Universidad de las Ciencias Informáticas

FACULTAD 4

Componente para la gestión de apuntes y el resaltado de texto de la Plataforma Educativa ZERA 2.0

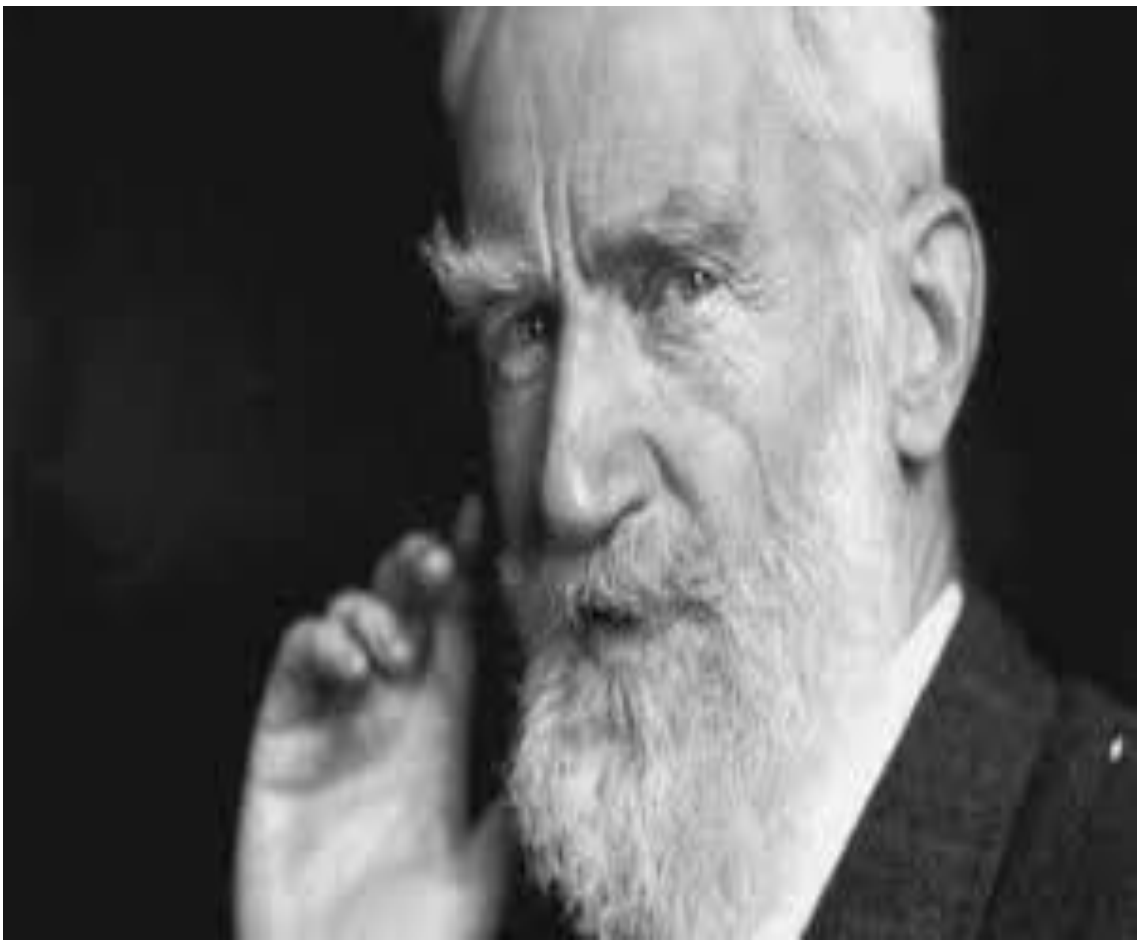
Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas.

Autor: Lisy Maday López Lugo

Tutores: Ing. Alberto Torres Junco.

Ing. Arletys González Madera.

La Habana, junio de 2017



"Solo triunfa en el mundo quien se levanta y busca a las circunstancias, creándolas si no las encuentra"

George Bernard Shaw

DEDICATORIA

Dedico el presente trabajo:

A mis padres: Teresa y Lázaro: por ser esa gran fuerza que he tenido durante estos años siendo mi escudo e inspiración, por hacer de mi lo que soy, por estar siempre presente en cada decisión que he tomada dándome su apoyo incondicional.

A mi hermanita: Sury por ser esa personita que siempre ha estado ahí conmigo, ser mi amiga incondicional, por ser mi confidente por ser mi otra mitad que siempre me ha apoyado en todo.

A mis abuelos: Eloina, Marcelino y Papo: por todo su amor incondicional, por darme tanta fuerza y por estar siempre presente en esta etapa tan importante de mi vida.

A toda mi familia y amigos que siempre creyeron en mí.

AGRADECIMIENTO

Agradezco:

A mis padres por todo su amor y cariño, por ser mi ejemplo a seguir, por darme tanta fuerza para vencer tantos obstáculos que he tenido en el camino. Les agradezco por tener tanta paciencia y dedicación durante este periodo de mi vida los cuales han estado tan presente que siempre han ido al apar conmigo.

