Universidad de las Ciencias Informáticas

Facultad 4

Componente de Integración con Sistemas de Almacenamiento en la Nube.

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

Autor:

Rosbel Caballero Ramírez

Tutor:

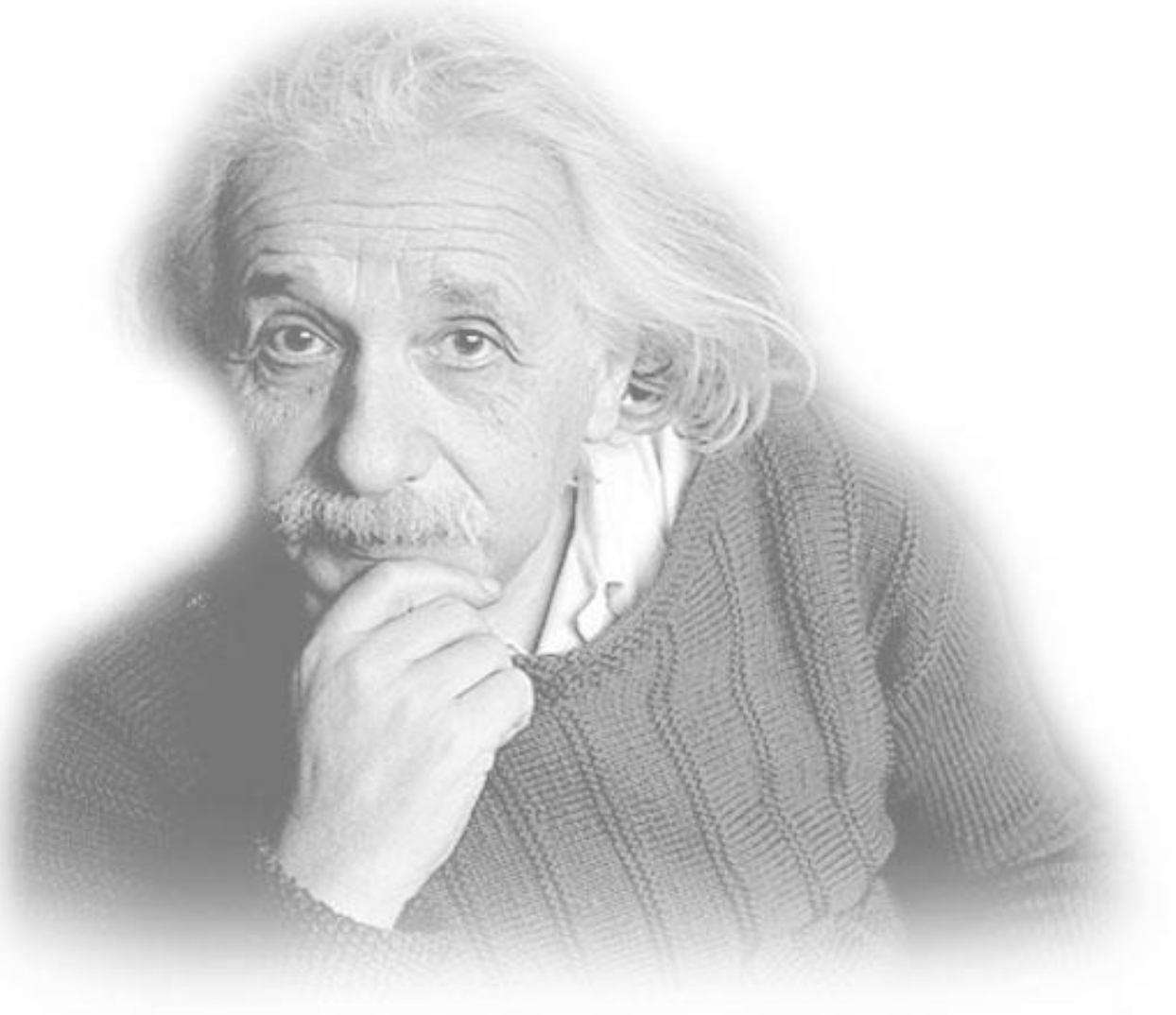
MSc. Yerandy Manso Guerra

Tutor:

Ing. Elías Bello Camps

La Habana, junio del 2016

"Año 58 de la Revolución"



Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad.

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaro ser el único autor de la presente tesis **Componente de Integración con Sistemas de Almacenamiento en la Nube** y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____

Firma del autor

Rosbel Caballero Ramírez

_____ _____

Firma del Tutor **Firma del Tutor**

MSc. Yerandy Manso Guerra Ing. Elías Bello Camps

Nombre y Apellidos: MSc. Yerandy Manso Guerra

Correo electrónico: ymguerra@uci.cu

Nombre y Apellidos: Ing. Elías Bello Camps

Correo electrónico: ebcamps@uci.cu

Nombre y Apellidos: Rosbel Caballero Ramírez

Correo electrónico: rcramirez@estudiantes.uci.cu

A mi mamá por darme tanto amor a lo largo de todos estos años, por todo el sacrificio que ha pasado para que yo llegara a ser lo que hoy soy.

A mi papá por inculcarme buenas ideas y disciplina, por su amor y ejemplo a seguir.

A mi hermana por su amor y cariño.

A mi padrastro por ser mi segundo padre, a mi madrastra por su cariño y amistad.

A mi hermano Quentin, mi tía Clarita y mi abuelo Ariel, aunque ya no estén físicamente en este mundo los recuerdo y los quiero por lo que significan en mi vida.

A todos mis familiares, primos, tíos, abuelos, todos ellos me han demostrado el verdadero valor de la palabra familia.

A todos mis amigos que hice en esta escuela, a pesar de conocerlos desde hace poco tiempo han hecho que los quiera como si fueran mi familia, a todos los que me ayudaron a lo largo de este difícil pero victorioso camino, especialmente al flaco, Alexis, Carlos, Yasmany, Jose, Yosle, Luis, Raúl, Yaniel, Alberto y tantos más.

A mis amigos de Buenaventura y los del IPVCE, por todos los momentos compartidos, a Bauta, Yordany, Lázaro, Alejandro, el tata, Jose y tantos más.

Agradezco especialmente a Alberto, por toda la ayuda y los conocimientos que me brindó, a Ángel por sus instrucciones, sin ellos no hubiera sido posible este trabajo.

A mis tutores por el tiempo dedicado.

A mi novia Verónica por toda su ayuda y comprensión.

En fin, a todos aquellos que me han ayudado en todos estos años de mi vida, y a los que han puesto su granito de arena para que este trabajo de diploma salga, mi agradecimiento eterno a todos ellos.

A mis padres, porque cada triunfo mío es suyo también y viceversa.

A mi hermana, que apenas comienza a andar en esta vida.

A todos mis amigos que han hecho cada preocupación mía en suya.

A todas las personas que me han brindado su amistad verdadera y su ayuda.

A todos aquellos que dudan que con voluntad, trabajo y sacrificio se alcanzan hasta las metas más difíciles.

Resumen

El uso de las Tecnologías de la Información y las Comunicaciones en la educación han traído consigo notables mejoras al Proceso de Enseñanza-Aprendizaje, permitiendo potenciar y posibilitar el intercambio del usuario con materiales educativos mediante el manejo de diversas herramientas informáticas. Las plataformas educativas reúnen un conjunto de herramientas para facilitar el proceso de enseñanza-aprendizaje dentro y fuera del aula, en las cuales los recursos educativos juegan un papel fundamental. El creciente volumen de estos recursos generados ha traído consigo la búsqueda de nuevas vías para el almacenamiento y disponibilidad de estos, en la cual el Almacenamiento en la Nube constituye una opción viable. La utilización de estos servicios permite almacenar los recursos educativos y acceder a estos desde cualquier dispositivo conectado a internet en cualquier momento y lugar. En la Universidad de las Ciencias Informáticas, específicamente en el Centro de Tecnologías para la Formación, se desarrolla la segunda versión de la plataforma educativa ZERA. En esta se genera un gran volumen de recursos educativos, que en ocasiones no se encuentran disponibles, no existe una vía fácil para compartirlos y ocupan mucho espacio en el servidor donde se encuentra la plataforma. La presente investigación tiene como objetivo desarrollar un componente para integrar ZERA 2.0 con Sistemas de Almacenamiento en la Nube que permita cargar, almacenar, descargar, compartir y visualizar recursos educativos desde o hacia la nube.

Palabras Claves: almacenamiento en la nube, componente, integrar, plataforma educativa, recursos educativos.

Índice General

Introducción.....1

Capítulo 1. Fundamentación Teórica.....6

1.1 Plataforma educativa.....6

1.1.1 Gestión de recursos educativos en la plataforma educativa ZERA.....7

1.2 Computación en la nube.....8

1.2.1 Modelos de Servicio de la computación en la nube.....10

1.2.2 Tipos de computación en la nube.....10

1.3 Almacenamiento en la Nube.....11

1.3.1 Sistemas de Almacenamiento en la Nube.....12

1.4 Integración de plataformas educativas con Sistemas de Almacenamiento en la Nube.....19

1.5 Metodología de desarrollo de software.....21

1.6 Tecnologías y herramientas.....23

1.6.1 API.....23

1.6.2 SDK.....24

1.6.3 *Componente*.....24

1.6.4 Frameworks.....25

1.6.5 Symfony.....26

1.6.6 Doctrine.....26

1.6.7 Lenguaje de modelado UML.....27

1.6.8 Herramienta CASE: Visual Paradigm.....27

1.6.9 Herramienta IDE: NetBeans.....28

1.5 Conclusiones parciales.....28

Capítulo 2. Características y Diseño de la solución propuesta.....29

2.1 Modelo de dominio.....29

2.1.1 Conceptos del dominio.....29

2.1.2 Diagrama del modelo de dominio 30
2.2 Requerimientos funcionales del componente 30
2.3 Requerimientos no funcionales del componente 31
2.4 Descripción de los actores del sistema 32
2.5 Historias de usuario 32
2.6 Patrón arquitectónico 34
2.7 Patrones de diseño empleados 35
2.7.1 Patrones GRASP 35
2.7.2 Patrones GOF 36
2.8 Modelado de diseño 37
2.8.1 Diagramas de clases del diseño 38
2.8.2 Diagramas de secuencia del diseño 39
2.9 Conclusiones parciales 39
Capítulo 3: Implementación y Prueba de la solución propuesta 41
3.1 Diagrama de componentes 41
3.2 Estándares de codificación 42
3.3 Resultado de la Implementación 42
3.4 Pruebas de software 43
3.4.1 Niveles de pruebas 44
3.4.2 Métodos de pruebas 44
3.4.3 Técnicas empleadas 45
3.4.4 Diseño de casos de pruebas 45
3.4.5 Análisis de los resultados 47
3.5 Conclusiones parciales 48
Conclusiones Generales 49
Recomendaciones 50

Índice de Tablas

Tabla 1.Comparación entre los Sistemas de Almacenamiento en la Nube.....19

Tabla 2. Actores del Sistema y su descripción.....32

Tabla 3. HU Añadir archivo almacenado en los Sistemas de Almacenamiento en la Nube como recurso educativo en los cursos de ZERA 2.0.....33

Tabla 4. DCP Añadir archivo almacenado en los Sistemas de Almacenamiento en la Nube como recurso educativo en los cursos de ZERA 2.0.....46

Tabla 5. Resultados de las pruebas.....48

Índice de Figuras

Figura 1. Modelo del dominio. 30

Figura 2 Funcionamiento interno de Symfony 2 35

Figura 3. DCD Añadir archivo almacenado en los Sistemas de Almacenamiento en la Nube como recurso educativo en los cursos de ZERA 2.0. 38

Figura 4. DSD Añadir archivo almacenado en los Sistemas de Almacenamiento en la Nube como recurso educativo en los cursos de ZERA 2.0. 39

Figura 5. Diagrama de componentes general. 41

Figura 6. Interfaz principal del componente. 43

Introducción

La revolución científico técnica iniciada en el siglo XX y sus consecuencias extraordinarias en el desarrollo tecnológico actual y de las Tecnologías de la Información y las Comunicaciones (TIC) son responsables de aumentos en la productividad en los más variados sectores. Las TIC son los sistemas, herramientas y aplicaciones que operan de todas las formas posibles sobre la información a través de máquinas y se fundamentan en la posibilidad de transformar cualquier "información", por lo que debe convertirse en datos digitales (1).

En la educación, las TIC median en el aprendizaje pues abren espacios para buscar, procesar, aplicar la información, el conocimiento (2); propician el intercambio con otros, para aprovechar las potencialidades educativas que ofrecen. El empleo de este tipo de tecnologías tiene un gran impacto, fomentando el conocimiento científico y la instrumentación tecnológica en función de mejorar el Proceso de Enseñanza-Aprendizaje (PEA) en los diferentes niveles de la educación, dando paso al surgimiento del *e-learning* (3). De esta forma se añaden notables mejoras al PEA, rompiendo fronteras directas profesor-alumno, permitiendo potenciar y posibilitar el intercambio del usuario con materiales educativos mediante el manejo de diversas herramientas informáticas.

Una de las consecuencias más evidentes de la influencia de las TIC en la educación está en la forma en la que se generan los recursos educativos[1] y en los medios por los que se transmiten (4). De forma generalizada, se crea tanto por docentes como por estudiantes un gran volumen de recursos educativos en formato digital. Muchos de estos se almacenan en dispositivos o se transmiten a otros también en este formato, a través de plataformas virtuales u otros sistemas de comunicación. Sin embargo, todos estos recursos educativos generados no siempre se encuentran disponibles, por lo que para el máximo aprovechamiento de estos se impone lograr la ubicuidad[2] como principal característica, haciendo necesario que los sistemas se intercomuniquen y compartan recursos de manera eficiente y transparente para los distintos usuarios.

Los estándares para el desarrollo del *e-learning* están marcando la pauta para crear sistemas que integren las aplicaciones para los procesos de enseñanza y aprendizaje en línea, en las que los

[1] **Recursos educativos:** Cualquier material que en un contexto educativo determinado, es utilizado con una finalidad didáctica o para facilitar el desarrollo de las actividades formativas. (65)

[2] **Ubicuidad:** Del término ubicuo tiene origen latino (ubique) y significa "en todas partes".

contenidos puedan ser reutilizados y compartidos, entre personas y entre sistemas (5). Esto ha traído consigo que las plataformas educativas busquen la forma de hacer de sus recursos educativos más presentes en la vida del estudiante a través de nuevas tecnologías relacionadas con internet. En este nuevo espacio para la enseñanza, un compañero de clases puede ser un miembro también de su comunidad virtual y utilizar la tecnología que se encuentra disponible para compartir sus recursos, intercambiarlos, trabajar colaborativamente, desde cualquier lugar o en cualquier momento y una vía para alcanzar este objetivo puede estar en utilizar servicios de Almacenamiento en la Nube.

El estudio de la vinculación de las distintas herramientas que hacen uso de recursos educativos u otros archivos generados en el proceso de enseñanza aprendizaje con los Sistemas de Almacenamiento en la Nube está motivado por la importancia que toma la información en la educación a distancia y la evolución en la forma de manipularla, además de la posibilidad de integración de estas aplicaciones.

La computación en la nube permite el consumo de recursos y servicios hardware/software a través de internet (6) (7). La computación en la nube y dentro de esta, el Almacenamiento en la Nube, responde a las necesidades del aumento de dispositivos conectados a internet y el creciente volumen de datos manejados, posibilitando disponer de la información en cualquier momento y lugar (8). Esto ha tenido como consecuencia la apertura del mercado, ofreciendo aplicaciones basadas en la nube como One Drive, Google Drive, Dropbox, Shared, solo por citar algunos, los que brindan cierta capacidad de almacenamiento para alojar archivos, incluso de manera gratis, y permite a otras aplicaciones conectarse y utilizar sus servicios a través de una *Application Programming Interface* (API)[3].

En el mundo existen diferentes plataformas educativas integradas a Sistemas de Almacenamiento en la Nube para almacenar los recursos educativos que se generan. De esta manera, estudiantes y profesores pueden tener acceso de forma flexible a los recursos educativos almacenados en la nube a través de las plataformas educativas, o del propio Sistema de Almacenamiento en la Nube, por medio de cualquier dispositivo conectado a internet, desde su casa, institución educativa, habitación o cualquier lugar, y lograr una rápida y eficiente comunicación, colaboración, intercambio y difusión de documentos, notas, audio/vídeo entre otros tipos de información.

