UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Componente de respaldo a cursos para la plataforma educativa Zera 2.0.

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Ulises Fernández-Nespral Vázquez

Tutor(es):

MSc. Yuniet del Carmen Toll Palma

Ing. Juan Manuel Veiga León

La Habana, junio de 2016

Declaración de Autoría

Declaración de Autoría

Declaro ser el autor del presente trabajo de diploma y otorgo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste se firma la presente a los ____ días del mes de _____ del año _____.

_____

Firma del Autor

Ulises Fernández-Nespral Vázquez

_____

Firma del Tutor

MSc. Yuniet del Carmen Toll Palma

_____

Firma del Tutor

Ing. Juan Manuel Veiga León

Pensamiento

«…La única lucha que se pierde es la que se abandona.’’»

Ernesto Che Guevara

Dedicatoria

Dedico este trabajo con todo mi corazón a mis padres, mi novia, a mis hermanos, y en especial a mi abuelita Luisa, a todos ellos por ser mi razón de ser y la causa por la que me levanto todos los días a dar lo mejor de mí.

Agradecimientos

En nuestra vida tenemos muchas personas que sin ellas nuestro existir no tendría sentido, a esas personas mis respetos y agradecimientos por ayudarme a alcanzar esta meta.

Primero agradecer a mi abuela Luisa por ser mi viejita del alma, por estar en mi vida desde siempre. Por soportar mis jaranas y responder con creces a ellas. Por no dejar que sus mil achaques le roben su sonrisa.

A mi mamá Yannis por aguantar mis malcriadeces, por estar en los momentos más difíciles de mi vida y ayudarme a salir de ellos, por darme todo su apoyo incondicional, gracias por tu amor, dedicación, paciencia, sin ti esto no hubiera sido posible, gracias mamá.

A mi papá Ulises por saber hacerme sentir seguro, por estar para escucharme y aconsejarme, por saber hacerme sonreír siempre, muchas gracias por existir.

A Lily por llegar en un momento importante de mi vida, para brindarme su más sincero amor y apoyo incondicional, por estar ahí para mí siempre que la necesito, por ser mi compañera y amiga, mi confidente, por corregir mis equivocaciones y escuchar mis problemas, por compartir mis tristezas y alegrías, por hacerme sentir especial y por traer felicidad a mi vida, te amo.

A mi padrastro Fabricio, por ser mi otro papá, por toda la ayuda que me ha dado en cada momento que lo necesitaba, por querernos tanto, por tener tanta paciencia con mi mamá, gracias por entrar en nuestras vidas.

A mi tía la flaka, por todo su apoyo, por traer felicidad a mi papá y regalarme un hermano.

A mis hermanos Jesica, Ian Carlos, Jesús y Hugo, por todos los momentos vividos y los que estamos por vivir.

A Liset y a Yadian gracias por dejarme entrar a su hogar y hacerme sentir otro miembro más de este, gracias.

A mi prima Diliam y mi tío Basulto, por apoyarme y brindarme todo su apoyo cuando más lo necesité.

A toda mi familia por las enseñanzas brindadas y por la confianza en mí.

Agradecimientos

Y a mis amigos Silverio, Neobel, Yadiel, Lugo, Luis Miguel, Rosmelys, a todas gracias por ser parte de mi vida, por todos los momentos de risas y tensiones, gracias por dejarme ser su amigo.

A mis tutores Yuniet y Juan Manuel por apoyarme y confiar en mí.

A el profe Piña por ser un tutor más y tener una increíble paciencia conmigo.

A la Revolución y a Fidel por darme la oportunidad de formarme como un profesional.

A todos los que de una forma u otra aportaron su granito de arena, para que hoy esto fuera realidad, gracias a todos.

Resumen

El respaldo de información constituye un proceso fundamental en todo sistema informático, ya sea en una gran empresa como en un ordenador de un hogar. Su utilización brinda un mecanismo de seguridad ante eventuales pérdidas de datos y posibilita la reutilización de información. El objetivo del presente trabajo es desarrollar un componente informático que permita respaldar información referente a los cursos virtuales en la plataforma educativa Zera 2.0. Como parte de las tareas realizadas, se revisó el estado del arte referente a los sistemas de respaldos de información utilizados en plataformas educativas, con el fin de identificar los procedimientos y métodos de empaquetamientos utilizados. Se realizó la selección de las tecnologías y herramientas necesarias para el desarrollo de la propuesta de solución. En correspondencia con la metodología de desarrollo de software AUP, en su versión para la Universidad de Ciencias Informáticas, se transitó por los diferentes flujos de trabajo y se generaron los artefactos correspondientes a cada una de las etapas. Para la validación del sistema se realizaron las pruebas de software necesarias para certificar la calidad del software.

Palabras clave: respaldo de información, cursos virtuales, plataforma educativa..

Índice

CONTENIDOS

Índice

Índice

Introducción

El uso de las tecnologías de la información y las comunicaciones (TIC) es cada vez mayor en la actualidad. Su aplicación en prácticamente todos los campos de la sociedad es posible, y es la educación una de las ramas más favorecidas con el desarrollo de la misma.

El sector educativo ha encontrado en las TIC un excelente medio para romper con las limitantes geográficas y temporales que los esquemas tradicionales de enseñanza-aprendizaje conllevan, revoluciona, y cambia a la vez, el concepto de educación a distancia. Su adopción y uso han sido amplios, lo que permite un desarrollo rápido y consistente en el que la web toma distintas formas dentro de los procesos educativos (Peñalvo, 2005).

La web se convierte en la infraestructura básica para desarrollar los procesos de enseñanza-aprendizaje no presenciales, combina servicios síncronos y asíncronos, lo que ha dado lugar a un modelo conocido como e-formación o e-learning, cada vez más valorado, no como sustituto de la formación presencial tradicional, sino como un complemento que se ha de adaptar según las necesidades y nivel de madurez del público receptor de esta formación (García y García, 2001).

Este nuevo concepto educativo es una revolucionaria modalidad de capacitación, que hoy se posiciona como la forma de enseñanza predominante en el futuro. Este sistema ha transformado la educación, porque abre puertas al aprendizaje individual y organizacional. Es por ello que hoy en día ocupa un lugar cada vez más destacado y reconocido dentro de las organizaciones empresariales y educativas. (e-ABC, 2011).

El proceso de formación debe ocurrir en un contexto, en el e-learning este se materializa en los entornos virtuales de aprendizaje o sistemas de gestión de aprendizaje (Learning Managment System, LMS).

Los LMS son sistemas que automatizan la administración de acciones de formación con el objetivo de permitir que el aprendizaje llegue a cualquier lugar y momento. (Cañellas,2011). Son aplicaciones web que integran un conjunto de herramientas y funcionalidades para el proceso de enseñanza-aprendizaje lo que permite una enseñanza no presencial (Lizárraga et al., 2013).

Introducción

Hoy en día existen en Internet una amplia gama de plataformas virtuales para crear cursos en línea. Se diferencian entre sí por el precio de las licencias de uso, la disponibilidad de recursos que ofrecen, tanto al gestor de los cursos como a los estudiantes, los requerimientos tecnológicos para su instalación y mantenimiento, entre otros (Rivero y Fernández,2014). Entre las privativas se encuentra BlackBoard y bajo licencia libre o código abierto se pueden encontrar Moodle, Claroline, Dokeos, Sakai, Ilias, Atutor, entre otras.

Según Lucía Rosario Malbernat la elección de la plataforma y de los medios a utilizar debe estar determinada por las características del modelo educativo subyacente al proyecto académico y por los requerimientos específicos de cada actividad. Pero para la evaluación de la calidad de la plataforma se debe observar una gran variedad de factores.

Entre los factores que señala la autora se encuentran los asociados a consideraciones técnicas, específicamente a las funciones de soporte técnicos relacionadas con la seguridad y protección de los datos, con las posibilidades de corrección de errores, de hacer **copias de seguridad** (backups) generales y particulares de manera centralizada y descentralizada, la posibilidad de migrar los contenidos, estructura o usuarios de un curso a otros en la misma plataforma o en otras instancias de la misma. Incluso, la capacidad de interoperabilidad con otras plataformas de modo que se pueda compartir contenidos con otras organizaciones, aunque utilicen plataformas diferentes (Malbernat, 2008).

El proceso de informatización en Cuba ha traído avances significativos a la educación, que van desde la incorporación de laboratorios de computación en las escuelas y los joven clubs de computación hasta el uso de sistemas de gestión de aprendizaje (LMS por sus siglas en inglés).

En Cuba existen múltiples experiencias en la utilización de plataformas de aprendizajes aplicadas al proceso docente, pero ha sido Moodle la más popular por sus bondades tecnológicas y pedagógicas. Muchas instituciones universitarias cubanas disponen de una plataforma interactiva Moodle como son la Universidad de la Habana con Moodle de la Universidad de La Habana (1) la Universidad de Pinar del Rio con Plataforma de Teleformación de la Universidad de Pinar del Río "Hermanos Saíz Montes de Oca"

para los procesos de superación profesional y académica (2), la Universidad Central de las Villas, la Universidad de Cienfuegos con su plataforma Moodle Pregrado. Universidad de Cienfuegos (3), la Universidad de Oriente entre otras (Rivero y Fernández,2014). [1]

Otra de las universidades que hace uso de las plataformas educativas es la Universidad de Ciencias Informáticas (UCI). La UCI cuenta con una infraestructura productiva donde se encuentra insertado el Centro de Tecnologías para la Formación (FORTES), el cual está especializado en la producción de aplicaciones y servicios orientados al sector educacional en todos los niveles curriculares. Entre los productos que ofrece en la cartera comercial del centro se encuentran la herramienta de autor web de objetos de aprendizaje Xauce CRODA, el repositorio de objetos de aprendizaje Xauxe RHODA, la plataforma educativa NAVIGO, servicios asociados a la plataforma Moodle y la plataforma educativa Xauce ZERA 1.0.

En la actualidad se desarrolla la plataforma educativa Xauce ZERA 2.0 para el Centro Nacional de Educación a Distancia (CENED). La misma pretende brindar a los usuarios la posibilidad de crear, administrar, almacenar, distribuir y gestionar las actividades de formación virtual, así como el seguimiento y control de dichas actividades; realizadas dentro de una plataforma altamente configurable e implementada con tecnologías libres y actuales. Sin embargo, el sistema no cuenta con un mecanismo que permita a los usuarios realizar respaldos personalizados de la información referente a los cursos y sus contenidos para su posterior restauración en otra instancia de la plataforma educativa Zera 2.0 o como mecanismo de seguridad ante eventuales pérdidas de información.

Luego de analizar la problemática anteriormente descrita se identifica como **problema de la investigación:** ¿Cómo lograr el respaldo de información de los cursos y sus contenidos en la plataforma educativa Zera en su versión 2.0?

---

[1] http://moodle.uh.cu

[2] http://moodle.ceces.upr.edu.cu

3 http://moodlepregrado.ucf.edu.cu

Introducción

La investigación en curso enmarca como **objeto de estudio:** el proceso de respaldo de información en plataformas educativas, delimita como **campo de acción:** la gestión de los respaldos de información a cursos en la plataforma educativa Zera 2.0.

Para dar solución al problema de la investigación se define como **objetivo general**: desarrollar un componente que permita realizar el respaldo de información a los cursos y sus contenidos en la plataforma educativa Zera en su versión 2.0

Se desglosan del objetivo general los siguientes **objetivos específicos:**

1. Realizar investigación sobre las definiciones, características principales, ventajas de la realización de los respaldos de información a cursos en las plataformas educativas existentes y en Zera 2.0.
2. Realizar el análisis y el diseño del componente para respaldo de información a cursos en la plataforma educativa Zera 2.0.
3. Implementar el componente para respaldo de información a cursos en la plataforma educativa Zera 2.0.
4. Validar a través de pruebas de software el funcionamiento del componente para respaldo de información a cursos en la plataforma educativa Zera 2.0.

Para dar cumplimiento al objetivo y resolver la problemática descrita se proponen las siguientes **tareas de la investigación.**

- ✓ Investigación de los procedimientos utilizados en LMS para el respaldo de información a cursos.
- ✓ Determinación de las principales funcionalidades del componente para respaldo de información a cursos en la plataforma educativa Zera 2.0.
- ✓ Selección de las herramientas, tecnologías y metodología a utilizar para el desarrollo de la solución.
- ✓ Modelado del modelo de dominio de la investigación.
- ✓ Modelado de los diagramas de Clases del Análisis correspondientes al componente para respaldo de información a cursos en la plataforma educativa Zera 2.0.
- ✓ Modelado de los diagramas de Clases del Diseño correspondientes al componente para respaldo de información a cursos en la plataforma educativa Zera 2.0.

- ✓ Implementación de las funcionalidades correspondientes al componente para respaldo de información a cursos en la plataforma educativa Zera2.0.
- ✓ Selección de los tipos de pruebas para garantizar la calidad de la solución.
- ✓ Validación de la solución a través de pruebas de software.

Métodos científicos de investigación

El desarrollo de la investigación requirió el empleo de los métodos teóricos y empíricos que se relacionan a continuación:

Métodos científicos de nivel teórico

- ✓ **Método de análisis y síntesis:** para analizar la teoría y descomponer mentalmente la problemática objeto de investigación en pequeñas partes, lo que permitió comprender el proceso de respaldo de información.
- ✓ **Modelación:** se utilizó para la elaboración de los artefactos generados durante el análisis y el diseño de la solución propuesta.

Métodos científicos de nivel empírico

- ✓ **Observación:** se utilizó para identificar buenas prácticas y vulnerabilidades de los sistemas similares.

Introducción

Descripción Capitular

Capítulo 1: Fundamentación Teórica.

En este capítulo se realiza un análisis de los principales conceptos asociados al dominio del problema, se efectúa un estudio de las soluciones similares y se caracterizan las herramientas, las tecnologías, la metodología y los lenguajes de programación empleados.

Capítulo 2: Descripción de la solución propuesta.

En este capítulo se hace una descripción de la propuesta de solución, se describe el modelo de dominio, y los conceptos asociados al mismo, se definen los requisitos funcionales y no funcionales. También se realiza el análisis y diseño de la solución propuesta y se generan los artefactos asociados a la metodología de desarrollo seleccionada.

Capítulo 3: Implementación y validación de la solución.

En este capítulo se describen aspectos relacionados con la implementación, refleja el empleo de buenas prácticas de programación y estándares de codificación. Además, se seleccionan los tipos de pruebas, se diseñan los casos de prueba y se analizan los resultados de la aplicación de dichas pruebas como parte de la validación de la solución.

Capítulo 1. Fundamentación teórica

1.1 Introducción

En el presente capítulo se introducen una serie de conceptos con el objetivo de facilitar la comprensión de los temas relacionados con la conceptualización de los LMS, cursos virtuales y los respaldos de información. Además, se realiza un estudio sobre sistemas de respaldos a cursos existentes en plataformas educativas, con el objetivo de comprender el funcionamiento de los mismos; además, se hace selección de las metodologías, lenguajes de programación y tecnologías a utilizar en el desarrollo de la propuesta de solución.

1.2 Conceptos asociados al dominio del problema

1.2.1 Sistemas de gestión de aprendizaje

Son muchas las definiciones de sistemas de gestión de aprendizaje tal es el caso de Jaime Sánchez que plantea que son "…cualquier programa computacional cuyas características estructurales y funcionales sirvan de apoyo al proceso de enseñar, aprender y administrar" (Sanchéz,1999).