A mis abuelos por ser esas personas que siempre me han aconsejados, por su gran amor y dedicación por todo lo que han hecho por mí.

A mi hermana: Sury: por ser tan buena amiga, compañera, por su amor y cariño, por estar siempre a mi lado por acompañarme durante este largo camino.

A mi tía Mary: por ser mi otra mamá por estar siempre presente en todas las etapas de mi vida por su amor y cariño incondicional por preocuparse tanto por mí.

A mi tío Lazarito: por estas siempre al tanto de mí, por su preocupación, por ayudarme sin límites, por todo su cariño y entrega.

A toda mi familia y amigos por todo su apoyo y dedicación en todo momento en especial a mi tío Miguel Ángel, y a mis vecinos que son más que amigos familia Dora y Paulo y su hija Marien.

A mi tutor Alberto: por toda su paciencia dedicación y tiempo incondicional, por guiarme durante todo este tiempo por su preocupación, por ser un tutor ejemplar, por nunca tener un no como respuesta, por su paciencia conmigo cuando lo molestaba más de la cuenta, por todas las cosas que me enseñó.

AGRADECIMIENTO

A mi tutora Arletys: por su gran apoyo, por su entrega por todo ese tiempo que estuvo presente desde el primer día hasta el último.

A Raidel por ser esa persona que siempre ha estado conmigo desde el momento en que lo conocí, por su apoyo, paciencia, su dedicación por su gran cariño durante estos 5 años. Le agradezco por formar parte de mi vida.

A mis amigas por ser mis confidentes por eso buenos momentos que me hicieron pasar junto a ellas Rosmery Gleidys y Rosa.

A mi amigo Ronnay: por todo su tiempo, apoyo incondicional por estar cuando más lo necesitaba, por nunca decir no, gracias por todo.

A mis amigos de grupo (4502) y edificios por esas buenas y malas experiencias que pasamos juntos pero que nunca olvidare, que nos los menciono porque son muchos, pero a todos muchas gracias.

A esos amigos incondicional que conocí durante estos años que siempre estuvieron presente dándome todo su apoyo y aportando su granito de arena en especial a Rodiel, Lijandy y Víctor

A todos esos profesores que desde mi primer año en la universidad fueron fundamentales para mi formación en especial a Desagüe Yuleisy y Graciela

A la Universidad de las Ciencias Informáticas por haberme permitido convertirme en una profesional, por haberme formado tal y como soy por esos valores que me inculcaron. Le agradezco por estos años que forme parte de esta linda familia.

A todas las personas que de una forma u otra aportaron su granito de arena para lograr culminar esta obra.

Declaración de autor

Declaro ser la única autora de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2017.

Lisy Maday López Lugo

_____

Firma de Autor

Ing. Alberto Torres Junco.

_____

Firma de Tutor

Ing. Arletys González Madera.

_____

Firma de Tutora

Resumen

El avance de las Tecnologías de la Información y las Comunicaciones (TIC) ha tenido un impacto significativo en la esfera de la educación. Nuevas herramientas como los Sistemas de Gestión del Aprendizaje(LMS) han sido incorporadas, logrando así el surgimiento de un paradigma educativo más personalizado y centrado en la actividad de los estudiantes. En el año 2016 en la Universidad de las Ciencias Informáticas fue desarrollada la Plataforma Educativa ZERA 2.0, un LMS capaz de guiar el proceso de enseñanza-aprendizaje en cualquier nivel de enseñanza. Esta nueva versión es innovadora, pero aun no logra aprovechar características presentes en la mayoría de los LMS modernos y que ayudan a elevar la calidad del aprendizaje mediante la colaboración entre sus usuarios.

La presente investigación persigue desarrollar un componente que permita elevar las opciones para el desarrollo del trabajo colaborativo y la autogestión del aprendizaje mediante el resaltado de texto y la gestión de apuntes en la Plataforma Educativa ZERA. Para lograrlo, se realizó un estudio de los elementos teóricos y las principales soluciones similares a nivel mundial. Además, se definieron las tecnologías y herramientas para el desarrollo de las funcionalidades propuestas mediante el uso de la metodología de desarrollo AUP-UCI y se realizaron las pruebas pertinentes con el objetivo de validar la calidad de la aplicación.

Palabras Clave: contenido, componente, Plataforma Educativa, trabajo colaborativo.

Índice

ÍNDICE

Índice de tablas

Índice de Figuras

Introducción

Las trasformaciones que hoy operan en el sistema educativo mundial y en Cuba particularmente, demandan del constante perfeccionamiento de los modelos docentes educativos; lo que conlleva a introducir conceptos, estrategias y nuevas formas organizativas mediadas por el empleo de las Tecnologías de la Información y las Comunicaciones(TIC).

Las TIC están siendo insertadas en todas las esferas de la sociedad, propiciando importantes cambios en ellas. La educación constituye una de las esferas donde la utilización de estas tecnologías, permite la creación de nuevas estrategias educativas que apoyan el desarrollo y perfeccionamiento del Proceso de Enseñanza y Aprendizaje (PEA).