En la Universidad de las Ciencias Informáticas (UCI), específicamente en el Centro de Tecnologías para la Formación (FORTES) se ha desarrollado la plataforma educativa ZERA para cursos en línea

[3] **API:** traducido al español Interfaz de Programación de Aplicaciones, es un conjunto de instrucciones de programación y de normas de acceso a una aplicación de software basada en web o herramienta web.

para la educación a distancia. Para el desarrollo de su segunda versión se realizó un análisis de las principales características y deficiencias del uso de los recursos educativos generados en los cursos.

Algunas de las limitaciones identificadas fueron la poca reutilización de los recursos educativos. En ocasiones, se creaban y/o utilizaban, y quedaban en el olvido, pudiendo ser reutilizados por el mismo usuario o por otros con la misma necesidad. Además, se limitaba el acceso y la disponibilidad de estos. No se podía tener acceso a menos que se estuviera conectado a la plataforma y el curso en donde se utilizó estuviera disponible, y en ocasiones, se eliminaban de forma intencional para liberar espacio de almacenamiento en el servidor donde se encuentra alojada la plataforma. Se dificultaba en gran medida la transmisión y el intercambio de lo utilizado, pues en caso requerido, se debía realizar a través de la copia del recurso educativo en un dispositivo de almacenamiento. A consecuencia de lo antes expuesto, se detectó como una limitación general que la plataforma no contaba con la interoperabilidad[4] con Sistemas de Almacenamiento en la Nube, como una vía para dar solución a las limitaciones encontradas, identificándose así para la presente investigación como **problema científico** ¿Cómo permitir que desde la plataforma educativa ZERA 2.0 se puedan cargar, almacenar, descargar, compartir y visualizar archivos desde o hacia los Sistemas de Almacenamiento en la Nube?

Con vista a la solución del problema se define como **objeto de estudio** la integración de plataformas educativas con Sistemas de Almacenamiento en la Nube.

La presente investigación se encuentra enmarcada en el **campo de acción**: Componente de Integración con Sistemas de Almacenamiento en la Nube.

Como **objetivo general** se propone desarrollar un componente para integrar ZERA 2.0 con Sistemas de Almacenamiento en la Nube que permita cargar, almacenar, descargar, compartir y visualizar recursos educativos desde o hacia la nube.

Como **objetivos específicos** se definen:

- Elaborar el marco teórico de la investigación para sustentar conceptos, herramientas y tecnologías a utilizar.
- Realizar la caracterización y el diseño del Componente de Integración con Sistemas de Almacenamiento en la Nube.

[4] **Interoperabilidad**: habilidad de dos o más sistemas o componentes para intercambiar información y utilizar esta información intercambiada (69).

- Realizar la implementación y prueba del Componente de Integración con Sistemas de Almacenamiento en la Nube.

Para dar cumplimiento a los objetivos específicos se trazaron las siguientes **tareas científicas**:

- Fundamentación teórica de la investigación.
- Levantamiento de requisitos para desarrollar la solución.
- Análisis de la arquitectura del proyecto ZERA 2.0.
- Estudio del *frameworks* de desarrollo Symfony.
- Estudio de las API y los Kit de Desarrollo de Software (SDK)[5] a utilizar.
- Diseño del componente de integración.
- Implementación del componente de integración.
- Aplicación de las pruebas de calidad.

Para llevar a cabo estas tareas se emplearán métodos empíricos y teóricos de la investigación científica. Dentro de los **métodos empíricos** se utilizará la entrevista, la cual posibilitará obtener la información referente a cómo se espera que se integre el componente y las funcionalidades que brindará. Las entrevistas serán realizadas a todo el personal involucrado en el desarrollo de la plataforma para conocer la opinión de las personas que van a hacer uso del componente y poder complementar los requerimientos necesarios. Todo esto permitirá obtener información y criterios referentes al tema, y corregir malos conceptos que se puedan tener para cumplir con los requerimientos deseados.

En los **métodos teóricos** se utilizará el análisis y la síntesis, para el análisis de condiciones específicas y poder llegar a conclusiones e inferencias de ideas generales. La inducción y la deducción se utilizarán en diferentes etapas de la investigación, especialmente cuando a partir de determinadas situaciones e ideas generales se deduzcan conocimientos particulares que redunden en beneficio de la teoría y la práctica. El histórico – lógico para centrarse en los problemas que a lo largo de la historia ha presentado la integración con Sistemas de Almacenamiento en la Nube; posibilita rectificarlos y buscar entre todas las soluciones existentes la más óptima.

[5] **SDK:** El *software development kit* o kit de desarrollo de software traducido al español es un conjunto de herramientas de desarrollo de software que permite a un programador crear programas y aplicaciones para un sistema o plataforma concretos.

Una vez concluida la investigación se esperan como **posible resultado** el Componente de Integración con Sistemas de Almacenamiento en la Nube para ZERA 2.0 que permita cargar, descargar, compartir y visualizar recursos educativos desde o hacia los Sistemas de Almacenamiento en la Nube.

Este trabajo se estructura en tres capítulos:

Capítulo 1: "Fundamentación teórica", se abordan los principales temas asociados al objeto de estudio, así como las distintas soluciones similares desarrolladas en el mundo para llegar a la propuesta de solución que sustenta la investigación. Además, se caracterizan las tecnologías y la metodología, además de las herramientas que se utilizan para el desarrollo del componente.

Capítulo 2: "Características y Diseño de la solución propuesta", se describe el modelo de dominio y los conceptos asociados al mismo, los requerimientos funcionales y no funcionales para dar cumplimiento a la investigación, y las historias de usuario para modelar los requisitos funcionales. Además, se define el patrón arquitectónico, los patrones de diseño a utilizar y se realiza el modelado de diseño.

Capítulo 3: "Implementación y Prueba de la solución propuesta", se describe cómo se implementan los elementos del diseño. Además, se define la estrategia de pruebas a emplear y se muestran los resultados de los diferentes tipos de pruebas realizadas.

Capítulo 1. Fundamentación Teórica

En el presente capítulo se hace referencia a la fundamentación teórica del trabajo, para proporcionar un mejor entendimiento de la investigación a partir del conocimiento disponible que pueda ayudar a conformar la solución propuesta. Se realizarán análisis como el de computación en la nube, Almacenamiento en la Nube, sus ventajas y desventajas, un estudio de los principales Sistemas de Almacenamiento en la Nube más conocidos, para seleccionar de estos los convenientes para la plataforma de aprendizaje ZERA 2.0. Para lograr una mejor comprensión de las características del componente a implementar se estudiarán integraciones similares de plataformas educativas son Sistemas de Almacenamiento en la Nube. Además, se conocerá la metodología, tecnologías y herramientas empleadas en el desarrollo del componente.

1.1 Plataforma educativa

Existen diversas denominaciones cuando se refiere al término de plataforma educativa o nombrado por otros como Plataforma Tecnológica Educativa, entre las que se pueden encontrar:

- *Course Management System* (CMS). Sistema de Gestión de Cursos.
- *Integrated Learning System* (ILS). Sistema Integrado de Aprendizaje.
- *Learning Management System* (LMS). Sistemas de Gestión de Aprendizaje.
- *Learning Platform* (LP). Plataforma de Aprendizaje.
- *Learning Support System* (LSS). Sistema de Soporte de Aprendizaje.
- *Managed Learning Environment* (MLE). Ambiente Controlado de Aprendizaje
- *Virtual Learning Environment* (VLE). Entorno Virtual de Aprendizaje

En el Reino Unido la Agencia Educativa Británica para Comunicaciones y Tecnología (BECTA) acuñó la expresión “Plataforma Tecnológica Educativa” para englobarlos. En los Estados Unidos se utilizan términos como el de LMS. En general, todos estos términos hacen referencia a lo mismo, aunque cada uno tiene sus particularidades. (5)

Una plataforma educativa virtual, es un entorno informático en el que se encuentran muchas herramientas agrupadas y optimizadas para fines docentes. Su función es permitir la creación y gestión de cursos completos para internet sin que sean necesarios conocimientos profundos de programación. Estos sistemas tecnológicos proporcionan a los usuarios espacios de trabajo compartidos destinados al

intercambio de contenidos e información, incorporan herramientas de comunicación (chats, correos, foros de debate, videoconferencias, blogs, etc.) y, en muchos casos, cuentan con un gran repositorio de objetos digitales de aprendizaje desarrollados por terceros, así como con herramientas propias para la generación de recursos. (9)

Se define el término plataforma educativa como una amplia gama de sistemas y herramientas dentro de las TIC que se utilizan en entornos de aprendizaje, instaladas en servidores, cuya función es la de poner a disposición del alumno y el profesor una serie de recursos informáticos para facilitar el proceso de enseñanza y aprendizaje dentro y fuera del aula (5) (10). Las plataformas buscan la creación de un espacio educativo y de aprendizaje en línea que ofrezca acceso a recursos educativos, herramientas de comunicación y un entorno colaborativo en el que desarrollar su trabajo.

1.1.1 Gestión de recursos educativos en la plataforma educativa ZERA

ZERA 1.0 es una plataforma educativa que nació producto del proyecto Alfaomega desarrollado en el centro FORTES, concebida para administrar, controlar y distribuir las actividades de formación de las instituciones relacionadas con la misma. Desde su primera versión, esta ha seguido las características de los LMS, los cuales permiten administrar, distribuir, monitorear, evaluar y apoyar las diferentes actividades previamente diseñadas dentro de un proceso de formación completamente virtual (*e-learning*), o de formación semi-presencial (11). En este tipo de sistemas se maneja un gran volumen de recursos educativos.

Los recursos educativos de la plataforma educativa ZERA 1.0 formaban parte de los contenidos publicados en los diferentes cursos. Estos podían ser creados, modificados, eliminados y publicados por el editor, usuario que poseía permisos para realizar todas estas funcionalidades. Por otra parte, los profesores tenían acceso a la modificación o eliminación de aquellos que fueran de su autoría. Hasta el momento, la publicación de dichos recursos se realizaba importando el archivo desde la pc en donde se encontraba el usuario. De esta forma, no se aprovechaban otros espacios en los cuales se almacenaban recursos educativos y que venían ganando en importancia y uso. Al no existir un servicio de almacenamiento externo a la plataforma que liberara a esta de espacio en el servidor, en el cual se pudiera acceder a los recursos educativos en cualquier momento y lugar, se dio a la tarea de integrar la nueva versión de ZERA con Sistemas de Almacenamiento en la Nube.

En la segunda versión de ZERA se mejora la gestión de los recursos educativos. Continúa con las características de LMS, enfocado a los cursos en línea para la educación a distancia, en donde se generan una gran cantidad de recursos educativos, pues se promueve la creación y utilización de estos por parte

de los estudiantes, profesores y editores de los cursos, como un método para la evaluación y participación o como un instrumento con una finalidad didáctica o formativa. En este caso, se desea que los implicados en este proceso tengan la oportunidad de seleccionar otros orígenes de sus recursos educativos como pueden ser los Sistemas de Almacenamiento en la Nube. Con la utilización de estos servicios se permitirá publicar o entregar un recurso educativo, o proporcionar un enlace para acceder a estos, y compartirlos con otras personas en la misma plataforma educativa.

1.2 Computación en la nube

No existe una definición aceptada universalmente; sin embargo, existen organismos internacionales cuyos objetivos son la estandarización de Tecnologías de la Información y en particular, de la computación en la nube. Uno de los organismos más reconocidos es el *National Institute of Standards and Technology* (**NIST**) y su *Information Technology Laboratory*, que define la computación en la nube como:

"Un modelo que permite el acceso bajo demanda a través de la Red a un conjunto compartido de recursos de computación configurables (redes, servidores, almacenamiento, aplicaciones y servicios) que se pueden aprovisionar rápidamente con el mínimo esfuerzo de gestión o interacción del proveedor del servicio"

La computación en la nube es la evolución de un conjunto de tecnologías que afectan al enfoque de las organizaciones y empresas en la construcción de sus infraestructuras de tecnologías de información. Al igual que ha sucedido con la evolución de la Web, la computación en la nube no incorpora nuevas tecnologías. Se han unido tecnologías potentes e innovadoras, para construir este nuevo modelo y arquitectura de la Web. (12)

La computación en la nube representa portabilidad, movilidad, optimización tanto en hardware como en software (13); se busca desplazar a los servidores físicos, y utilizar servidores virtuales mediante la nube, servicios dedicados al almacenamiento de información y aplicaciones que el usuario utiliza diariamente para el desarrollo de sus labores (14).

En general, computación en la nube define la arquitectura de red mediante la cual el usuario puede administrar y acceder a recursos que no se encuentran instalados o almacenados de forma local en el equipo de usuario. (14)

Ventajas

Entre las ventajas se mencionan las más importantes que permiten establecer y definir el Almacenamiento en la Nube como una solución eficaz y eficiente para los usuarios. Algunas de estas son (14):

- Reducción en el consumo de energía eléctrica.
- Costos bajos, en comparación a los costos tradicionales por aplicación y dispositivos de almacenamiento.
- Flexibilidad, se adapta a las necesidades de los usuarios.
- Adaptable a las necesidades del negocio.
- Servicio bajo demanda, el usuario puede acceder en todo momento y desde cualquier lugar, cuenta con una disponibilidad de 24/7.
- Particularmente, para los usuarios finales, la computación en la nube significa que no tienen que preocuparse por el mantenimiento de hardware o la compra de nuevos equipos, la obtención de licencias de software, actualización o mejora del software existente, la sincronización de datos, porque todos estos procesos están incluidos en el servicio “nube” (15).

Desventajas

Algunas de las desventajas de utilizar la computación en la nube son (14):

- Al almacenar la información en manos de terceros, los datos se pueden convertir en vulnerables y pueden sufrir algún tipo de acción no deseada, como puede ser la perdida de esta.
- La centralización de las aplicaciones y de los servicios hace que el cliente sea dependiente de los proveedores.
- Acceder a los servicios e información depende de una conexión a internet.
- La información reside en manos de terceros; el nivel de seguridad depende del proveedor, por lo que el nivel de vulnerabilidad puede ser alto en tanto a pérdida o robo de dicha información.
- La información debe viajar por algunos nodos hasta su destino final y esto puede constituir un punto de inseguridad.

1.2.1 Modelos de Servicio de la computación en la nube

Según el **NIST** se refieren a los servicios específicos a los que se puede acceder en una plataforma de computación en la nube (Software, Plataforma e Infraestructura como Servicios).

Software como servicio: Al usuario se le ofrece la capacidad de que las aplicaciones suministradas se desenvuelvan en una infraestructura de la nube, siendo las aplicaciones accesibles a través de un navegador web, como en el correo electrónico Web. Posiblemente, este es el ejemplo más representativo, por lo extendido de este modelo de servicio. El usuario carece de cualquier control sobre la infraestructura o sobre las propias aplicaciones, excepción hecha de las posibles configuraciones de usuario o personalizaciones que se le permitan (12).

Plataforma como servicio: Al usuario se le permite desplegar aplicaciones propias (adquiridas o desarrolladas por el propio usuario) en la infraestructura de la nube de su proveedor, que ofrece la plataforma de desarrollo y las herramientas de programación. En este caso, el usuario mantiene el control de la aplicación, aunque no de toda la infraestructura subyacente (12).

Infraestructura como servicio: El proveedor ofrece recursos como capacidad de procesamiento, de almacenamiento o comunicaciones, que el usuario puede utilizar para ejecutar cualquier software*;* desde sistemas operativos hasta aplicaciones.

En el modelo de software como servicio es en donde se encuentran los Sistemas de Almacenamiento en la Nube como Dropbox y One Drive. De esta forma se le ofrecen servicios a través de aplicaciones en la nube a los usuarios, en este caso para el almacenamiento de archivos.

1.2.2 Tipos de computación en la nube

En el modelo de despliegue, según el NITS, que se refiere a la posición (localización) y administración (gestión) de la infraestructura de la nube, donde existen cuatro modelos de computación en la nube:

Público: Servicio en el que se requiere poco control administrativo y que se puede acceder en línea por cualquier persona que esté autorizada. El Almacenamiento en la Nube pública utiliza un mismo conjunto de hardware para almacenar la información de varias personas, con medidas de seguridad y espacios virtuales para que cada usuario pueda ver específicamente la información que le corresponde. Este servicio es alojado externamente y se puede acceder mediante internet. (16)

Privado: Normalmente usado por empresas, es un sistema diseñado específicamente para cubrir las necesidades de la empresa o persona. En este modelo la empresa tiene el control administrativo y por lo tanto le es posible diseñar y operar el sistema de acuerdo a sus necesidades. (16)

Híbrido: Como su nombre indica, ofrece una combinación de nube pública y privada, de tal forma que los usuarios pueden personalizar las funciones y las aplicaciones para que se adapten mejor a sus necesidades. Un ejemplo típico de esta nube es que la información importante está alojada en la nube privada mientras que la información menos sensible se encuentre en la nube pública. (16)

Comunitario: Ha sido organizada para servir a una función o propósito común. Es preciso compartir objetivos comunes (misión, políticas, seguridad). Puede ser administrada bien por las organizaciones constituyentes, bien por terceras partes (16). Este modelo es definido por el NIST, aunque la mayoría de las organizaciones, proveedores y usuarios de la nube aceptan los tres modelos de despliegue: pública, privada e híbrida (12).

