetSQLValueString($this->cycle, "int"),
                                GetSQLValueString($this->description, "text"),GetSQLValueString($this->category_code, "text"),
                                GetSQLValueString($this->visibility, "int"),GetSQLValueString($this->show_score, "int"),
                                GetSQLValueString($this->tutor_name, "text"),GetSQLValueString($this->visual_code, "text"),
                                GetSQLValueString($this->department_name, "text"),GetSQLValueString($this->department_url,
"text"),
                                GetSQLValueString($this->disk_quota, "int"),GetSQLValueString($this->last_visit, "date"),
                                GetSQLValueString($this->last_edit, "date"),GetSQLValueString($this->creation_date, "date"),
                                GetSQLValueString($this->expiration_date, "date"),GetSQLValueString($this-
>target_course_code, "text"),
```

```
                        GetSQLValueString($this->subscribe, "int"),GetSQLValueString($this->unsubscribe, "int"),
                        GetSQLValueString($this->career_id, "int"),GetSQLValueString($this->type_course_id, "int"),
                        GetSQLValueString($this->sp_career_id, "int"),GetSQLValueString($this->credits, "int"));

                        $result = $conn->Execute($insertSQL) or die($conn->ErrorMsg());
                        $this->code = mysql_insert_id();
                }
                else
                {
                        $updateSQL = sprintf("UPDATE course SET directory=%s, db_name=%s, course_language=%s,
title=%s, `cycle`=%s, `description`=%s, category_code=%s, visibility=%s, show_score=%s, tutor_name=%s,
visual_code=%s, department_name=%s, department_url=%s, disk_quota=%s, last_visit=%s, last_edit=%s,
creation_date=%s, expiration_date=%s, target_course_code=%s, subscribe=%s, unsubscribe=%s, career_id=%s,
type_course_id=%s, credits=%s, sp_career_id=%s WHERE code=%s",
                        GetSQLValueString($this->directory, "text"),GetSQLValueString($this->db_name, "text"),
                        GetSQLValueString($this->course_language, "text"),GetSQLValueString($this->title, "text"),
                        GetSQLValueString($this->cycle, "int"),GetSQLValueString($this->description, "text"),
                        GetSQLValueString($this->category_code, "text"),GetSQLValueString($this->visibility, "int"),
                        GetSQLValueString($this->show_score, "int"),GetSQLValueString($this->tutor_name, "text"),
                      GetSQLValueString($this->visual_code, "text"),GetSQLValueString($this->department_name, "text"),
                        GetSQLValueString($this->department_url, "text"),GetSQLValueString($this->disk_quota, "int"),
                        GetSQLValueString($this->last_visit, "date"),GetSQLValueString($this->last_edit, "date"),
                     GetSQLValueString($this->creation_date, "date"),GetSQLValueString($this->expiration_date, "date"),
                        GetSQLValueString($this->target_course_code, "text"),GetSQLValueString($this->subscribe,
"int"),
                        GetSQLValueString($this->unsubscribe, "int"),GetSQLValueString($this->career_id, "int"),
                        GetSQLValueString($this->type_course_id, "int"),GetSQLValueString($this->credits, "int"),
                        GetSQLValueString($this->sp_career_id, "int"),GetSQLValueString($this->code, "text"));
                        $result = $conn->Execute($updateSQL) or die($conn->ErrorMsg());
                }
                return ($conn->Affected_Rows());
        }//end save()

        /**
        * Eliminar la información de un curso de la base de datos.
        */
        function delete($code)
        {
                global $conn;
                $deleteSQL = sprintf("DELETE FROM course WHERE code=%s",
                GetSQLValueString($this->code, "text"));
                $result = $conn->Execute($deleteSQL) or die($conn->ErrorMsg());
                return ($conn->Affected_Rows());
        } // end delete()

        /**
        * Obtener el número de usuarios de un curso. Este número será relativo, es decir dependerá del tipo de usuario
y el período académico.
        */
        function selectNbrUsers()
        {
                return sizeof($this->userList);
        }

        /**
        * Agregar a la lista de recursos o materiales un objeto recurso pasado como parámetro.
```

```
*/
function addToResourceList($resource)
{
	// checks if the resource ID is not in the list
	if(!$this->isInList($resource->resource_id,$this->resourceList))
	{
		$this->userList[$resource->resource_id]=$resource;
		return true;
	}
	return false;
}

/**
* Agregar a la lista de usuarios un objeto usuario pasado como parámetro.
*/
function addToUserList($user)
{
	// checks if the user ID is not in the list
	if(!$this->isInList($user->user_id,$this->userList))
	{
		$this->userList[$user->user_id]=$user;
		return true;
	}
	return false;
}

/**
* Remover de la lista de usuarios un objeto usuario pasado como parámetro.
*/
function removeFromUserList($user)
{
	// checks if the user is in the list
	if($this->isInList($user-> user_id,$this->userList))
	{
		unset($this->userList[$user->user_id]);
		return true;
	}
	return false;
}

/**
* Remover de la lista de Recursos o Materiales un objeto recurso pasado como parámetro.
*/
function removeFromResourceList($resource)
{
	// checks if the resource is in the list
	if($this->isInList($resource->resource_id,$this->resourceList))
	{
		unset($this->resourceList[$resource->resource_id]);
		return true;
	}
	return false;
}

/**
* Consultar si un objeto determinado se encuentra en un listado de Objetos determinado (Listado de usuarios,
```

```
recursos).
		*/
		function isInList($id,$list)
		{
			return array_key_exists($id,$list);
		}

		/**
		* Obtener el costo total de recursos de un curso.
		*/
		function getTotalResourcesOfCourse($code)
		{
			global $conn;
			// begin Recordset
			$query_rsTotalResourcesCourse = "SELECT SUM(resource_cost) total_resource FROM course c,
resource r WHERE c.code = r.course_code AND r.course_code = '$code'";
			$rsTotalResourcesCourse = $conn->SelectLimit($query_rsTotalResourcesCourse) or die($conn-
>ErrorMsg());
			$totalRows_rsTotalResourcesCourse = $rsTotalResourcesCourse->RecordCount();
			// end Recordset

			if($rsTotalResourcesCourse->Fields('total_resource') > 0)	return $rsTotalResourcesCourse-
>Fields('total_resource');
			else		return "0";
		}

		/**
		* Obtener los usuarios de un curso dado un cierto período académico y tipo de usuario.
		*/
		function getUsersOfCourse($academic_period_id = "%", $type_user = "%")
		{
			global $conn;
			$users = array();
			// begin Recordset
			$query_rsUsersCourse = "SELECT cu.* FROM course c, course_rel_user cu WHERE c.code =
cu.course_code AND cu.course_code = '$this->code' AND academic_period_id LIKE '$academic_period_id' AND cu.status
LIKE '$type_user'";
			$rsUsersCourse = $conn->SelectLimit($query_rsUsersCourse) or die($conn->ErrorMsg());
			$totalRows_rsUsersCourse = $rsUsersCourse->RecordCount();
			// end Recordset
			while (!$rsUsersCourse->EOF) {
				$users[$rsUsersCourse->Fields('user_id')] = new UserManager($rsUsersCourse-
>Fields('user_id'));
				$rsUsersCourse->MoveNext();
			}
			return $users;
		}

		/**
		* Obtener los cursos de capacitación o entrenamiento. Estos cursos no pertenecerán a una carrera o programa
académico.
		*/
		function getTrainingCourses()
		{
			global $conn;
			$courses = array();
```

```
                $query_rsCoursesList = "SELECT code FROM course c WHERE career_id = '0' AND type_course_id
= '1' ORDER BY title";
                $rsCoursesList = $conn->SelectLimit($query_rsCoursesList) or die($conn->ErrorMsg());
                $totalRows_rsCoursesList = $rsCoursesList->RecordCount();
                // end Recordset
                while (!$rsCoursesList->EOF) {
                        $courses[$rsCoursesList->Fields('code')] = new Cours($rsCoursesList->Fields('code'));
                        $rsCoursesList->MoveNext();
                }
                return $courses;
        }

} // end class CourseManager
```

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

NOTA: La implementación de las demás clases de que intervienen en esta Iteración estan descritas en detalle en el CD adjunto, documento: *Dokeos for University - Implementación/index.html*.

3.3.1.3 Pruebas

De acuerdo a lo estipulado en el capítulo 2, en la explicación de la metodología Extreme Programming se planteará un plan de pruebas para cada una de las Historias de Usuario de la Iteración 1

Plan de Pruebas

HISTORIA 1

- **Creación de carreras o programas académicos**

Esta prueba consiste en la creación de una carrera o programa académico.

Tabla 3-22: Casos de Prueba: Creación de carreras o programas académicos

Caso de Prueba	Entrada	Resultado Esperado
Ingresar los datos correctos en los distintos campos de ingreso.	Se ingresa los datos correctos que identifican a la carrera.	El sistema debe guardar los datos de la nueva carrera creada previa a la verificación de los mismos y trasladar al listado de carreras.
Obviar el ingreso de datos en campos obligatorios.	Se omite el ingreso de datos en los campos de ingreso que son obligatorios.	El sistema debe emitir un mensaje informando que se deben ingresar todos los datos para poder guardar correctamente a la carrera.
Ingreso de datos alfabéticos en campos numéricos.	Se ingresa datos alfabéticos en campos numéricos.	El sistema no debe permitir el ingreso de datos alfabéticos en campos numéricos y debe mostrar un mensaje de error que informe que existe un error en el tipo de dato

		ingresado (campo inválido).

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Listado de carreras o programas académicos**

Esta prueba consiste en la presentación de una lista de carreras o programas académicos.

Tabla 3-23: Casos de Prueba: Listado de carreras o programas académicos

Caso de Prueba	Entrada	Resultado Esperado
Listar las carreras o programas académicos registrados.	Seleccionar la opción Carreras/Lista de carreras.	El sistema debe presentar un listado de carreras registradas en el sistema con la siguiente información: título, modalidad de estudios, nivel de formación duración, número de cursos de la carrera, el período académico actual y las opciones de administración (visualización, actualización, eliminación, ingreso al área de gestión de pre-requisitos/co-requisitos, períodos académicos, menciones)
No existir carreras o programas académicos registrados.	Seleccionar la opción Carreras/Lista de carreras.	El sistema debe mostrar un mensaje que "No existen carreras registradas".

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

HISTORIA 2

- **Visualización detallada de una carrera o programa académico**

Esta prueba consiste en la visualización detallada de una carrera o programa académico.

Tabla 3-24: Casos de Prueba: Visualización detallada de una carrera o programa académico

Caso de Prueba	Entrada	Resultado Esperado
Mostrar la información detallada de una carrera o programa académico.	Seleccionar la carrera a mostrar la información detallada.	El sistema debe mostrar la información detallada de la carrera o programa académico que incluya el nombre, la modalidad de estudios, el nivel de formación, duración, los cursos que pertenecen a la carrera, etc.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Actualizar carreras o programas académicos**

Esta prueba consiste en la actualización o modificación de una carrera o programa académico.

Tabla 3-25: Casos de Prueba: Actualizar carreras o programas académicos

Caso de Prueba	Entrada	Resultado Esperado
Actualizar datos correctos en los distintos campos de ingreso de la carrera.	Ingreso los nuevos datos en los campos que se desee actualizar de la carrera.	El sistema debe buscar y leer a la carrera y debe actualizar los datos correspondientes y trasladar al listado de carreras con la operación efectuada con exito.
Actualización con datos nulos (vacíos) en campos de ingreso obligatorio.	Actualización de campos que requieren información obligatoria con datos nulos.	El sistema no debe permitir la actualización y debe emitir un mensaje de alerta informando que se deben ingresar todos los datos para poder guardar correctamente a la carrera.
Actualización de datos alfabéticos en campos numéricos.	Se ingresa datos alfabéticos en campos destinados a números.	El sistema no debe permitir el ingreso de datos alfabéticos en campos numéricos y debe mostrar un mensaje de error que informe que existe un error en el tipo de dato ingresado (campo inválido).

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Eliminar carreras o programas académicos**

Esta prueba consiste en la eliminación de una o varias carreras o programas académicos. Cabe mencionar que el sistema solamente eliminará la información de la carrera y no los cursos que dicha carrera pueda tener asociada.

Tabla 3-26: Casos de Prueba: Eliminar carreras o programas académicos

Caso de Prueba	Entrada	Resultado Esperado
Eliminación individual de carreras o programas académicos.	Se selecciona a la carrera que se desea eliminar.	El sistema debe emitir un mensaje de confirmación solicitando al usuario que confirme la acción a realizar.
Eliminación en grupo de carreras o programas académicos.	Se selecciona (de un checkbox) las carreras que se desean eliminar.	El sistema debe presentar la opción de eliminación de carreras seleccionadas y cuando el usuario selecciones esta opción el sistema debe emitir un mensaje de confirmación solicitando al usuario que confirme la acción a realizar.
Cancelación de la acción de eliminación de carreras o programas académicos.	El administrador procede con la eliminación individual o grupal y se le presenta el mensaje de confirmación.	El sistema al momento que se le presente el mensaje de confirmación con las opciones de aceptar o cancelar y cuando selecciona la opción de Cancelar, el sistema debe detener la acción de eliminación.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Buscar carreras o programas académicos**

Esta prueba consiste en la búsqueda de una o varias carreras o programas académicos.

Tabla 3-27: Casos de Prueba: Buscar carreras o programas académicos

Caso de Prueba	Entrada	Resultado Esperado
Búsqueda simple de carreras o programas académicos existentes.	Se ingresará una palabra clave que relacione al nombre de la carrera.	El sistema debe buscar las carreras que coincidan en el nombre con la palabra clave y debe presentar un listado de las mismas.
Búsqueda avanzada de carreras o programas académicos existentes.	Se ingresará ciertos parámetros de búsqueda (nombre, nivel de formación, modalidad de estudios, etc.).	El sistema debe buscar las carreras que coincidan con los parámetros de búsqueda y debe presentar un listado de las mismas.
Búsqueda simple o avanzada de carreras o programas académicos que no existen.	Se ingresará ciertos parámetros de búsqueda (nombre, nivel de formación, modalidad de estudios, etc.).	El sistema debe buscar las carreras que coincidan con los parámetros de búsqueda y si no lo encuentra debe mostrar un mensaje que "No se encontraron resultados".

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Dar de baja a los cursos que pertenecen a una carrera o programa académico**

Esta prueba consiste en desasociar a un curso de una carrera o programa académico.

Tabla 3-28: Casos de Prueba: Dar de baja a los cursos que pertenecen a una carrera o programa académico

Caso de Prueba	Entrada	Resultado Esperado
Dar de baja a los cursos que pertenecen a una carrera o programa académico.	En listado de cursos de la carrera, seleccionar a los cursos que se desea dar de baja.	El sistema debe presentar la opción de dar de baja a los cursos seleccionados del listado de cursos de la información detallada de la carrera. Cuando el usuario seleccione a los cursos a dar de baja el sistema debe emitir un mensaje de confirmación solicitando al usuario que confirme la acción a realizar. Con la confirmación el sistema debe desasociar a los cursos seleccionados de la carrera.
Cancelación de la acción de dar de baja a los cursos de una carrera o programa académico.	El administrador procede con la baja de los cursos seleccionados y se le presenta el mensaje de confirmación.	El sistema al momento que se le presente el mensaje de confirmación con las opciones de aceptar o cancelar y si el Administrador selecciona la opción de Cancelar, el sistema debe detener la acción de bajo de los cursos seleccionados.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

Procedimiento de Pruebas

Para cada uno de los procedimientos de prueba de esta iteración, el usuario

administrador tendrá que realizar los siguientes procedimientos básicos para acceder a las opciones de "*Administración de la Plataforma*".

1. En la pantalla correspondiente de ingreso al sistema el administrador de la plataforma ingresa su login y password.

2. El usuario administrador presiona el botón *OK.*

3. El sistema hace la verificación de los datos ingresados y si tiene perfil de administrador de la plataforma.

4. El administrador seleccionará la opción "*Administración de la Plataforma*" y encontrará los módulos de administración de usuarios, cursos, carrera y plataforma.

HISTORIA 1

- **Procedimiento de prueba para el caso de prueba de Creación de carreras o programas académicos**

  1 En el módulo correspondiente a la administración de Carreras seleccionar la opción de *Crear una carrera*.

  2 En el formulario, el administrador ingresará los datos solicitados y presiona el botón *OK*.

  3 El sistema verifica que todos los casos esté completos y correctos para proceder con la creación, en caso de no cumplir con esta normativa, el sistema despliega un mensaje de error indicando que se ingresen los datos completos y correctos.

- **Procedimiento de prueba para el caso de prueba Listado de carreras o programas académicos**

  1 En el módulo correspondiente a la administración de Carreras seleccionar la opción de *Lista de carreras*.

  2 El sistema presenta un listado de las carreras registradas con la

siguiente información: título, modalidad de estudios, nivel de formación duración, número de cursos de la carrera, el período académico actual y las opciones de administración (visualización, actualización, eliminación, ingreso al área de gestión de pre-requisitos/co-requisitos, períodos académicos, menciones)

HISTORIA 2

- **Procedimiento de prueba para el caso de prueba Visualización detallada de una carrera o programa académico**

  1 En el módulo correspondiente a la administración de Carreras seleccionar la opción de *Lista de carreras*".

  2 En el listado, el administrador pulsará el botón de *Información* que se presenta en cada una de las carreras registradas.

- **Actualizar carreras o programas académicos**

  1 En el módulo correspondiente a la administración de *Carreras* seleccionar la opción de *Lista de carreras*".

  2 En el listado, el administrador pulsará el botón de *Modificar* que se presenta en cada una de las carreras registradas.

  3 Se presenta un formulario con los datos de la carrera, el administrador actualiza los datos correspondientes y presiona el botón OK.

  4 El sistema verifica que todos los casos esté completos y correctos para proceder con la actualización, en caso de no cumplir con esta normativa, el sistema despliega un mensaje de error indicando que se ingresen los datos completos y correctos.

- **Eliminar carreras o programas académicos**

  1 En el módulo correspondiente a la administración de Carreras seleccionar la opción de *Lista de carreras*".

2 En el listado, el administrador pulsará el botón de *Eliminar* que se presenta en cada una de las carreras registradas.

3 Se presenta un mensaje de confirmación de la acción a realizar

4 El sistema elimina la información relacionada con la carrera

- **Buscar carreras o programas académicos**

1 En el módulo correspondiente a la administración de Carreras seleccionar la opción de *Lista de carreras*".

2 En la página del listado, el administrador efectuará una búsqueda simple o avanzada.

2.1 *Búsqueda Simple*: ingresará una palabra clave que relacione al nombre de la carrera y el usuario presionará el *Buscar*.

2.2 *Búsqueda Avanzada*: ingresarán palabras claves que relacione al nombre de la carrera, modalidad, nivel, etc., y presionará el *Ok*.

3 El sistema presenta un listado de las carreras que coincidan con los parámetros de búsqueda.

- **Dar de baja a los cursos que pertenecen a una carrera o programa académico**

1 En el módulo correspondiente a la administración de Carreras seleccionar la opción de *Lista de carreras*".

2 En el listado, el administrador pulsará el botón de *Información* que se presenta en cada una de las carreras registradas.

3 En la información detallada de la carrera, se muestran un listado de los cursos que pertenecen a esa carrera, y el administrador seleccionará a los cursos que desea dar de baja (desasociación de los cursos de la carrera).

4 Se presenta un mensaje de confirmación de la acción a realizar

5 El sistema da de baja la relación entre el curso y la carrera.

Resultado de Pruebas

HISTORIA 1

- **Creación de carreras o programas académicos**

Esta prueba consiste en la creación de una carrera o programa académico.

Tabla 3-29: Resultados de los Casos de Prueba: Creación de carreras o programas académicos

Caso de Prueba	Entrada	Resultado Obtenido
Ingresar los datos correctos en los distintos campos de ingreso.	Se ingresa los datos correctos que identifican a la carrera.	El sistema hace una verificación de los datos y guarda los datos de la nueva carrera creada y traslada al listado de carreras.
Obviar el ingreso de datos en campos obligatorios.	Se omite el ingreso de datos en los campos de ingreso que son obligatorios.	El sistema emite un mensaje informando que se deben ingresar todos los datos para proceder con el almacenamiento en la base de datos de la carrera.
Ingreso de datos alfabéticos en campos numéricos.	Se ingresa datos alfabéticos en campos numéricos.	El sistema no permite el ingreso de datos alfabéticos en campos numéricos y muestra un mensaje de error informando que el tipo de dato ingresado no es el correcto en un campo determinado.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Listado de carreras o programas académicos**

Esta prueba consiste en la presentación de una lista de carreras o programas académicos.

Tabla 3-30: Resultados de los Casos de Prueba: Listado de carreras o programas académicos

Caso de Prueba	Entrada	Resultado Obtenido
Listar las carreras o programas académicos registrados.	Seleccionar la opción Carreras/Lista de carreras.	El sistema muestra un listado de las carreras registradas en el sistema con toda la información requerida por el cliente.
No existir carreras o programas académicos registrados.	Seleccionar la opción Carreras/Lista de carreras.	El sistema emite un mensaje informando al usuario de que "No existen carreras registradas".

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

HISTORIA 2

- **Visualización detallada de una carrera o programa académico**

Esta prueba consiste en la visualización detallada de una carrera o programa académico.

Tabla 3-31: Resultados de los Casos de Prueba: Visualización detallada de una carrera o programa académico

Caso de Prueba	Entrada	Resultado Obtenido
Mostrar la información detallada de una carrera o programa académico.	Seleccionar la carrera a mostrar la información detallada.	El sistema presenta un reporte con toda la información detallada de la carrera o programa académico con la siguiente información: nombre, modalidad de estudios, nivel de formación, duración, los cursos que pertenecen a la carrera, etc.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Actualizar carreras o programas académicos**

Esta prueba consiste en la actualización o modificación de una carrera o programa académico.

Tabla 3-32: Resultados de los Casos de Prueba: Actualizar carreras o programas académicos

Caso de Prueba	Entrada	Resultado Obtenido
Actualizar datos correctos en los distintos campos de ingreso de la carrera.	Ingreso de los nuevos datos en los campos que se desee actualizar de la carrera.	El sistema busca y lee toda la información de la carrera y actualiza los datos correspondientes ingresados por el usuario de la carrera, trasladando al listado de carreras con la operación efectuada con éxito.
Actualización con datos nulos (vacíos) en campos de ingreso obligatorio.	Actualización de campos que requieren información obligatoria con datos nulos.	El sistema emite un mensaje de alerta informando la obligatoriedad de ingresar todos los datos y no ejecuta la actualización de la carrera.
Actualización de datos alfabéticos en campos numéricos.	Se ingresa datos alfabéticos en campos destinados a números.	El sistema muestra un mensaje de error informando que el tipo de dato ingresado no es el correcto en un campo numérico y no ejecuta la actualización de la carrera.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Eliminar carreras o programas académicos**

Esta prueba consiste en la eliminación de una o varias carreras o programas académicos. Cabe mencionar que el sistema solamente eliminará la información de la carrera y no los cursos que dicha carrera pueda tener asociada.

Tabla 3-33: Resultados de los Casos de Prueba: Eliminar carreras o programas académicos

Caso de Prueba	Entrada	Resultado Obtenido
Eliminación individual de carreras o programas académicos.	Se selecciona a la carrera que se desea eliminar.	El sistema muestra un mensaje de confirmación solicitando al usuario que confirme la acción a realizar y elimina definitivamente la información de la carrera.
Eliminación en grupo de carreras o programas académicos.	Se selecciona (de un checkbox) las carreras que se desean eliminar.	Con las carreras seleccionadas se escoge la opción de "eliminación de carreras seleccionadas" y el sistema muestra un mensaje de confirmación solicitando al usuario que confirme la acción a realizar y elimina definitivamente la información de las carreras seleccionadas.
Cancelación de la acción de eliminación de carreras o programas académicos.	El administrador procede con la eliminación individual o grupal y se le presenta el mensaje de confirmación.	El sistema al momento que se le presenta el mensaje de confirmación con las opciones de aceptar o cancelar y cuando se selecciona la opción de Cancelar, el sistema no ejecutó la operación de eliminación de las carreras.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Buscar carreras o programas académicos**

Esta prueba consiste en la búsqueda de una o varias carreras o programas académicos.

Tabla 3-34: Resultados de los Casos de Prueba: Buscar carreras o programas académicos

Caso de Prueba	Entrada	Resultado Obtenido
Búsqueda simple de carreras o programas académicos existentes.	Se ingresará una palabra clave que relacione al nombre de la carrera.	El sistema busca las carreras existentes que coinciden en el nombre con la palabra clave y presenta un listado de las mismas.
Búsqueda avanzada de carreras o programas académicos existentes.	Se ingresará ciertos parámetros de búsqueda (nombre, nivel de formación, modalidad de estudios, etc.).	El sistema buscar las carreras existentes que coincidan con los parámetros de búsqueda y presenta un listado de las mismas.
Búsqueda simple o avanzada de carreras o programas académicos que no existen.	Se ingresará ciertos parámetros de búsqueda (nombre, nivel de formación, modalidad de estudios, etc.).	El sistema busca las carreras que coincidan con los parámetros de búsqueda y si no coinciden emite un mensaje que "No se encontraron resultados".

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Dar de baja a los cursos que pertenecen a una carrera o programa académico**

Esta prueba consiste en desasociar a un curso de una carrera o programa académico.

Tabla 3-35: Resultados de los Casos de Prueba: Dar de baja a los cursos que pertenecen a una carrera o programa académico

Caso de Prueba	Entrada	Resultado Obtenido
Dar de baja a los cursos que pertenecen a una carrera o programa académico.	En listado de cursos de la carrera, seleccionar a los cursos que se desea dar de baja.	De la información detallada de la carrera, ser presenta un listado de cursos con la opción de dar de baja. Se selecciona a los cursos y se selecciona la opción de dar de baja a dichos cursos emite un mensaje de confirmación solicitando al usuario que confirme la acción a realizar.
Cancelación de la acción de dar de baja a los cursos de una carrera o programa académico.	El administrador procede con la baja de los cursos seleccionados y se le presenta el mensaje de confirmación.	Con el mensaje de confirmación con las opciones de aceptar o cancelar y si el Administrador selecciona la opción de Cancelar, el sistema no efectuó la operación de dar de baja a los cursos seleccionados.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

3.3.2 ITERACION 2

3.3.2.1 Diseño

En la fase de diseño se detallan las tarjetas CRC para las Historias 3, 4 y 5, según lo definido en las Historias de Usuarios y la Planificación de Entregas.

En este sentido se detallan las Tarjetas CRC de las historias correspondientes a esta iteración.

a. Clase Mención (MentionCareerManager)

Tabla 3-36: Iteración 2 - Tarjeta CRC para la Clase MentionCareerManager

Clase	**MentionCareerManager**
Descripción	Clase localizada en la capa del negocio para gestionar la información de las Menciones de las Carreras o Programas Académicos.
Atributos	- sp_career_id (integer) - sp_career_name (string) - career_id (integer)

Métodos	MentionCareerManager() read(parámetro) get() set(parámetro) save() delete()
Responsabilidades	- Inicializar el objeto (Constructor de la clase). - Obtener la información de una mención que pertenece a una carrera o programa académico. - Obtener el código de un objeto MentionCareerManager (código de la mención). - Asignar un nuevo código de mención al código de un objeto MentionCareerManager (código de mención). - Almacenar la información de una mención de carrera o programa académico (inserción o actualización). - Eliminar la información de una mención de carrera o programa académico.
Colaboradores	ADOdb

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

b. Clase Período Académico (AcademicPeriodManager)

Tabla 3-37: Iteración 2 - Tarjeta CRC para la Clase AcademicPeriodManager

Clase	**AcademicPeriodManager**
Descripción	Clase localizada en la capa del negocio para gestionar la información de los Períodos Académicos de las Carreras o Programas Académicos.
Atributos	- academic_period_id (integer) - start_date (date) - end_date (date) - status (string) - start_date_enroll (date) - end_date_enroll (date) - career_id (integer)
Métodos	AcademicPeriodManager () read(parámetro) get() set(parámetro) save() delete()

Responsabilidades	- Inicializar el objeto (Constructor de la clase). - Obtener la información de un Período Académico de una carrera o programa académico. - Obtener el código de un objeto AcademicPeriodManager (código del período académico). - Asignar un nuevo código de período académico al código de un objeto AcademicPeriodManager (código de período académico). - Almacenar la información de un período académico de una carrera o programa académico (inserción o actualización). - Eliminar la información de un período académico.
Colaboradores	ADOdb

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

c. Clase Requisitos (RequirementManager)

Tabla 3-38: Iteración 2 - Tarjeta CRC para la Clase RequirementManager

Clase	**RequirementManager**
Descripción	Clase localizada en la capa del negocio para gestionar la información de los requisitos (Pre-requisitos y Co-requisitos) de los cursos a nivel de malla curricular, que pertenecen a una carrera o programa académico.
Atributos	- course_code (integer) - depend_course_code (integer) - type_req_id (integer)
Métodos	RequirementManager() save() delete(parámetros) checkRequirement(parámetros) getRequerimentsCourse(parámetros)
Responsabilidades	- Inicializar el objeto (Constructor de la clase). - Almacenar los requisitos (Pre-requisitos y Co-requisitos) de un curso que pertenece a una carrera o programa académico (inserción o actualización). - Eliminar los requisitos de un curso (eliminar dependencias). - Consultar si un curso ya tiene relación de dependencia con otro curso dado el tipo de dependencia (Pre-requisitos y Co-requisitos). - Obtener los requisitos de un curso dado el tipo de dependencia (Pre-requisitos y Co-requisitos).
Colaboradores	ADOdb

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

3.3.2.2 Implementación

A continuación se muestra el código fuente de las clases antes diseñadas. Para mayor detalle de la implementación ver en el CD adjunto, documento: *Dokeos for University - Implementación/index.html*.

a. Clase Mención (MentionCareerManager)

Tabla 3-39: Iteración 2 – Implementación de la Clase MentionCareerManager

Clase: MentionCareerManager