Entre las herramientas que constituye un complemento del uso de las TIC en el PEA están los Sistemas de Gestión de Aprendizaje o LMS (*Learning Management System*, por sus siglas en inglés) que representan software o herramientas informáticas basadas en un servidor web que provee módulos para los procesos administrativos y de seguimiento que se requieren para un sistema de enseñanza, simplificando el control de tareas. Los LMS también facilitan el aprendizaje distribuido y colaborativo a partir de actividades y contenidos preelaborados, de forma síncrona o asíncrona, centrados en la actividad de los estudiantes. Además, se emplean para administrar, distribuir y controlar las actividades de formación de una institución u organización.(1)

En el mundo existe gran variedad de LMS que favorecen el PEA tales como *Moodle*[1]*, Canvas*[2]*, Sakai*[3]*, Claroline*[4] *y Chamilo*[5]*.* Cuba no se encuentra exenta de este proceso ya que cuenta con

[1] *Moodle* es un software diseñado para ayudar a los educadores a crear cursos en línea de alta calidad y entornos de aprendizaje virtuales.

[2] *Canvas* (o lienzo traducido al español) es un elemento HTML incorporado en HTML5 que permite la generación de gráficos dinámicamente por medio del scripting.

[3] *Sakai* es un LMS avalado por las mejores universidades del mundo y respaldada por una amplia comunidad de expertos.

instituciones donde se han desarrollado varios LMS como: Mundicampus y Aprendist desarrollados por la Universidad Tecnológica de la Habana (CUJAE) y CEPAD desarrollado por la Universidad Central de las Villas. La Universidad de las Ciencias Informáticas (UCI) desde sus inicios ha trabajado en esta línea, primero creando módulos para extender las funcionalidades de Moodle y luego en el desarrollo de una plataforma educativa propia.

En el Centro de Tecnologías para la Formación de la UCI (FORTES) que tiene como principal objetivo desarrollar tecnologías que permitan ofrecer servicios y productos para la implementación de soluciones de formación aplicando las TIC, fue desarrollada en el periodo 2010-2011 la Plataforma Educativa que lleva por nombre ZERA. (2)

La Plataforma Educativa ZERA fue concebida teniendo presente las necesidades educativas existentes en varios países latinoamericanos y la deficiente infraestructura y conectividad en el sector de la educación, principalmente en países del tercer mundo. Tuvo 3 versiones en la rama 1.x, una primera versión denominada 1.0 la cual fue desplegada en México. ZERA en su primera versión estaba basada en hiper-entornos de aprendizaje; permitiendo la creación de cursos con una organización capitular donde el contenido se muestra como la estructura de un libro. Otras de sus características son el avance del contenido (marcador de libro), el resaltado, los apuntes al contenido, y la creación de 30 tipos de recursos y 11 tipologías de ejercicios. A su vez posibilitaba el diseño de rutas de aprendizaje personalizadas, así como el análisis y actualización de reportes para apoyar la toma de decisiones. Además, incorporaba procesos comunes de la gestión académica y herramientas de colaboración, entre otras funcionalidades. (3)

Posteriormente surge la versión 1.1 teniendo el mismo objetivo que la anterior, pero con algunas transformaciones, aunque esta versión nunca fue desplegada. La última versión de esta rama fue la 1.2 que fue una adaptación a la primera versión para poder hacer uso de la

[4] *Claroline* es una plataforma de aprendizaje (*o LMS: Learning Management System*) y Software colaborativo de código abierto). Permite a cientos de instituciones de todo el mundo crear y administrar cursos y espacios de colaboración en línea.

[5] *Chamilo* LMS es una Plataforma de E-learning de software libre, licenciada bajo la GNU/GPLv3, de gestión del aprendizaje presencial, semi-presencial o virtual.

misma puesto que se hacía necesario realizar algunas variaciones para que se adaptara a los procesos de negocio de cualquier institución educacional cubana.

En el año 2016 se decide desarrollar una nueva versión de ZERA (2.0), reescribiendo el código desde cero y utilizando tecnologías modernas, pero respetando las principales características de la plataforma en las versiones anteriores. Esta versión es innovadora, flexible, fácil de usar, interactiva, adaptable y al igual que sus predecesoras basada en hiperentornos de aprendizaje. Esta plataforma reúne características que permiten enriquecer y guiar el Proceso de Enseñanza y Aprendizaje en instituciones de cualquier nivel de enseñanza; a partir de un conjunto poderoso de herramientas centradas en los estudiantes y ambientes de aprendizaje colaborativo. ZERA es capaz de proporcionar a los profesores y alumnos un sistema integrado en línea único, robusto, seguro y fácil de usar para crear ambientes de aprendizaje personalizados que puedan soportar las necesidades tanto de clases pequeñas, como de grandes organizaciones debido a su flexibilidad y escalabilidad. También fomenta la instrucción con tecnología educativa innovadora que ayuda a los profesores a adaptarse a las nuevas normas y a la personalización del aprendizaje. Ello permite ofrecer experiencias emocionantes de aprendizaje digital; garantizando que todos los estudiantes tengan la oportunidad de desarrollar su potencial.(4)

En este sentido, es de vital importancia destacar que esta nueva versión carece de algunas funcionalidades que estaban presentes en antiguas liberaciones de ZERA. Anteriormente, en las primeras versiones de la plataforma, los usuarios eran capaces de realizar comentarios asociados al contenido de una página, muy similar a los cuadernos de notas utilizados en las aulas tradicionales y algunos procesadores de textos tal como el Microsoft Office Word o al igual que otras plataformas como Moodle.