La nube pública se encuentra disponible al público en general a través de Internet. Los servicios de este tipo de nube pueden ser libres u ofrecidos a través de un modelo de pago por uso. Algunos de los más conocidos que serán tratados en esta investigación son:

- Dropbox, uno de los más populares ya que fue el primero en hacer famoso el uso del Almacenamiento en la Nube (17).
- Google Drive, servicio de almacenamiento de Google, originalmente denominado Google Docs.
- One Drive, servicio de Almacenamiento en la Nube de Microsoft.

1.3 Almacenamiento en la Nube

El Almacenamiento en la Nube es un nuevo modelo tecnológico que consiste en almacenar muchos datos en varios servidores virtuales, los cuales están administrados por terceros. Estos proveedores gestionan y operan grandes centros de datos y los usuarios finales compran o alquilan espacio dentro de sus servidores, dependiendo de las necesidades de cada uno. Los operadores de dicho centro, virtualizan los recursos de acuerdo a los requerimientos de los usuarios para que a su vez estos puedan utilizar el servicio para el almacenamiento de sus datos. (18) (19)

El Almacenamiento en la Nube es un servicio que se presta gracias a la infraestructura del proveedor de servicios para almacenar y acceder a archivos a través de internet como si se tratase de un disco duro local. (20)

Ventajas

Algunas de las ventajas que propician el uso del Almacenamiento en la Nube son (14):

- Ahorro basado en costos: las empresas ya no necesitan instalar costosos dispositivos físicos para el almacenamiento de su información.
- El costo se basa en el uso: cuando una empresa empieza sus actividades en la nube, el costo inicial es de cero, debido a que se paga lo que se utiliza.
- Elasticidad: la capacidad de almacenamiento se distribuye dependiendo de la necesidad de cada usuario lo que permite mayor escalabilidad de los recursos.
- Accesibilidad: facilita el acceso a los datos de forma inmediata desde cualquier sitio.
- Mantenimiento: la copia de seguridad, la replicación de los datos y la adquisición de un dispositivo adicional para el almacenamiento es responsabilidad del proveedor del servicio, de esta manera las empresas se enfocan directamente en la actividad de su negocio.

Desventajas

El uso de los Sistemas de Almacenamiento en la Nube también puede traer consigo desventajas como (14):

- La seguridad de los datos es un problema común, debido a la importancia y sensibilidad de datos que se almacenan en un proveedor.
- El desempeño y rendimiento pueden verse afectados en comparación a un almacenamiento local.
- La disponibilidad y confianza del acceso a los datos depende estrictamente de las reglas y de los niveles de precaución por parte del proveedor del servicio.
- La desestabilidad financiera del proveedor puede minar la calidad del servicio.

1.3.1 Sistemas de Almacenamiento en la Nube

Existen numerosos proyectos dedicados al Almacenamiento en la Nube. A continuación, se exponen las características de algunos de los sistemas más importantes dedicados a este tipo de servicio: Dropbox, Google Drive y One Drive.

- **Dropbox**

Dropbox es un servicio de almacenamiento de archivos en la nube fundado en 2007 que ofrece la posibilidad de sincronizar los archivos entre varios dispositivos haciendo parecer que es la misma carpeta independientemente del dispositivo que se esté usando.

Ofrece tres planes orientados a distintos usuarios finales partiendo de los 2 GB con la cuenta gratuita, 32 GB en cuentas Pro y hasta 1 TB para las cuentas de empresa. Inicialmente se dispone de 2 GB gratuitos con la posibilidad de poder aumentarlo realizando una serie de promociones ofrecidas por Dropbox. Por cada persona que se registre en Dropbox usando un enlace proporcionado por un usuario, se le añadirán a este último 500 MB hasta un máximo de 16 GB en la cuenta gratuita.

Se puede almacenar cualquier tipo de archivo pudiendo ser compartido a través de un enlace, teniendo un máximo de 20 GB de transferencia al día para las cuentas gratuitas y 200 GB para las cuentas Pro y de empresa. Los archivos subidos a través de la aplicación de escritorio o aplicaciones móviles no tienen límite de tamaño de archivo, los cargados en dropbox.com deberán ser de 20 GB o más pequeños y ser menores que el límite de almacenamiento. Por ejemplo, si la cuenta tiene una cuota de almacenamiento de 2 GB, puede cargar un archivo de 2 GB o más archivos que sumen 2 GB en total, sino Dropbox dejará de sincronizarse.

Los clientes oficiales están disponibles para Android, Windows Phone, Blackberry e iOS facilitando el acceso entre dispositivos y plataformas. En aplicaciones de escritorio, se tendrán todos los archivos almacenados en Dropbox disponibles sin acceso a internet (de forma física en el ordenador). En las aplicaciones móvil los archivos solo están disponibles con conexión a internet o mediante descarga previa. La seguridad en Dropbox es una prioridad según asegura la empresa. Para proteger los archivos en tránsito, se usa el protocolo SSL[6]/TLS[7] para la transferencia de archivos; de este modo, se crea un túnel seguro protegido por cifrado AES[8] de 128 bits o más. Los datos de los archivos de Dropbox se guardan en bloques de archivos discretos que se fragmentan y cifran mediante AES de 256 bits. Además, se respalda la confidencialidad de la comunicación, se marcan todas las cookies de autenticación como

[6] **SSL**: *Secure Sockets Layer* o Capa de Conexión Segura (traducido al español) son protocolos criptográficos que proporcionan comunicaciones seguras por una red, comúnmente internet.

[7] **TLS:** *Transport Layer Security* o Seguridad de la Capa de Transporte (traducido al español) es el sucesor de *Secure Sockets Layer* (SSL). Son protocolos criptográficos que proporcionan comunicaciones seguras por una red, comúnmente Internet.

[8] **AES:** *Advanced Encryption Standard* o Estándar de Cifrado (encriptación) Avanzado (traducido al español), es uno de los algoritmos más seguros y más utilizados hoy en día, disponible para uso público. (79)

seguras y se habilita la política HSTS[9]. De forma predeterminada, Dropbox conserva un historial de todos los archivos eliminados y de las versiones anteriores de estos, y permite restaurarlos durante un máximo de treinta días. Para proteger las cuentas de cada usuario se aconseja elegir una contraseña sólida y exclusiva y habilitar la verificación de dos pasos que agrega un nivel de protección adicional, en la cual se solicita la contraseña y un código de seguridad de seis dígitos enviado por mensaje de texto o por una aplicación de autenticación independiente. (21)

Dropbox se anuncia diciendo que ni si quiera los empleados tienen acceso a los datos guardados, aunque en el 2011 se pudo comprobar que un hacker obtuvo acceso a un grupo de cuentas, lo que puso en duda la seguridad. Aun así, hoy en día, Dropbox es considerado el servicio más popular para guardar datos en la nube (8). Según el propio sitio oficial en el 2016 se llegó a 500 millones de personas de todo el mundo que usan Dropbox para trabajar como lo desean, estén donde estén y en cualquier dispositivo. Además, reúne a más de 150.000 organizaciones en Dropbox Business. Representa un 24% del mercado frente a la competencia, que lo hacen ser todavía un punto de referencia para los demás sistemas (22).

Las características más destacadas que ofrece Dropbox son (23):

- Ofrece control de versiones.
- Cifrado de archivos.
- Descarga a móvil.
- Multimedia.
- Ficheros y directorios colaborativos.
- API Pública.
- Compartir archivos públicamente.
- SDK en PHP[10].

[9] **HSTS:** HTTP *Strict Transport Security* o Seguridad de transporte HTTP estricta (traducido al español) es un mecanismo de política de seguridad web según la cual un servidor web declara a los agentes de usuario compatibles (por ejemplo, un navegador web) que deben interactuar con ellos solamente mediante conexiones HTTP seguras (es decir, en capas HTTP sobre TLS/SSL). (80)

[10] **PHP:** es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.

API de Dropbox

El punto más fuerte de Dropbox es su API y sus distintos SDK en varios lenguajes. Con la explosión de los dispositivos móviles, disponer de una aplicación es prácticamente obligatorio para todas las empresas, y gracias a la API, integrar Almacenamiento en la Nube es una solución con un coste prácticamente nulo para la empresa (17).

La comunicación con la API se realiza siguiendo el protocolo de llamadas o peticiones HTTP usando los *endpoints* especificados en la documentación dependiendo de la acción que se necesite realizar. Cada acción requiere de unas cabeceras y cuerpos distintos, usando el formato JSON como lenguaje común.

Dropbox ofrece dos tipos distintos de acceso, Drop-ins y Core API, el primero es una forma rápida de integrar Almacenamiento en la Nube con Dropbox, pero con acceso limitado, mientras que el segundo ofrece un acceso completo siendo mucho más flexible. Además, ofrece varios SDK para distintos lenguajes.

- **Google Drive**

Google Drive es el servicio de Almacenamiento en la Nube que ofrece el gigante Google lanzado el 24 de abril de 2012 y que fue el sucesor de Google Docs. El espacio que ofrece inicialmente es de 15 GB compartidos entre todos los servicios como el de Gmail y Google+ Photos. Además, no solo ofrece espacio para almacenar archivos, sino que, también dispone de un procesador de texto, hojas de cálculo, presentaciones y previsualización de archivos.

Ofrece un plan de precios para poder disponer de la capacidad de almacenamiento que parte de 15 GB (gratis) hasta 30 TB ($299.99 al mes). También ofrece tres tipos de aplicaciones para acceder al contenido: web, escritorio y móvil, aunque con diferentes características a la hora de gestionar los archivos.

En la aplicación web es posible crear, ver, editar, copiar, mover, borrar y subir cualquier tipo de archivo mientras tengamos conexión. La aplicación de escritorio crea una carpeta en el sistema, en la cual todo el contenido estará sincronizado con Google Drive, siendo capaz de disponer de cualquier archivo que se almacene en dicha carpeta en la nube. Sin embargo, la aplicación para dispositivos móviles funciona de forma distinta ya que, en vez de descargar y mantener en todo momento sincronizados los archivos entre el dispositivo-nube, esta muestra simplemente el contenido sin llegar a realizar en ningún momento la descarga de los archivos.

Google utiliza HSTS en todos sus servicios e implementa medidas internas para el inicio de sesión. Además, ofrece la verificación de dos pasos al igual que otros sistemas como Dropbox y One Drive. En

cuanto a los datos en sí, son encriptados en tránsito (hacia y desde un dispositivo, y también entre los centros de datos de Google) mediante SSL, pero sólo almacena los archivos en reposo usando el algoritmo AES de 128 bits. Brinda varios SDK en distintos lenguajes para hacer más fácil el desarrollo de las aplicaciones vinculadas a sus servicios. (24)

Las características más destacadas que ofrece Google Drive son (25):

- Control de versiones.
- Cifrado de archivos.
- Descarga a móvil
- Ficheros y directorios colaborativos.
- API Pública.
- SDK en PHP.

API de Google Drive

La API de Google Drive se encuentra actualmente en su versión 2.0. La documentación está dividida en trece bloques principales: *files*, *about*, *changes*, *children*, *parents*, *permissions*, *revisions*, *apps*, *comments*, *replies*, *properties, channels* y *realtime*. Dentro de cada uno de ellos están las distintas acciones que se pueden realizar respecto a su bloque.

La comunicación con la API se realiza siguiendo el protocolo de llamadas o peticiones HTTP usando los *endpoints* especificados en la documentación dependiendo de la acción que se necesite realizar. Cada acción requiere de unas cabeceras y cuerpos distintos, usando el formato JSON como lenguaje común. Para poder realizar peticiones se necesita de una *API key*, *client id* y *client secret* que se puede obtener registrando una aplicación como desarrollador en Google.

- **One Drive**

Antes conocido como SkyDrive o Microsoft SkyDrive, One Drive es la propuesta de Microsoft para el alojamiento de archivos en la nube. Ofrece un almacenamiento y compartición de archivos en línea en un servicio gratuito. Permite editar tus documentos en Microsoft Office en línea, ofrece una versión *Lite*[11] en tres de los softwares más importantes del grupo Word, PowerPoint y Excel. Los documentos creados en

[11] **Lite:** versión de más bajo precio, con menos funcionalidades o con opciones deshabilitadas pero que puede ser útil para determinado tipo de usuario.

One Drive tienen el mismo tipo de extensión de archivo que los de Office, lo que constituye una excelente opción, que asegura la compatibilidad para trabajar con la herramienta de oficina en la nube o en la computadora.

El espacio inicial que provee es de 7 GB, que permite en un corto periodo de tiempo aumentarlo a 25 GB de forma gratuita. Los precios de espacio de almacenamiento parten de 20GB por $10 hasta 100GB por 50$ al año en planes de cuentas por pagos.

Se pueden compartir archivos con las personas que se desee y publicarlos en Facebook, LinkedIn o Twitter. El servicio utiliza las cuentas de Windows Live ID para controlar el acceso a los archivos del usuario, y les permite mantener la confidencialidad de estos, lo contrario cuando se comparten públicamente pues para acceder no se requiere de esta cuenta. Permite crear diferentes directorios compartidos. (26)

Los datos se cifran en tránsito utilizando SSL, pero permanecen sin cifrar en reposo, a menos que sea una cuenta de tipo One Drive Business, en la cual, a partir de finales del 2015, Microsoft introdujo el cifrado para cada fichero, que encripta archivos de forma individual cada uno con una clave única; por lo que si se ha visto comprometida una clave solo se puede acceder a un archivo individual en lugar de todos. Los usuarios consiguen el acceso a través de la verificación de dos pasos, que protege aún más el inicio de sesión a través de una aplicación o un mensaje de texto. (24)

Quienes tienen una cuenta de Hotmail, Windows Live o Outlook no necesitan crear una cuenta para utilizar el servicio de One Drive. En caso contrario es necesario crear una antes.

Las características más destacadas que ofrece One Drive son (26):

- Control de versiones.
- Cifrado de archivos.
- Sincronización de múltiples directorios.
- Descarga a móvil.
- *Streaming* multimedia.
- Ficheros y directorios colaborativos.
- API Pública.

- Compartir archivos públicamente.

API de One Drive

Microsoft ha publicado un conjunto de API para One Drive mediante Live Connect que permiten a los programadores desarrollar servicios web y aplicaciones que utilizan el almacenamiento en One Drive. Esto permite a los usuarios que hacen uso de los servicios web y el cliente de aplicaciones examinar, ver, subir, descargar o editar archivos guardados en One Drive. Actualmente está disponible un SDK para los desarrolladores web y los desarrolladores de aplicaciones de estilo Windows 8 o más avanzado, Windows Phone, iOS, o Android.

Se puede afirmar que, en temas de seguridad, el almacenamiento de archivos en la nube no es completamente seguro, pero la selección de un proveedor con determinada reputación en temas de seguridad y el seguimiento de buenas prácticas por parte del usuario como la de utilizar una contraseña segura y la verificación en dos pasos para la autenticación pueden significar un punto determinante en estos temas. También para una mayor seguridad de los archivos se aconseja cifrar estos antes de almacenarlos en los Sistemas de Almacenamiento en la Nube mediante una herramienta como puede ser BoxCryptor, por ejemplo, que comúnmente se encarga de cifrar los archivos antes de que sean subidos a la nube. Esta herramienta utiliza los algoritmos de encriptación AES 256 y RSA[12], que son aprobados para mantener la protección de la más importante información *"Top Secret"* (27).

Se debe tener en cuenta que para realizar el estudio de los Sistemas de Almacenamiento en la Nube se obviaron los privados y los que brindaban servicio necesariamente a partir de pagos por no constituir una opción viable para la propuesta de solución. Además, se considera preciso que los sistemas brinden un SDK estable desarrollado al menos en PHP para poder realizar la implementación en el plazo de tiempo esperado. Algunos inconvenientes presentados para la implementación impiden brindar una solución a través de los servicios de Google Drive y One Drive. El primero no permite crear una cuenta para la aplicación como desarrollador desde Cuba, por lo que es imposible obtener el *API key*, *client id* y *client secret* necesarios para consumir los servicios de este sistema. Del segundo ni siquiera se tiene acceso desde Cuba, y en caso de que se pudiera acceder a este es necesario disponer de una cuenta en Windows Live, Outlook o Hotmail, por lo que resultó como escogido el Sistema de Almacenamiento en la Nube

[12] **RSA:** Es un sistema criptográfico de clave pública. Es el primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar digitalmente. (82)

Dropbox como el único realmente viable para utilizar en la propuesta de solución. Del anterior estudio resultó la tabla resumen que a continuación se muestra.