```
class MentionCareerManager
{
	var  $sp_career_id;
	var  $sp_career_name;
	var  $career_id;
	/**
	* Constructor de la clase SpecialityCareerManager (Inicializar el objeto)
	*/
	function MentionCareerManager($sp_career_id = null)
	{
		global $conn;
		if($sp_career_id != null)
		{
			$this->sp_career_id = $sp_career_id;
			// begin Recordset
			$query_rsSpecManager = "SELECT * FROM speciality_career WHERE sp_career_id='$this-
>sp_career_id'";
			$rsSpecManager = $conn->SelectLimit($query_rsSpecManager) or die($conn->ErrorMsg());
			$totalRows_rsSpecManager = $rsSpecManager->RecordCount();
			// end Recordset
			if ($totalRows_rsSpecManager > 0)
			{
				$this->sp_career_name = $rsSpecManager->Fields('sp_career_name');
				$this->career_id = $rsSpecManager->Fields('career_id');
			}
			else
			{
				$this->sp_career_id = "";
				$this->sp_career_name = "";
				$this->career_id = "";
			}
		}
		else
		{
			$this->sp_career_id = "";
			$this->sp_career_name = "";
			$this->career_id = "";
		}
	}//end constructor SpecialityCareerManager

	/**
	* Obtener la información de una mención que pertenece a una carrera  o programa académico.
	*/
```

```
	function read($sp_career_id)
	{
		global $conn;
		$this->sp_career_id = $sp_career_id;
		// begin Recordset
		$query_rsSpecManager = "SELECT * FROM speciality_career WHERE sp_career_id='$this-
>sp_career_id'";
		$rsSpecManager = $conn->SelectLimit($query_rsSpecManager) or die($conn->ErrorMsg());
		$totalRows_rsSpecManager = $rsSpecManager->RecordCount();
		// end Recordset
		if ($totalRows_rsSpecManager > 0)
		{
			$this->sp_career_name = $rsSpecManager->Fields('sp_career_name');
			$this->career_id = $rsSpecManager->Fields('career_id');
			return true;
		}
		else
		{
			$this->sp_career_id = "";
			$this->sp_career_name = "";
			$this->career_id = "";
			return false;
		}
	}//end read

	/**
	* Almacenar la información de una mención de carrera o programa académico (inserción o actualización) en la
base de datos.
	*/
	function save()
	{
		global $conn;
		if ($this->sp_career_id == "" || $this->sp_career_id == 0)
		{
			$insertSQL = sprintf("INSERT INTO speciality_career (sp_career_name, career_id) VALUES
(%s, %s)",
			GetSQLValueString($this->sp_career_name, "date"),
			GetSQLValueString($this->career_id, "int"));
			$result = $conn->Execute($insertSQL) or die($conn->ErrorMsg());
			$this->sp_career_id = mysql_insert_id();
		}
		else
		{
			$updateSQL = sprintf("UPDATE speciality_career SET sp_career_name=%s, career_id=%s
WHERE sp_career_id=%s",
			GetSQLValueString($this->sp_career_name, "date"),
			GetSQLValueString($this->career_id, "int"),
			GetSQLValueString($this->sp_career_id, "int"));
			$result = $conn->Execute($updateSQL) or die($conn->ErrorMsg());
		}
		return ($conn->Affected_Rows());
	}//end save()

	/**
	* Eliminar la información de una mención de carrera o programa académico de la base de datos
	*/
```

```
	function delete()
	{
		global $conn;
		$deleteSQL = sprintf("DELETE FROM speciality_career WHERE sp_career_id=%s",
		GetSQLValueString($this->sp_career_id, "int"));
		$result = $conn->Execute($deleteSQL) or die($conn->ErrorMsg());
		return ($conn->Affected_Rows());
	} // end delete()
}
```

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

b. Clase Requisitos (RequirementManager)

Tabla 3-40: Iteración 2 – Implementación de la Clase RequirementManager

Clase: RequirementManager

```
class RequirementManager
{
	var  $course_code;
	var  $depend_course_code;
	var  $type_req_id;

	/**
	* Constructor de la clase RequirementManager (Inicializar el objeto)
	*/
	function RequirementManager($course_code = null, $depend_course_code = null)
	{
		global $conn;
		if($course_code != null && $depend_course_code != null)
		{
			$this->course_code = $course_code;
			$this->depend_course_code = $depend_course_code;
			// begin Recordset
			$query_rsRequirementManager = "SELECT * FROM dependency WHERE course_code =
'$this->course_code' AND depend_course_code = '$this->depend_course_code'";
			$rsRequirementManager = $conn->SelectLimit($query_rsRequirementManager) or
die($conn->ErrorMsg());
			$totalRows_rsRequirementManager = $rsRequirementManager->RecordCount();
			// end Recordset
			$rsRequirementManager->Fields('type_req_id');
			if ($totalRows_rsRequirementManager > 0)
			{
				$this->type_req_id = $rsRequirementManager->Fields('type_req_id');
			}
			else
			{
				$this->course_code = "";
				$this->depend_course_code = "";
				$this->type_req_id = "";
			}
		}
		else
		{
			$this->course_code = "";
			$this->depend_course_code = "";
```

```
                $this->type_req_id = "";
            }
        }//end constructor RequirementManager

        /**
        * Almacenar los requisitos (Pre-requisitos y Co-requisitos) de un curso que pertenece a una carrera o programa académico (inserción o actualización).
        */
        function save()
        {
            global $conn;
            if ($this->course_code != "" && $this->depend_course_code != "")
            {
                $insertSQL = sprintf("INSERT INTO dependency (course_code, depend_course_code, type_req_id) VALUES (%s, %s, %s)",
                GetSQLValueString($this->course_code, "text"),
                GetSQLValueString($this->depend_course_code, "text"),
                GetSQLValueString($this->type_req_id, "int"));
                $result = $conn->Execute($insertSQL) or die($conn->ErrorMsg());
            }
            else
            {
                $updateSQL = sprintf("UPDATE dependency SET type_req_id=%s WHERE course_code=%s AND depend_course_code=%s",
                GetSQLValueString($this->type_req_id, "int"),
                GetSQLValueString($this->course_code, "text"),
                GetSQLValueString($this->depend_course_code, "text"));
                $result = $conn->Execute($updateSQL) or die($conn->ErrorMsg());
            }
            return ($conn->Affected_Rows());
        }//end save()

        /**
        * Eliminar los requisitos de un curso (eliminar dependencias) en la base de datos.
        */
        function delete($course_code, $depend_course_code)
        {
            global $conn;
            $deleteSQL = sprintf("DELETE FROM dependency WHERE course_code=%s AND depend_course_code=%s",
            GetSQLValueString($course_code, "text"),
            GetSQLValueString($depend_course_code, "text"));
            $result = $conn->Execute($deleteSQL) or die($conn->ErrorMsg());
        } // end delete()

        /**
        * Consultar si un curso ya tiene relación de dependencia con otro curso dado el tipo de dependencia (Pre-requisitos y Co-requisitos).
        */
        function checkRequirement($course_code, $depend_course_code, $type_req_id)
        {
            global $conn;
            // begin Recordset
            $query_rsCheckRequirement = "SELECT * FROM dependency WHERE course_code = '$course_code' AND depend_course_code = '$depend_course_code' AND type_req_id LIKE '$type_req_id'";
            $rsCheckRequirement = $conn->SelectLimit($query_rsCheckRequirement) or die($conn->ErrorMsg());
```

```
                $totalRows_rsCheckRequirement = $rsCheckRequirement->RecordCount();
                // end Recordset

                if($totalRows_rsCheckRequirement > 0)
                return false;
                return true;
        }