Según Pérez Márquez, puntualiza la definición de LMS como, "...programas para ordenadores creados con la finalidad específica de ser utilizados como medio didáctico, es decir, para facilitar los procesos de enseñanza y aprendizaje (Márques, 1999).

Por otro lado, el autor Joseph Boneu plantea que "los sistemas de gestión del aprendizaje, son aplicaciones web que proveen las funciones administrativas y de seguimiento necesarias para posibilitar y controlar el acceso a los contenidos, implementar recursos de comunicaciones y llevar a cabo el seguimiento de quienes utilizan el sistema. Las plataformas educativas ofrecen a sus usuarios disímiles funcionalidades a través del uso de herramientas, entre ellas: las orientadas al aprendizaje, a la productividad, las herramientas para la implicación de los estudiantes, las de soporte, las destinadas a la publicación de cursos y recursos, los sistemas para la gestión del conocimiento en el ámbito educativo y los componentes para el diseño de planes de estudio. En general los LMS facilitan la interacción entre los docentes y los estudiantes, aportan herramientas para la gestión de los contenidos académicos y permiten el seguimiento y la evaluación (Boneu, 2007).

Se puede concluir que los sistemas de gestión de aprendizaje son programas computacionales encaminados a favorecer los procesos de enseñanza aprendizaje, para ello utilizan un conjunto de herramientas y funcionalidades que permiten la administración de cursos y sus contenidos, el seguimiento y evaluación de los usuarios en el sistema, la comunicación síncrona y asíncrona entre los docentes y estudiantes. En la presente investigación, el autor considera la definición de Boneu como la más completa y acorde al tema de investigación por lo que se adjudica a su enunciación de sistemas de gestión de aprendizaje.

1.2.2 Cursos en línea

Según Ivette un curso en línea se puede definir como una modalidad educativa que emplea el uso de las tecnologías de información y comunicación como el medio a través del cual se produce el aprendizaje, de tal forma que emplea recursos de comunicación síncronos y asíncronos (Ivette,2011).

De acuerdo con Sandìa, Montilva y Barrios, un curso en línea se fundamenta en el proceso de enseñanza-aprendizaje de tipo de interactivo, en el que el estudiante puede interactuar y comunicarse con el profesor y compañeros del curso, así como acceder al contenido mediante charlas, debates, exámenes, lecciones, entre otros, todo esto través de la web (Sandìa, Montilva y Barrios, 2005).

A partir de las definiciones anteriores, el autor de la presente investigación considera un curso en línea como, el conjunto de materiales con el propósito de impartir una materia en particular, que se fundamentan en el proceso de enseñanza-aprendizaje interactivo, donde el acceso a los contenidos y actividades se realiza a través de la web. Además, los cursos son el contexto donde ocurren la mayor parte de las actividades del proceso enseñanza aprendizaje dentro de los LMS.

1.2.3 La pérdida de datos

Dentro del desarrollo tecnológico en la actualidad, la información se ha convertido en el elemento más importante de cualquier persona, empresa, institución o gobierno. Su disponibilidad es una de las características fundamentales para la subsistencia del medio que la utilice, y su pérdida puede ocasionar grandes problemas, de los cuales es muy difícil recuperarse (Cruz,2008).

Debido a la gran cantidad de información con la que las personas interactúan hoy día, las copias de seguridad han pasado a un plano indispensable para el buen funcionamiento de cualquier tipo de

empresa, sin tener en cuenta sus dimensiones o perfil de negocio; la capacidad de reanudar sus servicios lo antes posible ante cualquier situación de pérdida o corrupción de la información es de vital importancia. El 40 % de las compañías que experimentan un desastre de gran magnitud quedan fuera del negocio si no pueden tener acceso a su información en 24 horas (Sánchez, 2003).

Existen múltiples factores que pueden influir en la pérdida de información, en este sentido se puede citar por ejemplo durante el 12 de julio del 2006 una universidad del sur de Chile se vio afectada un desborde del río Biobío, que además de verse obligado a cancelar sus actividades, se enfrentó al problema de perder todos los datos de un servidor, producto de la completa inundación (Radio Cooperativa, 2006). Los desastres naturales son uno de los factores, sin embargo, no son estos la principal causa de pérdida de información. Según el artículo ¨Las nuevas tendencias en respaldo y recuperación de datos¨ el 56% de los casos se atribuye a errores de hardware, el 26 % a los errores humanos, 9 % para la corrupción del software, 4% por virus y deja solamente un 2 % para catástrofes naturales naturaleza (Abad, 2005).

Ante esta situación las organizaciones adoptan políticas y estrategias que tributen a la seguridad de los datos; entre las acciones que se pueden citar están las copias de seguridad o backups. Las copias de seguridad en los sistemas informáticos tienen por objetivo mantener cierta capacidad de recuperación de la información ante posibles pérdidas o desastres.

1.2.4 Copias de seguridad

Existen numerosas definiciones acerca de lo que es el respaldo de información o copias de seguridad. A continuación, se presentan algunos de dichos conceptos:

Copia de ficheros o datos de forma que estén disponibles en caso de que un fallo produzca la pérdida de los originales. Esta sencilla acción evita numerosos, y a veces irremediables problemas si se realiza de forma habitual y periódica (Definicion.org, 2008).

Hacer una copia de seguridad o copia de respaldo (backup en inglés, el uso de este anglicismo está ampliamente extendido) se refiere a la copia de datos de tal forma que estas copias adicionales puedan restaurar un sistema después de una pérdida de información (Wikipedia, 2016).

Capítulo 1. Fundamentación teórica

Copia de seguridad es la actividad de copiar archivos o bases de datos a fin de que puedan ser conservadas en caso de fallo del equipo o de otra catástrofe. La copia de seguridad suele ser una parte rutinaria de la operación de grandes empresas con súper computadoras, así como los administradores de las empresas más pequeñas con menos recursos. Para los usuarios de computadoras personales, la copia de seguridad también es necesaria pero no se realiza muy a menudo. La recuperación de los archivos de la copia de seguridad se denomina: restablecimiento (Storage Technology Inforamation, 2016)

Archivo de copia de seguridad: copia de un archivo para efectos de la posterior reconstrucción del mismo, en caso de ser necesario. Nota: Un archivo de copia de seguridad puede utilizarse para preservar la integridad del archivo original y puede ser registrado por cualquier medio adecuado (Glossary, 1996).

El respaldo de información es la copia de los datos importantes de un dispositivo primario en uno o varios dispositivos secundarios, ello para que en caso de que el primer dispositivo sufra una avería electromecánica o un error en su estructura lógica, sea posible contar con la mayor parte de la información necesaria para continuar con las actividades rutinarias y evitar pérdida generalizada de datos (Informática moderna, 2007).

Copia de ficheros o datos de forma que estén disponibles en caso de que un fallo produzca la pérdida de los originales. Esta sencilla acción evita numerosos, y a veces irremediables, problemas si se realiza de forma habitual y periódica (Básico y Fácil, 2008).

A partir de las definiciones anteriores y que el autor toma como consideración que los conceptos backups, respaldo de información y copia de seguridad tiene un mismo objetivo, el autor de la presente investigación hará uso del término respaldo de información en alusión la copia parcial o total de información de manera que esta pueda ser restaurada ante una eventual pérdida de información.

En los sistemas de gestión de aprendizajes la utilización de respaldos de información brinda múltiples beneficios. El contar con un respaldo de un curso o parte de este, brinda la posibilidad de **reutilizar** dicho material en otra instancia del mismo LMS, u otro, con soporte para el mismo formato de respaldo. Si un curso es dañado por una manipulación indebida u otra causa, los administradores del sistema pueden restablecer el curso afectado a través del respaldo, de esta forma se mantiene la **accesibilidad** del mismo

para los usuarios. Además, los respaldos a cursos contribuyen a **reducir costos** en tiempo y esfuerzos a la institución que utiliza el LMS.

Para los LMS la disponibilidad y durabilidad de los cursos es fundamental ya que estos constituyen un eslabón imprescindible para el desarrollo de su actividad, de ahí que estos sistemas incorporen entre sus funcionalidades la realización de copias de seguridad a los cursos y los contenidos de los mismos.

1.3 Análisis de sistemas similares

En la actualidad existe un gran número de LMS alrededor de todo el mundo, entre las más conocidas están Claroline, Dokeos, Sakai, Moodle y Blackboard. A continuación, se realiza un análisis de las principales características y funcionalidades de algunos de estos sistemas.

1.3.1 Claroline

Claroline es una plataforma de aprendizaje que permite a los profesores construir cursos en línea y gestionar las actividades de aprendizaje y colaboración en la web. El proyecto Claroline fue iniciado en el año 2000, y tiene una gran comunidad de desarrolladores y usuarios en todo el mundo.

Claroline se distribuye bajo licencia libre. Está escrito en el lenguaje de programación PHP, utiliza MySQL como SGBD (Sistema Gestor de Base de Datos). Está disponible para plataformas Linux, Unix, Mac OS X y Windows (Álvarez, 2010).

Entre las funcionalidades que brinda Claroline están:

- Publicar recursos en diferentes formatos de archivo: word, pdf, html, vídeo, etc.
- Foros de discusión tanto, privados como públicos.
- Administrar listas de enlaces.
- Crear grupos de estudiantes.
- Confeccionar ejercicios.
- Agenda con tareas, plazos y calendario donde mostrar tareas y anuncios
- Hacer anuncios. Vía correo electrónico o en la portada de los cursos.
- Gestionar los envíos de los estudiantes: documentos, tareas, trabajos, etc.
- Crear y guardar chats.

- Supervisar el acceso y la progresión de los usuarios
- Agrupación de contenidos en temas o módulos.
- Uso de cursos SCORM.
- Soporte para contenido IMS
- Elaboración de test y listados de preguntas.
- Gestión de estadísticas de cursos y del sitio general.
- Configuración y seguimiento de itinerarios dentro de los cursos.

Además, una de las características que más destaca de Claroline es su sencilla interfaz, Claroline está diseñada de forma tal que sea fácil de usar por cualquier persona desde el primer momento, esto se constata con una interfaz pulcra y minimalista que permite a los usuarios noveles usar la plataforma sin complicaciones. También, Claroline cuenta con una excelente capacidad de extensión debido a que puede ampliar sus funcionalidades a través de módulos o plugins,

Los módulos aportan al sistema una serie de ventajas:

- Desacoplar el núcleo de las herramientas: permite que el núcleo y las herramientas sean desarrollados por separado. De esta forma se simplifica su desarrollo.
- Permite distribuciones a la carta: permite construir instalaciones con las herramientas necesarias para cada caso concreto.
- Simplifica las actualizaciones: permite que el núcleo y las herramientas se actualicen y corrijan independientemente, elimina así la necesidad de esperar para actualizaciones mayores.
- Incrementar la funcionalidad del sistema: por medio de herramientas creadas por la comunidad se puede aumentar la funcionalidad inicial de la plataforma.

Durante el desarrollo de la presente investigación no se encontró referencia que la plataforma Claroline contara con algún componente para el respaldo personalizado sobre los cursos. El mecanismo utilizado por los usuarios de Claroline para preservar la información de la plataforma y los cursos consiste en hacer un respaldo de la base de datos y copiar los directorios que contienen los ficheros de los cursos.

1.3.2 Dokeos

Dokeos es una suite de aprendizaje en línea basada en software libre. Provee todas las características que una aplicación de aprendizaje en línea necesita, desde la autoría de cursos hasta informes. Dokeos nació en 2004 a partir de Claroline. Para acometer este proyecto creó la empresa del mismo nombre, Dokeos, esta empresa se encarga del desarrollo de Dokeos y ofrece hospedaje, consultoría y soporte para empresas e instituciones. Inicialmente era una versión modificada de Claroline pero actualmente es una distribución independiente usada en 2010 por más de 9000 instituciones y organizaciones. Dokeos al igual que Claroline, aún es una aplicación libre, aunque algunas de las herramientas que puede incluir no lo son, esto condiciona que existan distribuciones libres y propietarias (Álvarez, 2010).

Las funcionalidades que brinda Dokeos en su versión libre son:

- Crear plantillas de contenido, con explicación de diagramas, vídeo, flash…
- Dinamizar las páginas
- Crear test: de respuesta múltiple, preguntas abiertas…
- Incorporar contenido SCORM
- Crear contenido SCORM
- Gestionar tutoriales
- Interacción: grupos, chat, foros…
- Crear y organizar encuestas
- Visualizar informes acerca del progreso de los alumnos
- Extender la comunidad a través de libros de notas, Wiki…

El sistema de extensiones de Dokeos es altamente manual y carece de una herramienta de administración e instalación dentro de la plataforma. Además, la instalación de muchas de las herramientas implica modificar ficheros del núcleo de la aplicación, esto va en contra de la ideología de una aplicación con extensiones (Álvarez, 2010).

Dokeos cuenta con un menú de configuraciones donde se muestran un grupo de acciones que se pueden realizar sobre un curso. Entre estas opciones se encuentran:

- ✓ Reciclar curso
- ✓ Copiar total o parcialmente el curso
- ✓ Copia de seguridad
- ✓ Importar curso

La opción reciclar curso brinda la posibilidad de vaciar de todos los componentes que contiene el curso seleccionado como documentos, foros, enlaces, etc. Este proceso puede ser total o personalizado. La opción copiar curso brinda la posibilidad de duplicar un curso o parte de este en el sistema para ser reutilizado. La opción copia de seguridad ofrece la posibilidad de respaldar totalmente o de forma personalizada los componentes de un curso, estas copias de seguridad están comprimidas en un fichero (zip) que puede ser descargado a una ubicación deseada por el usuario. La opción importar copia de seguridad brinda la posibilidad de seleccionar el fichero (zip) contenedor del respaldo, ya sea un fichero en el servidor o uno en una ubicación especificada por el usuario. Estos ficheros pueden ser utilizados para restaurar la información contenida en ellos en un curso vacío; y el usuario puede seleccionar si desea restaurar todo el contenido o componentes específicos de estos. Las características que brindan las funcionalidades copias de seguridad e importar curso, serán consideradas en el desarrollo del componente de respaldo a cursos para la plataforma educativa Zera 2.0.

1.3.3 Sakai

El Proyecto Sakai tiene su origen en la Universidad de Michigan y en la Universidad de Indiana, a las que se unieron el MIT y Stanford University, junto a la Iniciativa de Conocimiento Abierto (OKI) y el consorcio uPortal. En enero de 2004 comenzó la iniciativa para integrar las funcionalidades de un entorno virtual de enseñanza/aprendizaje en un portal institucional. El proyecto de Sakai, Collaboration and Learning Environment (CLE), es un entorno modular de código fuente abierto, cuyo objetivo es integrar diversas funcionalidades del e-learning en un portal académico. Para gestionar el Proyecto se ha creado la Fundación Sakai, a la que pertenecen más de 100 Universidades (Sakai 2010).

Sakai es una herramienta 100% software libre, desarrollada en java y que normalmente se distribuye en forma de binarios, archivos listos para su despliegue y puesta en marcha, o en forma de código fuente, código que es necesario compilar para poder usarlo.

Sakai divide sus funcionalidades en cuatro categorías de herramientas (Sakai 2010):

- Herramientas generales de colaboración:
- Herramientas de enseñanza y aprendizaje:
- Herramientas administrativas
- Herramientas de portafolios

El diseño de capas de Sakai permite que se hagan extensiones de cualquier parte de la aplicación. Esto permite que existan extensiones con cualquier funcionalidad, pero dificulta la creación de las mismas debido a que hay que conocer la arquitectura completa de la aplicación y no sólo las clases necesarias para crear extensiones (Álvarez, 2010).