También, se contaba con un espacio para que los docentes pudieran asociar recursos a los apuntes como método de apoyo al contenido, y a su vez, los mismos pudieran compartir dichos apuntes con sus estudiantes y otros profesores. Además, era posible el resaltado de información de gran relevancia a juicio del usuario, similar a como se realiza en un libro tradicional, permitiendo un mejor entendimiento del contenido y una mejor visualización de estos en la plataforma.

Estas opciones son de gran ayuda en el trabajo colaborativo dentro de la plataforma y permiten la formación de habilidades de interacción de contenido tanto por profesores como por los propios estudiantes. El desarrollo de este tipo de prácticas donde el estudiante es parte activa del proceso de interacción con contenido es una característica que persiguen los LMS modernos.

A partir de la problemática planteada anteriormente surge como **problema a resolver** de la investigación: La ausencia de un componente que permita resaltar texto, hacer apuntes y compartirlos con otros usuarios en ZERA 2.0 disminuyen las opciones para el desarrollo del trabajo colaborativo y la autogestión del aprendizaje.

La investigación en curso enmarca como **objeto de estudio** las herramientas colaborativas en los sistemas de gestión del aprendizaje, delimitándose como **campo de acción,** las herramientas colaborativas para la gestión de apuntes en los Sistemas de Gestión del Aprendizaje.

Para dar solución al problema se define como **objetivo general,** Desarrollar un componente para el resaltado de texto y la gestión de apuntes en la plataforma educativa ZERA 2.0 que permita elevar las opciones para el desarrollo del trabajo colaborativo y la autogestión del aprendizaje.

Para lograr el objetivo general se tendrán en cuenta los siguientes **objetivos específicos:**

- Elaborar el marco teórico de la investigación para obtener los fundamentos necesarios de la propuesta de solución.
- Realizar el diseño de la propuesta de solución para que el componente cumpla con las necesidades requeridas.
- Implementar las funcionalidades del componente de acuerdo con la estructura de diseño definida.
- Realizar pruebas a la solución propuesta a lo largo de todo el ciclo de desarrollo para comprobar el cumplimiento de los requerimientos especificados.

Preguntas científicas

¿Cómo se presenta en la literatura científica el proceso de desarrollo de componentes que contribuyan al trabajo colaborativo mediante la gestión de apuntes?

¿Cómo modelar un componente que gestione resaltados de textos y apuntes en la Plataforma Educativa ZERA 2.0?

¿Cómo desarrollar un componente para la gestión de apuntes y resaltados de textos en la Plataforma Educativa ZERA 2.0?

¿Cómo asegurar la calidad del componente para la gestión de resaltados de textos y apuntes en la Plataforma Educativa ZERA 2.0?

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

- Definición de la metodología, tecnologías, herramientas y lenguajes utilizados en el desarrollo de la solución.
- Elaboración del modelo de dominio o modelo conceptual del sistema.
- Definición los requerimientos funcionales y no funcionales del sistema solicitados por el cliente.
- Realización del modelado de análisis del sistema que se desea desarrollar
- Definición de los patrones de diseño a utilizar en el desarrollo de la propuesta de solución.
- Realización del modelado de diseño de todas las funcionalidades identificadas.
- Implementación de los requerimientos solicitados por el cliente.
- Validación de la propuesta de solución a través de pruebas de software

Como **resultado** de lo anteriormente planteado se espera obtener:

- Un componente que permita el resaltado de texto, la gestión de apuntes y compartirlos con otros usuarios de la plataforma ZERA 2.0, permitiendo elevar las opciones para el desarrollo del trabajo colaborativo y la autogestión del aprendizaje.

- Documentación de todo el proceso de desarrollo de la solución, de manera que pueda continuarse con el mantenimiento de la misma, así como el desarrollo de futuras versiones.

El desarrollo de la investigación requirió del empleo de los métodos científicos que se relacionan a continuación:

Métodos científicos del nivel teórico

- El **método analítico-sintético:** fue utilizado para analizar la documentación y determinar los principales conceptos a incluir en el marco teórico. También para llevar a cabo el estudio y análisis de la información relacionada con sistemas y plataformas educativas, lo que permitió obtener los principales elementos asociados a la mejora de la interacción entre los usuarios y el contenido en la plataforma educativa ZERA 2.0.

- El **método histórico-lógico:** para conocer la evolución del objeto de estudio de la investigación y para concluir qué aspectos son necesarios en el desarrollo de la solución que se propone.