        /**
        * Obtener los requisitos de un curso dado el tipo de dependencia (Pre-requisitos y Co-requisitos).
        */
        function getRequerimentsCourse($course_code, $type_req_id)
        {
                global $conn;
                $courses = array();
                if($type_req_id == 1) // Pre-requisitos
                {
                        // begin Recordset
                        $query_rsReqCoursePre = "SELECT * FROM dependency WHERE course_code =
'$course_code' AND type_req_id = '$type_req_id'";
                        $rsReqCoursePre = $conn->SelectLimit($query_rsReqCoursePre) or die($conn-
>ErrorMsg());
                        $totalRows_rsReqCoursePre = $rsReqCoursePre->RecordCount();
                        // end Recordset
                        while (!$rsReqCoursePre->EOF) {
                                $courses[] = $rsReqCoursePre->Fields('depend_course_code');
                                $rsReqCoursePre->MoveNext();
                        }
                }
                elseif($type_req_id == 2) // Co-requisitos
                {
                        // begin Recordset
                        $query_rsReqCourseCo = "SELECT * FROM dependency WHERE (course_code =
'$course_code' OR depend_course_code = '$course_code') AND type_req_id = '$type_req_id'";
                        $rsReqCourseCo = $conn->SelectLimit($query_rsReqCourseCo) or die($conn->ErrorMsg());
                        $totalRows_rsReqCourseCo = $rsReqCourseCo->RecordCount();
                        // end Recordset
                        while (!$rsReqCourseCo->EOF) {
                                if($rsReqCourseCo->Fields('course_code') != $course_code)
                                $courses[] = $rsReqCourseCo->Fields('course_code');
                                else
                                $courses[] = $rsReqCourseCo->Fields('depend_course_code');
                                $rsReqCourseCo->MoveNext();
                        }
                }
                $courses = array_unique($courses);
                return $courses;
        }
}
```

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

NOTA: La implementación de las demás clases de que intervienen en esta Iteración estan descritas en detalle en el CD adjunto, documento: *Dokeos for University - Implementación/index.html.*

3.3.2.3 Pruebas

De acuerdo a lo estipulado en el capítulo 2, en la explicación de la metodología Extreme Programming se planteará un plan de pruebas para cada una de las Historias de Usuario de la Iteración 2.

Plan de Pruebas

HISTORIA 3

- **Listado de menciones de una carrera o programa académico**

Esta prueba consiste en la presentación de una lista de menciones de una carrera o programa académico.

Tabla 3-41: Casos de Prueba: Listado de menciones de una carrera o programa académico

Caso de Prueba	Entrada	Resultado Esperado
Listar las menciones de una carrera o programa académico registrados.	Seleccionar Menciones del listado de carreras.	El sistema debe presentar un listado de las menciones que una carrera puede tener.
No existen menciones de una carrera o programa académico registrados.	Seleccionar Menciones del listado de carreras.	El sistema debe mostrar un mensaje que "No existen Menciones registradas para esta carrera".

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Creación de menciones de carreras o programas académicos**

Esta prueba consiste en la creación de una mención de carrera o programa académico.

Tabla 3-42: Casos de Prueba: Creación de menciones de carreras o programas académicos

Caso de Prueba	Entrada	Resultado Esperado
Ingresar el dato correcto en el nombre de la mención.	Se ingresa el nombre de la mención de la carrera.	El sistema debe guardar la mención de una carrera previa a la verificación del mismo. Y emitir un mensaje del éxito de la operación.
Obviar el ingreso del nombre de la mención.	Se omite el ingreso del nombre de la mención que es de ingreso obligatorio.	El sistema debe emitir un mensaje informando que se deben ingresar todos los datos para poder guardar correctamente a la mención de la carrera.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Actualizar menciones de carreras o programas académicos**

Esta prueba consiste en la actualización o modificación de una mención de una carrera o programa académico.

Tabla 3-43: Casos de Prueba: Actualizar una mención de una carrera o programa académico

Caso de Prueba	Entrada	Resultado Esperado
Actualizar el nombre de la mención de la carrera.	Ingreso del nuevo nombre de la mención.	El sistema debe buscar y leer a la mención de la carrera y debe actualizar el nombre de dicha mención. Y emitir un mensaje del éxito de la operación.
Actualización con datos nulos (vacíos) en el nombre de la mención que es de ingreso obligatorio.	Actualización del nombre de la mención que requiere información obligatoria con datos nulos.	El sistema no debe permitir la actualización y debe emitir un mensaje de alerta informando que se deben ingresar todos los datos para poder guardar correctamente a la mención de la carrera.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Eliminar menciones de carreras o programas académicos**

Esta prueba consiste en la eliminación de una o varias menciones de una carrera o programa académico.

Tabla 3-44: Casos de Prueba: Eliminar una mención de una carrera o programa académico

Caso de Prueba	Entrada	Resultado Esperado
Eliminación individual de menciones de una carrera o programa académico.	Se selecciona a la mención de la carrera que se desea eliminar.	El sistema debe emitir un mensaje de confirmación solicitando al usuario que confirme la acción a realizar. Y emitir un mensaje del éxito de la operación.
Eliminación en grupo de menciones de una carrera o programa académico.	Se selecciona (de un checkbox) las menciones de la carrera que se desean eliminar.	El sistema debe presentar la opción de eliminación de menciones seleccionadas y cuando el usuario seleccione esta opción el sistema debe emitir un mensaje de confirmación solicitando al usuario que confirme la acción a realizar. Y emitir un mensaje del éxito de la operación.
Cancelación de la acción de eliminación de menciones de una carrera o programa académico.	El administrador procede con la eliminación individual o grupal y se le presenta el mensaje de confirmación.	El sistema al momento que se le presente el mensaje de confirmación con las opciones de aceptar o cancelar y cuando se seleccione la opción de Cancelar, el sistema debe detener la acción de eliminación de las menciones.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

HISTORIA 4

- **Listado de períodos académicos de una carrera o programa académico**

Esta prueba consiste en la presentación de una lista de períodos académicos de una carrera.

Tabla 3-45: Casos de Prueba: Listado de períodos académicos de una carrera o programa académico

Caso de Prueba	Entrada	Resultado Esperado
Listar los períodos académicos de una carrera o programa académico registrados.	Seleccionar Períodos Académicos del listado de carreras.	El sistema debe presentar un listado de los períodos académicos de una carrera puede tener que incluya Fecha de Inicio y finalización del período, fecha de inicio y finalización de matrículas.
No existen períodos académicos de una carrera o programa académico registrados.	Seleccionar Períodos Académicos del listado de carreras.	El sistema debe mostrar un mensaje que "No existen períodos académicos registrados en la carrera".

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Creación de períodos académicos de carreras o programas académicos**

Esta prueba consiste en la creación de un período académico de una carrera o programa académico.

Tabla 3-46: Casos de Prueba: Creación de períodos académicos de una carreras o programas académicos

Caso de Prueba	Entrada	Resultado Esperado
Ingresar los datos correctos en los distintos campos de ingreso del período académico.	Se ingresa los datos correctos que identifican al período académico.	El sistema debe guardar el período académico de una carrera previo a la verificación del mismo. Y emitir un mensaje del éxito de la operación.
Obviar el ingreso de datos en campos obligatorios del período académico.	Se omite el ingreso de datos en los campos de ingreso que son obligatorios para el período académico.	El sistema debe emitir un mensaje informando que se deben ingresar todos los datos para poder guardar correctamente al período académico de la carrera.
Ingreso de datos que no son fechas en los campos del período académico	Se ingresa datos que no son fechas en los campos período académico.	El sistema no debe permitir el ingreso de datos que no son fechas en los campos del período académico y debe mostrar un mensaje de error que informe que existe un error en el tipo de dato ingresado (campo inválido).

Ingreso de un segundo período académico con estado Actual	Se ingresa el período académico con estado Actual	El sistema no debe permitir la creación de un segundo período académico con estado Actual. Y debe emitir el siguiente mensaje "Existe un período académico que contiene el estado Actual. Solamente puede estar un período académico con estado Actual o Activo"

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Actualizar períodos académicos de carreras o programas académicos**

Esta prueba consiste en la actualización o modificación de una mención de una carrera o programa académico.

Tabla 3-47: Casos de Prueba: Actualizar una período académico de una carrera o programa académico

Caso de Prueba	Entrada	Resultado Esperado
Actualizar el nombre de la mención de la carrera.	Ingreso del nuevo nombre de la mención.	El sistema debe buscar y leer a la mención de la carrera y debe actualizar el nombre de dicha mención. Y emitir un mensaje del éxito de la operación.
Actualización con datos nulos (vacíos) en el nombre de la mención que es de ingreso obligatorio.	Actualización de los datos del período académico que requieren información obligatoria con datos nulos.	El sistema no debe permitir la actualización y debe emitir un mensaje de alerta informando que se deben ingresar todos los datos para poder guardar correctamente al período académico de la carrera.
Actualización de un segundo período académico con estado Actual	Se actualiza el período académico con estado Actual	El sistema no debe permitir la actualización ya que existe un período académico con estado Actual. Y debe emitir el siguiente mensaje "Existe un período académico que contiene el estado Actual. Solamente puede estar un período académico con estado Actual o Activo"

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Eliminar períodos académicos de carreras o programas académicos**

Esta prueba consiste en la eliminación de uno o varios períodos académicos de una carrera o programa académico.

Tabla 3-48: Casos de Prueba: Eliminar un período académico de una carrera o programa académico

Caso de Prueba	Entrada	Resultado Esperado
Eliminación individual de períodos académicos de una carrera o programa académico.	Se selecciona el período académico de la carrera que se desea eliminar.	El sistema debe emitir un mensaje de confirmación solicitando al usuario que confirme la acción a realizar. Y emitir un mensaje del éxito de la operación.

Eliminación en grupo de períodos académicos de una carrera o programa académico.	Se selecciona (de un checkbox) los períodos académicos de la carrera que se desean eliminar.	El sistema debe presentar la opción de eliminación de los períodos académicos seleccionados y cuando el usuario seleccione esta opción el sistema debe emitir un mensaje de confirmación solicitando al usuario que confirme la acción a realizar. Y emitir un mensaje del éxito de la operación.
Cancelación de la acción de eliminación de períodos académicos de una carrera o programa académico.	El administrador procede con la eliminación individual o grupal y se le presenta el mensaje de confirmación.	El sistema al momento que se le presente el mensaje de confirmación con las opciones de aceptar o cancelar y cuando se seleccione la opción de Cancelar, el sistema debe detener la acción de eliminación de los períodos académicos.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

HISTORIA 5

- **Listado de requisitos de los cursos (pre-requisitos y co-requisitos) de una carrera o programa académico**

Esta prueba consiste en la presentación de una lista de requisitos de los cursos (pre-requisitos y co-requisitos) de una carrera.

Tabla 3-49: Casos de Prueba: Listado de requisitos de los cursos (pre-requisitos y co-requisitos) de una carrera o programa académico

Caso de Prueba	Entrada	Resultado Esperado
Listar los períodos académicos de una carrera o programa académico registrados.	Seleccionar Pre-requisitos y Co-requisitos del listado de carreras.	El sistema debe presentar un listado de los requisitos (pre-requisitos y co-requisitos) de cada curso.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Creación de requisitos (pre-requisitos y/o co-requisitos) de un curso perteneciente a una carrera o programa académico**

Esta prueba consiste en la creación de un requisito para un curso (pre-requisitos y/o co-requisitos) de una carrera o programa académico.

Tabla 3-50: Casos de Prueba: Creación de requisitos de los cursos (pre-requisitos y/o co-requisitos) de una carrera o programa académico

Caso de Prueba	Entrada	Resultado Esperado
Agregar un Pre-requisito o Co-requisito a un curso	Se selecciona del listado de cursos principales, los cursos que serán pre-	El sistema debe guardar la relación (pre-requisito o co-requisito) entre los cursos. Y emitir un mensaje del éxito de la operación. El sistema debe mostrar los códigos de los

	requisitos o co-requisitos de un curso seleccionado en la lista de cursos dependientes, y seleccionar el tipo de relación.	cursos del cual son pre-requisitos o co-requisitos de un curso determinado.
Agregar un Pre-requisito o Co-requisito entre dos cursos con una relación ya existente	Se selecciona del listado de cursos principales, los cursos que serán pre-requisitos o co-requisitos de un curso seleccionado en la lista de cursos dependientes, y seleccionar el tipo de relación. Existe un relación entre dos curso	El sistema no debe permitir el registro de una relación adicional entre dos cursos debido a la existencia de una relación (pre-requisito o co-requisito). Y debe emitir un mensaje de alerta informando al usuario que "Ya existe una relación entre los dos cursos".

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Eliminar requisitos (pre-requisitos y/o co-requisitos) de un curso perteneciente a una carrera o programa académico**

Esta prueba consiste en la eliminación de requisitos (pre-requisitos y/o co-requisitos) de un curso de una carrera o programa académico.

Tabla 3-51: Casos de Prueba: Eliminar los requisitos de los cursos (pre-requisitos y/o co-requisitos) de una carrera o programa académico

Caso de Prueba	Entrada	Resultado Esperado
Eliminación pre-requisitos y/o co-requisitos de un curso que pertenece a una carrera o programa académico.	Se selecciona (de un checkbox) los cursos que se desean eliminar los requisitos y seleccionar la opción de eliminación: (1) Pre-requisitos, (2) Co-requisitos ó (3) Pre-requisitos y Co-requisitos	El sistema debe presentar la opción de eliminación de los requisitos de los cursos seleccionados y cuando el usuario seleccione esta opción el sistema debe emitir un mensaje de confirmación solicitando al usuario que confirme la acción a realizar. Y emitir un mensaje del éxito de la operación.
Cancelación de la acción de eliminación de los requisitos de los cursos de una carrera o programa académico.	El administrador procede con la eliminación y se le presenta el mensaje de confirmación.	El sistema al momento que se le presente el mensaje de confirmación con las opciones de aceptar o cancelar y cuando se selecciona la opción de Cancelar, el sistema debe detener la acción de eliminación de los requisitos de los cursos.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

Procedimiento de Pruebas

Para cada uno de los procedimientos de prueba de esta iteración, el usuario administrador tendrá que realizar los siguientes procedimientos básicos para acceder a las opciones de "*Administración de la Plataforma*".

1. En la pantalla correspondiente de ingreso al sistema el administrador de la plataforma ingresa su *login* y *password*.
2. El usuario administrador presiona el botón *OK.*
3. El sistema hace la verificación de los datos ingresados y si tiene perfil de administrador de la plataforma.
4. El administrador seleccionará la opción "*Administración de la Plataforma*" y encontrará los módulos de administración de usuarios, cursos, carrera y plataforma.

HISTORIA 3

- **Procedimiento de prueba para el caso de prueba Listado de menciones de una carrera o programa académico**

  1 En el módulo correspondiente a la administración de Carreras seleccionar la opción de *Lista de carreras*".

  2 En el listado, el administrador pulsará el botón de *Menciones* que se presenta en cada una de las carreras registradas.

  3 Se presenta un listado de todas las menciones que una carrera puede tener.

- **Procedimiento de prueba para el caso de prueba Creación de menciones de carreras o programas académicos**

  1 En el módulo correspondiente a la administración de Carreras seleccionar la opción de *Lista de carreras*".

2 En el listado, el administrador pulsará el botón de *Menciones* que se presenta en cada una de las carreras registradas.

3 Se presenta un listado de todas las menciones que una carrera puede tener.

4 Pulsar sobre la opción *Añadir una mención.*

5 En el formulario, el administrador ingresará los datos solicitados y presiona el botón *OK*.

6 El sistema verifica que todos los casos esté completos y correctos para proceder con la creación, en caso de no cumplir con esta normativa, el sistema despliega un mensaje de error indicando que se ingresen los datos completos y correctos.

- **Procedimiento de prueba para el caso de prueba Actualizar menciones de carreras o programas académicos**

1 En el módulo correspondiente a la administración de Carreras seleccionar la opción de *Lista de carreras*".

2 En el listado, el administrador pulsará el botón de *Menciones* que se presenta en cada una de las carreras registradas.

3 Se presenta un listado de todas las menciones que una carrera puede tener.

4 En el listado, el administrador pulsará el botón de *Modificar* que se presenta en cada una de las menciones registradas.

5 Se presenta un formulario con los datos de la mención, el administrador actualiza los datos correspondientes y presiona el botón *OK*.

6 El sistema verifica que todos los casos esté completos y correctos para proceder con la actualización, en caso de no cumplir con esta normativa, el sistema despliega un mensaje de error indicando que se ingresen los datos completos y correctos.

- **Procedimiento de prueba para el caso de prueba Eliminar menciones de carreras o programas académicos**

  1 En el módulo correspondiente a la administración de Carreras seleccionar la opción de *Lista de carreras*".

  2 En el listado, el administrador pulsará el botón de *Menciones* que se presenta en cada una de las carreras registradas.

  3 Se presenta un listado de todas las menciones que una carrera puede tener.

  4 En el listado, el administrador pulsará el botón de *Eliminar* que se presenta en cada una de las menciones registradas.

  5 Se presenta un mensaje de confirmación de la acción a realizar

  6 El sistema elimina la información relacionada con la mención de la carrera.

HISTORIA 4

- **Procedimiento de prueba para el caso de prueba Listado de períodos académicos de una carrera o programa académico**

  1 En el módulo correspondiente a la administración de Carreras seleccionar la opción de *Lista de carreras*".

  2 En el listado, el administrador pulsará el botón de *Períodos Académicos* que se presenta en cada una de las carreras registradas.

  3 Se presenta un listado de todos los períodos académicos.

- **Procedimiento de prueba para el caso de prueba Creación de períodos académicos de carreras o programas académicos**

  1 En el módulo correspondiente a la administración de Carreras seleccionar la opción de *Lista de carreras*".

2 En el listado, el administrador pulsará el botón de *Períodos Académicos* que se presenta en cada una de las carreras registradas.

3 Se presenta un listado de todos los períodos académicos.

4 Pulsar sobre la opción *Añadir un período académico*.

5 En el formulario, el administrador ingresará los datos solicitados y presiona el botón *OK*.

6 El sistema verifica que todos los casos esté completos y correctos para proceder con la creación, en caso de no cumplir con esta normativa, el sistema despliega un mensaje de error indicando que se ingresen los datos completos y correctos.

- **Procedimiento de prueba para el caso de prueba Actualizar períodos académicos de carreras o programas académicos**

1 En el módulo correspondiente a la administración de Carreras seleccionar la opción de *Lista de carreras*".

2 En el listado, el administrador pulsará el botón de *Períodos Académicos* que se presenta en cada una de las carreras registradas.

3 Se presenta un listado de todos los períodos académicos.

4 En el listado, el administrador pulsará el botón de *Modificar* que se presenta en cada uno de los Períodos Académicos registrados.

5 Se presenta un formulario con los datos del período académico, el administrador actualiza los datos correspondientes y presiona el botón *OK*.

6 El sistema verifica que todos los casos esté completos y correctos para proceder con la actualización, en caso de no cumplir con esta normativa, el sistema despliega un mensaje de error indicando que se ingresen los datos completos y correctos.

- **Procedimiento de prueba para el caso de prueba Eliminar períodos académicos de carreras o programas académicos**

  1 En el módulo correspondiente a la administración de Carreras seleccionar la opción de *Lista de carreras*".

  2 En el listado, el administrador pulsará el botón de *Períodos Académicos* que se presenta en cada una de las carreras registradas.

  3 Se presenta un listado de todos los períodos académicos.

  4 En el listado, el administrador pulsará el botón de *Eliminar* que se presenta en cada uno de los Períodos Académicos registrados.

  5 Se presenta un mensaje de confirmación de la acción a realizar

  6 El sistema elimina la información relacionada con el período académico de la carrera.

HISTORIA 5

- **Procedimiento de prueba para el caso de prueba Listado de requisitos de los cursos (pre-requisitos y co-requisitos) de una carrera o programa académico**

  1 En el módulo correspondiente a la administración de Carreras seleccionar la opción de *Lista de carreras*".

  2 En el listado, el administrador pulsará el botón de *Pre-requisitos y Co-requisitos* que se presenta en cada una de las carreras registradas.

  3 Se presenta un listado de todos los requisitos de los cursos de la carrera.

- **Procedimiento de prueba para el caso de prueba Creación de requisitos (pre-requisitos y/o co-requisitos) de un curso perteneciente a una carrera o programa académico**

1 En el módulo correspondiente a la administración de Carreras seleccionar la opción de *Lista de carreras*".

2 En el listado, el administrador pulsará el botón de *Pre-requisitos y Co-requisitos* que se presenta en cada una de las carreras registradas.

3 Se presenta un listado de todos los requisitos de los cursos de la carrera.

4 De la Lista de Cursos Principales seleccionar el (los) curso(s) que serán pre-requisitos o co-requisitos del curso seleccionado de la Lista de Cursos Dependientes.

5 Escoger el tipo de relación de dependencia: pre-requisito o co-requisito y presionar el botón *OK*.

6 El sistema verifica que la consistencia de los datos entre la relación de dos cursos para proceder con la creación, en caso de no cumplir con esta normativa, el sistema despliega un mensaje de error.

- **Procedimiento de prueba para el caso de prueba Eliminar requisitos (pre-requisitos y/o co-requisitos) de un curso perteneciente a una carrera o programa académico**

1 En el módulo correspondiente a la administración de Carreras seleccionar la opción de *Lista de carreras*".

2 En el listado, el administrador pulsará el botón de *Pre-requisitos y Co-requisitos* que se presenta en cada una de las carreras registradas.

3 Se presenta un listado de todos los requisitos de los cursos de la carrera.

4 En el listado, el administrador seleccionará a los cursos que desee eliminar la relación de dependencia (pre-requisitos, co-requisitos ó pre-requisitos y co-requisitos de los cursos seleccionados) y luego pulsará el botón de *OK*.

5 En el listado, el administrador podrá seleccionar los cursos de los cuales desea eliminar la relación de dependencia: Pre-requisitos y/o Co-requisitos.

6 Se presenta un mensaje de confirmación de la acción a realizar

7 El sistema elimina la relación de dependencia entre los cursos.

Resultados Obtenidos

HISTORIA 3

- **Listado de menciones de una carrera o programa académico**

Esta prueba consiste en la presentación de una lista de menciones de una carrera o programa académico.

Tabla 3-52: Resultados de los Casos de Prueba: Listado de menciones de una carrera o programa académico

Caso de Prueba	Entrada	Resultado Obtenido
Listar las menciones de una carrera o programa académico registrados.	Seleccionar Menciones del listado de carreras.	El sistema presenta un listado de las menciones detalladas por elnombre que una carrera puede tener.
No existen menciones de una carrera o programa académico registrados.	Seleccionar Menciones del listado de carreras.	El sistema muestra un mensaje que "No existen Menciones registradas para esta carrera". Quiere decir que la carrera no requiere de menciones o que aún no las tiene registradas.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Creación de menciones de carreras o programas académicos**

Esta prueba consiste en la creación de una mención de carrera o programa académico.

Tabla 3-53: Resultados de los Casos de Prueba: Creación de menciones de carreras o programas académicos

Caso de Prueba	Entrada	Resultado Obtenido
Ingresar el dato correcto en el nombre de la mención.	Se ingresa el nombre de la mención de la carrera.	El sistema almacena la mención de una carrera previa a la verificación del mismo. Y muestra emitir un mensaje del éxito de la operación.
Obviar el ingreso del nombre de la mención.	Se omite el ingreso del nombre de la mención que es de ingreso	El sistema muestra un mensaje informando al usuario que es necesario ingresar el nombre de la mención para poder guardar

	obligatorio.	satisfactoriamente.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Actualizar menciones de carreras o programas académicos**

Esta prueba consiste en la actualización o modificación de una mención de una carrera o programa académico.

Tabla 3-54: Resultados de los Casos de Prueba: Actualizar una mención de una carrera o programa académico

Caso de Prueba	Entrada	Resultado Obtenido
Actualizar el nombre de la mención de la carrera.	Ingreso del nuevo nombre de la mención.	El sistema busca, lee y actualiza los datos de la mención. Y emite un mensaje del éxito de la operación realizada.
Actualización con datos nulos (vacíos) en el nombre de la mención que es de ingreso obligatorio.	Actualización del nombre de la mención que requiere información obligatoria con datos nulos.	El sistema cancela la actualización y muestra un mensaje de alerta informando que se deben ingresar todos los datos para poder guardar correctamente la mención.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Eliminar menciones de carreras o programas académicos**

Esta prueba consiste en la eliminación de una o varias menciones de una carrera o programa académico.

Tabla 3-55: Resultados de los Casos de Prueba: Eliminar una mención de una carrera o programa académico

Caso de Prueba	Entrada	Resultado Obtenido
Eliminación individual de menciones de una carrera o programa académico.	Se selecciona a la mención de la carrera que se desea eliminar.	El sistema emite un mensaje de confirmación solicitando al usuario que confirme la acción a realizar. Y emite un mensaje del éxito de la operación.
Eliminación en grupo de menciones de una carrera o programa académico.	Se selecciona (de un checkbox) las menciones de la carrera que se desean eliminar.	El sistema, con las menciones seleccionadas por el usuario, emite un mensaje de confirmación solicitando que confirme la acción a realizar. Y emitir un mensaje del éxito de la operación.
Cancelación de la acción de eliminación de menciones de una carrera o programa académico.	El administrador procede con la eliminación individual o grupal y se le presenta el mensaje de confirmación.	El sistema al momento que realiza el proceso de eliminación (individual o grupal) emite un mensaje de confirmación con las opciones de aceptar o cancelar. Al presionar la opción de Cancelar, el sistema detiene la acción de eliminación de las menciones.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

HISTORIA 4

- **Listado de períodos académicos de una carrera o programa académico**

Esta prueba consiste en la presentación de una lista de períodos académicos de una carrera.

Tabla 3-56: Resultados de los Casos de Prueba: Listado de períodos académicos de una carrera o programa académico

Caso de Prueba	Entrada	Resultado Obtenido
Listar los períodos académicos de una carrera o programa académico registrados.	Seleccionar Períodos Académicos del listado de carreras.	El sistema muestra un listado de los períodos académicos de la carrera con los datos requeridos.
No existen períodos académicos de una carrera o programa académico registrados.	Seleccionar Períodos Académicos del listado de carreras.	El sistema muestra un mensaje que "No existen No existen períodos académicos registrados en la carrera".

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Creación de períodos académicos de carreras o programas académicos**

Esta prueba consiste en la creación de un período académico de una carrera o programa académico.

Tabla 3-57: Resultados de los Casos de Prueba: Creación de períodos académicos de una carreras o programas académicos

Caso de Prueba	Entrada	Resultado Obtenido
Ingresar los datos correctos en los distintos campos de ingreso del período académico.	Se ingresa los datos correctos que identifican al período académico.	El sistema verifica y almacena los datos del período académico y luego emite un mensaje del éxito de la operación.
Obviar el ingreso de datos en campos obligatorios del período académico.	Se omite el ingreso de datos en los campos de ingreso que son obligatorios para el período académico.	El sistema emite un mensaje informando que se deben ingresar todos los datos para poder guardar correctamente al período académico de la carrera.
Ingreso de datos que no son fechas en los campos del período académico	Se ingresa datos que no son fechas en los campos período académico.	El sistema no permite el ingreso de datos en los campos que se requieren de fechas y muestra un mensaje de error informando que el tipo de dato ingresado es inválido.
Ingreso de un segundo	Se ingresa el período	El sistema no permite insertar un segundo

período académico con estado Actual	académico con estado Actual	período académico con estado Actual debido a la existencia de un período académico con estado Actual o Activo. Y emite un mensaje de alerta sobre la operación.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Actualizar períodos académicos de carreras o programas académicos**

Esta prueba consiste en la actualización o modificación de una mención de una carrera o programa académico.

Tabla 3-58: Resultados de los Casos de Prueba: Actualizar una período académico de una carrera o programa académico

Caso de Prueba	Entrada	Resultado Esperado
Actualizar el nombre de la mención de la carrera.	Ingreso del nuevo nombre de la mención.	El sistema buscar, leer y actualiza el nombre de la mención y luego emite un mensaje del éxito de la operación.
Actualización con datos nulos (vacíos) en el nombre de la mención que es de ingreso obligatorio.	Actualización de los datos del período académico que requieren información obligatoria con datos nulos.	El sistema no permite la actualización y muestra un mensaje de alerta informando sobre la obligatoriedad de ingresar todos los datos para proceder con el almacenamientos de del período académico.
Actualización de un segundo período académico con estado Actual	Se actualiza el período académico con estado Actual	El sistema no permite actualización de un segundo período académico con estado Actual debido a la existencia de un período académico con estado Actual o Activo. Y emite un mensaje de alerta sobre la operación.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Eliminar períodos académicos de carreras o programas académicos**

Esta prueba consiste en la eliminación de uno o varios períodos académicos de una carrera o programa académico.

Tabla 3-59: Resultados de los Casos de Prueba: Eliminar un período académico de una carrera o programa académico

Caso de Prueba	Entrada	Resultado Esperado
Eliminación individual de períodos académicos de una carrera o programa académico.	Se selecciona el período académico de la carrera que se desea eliminar.	El sistema emite un mensaje de confirmación solicitando al usuario que confirme la acción a realizar. Y emitir un mensaje del éxito de la operación.
Eliminación en grupo de períodos académicos de una carrera o programa académico.	Se selecciona (de un checkbox) los períodos académicos de la carrera que se desean eliminar.	El sistema debe presentar la opción de eliminación de los períodos académicos seleccionados y cuando el usuario seleccione esta opción el sistema debe emitir un mensaje de confirmación solicitando al usuario que confirme la acción a realizar. Y

		emitir un mensaje del éxito de la operación.
Cancelación de la acción de eliminación de períodos académicos de una carrera o programa académico.	El administrador procede con la eliminación individual o grupal y se le presenta el mensaje de confirmación.	El sistema al momento que se le presente el mensaje de confirmación con las opciones de aceptar o cancelar y cuando se selecciona la opción de Cancelar, el sistema debe detener la acción de eliminación de los períodos académicos.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

HISTORIA 5

- **Listado de requisitos de los cursos (pre-requisitos y co-requisitos) de una carrera o programa académico**

Esta prueba consiste en la presentación de una lista de requisitos de los cursos (pre-requisitos y co-requisitos) de una carrera.

Tabla 3-60: Resultados de los Casos de Prueba: Listado de requisitos de los cursos (pre-requisitos y co-requisitos) de una carrera o programa académico

Caso de Prueba	Entrada	Resultado Esperado
Listar los períodos académicos de una carrera o programa académico registrados.	Seleccionar Pre-requisitos y Co-requisitos del listado de carreras.	El sistema muestra un listado de los requisitos (pre-requisitos y co-requisitos) de cada curso. El sistema muestra los códigos de los cursos del cual son pre-requisitos o co-requisitos de un curso determinado.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Creación de requisitos (pre-requisitos y/o co-requisitos) de un curso perteneciente a una carrera o programa académico**

Esta prueba consiste en la creación de un requisito para un curso (pre-requisitos y/o co-requisitos) de una carrera o programa académico.

Tabla 3-61: Resultados de los Casos de Prueba: Creación de requisitos de los cursos (pre-requisitos y/o co-requisitos) de una carrera o programa académico

Caso de Prueba	Entrada	Resultado Esperado
Agregar un Pre-requisito o Co-requisito a un curso	Se selecciona del listado de cursos principales, los cursos que serán pre-requisitos o co-requisitos de un curso seleccionado en la lista de cursos dependientes, y seleccionar el tipo de relación.	El sistema verifica que no exista alguna relación entre dos cursos y guarda la relación (pre-requisito o co-requisito) entre los cursos y emite un mensaje del éxito de la operación.

Agregar un Pre-requisito o Co-requisito entre dos cursos con una relación ya existente	Se selecciona del listado de cursos principales, los cursos que serán pre-requisitos o co-requisitos de un curso seleccionado en la lista de cursos dependientes, y seleccionar el tipo de relación. Existe un relación entre dos curso	El sistema no permite el registro de una relación adicional entre dos cursos debido a la existencia de una relación previamente registrada. Y emite un mensaje informando al usuario que "Ya existe una relación entre dos cursos".

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Eliminar requisitos (pre-requisitos y/o co-requisitos) de un curso perteneciente a una carrera o programa académico**

Esta prueba consiste en la eliminación de requisitos (pre-requisitos y/o co-requisitos) de un curso de una carrera o programa académico.

Tabla 3-62: Resultados de los Casos de Prueba: Eliminar los requisitos de los cursos (pre-requisitos y/o co-requisitos) de una carrera o programa académico

Caso de Prueba	Entrada	Resultado Esperado
Eliminación pre-requisitos y/o co-requisitos de un curso que pertenece a una carrera o programa académico.	Se selecciona (de un checkbox) los cursos que se desean eliminar los requisitos y seleccionar la opción de eliminación: (1) Pre-requisitos, (2) Co-requisitos ó (3) Pre-requisitos y Co-requisitos	El sistema presenta la opción de eliminación de los requisitos de los cursos seleccionados y cuando el usuario seleccione cualquiera de las 3 opciones emite un mensaje de confirmación solicitando al usuario que confirme la acción a realizar. Luego emite un mensaje del éxito de la operación.
Cancelación de la acción de eliminación de los requisitos de los cursos de una carrera o programa académico.	El administrador procede con la eliminación y se le presenta el mensaje de confirmación.	El sistema al momento que se le presente el mensaje de confirmación con las opciones de aceptar o cancelar y cuando se selecciona la opción de Cancelar, el sistema detiene el proceso de eliminación.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

3.3.3 ITERACION 3

3.3.3.1 Diseño

En la fase de diseño se detallan las tarjetas CRC para las Historias 6 y 9, según lo definido en las Historias de Usuarios y la Planificación de Entregas.

En este sentido se detallan las Tarjetas CRC de las historias correspondientes a esta iteración.

a. Clase Categoría de Calificaciones (ScoreCategoryManager)

Tabla 3-63: Iteración 3 - Tarjeta CRC para la Clase ScoreCategotyManager

Clase	**ScoreCategoryManager**
Descripción	Clase localizada en la capa del negocio para gestionar la información de las Categorías de las Calificaciones para un curso.
Atributos	- score_cat_id (integer) - score_cat_name (string) - score_cat_position (integer)
Métodos	ScoreCategotyManager () getScoreCategories()
Responsabilidades	- Inicializar el objeto (Constructor de la clase). - Obtener un conjunto de categorías de calificaciones ordenadas por la posición.
Colaboradores	ADOdb

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

b. Clase Concepto de Calificaciones (ScoreConceptManager)

Tabla 3-64: Iteración 3 - Tarjeta CRC para la Clase ScoreConceptManager

Clase	**ScoreConceptManager**
Descripción	Clase localizada en la capa del negocio para gestionar la información de los Conceptos de Calificaciones registradas por el profesor para un curso en una categoría determinada. Un concepto de calificación es el nombre de una calificación.
Atributos	- score_concept_id (integer) - course_code (string) - score_cat_id (integer) - score_concept_name (string) - score_concept_comment (string) - score_concept_position (string) - score_concept_visibility (integer)
Métodos	ScoreConceptManager () read() get() set(parámetro) save()

	delete() getScoreConceptCourse(parámetros) moveScore(parámetros)
Responsabilidades	- Inicializar el objeto (Constructor de la clase). - Obtener la información de un concepto de calificación de un curso. - Obtener el código de un objeto ScoreConceptManager (código del concepto de calificación). - Asignar un nuevo código de concepto de calificación al código de un objeto ScoreConceptManager (código del concepto de calificación). - Almacenar la información de los conceptos de calificación (inserción o actualización). - Eliminar la información de conceptos de calificación registradas por el profesor. - Obtener los conceptos de calificación de un curso y período académico particular, de acuerdo a la visibilidad o no del concepto de calificación que el profesor asigne. - Cambiar el orden (posición) de los conceptos de calificación por parte del profesor.
Colaboradores	ADOdb ScoreCategoryManager

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

c. Clase Concepto de Calificaciones (ScoreManager)

Tabla 3-65: Iteración 3 - Tarjeta CRC para la Clase ScoreManager

Clase	**ScoreManager**
Descripción	Clase localizada en la capa del negocio para gestionar la información de las Calificaciones registradas por el profesor de los estudiantes de un curso en una categoría, concepto y período académico determinado.
Atributos	- score_id (integer) - course_code (string) - user_id (integer) - academic_period_id (integer) - score_concept_id (integer) - score_obtained (float) - score_date_revision (date) - score_date_insert (date) - score_date_update (date) - score_observation (string)
Métodos	ScoreManager() read() get()

	set(parámetro) save() delete() getScoreUser(parámetros)
Responsabilidades	- Inicializar el objeto (Constructor de la clase). - Obtener la calificación obtenida por un estudiante dentro de un curso dado el ID de la calificación. - Obtener el código de un objeto ScoreManager (código de calificación). - Asignar un nuevo código de calificación al código de un objeto ScoreManager (código de calificación). - Almacenar la calificación de un estudiante dentro de un curso en una categoría, concepto y período académico determinado (inserción o actualización). - Eliminar la calificación de un estudiante por parte del profesor. - Obtener la calificación obtenida por un estudiante dentro de un curso dado un período académico y el tipo de visibilidad de la calificación.
Colaboradores	ADOdb ScoreConceptManager

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

d. Clase Currículum Académico (AcademicRecordManager)

Tabla 3-66: Iteración 3 - Tarjeta CRC para la Clase AcademicRecordManager

Clase	**AcademicRecordManager**
Descripción	Clase localizada en la capa del negocio para gestionar la información del currículum académico del usuario de una Carrera o Programa Académico.
Atributos	- course_code (string) - user_id (integer) - academic_period_id (integer) - status (integer) - role (string) - group_id (integer) - tutor_id (integer) - sort (integer) - user_course_cat (integer) - approval (char) - score (float) - number_enroll (integer) - category_name (string)
Métodos	AcademicRecordManager() read(parámetro)

	save() delete() getStudiedCourses(parámetros) getFailedCourses(parámetros) getApprovedCourses(parámetros) isNewStudent(parámetros) getNumEnroll(parámetros)
Responsabilidades	- Inicializar el objeto (Constructor de la clase). - Obtener la información académica de un curso tomado por un usuario en una carrera o programa académico. - Almacenar la información académica de un curso tomado por un usuario en una carrera o programa académico (inserción o actualización). - Eliminar la información académica de un curso tomado por un usuario en una carrera o programa académico. - Obtener los cursos tomados por un usuario en una carrera o programa académico. - Obtener los cursos fallidos por un usuario en una carrera o programa académico. - Obtener los cursos aprobados por un usuario en una carrera o programa académico. - Consultar si un usuario es nuevo estudiante en una carrera o programa académico. - Obtener el número de matrículas de un usuario en un curso determinado.
Colaboradores	ADOdb CareerManager CourseManager UserManager

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

3.3.3.2 Implementación

A continuación se muestra el código fuente de las clases antes diseñadas. Para mayor detalle de la implementación ver en el CD adjunto, documento: *Dokeos for University - Implementación/index.html*.

a. Clase Concepto de Calificaciones (ScoreConceptManager)

Tabla 3-67: Iteración 3 – Implementación de la Clase ScoreConceptManager

Clase: ScoreConceptManager
class ScoreConceptManager {