Durante el transcurso de la presente investigación no se encontró referencia sobre algún componente que diera la posibilidad de realizar copias de seguridad sobre los cursos, para salvaguardar la información en la plataforma educativa Sakai es necesario realizar copias de seguridad a la base de datos del sistema.

1.3.4 Chamilo

Chamilo es una solución de software libre, licenciada bajo la GNU/GPLv3, de gestión del E-learning o aprendizaje electrónico, desarrollada con el objetivo de mejorar el acceso a la educación y el conocimiento globalmente. Está sustentado por la Asociación Chamilo (asociación sin fines de lucro), la cual tiene como objetivo la promoción del software para la educación (y en particular de Chamilo), el mantenimiento de un canal de comunicación claro y la construcción de una red de proveedores de servicios y contribuidores al software. El proyecto Chamilo intenta asegurar la disponibilidad y la calidad de la educación a un costo reducido a través de la distribución gratuita y abierta de su software, la adaptación de su interfaz a dispositivos de países del Tercer mundo2 y provisión de un campus e-learning de acceso libre. Chamilo sostiene dos proyectos de software: Chamilo LMS (llamado anteriormente Chamilo 1.8), una versión que fue, en sus inicios, basada en el software Dokeos, y Chamilo LCMS Connect (previamente Chamilo 2), una reimplementación completa de la plataforma para el e-learning y la colaboración. Chamilo está desarrollado principalmente en PHP y depende de un sistema LAMP o WAMP en el servidor. Del lado cliente, solo requiere un navegador moderno (menos de 3 años de antigüedad) y, de manera opcional, requiere el plugin Flash para hacer uso de algunas funcionalidades avanzadas (Álvarez, 2010).

Entre las funcionalidades que se encuentran en Chamilo están:

- ✓ gestión de cursos, usuarios y ciclos formativos (incluyen servicios web en SOAP para gestión remota)
- ✓ compatibilidad con SCORM 1.2 y herramientas de autoría rápida
- ✓ modo multi-instituciones (con portal de gestión centralizado)
- ✓ exámenes controlados por tiempo
- ✓ internacionalización con UTF-8
- ✓ zonas horarias
- ✓ generación automática de certificados
- ✓ seguimiento del progreso de los usuarios
- ✓ red social incorporada

Chamilo cuenta con un menú de configuraciones sobre los cursos donde se pueden encontrar las funcionalidades:

- ✓ Reciclar curso
- ✓ Copiar total o parcialmente el curso
- ✓ Copia de seguridad

Al igual que en Dokeos el fichero contenedor del respaldo es un archivo (zip), el cual contiene una estructura de carpetas con los componentes que integran el curso, un archivo llamado **document** que contiene la información de las entidades. En Chamilo la información referente a los usuarios no es respaldada por lo que no se puede restaurar.

1.3.5 Moodle

Moodle es un paquete de software para la creación de cursos de e-learning. Es un proyecto en desarrollo, diseñado para dar soporte a un marco de educación social constructivista. La primera versión de la herramienta apareció el 20 de agosto de 2002, actualmente Moodle cuenta con más de 37 millones de usuarios en casi 50.000 sitios registrados en su base de datos. Moodle está desarrollado en PHP y, al igual que Dokeos o Claroline, necesita una plataforma que cuente con un sistema gestor de bases de

datos y un servidor web. A diferencia de las otras plataformas Moodle ha sido desarrollado para lograr la portabilidad, por lo que soporta los sistemas de base de datos más importantes: PostgreSQL, MySQL, SQL Server, Oracle SQL, entre otros (Álvarez, 2010).

Moodle es una plataforma de enseñanza virtual modular, todas sus funcionalidades se encuentran en módulos que es posible incorporar al sistema, por este motivo una de las mejores formas de ver sus características es ver los módulos principales del sistema. Moodle divide sus extensiones en tres grandes categorías:

- Módulos de actividades: Son los correspondientes a las actividades y los recursos que se pueden incluir en los cursos.

- Bloques: Los bloques son los elementos modulares que forman parte de la estructura tabular de Moodle, los bloques se muestran en los laterales de la página.

- Filtros: Son aplicaciones que analizan el texto que se introduce en las actividades y en los recursos y aplica filtros que modifican el resultado final.

Moodle ofrece una amplia cantidad de maneras de personalizar el sitio, por este motivo actualmente en su repositorio se pueden encontrar más de 600 extensiones y continuamente se incluyen nuevas o se actualizan las existentes. Otro factor que ayuda a que las capacidades de extensión de Moodle sean ideales, es la amplia documentación que se puede encontrar en su página web y que la instalación de todos los módulos es muy simple.

Moodle ofrece a los usuarios con rol manager o profesor realizar respaldos de información a los cursos de forma total o personalizada. El respaldo consiste en un fichero (mbz) con los datos de las entidades correspondientes a la selección del tipo de respaldo. Este fichero puede ser utilizado en la restauración de un curso nuevo o como contenido de otro. Los respaldos en Moodle pueden incluir información referente a los usuarios si así se selecciona en el menú de opciones de respaldo. En el desarrollo del componente de respaldo a cursos para la plataforma educativa Zera 2.0 se tendrá en cuenta el carácter modular de sus componentes, lo que permitirá su extensión a otros elementos de los cursos y la plataforma.

1.3.6 BlackBoard

Es una plataforma que integra un ambiente sólido de enseñanza y aprendizaje en línea. Se caracteriza por administrar un conjunto de recursos que permiten desarrollar cursos virtuales, específicamente: impartir y distribuir contenidos que se encuentran presentados en diversos formatos, realizar evaluaciones en línea, llevar a cabo el seguimiento académico de los estudiantes participantes, asignar tareas, desarrollar actividades en ambientes colaborativos y, además, es un software bajo licencia propietaria. Hasta el 2005, Blackboard desarrolló y licenció aplicaciones de programas empresariales y servicios relacionados a más de 2200 instituciones educativas en más de 60 países. Estas instituciones usan el programa de Blackboard para administrar aprendizaje en línea (e-learning), procesamiento de transacciones, comercio electrónico (e-commerce), y manejo de comunidades en línea (online). Las capacidades de la plataforma se agrupan en tres categorías fundamentales:

- Enseñanza, comunicación y evaluación.
- Conexiones, personalización y comercio electrónico.
- Recopilar, compartir y descubrir

Dentro de estas categorías se encuentran capacidades como es la gestión de cursos, creación de contenidos, desarrollador del plan de estudios, gestión de archivos y contenidos, entre otros (BlackBoard, 2016).

El estudio realizado anteriormente ha servido para identificar los elementos fundamentales que se gestionan en las plataformas educativas. Estos elementos serán la base de las funcionalidades de la propuesta de solución del componente de respaldo a curso para la plataforma educativa Zera 2.0. A continuación algunas características del sistema.

Algunas de las características del componente para respaldo a curso para la plataforma educativa Zera 2.0 a tener en cuenta son:

- El componente a implementar será modular, lo que posibilita su crecimiento a otros elementos de los cursos.
- Los elementos que podrán ser respaldados son los recursos y las actividades (Fórum, tareas, cuestionarios de ejercicios) contenidas en un curso.

- El respaldo consistirá en un fichero comprimido (zip) el cuál contendrá un fichero de texto con los datos de los elementos respaldados y los recursos multimedia contenidos en recursos y actividades
- El componente permitirá la restauración de un fichero previamente respaldado en la misma u otra instancia de la plataforma Zera 2.0
- Se podrán importar y descargar respaldos realizados en la instancia de la plataforma Zera 2.0 u otra instancia de la misma.

1.4 Herramientas, tecnologías, lenguajes y metodología de desarrollo de software

Para el desarrollo de la presente investigación se realizó el análisis de diferentes herramientas, tecnologías, lenguajes de programación y metodologías de desarrollo de software, con el objetivo de lograr el desarrollo del componente para respaldo de información a cursos en la plataforma educativa Zera 2.0. Para la selección de las herramientas de software utilizadas se tuvo en cuenta el uso de las mismas a nivel internacional, la presencia de bibliografía actualizada, el dominio sobre las herramientas por el autor de la investigación y el carácter libre de las mismas.

1.4.1 Metodología de desarrollo de software

Una metodología impone un proceso de forma disciplinada sobre el desarrollo de software con el objetivo de hacerlo más predecible y eficiente. Una metodología define una representación que permite facilitar la manipulación de modelos, y la comunicación e intercambio de información entre todas las partes involucradas en la construcción de un sistema (Gacitúa,2003).

Una metodología de desarrollo de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. Comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto software desde que surge la necesidad del producto hasta que se alcanza el objetivo por el cual fue creado. Las metodologías definen con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas (Laboratorio Nacional de Calidad del Software de INTECO, 2009).

Las metodologías de software se dividen en dos grandes grupos, metodologías ágiles y metodologías robustas o tradicionales

1.4.1.1 Metodologías ágiles

En una reunión celebrada en febrero de 2001 en Utah-EEUU, nace el término "ágil" aplicado al desarrollo de software. En esta reunión participan un grupo de 17 expertos de la industria del software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y responder a los cambios que puedan surgir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas. Varias de las denominadas metodologías ágiles ya se eran utilizadas con éxito en proyectos reales, pero les faltaba una mayor difusión y reconocimiento.

Tras esta reunión se creó The Agile Alliance, una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue el Manifiesto Ágil, un documento que resume la filosofía "ágil" (Patricio Letelier, Mª Carmen Penadés. 2006).

El manifiesto comienza con la enumeración los valores a partir de los cuáles se rige el desarrollo ágil entre estos se puede encontrar: al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas, desarrollar software que funciona más que conseguir una buena documentación, la colaboración con el cliente más que la negociación de un contrato y responder a los cambios más que seguir estrictamente un plan.

Cada metodología tiene características propias y hace hincapié en algunos aspectos más específicos, a pesar de que cada una se rige por los valores y principios establecidos en el manifiesto ágil. A continuación, se muestra un resumen de algunas de las metodologías ágiles más populares en la actualidad.

SCRUM: Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda

característica importante son las reuniones a lo largo proyecto. Éstas son las verdaderas protagonistas, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración (Schwaber, Beedle M., Martin, 2001).

Metodologías Crystal (Crystal Methodologies): Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo (de ellas depende el éxito del proyecto) y la reducción al máximo del número de artefactos producidos. Han sido desarrolladas por AlistairCockburn. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo, Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros) (Addison-Wesley, 2001).

Método de desarrollo de sistemas dinámicos (Dynamic Systems Development Method o DSDM): Define el marco para desarrollar un proceso de producción de software. Nace en 1994 con el objetivo el objetivo de crear una metodología RAD unificada. Sus principales características son: es un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos. Propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción, y finalmente implementación. Las tres últimas son iterativas, además de existir realimentación a todas las fases (Addison-Wesley, 1997).

Desarrollo Adaptable de Software (Adaptive Software Development o ASD): Su impulsor es JimHighsmith. Sus principales características son: iterativo, orientado a los componentes software más que a las tareas y tolerante a los cambios. El ciclo de vida que propone tiene tres fases esenciales: especulación, colaboración y aprendizaje. En la primera de ellas se inicia el proyecto y se planifican las características del software; en la segunda desarrollan las características y finalmente en la tercera se revisa su calidad, y se entrega al cliente. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo (Highsmith J, Orr K, 2000).

Desarrolla basado en funcionalidades (Feature DrivenDevelopment o FDD): Define un proceso iterativo que consta de 5 pasos. Las iteraciones son cortas (hasta 2 semanas). Se centra en las fases de

diseño e implementación del sistema a partir de una lista de características que debe reunir el software. (Coad P., Lefebvre E., De Luca J, 1999).

Programación extrema (Xtreme Programing o XP): es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promueve el trabajo en equipo, se preocupa por el aprendizaje de los desarrolladores, y propicia un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico (Beck. K, 2000).

Proceso Unificado Ágil (AUP): Es una versión simplificada de la metodología robusta RUP**.** Esta describe de una manera fácil y simple cómo desarrollar aplicaciones de software a través de técnicas ágiles y conceptos que aún se mantienen válidos en RUP. AUP se preocupa especialmente de la gestión de riesgos. Propone que aquellos elementos con alto riesgo obtengan prioridad en el proceso de desarrollo y sean abordados en etapas tempranas del mismo. El proceso AUP establece un modelo más simple que el que aparece en RUP por lo que reúne en una única disciplina las disciplinas de Modelado de Negocio, Requisitos y Análisis y Diseño. El resto de disciplinas (Implementación, Pruebas, Despliegue, Gestión de Configuración, Gestión y Entorno) coinciden con las restantes de RUP (Universidad Unión Bolivariana, 2014).

AUP versión para la UCI: Como parte de un proceso de estandarización llevado a cabo en los centros productivos de la Universidad de Ciencias Informáticas se adopta el uso de una versión de la metodología AUP para todos los centros productivos de la universidad. Entre las principales diferencias presentes en esta versión están los cambios en las fases de desarrollo donde AUP define 4 fases (Inicio, Elaboración, Construcción, Transición), y su versión para la UCI mantiene la fase de Inicio, pero se modifica el objetivo, agrupa las otras 3 fases en una llamada Ejecución e incorpora una llamada Cierre. Otras diferencias están presentes en las disciplinas, ya que AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), la metodología definida en la UCI tiene 8 disciplinas. Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran cada uno de ellos disciplinas. Específicamente en la disciplina Requisitos plantean 4 posibles escenarios para la

identificación y descripción de requisitos. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación y la disciplina Despliegue se considera opcional. También existen diferencias en cuanto al número de roles definidos de 9 roles de AUP a 11 en su versión para la Universidad de Ciencias Informáticas. Esta versión plantea el desarrollo iterativo, donde se obtenienen incrementos en cada una de las iteraciones.

Las metodologías ágiles facilitan el trabajo en equipos pequeños de desarrollo, ya que se concentran más en el desarrollo de la solución que en la generación de documentación exhaustiva, pero necesitan de una constante comunicación entre los clientes y desarrolladores.

1.4.1.2 Metodologías robustas

Las metodologías tradicionales o robustas se focalizan en documentación, planificación y procesos. Las metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del producto software. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Además, las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son métodos apropiados en el entorno donde se trabaja los requisitos no pueden predecirse o bien pueden variar. Entre las principales metodologías robustas que existen en la actualidad están:

Proceso Unificado Relacional (RUP): es un proceso formal. Provee un acercamiento disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad que satisfaga los requerimientos de los usuarios finales. Fue desarrollado por Rational Software, y está integrado con toda la suite Rational de herramientas. Puede ser adaptado y extendido para satisfacer las necesidades de la organización que lo adopte. Es guiado por casos de uso y centrado en la arquitectura, y utiliza UML como lenguaje de notación (Figueroa, Solís, Cabrera, 2003).

Microsoft Solution Framework (MSF): Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de

proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo, deja en un segundo plano las elecciones tecnológicas. Esta metodología es adaptable, escalable y flexible. MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación. La Metodología MSF se adapta a proyectos de cualquier dimensión y de cualquier tecnología (León, 2012).

A partir de las características antes descritas de los enfoques ágiles y robustos y sus diferentes metodologías, para el desarrollo de la solución propuesta en la presente investigación se selecciona dentro de las metodologías ágiles la versión de AUP para la UCI. El autor de la presente investigación se basa fundamentalmente en que se desarrolló por un equipo de desarrollo pequeño, y a que es necesario generar toda la documentación necesaria que permita el posterior mantenimiento a la solución propuesta, y su correcta integración con la plataforma educativa Zera 2.0.