***Tabla 1**.Comparación entre los Sistemas de Almacenamiento en la Nube.*

Características	Google Drive	Dropbox	One Drive
Tipo de Almacenamiento	Público	público	público
API pública	Si	Si	Si
Capacidad de almacenamiento gratis	15GB	2GB(ampliable hasta 16GB con promociones)	7GB(ampliable hasta 25GB con promociones)
Seguridad de archivos en tránsito	Si	Si	Si
Cifrado de archivos almacenados	Cifrado	Cifrado	Cifrado solo para cuentas “Business”
Compartir archivos públicamente	Si	Si	Si
Acceso desde Cuba	Si	Si	No
Permite registro de aplicaciones desde Cuba	No	Si	No

1.4 Integración de plataformas educativas con Sistemas de Almacenamiento en la Nube

La rápida expansión de internet ocurrida en todos los niveles de la sociedad también se ha reflejado en el ámbito educativo puesto que la explotación didáctica de la Web permite ampliar la oferta educativa, la calidad de la enseñanza y el acceso a la educación. Hace un tiempo la novedad dentro de las TIC era lo electrónico (*e-learning*), ahora se habla de ubicuidad. La tecnología permite estar presente en diferentes lugares al mismo tiempo, tener la información disponible a cualquier hora y en cualquier lugar, porque los dispositivos tecnológicos modifican la manera de acceder a la información y el conocimiento. (28)

Como se refleja en el informe Horizon 2015, documento que pretende recoger las principales tendencias actuales y futuras en el ámbito educativo, apunta a que la distribución de contenidos a través de medios digitales toma más fuerza que nunca. La rápida adopción de dispositivos como *tablets* y *smartphones* unido al uso educativo de los ordenadores está permitiendo que los formatos digitales sean un medio rápido, cómodo y eficaz para distribuir contenidos, impulsado por el paradigma de aprendizaje activo emergente en estos días, que requiere mayor movilidad, flexibilidad y permitir el uso de múltiples dispositivos (29). El valor de los recursos educativos abiertos se va extendiendo progresivamente, no solo en lo gratuito, sino también copiable y reutilizable sin límites para usos educativos (30).

La computación en la nube en los entornos educativos otorga flexibilidad tanto a estudiantes como a profesores para crear, consultar o descargar materiales educativos en el momento pertinente apoyados en un ordenador con acceso a internet. (31) Es por eso que algunas plataformas educativas buscan apoyo en la computación en la nube y dentro de esta en los Sistemas de Almacenamiento en la Nube.

Una de las formas de integrar la computación en la nube en las plataformas educativas es utilizando las herramientas de creación de documentos que pueden ser de tipo office, y a su vez, permiten que estos se almacenen dentro de la propia nube.

Un ejemplo de este caso es el de **VirtusClass** que es una plataforma que integra las soluciones de Google (Google Apps) o Microsoft (Office365) para la creación de documentos de office, que también permite de forma gratuita almacenar estos últimos en la nube donde se crearon.

Otro caso de integración de plataformas educativas con Sistemas de Almacenamiento en la Nube están las que utilizan estos últimos para almacenar archivos. Este caso particular es el más semejante a la solución que se quiere desarrollar. Algunas de estas plataformas son:

Paradiso: plataforma virtual LMS colombiana para capacitadores y empresas formadoras, educativas y de consultoría, que trabaja en conjunto con las aplicaciones de Almacenamiento en la Nube como Dropbox y One Drive, que da la posibilidad de acceder, compartir y colaborar en vivo en todos los archivos que se han almacenado allí, importando fácilmente los documentos y archivos para que posteriormente se puedan compartir y realizar un trabajo conjunto.

School Alive: plataforma que permite interactuar a toda la comunidad educativa de una forma integradora con el objetivo de favorecer el proceso de aprendizaje de los alumnos. Entre sus componentes y herramientas principales cuenta con Aula: Aprender, basada en una adaptación de Moodle 2.3 que está totalmente integrada dentro de la plataforma. Dispone de funcionalidades avanzadas como el arrastrar y

soltar para cargar nuevos contenidos, carga de contenidos desde la carpeta de Google Drive y con un portafolio virtual integrado con el mismo Google Drive.

La plataforma de la Facultad de Informática de la Universidad Nacional de La Plata, Argentina, también es uno de los casos de integración con Sistemas de Almacenamiento en la Nube. El LMS Moodle viene siendo utilizado en la Facultad para cursos que complementan las clases presenciales. Se genera una gran cantidad de material académico, por lo que se comenzó con el proyecto de creación del repositorio para albergar y poner a disposición en forma pública el contenido, como recursos educativos abiertos[13], en Sistemas de Almacenamiento en la Nube como Dropbox y Google Drive. Los accesos a estos últimos aparecen dentro de la vista de un ítem de una colección. Se representan por medio de enlaces con los íconos identificatorios de ambas plataformas y se asocian a cada archivo del ítem. Su función es la de transferir el archivo correspondiente a una cuenta de Dropbox o de Google Drive en particular. Al activarse este enlace, se pide el inicio de sesión de una cuenta mediante una ventana emergente y luego comienza la transferencia. Si la sesión ya estaba iniciada, el archivo se transfiere automáticamente.

En Cuba no se tiene conocimiento de que existan plataformas educativas integradas con Sistemas de Almacenamiento en la Nube.

1.5 Metodología de desarrollo de software

La metodología de desarrollo de software constituye un enfoque estructurado para el desarrollo de software que incluye modelos de sistemas, notaciones, reglas, sugerencias de diseño y guías de procesos (32).

Particularmente, una metodología se basa en una combinación de los modelos de procesos genéricos para obtener como beneficio un software que solucione un problema. Adicionalmente, una metodología debería definir con precisión los artefactos, roles y actividades, junto con prácticas, técnicas recomendadas y guías de adaptación de la metodología al proyecto. La complejidad del proceso de creación de software es netamente dependiente de la naturaleza del proyecto mismo. (33)

Existen dos tipos de metodologías de desarrollo de software, las robustas o tradicionales y las ágiles. Las metodologías ágiles permiten enfocarse más en el desarrollo y verificación del software en lugar del diseño

[13] **Recursos educativos abiertos:** son documentos o material multimedia con fines relacionados con la educación como la enseñanza, el aprendizaje, entre otros, cuya principal característica es que son de acceso libre y por lo general bajo licencia abierta (74).

y la documentación. Estas metodologías se apoyan en el enfoque incremental para la especificación, el desarrollo y la entrega del software. (34)

Para conocer la metodología de desarrollo de software utilizada para el desarrollo de la propuesta de solución se propone consultar la siguiente sección, que aborda las características más generales de esta, la que fue definida por la arquitectura del proyecto ZERA 2.0.

Agile Unified Process (AUP) variante UCI

AUP es una versión simplificada del Proceso Unificado de *Rational* (RUP). Descrito en una forma simple, fácil de entender y brinda un enfoque de desarrollo de software utilizando técnicas ágiles y conceptos del RUP. El proceso unificado (*Unified Process* o UP) es un marco de desarrollo de software iterativo e incremental. (35)

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. (35)

AUP-UCI divide el ciclo de desarrollo en 3 fases (35):

- **Inicio:** Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
- **Ejecución:** En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
- **Cierre:** En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

AUP-UCI propone las siguientes disciplinas (35):

- **Modelado de negocio**: Es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito.

- **Requisitos:** El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto.
- **Análisis y diseño:** En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis.
- **Implementación:** En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema.
- **Pruebas internas:** En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas.
- **Pruebas de Aceptación:** Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

AUP-UCI define actividades que el desarrollador debe realizar para construir, validar y entregar un software que satisfaga las necesidades de las partes interesadas. A partir de esta se obtendrán los artefactos necesarios para conformar la propuesta de solución.

1.6 Tecnologías y herramientas

En este epígrafe se hace un análisis de las principales tecnologías y herramientas actuales a considerar, con el objetivo de conocer sus características y así dar cumplimiento con la mayor eficiencia y calidad posible a la solución de la presente investigación, respetando el modo de trabajo definido en la arquitectura de la plataforma educativa ZERA 2.0.

1.6.1 API

Una interfaz de programación de aplicaciones (API) es un conjunto de instrucciones de programación y de normas de acceso a una aplicación de software basada en web o herramienta web. Una compañía de software libera su API para el público para que otros desarrolladores de software puedan diseñar productos que son alimentados por su servicio. (36)

Las API son especialmente importantes porque determinan cómo los desarrolladores pueden crear nuevas aplicaciones que aprovechan grandes servicios web (37) como los servicios de Almacenamiento en la Nube de Google Drive, One Drive o Dropbox. Cada uno de estos ofrece su API de manera distinta y con su propio formato de entrada/salida, sin embargo, algo en lo que coinciden todas estas API es en el uso del protocolo HTTP (*Hypertext Transfer Protocol*) para la comunicación con las aplicaciones.

Algunas de estas API son liberadas públicamente en distintas versiones para el uso de sus servicios en distintos tipos de aplicaciones. El desarrollador de una aplicación web, por ejemplo, puede utilizar la API de Dropbox para permitir a los usuarios almacenar sus juegos guardados y el avance de estos periódicamente en Dropbox en lugar de llevar a cabo alguna otra opción tradicional de Almacenamiento en la Nube desde cero. Las API son grandes ahorradoras de tiempo que con su amplia disponibilidad para servicios importantes hizo posible la experiencia web moderna (37).

1.6.2 SDK

El kit de desarrollo de software o más conocido como SDK es por norma general un conjunto de herramientas de desarrollo de software que permite a un programador crear programas y aplicaciones para un sistema o plataforma concretos. Suelen incluir herramientas, bibliotecas, documentación, código de ejemplo, algún tipo de explicación de lo que contiene y otras utilidades que ayudarían a un programador a desarrollar una aplicación. (38)

La mayoría de los kits de desarrollo de software son gratuitos y se distribuyen libremente por internet para fomentar la colaboración de los desarrolladores, mejorar sus productos y para que utilicen su lenguaje. También se utiliza como herramienta de marketing, ya que, por ejemplo, si se alienta a la gente a programar sus propias aplicaciones, estos comprarán los productos porque también seguirán creando para ellos. (38)

Se puede citar como ejemplo de SDK el de Dropbox, conocido Sistema de Almacenamiento en la Nube, que incluye bibliotecas específicas en varios lenguajes construidos en la parte superior de su propia API, que ayuda al desarrollador a crear aplicaciones para consumir servicios de esta. Son esencialmente una envoltura alrededor de la API para permitir un desarrollo más rápido en comparación con los que trabajan directamente con la API (39).

1.6.3 *Componente*

El *Object Management Group* (OMG), institución dedicada al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos, define componente en su especificación UML, como una

unidad modular con interfaces bien definidas, que es reemplazable dentro de su entorno. Así, un componente define su comportamiento en términos de interfaces proveídas y requeridas; y dicho componente será totalmente reemplazable por otro que cumpla con las interfaces declaradas. Un componente siempre se puede considerar una unidad autónoma dentro de un sistema o subsistema. (40)

Constituye una unidad de composición con interfaces especificadas contractualmente, con dependencias explícitas de acuerdo al contexto antes dicho. Un componente de software puede ser desplegado de forma independiente y puede participar en composiciones de terceras partes (41). Claramente extiende una o varias funcionalidades dentro de un contexto en la arquitectura de un software.

1.6.4 Frameworks

El término "*framework*", cuya traducción aproximada sería "marco de trabajo", es un concepto que los que se dedican al desarrollo de software pueden llegar a utilizar o conocer. Sin embargo, no es un concepto tan simple y sencillo de definir.

Puede ser considerado como un diseño abstracto orientado a objetos para un determinado tipo de aplicación, que se compone de una clase abstracta para cada componente principal del diseño; contendrá normalmente una librería de subclases que pueden ser utilizadas como componentes del diseño. (42)

Es un conjunto de funciones o código genérico para realizar tareas comunes y frecuentes en todo tipo de aplicaciones (creación de objetos, conexión a base de datos, limpieza de *string*, etc.). Esto da la oportunidad para mantener una sólida base sobre la cual desarrollar aplicaciones específicas, permitiendo obviar los componentes más triviales y genéricos del desarrollo. (43)

Simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas. (44)

Los objetivos principales que persigue un *framework* son (43):

- Acelerar el proceso de desarrollo.
- Reutilizar código ya existente.
- Promover buenas prácticas de desarrollo como el uso de patrones.

1.6.5 Symfony

Symfony es un completo *framework* diseñado para optimizar el desarrollo de las aplicaciones web. Emplea el tradicional patrón de diseño MVC[14] (modelo-vista-controlador) para separar las distintas partes que forman una aplicación web, la lógica de negocio, la lógica de servidor y la presentación. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar algo ya hecho cada vez que se crea una nueva aplicación web. (44)

Symfony está desarrollado completamente con PHP 5. Es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows. Los desarrolladores de Symfony alrededor del mundo están conectados en una comunidad que muestra muchas ganas de colaboración entre sus integrantes. Dentro de la misma se encuentran también los creadores del *framework* lo que nutre en gran medida las discusiones que se realizan en dicha comunidad. (44)

Se utilizará la versión 2.7.13 de Symfony para la implementación del componente. El uso de este *framework* para la propuesta de solución estará potenciado por la utilización de los SDK y su respectiva documentación proporcionados por algunos de los Sistemas de Almacenamiento en la Nube. Estos SDK son implementados en lenguajes como PHP 5, que implementa a Symfony, por lo que apoyará en gran medida la implementación del componente.

1.6.6 Doctrine

Doctrine es un mapeador de objetos-relacional (ORM) escrito en PHP que proporciona una capa de persistencia para objetos del mismo lenguaje. Es una capa de abstracción que se sitúa justo encima de un sistema de gestión de bases de datos (45).

Una característica de Doctrine es el bajo nivel de configuración que se necesita para empezar un proyecto. Este ORM puede generar clases a partir de una base de datos existente, en la cual el programador puede especificar relaciones y añadir funcionalidad extra a las clases autogeneradas. No es necesario generar o mantener complejos esquemas de base de datos como en otros frameworks. Posibilita escribir consultas de base de datos utilizando un dialecto de SQL denominado DQL (Doctrine Query Language). Tiene

[14] **MVC**: es un patrón de diseño de software dirigido a su arquitectura, en el cual todo el proceso está dividido en 3 capas. Típicamente estas capas son el Modelo, la Vista y el Controlador.

soporte para datos jerárquicos y herencia. Se utilizará la segunda versión de esta herramienta para el desarrollo de la propuesta de solución.

1.6.7 Lenguaje de modelado UML

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. (46)

La especificación de UML no define un proceso estándar, pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos. UML también contiene construcciones organizativas para agrupar los modelos en paquetes, lo que permite a los equipos de software dividir grandes sistemas en piezas de trabajo, para entender y controlar las dependencias entre paquetes, y para gestionar las versiones de las unidades del modelo, en un entorno de desarrollo complejo. (46)

1.6.8 Herramienta CASE: Visual Paradigm

Visual Paradigm es una poderosa herramienta CASE[15] que utiliza UML (Lenguaje Unificado de Modelado) como lenguaje de modelado. Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite crear tipos diferentes de diagramas en un ambiente totalmente visual. Es muy sencillo de usar, fácil de instalar y actualizar. Genera código para varios lenguajes. (47)

Posibilita la representación gráfica de los diagramas permitiendo ver el sistema desde diferentes perspectivas como: componentes, despliegue, secuencia, casos de uso, clase, actividad, estado, entre otros. Además, identifica requisitos y comunica información, se centra en cómo los componentes del sistema interactúan entre ellos, sin entrar en detalles excesivos. También permite ver las relaciones entre los componentes del diseño y mejorar la comunicación entre los miembros del equipo usando un lenguaje gráfico. Soporta la realización de ingeniería tanto directa como inversa (47). Para el desarrollo de la propuesta de solución se utilizará la versión 8.0 de esta herramienta.

[15] **CASE:** (*Computer Aided Software Engineering* o traducido al español como Ingeniería de Software Asistida por Computadora) son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software. Ayudan en todos los aspectos y tareas del ciclo de vida de desarrollo del software.