Métodos científicos del nivel empírico

- **El método observación**: se utilizó para identificar buenas prácticas y vulnerabilidades de los sistemas similares, además para obtener información de las necesidades existentes a lo largo del desarrollo de la herramienta. Permite una comparación de los resultados obtenidos por diferentes vías, contribuyendo alcanzar una mayor precisión en la información recogida.

La presente investigación está conformada por tres capítulos, quedando estructurados de la siguiente manera:

Capítulo 1: Fundamentación Teórica.

En este capítulo se relacionan los conceptos y aspectos teóricos que sustentan el desarrollo de la solución propuesta. Se fundamenta además, la selección de la metodología, las tecnologías, herramientas, lenguajes de programación y de modelado utilizados.

Capítulo 2: Diseño.

En este capítulo se documentan los procesos identificados en el sistema, se describe el modelo de dominio, y los conceptos asociados al mismo. Se realiza además, el diseño de la propuesta de solución según los requisitos funcionales y no funcionales obtenidos.

Capítulo 3: Implementación y prueba.

En este capítulo se describen los principales aspectos de la implementación, reflejando el empleo de buenas prácticas de programación y estándares de codificación. Incluye además la estrategia seguida para aplicar las pruebas al sistema.

Capítulo 1 Fundamentación Teórica

En el presente capítulo se abordan de manera sintetizada algunos de los principales conceptos asociados al objeto de estudio con el fin de lograr una mejor comprensión del problema de la investigación, los fundamentos teóricos que la sustentan, el comportamiento de sistemas similares en otras plataformas a nivel mundial. Además, se justifican las metodologías, tecnologías, herramientas y lenguajes usados en el desarrollo de componentes que permitan la gestión de notas, para dar soporte a la solución propuesta, así como una explicación de las razones que llevan a su solución.

1.1 Conceptos asociados al dominio del problema

Para la correcta comprensión del problema y su solución es necesario conocer los principales conceptos y definiciones asociados al dominio del problema.

Trabajo colaborativo: es un método de enseñanza adaptable a cualquier modalidad educativa, que a través del trabajo en grupo logra alcanzar objetivos comunes utilizando diferentes técnicas o estrategias, busca mejorar la adquisición y desarrollo de habilidades cognitivas o intelectuales y habilidades sociales, mediante la interacción, la interdependencia y las relaciones interpersonales que se produzcan entre los grupos, durante y después de la aplicación del método. (5)

Plataforma educativa: es una herramienta física, virtual o una combinación de ambas, que brinda la capacidad de interactuar con uno o varios usuarios con fines pedagógicos. Se considera además, que contribuyen en la evolución de los procesos de aprendizaje y enseñanza, complementando o presentando alternativas a las prácticas de educación tradicional. (6)

Herramientas colaborativas: son utilizadas en la comunicación entre personas, aunque ésta sólo sea de forma virtual. Actualmente existen diversas herramientas de este tipo tales como las redes sociales, wikis, blogs, chats y otros recursos específicos. Estas nuevas herramientas de la comunicación han permitido de forma eficaz y rápida el traspaso de información y han

acortado de una forma u otra las distancias. En el ámbito académico, en particular en educación superior, las herramientas de trabajo colaborativo son un potente recurso que fácilmente se adapta a diversas necesidades y objetivos.(7)

Sistemas de Gestión de Aprendizaje (LMS): es un *software* basado en un servidor web que provee módulos para los procesos administrativos y de seguimiento que se requieren para un sistema de enseñanza, simplificando el control de estas tareas. También facilitan el aprendizaje distribuido y colaborativo a partir de actividades y contenidos pre elaborados, de forma síncrona o asíncrona, utilizando los servicios de comunicación de Internet como el correo, los foros, las videoconferencias o el chat.(1)

Autogestión del aprendizaje: se entiende como el marco en el cual el estudiante es el principal responsable y administrador autónomo de su proceso de aprendizaje, encuentra sus objetivos académicos, gestiona recursos tanto de tipo material como humano, prioriza sus decisiones y tareas en todo el proceso de su circuito de aprendizaje.(8)

1.2 Estudio comparativo de herramientas similares

Los sistemas orientados a la formación cuentan con un espacio donde se habilita a los usuarios el contenido a estudiar, siendo de especial interés pues la mayor parte de las actividades educativas giran en su entorno. Existes varias formas de mostrar este contenido tal es el caso de los LMS que son capaces de similar un libro tradicional, permitiendo un mejor entendimiento del contenido, algunos ejemplos de ellos son Moodle y Claroline.

1.2.1 Ejemplos de soluciones similares en el mundo

MOODLE: es una herramienta de software libre y gratis. Además, se retroalimenta del trabajo realizado por múltiples instituciones y participantes que colaboran en red, lo cual permite acceder libremente e incorporar a nuestra asignatura múltiples módulos y recursos creados por otros usuarios.(9)

Esta plataforma presenta las siguientes funcionalidades:

- permite a los usuarios, tanto a alumnos como a profesores, además de escribir texto como tradicionalmente se hacía en los apuntes o trabajos, incluir o enlazar las más variadas fuentes y recursos 2.0, como múltiples blogs, web- quest, imágenes, videos o documentos, que harán mucho más rico y variado el contenido.