1.4.2 Lenguaje de modelado

El modelado es el diseño de aplicaciones de software antes de la codificación. El modelado es una parte esencial de grandes proyectos de software, útil para proyectos medianos e incluso pequeños. Si se utiliza un modelo, los responsables del éxito de un proyecto de desarrollo de software puede asegurarse a sí mismos de que la funcionalidad del negocio es completa y correcta, que se satisfacen las necesidades de los usuarios finales, y el diseño del programa es compatible con los requisitos de escalabilidad, robustez, seguridad, extensibilidad, y otras características. Es el modelado la única manera de visualizar el diseño y cotejarla con los requisitos antes de que se comience a escribir código (Inc.Object Management Group®, 2004).

Lenguaje unificado de modelado (Unified Modeling Language o UML): Le ayuda a especificar, visualizar y documentar esquemas de sistemas de software, incluyen su estructura y diseño, de manera que cumpla con todos estos requisitos. El uso de cualquier herramienta basadas en UML, puede analizar los requisitos de su aplicación en el futuro y diseñar una solución que les satisfaga (Inc.Object Management Group®, 2004). La versión que se utilizó es la 2.0.

1.4.3 Herramienta de modelado

Visual Paradigm for UML: es una herramienta CASE que soporta el modelado mediante UML y proporciona asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. Facilita el modelado de UML, ya que proporciona herramientas específicas para ello. Esto también permite la estandarización de la documentación, ya que la misma se ajusta al estándar soportado por la herramienta. Al disponer de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, se evita así duplicidades. Permite integrarse con otras aplicaciones, como herramientas ofimáticas, lo cual aumenta la productividad. Favorece el trabajo en grupo, ya que proporciona herramientas de compartición de trabajo. Permite generar código de forma automática, lo que reduce los tiempos de desarrollo, evita errores en la codificación del software y admite generar diversos informes a partir de la información introducida en la herramienta (Visual Parading, 2016). La versión que el autor utiliza en la presente investigación es la 8.0.

1.4.4 Tecnologías y lenguaje del lado del cliente

Lenguaje de marcas de hipertexto (Hyper Text Markup Language o HTML): Se trata de un lenguaje de marcas que nos permite representar de forma rica el contenido y también referenciar otros recursos, enlaces a otros documentos, mostrar formularios para posteriormente procesarlos.

Es un lenguaje que se utiliza para crear documentos que muestren una estructura hipertexto. Además, permite crear documentos de tipo multimedia, es decir, que contengan información más allá de la simplemente textual como, por ejemplo: imágenes, videos, sonido, entre otros. Los documentos HTML se conforman como documentos de texto planos, en los que todo el formato del texto se especifica mediante marcas de texto (nombradas etiquetas), que delimitan los contenidos a los que afecta la etiqueta (Mateu, 2004). En el desarrollo del componente de respaldo la versión a utilizar será la 5.0.

Hoja de estilo en cascada (Casaca de Style Sheets o CSS): es un lenguaje que describe la presentación de los documentos estructurados en hojas de estilo para diferentes métodos de interpretación, es decir, describe cómo se va a mostrar un documento en pantalla.*CSS* es una especificación desarrollada por el W3C (World Wide Web Consortium) para permitir la separación de los contenidos de los documentos escritos en HTML, XML, XHTML, SVG, o XUL de la presentación del

documento. Las hojas de estilo, incluyen elementos tales como los colores, fondos, márgenes, bordes, tipos de letra, permite a los desarrolladores modificar la apariencia de una página web de una forma más sencilla (Pérez, 2008). La versión a utilizar es la 3.0.

Java Script: es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos (Eguiluz, 2016). La versión a utilizar es la 1.8.

1.4.5 Tecnologías y lenguajes del lado del servidor

Preprocesador de hipertexto (Hypertext Preprocessor o PHP): es un lenguaje script (no se compila para conseguir códigos máquina si no que existe un intérprete que lee el código y se encarga de ejecutar las instrucciones que contiene éste código), para el desarrollo de páginas web dinámicas del lado del servidor, cuyos fragmentos de código se intercalan fácilmente en páginas HTML, debido a esto, y a que es de Open Source (código abierto), es el más popular y extendido en la web. PHP es capaz de realizar determinadas acciones de una forma fácil y eficaz sin tener que generar programas programados en un lenguaje distinto al HTML. Esto se debe a que PHP ofrece un extenso conjunto de funciones para la explotación de bases de datos sin complicaciones (Mateu, 2004). Durante el desarrollo de la solución propuesta se utilizó la versión 5.5.9

1.4.6 Framework del lado del servidor

Symfony: es un proyecto PHP de software libre que permite crear aplicaciones y sitios web rápidos y seguros de forma profesional. Su código, y el de todos los componentes y librerías que incluye, se publican bajo la licencia de software libre. La documentación del proyecto también es libre e incluye varios libros y decenas de tutoriales específicos. La excelente herramienta Composer, que simplifica de forma radical la instalación y gestión de las dependencias de las aplicaciones PHP, también ha sido creada por varios miembros de la comunidad Symfony (Symfony, 2010). Para el desarrollo de la propuesta de solución se utilizó la versión 2.7.9.

1.4.7 Capa de acceso a datos

Doctrine ORM: es una técnica para el mapeo de objetos-relacional (Object Relational Mapping ORM) escrito en PHP que proporciona una capa de persistencia para objetos PHP. Es una capa de abstracción que se sitúa justo encima de un SGBD (sistema de gestión de bases de datos). Doctrine puede generar clases a partir de una base de datos existente y después el programador puede especificar relaciones y añadir funcionalidad extra a las clases autogeneradas. No es necesario generar o mantener complejos esquemas XML de base de datos como en otros frameworks. (Doctrine-Project, 2006). La versión utilizada es la 2.0.

1.4.8 Sistema gestor de base de datos

Postgres SQL: es uno de los Sistemas gestores de base de datos más antiguos y conocidos del mundo del código libre. Las características más destacadas son: soporte de transacciones, subconsultas, soporte de vistas, integridad referencial, herencia de tablas y tipos de datos definidos por el usuario. Además, permite añadir tablas en tiempo de ejecución, posee funciones de agregación definibles por el usuario, entre otras (Mateu, 2004). La versión utilizada en la solución propuesta es la 9.4.

1.4.9 Servidor web

Apache HTTP: También llamado Apache, es un servidor web HTTP de código abierto para la creación de páginas y servicios web. Es un servidor multiplataforma, gratuito, muy robusto y que destaca por su seguridad y rendimiento (Apache Software Foundation, 1999).

1.4.10 Entorno integrado de desarrollo

Netbeans IDE: es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso. Ofrece todas las funciones de los entornos de desarrollo integrado avanzados como diseño de interfaces, asistentes para la conexión con bases de datos, creación automática de propiedades y clases, etc (Netbeans, 2011). La versión a utilizar será la 8.0.

1.5 Conclusiones parciales

En este capítulo se realizó un estudio referente a los principales conceptos asociados al dominio del problema. Se realizó el análisis de otras plataformas educativas y la importancia de sistemas de respaldos a cursos en las mismas.

Se decidió desarrollar un sistema de respaldo a cursos para ello se utilizó como lenguaje de programación PHP v5.5.9 y JavaScript v1.8, como framework de desarrollo del lado del servidor Symfony v2.7.9. Además, se utilizó como servidor web Apache en su versión v2.4.7, como SGBD PostgreSQL v9.4 y Doctrine v2.0 como técnica para mapear objetos relacionales. Las herramientas Netbeans y Visual Paradigm, ambas en su versión 8.0 fueron utilizadas como entorno de desarrollo y para el modelado respectivamente. Todo el proceso de desarrollo del software fue guiado y controlado por la metodología de desarrollo, AUP en su versión para la UCI.

Capítulo 2. Descripción del sistema propuesto

2.1 Introducción

En este capítulo se realiza la descripción de la propuesta de solución. La metodología AUP, en su versión UCI, define en cada una de sus fases un grupo de artefactos que tienen como fin facilitar la comprensión del negocio y del sistema a desarrollar. Entre los artefactos que se presentan se encuentran: el modelo de domino, las historias de usuarios, los diagramas de clases del análisis y los diagramas de clases del diseño. Además, se hace referencia a los patrones de diseño utilizados en aras de obtener un software de calidad.

Modelo de dominio

Un modelo del dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. También se les denomina modelos conceptuales, modelo de objetos del dominio y modelos de objetos de análisis. Un modelo del dominio es una representación de las clases conceptuales del mundo real, no de componentes software. No se trata de un conjunto de diagramas que describen clases software, u objetos software con responsabilidades (Larman, 2003).

2.1.1 Definición de las clases del modelo de dominio

Plataforma Educativa Zera 2.0: Es la plataforma educativa donde se realiza el respaldo o la restauración de cursos y sus contenidos.

Curso: Contexto perteneciente a la plataforma donde ocurre el proceso enseñanza aprendizaje.

Estructura: Forma en que están distribuidos los diferentes elementos que componen un curso.

Recurso: Material multimedia utilizado como apoyo en el proceso enseñanza aprendizaje.

Actividad: Elemento creado por el profesor, donde los estudiantes realizan determinada acción y reciben o no una calificación.

Tarea: Tipo de actividad que contiene una orientación concreta realizada por el profesor a los estudiantes.

Fórum: Tipo de actividad que brinda el espacio de comunicación dentro de la plataforma donde los estudiantes dan sus criterios sobre determinada materia.

Tema: Contenido específico al que está destinado un espacio.

Cuestionario: Tipo de actividad que agrupa uno o más ejercicios en la plataforma educativa Zera 2.0

Ejercicio: Una unidad utilizada para adquirir conocimiento o habilidad en determinada materia.

Componente de respaldo: Componente con las responsabilidades de realizar la copia de seguridad a los elementos de un determinado curso.

Respaldo: Unidad que contiene datos de un curso y que cuenta con la información necesaria para poder restaurar estos datos en una instancia de la plataforma educativa Zera 2.0.

Opciones: Acciones que se pueden realizar sobre los respaldos.

Realizar: Almacena los datos seleccionados del curso.

Restaurar: Restaura los datos contenidos en un respaldo.

Eliminar: Elimina la unidad de respaldo seleccionada disponible en el sistema.

Descargar: Permite la ubicación en un directorio definido por el usuario del archivo contenedor del respaldo.

2.1.2 Diagrama de clases del modelo de dominio

Los conceptos identificados con anterioridad son representados en el siguiente diagrama (Figura 1) donde se muestran las relaciones que se establecen entre ellos.

Capítulo 2. Descripción del sistema propuesto

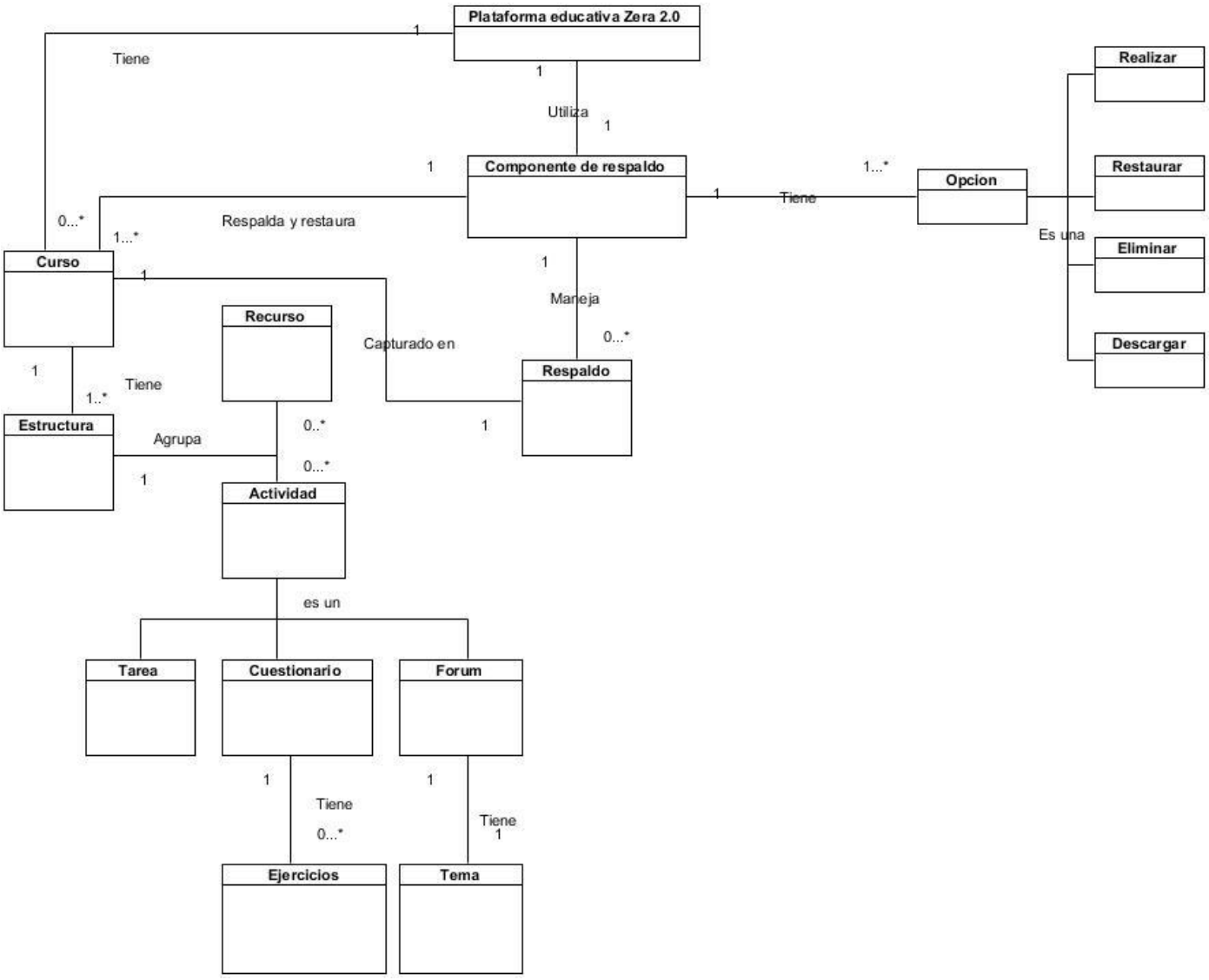


Figura 1. Modelo de Dominio

Capítulo 2. Descripción del sistema propuesto

2.2 Descripción del componente de respaldo de información a cursos en la plataforma educativa Zera 2.0.

El componente de respaldo de información a los cursos facilita la reusabilidad de materiales educativos y brinda un mecanismo de seguridad ante eventuales pérdidas de información en la plataforma educativa Zera 2.0. El mismo permite hacer una copia personalizada de los elementos que componen un curso dentro de la plataforma y la restauración de datos previamente respaldados.