1.6.9 Herramienta IDE: NetBeans

NetBeans es una herramienta IDE[16] de código abierto para desarrolladores. Posee todas las herramientas necesarias para crear aplicaciones web, de escritorio y para teléfonos móviles con los lenguajes de programación Java, C, C++ e incluso lenguajes dinámicos como PHP, JavaScript, Groovy, y Ruby. Es fácil de usar, instalar y puede ser ejecutado en múltiples plataformas como Windows, Linux, Mac OS X y Solaris. (48)

Permite una visión clara de aplicaciones de gran tamaño. Proporciona diferentes vistas de los datos, de múltiples ventanas de proyectos a herramientas útiles para la creación y gestión de aplicaciones de manera eficiente (48). Para el desarrollo de la propuesta de solución se utilizará la versión 8.1 de esta herramienta.

1.5 Conclusiones parciales

En tiempos en los que la tecnología da soporte a toda actividad humana, el Almacenamiento en la Nube se alza como una interesante opción para apoyar a la educación y a las plataformas educativas en la búsqueda de nuevas vías más eficientes de acceso a los recursos educativos. Una vez concluido el estudio de soluciones similares y de las diferentes tecnologías para dar solución al problema planteado, se puede aseverar que la integración de ZERA 2.0 con Sistemas de Almacenamiento en la Nube constituye una opción viable para el almacenamiento de archivos alojados externamente, pues puede dotar a la plataforma de mayor accesibilidad, disponibilidad, reutilización y difusión de los recursos educativos que se generan. La metodología de desarrollo de software a utilizar sentará las bases y guías que darán pie al desarrollo de la solución propuesta, y a partir de esta, se obtendrán los artefactos en cada etapa del desarrollo del componente. El estudio realizado de las tecnologías y herramientas, permitió conocer las características y funcionalidades de cada una, para llevar a cabo el desarrollo del componente con la mayor eficiencia y calidad posible. La utilización del *framework* Symfony permitirá aprovechar un conjunto de funcionalidades que posibilitarán aumentar al máximo la reutilización de código y disminución del tiempo de desarrollo. La implementación del componente, o particularmente la implementación de un bundle es la forma natural que propone Symfony para extender el proyecto y compartir código entre distintas aplicaciones de este. Esta capacidad será utilizada en la propuesta de solución.

[16] **IDE:** (*Integrated Development Environment* o traducido al español como herramienta de desarrollo integrado) son herramientas para el programador, que suelen incluir en una misma suite un editor de texto, un compilador, un intérprete, un depurador, que tenga posibilidad de ofrecer un sistema de control de versiones y que ayude en la construcción de interfaces gráficas de usuario. (68)

Capítulo 2. Características y Diseño de la solución propuesta

La metodología AUP define en cada una de sus fases un grupo de artefactos que brindan una comprensión clara del componente a desarrollar. Al tener la misma un enfoque ágil permite que se elabore la documentación necesaria que facilite la comunicación entre el desarrollador y las partes interesadas. En este capítulo se presentarán los siguientes artefactos: el modelo del dominio, la especificación de requisitos funcionales y no funcionales que tendrá la propuesta de solución, las historias de usuario y los diagramas de clases del diseño y de secuencias del diseño. Además, se describirá el patrón arquitectónico y los patrones de diseño empleados en la propuesta de solución.

2.1 Modelo de dominio

Un modelo de dominio o modelo conceptual es una representación de conceptos del dominio de un problema determinado. Mediante el uso de UML este se puede modelar utilizando un grupo de diagramas de estructura estática que no representa ninguna operación. Permite representar elementos del mundo real y no componentes del software. Provee conocimiento de la nomenclatura del dominio al comunicarle a las partes interesadas los términos importantes y como se relacionan entre sí, siendo un artefacto fundamental para realizar el análisis orientado a objetos. (49)

2.1.1 Conceptos del dominio

Para un mayor entendimiento del problema se procede a esclarecer las terminologías asociadas al dominio, definiendo los términos más importantes que serán de utilidad para la posterior confección del modelo.

- **Usuario:** Existen siete tipos de usuarios definidos por ZERA 2.0 que podrán hacer uso de los servicios del componente, estos son: administrador, profesor principal, profesor editor, facilitador, observador, observador global y estudiante. Estos pueden consumir los servicios del componente, ya que cualquiera puede hacer uso de los archivos que tiene almacenados en los Sistemas de Almacenamiento en la Nube desde la plataforma para el trabajo en la sección de los cursos.

- **Sistema de Almacenamiento en la Nube:** Es el sistema escogido para utilizar el servicio de almacenamiento, en este caso constituido por Dropbox.

- **Archivo:** Constituye toda aquella información en formato digital que en este caso particular esta almacenada en algún Sistema de Almacenamiento en la Nube.

- **Recurso educativo**: Es el tipo de archivo que realmente se desea manejar en el problema, el cual persigue una finalidad didáctica o que facilite el desarrollo de las actividades formativas.

2.1.2 Diagrama del modelo de dominio

El siguiente diagrama muestra la relación entre los conceptos identificados anteriormente descritos en el problema.

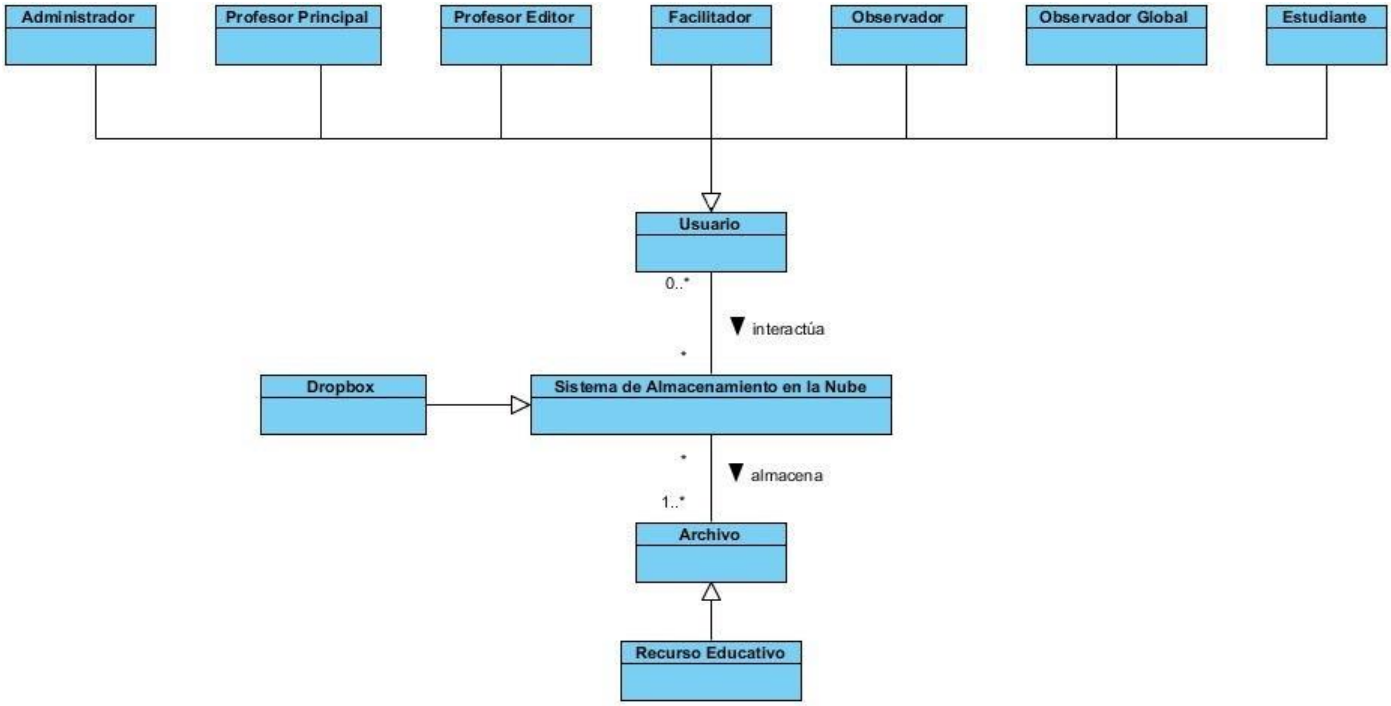


Figura 1. *Modelo del dominio.*

2.2 Requerimientos funcionales del componente

La definición de requisitos del sistema permite establecer las capacidades del mismo y los detalles de las funciones, servicios y restricciones operativas que el sistema debe cumplir. En algunos casos los requerimientos funcionales de los sistemas pueden declarar explícitamente lo que este no debe hacer (50). Los mismos son planteados para lograr un mejor entendimiento entre el cliente y el desarrollador, con el fin de satisfacer sus verdaderas necesidades.

A continuación, se presentan los requisitos funcionales del sistema.

- **RF1** Añadir archivo almacenado en los Sistemas de Almacenamiento en la Nube como recurso educativo en los cursos de ZERA 2.0.

- **RF2** Autenticar usuario en los Sistemas de Almacenamiento en la Nube.
- **RF3** Listar archivos almacenados en los Sistemas de Almacenamiento en la Nube.
- **RF4** Descargar archivos de los Sistemas de Almacenamiento en la Nube.
- **RF5** Subir archivos hacia los Sistemas de Almacenamiento en la Nube.
- **RF6** Obtener enlace de un archivo compartido en los Sistemas de Almacenamiento en la Nube.
- **RF7** Añadir enlace de un archivo compartido en los Sistemas de Almacenamiento en la Nube en los cursos de ZERA 2.0.

2.3 Requerimientos no funcionales del componente

Los siguientes requisitos no funcionales definen cualidades que complementan el funcionamiento del sistema.

Hardware

- La PC donde se quiera tener acceso a los archivos almacenados en cuentas de los Sistemas de Almacenamiento en la Nube debe estar conectada a internet.

Usabilidad

- El componente debe poseer una interfaz fácil de utilizar para cualquier tipo de usuario con conocimientos básicos de informática y en el manejo de ordenadores.

Seguridad

- El acceso a la información almacenada en los Sistemas de Almacenamiento en la Nube estará restringido por usuario y contraseña, a no ser que el archivo haya sido compartido públicamente.
- La seguridad de los datos proporcionados por el usuario cuando este se autentique o registre en el sistema correrá a cargo de los Sistemas de Almacenamiento en la Nube, ya que ese proceso se desarrolla en estos.
- La conexión con los Sistemas de Almacenamiento en la Nube será a través del protocolo https.
- La seguridad de los archivos almacenados en las cuentas de los Sistemas de Almacenamiento en la Nube es de completa responsabilidad del proveedor del servicio.

Accesibilidad

- El acceso a los archivos almacenados en los Sistemas de Almacenamiento en la Nube debe estar disponible desde cualquier estación de trabajo conectada a internet si el proveedor se encuentra brindando el servicio.

Apariencia

- La interfaz debe ser sencilla, fácil y cómoda, respetando la interfaz ya definida del proyecto ZERA 2.0, y que permita a los usuarios hacer uso de los servicios del componente aún sin tener amplios conocimientos del funcionamiento de este.

- El sistema debe tener una interfaz con colores adecuados al contenido, predominando un estilo de diseño sencillo.

- Se debe informar al usuario sobre el estado de cualquier acción que ejecute, informándole mediante notificaciones el resultado de estas, tanto si es error, alerta o éxito.

2.4 Descripción de los actores del sistema

A continuación, se definen y describen los actores que interactúan con los servicios del componente.

Tabla 2. *Actores del Sistema y su descripción.*

Actor	Descripción
Usuario	Actor que podrá acceder a las funcionalidades que brinda el componente, tales como: autenticarse, listar archivos, subir archivos, descargar archivos, añadir archivo como recurso, obtener enlace de compartido, entre otras.

2.5 Historias de usuario

Entre los artefactos que define la metodología utilizada se encuentran las historias de usuario (HU) que se encargan de especificar las funcionalidades que brindará el componente. Cada HU es una representación de un requerimiento de software escrito en una o dos frases utilizando el lenguaje común del usuario. Representan una forma rápida de administrar los requerimientos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Las HU están compuestas por los siguientes campos:

- **Número:** Número de identificación para las HU, sería incremental en el tiempo.

- **Nombre del requisito:** Es el nombre del requisito para el que se va a realizar la HU, sirve para identificarlo fácilmente tanto para el desarrollador como para los clientes.
- **Programador:** Nombre del programador encargado de implementar la HU.
- **Iteración asignada:** Iteración a que corresponde, número de la iteración en la que se desarrollará el requisito.
- **Prioridad:** Qué tan importante es para el cliente, se clasifica en Alta, Media y Baja.
- **Tiempo Estimado:** Tiempo estimado en horas que se le asignará al requisito.
- **Tiempo Real:** Tiempo real dedicado a la realización de la HU en horas.
- **Descripción:** Es la descripción de la historia, detallando las operaciones del usuario y las respuestas del sistema.
- **Observaciones:** Informaciones de interés, como glosarios, detalles del usuario, etc.

A continuación, se muestra la historia de usuario del requisito Añadir archivo como recurso educativo en los cursos de ZERA 2.0, las restantes se encuentran en el ***Anexo 1***.

Tabla 3*. HU Añadir archivo almacenado en los Sistemas de Almacenamiento en la Nube como recurso educativo en los cursos de ZERA 2.0.*

Historia de Usuario	
Nombre del requisito: Añadir archivo almacenado en los Sistemas de Almacenamiento en la Nube como recurso educativo en los cursos de ZERA 2.0.	
Número: HU_1	**Programador:** Rosbel Caballero Ramírez
Iteración Asignada: 1	**Prioridad:** Alta
Tiempo Estimado: 72	**Tiempo Real:** 80
Descripción: El componente debe permitir añadir un archivo almacenado en el Sistema de Almacenamiento en la Nube como recurso educativo en los cursos de ZERA 2.0. La forma de añadir el recurso educativo al curso debe coincidir con las otras formas implementadas por ZERA 2.0.	
Observaciones: El usuario debe haberse autenticado en el Sistema de Almacenamiento en la Nube por medio de ZERA 2.0.	

2.6 Patrón arquitectónico

Los patrones arquitectónicos son patrones de diseño de software que ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. En comparación con los patrones de diseño, los patrones arquitectónicos tienen un nivel de abstracción mayor. (51)

Para el desarrollo del componente se utiliza el patrón arquitectónico Modelo-Vista-Controlador (MVC) en el cual está basado Symfony, aunque de la versión 2 del *framework* su creador dijera que Symfony 2 no es MVC, pues sólo proporciona herramientas para la parte del controlador y de la vista, la parte del modelo es responsabilidad del ORM Doctrine.

Resulta esencial conocer qué es una arquitectura MVC y cómo se aplican sus principios fundamentales a las aplicaciones en Symfony 2. El patrón MVC define tres niveles:

El modelo: representa la información relacionada con el dominio de la aplicación, es decir, su lógica del negocio. Abstrae la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes del tipo de gestor de base de datos utilizado. En la solución se utiliza Doctrine, una biblioteca cuyo objetivo es brindar poderosas herramientas para el acceso a una base de datos relacional. El ORM de Doctrine permite asociar objetos a una base de datos tal como MySQL, PostgreSQL o Microsoft SQL. El componente mediante las funcionalidades de Doctrine, puede recuperar objetos completos de la base de datos, tales como el DropboxFile.

La vista: transforma el modelo en interfaces de usuario permitiendo la interacción con el sistema. Solo se encarga de mostrar información. Twig es un motor y lenguaje de plantillas para PHP muy rápido y eficiente. Symfony 2 recomienda utilizar Twig para crear las plantillas de la aplicación. La sintaxis de Twig se ha diseñado para que las plantillas sean concisas y muy fáciles de leer y de escribir. En el componente se muestran varias plantillas, un ejemplo de estas es index.html.twig, que se utiliza para listar los archivos almacenados en la nube.

El controlador: se encarga de procesar las peticiones realizadas desde el navegador realizando los cambios necesarios en el modelo o en la vista. Este se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comando, etc.). En el componente los controladores son funciones PHP que toman información de la petición HTTP y construyen una

respuesta HTTP como un objeto de Symfony 2. Los controladores tienen la lógica que el componente necesita para reproducir el contenido de las páginas. La clase controladora DropboxController.php tiene varias acciones, por ejemplo, addNewResource que se encarga de descargar un archivo, adicionar la información en la entidad correspondiente y mostrar un mensaje de confirmación.