```
	var  $score_concept_id;
	var  $course_code;
	var  $score_cat_id;
	var  $score_concept_name;
	var  $score_concept_comment;
	var  $score_concept_position;
	var  $score_concept_visibility;
	/**
	* Constructor de la clase ScoreConceptManager (Inicializar el objeto)
	*/
	function ScoreConceptManager($score_concept_id = null)
	{
		global $conn;
		if($score_concept_id != null)
		{
			$this->score_concept_id = $score_concept_id;

			// begin Recordset
			$query_rsScoreConceptManager = "SELECT * FROM score_concept  WHERE
score_concept_id='$this->score_concept_id'";
			$rsScoreConceptManager = $conn->SelectLimit($query_rsScoreConceptManager) or
die($conn->ErrorMsg());
			$totalRows_rsScoreConceptManager = $rsScoreConceptManager->RecordCount();
			// end Recordset

			if ($totalRows_rsScoreConceptManager > 0)
			{
				$this->course_code = $rsScoreConceptManager->Fields('course_code');
				$this->score_cat_id = $rsScoreConceptManager->Fields('score_cat_id');
				$this->score_concept_name = $rsScoreConceptManager-
>Fields('score_concept_name');
				$this->score_concept_comment = $rsScoreConceptManager-
>Fields('score_concept_comment');
				$this->score_concept_position = $rsScoreConceptManager-
>Fields('score_concept_position');
				$this->score_concept_visibility = $rsScoreConceptManager-
>Fields('score_concept_visibility');
			}
			else
			{
				$this->score_concept_id = "";
				$this->course_code = "";
				$this->score_cat_id = "";
				$this->score_concept_name = "";
				$this->score_concept_comment = "";
				$this->score_concept_position = "";
				$this->score_concept_visibility = "";
			}
		}
		else
		{
			$this->score_concept_id = "";
			$this->course_code = "";
			$this->score_cat_id = "";
			$this->score_concept_name = "";
			$this->score_concept_comment = "";
```

```
                $this->score_concept_position = "";
                $this->score_concept_visibility = "";
            }
        }//end constructor ScoreConceptManager

        /**
        * Obtener la información de un concepto de calificación de un curso.
        */
        function read($score_concept_id)
        {
            global $conn;
            if($score_concept_id != null)
            {
                $this->score_concept_id = $score_concept_id;

                // begin Recordset
                $query_rsScoreConceptManager = "SELECT * FROM score_concept  WHERE
score_concept_id='$this->score_concept_id'";
                $rsScoreConceptManager = $conn->SelectLimit($query_rsScoreConceptManager) or
die($conn->ErrorMsg());
                $totalRows_rsScoreConceptManager = $rsScoreConceptManager->RecordCount();
                // end Recordset

                if ($totalRows_rsScoreConceptManager > 0)
                {
                    $this->course_code = $rsScoreConceptManager->Fields('course_code');
                    $this->score_cat_id = $rsScoreConceptManager->Fields('score_cat_id');
                    $this->score_concept_name = $rsScoreConceptManager-
>Fields('score_concept_name');
                    $this->score_concept_comment = $rsScoreConceptManager-
>Fields('score_concept_comment');
                    $this->score_concept_position = $rsScoreConceptManager-
>Fields('score_concept_position');
                    $this->score_concept_visibility = $rsScoreConceptManager-
>Fields('score_concept_visibility');
                    return true;
                }
                else
                {
                    $this->score_concept_id = "";
                    $this->course_code = "";
                    $this->score_cat_id = "";
                    $this->score_concept_name = "";
                    $this->score_concept_comment = "";
                    $this->score_concept_position = "";
                    $this->score_concept_visibility = "";
                    return false;
                }
            }
            else
            {
                $this->score_concept_id = "";
                $this->course_code = "";
                $this->score_cat_id = "";
                $this->score_concept_name = "";
                $this->score_concept_comment = "";
```

```
                $this->score_concept_position = "";
                $this->score_concept_visibility = "";
                return false;
        }
    }//end read

    /**
    * Almacenar la información de los conceptos de calificación (inserción o actualización).
    */
    function save()
    {
        global $conn;
        if ($this->score_concept_id == "" || $this->score_concept_id == 0)
        {
            $insertSQL = sprintf("INSERT INTO score_concept (course_code, score_cat_id,
score_concept_name, score_concept_comment, score_concept_position, score_concept_visibility) VALUES (%s, %s, %s,
%s, %s, %s)",
                GetSQLValueString($this->course_code, "text"),
                GetSQLValueString($this->score_cat_id, "int"),
                GetSQLValueString($this->score_concept_name, "text"),
                GetSQLValueString($this->score_concept_comment, "text"),
                GetSQLValueString($this->score_concept_position, "int"),
                GetSQLValueString($this->score_concept_visibility, "int"));
            $result = $conn->Execute($insertSQL) or die($conn->ErrorMsg());
            $this->score_concept_id = mysql_insert_id();
            return ($conn->Affected_Rows());
        }
        else
        {
            $updateSQL = sprintf("UPDATE score_concept SET course_code=%s, score_cat_id=%s,
score_concept_name=%s, score_concept_comment=%s, score_concept_position=%s, score_concept_visibility=%s
WHERE score_concept_id=%s",
                GetSQLValueString($this->course_code, "text"),
                GetSQLValueString($this->score_cat_id, "int"),
                GetSQLValueString($this->score_concept_name, "text"),
                GetSQLValueString($this->score_concept_comment, "text"),
                GetSQLValueString($this->score_concept_position, "text"),
                GetSQLValueString($this->score_concept_visibility, "int"),
                GetSQLValueString($this->score_concept_id, "int"));
            $result = $conn->Execute($updateSQL) or die($conn->ErrorMsg());
            return ($conn->Affected_Rows());
        }

    }//end save()

    /**
    * Eliminar la información de conceptos de calificación registradas por el profesor.
    */
    function delete($score_concept_id)
    {
        global $conn;
        $deleteSQL = sprintf("DELETE FROM score_concept WHERE score_concept_id=%s",
        GetSQLValueString($this->score_concept_id, "int"));
        $result = $conn->Execute($deleteSQL) or die($conn->ErrorMsg());
        // Delete user's score
        $deleteSQL = sprintf("DELETE FROM score WHERE score_concept_id=%s",
```

```
                GetSQLValueString($this->score_concept_id, "int"));
                $result = $conn->Execute($deleteSQL) or die($conn->ErrorMsg());
                return ($conn->Affected_Rows());
        } // end delete()