El proceso de respaldo consiste en almacenar los datos de las entidades a respaldar, para ello se realiza un recorrido recursivo por el modelo de datos a través de las relaciones que se establecen entre dichas entidades. Pueden existir diferentes tipos de respaldo, donde cada tipo define que entidades incluirá, y cuál o cuáles serán las entidades de partida para realizar el respaldo. Los datos son almacenados en una estructura de arreglos anidados, estos son serializados y almacenados en un fichero (yml). También se almacena una estructura de carpetas con los recursos multimedia utilizados como recursos educativos o material de apoyo en las actividades del curso. Estos elementos son comprimidos en un fichero (zip). Además, el sistema permite descargar este fichero zip en un directorio determinado por el usuario. El proceso de restaura se inicia con la selección de un fichero de restauración cargado en el servidor, el usuario indica qué fichero de restauración desea recuperar y los datos del respaldo se integran a la plataforma, de esta manera se recupera la información contenida en el respaldo. La utilización del componente de respaldo a cursos posibilita a la plataforma educativa Zera 2.0:

- La reusabilidad de recursos y actividades en otros cursos de la plataforma educativa Zera 2.0 u otra instancia de la misma.
- Colabora a la disponibilidad de los cursos.
- Reduce costos de tiempo y esfuerzo en la creación de cursos.
- Un mecanismo de seguridad ante eventuales pérdidas de información.
- La reusabilidad de cursos en varias instancias de la plataforma educativa Zera 2.0.

2.3 Requisitos del software

Un requisito de software constituye una necesidad bien documentada sobre una forma o funcionalidad que debe ser cumplida por un sistema de software. En algunos casos los requisitos funcionales pueden declarar explícitamente lo que el sistema no debe hacer. (Sommerville, 2005). En otras palabras, se

plantean con el fin de lograr un entendimiento entre el cliente y el equipo de desarrollo, en aras de satisfacer sus necesidades.

2.3.1 Requisitos funcionales del sistema.

Los requisitos funcionales describen lo que el sistema o software debe hacer. Una función es una capacidad útil proporcionada por uno o más componentes de un sistema (Young, 2004).

A continuación, se relacionan los requisitos funcionales del componente de respaldo de información cursos en la plataforma educativa Zera 2.0.

Tabla 1. Requisitos funcionales del Sistema.

No.	Nombre del requisito	Descripción	Prioridad	Complejidad
RF 1	Respaldar recursos	El sistema debe permitir respaldar la información de los recursos contenidos en un curso.	Alta	Media
RF 2	Respaldar fórums	El sistema debe permitir respaldar la información de las actividades de tipo Fórum contenidos en un curso.	Alta	Media
RF 3	Respaldar tareas	El sistema debe permitir respaldar la información de las actividades de tipo Tarea contenidos en un curso.	Alta	Media
RF 4	Respaldar cuestionarios	El sistema debe permitir respaldar la información de las actividades de tipo Cuestionario	Alta	Media

		contenidos en un curso.		
RF 5	Respaldar curso	El sistema debe permitir respaldar los cursos de la plataforma.	Alta	Alta
RF 6	Restaurar respaldo	El sistema debe permitir restaurar un respaldo previamente realizado en la plataforma educativa Zera 2.0 u otra instancia de la misma.	Alta	Alta
RF 7	Importar respaldo	El sistema debe permitir importar al sistema un respaldo de información previamente realizado en la plataforma educativa Zera 2.0 u otra instancia de la misma.	Media	Baja
RF 8	Eliminar respaldo	El sistema de debe permitir eliminar un respaldo de información existente en el sistema.	Media	Baja
RF 9	Listar respaldos	El sistema debe permitir listar los respaldos de información existentes en el sistema.	Media	Alta
RF 10	Descargar respaldo	El sistema debe permitir la descarga del respaldo de información (comprimido Zip) en un directorio determinado por el	Media	Baja

		usuario.		

2.3.2 Requisitos no funcionales del sistema.

Los requisitos no funcionales especifican las propiedades del sistema, tales como la fiabilidad y seguridad. (Young, 2004). A continuación, la tabla 2 muestra la relación de requisitos no funcionales para la plataforma educativa Zera 2.0.

Tabla 2. Requisitos no funcionales del Sistema.

Atributo de Calidad	Funcionalidad
Sub-atributos/Sub-características	Seguridad
Objetivo	El sistema cumplirá con requisitos de seguridad tales como: • Garantizar la protección de información de accesos no autorizados, • Garantizar el acceso a las funcionalidades definidas para los usuarios de acuerdo a los roles que posean,
Origen	Interno al sistema/ externo al sistema
Artefacto	Servicios del sistema/Datos del sistema
Entorno	En línea o fuera de línea, conectado o desconectado, cortafuegos o abierta
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1.a Ataques de autenticación	
Intentos erróneos consecutivos de acceso.	• Se solicita un captcha para comprobar que no es un ataque por parte de un sistema automatizado.

	• Registro de eventos en los logs**.**
1.b Acceso no autorizado	
Intentos de acceso a una acción sin privilegios.	• Registro de eventos en los logs. • Se muestra el mensaje de información: No tiene suficientes privilegios para realizar esta acción. • Se deniega el acceso a la acción.
1.c Inyección SQL	
Intentos de inyección SQL en los formularios.	• Registro de eventos en los logs. • No se procesan los datos y se muestra el mensaje de error: Datos incorrectos. • Bloquea el usuario al intentar más de 3 veces y muestra el mensaje de información: Su usuario ha sido bloqueado contacte con el administrador.
Medida de respuesta	
Número de intentos de acceso de autenticación, debe ser mayor a 3. Intentos de inyección SQL, debe ser mayor a 3.	
Atributo de Calidad	Usabilidad
Sub-atributos/Sub-características	Conformidad
Objetivo	Cumplir con las pautas de diseño establecidas en la Estrategia Marcaria de la Universidad.
Origen	UCI
Artefacto	Interfaces del sistema
Entorno	En línea
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1.a Realizar una acción en el sistema.	
En el caso de que se cree/actualice/elimine un elemento, así como el cancelar.	Se muestra un mensaje con el resultado de la acción.
1.b Informar al usuario	
Cuando se ubica sobre un botón/hipervínculo.	El puntero del mouse cambia y cambia el color del elemento.
Cuando se solicite información del usuario.	Cada campo tiene asociado una pequeña ayuda.
Medida de respuesta	
NA	

Capítulo 2. Descripción del sistema propuesto

2.4 Historia de Usuario

Una historia de usuario es una representación de un requisito escrito en una o dos frases donde utiliza el lenguaje común del usuario. Las historias de usuario son utilizadas en las metodologías de desarrollo ágiles para la especificación de requisitos. Cada historia de usuario debe ser limitada. Las historias de usuario son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Las historias de usuario permiten responder rápidamente a los requisitos cambiantes (Scrum Manager®, 2014). Para la especificación de los requisitos en la presente investigación se hizo uso de la documentación establecida por el centro productivo FORTES.

A continuación, la Tabla 3 muestra la HU para el requisito funcional Respaldar recursos.

Tabla 3: HU Respaldar recursos

Número: 1	**Nombre del requisito:** Respaldar recursos
Programador: Ulises Fernández-Nespral Vázquez	**Iteración Asignada:** 1era
Alta	**Tiempo Estimado:** 7 días
Riesgo en Desarrollo:	**Tiempo Real:** 6 días
Descripción: **1- Objetivo:** Permitir respaldar los datos de los recursos de un curso en el sistema. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para respaldar los recursos pertenecientes a un curso hay que: - Estar autenticado en el sistema con el rol Administrador o Profesor Editor. - Debe existir en el sistema al menos un curso.	

3- Flujo de la acción a realizar:

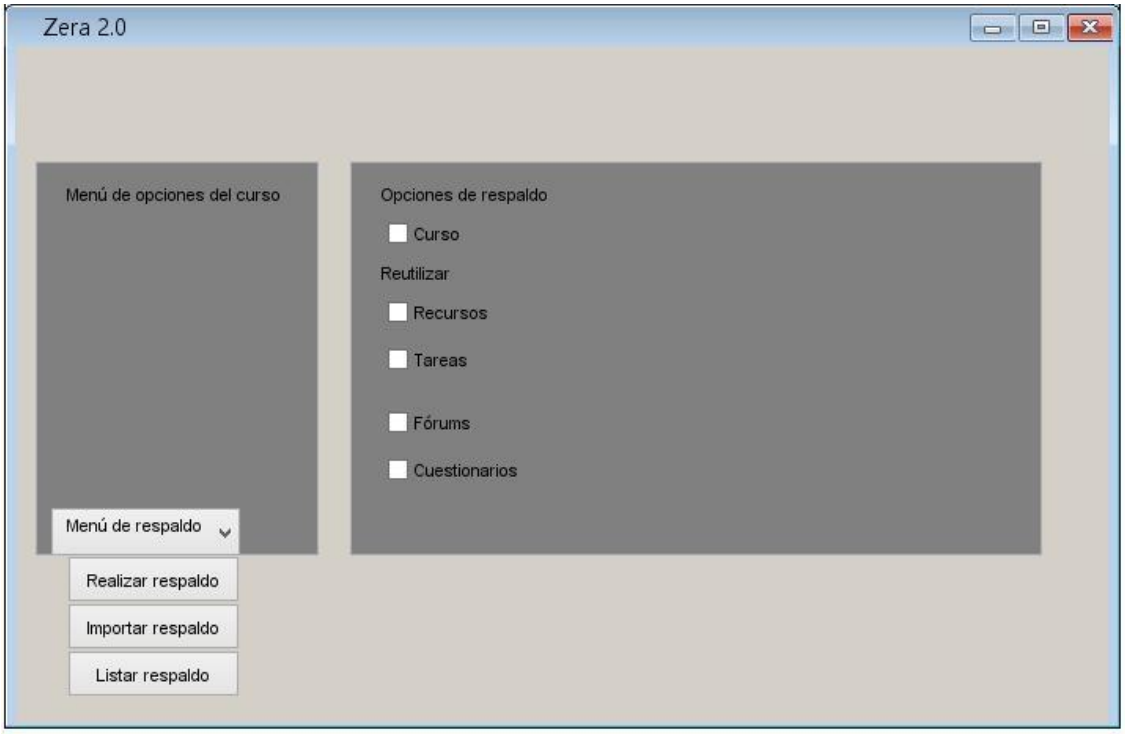
-Si el usuario selecciona la opción Respaldar recursos se genera un fichero Zip en la carpeta upload/backup. Este fichero contiene los recursos y un fichero yml con los datos de las entidades recurso.

-Se genera una entidad Backup_Restore que tiene los datos de dicho respaldo.

-Se listan los respaldos de información del curso

Observaciones: los respaldos de información son para almacenar datos de los cursos.

Prototipo de interfaz:



2.5 Diagrama de clases del análisis

Una clase de análisis representa una abstracción de una o varias clases y/o subsistemas del diseño del sistema. Las clases del análisis siempre encajan en uno de los tres estereotipos básicos: clase de interfaz, de control y de entidad (Jacobson, 2000). A continuación, en la figura 2 se muestra el diagrama de clase

del análisis correspondiente al HU Respaldar recursos. Para el estudio de los demás diagramas de clases del análisis, remitirse a los Anexo 2.

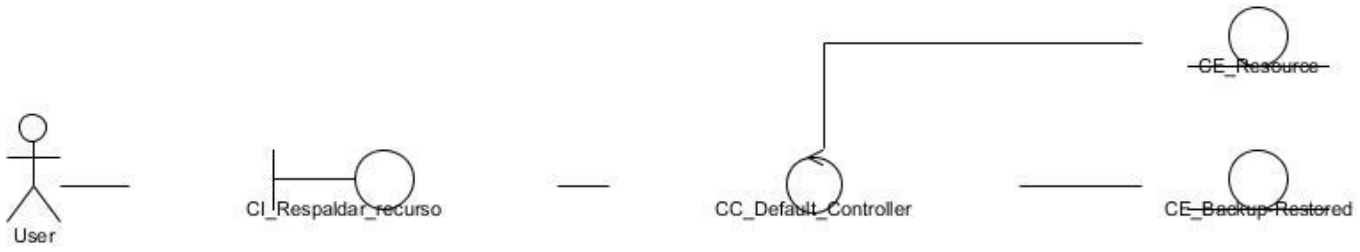


Figura 2: Diagrama de clases del Análisis HU Respaldar recursos

2.6 Diagrama de colaboración del análisis

Los diagramas de colaboración muestran las interacciones entre objetos crea enlaces entre ellos y añade mensajes a esos enlaces. El nombre de un mensaje debería denotar el propósito del objeto invocante en la interacción con el objeto invocado (Jacobson et al., 2000). A continuación, la Figura 3 muestra el Diagrama de Colaboración del Análisis correspondiente a la HU Respaldar recursos. Para el estudio de los demás diagramas de clases del análisis, remitirse a los Anexo 3.

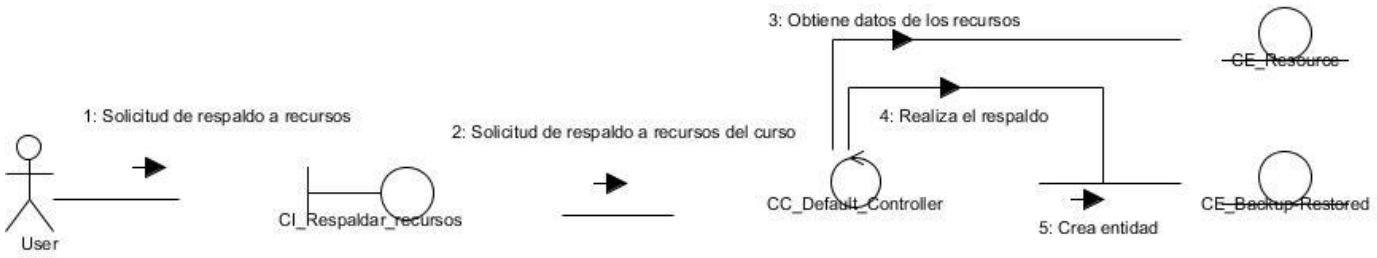


Figura 3: Diagrama de colaboración del análisis HU Respaldar recursos

2.7 Patrones de arquitectura

Los patrones de arquitectura expresan el esquema fundamental de organización para sistemas de software. Proveen un conjunto de subsistemas predefinidos; especifican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos (Welicki, 2016).

2.7.1 Patrón Modelo - Vista - Controlador

Symfony está basado en un patrón clásico del diseño web conocido como arquitectura Modelo Vista Controlador (MVC), que está formado por tres niveles:

- El Modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La Vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El Controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, así la vista y las acciones son independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación (Potencier y Zaninotto, 2005).

El uso del patrón arquitectónico MVC en la propuesta de solución permite separar las clases encargadas de la lógica del negocio como, Backup_Restored.php y las clases encargadas de controlar las peticiones del usuario como DefaultController, y del código encargado de mostrar los datos como listar_backup.html.twig, backup_restore.html.twig, backupIndex.html.twig. Además, facilita la creación de

una estructura de directorios dentro de cada Bundle para separar los objetos del modelo, la vista y el controlador.

2.7.2 Patrones de diseño

Los patrones de diseño constituyen la solución de un problema determinado y se pueden aplicar en diferentes contextos, su finalidad no es expresar nuevas ideas en el diseño sino resolver los problemas mediante una solución ya probada y fiable (UML y patrones, 2003).

Para el desarrollo de un software de calidad, legible y entendible se deben tener en cuenta cinco principios básicos: Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversión (SOLID). La utilización de estos principios para el desarrollo y diseño de software permite eliminar código sucio lo que provoca que el programador tenga que refactorizar el código fuente hasta que sea legible y extensible. A continuación, se mencionan los principios SOLID a tener en cuenta en el desarrollo del componente de respaldo a cursos en la plataforma educativa Zera en su versión 2.0

Principio de responsabilidad única (Single Responsibility Principle o SPR): el principio de única responsabilidad hace referencia a que una clase debería concentrarse solo en hacer una cosa de tal forma que si cambia algún requisito en mayor o menor medida este cambio solo afecte a dicha clase por una razón.