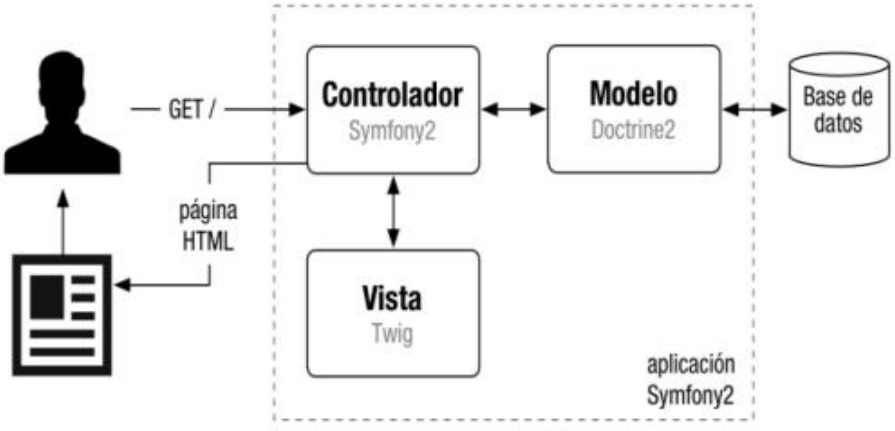


Figura 2 *Funcionamiento interno de Symfony 2*

2.7 Patrones de diseño empleados

Symfony sigue la mayoría de las mejores prácticas y patrones de diseño para la web (44). Para dar solución a la implementación de los requisitos funcionales del componente planteados en el anterior capítulo se aplican algunos patrones que guían el proceso de desarrollo. Estos 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 (52).

2.7.1 Patrones GRASP

Los patrones GRASP (Patrones Generales de Software para Asignar Responsabilidades, según siglas traducidas al español), describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones, y definen un grupo de principios fundamentales para diseñar eficazmente un software. Algunos de los utilizados en la solución que son implementados por Symfony se describen a continuación:

- **Experto:** Expresa que los objetos realizan funciones de acuerdo a la información con que cuentan. Esto trae como beneficio que se conserve el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Además, favorece el bajo acoplamiento. El comportamiento

se distribuye entre las clases que cuentan con la información requerida alentando con ello definiciones de clases sencillas que son fáciles de comprender y mantener. Se pone de manifiesto cuando la clase controladora DropboxController.php es llamada a responder determinado pedido y esta cuenta con la información necesaria para satisfacer la petición.

- **Creador:** Permite asignar quién debería ser el responsable de la creación de una nueva instancia de alguna clase. Guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito general de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Se manifiesta en gran parte de la solución al ser creadas instancias de las clases necesarias para ejecutar las funcionalidades, ejemplo de esto cuando se crea un nuevo recurso, se hace necesario instanciar la clase resource.php que se encuentra en LearningResourceBundle.

- **Alta Cohesión:** El objetivo de este patrón es asignar responsabilidades de tal forma que la cohesión siga siendo alta. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Es utilizado en la inyección de dependencias de la clase controladora, cuando se pasa (inyecta) a esta clase todos los objetos que necesita (dependencias) ya creados y configurados.

- **Controlador:** El propósito de este patrón es asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. El controlador maneja todas las peticiones web en el componente. Se evidencia en la solución cuando la clase controladora convierte una petición generada por el usuario en la respuesta de un método que utiliza una clase específica del SDK de Dropbox, convirtiéndose así en un controlador del flujo de eventos generados en el componente.

- **Bajo Acoplamiento:** El bajo acoplamiento soporta el diseño de clases más independientes, que reducen el impacto de los cambios. Este patrón especifica cómo dar soporte a una dependencia escasa y a un aumento de la reutilización. Se pone de evidencia al utilizar la inyección de dependencias, que además de crear una alta cohesión, permite crear un bajo acoplamiento entre las clases, pues se asignan responsabilidades a clases específicas en el sistema y se utilizan según sea necesario.

2.7.2 Patrones GOF

Los patrones GOF se descubren como una forma indispensable de enfrentarse a la programación a raíz del libro "*Design Patterns, Elements of Reusable Software*", a partir de entonces estos patrones

son conocidos como los patrones de la pandilla de los cuatro (GOF, *gang of four*), haciendo referencia a los cuatro autores de este libro. (53) Algunos utilizados por el *framework* de desarrollo Symfony presentes en la solución son:

- **Decorador (Envoltorio):** Añade funcionalidad a una clase, dinámicamente. El archivo layout.php, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página.

- **Fachada:** Symfony, mediante el sistema de configuración de archivos .yml, implementa este patrón permitiendo acceder a diferentes configuraciones del *framework* desde un único lugar. Además de que el acceso a la aplicación es a través del controlador frontal que implementa este patrón.

- **Adapter (Adaptador):** Symfony da la posibilidad de cambiar a otro sistema de base de datos completamente diferente a mitad de desarrollo, solo basta con configurar un archivo .yml. La capa de abstracción utilizada encapsula toda la lógica de los datos. El resto de la aplicación no tiene que preocuparse por las consultas SQL y el código SQL que se encarga del acceso a la base de datos.

2.8 Modelado de diseño

Antes de pasar a las particularidades del diseño, se hace necesario justificar por qué no se realiza el análisis. Como un elemento determinante se tiene el hecho de que las acciones de los usuarios que interactúan con el componente son muy sencillas y básicas aún para cualquier persona con pocos conocimientos de informática (dígase autenticarse, adjuntar un archivo para subirlo, descargar archivos, etc.), por lo que el proceso de captura de requisitos fue claro y explícito, y por la parte del desarrollador la manera de implementar el acceso e interacción con los Sistemas de Almacenamiento en la Nube está definido por estos mismos a través de las funcionalidades y características de sus respectivas API, que viene descrita en la documentación para desarrolladores en cada uno de estos sistemas. Dentro de la metodología en uso, mediante el análisis los requisitos pueden ser refinados y estructurados para conseguir una mejor comprensión y estructuración del sistema (35), pero por lo antes expuesto, el desarrollador decidió ofrecer la visión general del sistema a través del estudio de los resultados del diseño y la implementación.

El modelado del diseño crea un modelo de software enfocado en la representación de los datos, las funciones y el comportamiento requerido. Permite al ingeniero de software modelar el sistema o producto que se va a construir, posibilitando evaluar su calidad y efectuar mejoras antes de generar el código. (54)

2.8.1 Diagramas de clases del diseño

Los diagramas de clases del diseño representan la vista estática del sistema y modelan los conceptos asociados al dominio de la aplicación, así como los conceptos internos definidos para la parte de la implementación. No se describe el comportamiento del sistema dependiente del tiempo y se representa mediante clases y sus relaciones (46).

A continuación, se presenta el diagrama de clases del diseño del requisito funcional añadir archivo como recurso educativo en los cursos de ZERA 2.0, el resto de los diagramas desarrollados se encuentran en el Anexo #2 y son los más representativos según la importancia de las funcionalidades.

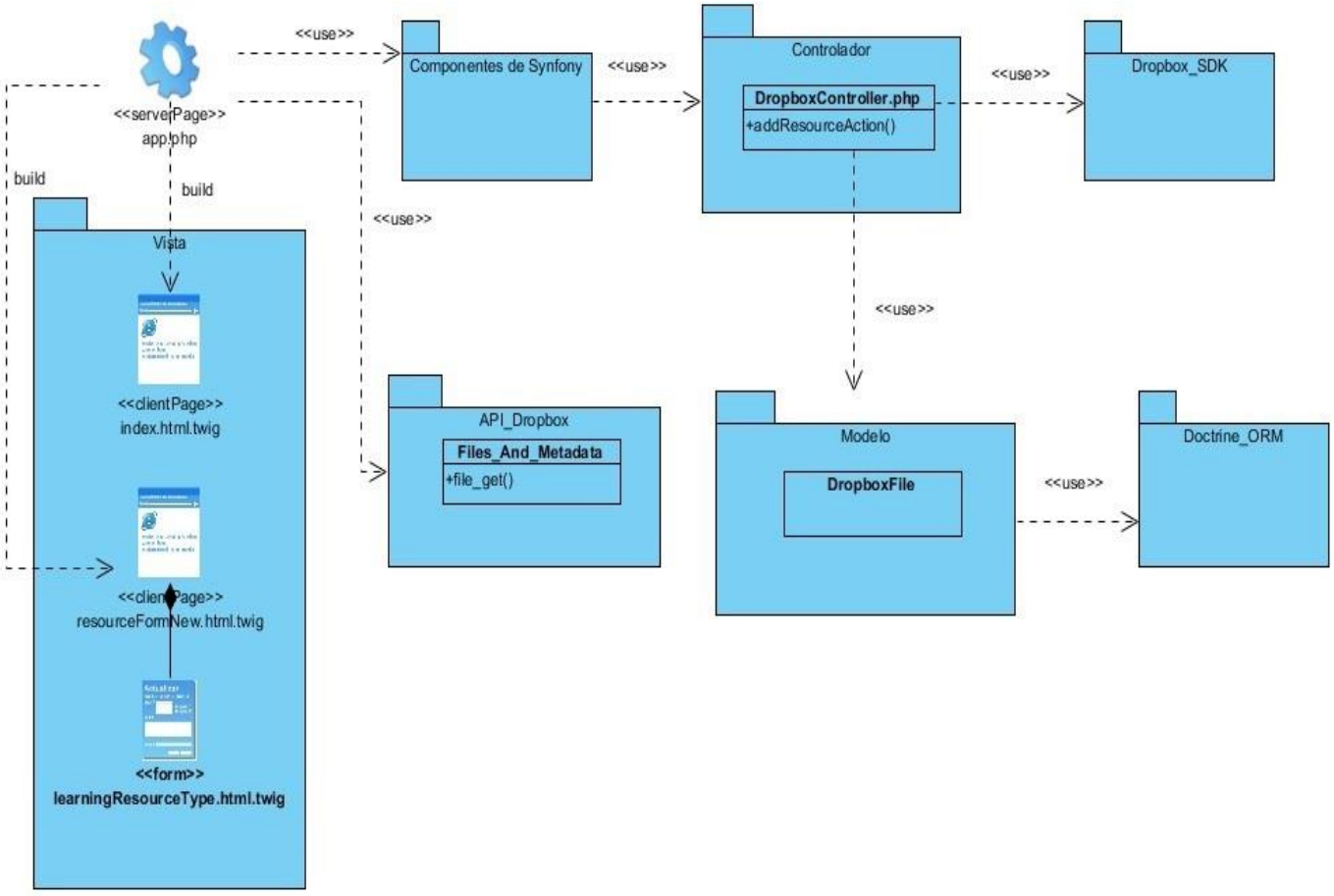


***Figura 3**. DCD Añadir archivo almacenado en los Sistemas de Almacenamiento en la Nube como recurso educativo en los cursos de ZERA 2.0.*

2.8.2 Diagramas de secuencia del diseño

Un diagrama de secuencia muestra un conjunto de mensajes, dispuestos en una secuencia temporal, donde cada rol en la secuencia se muestra como una línea de vida que representa el rol durante cierto plazo del tiempo, con la interacción completa. Los mensajes se muestran como flechas entre las líneas de vida. Un diagrama de secuencia puede mostrar un escenario, es decir, una historia individual de una transacción (46).

A continuación, se presenta el diagrama de secuencia del diseño del requisito funcional añadir archivo como recurso educativo en los cursos de ZERA 2.0, el resto de los diagramas desarrollados se encuentran en el Anexo #3 y son los más representativos según la importancia de las funcionalidades.

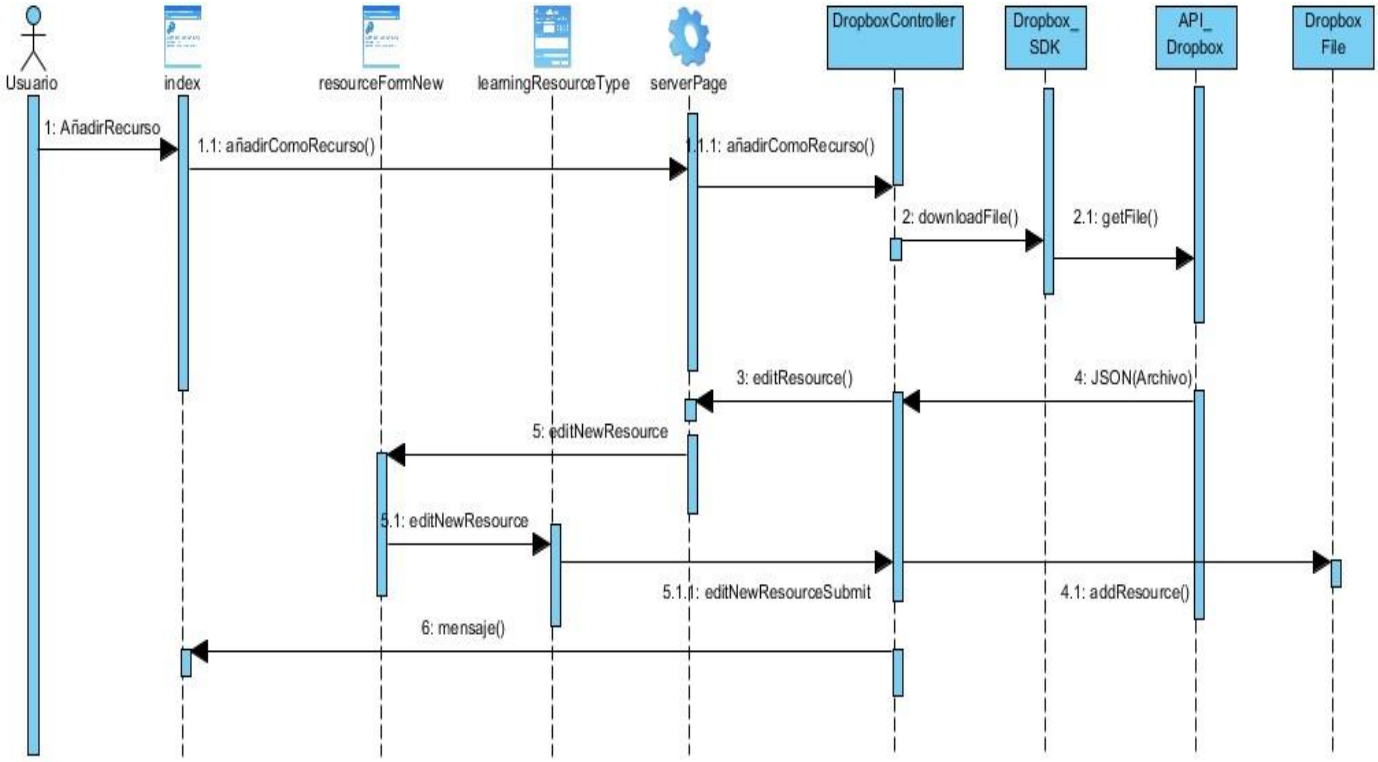


***Figura 4**. DSD Añadir archivo almacenado en los Sistemas de Almacenamiento en la Nube como recurso educativo en los cursos de ZERA 2.0.*

2.9 Conclusiones parciales

En este capítulo quedaron expuestos los principales conceptos del entorno en el que se encuentra enmarcado el componente a desarrollar, sus características y diseño. Mediante el diagrama del modelo

de dominio se obtuvo una mejor comprensión del entorno donde coexiste el problema, definiéndose a partir de este los principales conceptos y sus relaciones. La calidad y exactitud en la captura de los requerimientos, permitió definir las funcionalidades, características y diseño que deberá cumplir el componente, con el objetivo de satisfacer las necesidades del cliente. La aplicación del patrón arquitectónico Modelo Vista Controlador (MVC) permitirá expresar un correcto esquema organizativo estructural del sistema. La utilización de patrones de diseño permitió aplicar buenas prácticas en el proceso de modelado e implementación del software, logrando una mayor reutilización y mantenibilidad del código. De la realización del modelado del sistema se obtuvieron los diagramas de clases del diseño y de secuencia, los cuales guiaron al desarrollador para la construcción de los objetos, atributos y comportamientos de las clases, su relación y la forma en la que interactúan las mismas.

Capítulo 3: Implementación y Prueba de la solución propuesta

En este capítulo se presentan las fases de implementación y de prueba del componente realizado, así como el diagrama de componente. Partiendo del diseño elaborado en el anterior capítulo se comienza con la implementación para dar paso a la etapa de pruebas; en esta última se realiza el diseño de los casos de pruebas para los requerimientos funcionales del componente y los resultados de las pruebas.