- cuenta con un sistema de anotación que permite realizar marcas y anotaciones en los márgenes de los textos de los foros de Moodle.

Plataforma Educativa Claroline: constituye una plataforma de aprendizaje virtual (eLearning) y de trabajo virtual (eWorking). Claroline está basada en herramientas y lenguajes libres como PHP e integra estándares actuales como SCORM para intercambiar contenidos. Esta plataforma está publicada bajo una licencia Open Source (de código abierto o software libre), permite crear y administrar cursos a cientos de organizaciones de 93 países diferentes y la colaboración de espacios online. (11)

Claroline les permite a los profesores realizar operaciones tales como:

- publicar archivos que puedan ser de interés para otros miembros del curso.

- editar zona de texto fácilmente, añadir imágenes y enlaces, modificar y resaltar textos.

1.2.2 Ejemplos de soluciones similares en Cuba

La Plataforma Educativa ZERA versión 1.0: sistema para la gestión académica, el aprendizaje y la creación de recursos educativos. Se basa en la concepción pedagógica cubana de Hiperentornos de Aprendizaje. Dota a los usuarios de una plataforma flexible e interactiva, donde los contenidos simulan la metáfora de los libros de textos tradicionales. (10)

Esta plataforma presenta las siguientes funcionalidades:

- los usuarios pueden crear apuntes asociados a un texto seleccionado del contenido de una página o a un recurso en la biblioteca.

- permite a los estudiantes realizar resaltado de texto de un contenido específico.

1.2.3 Resumen sobre análisis de las herramientas similares existentes

Las funcionalidades analizadas en cada una de las plataformas estudiadas tienen características similares a las solicitadas en los requisitos del cliente, por esta razón fueron tomados como referencia para el desarrollo de la Plataforma ZERA 2.0. Dentro de las características que sirvieron de apoyo para el desarrollo es que cuentan con un sistema de anotación que pueden realizar marcas y anotaciones en los márgenes de los textos. Algunas características que se detectaron ausentes en estas plataformas fueron: permitir resaltar el texto de varios colores y no de un solo color como lo tienen implementado, además de guardar todos los cambios hecho por el usuario luego que el administrador haga algún cambio en el contenido.

1.3 Metodología de desarrollo de *software*

Una metodología de desarrollo de *software* es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. Comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto de *software* desde que surge la necesidad del producto hasta que se alcanza el objetivo por el cual fue creado. Las metodologías definen con precisión los artefactos, roles y actividades involucradas, junto con prácticas y técnicas recomendadas. (11)

En la actualidad las metodologías de desarrollo de software se dividen en 2 grandes grupos: ágiles y tradicionales.

Metodologías tradicionales: centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto, definido todo esto, en la fase inicial del desarrollo del proyecto. Otra de las características importantes dentro de este enfoque, son los altos costes al implementar un cambio y la falta de flexibilidad en proyectos donde el entorno es volátil. Las metodologías tradicionales se focalizan en la documentación, planificación y procesos (plantillas, técnicas de administración, revisiones, etc. (11) También, imponen una disciplina de trabajo sobre el proceso de desarrollo del *software* con el fin de conseguir un *software* más eficiente.

Capítulo I

Fundamentación Teórica

Metodologías ágiles: nacen como respuesta a los problemas que puedan ocasionar las metodologías tradicionales y se basan en dos aspectos fundamentales, retrasar las decisiones y la planificación adaptativa. Basan su fundamento en la adaptabilidad de los procesos de desarrollo. Estas metodologías ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan.(11) Proporcionan una serie de pautas y principios junto a técnicas pragmáticas que harán la entrega del proyecto menos complicada y más satisfactoria tanto para los clientes como para los equipos de entrega. Entre las principales metodologías de este tipo se encuentran: Programación Extrema *o Extreme Programming* (XP), Proceso Abierto Unificado o (openUP) y Proceso Ágil Unificado o *Agile Unified Process*(AUP).

Las metodologías tradicionales han intentado abordar la mayor cantidad de situaciones de contexto del proyecto, exigiendo un esfuerzo considerable para ser adaptadas, sobre todo en proyectos pequeños y con requisitos muy cambiantes. Las metodologías ágiles ofrecen una solución casi a medida para una gran cantidad de proyectos que tienen estas características.(12) La Tabla 1 recoge esquemáticamente las principales diferencias de las metodologías ágiles con respecto a las tradicionales.

Tabla 1.Diferencias entre las metodologías ágiles y las tradicionales.

Metodologías Ágiles	**Metodologías Tradicionales**
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparadas para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo).	Impuestas externamente.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.