Principio de Abierto/ Cerrado (Open/Closed Principle u OCP): el principio abierto/cerrado cambia el comportamiento de una clase mediante herencia, polimorfismo y composición.

Principio de sustitución de Liskov (Liskov Substitution Principle o LPS): el principio de sustitución de Liskov hace referencia a que las subclases deben comportarse adecuadamente en el momento sean usadas en lugar de sus clases base.

Principio de segregación de interfaz (Interface Segregation Principle o ISP): el principio de segregación de interfaces permite mantener las interfaces pequeñas y cohesivas, que puedan coexistir unas con otras.

Capítulo 2. Descripción del sistema propuesto

Patrones generales de software para asignación de responsabilidades (General Responsibility Asignment Software Patterns o GRASP):

Experto: se aplica en el modelo a las clases encargadas de la abstracción de datos. Estas son las clases responsables de hacer las consultas a la base de datos para ello utilizan Doctrine, pues cuentan con los atributos necesarios para ejecutar esa función, por lo que tienen las responsabilidades de realizar directamente las acciones sobre la base de datos. La utilización de este patrón se evidencia en la clase BackupRepository.

Creador: es el encargado de identificar quien debe ser el responsable de la creación o instanciación de nuevos objetos o clases (García, 2012). Un ejemplo de la utilización de este patrón se muestra en las clases DefaultController.

Bajo acoplamiento: se evidencia en la capa modelo. Las clases de acceso a los datos tienen bastante independencia de las clases de abstracción de datos. La poca dependencia entre las clases permite una mayor reutilización (I. I. Santiesteban, 2011). Las clases BackupRepository.php y BackupRestore.php son ejemplo de la utilización de dicho patrón.

Alta cohesión: debido a la estructura de los proyectos en Symfony que facilita la organización del trabajo es posible crear y trabajar con clases con una alta cohesión. Esto hace posible que el software sea flexible a cambios sustanciales con efecto mínimo (Santiesteban et al., 2011). La utilización de este patrón se evidencia en las clases BackupRepository.php y BackupManager.php.

Controlador: sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es el controlador quien recibe los datos del usuario y quien los envía a las distintas clases según el método llamado (García, 2012). Un ejemplo de la utilización de este patrón se muestra en las clases DefaultController.

Patrones Pandilla de los Cuatro (Gang of Four o GoF):

Patrón Instancia única (The Singleton Pattern): asegura que una clase tiene solo una instancia, y proporciona un punto de acceso global a la misma (Freeman and Freeman, 2004). Un ejemplo de la utilización de este patrón se muestra en la clase del componente de rutas que utiliza el framework.

Patrón la fábrica (The Factory Pattern): define una interfaz para crear un objeto, pero permite a las subclases decidir qué clase instanciar. Además, permite aplazar una instanciación de clases a las subclases (Freeman and Freeman, 2004). Un ejemplo de la utilización de este patrón se evidencia en la clase BaseBackup.php.

Patrón observador (The Observer Pattern): define de una a muchas dependencias entre objetos de forma que, si un objeto cambia de estado, todas sus dependientes son notificadas y actualizadas de forma automática (Freeman and Freeman, 2004). La utilización de este patrón se evidencia en la clase BackupRestored.php.

Patrón inyección de dependencia (Dependency Injection Pattern): es donde los componentes dan sus dependencias a través de sus constructores, métodos, o directamente en los campos. Además, la mayoría de los frameworks PHP modernos utilizan inyección de dependencias para proporcionar un conjunto de componentes desacoplados pero cohesionados (Potencier, 2009). Un ejemplo de la utilización de este patrón se evidencia en la clase BackupManager.

2.8 Modelo de diseño

El diseño es técnicamente la parte central de la ingeniería del software. Durante el diseño se desarrollan, revisan y se documentan los refinamientos progresivos de las estructuras de datos, de la estructura del programa y de los detalles procedimentales (Gil, 2016).

2.8.1 Diagrama de clases del diseño

Los diagramas de clases del diseño se encuentran conectados a la realización de un caso de uso o una historia de usuario. Son los encargados de mostrar las clases participantes, subsistemas y sus relaciones, así como los atributos y operaciones correspondientes a cada clase (Jacobson et al., 2000).

A continuación, la Figura 4 muestra el diagrama de clase del diseño correspondiente al HU Respaldar recursos. Para el estudio de los demás diagramas de clases del diseño remitirse a los Anexo 4.

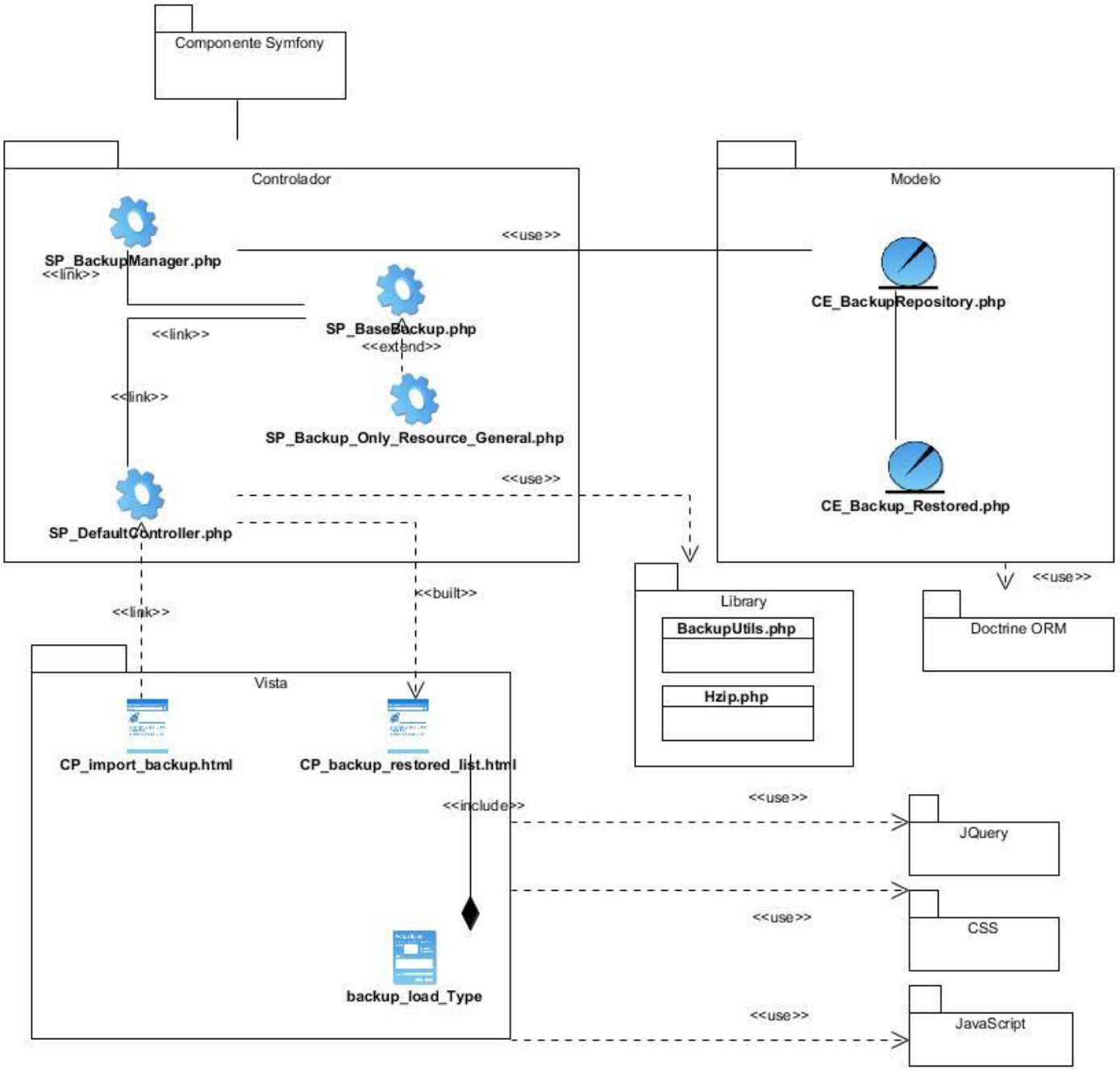


Figura 4. Diagrama de clases del diseño HU Respaldar recursos

2.8.2 Diagrama de secuencia del diseño

Los diagramas de secuencia muestran las interacciones entre objetos mediante transferencia de mensajes entre objetos o subsistemas. El nombre del mensaje debería indicar una operación del objeto que recibe la invocación o de una interfaz que el objeto proporciona (Jacobsonetal., 2000).

A continuación, la Figura 5 muestra el diagrama de secuencia del diseño correspondiente a la HU Respaldar recursos. Para el estudio de los demás diagramas de secuencia del diseño remitirse a los Anexo 5.

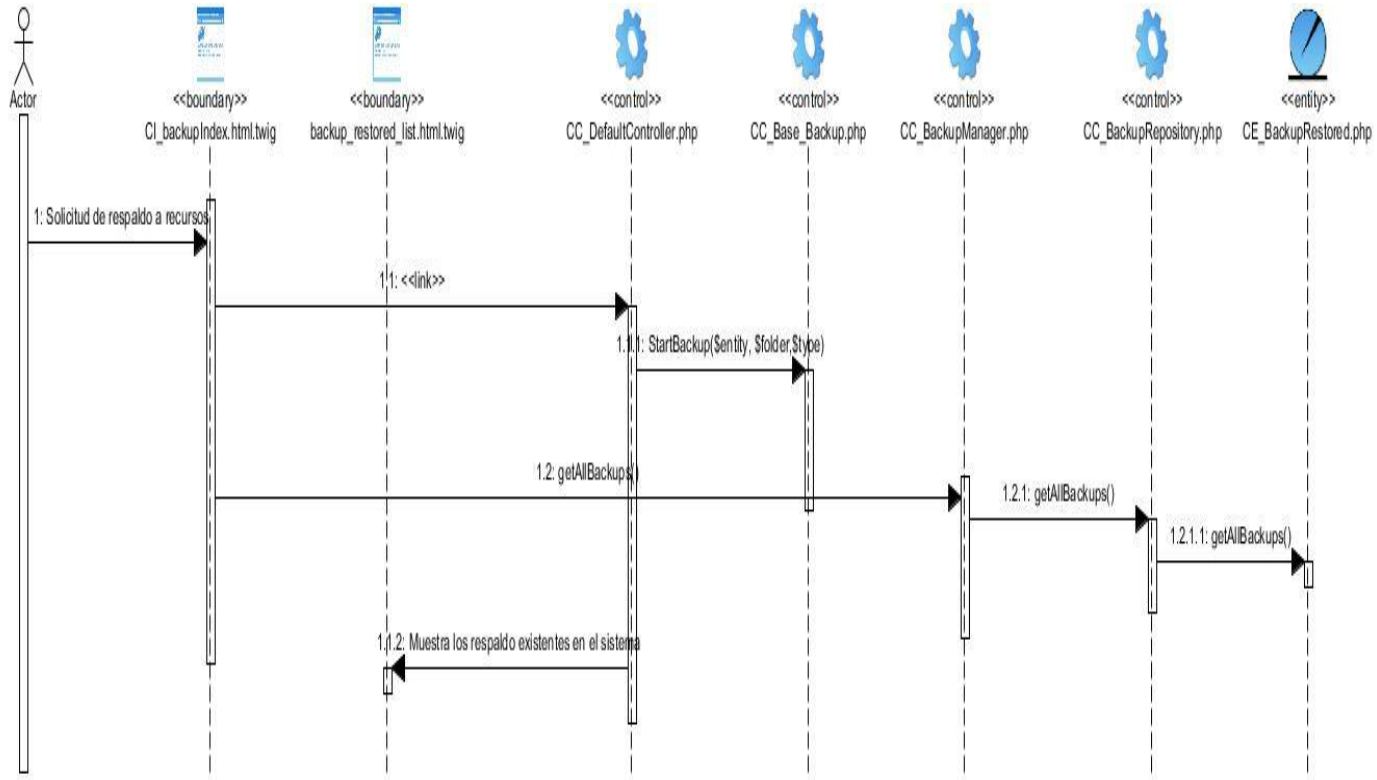


Figura 5. Diagrama de secuencia del diseño HU Respaldar recursos

2.8.3 Diagrama de despliegue

El diagrama de despliegue es un modelo de objetos que describe la distribución física del sistema. Se utiliza como entrada principal en las actividades de diseño e implementación, debido a que la distribución del sistema tiene una influencia principal en su diseño (Jacobsonetal,2000).

A continuación, la figura 6 presenta el Diagrama de despliegue propuesto para el sistema:

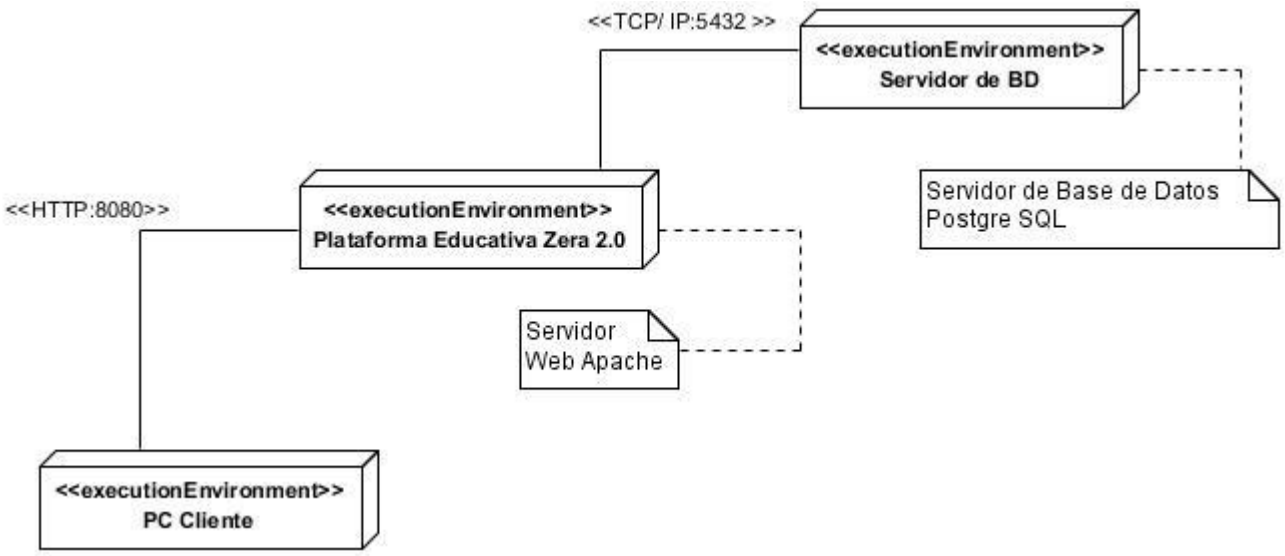


Figura 6: Diagrama de despliegue

2.8.4 Diseño de base de datos

En esta sección se presenta el modelo de datos que está compuesto por las entidades que pasaron a ser las tablas de la base de datos, para ser utilizadas por las funcionalidades a desarrollar. A continuación, la figura 6 muestra el diagrama Entidad-Relación

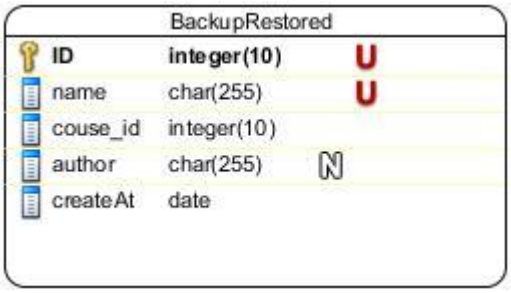


Figura 7: Diagrama entidad-relación

2.8.5 Descripción del modelo entidad relación

En esta sección se presenta una descripción de los atributos de la tabla tb_backup_restored.

tb_backup_restored		
Descripción: En la presente tabla se agrupa la información correspondiente a los respaldos realizados en a cursos en la plataforma educativa Zera 2.0		
Atributo	Tipo	Valor
id	integer	Etiqueta única que identifica al objeto en la tabla.
name	varchar	Almacena el nombre del respaldo.
crateAt	date	Almacena la fecha en que se realizó el respaldo.
author	varchar	Almacena el nombre del usuario que realizó alguna opción de respaldo.
course_id	Integer	Almacena el identificador del curso respaldado.