        /**
        * Obtener los conceptos de calificación de un curso y período académico particular, de acuerdo a la visibilidad o
no del concepto de calificación que el profesor asigne.
        */
        function getScoreConceptCourse($code, $academic_period_id, $visibility = "%")
        {
                global $conn;
                $score_concept = array();
                // begin Recordset
                $query_rsGetSCC = "SELECT s.* FROM score s, score_concept sc WHERE s.score_concept_id =
sc.score_concept_id AND s.course_code = '$code' AND academic_period_id = '$academic_period_id' AND
score_concept_visibility LIKE '$visibility' GROUP BY score_concept_id ORDER BY score_concept_position";
                $rsGetSCC = $conn->SelectLimit($query_rsGetSCC) or die($conn->ErrorMsg());
                $totalRows_rsGetSCC = $rsGetSCC->RecordCount();
                // end Recordset
                while (!$rsGetSCC->EOF) {
                        $score_concept[$rsGetSCC->Fields('score_concept_id')] = new
ScoreConceptManager($rsGetSCC->Fields('score_concept_id'));
                        $rsGetSCC->MoveNext();
                }
                return $score_concept;
        }

        /**
        * Cambiar el orden (posición) de los conceptos de calificación por parte del profesor.
        */
        function moveScore($code, $academic_period_id, $action)
        {
                global $conn;
                $this_score_concept_id = $this->score_concept_id;
                $this_score_concept_position = $this->score_concept_position;
                $this_score_concept_found = false;
                // begin Recordset
                $query_rsMoveScoreConceptCourse = "SELECT * FROM score s, score_concept sc WHERE
s.score_concept_id = sc.score_concept_id AND s.course_code = '$code' AND academic_period_id =
'$academic_period_id' GROUP BY s.score_concept_id ORDER BY score_concept_position $action";
                $rsMoveScoreConceptCourse = $conn->SelectLimit($query_rsMoveScoreConceptCourse) or
die($conn->ErrorMsg());
                $totalRows_rsMoveScoreConceptCourse = $rsMoveScoreConceptCourse->RecordCount();
                // end Recordset
                while (!$rsMoveScoreConceptCourse->EOF)
                {
                        if ($rsMoveScoreConceptCourse->Fields('score_concept_id') == $this_score_concept_id)
                        {
                                $this_score_concept_found = true;
                        }
                        $rsMoveScoreConceptCourse->MoveNext();
                        if (isset($this_score_concept_found) && $this_score_concept_found)
                        {
                                $next_score_concept_obj = new ScoreConceptManager();
                                if($next_score_concept_obj->read($rsMoveScoreConceptCourse-
```

```
>Fields('score_concept_id')))
                                        {
                                                $next_score_concept_position = $next_score_concept_obj-
>score_concept_position;
                                                $this->score_concept_position = $next_score_concept_position;
                                                $next_score_concept_obj->score_concept_position =
$this_score_concept_position;
                                                $this->save();
                                                $next_score_concept_obj->save();
                                                return true;
                                        }
                                        break;
                                }
                        }
                        return false;
                }

}
```

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

b. Clase Concepto de Calificaciones (ScoreManager)

Tabla 3-68: Iteración 3 – Implementación de la Clase ScoreManager

Clase: ScoreManager

```
class ScoreManager
{
        var  $score_id;
        var  $course_code;
        var  $user_id;
        var  $academic_period_id;
        var  $score_concept_id;
        var  $score_obtained;
        var  $score_date_revision;
        var  $score_date_insert;
        var  $score_date_update;
        var  $score_observation;

        /**
        * Constructor de la clase ScoreManager (Inicializar el objeto)
        */
        function ScoreManager($score_id = null)
        {
                global $conn;
                if($score_id != null)
                {
                        $this->score_id = $score_id;
                        // begin Recordset
                        $query_rsScoreManager = "SELECT * FROM score WHERE score_id='$this->score_id'";
                        $rsScoreManager = $conn->SelectLimit($query_rsScoreManager) or die($conn-
>ErrorMsg());
                        $totalRows_rsScoreManager = $rsScoreManager->RecordCount();
                        // end Recordset

                        if ($totalRows_rsScoreManager > 0)
```

```
                {
                        $this->course_code = $rsScoreManager->Fields('course_code');
                        $this->user_id = $rsScoreManager->Fields('user_id');
                        $this->academic_period_id = $rsScoreManager->Fields('academic_period_id');
                        $this->score_concept_id = $rsScoreManager->Fields('score_concept_id');
                        $this->score_obtained = $rsScoreManager->Fields('score_obtained');
                        $this->score_date_revision = $rsScoreManager->Fields('score_date_revision');
                        $this->score_date_insert = $rsScoreManager->Fields('score_date_insert');
                        $this->score_date_update = $rsScoreManager->Fields('score_date_update');
                        $this->score_observation = $rsScoreManager->Fields('score_observation');
                }
                else
                {
                        $this->score_id = "";
                        $this->course_code = "";
                        $this->user_id = "";
                        $this->academic_period_id = "";
                        $this->score_concept_id = "";
                        $this->score_obtained = "";
                        $this->score_date_revision = "";
                        $this->score_date_insert = "";
                        $this->score_date_update = "";
                        $this->score_observation = "";
                }
        }
        else
        {
                $this->score_id = "";
                $this->course_code = "";
                $this->user_id = "";
                $this->academic_period_id = "";
                $this->score_concept_id = "";
                $this->score_obtained = "";
                $this->score_date_revision = "";
                $this->score_date_insert = "";
                $this->score_date_update = "";
                $this->score_observation = "";
        }
}//end constructor ScoreManager

/**
* Obtener la calificación obtenida por un estudiante dentro de un curso dado el ID de la calificación.
*/
function read($score_id)
{
        global $conn;
        $this->score_id = $score_id;
        // begin Recordset
        $query_rsScoreRead = "SELECT * FROM score WHERE score_id = '$this->score_id'";
        $rsScoreRead = $conn->SelectLimit($query_rsScoreRead) or die($conn->ErrorMsg());
        $totalRows_rsScoreRead = $rsScoreRead->RecordCount();
        // end Recordset

        if ($totalRows_rsScoreRead > 0)
        {
                $this->score_id = $rsScoreRead->Fields('score_id');
```

```
                $this->course_code = $rsScoreRead->Fields('course_code');
                $this->user_id = $rsScoreRead->Fields('user_id');
                $this->academic_period_id = $rsScoreRead->Fields('academic_period_id');
                $this->score_concept_id = $rsScoreRead->Fields('score_concept_id');
                $this->score_obtained = $rsScoreRead->Fields('score_obtained');
                $this->score_date_revision = $rsScoreRead->Fields('score_date_revision');
                $this->score_date_insert = $rsScoreRead->Fields('score_date_insert');
                $this->score_date_update = $rsScoreRead->Fields('score_date_update');
                $this->score_observation = $rsScoreRead->Fields('score_observation');
                return true;
        }
        else
        {
                $this->score_id = "";
                $this->course_code = "";
                $this->user_id = "";
                $this->academic_period_id = "";
                $this->score_concept_id = "";
                $this->score_obtained = "";
                $this->score_date_revision = "";
                $this->score_date_insert = "";
                $this->score_date_update = "";
                $this->score_observation = "";
                return false;
        }
    }//end read

    /**
    * Almacenar la calificación de un estudiante dentro de un curso en una categoría, concepto y período académico
determinado (inserción o actualización).
    */
    function save()
    {
        global $conn;
        $now = date("Y-m-d H:i:s");
        if ($this->score_id == "" || $this->score_id == 0)
        {
                $insertSQL = sprintf("INSERT INTO score (course_code, user_id, academic_period_id,
score_concept_id, score_obtained, score_date_revision, score_date_insert, score_date_update, score_observation)
VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)",
                GetSQLValueString($this->course_code, "text"),
                GetSQLValueString($this->user_id, "int"),
                GetSQLValueString($this->academic_period_id, "int"),
                GetSQLValueString($this->score_concept_id, "int"),
                GetSQLValueString($this->score_obtained, "double"),
                GetSQLValueString($this->score_date_revision, "date"),
                GetSQLValueString($now, "date"),
                GetSQLValueString($now, "date"),
                GetSQLValueString($this->score_observation, "text"));
                $result = $conn->Execute($insertSQL) or die($conn->ErrorMsg());
                $this->score_id = mysql_insert_id();
                return ($conn->Affected_Rows());
        }
        else
        {
                $updateSQL = sprintf("UPDATE score SET course_code=%s, user_id=%s,
```

```
academic_period_id=%s, score_concept_id=%s, score_obtained=%s, score_date_revision=%s, score_date_insert=%s,
score_date_update=%s, score_observation=%s WHERE score_id=%s",
                                GetSQLValueString($this->course_code, "text"),
                                GetSQLValueString($this->user_id, "int"),
                                GetSQLValueString($this->academic_period_id, "int"),
                                GetSQLValueString($this->score_concept_id, "int"),
                                GetSQLValueString($this->score_obtained, "double"),
                                GetSQLValueString($this->score_date_revision, "date"),
                                GetSQLValueString($this->score_date_insert, "date"),
                                GetSQLValueString($now, "date"),
                                GetSQLValueString($this->score_observation, "text"),
                                GetSQLValueString($this->score_id, "int"));
                                $result = $conn->Execute($updateSQL) or die($conn->ErrorMsg());
                                return ($conn->Affected_Rows());
                        }
        }//end save()

        /**
        * Eliminar la calificación de un estudiante por parte del profesor.
        */
        function delete()
        {
                $deleteSQL = sprintf("DELETE FROM score WHERE score_id=%s",
                GetSQLValueString($this->score_id, "int"));
                $result = $conn->Execute($deleteSQL) or die($conn->ErrorMsg());
                return ($conn->Affected_Rows());
        } // end delete()

        /**
        * Obtener la calificación obtenida por un estudiante dentro de un curso dado un período académico y el tipo de
visibilidad de la calificación.
        */
        function getScoreUser($code, $user_id, $academic_period_id, $visibility = "%")
        {
                global $conn;
                $score = array();
                // begin Recordset
                $query_rsGetScoreUser = "SELECT s.* FROM score s, score_concept sc WHERE s.score_concept_id
= sc.score_concept_id AND s.course_code = '$code' AND user_id = '$user_id' AND academic_period_id =
'$academic_period_id' AND score_concept_visibility LIKE '$visibility' ORDER BY score_concept_position";
                $rsGetScoreUser = $conn->SelectLimit($query_rsGetScoreUser) or die($conn->ErrorMsg());
                $totalRows_rsGetScoreUser = $rsGetScoreUser->RecordCount();
                // end Recordset
                while (!$rsGetScoreUser->EOF) {
                        $score[$rsGetScoreUser->Fields('score_id')] = new ScoreManager($rsGetScoreUser-
>Fields('score_id'));
                        $rsGetScoreUser->MoveNext();
                }
                return $score;
        }

}
```

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

c. Clase Currículum Académico (AcademicRecordManager)

Tabla 3-69: Iteración 3 - Implementación la Clase AcademicRecordManager

Clase: AcademicRecordManager

```
class AcademicRecordManager
{
	var  $course_code;
	var  $user_id;
	var  $academic_period_id;
	var  $course;
	var  $user;
	var  $academic_period;
	var  $status;
	var  $role;
	var  $group_id;
	var  $tutor_id;
	var  $sort;
	var  $user_course_cat;
	var  $approval;
	var  $score;
	var  $number_enroll;
	var  $category_name;
	/**
	* Constructor de la clase AcademicRecordManager (Inicializar el objeto)
	*/
	function AcademicRecordManager()
	{
		$this->course_code = "";
		$this->user_id = "";
		$this->academic_period_id = "";
		$this->course = null;
		$this->user = null;
		$this->academic_period = null;
		$this->status = "";
		$this->role = "";
		$this->group_id = "";
		$this->tutor_id = "";
		$this->sort = "";
		$this->user_course_cat = "";
		$this->approval = "";
		$this->score = "";
		$this->number_enroll = "";
		$this->category_name = "";
	}//end constructor AcademicRecordManager

	/**
	* Obtener la información académica de un curso tomado por un usuario en una carrera  o programa académico.
	*/
	function read($course_code, $user_id, $academic_period_id)
	{
		global $conn;
		$this->course_code = $course_code;
		$this->user_id = $user_id;
		$this->academic_period_id = $academic_period_id;
		// begin Recordset
		$query_rsReadAcademicRecord = "SELECT cu.*,cc.name as category_name FROM course_rel_user
```

```
cu, course c, course_category cc WHERE cu.course_code = c.code AND c.category_code = cc.code AND cu.course_code
= '$this->course_code' AND user_id = '$this->user_id' AND academic_period_id = '$this->academic_period_id'";
		$rsReadAcademicRecord = $conn->SelectLimit($query_rsReadAcademicRecord) or die($conn-
>ErrorMsg());
		$totalRows_rsReadAcademicRecord = $rsReadAcademicRecord->RecordCount();
		// end Recordset
		if ($totalRows_rsReadAcademicRecord > 0)
		{
			$this->course_code = $rsReadAcademicRecord->Fields('course_code');
			$this->user_id = $rsReadAcademicRecord->Fields('user_id');
			$this->academic_period_id = $rsReadAcademicRecord->Fields('academic_period_id');
			$this->status = $rsReadAcademicRecord->Fields('status');
			$this->role = $rsReadAcademicRecord->Fields('role');
			$this->group_id =$rsReadAcademicRecord->Fields('group_id');
			$this->tutor_id = $rsReadAcademicRecord->Fields('tutor_id');
			$this->sort = $rsReadAcademicRecord->Fields('sort');
			$this->user_course_cat = $rsReadAcademicRecord->Fields('user_course_cat');
			$this->approval = $rsReadAcademicRecord->Fields('approval');
			$this->score = $rsReadAcademicRecord->Fields('score');
			$this->number_enroll = $rsReadAcademicRecord->Fields('number_enroll');
			$this->category_name = $rsReadAcademicRecord->Fields('category_name');
			return true;
		}
		else
		{
			$this->course_code = "";
			$this->user_id = "";
			$this->academic_period_id = "";
			$this->status = "";
			$this->role = "";
			$this->group_id = "";
			$this->tutor_id = "";
			$this->sort = "";
			$this->user_course_cat = "";
			$this->approval = "";
			$this->score = "";
			$this->number_enroll = "";
			$this->category_name = "";
			return false;
		}
	}//end read

	/**
	* Almacenar la información académica de un curso tomado por un usuario en una carrera  o programa
académico (inserción o actualización).
	*/
	function save($insert = true)
	{
		global $conn;
		if ($insert)
		{
			$insertSQL = sprintf("INSERT INTO course_rel_user (course_code, user_id,
academic_period_id, status, `role`, group_id, tutor_id, sort, user_course_cat, approval, score, number_enroll) VALUES
(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)",
			GetSQLValueString($this->course_code, "text"),
			GetSQLValueString($this->user_id, "int"),
```

```
                                GetSQLValueString($this->academic_period_id, "int"),
                                GetSQLValueString($this->status, "int"),
                                GetSQLValueString($this->role, "text"),
                                GetSQLValueString($this->group_id, "int"),
                                GetSQLValueString($this->tutor_id, "int"),
                                GetSQLValueString($this->sort, "int"),
                                GetSQLValueString($this->user_course_cat, "int"),
                                GetSQLValueString($this->approval, "text"),
                                GetSQLValueString($this->score, "double"),
                                GetSQLValueString($this->number_enroll, "int"));
                                $result = $conn->Execute($insertSQL) or die($conn->ErrorMsg());
                        }
                        else
                        {
                                $updateSQL = sprintf("UPDATE course_rel_user SET status=%s, role=%s, group_id=%s,
tutor_id=%s, sort=%s, user_course_cat=%s, approval=%s, score=%s, number_enroll=%s WHERE course_code=%s AND
user_id=%s AND academic_period_id=%s",
                                GetSQLValueString($this->status, "int"),
                                GetSQLValueString($this->role, "text"),
                                GetSQLValueString($this->group_id, "int"),
                                GetSQLValueString($this->tutor_id, "int"),
                                GetSQLValueString($this->sort, "int"),
                                GetSQLValueString($this->user_course_cat, "int"),
                                GetSQLValueString($this->approval, "text"),
                                GetSQLValueString($this->score, "double"),
                                GetSQLValueString($this->number_enroll, "int"),
                                GetSQLValueString($this->course_code, "text"),
                                GetSQLValueString($this->user_id, "int"),
                                GetSQLValueString($this->academic_period_id, "int"));
                                $result = $conn->Execute($updateSQL) or die($conn->ErrorMsg());
                        }
                        return ($conn->Affected_Rows());
                }//end save()

                /**
                * Eliminar la información académica de un curso tomado por un usuario en una carrera  o programa académico.
                */
                function delete()
                {
                        global $conn;
                        $deleteSQL = sprintf("DELETE FROM course_rel_user WHERE course_code LIKE %s AND user_id
LIKE %s AND academic_period_id LIKE %s",
                        GetSQLValueString($this->course_code, "text"),
                        GetSQLValueString($this->user_id, "int"),
                        GetSQLValueString($this->academic_period_id, "int"));
                        $result = $conn->Execute($deleteSQL) or die($conn->ErrorMsg());
                        return ($conn->Affected_Rows());
                } // end delete()

                /**
                * Obtener los cursos tomados por un usuario en una carrera o programa académico.
                */
                function getStudiedCourses($career_id,$user_id)
                {
                        global $conn;
                        $courses = array();
```

```
                        // begin Recordset
                        $query_rsStudiedCourse = "SELECT cu.*, cc.name as category_name FROM course_rel_user cu,
course c, course_category cc WHERE cu.course_code = c.code AND c.category_code = cc.code AND c.career_id =
'$career_id' AND user_id = '$user_id' AND cu.approval != '-' ORDER BY academic_period_id, cu.course_code";
                        $rsStudiedCourse = $conn->SelectLimit($query_rsStudiedCourse) or die($conn->ErrorMsg());
                        $totalRows_rsStudiedCourse = $rsStudiedCourse->RecordCount();
                        // end Recordset
                        while (!$rsStudiedCourse->EOF) {
                                $ar = new AcademicRecordManager();
                                $ar->read($rsStudiedCourse->Fields('course_code'),$rsStudiedCourse-
>Fields('user_id'),$rsStudiedCourse->Fields('academic_period_id'));
                                $courses[] = get_object_vars($ar);
                                $rsStudiedCourse->MoveNext();
                        }
                        return $courses;
                }

                /**
                * Obtener los cursos fallidos por un usuario en una carrera o programa académico.
                */
                function getFailedCourses($career_id,$user_id)
                {
                        global $conn;
                        $code_courses = array();
                        /*$query_rsFailureCourse = "SELECT course.code FROM ((course_rel_user LEFT JOIN
academic_period ON academic_period.academic_period_id=course_rel_user.academic_period_id) LEFT JOIN course ON
course.code=course_rel_user.course_code) WHERE course_rel_user.user_id='$user_id' AND
course.career_id='$career_id' AND (course_rel_user.approval='F' || course_rel_user.approval='R' ||
course_rel_user.approval='N') AND academic_period.status = 'Histórico'";*/
                        // begin Recordset
                        $query_rsFailureCourse = "SELECT `c`.`code` FROM `course` AS `c` Inner Join `course_rel_user` AS
`cu` ON `c`.`code` = `cu`.`course_code` Inner Join `academic_period` AS `ap` ON `ap`.`academic_period_id` =
`cu`.`academic_period_id` WHERE `cu`.`approval` IN ('F', 'R', 'N') AND `cu`.`user_id` = '$user_id' AND `c`.`career_id` =
'$career_id' AND `ap`.`status` = 'Histórico'";
                        $rsFailureCourse = $conn->SelectLimit($query_rsFailureCourse) or die($conn->ErrorMsg());
                        $totalRows_rsFailureCourse = $rsFailureCourse->RecordCount();
                        // end Recordset

                        while (!$rsFailureCourse->EOF) {
                                $code_courses[] = $rsFailureCourse->Fields('code');
                                $rsFailureCourse->MoveNext();
                        }
                        return $code_courses;
                }

                /**
                * Obtener los cursos aprobados por un usuario en una carrera o programa académico.
                */
                function getApprovedCourses($career_id,$user_id)
                {
                        global $conn;
                        $code_courses = array();
                        /*$query_rsApprovedCourse = "SELECT course.code FROM ((course_rel_user LEFT JOIN
academic_period ON academic_period.academic_period_id=course_rel_user.academic_period_id) LEFT JOIN course ON
course.code=course_rel_user.course_code) WHERE course_rel_user.user_id='$user_id' AND
course.career_id='$career_id' AND (course_rel_user.approval='E' || course_rel_user.approval='A' ||
```