3.1 Diagrama de componentes

Los diagramas de componentes permiten modelar la vista de implementación del sistema a partir del cual se construye la aplicación, se representan las dependencias entre los componentes para poder determinar el impacto de un cambio y modela además la asignación de clases y otros elementos del modelo a los componentes. (46)

La siguiente figura muestra el diagrama de componentes general. Este se compone por tres paquetes fundamentales, el modelo, la vista y el controlador. En el modelo se encuentran las clases entidad, que se asocian al Doctrine ORM. La vista contiene las páginas que se mostrarán en el componente, relacionadas con los paquetes css, js e img. El controlador está formado por la clase controladora, que se asocia con el modelo, la vista, el paquete Symfony *vendor*, que contiene el núcleo del *framework* y el SDK de Dropbox.

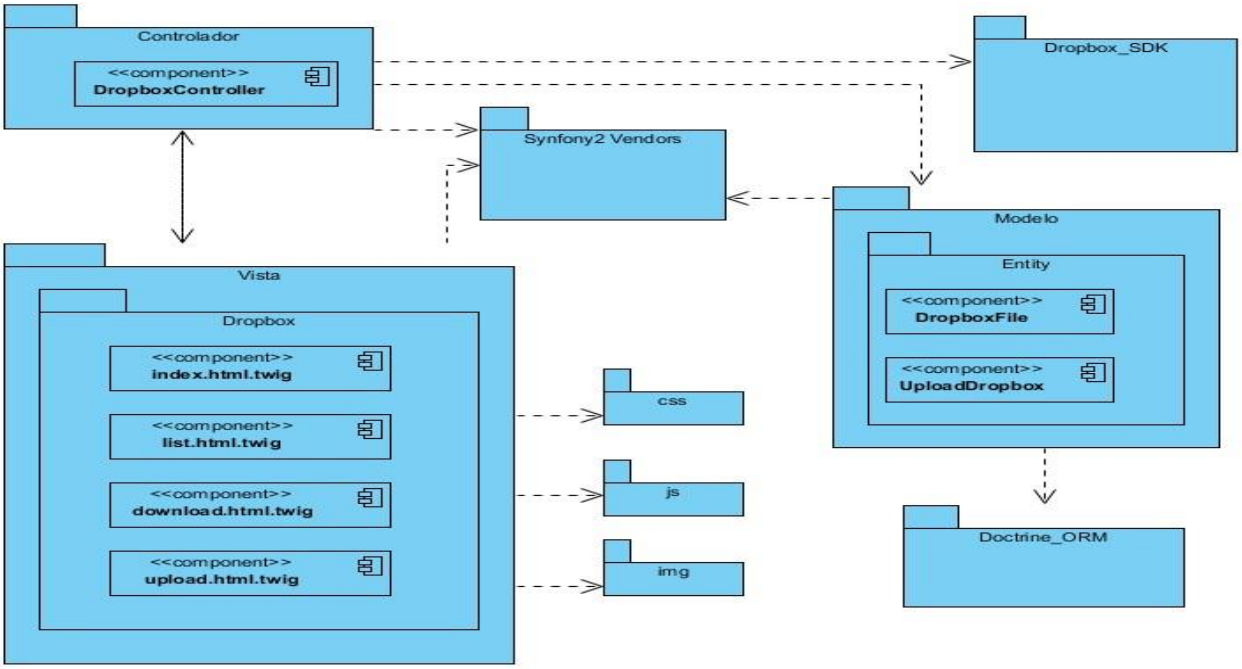


***Figura 5**. Diagrama de componentes general.*

3.2 Estándares de codificación

Un estándar de codificación comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de software, se debe establecer un estándar de codificación para asegurar de que todos los programadores del proyecto trabajen de forma coordinada y utilizarlo desde el inicio hasta el final del desarrollo. (55)

Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento (55). Algunos de los establecidos por ZERA 2.0 utilizados para el desarrollo del componente son:

- Utilizar mayúsculas intercaladas -sin guiones bajos- en nombres de variable, función, método o argumentos.
- Un espacio de nombres y la clase completamente calificado debe tener la siguiente estructura \<*Vendor Name*>\(<*Namespace*>\)*<*Class Name*>.
- PHP debe usar las etiquetas <?php ?> o la etiqueta corta <?= ?>, no debe usarse otra variante.
- Las llaves de las clases se deben abrir en la línea siguiente a donde se comenzó a declarar esta, y cerrarse en una línea después del cuerpo de la clase.
- La visibilidad de las propiedades y métodos se deben declarar siempre; *abstract* y *final* se deben declarar antes de la visibilidad; *static* se debe declarar después de la visibilidad.
- Las líneas no deben exceder de 80 caracteres, las que sean más largas se deben segmentar en líneas de no más de 80 caracteres cada una.
- No debe existir más de una instrucción por línea.

3.3 Resultado de la Implementación

Como resultado de la implementación se obtuvo un componente que permite a los usuarios de la plataforma educativa ZERA 2.0 almacenar los recursos educativos en el sistema de almacenamiento en la nube Dropbox. A continuación, se muestra la interfaz principal que se visualizará cuando se defina el origen del recurso educativo desde la nube, luego de que el usuario se autentique en Dropbox.

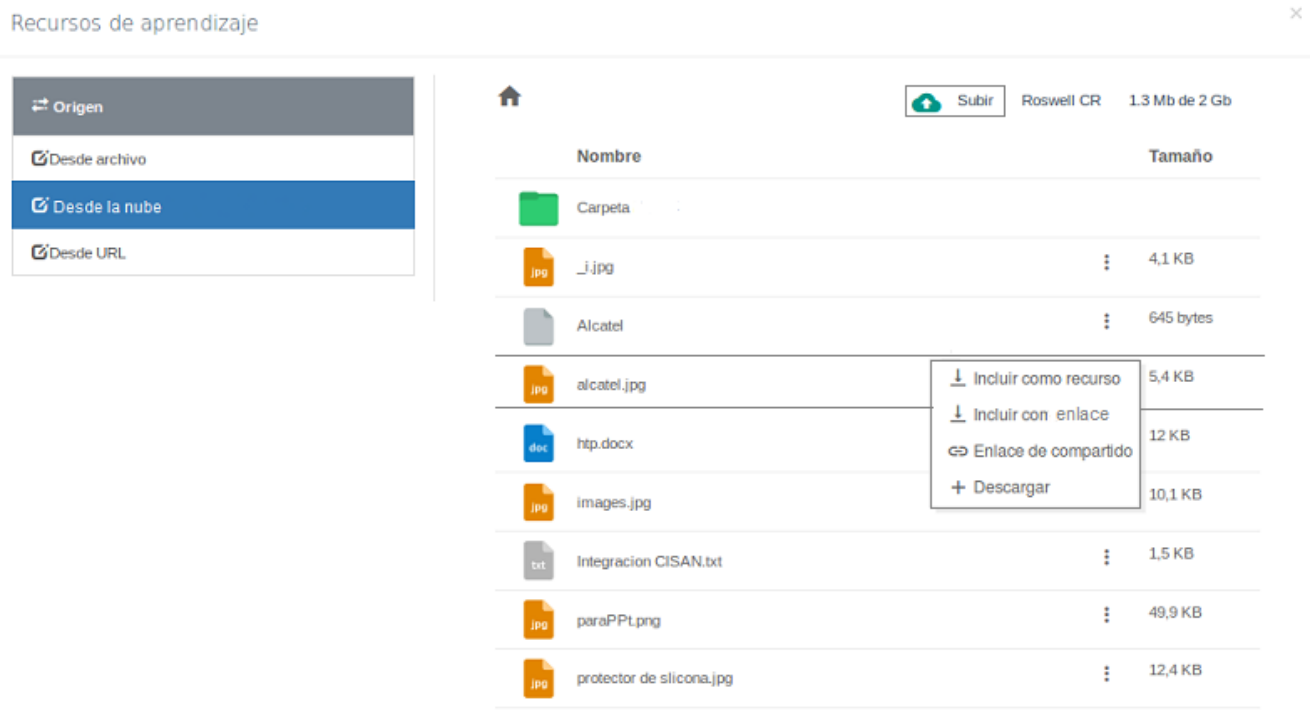


Figura 6. *Interfaz principal del componente.*

En la figura anterior se muestran las funcionalidades que podrá ejecutar el usuario, tales como, subir archivo, incluirlo como recurso educativo, incluir su enlace, descargarlo y obtener el enlace de compartido. También se muestra la cantidad de espacio total y el disponible de la cuenta.

3.4 Pruebas de software

El único instrumento adecuado para determinar el status de la calidad de un producto software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad (56). Las pruebas son un conjunto de actividades para asegurar que el software implemente correctamente una función específica y que se corresponda con los requisitos del cliente (54).

Implementar el proceso de pruebas de software es la vía correcta para asegurar que el producto cumple con la calidad deseada por el cliente. Con esto no se consigue un sistema íntegramente libre de errores, pero si apto para ser usado por el usuario final.

3.4.1 Niveles de pruebas

El proceso de realización de las pruebas está compuesto por una serie de niveles entre los que se encuentran: el nivel de pruebas unitarias, el de pruebas de integración, el de pruebas del sistema y el de pruebas de aceptación (54). Una vez implementado el componente, fue sometido a los niveles de pruebas que se describen en lo adelante, los cuales ayudaron a la detección de errores existentes.

Pruebas de integración

La necesidad de realizar las pruebas de integración viene dada por el hecho de que los módulos que forman un programa suelen fallar cuando trabajan de forma conjunta, aunque previamente se haya demostrado que funcionan correctamente de manera individual. Con el uso de estas pruebas se consigue ir formando el programa global a medida que se comprueba como los distintos componentes interaccionan y se comunican libres de errores (57).

Las pruebas de integración se realizan para garantizar que los componentes del modelo de implementación funcionan correctamente cuando se combinan para ejecutar un guion de uso. El destino de esta prueba es el componente realizado con sus respectivas funcionalidades y ZERA 2.0 tomado como el sistema en el cual este se encuentra. Al realizarse este tipo de pruebas, se aseguró que el componente una vez instalado ejecutaba correctamente las funcionalidades implementadas y no repercutía negativamente en el correcto funcionamiento de ZERA 2.0.

3.4.2 Métodos de pruebas

Existen dos métodos de pruebas fundamentales con el objetivo de validar el software, estas son: pruebas de Caja Blanca y las pruebas de Caja Negra. Para la validación de la propuesta de solución será aplicado el método de Caja Negra, debido a que este permitirá corregir problemas de rendimiento, en la interfaz de usuario y se comprobará que el componente realiza las funciones solicitadas por el usuario.

No se realizarán pruebas de Caja Blanca ya que la lógica y la estructura interna del código para la comunicación con la API de Dropbox se encuentra previamente definida en la documentación de este sistema.

Método de Caja Negra

Las pruebas de Caja Negra son las que se llevan a cabo sobre la interfaz del software. Se concentran en los requisitos funcionales del software. Es decir, permiten derivar un conjunto de condiciones de entrada que ejercitarán los requisitos funcionales de un programa. Encuentran errores en las siguientes

categorías: funciones incorrectas o faltantes, errores de interfaz, errores en estructuras de datos o en acceso a bases de datos externas, errores de comportamiento o desempeño y errores de inicialización y término (54).

3.4.3 Técnicas empleadas

Para el desarrollo de las pruebas de Caja Negra se utilizó la técnica de partición de equivalencia. A partir de esta se elaborarán los casos de pruebas que son aplicados para validar que el componente cumple con los requerimientos deseados.

Partición de equivalencia

Es una técnica de caja negra que se basa en dividir en subconjuntos equivalentes respecto a una relación específica (clase de equivalencia) el dominio de las entradas. La prueba de un valor representativo de una clase permite suponer que el resultado obtenido será el mismo que para otro valor de la clase. Se realiza un conjunto representativo de casos de prueba para cada clase de equivalencia. (58)

Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Regularmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica. (54)

3.4.4 Diseño de casos de pruebas

El diseño de casos de prueba es una parte de las pruebas de componentes y sistemas en las que se diseñan las entradas y las salidas esperadas para probar el sistema, el objetivo que persigue es crear un conjunto de casos de prueba que sean efectivos descubriendo defectos en los programas y muestren que el sistema satisface los requerimientos. (50)

El diseño de casos de pruebas tiene como objetivo principal comprobar si los requerimientos del componente son cumplidos o no, parcial o completamente y proveer una descripción del comportamiento de las funcionalidades bajo determinadas acciones. Las celdas de las tablas que se muestran a continuación pueden contener valores tales como: V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario suministrar un valor del dato, pues este es irrelevante.

Tabla 4. *DCP Añadir archivo almacenado en los Sistemas de Almacenamiento en la Nube como recurso educativo en los cursos de ZERA 2.0.*

<table>
<tr><td colspan="9">CP Añadir archivo almacenado en los Sistemas de Almacenamiento en la Nube como recurso educativo en los cursos de ZERA 2.0.</td></tr>
<tr><td colspan="9">Descripción general: El caso de prueba inicia cuando el actor decide añadir un archivo de los almacenados en los Sistemas de Almacenamiento en la Nube a un curso de ZERA 2.0. Para ello el actor selecciona la opción “Añadir como recurso”.

Condiciones de ejecución: El usuario debe estar autenticado en el Sistema de Almacenamiento en la Nube.

SC Añadir como recurso</td></tr>
<tr><th>Escenario</th><th>Descripción</th><th>Nombre</th><th>Descripción</th><th>Tipo de recurso</th><th>Categoría</th><th>Curso</th><th>Respuesta del sistema</th><th>Flujo central</th></tr>
<tr><td>EC 1.1 Selecciona “Añadir como recurso educativo”.</td><td>El actor selecciona la opción de añadir un archivo como recurso educativo en los cursos de ZERA 2.0.</td><td>N/A</td><td>N/A</td><td>N/A</td><td>N/A</td><td>N/A</td><td>El componente debe añadir el archivo seleccionado como recurso educativo al curso. Se brinda la posibilidad de cambiar o seleccionar los siguientes datos sobre el archivo a la hora de añadirlo al curso:
Nombre
Descripción
Tipo de recurso
Categoría
Curso</td><td>Lista Archivos/ Añadir como recurso</td></tr>
</table>

EC 1.2 Selecciona la opción "Guardar Cambios".	Selecciona o cambia alguno de los datos del formulario. Selecciona la opción "Guardar Cambios".	V	V	V	V	V	Valida los datos. Se crea un recurso educativo con los datos del formulario. Se muestra un mensaje de información.	Lista Archivos/ Añadir como recurso
EC 1.3 Existen datos obligatorios incompletos o incorrectos.	Existen datos obligatorios incompletos o incorrectos en el formulario.	I	V	V	V	V	Muestra un mensaje de información. Muestra un indicador sobre los campos vacíos.	Lista Archivos/ Añadir como recurso
		V	I	V	V	V		
		V	V	I	V	V		
		V	V	V	I	V		
		V	V	V	V	I		

3.4.5 Análisis de los resultados

Después de aplicar el método de Caja Negra y generar a partir de este los casos de prueba se demostró que las funcionalidades del componente son operativas. En el transcurso de este proceso se obtuvieron entradas aceptables y se produjeron resultados correctos, aunque se detectaron algunas no conformidades relacionadas con funcionalidades y errores en la interfaz.

Para ejecutar satisfactoriamente la detección de no conformidades se realizaron 3 iteraciones. Algunas estuvieron estrechamente relacionadas con un parámetro de la librería CURL que impedía la conexión a través del proxy UCI con el Sistema de Almacenamiento en la Nube, lo que impedía el correcto funcionamiento del componente en algunas PC que no contaban con algún programa con la configuración correcta del puerto y el proxy para la navegación en internet. También se detectaron errores de interfaz gráfica y de validaciones. En la primera iteración realizada se detectaron 16 no conformidades, dando solución a 16 de ellas. En la segunda iteración de pruebas se encontraron 7 no conformidades, a las cuales se les dio solución en su totalidad, dándole paso a una tercera y última iteración de pruebas, en la cual no se detectaron no conformidades, quedando de esta manera el

componente listo para su correcto funcionamiento. A continuación, se muestran los resultados de las pruebas:

Tabla 5. *Resultados de las pruebas.*

Iteraciones	No conformidades detectadas	No conformidades resueltas
Iteración 1	16	16
Iteración 2	7	7
Iteración 3	0	0

3.5 Conclusiones parciales

A partir de las fases de implementación y prueba se obtuvo como resultado el componente con las funcionalidades requeridas por el cliente. Del proceso de implementación se obtuvo el diagrama de componentes, que permitió representar la vista de implementación del sistema. El uso de estándares de codificación ayudó a mejorar el entendimiento del código a la hora de integrar el componente y propició la utilización de buenas prácticas de programación. Las pruebas de integración garantizaron el correcto funcionamiento del componente una vez instalado en ZERA 2.0. Para la validación del software se realizaron las pruebas mediante el método de caja negra, que permitió detectar un grupo de no conformidades en cada iteración, las cuales fueron resueltas en su totalidad, trayendo consigo la realización de mejoras sustanciales en la implementación de funcionalidades e interfaces para así obtener el producto final deseado.