Capítulo 2. Descripción del sistema propuesto

2.9 Conclusiones parciales

Durante la fase de elaboración de la metodología de desarrollo AUP en su versión para la UCI, llevado a cabo en el desarrollo del componente de respaldo a cursos para la plataforma educativa Zera 2.0, se generó:

- El diagrama de modelo de dominio, el cual posibilitó comprender los conceptos fundamentales que forman parte del dominio del sistema.
- Se realizó el levantamiento de los requisitos funcionales y no funcionales y para su especificación se utilizó el artefacto HU según lo establecido en el escenario número 4 de la metodología AUP en su versión para la UCI.
- Los diagramas de clases del análisis y colaboración correspondientes a los requisitos funcionales.
- Los diagramas de secuencias correspondientes a las HU
- El diagrama entidad relación que define la estructura de la base de datos.
- Se definió la arquitectura MVC como arquitectura a aplicar por las facilidades que proporciona, se utilizaron varios patrones para el diseño que facilitan la reutilización y mejor organización del componente de respaldo a cursos.

Capítulo 3. Implementación y prueba

3.1 Introducción

En este capítulo se documenta el proceso de implementación de los elementos identificados durante la realización del diseño. Para ello se modeló el diagrama de componentes con el objetivo mostrar la organización y las dependencias lógicas que existen entre los ficheros que contienen código fuente. También se documentan las pruebas de software realizadas para validar el correcto funcionamiento de los componentes desarrollados.

3.2 Implementación

El propósito principal de la implementación es desarrollar la arquitectura y el sistema como un todo. Se parte del resultado del diseño y se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares. Durante la etapa de prueba cada construcción generada durante la implementación es sometida a pruebas de integración, y a pruebas de sistema (Jacobson, Booch, Rumbaugh,2000).

3.2.1 Diagrama de componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación en término de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación. Se utilizan para modelar la vista estática de un sistema. Muestran la organización y las dependencias lógicas entre un conjunto de componentes de software, sean estos componentes de código fuente, librerías, binarios o ejecutables (Jacobson, Booch y Rumbaugh,1999).

El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo de implementación, muestra las organizaciones y las dependencias entre tipos de componentes y organiza los subsistemas de implementación en capas.
A continuación, la Figura 8 muestra el Diagrama de componentes.

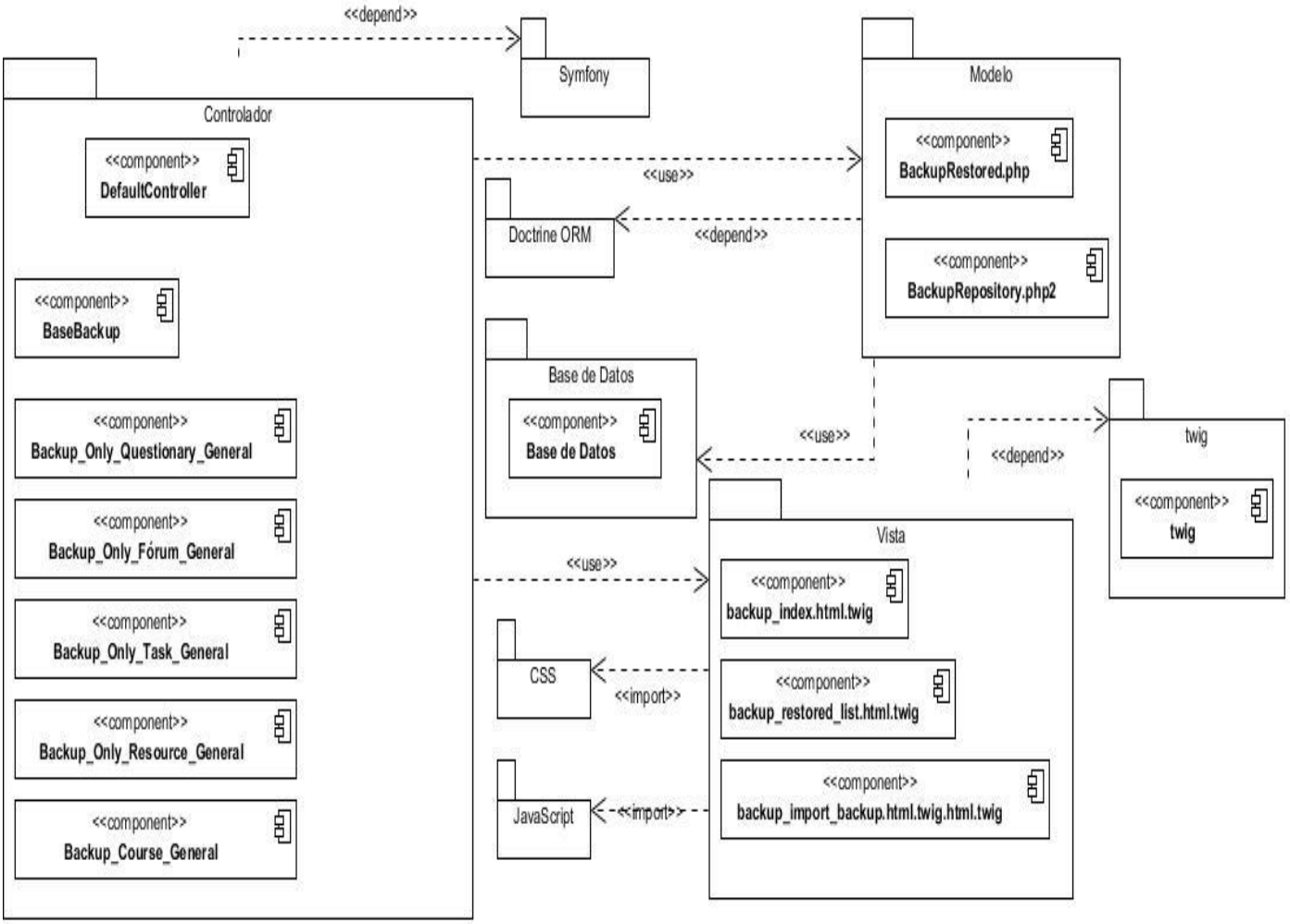


Figura 8. Diagrama de Componentes

3.3 Estándares de codificación

Los estándares de codificación son una serie de convenciones que deben seguir los desarrolladores dode se mantiene buenas prácticas definidas por la ingeniería de software, para obtener un código fácil de comprender y de alta calidad. Facilitan el mantenimiento de una aplicación. Posibilitan que un equipo de programadores mantenga un código legible sobre el que se efectuarán luego revisiones; con el objetivo de regular la calidad de la implementación, se establece un estándar de desarrollo común por el cual se rige la programación del sistema (Microsoft, 2016).

Capítulo 3. Implementación y pruebas

Durante el desarrollo del componente para respaldos a curso en la plataforma educativa Zera 2.0 el estilo de codificación utilizado fue:

División de líneas

Si una expresión ocupe más de una línea, esta se podrá romper o dividir en función de los siguientes criterios:

- ✓ Tras una coma.
- ✓ Se recomienda las rupturas de nivel superior a las de nivel inferior.

Ejemplos:

```
$data['relationsReference'][] = array('entity' => get_class($collection),

        'entityName' => $this->getEntityName($assoc['targetEntity']),

                'id' => $collection->getId(), 'field' => $field);
```

Una declaración por línea

Se recomienda el uso de una declaración por línea.

Ejemplo:

```
$curso_id = $request->get('curse_id');

$curso = $this->get('fortes_course.manager')->getCourseById($curso_id);

$folderBackup = 'backup' . uniqid();
```

Declaración de clases / interfaces

Durante el desarrollo de clases / interfaces se deben seguir las siguientes reglas de formato:

- ✓ No incluir ningún espacio entre el nombre del método y el paréntesis inicial del listado de parámetros.
- ✓ El carácter inicio de bloque ("{") debe aparecer en una línea independiente a la línea que contiene la sentencia de declaración.

- ✓ El carácter fin de bloque ("}") se sitúa en una nueva línea tabulada al mismo nivel que su correspondiente sentencia de inicio de bloque, excepto si la sentencia es nula, en tal caso se situará detrás de "{".

```
class BackupRestored

{

    private $id;

    private $author;

    private $course_id

  }

  …

  }
```

Clases e interfaces

Los nombres de clases deben ser sustantivos y deben tener la primera letra en mayúscula. Si el nombre es compuesto, cada palabra componente deberá comenzar con mayúscula. Los nombres serán simples y descriptivos. Debe evitarse el uso de acrónimos o abreviaturas, salvo en aquellos casos en los que dicha abreviatura sea más utilizada que la palabra que representa.

```
class BackupManager{

        …

}
```

Métodos

Los métodos deben ser infinitivos escritos en minúscula. Si el método esté compuesto por varias palabras cada una de ella tendrá la primera letra en mayúscula, se exceptúa la primera. A esta notación es conocida como Camel Case en su variante lowerCamelCase

```
form = $this->createFormBuilder();
```

```
$backup=$em->getRepository("BackupBundle: BackupRestored")->getAllBackups();
```

Con el uso de los estándares de codificación en la implementación del Componente de respaldo a cursos y sus contenidos en la plataforma educativa Zera 2.0 se formalizó un código legible y fácil de comprender, con facilidades de mantenimiento y posteriores revisiones.

3.4 Pruebas de software

Las pruebas de software son un proceso iterativo que implican ejercer una implementación del software con datos de prueba. Durante su aplicación se examinan las salidas del software y su entorno operacional para comprobar que funciona tal y como se quiere. Las pruebas son una técnica dinámica de verificación y validación (Sommerville, 2005).

Las pruebas de software son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador; probando el comportamiento del mismo. Las pruebas van dirigidas a componentes del sistema o al sistema de software en su totalidad, con el objetivo de medir el grado en que la aplicación cumple con los requerimientos.

3.4.1 Niveles de prueba

De acuerdo a determinados objetivos, las pruebas se clasifican en varios niveles. En cada uno de estos niveles de prueba, se podrán ejecutar diferentes tipos de prueba tales como: pruebas funcionales, no funcionales, de arquitectura y asociadas el cambio de los productos (Zapata, 2013).

A continuación, una breve descripción de cada nivel de prueba:

Pruebas unitarias: este tipo de pruebas son ejecutadas normalmente por el equipo de desarrollo, básicamente consisten en la ejecución de actividades que le permitan verificar al desarrollador que los componentes unitarios están codificados bajo condiciones de robustez. Adicionalmente, las pruebas sobre componentes unitarios, suelen denominarse pruebas de módulos o pruebas de clases (Zapata, 2013).

Pruebas de integración: este tipo de pruebas son ejecutas por el equipo de desarrollo y consisten en la comprobación de que funcionan de manera correcta elementos del software que interactúan entre sí (Zapata, 2013).

Pruebas de sistema: este tipo de pruebas deben ser ejecutadas idealmente por un equipo de pruebas ajeno al equipo de desarrollo, una buena práctica en este punto corresponde a la tercerización de esta responsabilidad. La obligación de este equipo, consiste en la ejecución de actividades de prueba en donde se debe verificar que la funcionalidad total de un sistema fue implementada de acuerdo a los documentos de especificación definidos en el proyecto (Zapata, 2013).

Pruebas de aceptación: en este nivel se verifica el comportamiento del sistema frente a los requisitos del cliente, generalmente participa el mismo cliente o los usuarios (Zapata, 2013).

3.4.2 Métodos de prueba

A nivel internacional en el desarrollo de software, se reconocen dos métodos de prueba fundamentales: caja blanca y caja negra. A continuación, se describen a continuación ambos métodos.

Caja blanca: Se comprueban los caminos lógicos de software, se examina el estado del programa en varios puntos para determinar si el estado real coincide con el esperado.

Caja negra: Se centra en lo que *"se quiere"* de un módulo o sección específica de un software, es decir, es una manera de encontrar casos específicos en ese módulo que atiendan a su especificación. Las pruebas de caja negra se limitan a que el probador compruebe con *"datos"* de entrada y estudie la salida del sistema, sin preocuparse de lo que ocurre en el interior. Éstas, principalmente, se centran en módulos de interfaz de usuario (pantalla, ficheros, canales de comunicación, entre otros) pero suelen ser útiles en

cualquier módulo ya que todos o la mayoría tienen datos de entrada y salida que se pueden comprobar y verificar (Globe Testing, 2016).

Para desarrollar el método de caja negra existen varias técnicas, entre ellas están (S. Pressman, 2000):

- ✓ Técnica de la Partición de Equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- ✓ Técnica del Análisis de Valores Límites: esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- ✓ Técnica de Grafos de Causa-Efecto: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

En la presente investigación se utiliza la técnica Partición de Equivalencia, pues permite examinar los valores válidos e inválidos de las entradas existentes en el módulo. A continuación, se muestra el caso de pruebas (CP) correspondiente al RF 1 Respaldar recursos. Para consultar los restantes casos de pruebas ver Anexo 6.

Tabla 5. Caso de prueba RF Respaldar recursos

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción respaldo de información a recursos.	Permite respaldar los recursos multimedias de un curso	1-Se crea entidad de respaldo de información. 2-Se crea el fichero Zip contenedor del respaldo información en el directorio upload/backups. 3-Muestra mensaje ¨Respaldo realizado con éxito¨. 4- Muestra el listado de respaldos de información actualizado.	Menú de Opciones del Curso/Opciones de respaldo/Realizar respaldo/Respaldar recursos /Realizar respaldo

3.5 Resultados obtenidos

Durante el proceso de validación del componente de respaldo de información a cursos en la plataforma educativa Zera 2.0, fueron aplicadas 18 pruebas unitarias para ello se utilizó la librería PHPUnit

para la comprobación del correcto desempeño de las funcionalidades implementadas. A continuación, la Figura 9 muestra el resultado de la realización de dichas pruebas.