```
course_rel_user.approval='C') AND academic_period.status = 'Histórico'";*/
                        // begin Recordset
                        $query_rsApprovedCourse = "SELECT `c`.`code` FROM `course` AS `c` Inner Join `course_rel_user`
AS `cu` ON `c`.`code` = `cu`.`course_code` Inner Join `academic_period` AS `ap` ON `ap`.`academic_period_id` =
`cu`.`academic_period_id` WHERE `cu`.`approval` IN ('A', 'E', 'C') AND `cu`.`user_id` = '$user_id' AND `c`.`career_id` =
'$career_id' AND `ap`.`status` = 'Histórico'";
                        $rsApprovedCourse = $conn->SelectLimit($query_rsApprovedCourse) or die($conn->ErrorMsg());
                        $totalRows_rsApprovedCourse = $rsApprovedCourse->RecordCount();
                        // end Recordset

                        while (!$rsApprovedCourse->EOF) {
                                $code_courses[] = $rsApprovedCourse->Fields('code');
                                $rsApprovedCourse->MoveNext();
                        }
                        return $code_courses;
                }

                /**
                * Consultar si un usuario es nuevo estudiante en una carrera o programa académico.
                */
                function isNewStudent($career_id,$user_id)
                {
                        global $conn;
                        // begin Recordset
                        $query_rsIsNewStudent = "SELECT COUNT(*) number_courses FROM user u, course c,
course_rel_user cu WHERE u.user_id = cu.user_id AND c.code = cu.course_code AND c.career_id = '$career_id' AND
u.user_id = '$user_id'";
                        $rsIsNewStudent = $conn->SelectLimit($query_rsIsNewStudent) or die($conn->ErrorMsg());
                        $totalRows_rsIsNewStudent = $rsIsNewStudent->RecordCount();
                        // end Recordset
                        if($rsIsNewStudent->Fields('number_courses') > 0)               return false;
                        else                                    return true;
                }

                /**
                * Obtener  el número de matrículas de un usuario en un curso determinado.
                */
                function getNumEnroll($course_code, $user_id)
                {
                        global $conn;
                        // begin Recordset
                        $query_rsNumEnroll = "SELECT * FROM course_rel_user WHERE course_code = '$course_code'
AND user_id = '$user_id' AND approval IN ('F','R','N')";
                        $rsNumEnroll = $conn->SelectLimit($query_rsNumEnroll) or die($conn->ErrorMsg());
                        $totalRows_rsNumEnroll = $rsNumEnroll->RecordCount();
                        // end Recordset
                        return $totalRows_rsNumEnroll;
                }//end getNumEnroll
}
```

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

NOTA: La implementación de las demás clases de que intervienen en esta Iteración estan descritas en detalle en el CD adjunto, documento: *Dokeos for University - Implementación/index.html*

3.3.3.3 Pruebas

De acuerdo a lo estipulado en el capítulo 2, en la explicación de la metodología Extreme Programming se planteará un plan de pruebas para cada una de las Historias de Usuario de la Iteración 3.

Plan de Pruebas

HISTORIA 6

- **Ingresar calificaciones de los estudiantes de un curso**

Esta prueba consiste en la adición de calificaciones de los estudiantes inscritos en un curso.

Tabla 3-70: Casos de Prueba: Añadir calificaciones de los estudiantes de un curso

Caso de Prueba	Entrada	Resultado Esperado
Ingresar calificaciones parciales de los estudiantes de un curso.	Se ingresa los datos correspondientes al concepto de calificación (categoría, nombre, etc.). Se ingresa la calificación obtenida por cada uno de los estudiantes.	El sistema debe guardar las calificaciones parciales obtenidas por los estudiantes de acuerdo a un concepto de calificación ingresado por el profesor. Y emitir un mensaje del éxito de la operación.
Ingresar calificaciones finales de los estudiantes de un curso.	Se ingresa la calificación obtenida y el tipo de aprobación (exonerado, aprobado, fallido, etc.) por cada uno de los estudiantes.	El sistema debe guardar las calificaciones finales (calificaciones de aprobación del curso) obtenidas por los estudiantes. Y emitir un mensaje del éxito de la operación.
Obviar el ingreso de las calificaciones de los estudiantes de un curso.	Se omite el ingreso de alguna o todas las calificaciones de los estudiantes que es de ingreso obligatorio.	El sistema debe emitir un mensaje informando que se deben ingresar todos los datos para poder guardar correctamente las calificaciones de los estudiantes.
Ingreso de datos no numéricos en campos numéricos.	Se ingresa datos no numéricos en campos requeridos como numéricos.	El sistema no debe permitir el ingreso de datos en las calificaciones que no son numéricos y debe mostrar un mensaje de error que informe la existencia de un error en el tipo de dato ingresado (campo inválido).

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Actualizar calificaciones de los estudiantes de un curso**

Esta prueba consiste en la actualización o modificación de calificaciones de los

estudiantes inscritos en un curso.

Tabla 3-71: Casos de Prueba: Actualizar calificaciones de los estudiantes de un curso

Caso de Prueba	Entrada	Resultado Esperado
Actualizar las calificaciones de un estudiante de un curso.	Se ingresa la nueva calificación obtenida en cada uno de los conceptos de calificación de un estudiante.	El sistema debe actualizar las calificaciones obtenidas en los diferentes conceptos de calificación de un estudiante.
Actualización con datos nulos (vacíos) en las calificaciones de un estudiante de un curso, que es de ingreso obligatorio.	Se omite el ingreso de alguna o todas las calificaciones obtenidas en cada uno de los conceptos de calificación de un estudiante, que son de ingreso obligatorio.	El sistema no debe permitir la actualización y debe emitir un mensaje informando que se deben ingresar todos los datos para poder guardar correctamente las calificaciones del estudiante.
Actualización con datos no numéricos en campos numéricos.	Se ingresa datos no numéricos en campos requeridos como numéricos.	El sistema no debe permitir la actualización de datos en las calificaciones que no son numéricos y debe mostrar un mensaje de error que informe la existencia de un error en el tipo de dato ingresado (campo inválido).
Actualizar los datos del concepto de calificación.	Ingreso de nuevos datos en el concepto de calificación	El sistema debe buscar, leer y actualizar los datos del concepto de calificación. Y emitir un mensaje del éxito de la operación efectuada.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Eliminar calificaciones de los estudiantes de un curso**

Esta prueba consiste en la eliminación de calificaciones de los estudiantes de un curso**.**

Tabla 3-72: Casos de Prueba: Eliminar calificaciones de los estudiantes de un curso

Caso de Prueba	Entrada	Resultado Esperado
Eliminación de conceptos de calificación de los estudiantes de un curso.	Se selecciona al concepto de calificación que se desea eliminar.	El sistema debe emitir un mensaje de confirmación solicitando al usuario que confirme la acción a realizar. Y emitir un mensaje del éxito de la operación.
Cancelación de la acción de eliminación de conceptos de calificación de los estudiantes de un curso.	El administrador procede con la eliminación del concepto de calificación y se le presenta el mensaje de confirmación.	El sistema al momento que se le presente el mensaje de confirmación con las opciones de aceptar o cancelar y cuando se seleccione la opción de Cancelar, el sistema debe detener la acción de eliminación del concepto de calificación.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Cambiar el orden de las calificaciones de los estudiantes de un curso**

Esta prueba consiste en el cambio de orden entre los conceptos de calificación de los de los estudiantes de un curso**.**

Tabla 3-73: Casos de Prueba: Cambiar el orden de las calificaciones de los estudiantes de un curso

Caso de Prueba	Entrada	Resultado Esperado
Mover ascendente o descendente el concepto de calificación	Se selecciona en el concepto de calificación el movimiento ascendente o descendente.	El sistema debe permitir el cambio de orden (posicionamiento) entre conceptos de calificación. Y emitir un mensaje del éxito de la operación.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Mostrar/Ocultar las calificaciones de los estudiantes de un curso**

Esta prueba consiste en la visibilidad o no de una calificación de los estudiantes.

Tabla 3-74: Casos de Prueba: Mostrar/Ocultar las calificaciones de los estudiantes de un curso

Caso de Prueba	Entrada	Resultado Esperado
Mostrar/Ocultar las calificaciones de los estudiantes de un curso	Se selecciona en el concepto de calificación la opción de ocultamiento o visibilidad.	El sistema debe permitir cambiar el estado de visibilidad (hacer visible ó hacer invisible) de una calificación de los estudiantes. Y emitir un mensaje del éxito de la operación seleccionada.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

HISTORIA 9

- **Visualización del Currículum Académico del Estudiante**

Esta prueba consiste en la presentación del Currículum Académico del estudiante en relación a una carrera o programa académico.

Tabla 3-75: Casos de Prueba: Visualización del Currículum Académico del Estudiante

Caso de Prueba	Entrada	Resultado Esperado
Visualizar el Currículum o expediente académico del estudiante.	Seleccionar la carrera o programa académico	El sistema debe presentar el currículum del estudiante con la siguiente información: Nombre del curso o materia, calificación, tipo de aprobación, créditos, entre otros, así como también los datos de la carrera y datos del estudiante.
No existe información académica del	Seleccionar la carrera o	El sistema debe mostrar un mensaje que “No

estudiante.	programa académico	existe información académica del estudiante".

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

Procedimiento de Pruebas

HISTORIA 6

Para cada uno de los procedimientos de prueba de esta iteración y específicamente de esta Historia, el usuario profesor tendrá que realizar los siguientes procedimientos básicos para acceder a la herramienta de "*Calificaciones*". Estos procedimientos básicos serán los pasos iniciales para los subsiguientes procedimientos de los casos de prueba de esta Historia.

1 En la pantalla correspondiente de ingreso al sistema el Profesor de un curso particular ingresa su *login* y *password*.

2 El usuario profesor presiona el botón *OK*.

3 El sistema hace la verificación de los datos ingresados y si tiene perfil de profesor de un curso en particular.

4 El profesor seleccionará del listado de sus cursos, al curso que desee dar ingresar las calificaciones.

5 En todos los recursos disponibles para el curso, el profesor seleccionará la herramienta Calificaciones.

- **Procedimiento de prueba para el caso de prueba Ingresar calificaciones de los estudiantes de un curso**

1 En la herramienta Calificaciones se muestra el listado de todos los estudiantes inscritos en el curso seleccionado.

2 Pulsar sobre la opción *Añadir calificación parcial o Añadir calificación final.*

2.1 *Añadir calificación parcial*: Se ingresarán los datos correspondientes al concepto de calificación parcial y las

calificaciones parciales obtenidas por los estudiantes.

2.2 *Añadir calificación fina*l: Se ingresarán las calificaciones de aprobación del curso junto con su tipo de aprobación (exonerado, aprobado, fallido, etc.).

3 Pulsar el botón *Guardar Calificaciones*.

4 El sistema verifica que todos los casos esté completos y correctos para proceder con el ingreso, en caso de no cumplir con esta normativa, el sistema despliega un mensaje de error indicando que se ingresen los datos completos y correctos de todos los estudiantes.

- **Procedimiento de prueba para el caso de prueba Actualizar calificaciones de los estudiantes de un curso**

1 En la herramienta Calificaciones se muestra el listado de todos los estudiantes inscritos en el curso seleccionado junto con las calificaciones registradas por el profesor y una leyenda informativa de calificaciones.

2 Para la actualización de calificaciones se presentan dos casos:

2.1 *Actualización de las calificaciones de un estudiante*: El profesor podrá actualizar las calificaciones de un estudiante en los diferentes conceptos de calificación.

2.2 *Actualización del concepto de calificación*: El profesor podrá actualizar los datos informativos de la calificación (concepto de calificación)

3 Pulsar el botón *OK*.

4 El sistema verifica que todos los casos esté completos y correctos para proceder con la actualización, en caso de no cumplir con esta normativa, el sistema despliega un mensaje de error indicando que se ingresen los datos completos y correctos de todos los estudiantes.

- **Procedimiento de prueba para el caso de prueba Eliminar calificaciones de los estudiantes de un curso**

  1 En la herramienta Calificaciones se muestra el listado de todos los estudiantes inscritos en el curso seleccionado junto con las calificaciones registradas por el profesor y una leyenda informativa de calificaciones.

  2 En la leyenda de calificaciones pulsará el botón de *Eliminar* que se presenta en cada uno de los conceptos de calificaciones registradas.

  3 Se presenta un mensaje de confirmación de la acción a realizar

  4 El sistema elimina la información relacionada con el concepto de calificación.

- **Procedimiento de prueba para el caso de prueba Cambiar el orden de las calificaciones de los estudiantes de un curso**

  1 En la herramienta Calificaciones se muestra el listado de todos los estudiantes inscritos en el curso seleccionado junto con las calificaciones registradas por el profesor y una leyenda informativa de calificaciones.

  2 En la leyenda de calificaciones pulsará el botón que indique el movimiento ascendente o descendente del concepto de calificación.

  3 El sistema cambia la posición (el orden) del concepto de calificación según los requerimientos del profesor.

- **Procedimiento de prueba para el caso de prueba Mostrar/Ocultar las calificaciones de los estudiantes de un curso**

  1 En la herramienta Calificaciones se muestra el listado de todos los estudiantes inscritos en el curso seleccionado junto con las calificaciones registradas por el profesor y una leyenda informativa de calificaciones.

2 En la leyenda de calificaciones pulsará el botón que muestre u oculte el concepto de calificación.

3 El sistema cambia la visibilidad del concepto de calificación. Un concepto de calificación oculto no será visible por los estudiantes.

HISTORIA 9

- **Procedimiento de prueba para el caso de prueba Visualización del Currículum Académico del Estudiante**

1 En la pantalla correspondiente de ingreso al sistema el Estudiante de un curso particular ingresa su login y password.

2 El estudiante presiona el botón OK.

3 El sistema hace la verificación de los datos ingresados y si tiene perfil de estudiante.

4 En la sección de servicios en línea, seleccionar la opción de *Currículum Académico*.

5 El estudiante presionará el botón *Currículum Académico* en la carrera deseada.

6 Se visualizará el currículum o expediente académico con la siguiente información: Nombre del curso o materia, calificación, tipo de aprobación, créditos, entre otros, así como también los datos de la carrera y datos del estudiante.

Resultados Obtenidos

HISTORIA 6

- **Ingresar calificaciones de los estudiantes de un curso**

Esta prueba consiste en la adición de calificaciones de los estudiantes inscritos en

un curso.

Tabla 3-76: Resultados de los Casos de Prueba: Añadir calificaciones de los estudiantes de un curso

Caso de Prueba	Entrada	Resultado Obtenido
Ingresar calificaciones parciales de los estudiantes de un curso.	Se ingresa los datos correspondientes al concepto de la calificación (categoría, nombre, etc.). Se ingresa la calificación obtenida por cada uno de los estudiantes.	El sistema almacena las calificaciones parciales obtenidas por los estudiantes en base al concepto de calificación ingresado por el profesor, y como éxito de la operación emite un mensaje.
Ingresar calificaciones finales de los estudiantes de un curso.	Se ingresa la calificación obtenida y el tipo de aprobación (exonerado, aprobado, fallido, etc.) por cada uno de los estudiantes.	El sistema almacena las calificaciones finales (calificaciones de aprobación del curso) obtenidas por los estudiantes junto con su tipo de aprobación (exonerado, aprobado, fallido, etc.). Y emite un mensaje del éxito de la operación.
Obviar el ingreso de las calificaciones de los estudiantes de un curso.	Se omite el ingreso de alguna o todas las calificaciones de los estudiantes que es de ingreso obligatorio.	El sistema en el momento que detecta que la calificación de un estudiante es obviada emite un mensaje informando que se deben ingresar todos los datos para proceder con el almacenamiento de las calificaciones de los estudiantes, previa a la verificación de los mismos.
Ingreso de datos no numéricos en campos numéricos.	Se ingresa datos no numéricos en campos requeridos como numéricos.	El sistema solamente acepta caracteres numéricos (reales positivos) para el ingreso de las calificaciones de los estudiantes, omitiendo el ingreso de caracteres como letras, símbolos, etc. Muestra un mensaje de error que informa la existencia de un error en el tipo de dato ingresado (campo inválido).

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Actualizar calificaciones de los estudiantes de un curso**

Esta prueba consiste en la actualización o modificación de calificaciones de los estudiantes inscritos en un curso.

Tabla 3-77: Resultados de los Casos de Prueba: Actualizar calificaciones de los estudiantes de un curso

Caso de Prueba	Entrada	Resultado Obtenido
Actualizar las calificaciones de un estudiante de un curso.	Se ingresa la nueva calificación obtenida en cada uno de los conceptos de calificación de un estudiante.	El sistema actualiza las calificaciones obtenidas en los diferentes conceptos de calificación de un estudiante, previo a la verificación de los mismos.
Actualización con	Se omite el ingreso de	El sistema no permite la actualización y

datos nulos (vacíos) en las calificaciones de un estudiante de un curso, que es de ingreso obligatorio.	alguna o todas las calificaciones obtenidas en cada uno de los conceptos de calificación de un estudiante, que son de ingreso obligatorio.	presenta un mensaje indicando que es obligatorio ingresar todos los datos para poder actualizar correctamente las calificaciones del estudiante.
Actualización con datos no numéricos en campos numéricos.	Se ingresa datos no numéricos en campos requeridos como numéricos.	El sistema no permite la actualización de datos en las calificaciones que no son numéricos indicando a través de un mensaje que se deben ingresar únicamente números (enteros o decimales)
Actualizar los datos del concepto de calificación.	Ingreso de nuevos datos en el concepto de calificación	El sistema buscar, lee y actualiza los datos del concepto de calificación. Y emite un mensaje del éxito de la actualización efectuada.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Eliminar calificaciones de los estudiantes de un curso**

Esta prueba consiste en la eliminación de calificaciones de los estudiantes de un curso**.**

Tabla 3-78: Resultados de los Casos de Prueba: Eliminar calificaciones de los estudiantes de un curso

Caso de Prueba	Entrada	Resultado Obtenido
Eliminación de conceptos de calificación de los estudiantes de un curso.	Se selecciona al concepto de calificación que se desea eliminar.	El sistema presenta un mensaje de confirmación solicitando al usuario que confirme la acción a realizar. Y emite un mensaje del éxito de la operación.
Cancelación de la acción de eliminación de conceptos de calificación de los estudiantes de un curso.	El administrador procede con la eliminación del concepto de calificación y se le presenta el mensaje de confirmación.	El sistema con el mensaje de confirmación que se le presenta con las opciones de aceptar o cancelar y cuando se selecciona la opción Cancelar, el sistema cancela la acción de eliminación del concepto de calificación.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Cambiar el orden de las calificaciones de los estudiantes de un curso**

Esta prueba consiste en el cambio de orden entre los conceptos de calificación de los de los estudiantes de un curso**.**

Tabla 3-79: Resultados de los Casos de Prueba: Cambiar el orden de las calificaciones de los estudiantes de un curso

Caso de Prueba	Entrada	Resultado Obtenido
Mover ascendente o	Se selecciona en el	El sistema permite el ordenamiento de los

descendente el concepto de calificación	concepto de calificación el movimiento ascendente o descendente.	conceptos de calificación a través del movimiento ascendente o descendente de los mismos. Y emite un mensaje del éxito de la operación.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Mostrar/Ocultar las calificaciones de los estudiantes de un curso**

Esta prueba consiste en la visibilidad o no de una calificación de los estudiantes.

Tabla 3-80: Casos de Prueba: Mostrar/Ocultar las calificaciones de los estudiantes de un curso

Caso de Prueba	Entrada	Resultado Esperado
Mostrar/Ocultar las calificaciones de los estudiantes de un curso	Se selecciona en el concepto de calificación la opción de ocultamiento o visibilidad.	El sistema permite cambiar el estado de visibilidad (hacer visible ó hacer invisible) de una calificación de los estudiantes. Y emite un mensaje del éxito de la operación seleccionada.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

HISTORIA 9

- **Visualización del Currículum Académico del Estudiante**

Esta prueba consiste en la presentación del Currículum Académico del estudiante en relación a una carrera o programa académico.

Tabla 3-81: Resultados de los Casos de Prueba: Visualización del Currículum Académico del Estudiante

Caso de Prueba	Entrada	Resultado Obtenido
Visualizar el Currículum o expediente académico del estudiante.	Seleccionar la carrera o programa académico	El sistema muestra en forma de reporte el currículum del estudiante con la siguiente información: Nombre del curso o materia, calificación, tipo de aprobación, créditos, entre otros, así como también los datos de la carrera y datos del estudiante.
No existe información académica del estudiante.	Seleccionar la carrera o programa académico	El sistema muestra un mensaje en el que se indica que “No existe información académica del estudiante”.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

3.3.4 ITERACION 4

3.3.4.1 Diseño

En la fase de diseño se detallan las tarjetas CRC para las Historias 7 y 8, según lo

definido en las Historias de Usuarios y la Planificación de Entregas.

En este sentido se detallan las Tarjetas CRC de las historias correspondientes a esta iteración.

a. Clase Matricula (EnrollManager)

Tabla 3-82: Iteración 4 - Tarjeta CRC para la Clase EnrollManager

Clase	**EnrollManager**
Descripción	Clase localizada en la capa del negocio para gestionar la matrícula de un estudiante en una Carrera o Programa Académico en un Período Académico determinado.
Atributos	- enroll_id (integer) - enroll_number (string) - pay_met_id (integer) - user_id (integer) - academic_period_id (integer) - enroll_date (date) - enroll_administrative_rate (float) - enroll_subtotal (float) - enroll_total (float) - enroll_status (char) - enroll_user_id (integer) - courseList (array) - resourceList (array)
Métodos	EnrollManager() read(parámetros) get() set(parámetro) save() delete() isInList(parámetros) selectNbrCourses() selectNbrResources() addToCourseList(parámetro) addToResourceList(parámetro) getCoursesAvailable(parámetros) getCoursesOfEnrollment() getResourcesOfEnrollment() ConfirmEnrollment()

Responsabilidades	- Inicializar el objeto (Constructor de la clase). - Obtener la información de la matrícula de un estudiante en un determinado período académico. - Obtener el código de un objeto EnrollManager (código de la matrícula). - Asignar un nuevo código de matrícula al código de un objeto EnrollManager (código de la matrícula). - Almacenar la información del proceso de matrícula (inserción o actualización). - Eliminar la información de una matrícula. - Consultar si un objeto determinado se encuentra en un listado determinado (Listado de cursos, materiales o recursos). - Obtener el número de cursos registrados en la matrícula. - Obtener el número de materiales o recursos registrados en la matrícula. - Agregar a la lista de cursos de la matrícula a efectuarse, un objeto curso pasado como parámetro. - Agregar a la lista de materiales o recursos de la matrícula a efectuarse, un objeto recurso pasado como parámetro. - Obtener los cursos en los que el usuario podría matricularse dentro de una carrera especificada como parámetro, en base a una malla curricular. - Obtener los cursos registrados en el proceso de matrícula de un usuario. - Obtener los materiales o recursos registrados en el proceso de matrícula de un usuario. - Confirmar matrícula de un estudiante para el uso del aula virtual.
Colaboradores	ADOdb CourseManager ResourceManager

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

3.3.4.2 Implementación

A continuación se muestra el código fuente de las clases antes diseñadas. Para mayor detalle de la implementación ver en el CD adjunto, documento: *Dokeos for University - Implementación/index.html*.

a. Clase Matricula (EnrollManager)

Tabla 3-83: Iteración 4 – Implementación de la Clase EnrollManager

Clase: EnrollManager
class EnrollManager { var $enroll_id; var $enroll_number; var $pay_met_id;