No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Más artefactos.
Pocos roles.	Más roles.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

No existe una metodología universal para hacer frente con éxito a cualquier proyecto de desarrollo de *software*. Toda metodología debe ser adaptada al contexto del proyecto (recursos técnicos y humanos, tiempo de desarrollo y tipo de sistema. (12)

1.3.1 Metodologías AUP y AUP-UCI

Al no existir una metodología de *software* universal, ya que toda metodología debe ser adaptada a las características de cada proyecto. La Universidad de las Ciencias Informáticas decide hacer una variación de la metodología AUP (AUP-UCI), de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la universidad. La adaptación de AUP-UCI logra estandarizar el proceso de desarrollo del *software* dando cumplimiento las buenas prácticas de desarrollo. Se logra hablar un lenguaje común en cuanto a fases, disciplinas, roles y productos de trabajos. A continuación, algunas diferenciaciones que sufre la metodología AUP

- De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que se denominará Ejecución. Además, se agrega la fase de Cierre.

- AUP propone 9 roles se decide para el ciclo de vida de los proyectos de la UCI tener 11 roles, manteniendo algunos de los propuestos por AUP y unificando o agregando otros.

- AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener 8 disciplinas, pero a un nivel más atómico que el definido en AUP.

Fundamentación de la metodología a utilizar

Con el objetivo de lograr estandarizar la documentación almacenada en los centros productivos de la universidad, se decide utilizar para el desarrollo del componente la metodología AUP-UCI, pues esta metodología facilita el trabajo en proyectos de pequeña envergadura y proporciona un ambiente de desarrollo iterativo e incremental. También describe un enfoque simple y fácil de entender para el desarrollo de *software* usando técnicas y conceptos que aún se mantienen vigentes en RUP. Esta metodología aplica técnicas ágiles incluyendo el Desarrollo Dirigido por Pruebas, Modelado Ágil y Gestión del Cambio Ágil .(13) Además, se preocupa por la gestión de riesgo, propone que aquellos elementos con alto riesgo obtengan prioridad en el proceso de desarrollo y sean abordados en etapas tempranas del mismo. De igual forma AUP-UCI es la metodología empleada por los miembros del proyecto al cual va dirigido el presente trabajo de diploma basándose en el escenario 4 ya que se aplica a los proyectos que no modelen negocio.

1.4 Herramientas y tecnologías

Para dar inicio al desarrollo del componente para la gestión de apuntes y el resaltado de texto se realizó previamente un estudio riguroso y detallado de tecnologías, herramientas de modelado, lenguajes de modelado, *frameworks*, servidores web, gestores de bases de datos y lenguajes de desarrollo para escoger aquellas que se adapten mejor a las características con las que cuenta el proyecto.

1.4.1 Sistema Gestor de Base de Datos

Un **Sistema Gestor de Bases de datos (SGBD)** es un sistema de *software* que accede a los datos y se encarga de gestionarlos. En estos sistemas se proporciona un conjunto coordinado de programas, procedimientos y lenguajes, que permiten a los usuarios realizar las tareas habituales con los datos, es el encargado de la seguridad y se puede centrar en ella de forma independiente a las aplicaciones.(14)

Fundamentación sobre el SGBD

En el desarrollo de aplicaciones web son empleados varios Sistemas Gestores de Bases de Datos como Microsoft SQL Server, MySQL y PostgreSQL.

PostgreSQL

Sistema Gestor de Bases de Datos de código abierto, desarrollado por PGDG (*PostgreSQL Global Development Group*). Posibilita el Control de Concurrencia mediante Versiones Múltiples (*MVCC*, por sus siglas en inglés) que permite trabajar con grandes volúmenes de datos; soporta gran parte de la sintaxis SQL y cuenta con un extenso grupo de enlaces con lenguajes de programación. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.(15)

Principales aspectos

- Diseñado para ambientes de alto volumen de información, PostgreSQL usa una estrategia de almacenamiento de filas llamada Control de Concurrencia para Múltiples Versiones (MVCC) para conseguir una o mejor respuesta en ambientes de grandes volúmenes, y funciona muy bien con una alta concurrencia de usuarios accediendo a la vez al sistema.

- Es multiplataforma y tiene la capacidad de replicación de datos, además de integridad transaccional, herencia de tablas y consulta sobre índices.

- Utiliza nativamente el lenguaje procedural PL/PGSQL para implementar funciones, además de soportar otros tales como: C, C++, Java PL, Java Web, PL/PHP, PL/Python.

Para el desarrollo del trabajo de diploma se decidió el uso del gestor PostgreSQL en su versión 9.4 por las características descritas anteriormente, siendo fácil de administrar, permite además seguridad y autentificación. Es un sistema de base de datos objeto-relacional libre, de código abierto y gratis. También presenta una amplia y activa comunidad de usuarios y documentación. El gestor de base de datos PostgreSQL es utilizado en el proyecto al cual va dirigido el presente trabajo de diploma y es objeto de estudio en la asignatura de Bases de Datos lo que le permite al equipo de desarrollo poseer conocimiento y dominio del uso de dicha herramienta.

1.4.2 Lenguaje de modelado

El lenguaje de modelado es la notación (principalmente gráfica) que usan los métodos para expresar un diseño.(16) Además de poseer un conjunto de vistas, diagramas, símbolos y mecanismos generales que ayudan a la confección de los diagramas a implementar en el sistema. Entre los lenguajes se destacan BPM y UML.