Conclusiones Generales

El desarrollo de la presente investigación permitió la integración de la plataforma educativa ZERA 2.0 con los Sistemas de Almacenamiento en la Nube para extender el almacenamiento y uso de los recursos educativos generados. De esta manera se puede concluir que:

- El análisis del marco teórico de la investigación a partir de los métodos teóricos y empíricos facilitó conocer el estado de soluciones similares y de los Sistemas de Almacenamiento en la Nube viables para la integración con ZERA 2.0, además de permitir una adecuada selección de las tecnologías y herramientas para el desarrollo del componente.

- La definición de los requisitos del sistema en conjunto con el diseño de clases posibilitó la definición de las funcionalidades del componente para cumplir con los objetivos planteados.

- Mediante las pruebas a las que fueron sometidas las funcionalidades del componente una vez desarrollado, permitió validar su correcto funcionamiento y de esta forma verificar que se consumaran los objetivos propuestos para dar solución a la problemática planteada.

Recomendaciones

Al finalizar el presente Trabajo de Diploma se recomiendan un conjunto de aspectos considerados importantes, para ser tomados en cuenta para su mejora. A continuación, se enuncian:

- Extender el uso de los servicios de otros Sistemas de Almacenamiento en la Nube en el componente.
- Ampliar las opciones sobre los archivos en el componente, dígase renombrar, eliminar, entre otras.

Referencias bibliográficas

1. **LAPEYRE, Juan.** *El espacio pedagógico de las TIC.* Lima : s.n., 2015.

2. *La mediación de las TIC en la enseñanza-aprendizaje de la educación superior.* **GALLAR Pérez, Yamirlis, RODRÍGUEZ Saldívar, Iris Esther y BARRIOS Queipo, Enrique Aurelio.** 6, Las Tunas, Granma : Didasc@lia: Didáctica y Educación., 2015, Vol. XI. ISSN 2224-2643.

3. *El e-learning, una respuesta educativa a las demandas de la sociedad del siglo XXI.* **BAELO Álvarez, Roberto.** 35, Universidad de León, España : Pixel-Bit. Revista de Medios y Educación, 2009.

4. **LINTI- Laboratorio de Investigación en nuevas tecnologías informáticas. Facultad de informática.** *Integrando un Repositorio Digital de Objetos de Aprendizaje con Servicios que Promuevan su Uso y Mantenimiento.* La Plata, Argentina : LACLO, 2014.

5. *Importancia del uso de estándares en las plataformas tecnológicas educativas.* **RAMÍREZ Hernández, Moramay, DÍAZ Alva, Angelina y TÉLLEZ Barrientos, Omar.** México : Centro de Estudios e Investigaciones para el Desarrollo Docente. CENID A.C., 2015. ISSN: 2395-8006.

6. **JANSEN, Wayne y GRANCE, Timothy.** *Guidelines on Security and Privacy in Public Cloud Computing.* Gaithersburg : NIST Special Publication, 2011.

7. **UC Berkeley Reliable Adaptive Distributed Systems Laboratory.** *Above the Clouds: A Berkeley View of Cloud Computing.* California : s.n., 2010.

8. **SCHIAVÓN Raineri, Ignacio Nicolás.** *VOCS, Sistema de almacenamiento voluntario en la nube.* Universidad Carlos III de Madrid : s.n., 2013.

9. *Plataformas Educativas, un Entorno para Profesores y Alumnos .* **DÍAZ Becerro, Sebastián.** 2, Andalucía, España : Revista Digital para Profesionales de la Enseñanza, 2009. ISSN: 1989-4023.

10. **ÁVILA Jiménez, José Luis, GÁMEZ Granados, Juan Carlos y VENTURA Soto, Sebastián.** *Plataformas de enseñanza virtual.* Córdoba : Revista Electrónica Universidad de Jaén, 2013.

11. **Centro de Comunicación y Pedagogía. [En línea] [Citado el: 2 de 6 de 2016.] https://www.centrocp.com/lms-y-lcms-funcionalidades-y-beneficios/.**

12. *Computación en la Nube, Notas para una estrategia española en Cloud Computing.* JOYANES Aguilar, Luis. 0, Universidad Pontífica de Salamanca : Revista del Instituto Español de Estudios Estratégicos, 2012.

13. Nexsys. Nexsys, Cloud Computing. [En línea] [Citado el: 14 de 3 de 2013.] http://cloud.nexsysla.com/?q=cloud_computing.

14. GOYAS Gutiérrez, Marco Agustín y VARGAS Cruz, Jhonny Daniel. *Almacenamiento en La Nube.* Escuela Superior Técnica del Litoral, Guayaquil : s.n., 2014.

15. ***Computación en la Nube, Tendencia de Importancia y Transcendencia en la Educación Superior.*** **BALLESTEROS Ricaurte, Javier Antonio y MEJÍA Ortega, Iván Darío. Tunja, Colombia : Revista Ingenio Magno, 2014, Vol. 5.**

16. National Institute of Standards and Technology, NITS. ***The NIST Definition of Cloud Computing.*** **Gaithersburg : U.S. Department of Commerce, 2011.**

17. CASARRUBIO Prieto, Domingo. ***Uso de las API de Dropbox y Google Drive para la integración de datos en la nube.*** **Universidad Politécnica de Valencia, España : s.n., 2014.**

18. VELTE, Toby, VELTE, Anthony y ELSENPETER, Robert. ***Cloud Computing, A Practical Approach.*** **Nueva York : McGraw-Hill, Inc., 2009. ISBN:0071626948 9780071626941.**

19. SOSINSKY, Barrie. ***Cloud Computing Bible.*** **Indianapolis : Wiley Publishing, Inc., 2011. ISBN: 978-0-470-90356-8.**

20. DE LA HOZ Freyle, Javier, CARRILLO Rincón, Elberto y GÓMEZ Flórez, Luis Carlos. ***Memorias organizacionales en la era del almacenamiento en la nube.*** **Universidad Industrial de Santander, Colombia : s.n., 2013.**

21. Drpbox, Inc. [En línea] [Citado el: 2 de 5 de 2016.] https://www.dropbox.com/help/topics/security_and_privacy.

22. Fayerwayer. [En línea] 25 de 6 de 2015. [Citado el: 7 de 5 de 2016.] https://www.fayerwayer.com/2015/06/dropbox-ya-tiene-mas-de-400-millones-de-usuarios/.

23. Dropbox Inc. [En línea] [Citado el: 4 de 4 de 2016.] https://www.dropbox.com/es.

24. alphr.com. [En línea] 3 de 5 de 2016. [Citado el: 20 de 5 de 2016.] http://www.alphr.com/dropbox/1000326/how-secure-are-dropbox-microsoft-onedrive-google-drive-and-apple-icloud-cloud.

25. Google. [En línea] [Citado el: 6 de 4 de 2016.] https://www.google.com/intl/es-419/drive/.

26. Microsoft Corporation. [En línea] [Citado el: 10 de 4 de 2016.] www.windowscentral.com/onedrive.

27. ***BoxCryptor.*** **[En línea] [Citado el: 23 de 5 de 2016.] https://www.boxcryptor.com/es/boxcryptor.**

28. CLARENC, Claudio Ariel, y otros. ***Analizamos 19 plataformas de e-learning, Investigación colaborativa sobre LMS.*** **2013. ISBN 978-1-291-53343-9.**

29. DÍAZ, Javier. javierdisan. [En línea] 2 de 2 de 2015. [Citado el: 1 de 3 de 2015.] http://javierdisan.com/2015/02/02/informe-horizon-2015/.

30. octeto. [En línea] Universidad Jaume I. [Citado el: 5 de 3 de 2016.] http://cent.uji.es/octeto/node/4469.

31. ***Educación en la nube. Un nuevo reto para los docentes de Educación Media Superior.*** **TORRES Sánchez, Sara Marlen. 10, Colegio de Bachilleres del Estado de Querétaro : Revista Iberoamericana para la Investigación y el Desarrollo Educativo, 2013. ISSN 2007 - 2619.**

32. SOMMERVILLE, Ian. ***Ingeniería de Software.*** **2002.**

33. Academia. [En línea] [Citado el: 9 de 2 de 2016.] http://www.academia.edu/9953322/Metodologias_para_el_desarrollo_de_software.

34. ***Estudio de metodologias ágiles para proyectos de software en corto tiempo.*** **GUALTEROS Gualteros, Ana Celmira y ORJUELA Escobar, Diana Paola. 1, Universidad Distrital Francisco José de Caldas, Bogotá : TIA, 2013, Vol. 2. ISSN: 2344-8288.**

35. RODRÍGUEZ Sánchez, Tamara . ***Metodología de desarrollo para la Actividad productiva de la UCI. Programa de mejora.*** **Universidad de las Ciencias Informaticas, La Habana : s.n.**

36. HowStuffWorks. [En línea] InfoSpace LLC. [Citado el: 10 de 2 de 2016.] http://money.howstuffworks.com/business-communications/how-to-leverage-an-api-for-conferencing1.htm.

37. readwrite. [En línea] [Citado el: 8 de 2 de 2016.] http://readwrite.com/2013/09/19/api-defined.

38. Programación Desarrollo. [En línea] [Citado el: 3 de 2 de 2016.] http://programaciondesarrollo.es/¿que-es-sdk/.

39. Box. [En línea] [Citado el: 4 de 2 de 2016.] https://box-content.readme.io/docs/sdks-faq.

40. Object Management Group. ***OMG Unified Modeling LanguageTM (OMG UML), Superstructure.*** **s.l. : OMG, 2011.**

41. ***Workshop Component Oriented Programming (WCOP). Bruselas, Bélgica : s.n., 1998. ISBN: 3540654607, 9783540654605.***

42. Designing Reusable Classes. FOOTE, Brian y JHONSON, Ralph E. 2, University of Illinois : Journal of Object-Oriented Programming, 1998, Vol. 1.

43. GUTIÉRREZ, Javier J. cssblog.es. [En línea] [Citado el: 6 de 2 de 2016.] http://www.cssblog.es/guias/Framework.pdf.

44. POTENCIER, Fabien y ZANINOTTO, François. Symfony la guía definitiva. 2008.

45. Doctrine-Project. Doctrine. [En línea] [Citado el: 15 de 6 de 2016.] http://www.doctrine-project.org/.

46. RUMBAUGH, James, BOOCH, Grady y JACOBSON, Ivar. El Lenguaje Unificado de Modelado. Manual de Referencia. Madrid, España : Addison Wesley, 2000. ISBN: 84-7829-037-0.

47. Visual Paradigm. [En línea] [Citado el: 5 de 3 de 2016.] https://www.visual-paradigm.com/features/.

48. NetBeans. [En línea] Oracle Corporation. [Citado el: 5 de 3 de 2016.] https://netbeans.org/features/index.html#o2.

49. CRAIG, Larman. UML y Patrones. Introducción al análisis orientado a objetos. 1999.

50. SOMMERVILLE, Ian. Ingeniería de software 7ma edición. s.l. : Pearson-Addison Wesley, 2005.

51. PRESSMAN, Roger. Ingeniería de Software, un enfoque práctico. Madrid : Editorial McGraw-Hill, 2005.

52. OLIVARES Rojas, Juan Carlos. Patrones de Diseño. México : Sistema Nacional de Educación Superior.

53. PÉREZ Mariñán, Martín . Patrones de Diseño (Design Patterns).

54. PRESSMAN, Roger. Ingeniería de Software: Un enfoque práctico 5ta edición. s.l. : McGraw Hill, 2002.

55. Microsoft. Microsoft Developer Network. [En línea] [Citado el: 20 de 4 de 2016.] https://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx.

56. pruebasdesoftware. [En línea] [Citado el: 24 de 4 de 2016.] http://www.pruebasdesoftware.com/laspruebasdesoftware.htm.

57. VENCE Pérez, Jose María. Plan de pruebas de integración. Madrid : s.n., 2010.

58. BLANCO Bueno, Carlos. Ingeniería de Software - Construcción y pruebas. [En línea] [Citado el: 25 de 4 de 2016.] http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-software-ii/materiales/tema1-pruebasSistemasSoftware.pdf..

59. CENDEJAS Valdéz , José Luis. eumed.net. Encicopedia Virtual. [En línea] [Citado el: 8 de 2 de 2016.] http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm.

60. CORDERO, Jorge Luis. Metodologías Ágiles " Proceso Unificado Ágil (AUP)". La Paz, Bolivia : s.n.

61. Percepciones acerca de la integración de las TIC en el proceso de enseñanza-aprendizaje de la universidad. MORALES Capilla, Marina, TRUJILLO Torres, Juan Manuel y RAZO Sanchez, Francisco. 46, Granada : Píxel-Bit. Revista de Medios y Educación., 2015. ISSN: 1133-8482..

62. EGUILUZ, Javier. Desarrollo web ágil con Symfony2. 2013.

63. CARNEIRO, Roberto, TOSCANO, Juan Carlos y DÍAZ, Tamara. Los desafíos de las TIC para el cambio educativo. Madrid : OEI, Fundación Santillana, 2012. ISBN: 978-84-7666-197-0.

64. Blog de Fernando Santamaría. [En línea] [Citado el: 10 de 2 de 2016.] http://fernandosantamaria.com/blog/2012/11/nuevas-plataformas-de-aprendizaje-en-el-contexto-de-educacion-superior-global-y-ii/.

65. BuenasTareas. [En línea] 13 de 1 de 2014. [Citado el: 28 de 2 de 2016.] http://www.buenastareas.com/ensayos/Conceptos-De-Material-Did%C3%A1ctico-Medio-Did%C3%A1ctico/46304736.html.

66. [En línea] [Citado el: 2016 de 2 de 29.] http://plataformas-educativas.blogspot.com/2009/11/plataformas-educativas.html.

67. DE NATA Pérez, Héctor. Mejora del Sistema de almacenamiento voluntario en la nube. Sistema de etiquetas. Universidad Carlos III de Madrid, España : s.n., 2013.

68. ProgramaciónDesarrollo.es. [En línea] [Citado el: 5 de 3 de 2016.] http://programaciondesarrollo.es/que-es-un-entorno-de-desarrollo-integrado-ide/.

69. Institute of Electrical and Electronics Engineers. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York : s.n.

70. LibrosWeb.es. [En línea] [Citado el: 20 de 5 de 2016.] http://librosweb.es/libro/composer/capitulo_2/packagist.html.

71. Dropbox. Core API. Dropbox. [En línea] [Citado el: 4 de 2 de 2016.] https://www.dropbox.com/developers-v1/core/docs#delta.

72. Google. Google APIs. [En línea] [Citado el: 6 de 2 de 2016.] https://console.developers.google.com/apis/library?hl=ES.

73. Microsoft Corporation. Microsoft. Getting started with OneDrive. [En línea] [Citado el: 12 de 2 de 2016.] http://windows.microsoft.com/en-us/windows-8/getting-started-onedrive-tutorial.

74. Open Educational Resources Serve the World. JOHNSTONE, Sally M. s.l. : Educause Quarterly, 2005, Vol. 28.

75. Paradiso. [En línea] [Citado el: 1 de 2 de 2016.] https://www.paradisosolutions.com/es/.

76. Facultad de Ingeniería. Universidad Nacional de La Plata. [En línea] [Citado el: 5 de 2 de 2016.] http://www.info.unlp.edu.ar/.

77. VirtusClass. [En línea] [Citado el: 8 de 2 de 2016.] http://www.virtusclass.com/.

78. School a Live. [En línea] [Citado el: 15 de 2 de 2016.] https://www.schoolalive.com/.

79. Boxcryptor. [En línea] [Citado el: 5 de 5 de 2016.] https://www.boxcryptor.com/es/cifrado.

80. HODGES, Jeff, JACKSON, Collin y BARTH, Adam. Section 5.2. HSTS Policy. s.l. : IETF, 2012.

81. Cloud Computing Latinoamérica. [En línea] [Citado el: 23 de 3 de 2016.] http://www.cloudcomputingla.com/2010/05/tipos-de-computacion-en-nube-publica.html.

82. POINTCHEVA, David. How to Encrypt Properly with RSA. s.l. : CryptoBytes, 2002. 1.