```
	var  $user_id;
	var  $academic_period_id;
	var  $enroll_date;
	var  $enroll_administrative_rate;
	var  $enroll_subtotal;
	var  $enroll_total;
	var  $enroll_status;
	var  $enroll_user_id;
	var  $courseList;  // array with the list of this enroll's course
	var  $resourceList;  // array with the list of this enroll's resource
	var  $paymentList;
	/**
	* Constructor de la clase EnrollManager (Inicializar el objeto)
	*/
	function EnrollManager($enroll_id = null)
	{
		global $conn;
		if($enroll_id != null)
		{
			$this->enroll_id = $enroll_id;
			// begin Recordset
			$query_rsEnrollManager = "SELECT * FROM enroll WHERE enroll_id='$this->enroll_id'";
			$rsEnrollManager = $conn->SelectLimit($query_rsEnrollManager) or die($conn-
>ErrorMsg());
			$totalRows_rsEnrollManager = $rsEnrollManager->RecordCount();
			// end Recordset

			if ($totalRows_rsEnrollManager>0)
			{
				$this->enroll_id = $rsEnrollManager->Fields('enroll_id');
				$this->enroll_number = $rsEnrollManager->Fields('enroll_number');
				$this->pay_met_id = $rsEnrollManager->Fields('pay_met_id');
				$this->user_id = $rsEnrollManager->Fields('user_id');
				$this->academic_period_id = $rsEnrollManager->Fields('academic_period_id');
				$this->enroll_date = $rsEnrollManager->Fields('enroll_date');
				$this->enroll_administrative_rate = $rsEnrollManager-
>Fields('enroll_administrative_rate');
				$this->enroll_subtotal = $rsEnrollManager->Fields('enroll_subtotal');
				$this->enroll_total = $rsEnrollManager->Fields('enroll_total');
				$this->enroll_status = $rsEnrollManager->Fields('enroll_status');
				$this->enroll_user_id = $rsEnrollManager->Fields('enroll_user_id');
				$this->courseList = array();
				$this->resourceList = array();
				$this->paymentList = array();
				// Read Courses of Enroll
				$this->getCoursesOfEnrollment();
				// Read Resources of Enroll
				$this->getResourcesOfEnrollment();
			}
			else
			{
				$this->enroll_id = "";
				$this->enroll_number = "";
				$this->pay_met_id = "";
				$this->user_id = "";
				$this->academic_period_id = "";
```

```
                                        $this->enroll_date = "";
                                        $this->enroll_administrative_rate = "";
                                        $this->enroll_subtotal = "";
                                        $this->enroll_total = "";
                                        $this->enroll_status = "";
                                        $this->enroll_user_id = "";
                                        $this->courseList = array();
                                        $this->resourceList = array();
                                        $this->paymentList = array();
                                }
                        }
                        else
                        {
                                $this->enroll_id = "";
                                $this->enroll_number = "";
                                $this->pay_met_id = "";
                                $this->user_id = "";
                                $this->academic_period_id = "";
                                $this->enroll_date = "";
                                $this->enroll_administrative_rate = "";
                                $this->enroll_subtotal = "";
                                $this->enroll_total = "";
                                $this->enroll_status = "";
                                $this->enroll_user_id = "";
                                $this->courseList = array();
                                $this->resourceList = array();
                                $this->paymentList = array();
                        }
                }//end constructor EnrollManager

                /**
                * Obtener la información de la matrícula de un estudiante en un determinado período académico.
                */
                function read($enroll_id, $user_id, $academic_period_id)
                {
                        global $conn;
                        $this->enroll_id = $enroll_id;
                        $this->user_id = $user_id;
                        $this->academic_period_id = $academic_period_id;
                        // begin Recordset
                        $query_rsEnrollManager = "SELECT * FROM enroll WHERE enroll_id LIKE '$this->enroll_id' AND
user_id LIKE '$this->user_id' AND academic_period_id LIKE '$this->academic_period_id'";
                        $rsEnrollManager = $conn->SelectLimit($query_rsEnrollManager) or die($conn->ErrorMsg());
                        $totalRows_rsEnrollManager = $rsEnrollManager->RecordCount();
                        // end Recordset

                        if ($totalRows_rsEnrollManager == 1)
                        {
                                $this->enroll_id = $rsEnrollManager->Fields('enroll_id');
                                $this->enroll_number = $rsEnrollManager->Fields('enroll_number');
                                $this->pay_met_id = $rsEnrollManager->Fields('pay_met_id');
                                $this->user_id = $rsEnrollManager->Fields('user_id');
                                $this->academic_period_id = $rsEnrollManager->Fields('academic_period_id');
                                $this->enroll_date = $rsEnrollManager->Fields('enroll_date');
                                $this->enroll_administrative_rate = $rsEnrollManager->Fields('enroll_administrative_rate');
                                $this->enroll_subtotal = $rsEnrollManager->Fields('enroll_subtotal');
```

```
                $this->enroll_total = $rsEnrollManager->Fields('enroll_total');
                $this->enroll_status = $rsEnrollManager->Fields('enroll_status');
                $this->enroll_user_id = $rsEnrollManager->Fields('enroll_user_id');
                $this->courseList = array();
                $this->resourceList = array();
                $this->paymentList = array();

                // Read Courses of Enroll
                $this->getCoursesOfEnrollment();
                // Read Resources of Enroll
                $this->getResourcesOfEnrollment();
                return true;
        }
        else
        {
                $this->enroll_id = "";
                $this->enroll_number = "";
                $this->pay_met_id = "";
                $this->user_id = "";
                $this->academic_period_id = "";
                $this->enroll_date = "";
                $this->enroll_administrative_rate = "";
                $this->enroll_subtotal = "";
                $this->enroll_total = "";
                $this->enroll_status = "";
                $this->enroll_user_id = "";
                $this->courseList = array();
                $this->resourceList = array();
                $this->paymentList = array();
                return false;
        }
}//end read

/**
* Almacenar la información del proceso de matrícula (inserción o actualización) en la base de datos.
*/
function save()
{
        global $conn;
        $now = date("Y-m-d H:i:s");
        if ($this->enroll_id == "")
        {
                $insertSQL = sprintf("INSERT INTO enroll (enroll_number, pay_met_id, user_id,
academic_period_id, enroll_date, enroll_administrative_rate, enroll_subtotal, enroll_total, enroll_status, enroll_user_id)
VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)",
                GetSQLValueString($this->enroll_number, "text"),
                GetSQLValueString($this->pay_met_id, "int"),
                GetSQLValueString($this->user_id, "int"),
                GetSQLValueString($this->academic_period_id, "int"),
                GetSQLValueString($now, "date"),
                GetSQLValueString($this->enroll_administrative_rate, "double"),
                GetSQLValueString($this->enroll_subtotal, "double"),
                GetSQLValueString($this->enroll_total, "double"),
                GetSQLValueString($this->enroll_status, "text"),
                GetSQLValueString($this->enroll_user_id, "int"));
                $result = $conn->Execute($insertSQL) or die($conn->ErrorMsg());
```

```
                        if($conn->Affected_Rows())
                        {
                                $this->enroll_id = mysql_insert_id();
                                // Insert the enroll_detail
                                foreach($this->courseList as $course)
                                {
                                        $insertSQL = sprintf("INSERT INTO enroll_detail (code, enroll_id,
enroll_detail_cost) VALUES (%s, %s, %s)",
                                        GetSQLValueString($course->code, "text"),
                                        GetSQLValueString($this->enroll_id, "int"),
                                        GetSQLValueString($course->cost_course, "double"));
                                        $result = $conn->Execute($insertSQL) or die($conn->ErrorMsg());
                                }
                                // Insert the enroll_resource_detail
                                foreach($this->resourceList as $resource)
                                {
                                        $insertSQL = sprintf("INSERT INTO enroll_resource_detail (resource_id,
enroll_id, enroll_resource_detail_cost) VALUES (%s, %s, %s)",
                                        GetSQLValueString($resource->resource_id, "int"),
                                        GetSQLValueString($this->enroll_id, "int"),
                                        GetSQLValueString($resource->resource_cost, "double"));
                                        $result = $conn->Execute($insertSQL) or die($conn->ErrorMsg());
                                }
                                return $conn->Affected_Rows();
                        }
                }
                else
                {
                        $updateSQL = sprintf("UPDATE enroll SET enroll_number=%s, pay_met_id=%s,
user_id=%s, academic_period_id=%s, enroll_date=%s, enroll_administrative_rate=%s, enroll_subtotal=%s,
enroll_total=%s, enroll_status=%s WHERE enroll_id=%s",
                        GetSQLValueString($this->enroll_number, "text"),
                        GetSQLValueString($this->pay_met_id, "int"),
                        GetSQLValueString($this->user_id, "int"),
                        GetSQLValueString($this->academic_period_id, "int"),
                        GetSQLValueString($now, "date"),
                        GetSQLValueString($this->enroll_administrative_rate, "double"),
                        GetSQLValueString($this->enroll_subtotal, "double"),
                        GetSQLValueString($this->enroll_total, "double"),
                        GetSQLValueString($this->enroll_status, "text"),
                        GetSQLValueString($this->enroll_id, "int"));
                        $result = $conn->Execute($updateSQL) or die($conn->ErrorMsg());
                        return $conn->Affected_Rows();
                }
                return false;
        }//end save()

        /**
        * Eliminar la información de una matrícula.
        */
        function delete()
        {
                global $conn;
                $deleteSQL = sprintf("DELETE FROM enroll WHERE enroll_id=%s",
                GetSQLValueString($this->enroll_id, "int"));
                $result = $conn->Execute($deleteSQL) or die($conn->ErrorMsg());
```

```
                $deleteSQL = sprintf("DELETE FROM enroll_detail WHERE enroll_id=%s",
                GetSQLValueString($this->enroll_id, "int"));
                $result = $conn->Execute($deleteSQL) or die($conn->ErrorMsg());

                $deleteSQL = sprintf("DELETE FROM enroll_resource_detail WHERE enroll_id=%s",
                GetSQLValueString($this->enroll_id, "int"));
                $result = $conn->Execute($deleteSQL) or die($conn->ErrorMsg());
                return $conn->Affected_Rows();
        } // end delete()


        /**
        * Consultar si un objeto determinado se encuentra en un listado determinado (Listado de cursos, materiales o
recursos).
        */
        function isInList($id,$list)
        {
                return array_key_exists($id,$list);
        }

        /**
        * Obtener el número de cursos registrados en la matrícula.
        */
        function selectNbrCourses()
        {
                return sizeof($this->courseList);
        }

        /**
        * Obtener el número de materiales o recursos registrados en la matrícula.
        */
        function selectNbrResources()
        {
                return sizeof($this->resourceList);
        }

        /**
        * Agregar a la lista de cursos de la matrícula a efectuarse, un objeto curso pasado como parámetro.
        */
        function addToCourseList($course)
        {
                // checks if the course object is not in the list
                if(!$this->isInList($course->code,$this->courseList))
                {
                        $this->courseList[$course->code] = $course;
                        return true;
                }
                return false;
        }

        /**
        * Agregar a la lista de materiales o recursos de la matrícula a efectuarse, un objeto recurso pasado como
parámetro.
        */
        function addToResourceList($resource)
```

```
	{
		// checks if the resource object is not in the list
		if(!$this->isInList($resource->resource_id,$this->resourceList))
		{
			$this->resourceList[$resource->resource_id]=$resource;
			return true;
		}
		return false;
	}

	/**
	* Obtener los cursos en los que el usuario podría matricularse dentro de una carrera especificada como parámetro, en base a una malla curricular.
	*/
	function getCoursesAvailable($career_id,$user_id)
	{
		$approved_courses = array();
		$failed_courses = array();
		$parents_courses = array();
		$available_courses = array();
		$candidates_courses = array();

		$tmp = array();
		$ar = new AcademicRecordManager();

		if($ar->isNewStudent($career_id,$user_id))
		{
			//$available_courses = array_merge($ar->getCourseByLevel($career_id, "1"),$ar->getOptativeCoursesWithOutDependency($career_id));
			$available_courses = array_merge($ar->getCourseByLevel($career_id, "1"));
		}
		else
		{
			// Loop to get Possibles courses to take by user
			$approved_courses = $ar->getApprovedCourses($career_id,$user_id);
			$failed_courses = $ar->getFailedCourses($career_id,$user_id);
			//echo "<pre><strong>CURSOS APROBADOS:<BR></strong>".print_r($approved_courses,true)."</pre>";
			//echo "<pre><strong>CURSOS FALLADOS:<BR></strong>".print_r($failed_courses,true)."</pre>";
			if(count($approved_courses)>0)
			{
				// Bucle para obtener los posibles cursos que podría tomar es usuario
				foreach($approved_courses as $ac)
				{
					//Obtiene los posibles cursos a tomar dado un curso aprobado
					$tmp = $ar->getCoursesChildren($career_id,$ac,'1');
					foreach($tmp as $t) $candidates_courses[] = $t;
					$candidates_courses = array_unique($candidates_courses);
				}
				foreach($candidates_courses as $cc)
				{
					$flag1 = true;		$flag2 = true;
					// Obtiene los cursos de los que depende un curso - Pre_requisitos
					$parents_courses = $ar->getCoursesParents($career_id,$cc,'1');
					foreach($parents_courses as $pc)
```

```
                                        {
                                                if (in_array($pc,$failed_courses) ||
!in_array($pc,$approved_courses))
                                                {
                                                        $flag1 = false;         break 1;
                                                }
                                        }
                                        // Obtiene los cursos de los que depende un curso - Co_requisitos
                                        $co_req_courses = $ar->getCoursesParents($career_id,$cc,'2');
                                        //echo "<pre><strong>CURSOS CORREQUISITOS DEL CURSO:
$cc<BR></strong>".print_r($co_req_courses,true)."</pre>";
                                        foreach($co_req_courses as $crc)
                                        {
                                                $parents_courses = $ar-
>getCoursesParents($career_id,$crc,'1');
                                                foreach($parents_courses as $pc)
                                                {
                                                        if (in_array($pc,$failed_courses) ||
!in_array($pc,$approved_courses))
                                                                $flag2 = false;         break 1;
                                                }
                                        }
                                        if($flag1 && $flag2 && !in_array($cc,$available_courses))
                                        $available_courses[] = $cc;
                                }
                                $available_courses = array_merge($failed_courses,$available_courses);
                                unset($tmp);
                                $tmp = $ar->getOptativeCoursesWithOutDependency($career_id);
                                //Bucle para agregar los cursos optativos si todavia no los cursa
                                foreach($tmp as $t)
                                {
                                        if(!in_array($t,$approved_courses) && !in_array($t,$available_courses))
                                        $available_courses[] = $t;
                                }
                        }
                        else
                        {
                                //$available_courses = array_merge($ar-
>getFailedCourses($career_id,$user_id),$ar->getOptativeCoursesWithOutDependency($career_id));
                                $available_courses = array_merge($ar->getCourseByLevel($career_id, "1"),$ar-
>getFailedCourses($career_id,$user_id));
                        }
                }
                return array_unique($available_courses);
                //return $available_courses;
        }

        /**
        * Obtener los cursos registrados en el proceso de matrícula de un usuario.
        */
        function getCoursesOfEnrollment()
        {
                global $conn;
                // begin Recordset
                $query_rsCoursesOfEnrollment = "SELECT * FROM enroll e, enroll_detail ed WHERE e.enroll_id =
ed.enroll_id AND e.enroll_id = '$this->enroll_id'";
```

```
                $rsCoursesOfEnrollment = $conn->SelectLimit($query_rsCoursesOfEnrollment) or die($conn-
>ErrorMsg());
                $totalRows_rsCoursesOfEnrollment = $rsCoursesOfEnrollment->RecordCount();
                // end Recordset
                while (!$rsCoursesOfEnrollment->EOF) {
                        $course = new Course($rsCoursesOfEnrollment->Fields('code'));
                        //Asignamos al costo del curso, el valor que costó cuando se efectuó la matriculación
                        $course->cost_course = $rsCoursesOfEnrollment->Fields('enroll_detail_cost');
                        $this->courseList[$course->code] = $course;
                        $rsCoursesOfEnrollment->MoveNext();
                }
        }