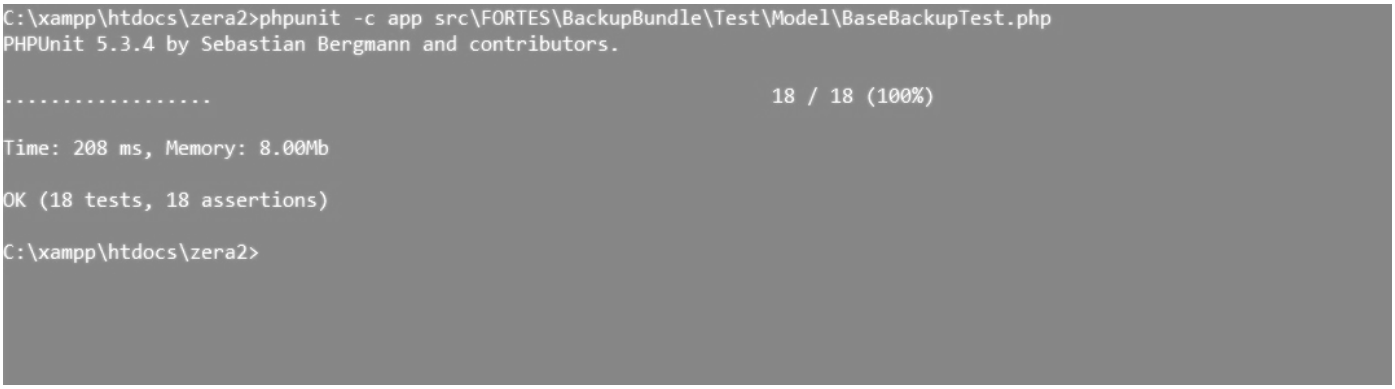


Figura 9. Pruebas unitarias

Capítulo 3. Implementación y pruebas

En la aplicación del método de caja negra fueron aplicadas 2 iteraciones, estas permitieron obtener un conjunto de No Conformidades (NC), en la 1ra iteración, las no conformidades fueron agrupadas en la Tabla 8, evaluadas en un rango comprendido entre: Alta, Media, Baja y No procede. A continuación, se presentan los resultados arrojado por estas pruebas:

Tabla 8. No conformidades detectadas en la 1ra iteración

No. CP	Caso de Prueba	No conformidades				
		Alta	Media	Baja	No procede	Total
1	Realizar respaldo	-	-	5	-	5
2	Importar respaldo	-	-	2	-	2
3	Eliminar respaldo	-	-	1	-	1

[2]

Para verificar las No Conformidades (NC) detectadas se presenta una tabla con los siguientes datos: el Requisito funcional (RF), NC detectada, Descripción (clasificada en Alta, Media o Baja) y estado con respecto a la Solución (RA: Resuelta y Aprobada por el revisor, PD: Pendiente por solución del equipo de desarrollo, NP: No Procede, AV: Aplazada para resolver en próximas versiones).

Tabla 9: Descripción de las No conformidades detectadas en la 1ra iteración.

No. NC	No. CP	Descripción	Complejidad	Estado
1	1	No se muestra mensaje con resultado de la acción realizada	Baja	RA

[2] No conformidad: Diferencia entre los resultados obtenidos y los compromisos adquiridos, de acuerdo con la meta, los criterios de desempeño o las evidencias preestablecidas.

2	1	No se muestra el listado de respaldos disponibles	Baja	RA
3	1	No está internacionalizada la opción respaldar recursos	Baja	RA
4	1	Respaldar está escrito en minúscula.	Baja	RA
5	1	No está internacionalizada la opción respaldar foros	Baja	RA
6	2	No está validada la subida de solo ficheros Zip	Baja	RA
7	2	No se muestra el listado de respaldos disponibles	Baja	RA
8	3	No se muestra el listado de respaldos disponibles	Baja	RA

Las no conformidades detectadas en la primera iteración fueron resueltas. Posteriormente se realizó la segunda iteración obteniéndose 0 no conformidades, se minimizó así la ocurrencia de errores en el sistema. A continuación, la Figura 10 muestra los resultados obtenidos en cada una de las iteraciones.

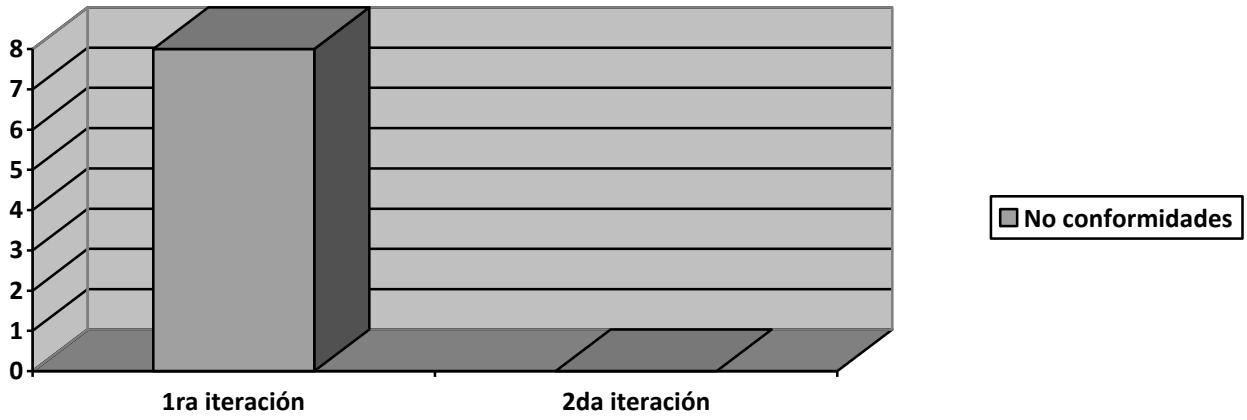


Figura 10. Resultados de la aplicación del método de caja negra

Capítulo 3. Implementación y pruebas

3.6 Conclusiones parciales

El diseño de los diagramas de componentes permitió obtener una vista del sistema, con la organización y las dependencias entre los diferentes elementos que componen la solución propuesta. La aplicación de estándares de codificación garantizó la legibilidad del código lo que favorece el soporte al componente de respaldo a cursos en la plataforma educativa Zera 2.0. Además, los diferentes tipos de pruebas aplicados permitieron la detección de un grupo de no conformidades las cuáles fueron resueltas, obteniéndose finalmente un producto de software con las funcionalidades requeridas.

CONCLUSIONES

Con la culminación de la presente investigación se obtuvo como resultado principal, la implementación del Componente de respaldo de información a cursos para la plataforma educativa Zera 2.0. Asimismo:

- El estudio realizado sobre los sistemas de respaldos a cursos en otras plataformas sirvió de apoyo para la comprensión del procedimiento de realización de respaldos de información.
- El esbozado del modelo de dominio permitió mayor entendimiento de la lógica del negocio. Lo que favoreció el análisis y diseño del componente de respaldos a cursos en la plataforma educativa Zera 2.0.
- El empleo de la metodología, las herramientas, tecnologías y lenguajes de desarrollo seleccionados, permitieron la generación de toda la documentación y artefactos que garantizan el entendimiento de la propuesta de solución y posibilitaron la implementación del componente de respaldo a cursos en la plataforma educativa Zera 2.0.
- La implementación del Componente de respaldo de información a cursos en la plataforma educativa Zera 2.0 contribuye en la reutilización de materiales educativos virtuales. Además, brinda un mecanismo de seguridad ante eventuales pérdidas de información y reduce costos en tiempo y esfuerzo en la creación de cursos virtuales en el sistema.
- La realización de las pruebas al componente para respaldo de información a cursos en la plataforma educativa Zera 2.0, permitió detectar y corregir errores presentes en la propuesta de solución.

Recomendaciones

A partir de los resultados obtenidos en la investigación, se proponen las siguientes recomendaciones:

- Extender el componente de respaldo a otras áreas del curso y de la plataforma.
- Incorporar un módulo que permita exportar los cursos y sus contenidos bajo estándares internacionales.
- Generar un manual para la extensión a otras áreas del componente de respaldo de información a cursos en la plataforma educativa Zera 2.0

Referencias bibliográficas

Globe Testing. 2016. *Pruebas de software.* 2016.

Álvarez, Diego Macías. 2010. *Plataformas de enseñanza virtual libres y sus características de extensión: Desarrollo de un bloque para la gestión de tutorías en Moodle.* Madrid : s.n., 2010.

Ana Moreno M., Natalia Juristo, y Sira Vegas. 2006. *TÉCNICAS DE EVALUACIÓN DE SOFTWARE.* 2006.

Análisis Comparativo de las Plataformas Educativas Virtuales Moodle y Dokeos. **R. E. Lizarraga, A. Z Colado, J. F. P Garzon. 2013.** s.l. : Revista Iberoamericana para la Investigación y el Desarrollo Educativo, 2013.

Apache software foundation. 1999. Apache. [En línea] 1999. apache.org.

Beatriz Sandia E, C Montilva, Jonás y Barrios. 2005. *Cómo evaluar cursos en línea.* 2005.

Beck, Kent. 2000. *Extreme Programming Explained. Embrace Change, Pearson Education.* s.l. : Addison Wesley, 2000.

C.Mateu. 2004. *Desarrollo de aplicaciones web.* 2004.

CMS, LMS LCMS. Definición y diferencias: Explicación de los conceptos CMS, LMS y LCMS, dando a conocer sus diferencias. **Mayor, Alicia Cañellas. 2011.** 251-252, s.l. : Primeras Noticias: Comunicación y Pedagogía, 2011.

Cockbun, A. 2001. *Agile Software Development.* s.l. : Addison-Wesley, 2001.

Cómo evaluar cursos en línea. **E Beatriz Sandia., Montilva C, Jonás y Barrios Albornoz, Judith del Rosario. 2006.** Mérida : Educere, 2006.

Construyendo y aprendiendo con el computador. **Sánchez, Jaime. 1999.** Santiago de Chile : s.n., 1999. 956-291 -177-2.

Cruz, Juan Carlos Delgado. 2008. *Salva Automática Centralizada.* 2008.

Definicion.org. . 2008. [En línea] 2008. http//www.dedinicionde.com.

Doctrine-Project. 2006-2016. Doctrine. [En línea] 2006-2016. http://www.doctrine-project.org/.

Eguiluz, Javier. 2016. *Introducción a JavaScript.* 2016.

Fabien Potencier, François Zaninotto. 2005. *Symfony 1.1, la guía definitiva.* 2005.

Referencias Bibliográficas

Gil, Manuel Torres. 2016. *Fundamentos del diseño de software.* 2016.

Glossary. 1996. Glossary. [En línea] 23 de agosto de 1996. [Citado el: 10 de febrero de 2016.] http://glossary.its.bldrdoc.gov/fs-1037/dir-004/_0509.htm.

HP Storage Mirroring. **Sánchez, Andrés. 2007.** 2007.

I Jacobson, G Booch and J Rumbaugh. 2000. *El proceso unificado de desarrollo de software.* 2000.

I. I. Santiesteban, M. Medina, J.L Piña, Y. V Garrido. 2011. *Desarrollo de funcionalidades que faciliten al docente su preparación y el control del aprendizaje de los estudiantes en la plataforma educativa.* 2011.

Inc.Object Management Group®. 2004. UML. [En línea] 2004. [Citado el: 10 de Febrero de 2016.] http://www.uml.org/.

J Highsmith, K Orr. 2000. *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems.* s.l. : Dorset House, 2000.

J.F GARCÍA PEÑALVO, J.GARCÍA CARRASCO. 2001. *Los espacios virtuales educativos en el ámbito de Internet: Un refuerzo a la formación tradicional, Teoría de la Educación. Educación y Cultura en la Sociedad de la Información.* 2001.

K Schwaber, M Beedle, R.C. Martin. 2001. *Agile Software Development with SCRUM.* . s.l. : Prentice Hall, 2001.

Laboratorio Nacional de Calidad del Software de INTECO. 2009. *Ingeniería del software: Metodologías y ciclos de vida.* 2009.

Las nuevas tendencias en respaldo y recuperación de datos. **Abad, Eduard. 2005.** 2005.

Las plataformas de aprendizajes, una alternativa a tener en cuenta en el proceso de enseñanza aprendizaje. **Rivero, Anaydal Fernández.** s.l. : Revista Cubana de Informática Médica, Vol. 6.

León, Felipe Lozano. 2012. *MSF.* 2012. Presentacion MSF. 2012.

Malbernat, Lucía Rosario. 2008. *Cómo elegir una plataforma para e-learning: El problema de los medios y las plataformas instruccionales. Calidad y Materiales educativos y Herramientas tecnológicas en Educación a Distancia.* 2008.

Márques, Pérez. 1999. *Concepción del software educativo.* Barcelona : Biblioteca Virtual de Derecho, Economía y Ciencias Sociales., 1999.

Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). **Patricio Letelier, Mª Carmen Penadés. 2006.** 26, Valencia : s.n., 2006, Vol. 05.

Referencias Bibliográficas

Métodos de desarrollo de software: El desafío pendiente de la estandarización. Software . **Bustos, Ricardo A Gacitúa. 2003.** Chile : s.n., 2003, Vol. 12. 0717-196X.

Métodos de desarrollo de software: El desafío pendiente de la estandarización. Software. **Bustos, Ricardo A Gacitúa. 2003.** 1, Chile : s.n., 2003, Vol. 12. 0717-196X.

Microsoft. 2016. [En línea] 2016. https://msdn.microsoft.com/es-s/library/aa291591%28v=vs.71%29.aspx.

Netbeans. NetbeansIDE. [En línea] www.netbeans.org.

Perez, J. E. 2008. *Introducción a CSS.* 2008.

Plataformas abiertas de e-learning para el soporte de contenidos educativos abiertos. **Boneu, Joseph. 2007.** Catalunya : Revista de Universidad y Sociedad del Conocimiento, 2007, Vol. 4.1.

Roberth G. Figueroa, Camilo J. Solís , Armando A. Cabrera. METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES. *METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES.* Loja : s.n.

Rodriguez, Tamara. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana : s.n., 2015.

Roger, Pressman. 2000. *Ingeniería del Software: Un enfoque práctico.* 2000.

Sakai Project. 2010. Sakai Project- an Open Source suite of learning, portfolio, library and project tools. [En línea] 2010. http://sakaiproject.org/.

Scrum Manager®. 2014. Scrum Manager Body of Knowledge. [En línea] marzo de 2014. http://www.scrummanager.net/bok/index.php?title=Historia_de_usuario.

Sommerville, I. 2005. *Ingeniería del Software.* 2005.

Sommerville, Ian. 2005. *Ingeniería de Software 7ma edición.* s.l. : Pearson-Addison Wesley, 2005.

Storage Technology Inforamation. 2016. Storage Technology Inforamation. [En línea] 2016. http://searchstorage.techtarget.com/sDefinition/0,,sid5_gci211633,00.html..

Synfony. 2010. Synfony. [En línea] 2010. Synfony.es.

The Free Dictionary. 2006. The Free Dictionary. [En línea] 2006. http://www.thefreedictionary.com/backup..

UML y patrones. **Larman, Craig. 2003.** 2003.

Referencias Bibliográficas

Universidad Unión Bolivariana. 2014. Carrera de Ingeniería de Sistemas. METODOLOGIAS AGILES "PROCESO UNIFICADO AGIL (AUP). [En línea] 2014. http://ingenieriadesoftware.mex.tl/images/18149/METODOLOGIAS%20AGILES.pdf.

Visual Parading. 2016. Visual Parading Web site. [En línea] 2016. [Citado el: 10 de febrero de 2016.] https://www.visual-paradigm.com/.

Welicki, León. 2016. *Patrones y Antipatrones: una Introducción - Parte II.* [En línea] 2016. [Citado el: 1 de Marzo de 2016.] https://msdn.microsoft.com/es-es/library/bb972251.aspx.

Wikipedia. 2016. Wikipedia. *Copia de Seguridad.* [En línea] 2016. http://es.wikipedia.org/wiki/Copia_de_seguridad..

Young, R. 2004. *The Requirements Engineering Handbook.* 2004.

Zapata, Javier. 2013. *Pruebas de Software.* 2013.