        /**
        * Obtener los materiales o recursos registrados en el proceso de matrícula de un usuario.
        */
        function getResourcesOfEnrollment()
        {
                global $conn;
                // begin Recordset
                $query_rsResourcesOfEnrollment = "SELECT * FROM enroll e, enroll_resource_detail ed WHERE
e.enroll_id = ed.enroll_id AND e.enroll_id = '$this->enroll_id'";
                $rsResourcesOfEnrollment = $conn->SelectLimit($query_rsResourcesOfEnrollment) or die($conn-
>ErrorMsg());
                $totalRows_rsResourcesOfEnrollment = $rsResourcesOfEnrollment->RecordCount();
                // end Recordset
                while (!$rsResourcesOfEnrollment->EOF) {
                        $resource = new ResourceManager($rsResourcesOfEnrollment->Fields('resource_id'));
                        //Asignamos al costo del recurso, el valor que costó cuando se efectuó la matriculación
                        $resource->resource_cost = $rsResourcesOfEnrollment-
>Fields('enroll_resource_detail_cost');
                        $this->resourceList[$resource->resource_id] = $resource;
                        $rsResourcesOfEnrollment->MoveNext();
                }
        }
}
```

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

3.3.4.3 Pruebas

De acuerdo a lo estipulado en el capítulo 2, en la explicación de la metodología Extreme Programming se planteará un plan de pruebas para cada una de las Historias de Usuario de la Iteración 4.

Plan de Pruebas

HISTORIA 7

- **Visualización de cursos disponibles para la matriculación de un estudiante en una carrera o programa académico**

Esta prueba consiste en la presentación de cursos disponibles para matriculación de un estudiante en una carrera o programa académico dentro de un período académico.

Tabla 3-84: Casos de Prueba: Visualización de cursos disponibles para la matriculación de un estudiante en una carrera o programa académico

Caso de Prueba	Entrada	Resultado Esperado
Visualización de cursos o materias disponibles para matriculación de un estudiante en una carrera.	Identificación del estudiante (ID) y la carrera a matricularse.	El sistema debe presentar los cursos o materias disponibles para matriculación de un estudiante en una carrera, en base a los cursos aprobados por el estudiante dentro de dicha carrera y de efectuar un chequeo de pre-requisitos.
Verificar períodos académicos con estado Actual y es temporada de matrículas	Información de la carrera a matricularse.	El sistema debe comprobar si la carrera cuanta con un período académico activo y además si es temporada de matrículas, caso contrario debe emitir un mensaje indicando que no es posible efectuar el proceso de matrícula.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Seleccionar los cursos que el estudiante desea tomar en el presente período académico de la carrera**

Esta prueba consiste en la selección de cursos y/o materiales (recursos) en el proceso de matrícula.

Tabla 3-85: Casos de Prueba: Seleccionar los cursos que el estudiante desea tomar en el presente período académico de la carrera

Caso de Prueba	Entrada	Resultado Esperado
Selección de cursos	Se selecciona (de un checkbox) los cursos a agregar a la matrícula.	El sistema debe recalcular automáticamente el valor de la matrícula en cada selección de un curso.
Selección de materiales o recursos	Se selecciona (de un checkbox) los materiales a agregar a la matrícula.	El sistema debe recalcular automáticamente el valor de la matrícula en cada selección de un material.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Registrar matrícula de un estudiante en una carrera**

Tabla 3-86: Casos de Prueba: Registrar matrícula de un estudiante en una carrera

Caso de Prueba	Entrada	Resultado Esperado
Registrar los cursos y materiales en el	Cursos y materiales seleccionados por el	El sistema debe permitir el registro de los cursos y materiales seleccionados, siempre y

proceso de matrícula	usuario	cuando se haya ingresado correctamente el código de seguridad[26] y generar el comprobante de matrícula como indicador de del éxito de la operación en el proceso de matrícula. El comprobante de matrícula debe contener los cursos y materiales registrados, valor de la matrícula, entre otros.
No seleccionar ningún curso	Ningún curso seleccionado	El sistema no debe permitir el registro de la matrícula y debe emitir un mensaje indicando que se debe seleccionar algún curso.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Actualizar la matrícula de un estudiante por parte del administrador**

Esta prueba consiste en la actualización o modificación de la matrícula de un estudiante por parte del administrador.

Tabla 3-87: Casos de Prueba: Actualizar la matrícula de un estudiante por parte del administrador

Caso de Prueba	Entrada	Resultado Esperado
Agregar o quitar cursos o materiales de una matrícula de una	Matrícula registrada en el período académico con estado actual. Cursos y materiales registrados en la matrícula de un estudiante.	El sistema debe permitir la adición o la eliminación de cursos y/o materiales de una matrícula. El sistema debe generar un nuevo comprobante de matrícula como indicador de del éxito de la operación en el proceso de matrícula.
No seleccionar ningún curso	Ningún curso seleccionado	El sistema no debe permitir el registro de la matrícula y debe emitir un mensaje indicando que se debe seleccionar algún curso.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Eliminar la matrícula de un estudiante por parte del administrador**

Esta prueba consiste en la eliminación o anulación de la matrícula de un estudiante por parte del administrador**.**

Tabla 3-88: Casos de Prueba: Eliminar la matrícula de un estudiante por parte del administrador

Caso de Prueba	Entrada	Resultado Esperado
Eliminación de la matrícula de un estudiante.	Matrícula registrada en el período académico con estado actual. Se selecciona a la matrícula del estudiante	El sistema debe emitir un mensaje de confirmación solicitando al usuario que confirme la acción a realizar. Y emitir un mensaje del éxito de la operación.

[26] Código de seguridad: Código que el usuario puede visualizar en pantalla y que para proseguir con el proceso de matrícula es necesario su ingreso, como señal de que el proceso esta siendo ejecutado por una persona.

Cancelación de la acción de eliminación de la matrícula de un estudiante.	El administrador procede con la eliminación de la matrícula de un estudiante y se le presenta el mensaje de confirmación.	El sistema al momento que se le presente el mensaje de confirmación con las opciones de aceptar o cancelar y cuando se seleccione la opción de Cancelar, el sistema debe detener la acción de eliminación de la matrícula.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

Procedimiento de Pruebas

HISTORIA 7

Para cada uno de los procedimientos de prueba de esta iteración y específicamente de esta Historia, el estudiante tendrá que realizar los siguientes procedimientos básicos para acceder al servicio de *Matriculación*. Estos procedimientos básicos serán los pasos iniciales para los subsiguientes procedimientos de los casos de prueba de esta Historia.

1 En la pantalla correspondiente de ingreso al sistema el usuario ingresa su login y password.

2 El usuario presiona el botón OK.

3 El sistema hace la verificación de los datos ingresados.

4 El estudiante seleccionará la opción "*Servicios en línea*" y seleccionará la opción *Matricula.*

5 El sistema presentará un listado de las carreras o programas académicos disponibles para matriculación y seleccionará una en la que desee matricularse.

- **Procedimiento de prueba para el caso de prueba Visualización de cursos disponibles para la matriculación de un estudiante en una carrera o programa académico**

1 El sistema presenta los cursos disponibles en base a dos criterios:

1.1 *Estudiante nuevo*: presenta todos los cursos o materias de primer

nivel.

1.2 *Estudiante de la carrera*: presenta los cursos disponibles para matriculación en base a los cursos aprobados por el estudiante dentro de dicha carrera y de efectuar un chequeo de pre-requisitos.

2 El sistema presentará los cursos con su respectiva información y recursos o materiales relacionados con cada curso.

- **Procedimiento de prueba para el caso de prueba Seleccionar los cursos que el estudiante desea tomar en el presente período académico de la carrera**

1 El sistema desplegará los cursos que el usuario efectivamente puede tomar (de acuerdo a la malla curricular y materias aprobadas por el estudiante) junto con los materiales relacionados a dichos cursos.

2 El estudiante tiene la opción de seleccionar las cursos que desea tomar y de ser necesario los materiales asociados a los cursos.

3 El sistema recalculará automáticamente el valor de la matrícula con cada selección de un curso y/o material o recurso.

- **Procedimiento de prueba para el caso de prueba Registrar matrícula de un estudiante en una carrera**

1 El sistema desplegará los cursos que el usuario efectivamente puede tomar (de acuerdo a la malla curricular y materias aprobadas por el estudiante) junto con los materiales relacionados a dichos cursos.

2 El estudiante tiene la opción de seleccionar las cursos que desea tomar y de ser necesario los materiales asociados a los cursos.

3 El sistema recalculará automáticamente el valor de la matrícula con cada selección de un curso y/o material o recurso.

4 El estudiante presionará el botón *Registrar Factura.*

5 El sistema realizará la verificación de pre-requisitos y co-requisitos para proceder con el registro, en caso de no cumplir con esta normativa, el sistema despliega un mensaje de alerta informando sobre el resultado de la verificación.

- **Procedimiento de prueba para el caso de prueba Actualizar la matrícula de un estudiante por parte del administrador**

1 En el módulo de matrículas, al administrador se le presenta un listado de todos los estudiantes inscritos en la carrera con las opciones de edición de las matriculas registradas para un período académico con estado Actual. Si se trata de períodos académicos históricos esta opción no esta disponible.

2 El administrador pulsará el botón *Modificar* de una de las matrículas de los estudiantes de la carrera.

3 El sistema desplegará los cursos y los materiales que dicha matrícula posee junto con los cursos que el usuario puede optar.

4 El estudiante tiene la opción de seleccionar nuevos cursos y quitar otros, de igual manera en el caso de los materiales.

5 El sistema recalculará automáticamente el valor de la matrícula con cada selección de un curso y/o material o recurso.

6 El estudiante presionará el botón *Registrar Factura.*

7 El sistema realizará la verificación de pre-requisitos y co-requisitos para proceder con la actualización y generación de un nuevo comprobante de matrícula, en caso de no cumplir con esta normativa, el sistema despliega un mensaje de alerta informando sobre el resultado de la verificación.

- **Procedimiento de prueba para el caso de prueba Eliminar la matrícula de un estudiante por parte del administrador**

1 En el módulo de matrículas, al administrador se le presenta un listado de todos los estudiantes inscritos en la carrera con las opciones de eliminación de las matriculas registradas para un período académico con estado Actual. Si se trata de períodos académicos históricos esta opción no esta disponible.

2 El administrador pulsará el botón *Eliminar* de una de las matrículas de los estudiantes de la carrera.

3 Se presenta un mensaje de confirmación de la acción a realizar.

4 El sistema elimina la información relacionada con la matrícula del estudiante en esa carrera y en ese período académico.

Resultados Obtenidos

HISTORIA 7

- **Visualización de cursos disponibles para la matriculación de un estudiante en una carrera o programa académico**

Esta prueba consiste en la presentación de cursos disponibles para matriculación de un estudiante en una carrera o programa académico dentro de un período académico.

Tabla 3-89: Resultados de los Casos de Prueba: Visualización de cursos disponibles para la matriculación de un estudiante en una carrera o programa académico

Caso de Prueba	Entrada	Resultado Obtenido
Visualización de cursos o materias disponibles para matriculación de un estudiante en una carrera.	Identificación del estudiante (ID) y la carrera a matricularse.	El sistema despliega los cursos o materias disponibles para matriculación de un estudiante en una carrera, en base a los cursos aprobados por el estudiante dentro de dicha carrera y de efectuar un chequeo de pre-requisitos.
Verificar períodos académicos con	Información de la carrera a matricularse.	El sistema verifica si la carrera cuanta con un período académico activo y si es temporada

estado Actual y es temporada de matrículas.		de matrículas, en cualquier otro caso emite un mensaje indicando que no es posible efectuar el proceso de matrícula.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Seleccionar los cursos que el estudiante desea tomar en el presente período académico de la carrera**

Esta prueba consiste en la selección de cursos y/o materiales (recursos) en el proceso de matrícula.

Tabla 3-90: Resultados de los Casos de Prueba: Seleccionar los cursos que el estudiante desea tomar en el presente período académico de la carrera

Caso de Prueba	Entrada	Resultado Obtenido
Selección de cursos	Se selecciona (de un checkbox) los cursos a agregar a la matrícula.	El sistema recalcula automáticamente el valor de la matrícula en cada selección de un curso.
Selección de materiales o recursos	Se selecciona (de un checkbox) los materiales a agregar a la matrícula.	El sistema recalcula automáticamente el valor de la matrícula en cada selección de un material.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Registrar matrícula de un estudiante en una carrera**

Tabla 3-91: Resultados de los Casos de Prueba: Registrar matrícula de un estudiante en una carrera

Caso de Prueba	Entrada	Resultado Obtenido
Registrar los cursos y materiales en el proceso de matrícula	Cursos y materiales seleccionados por el usuario	El sistema permite el registro de los cursos y materiales seleccionados, siempre y cuando se haya ingresado correctamente el código de seguridad[27] y genera el comprobante para indicar que el registro de la matricula se efectuó satisfactoriamente.
No seleccionar ningún curso	Ningún curso seleccionado	El sistema no permite el registro de la matrícula. Siempre debe estar seleccionado algún curso para proceder con el registro. Emite un mensaje indicando que se debe seleccionar algún curso.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

[27] Código de seguridad: Código que el usuario puede visualizar en pantalla y que para proseguir con el proceso de matrícula es necesario su ingreso, como señal de que el proceso esta siendo ejecutado por una persona.

- **Actualizar la matrícula de un estudiante por parte del administrador**

Esta prueba consiste en la actualización o modificación de la matrícula de un estudiante por parte del administrador.

Tabla 3-92: Resultados de los Casos de Prueba: Actualizar la matrícula de un estudiante por parte del administrador

Caso de Prueba	Entrada	Resultado Obtenido
Agregar o quitar cursos o materiales de una matrícula de una	Matrícula registrada en el período académico con estado actual. Cursos y materiales registrados en la matrícula de un estudiante.	El sistema permitir la agregación o eliminación de cursos y/o materiales de una matrícula. Se genera un nuevo comprobante de matrícula como indicador de del éxito de la operación en el proceso de matrícula.
No seleccionar ningún curso	Ningún curso seleccionado	El sistema no permite el registro de la matrícula. Siempre debe estar seleccionado algún curso para proceder con el registro. Emite un mensaje indicando que se debe seleccionar algún curso.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

- **Eliminar la matrícula de un estudiante por parte del administrador**

Esta prueba consiste en la eliminación o anulación de la matrícula de un estudiante por parte del administrador**.**

Tabla 3-93: Resultados de los Casos de Prueba: Eliminar la matrícula de un estudiante por parte del administrador

Caso de Prueba	Entrada	Resultado Obtenido
Eliminación de la matrícula de un estudiante.	Matrícula registrada en el período académico con estado actual. Se selecciona a la matrícula del estudiante	El sistema emite un mensaje de confirmación solicitando al usuario que confirme la acción a realizar. Para indicar el éxito de la operación emite un mensaje.
Cancelación de la acción de eliminación de la matrícula de un estudiante.	El administrador procede con la eliminación de la matrícula de un estudiante y se le presenta el mensaje de confirmación.	El sistema al momento que se le presente el mensaje de confirmación con las opciones de aceptar o cancelar y cuando se selecciona la opción de Cancelar, el sistema cancela la acción de eliminación de la matrícula.

Fuente: LLUGSHA, William
Elaborado por: LLUGSHA, William

CAPITULO 4. CONCLUSIONES Y RECOMENDACIONES

Una vez finalizado este proyecto en el cuál he trabajado arduamente junto a la dirección de mi Director de Tesis, puedo sentirme satisfecho del resultado de este proyecto que servirá de soporte para las Instituciones de Educación Superior que buscan el fortalecimiento de la enseñanza aprendizaje.

Para concluir con el presente trabajo, he considerado importante emitir algunos comentarios, conclusiones, y recomendaciones basadas en esta experiencia, que esperamos sean de utilidad para futuras generaciones, las mismas las detallamos a continuación:

4.1 CONCLUSIONES

- La adopción de una plataforma de software libre tiene ventajas significativas. Al igual que en los desarrollos propios, la institución queda liberada de la dependencia cerrada respecto al vendedor, puesto que el software se distribuye con código abierto y con licencia de modificación libre. Las instituciones pueden aprovechar su capacidad técnica-informática con el objeto de integrar o generar modificaciones sobre el código original de un LMS a fin de adaptarlo a sus gustos y necesidades. Además esta opción significa un ahorro de recursos económicos debido a que los productos de software libre suelen ser distribuidos gratuitamente.

- El éxito que un LMS puede tener dependerá directamente del uso que le den los usuarios (profesores y estudiantes) a la plataforma y estar concientes que la educación con el apoyo de nuevas herramientas tecnológicas brindarán una educación más pertinente y de mejor calidad.

- Es preciso mencionar que con la implementación de estos nuevos módulos, este proyecto no pretende ser un Sistema Administrativo para una Institución de Educación Superior, sino ofrecer al usuario “herramientas en línea”, que apoye el proceso de enseñanza-aprendizaje

en programas de pregrado y postgrado de educación superior.

- La adopción de un LMS en una institución de Educación Superior no solamente puede servir de apoyo a las modalidades a distancia, sino también a las modalidades presencial y semipresencial debido a las herramientas de comunicación que un LMS puede ofrecer.

- El desarrollo de Dokeos está siendo conducido por una comunidad de usuarios cada vez más amplia y abierta a la participación, lo que ha dado lugar a una evolución del producto más rápida de lo previsto y al desarrollo de módulos y características adicionales en un período muy breve de tiempo.

- Como ciclo de desarrollo del proyecto se optó por utilizar la metodología de desarrollo Extreme Programming, debido a que es un proceso de desarrollo ligero centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Por otro lado, la metodología de desarrollo Dokeos es XP ya que promueve las prácticas inmersas en XP.

- Las herramientas utilizadas para el desarrollo del presente proyecto han sido en su mayoría open-source, economizando de está manera los costos de construcción y mantenimiento del sistema.

4.2 RECOMENDACIONES

- Escoger a Dokeos como entorno virtual de enseñanza/aprendizaje de una Institución de Educación Superior por su combinación de flexibilidad y sofisticación didáctica, su flexibilidad tecnológica, el dinamismo de su comunidad de desarrollo y por su facilidad de uso para estudiantes y profesores.

- Se recomienda estar pendiente de los nuevos releases o versiones liberadas por Dokeos, con el objeto de incrementar la funcionalidad general de la plataforma, así como también el continuo mantenimiento de éste

sistema, pues los requerimientos de una institución que adopte como plataforma de enseñanza/aprendizaje a Dokeos van cada día en aumento.

- Se recomienda el uso de software libre, pues la tendencia actual es la utilización de estas herramientas que a parte de economizar totalmente la construcción de un sistema, facilitan en gran manera la misma, a más de que permiten reutilizar el código y optimizarlo, dando con esto un buen mantenimiento al sistema que se haya construido.

REFERENCIAS BIBLIOGRÁFICAS

LIBROS

1. LARMAN, Craig. Applying UML and Patters. Segunda Edición. ISBN 0 -13-092569-1. 2002.
2. RATIONAL SOFTWARE. Rational Unified Process, Best Practices for Software Development Teams. 1998.
3. AGUILAR Ruth, Guía Didáctica – Postgrado de Educación a Distancia. Edición 1. Editorial UTPL. Loja-Ecuador, 2005

DIRECCIONES ELECTRÓNICAS

4. Dokeos, 2006
   http://www.dokeos.com
5. Moodle, 2006
   http://www.moodle.org
6. ATutor, 2006
   http://www.atutor.ca
7. Márquez Flórez Oscar W. “Una plataforma de tele-enseñanza de software libre”
   http://dotlrn.org/file-storage/view/madrid05/14.pdf
8. Javahispano, Procesos de desarrollo: RUP, XP y FDD. 2003
   http://www.javahispano.org/download/articulos/metodos_desarrollo.pdf
9. Edutools – Course Management Systems, 2005
   http://www.edutools.info
10. Selección de un entorno virtual de enseñanza/aprendizaje de código fuente abierto para la Universitat Jaume I. Centre d’Educació i Noves Tecnològies de la UJI, 2005
    http://cent.uji.es/doc/eveauji_es.pdf
11. Educaline, 2005
    http://www.educaline.com/index.htm
12. Mejoras de la Metodología del RUP para el desarrollo de aulas virtuales, 2005
    http://www.lisi.usb.ve/publicaciones/07%20integracion%20de%20sistemas/integracion_18.pdf
13. Online Education and Learning Management Systems, 2003
    http://www.studymentor.com/studymentor/
14. Desafíos y fundamentos de educación virtual, 2004
    http://www.ilustrados.com/publicaciones/EpZFEykElAxyPTGBhN.php

PROYECTOS DE TITULACIÓN

15. BAYAS, Diego, JARAMILLO, María. Características y uso de la Programación Extrema. 2002
16. MOSQUERA, Hilda, SEGOVIA Luis, El aula virtual: Teoría y aplicaciones. 2000
17. TACO, Marlon. Diseño de un prototipo de Aula Virtual basado en herramientas comerciales de Chat aplicado a la entrega de tutoriales previamente desarrollados. 2001

GLOSARIO

CMS: Course Management System: Es un sistema que integra servicios y herramientas para permitir y favorecer el trabajo entre alumnos y docentes. Parte fundamental en un LMS.

Content: La propiedad intelectual y el conocimiento a ser impartido. Diferentes formatos para los contenidos de e_Learning: texto, audio, video, animación y simulación.

Courseware: Cualquier tipo de curso educativo puesto sobre un programa de software o sobre la Web.

Distance education: La situación educativa en donde el instructor y los estudiantes están separados por el tiempo, lugar o ambos. Los cursos o materias son entregados a lugares remotos vía un medio sincrónico o asincrónico incluyendo correspondencia escrita, texto, gráficos, audio y video tape, CDROM, aprendizaje on line, audio y video conferencia, TV interactivo y fax. La educación a distancia no desecha el uso de la clase tradicional. La definición de educación a distancia es más amplia que, e incluye la definición de e_Learning.

E_Learning: Cubre una amplio juego de aplicaciones y procesos tales como aprendizaje basado en la Web, CBT, clases virtuales, colaboración digital. Incluye la entrega de contenido vía Internet, Intranet/extranet (LAN/WAN), audio y videotape. Satélite, TV interactiva y CD-ROM.

LMS: Learning Management System (=LCMS Learning Content Management System): Un LMS es el componente virtual de la educación tradicional, es un software que permite a los docentes y alumnos las funciones administrativas y académicas de la capacitación. Los alumnos y los profesores pueden comunicarse, transferir información, evaluar y ser evaluados, pagar, registrase, realizar aprendizaje cooperativo, emplear un chat especializado, cartelera de mensajes, emite informes y otros servicios.

SCORM: (Sharable Content Object Reference Model) Un conjunto de estándares

que cuando se aplican sobre los contenidos de un curso, produce pequeños, reusables objetos de aprendizaje. Como resultado de la iniciativa del departamento de Avanzado Aprendizaje distribuido de la defensa, los coursware que cumplimenten SCORM pueden ser fácilmente mezclados con otros elementos que también cumplimente SCORM, para producir un repositorio de materiales de educación modularizados.

JOIN: Proyecto europeo financiado por la iniciativa eLearning de la Comisión Europea, que evalúa la calidad de LMS open source para poder ofrecer información y apoyo a toda la comunidad que desee adoptar alguno de estos sistemas.