Fundamentación del lenguaje de modelado seleccionado

UML: (*Unified Modeling Languaje* o Lenguaje de Modelado Unificado) es la sucesión de una serie de métodos de análisis y diseño orientadas a objetos, permite visualizar, especificar, construir y documentar un sistema. Ofrece un estándar para describir el modelo, incluyendo aspectos conceptuales tales como procesos, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados. Posee notaciones estandarizadas que permiten construir numerosos tipos de diagramas y artefactos tanto en el modelo de análisis y diseño del sistema. Simplifica en gran medida la representación de los conceptos facilitando su uso y la comunicación entre los involucrados del proyecto.

Para la modelación de la solución se propone el uso de UML en su versión 2.0 por sus características ya que es el lenguaje usado para la modelación de los procesos de negocio utilizado en el proyecto al cual va dirigido el presente trabajo de diploma.

1.4.3 Herramienta CASE

Para el uso de un lenguaje de modelado existen varias herramientas CASE (Ingeniería de Software Asistida por Computadoras), estas son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas. Su naturaleza consiste en conseguir una mejora en la productividad y en la calidad del producto final, que intenta proporcionar ayuda automatizada a las actividades del proceso de desarrollo de software.(17)

Análisis de la herramienta CASE Visual Paradigm

Visual Paradigm: herramienta CASE que propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Esta herramienta ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas.(18)

Fundamentación de la herramienta CASE seleccionada

A partir del estudio realizado se decide que para el desarrollo de la solución se debe emplear Visual Paradigm en su versión 8.0 debido a que es multiplataforma, tiene un diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad. Permite editar especificaciones de un elemento del modelo o de un diagrama y navegar entre elementos del modelado. También se puede crear diagramas asociados a elementos del modelo. Ofrece navegación intuitiva entre la escritura del código y su visualización, potente generador de informes en formato PDF/HTML y también la UCI tiene licencia para su uso. Posee capacidades de ingeniería directa e inversa. A su vez es la herramienta CASE que es el objeto de estudio de la asignatura de Ingeniería de *Software* impartida en la carrera, por lo que los miembros del equipo de desarrollo tienen experiencia en su manejo y utilización.

1.4.4 Lenguajes de desarrollo

Lenguaje de programación: es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina.(19)

Fundamentación de los lenguajes a utilizar

Existe una gran variedad de lenguajes divididos en dos grupos, lenguajes del lado del cliente y lenguajes del lado del servidor para el desarrollo de aplicaciones web.

Lenguajes del lado del cliente

Los lenguajes del lado cliente se usan para su integración en páginas web. Un código escrito en un lenguaje de script se incorpora directamente dentro de un código HTML y se ejecuta interpretado, no compilado son totalmente independientes del servidor. Son muy utilizados en la capa de presentación o la vista de los usuarios. (20)

Los lenguajes utilizados en este caso son el conocido HTML, JavaScript y las hojas de estilos CSS para lograr que la herramienta tenga un aspecto uniforme.

HTML v5: usa un lenguaje de etiquetas para construir páginas web, es extremadamente flexible en cuanto a la estructura y elementos utilizados para construirla. Provee básicamente tres características: estructura, estilo y funcionalidad, además de ser versátil y sus documentos encontrarse estrictamente organizados.(21)

CSS v3: lenguaje de hojas de estilo, del tipo HTML o XML. Aplicables a cualquier navegador, admiten un mayor control sobre los distintos elementos de una página, permitiendo definir el estilo de las fuentes, el color, el espaciado del texto, la posición del contenido, e incluso variaciones en el sonido en los elementos auditivos. Estos estilos pueden definirse para luego ser aplicados al código de cualquier documento.(22)

JavaScript: es el lenguaje interpretado más utilizado, principalmente en la construcción de páginas Web, con una sintaxis muy semejante a Java y a C. Pero, al contrario que Java, no se trata de un lenguaje orientado a objetos propiamente dicho, sino que éste está basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad.(23)

Twig: es un motor de plantillas para el lenguaje de programación PHP se ocupa de brindar una solución al tratamiento de las cuestiones visuales alrededor de una aplicación desarrollada en este lenguaje. Entre sus principales características se destaca ser flexible, seguro y rápido además de ser multiplataforma.(24)

Lenguajes del lado del servidor

Los lenguajes de programación del lado del servidor son ejecutados por el servidor y lo que se envía al cliente es la respuesta o resultado de dicha ejecución. Al cliente solo se le transfiere el resultado de la ejecución del programa. El código fuente permanece en el servidor, se conserva su privacidad y los clientes no tienen acceso a él. (25)

Para el desarrollo de la solución se utilizará PHP como lenguaje de programación del lado del servidor ya que es el utilizado en el proyecto al cual va dirigido el presente trabajo de diploma.

PHP v7.0: es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. La característica más significativa de PHP es su soporte para una gran cantidad de bases de datos. Adicionalmente, soporta ODBC (Estándar Abierto de Conexión con Bases de Datos), así que puede conectarse a cualquier base de datos que soporte tal estándar. Puede trabajar en varios sistemas operativos, soportando la mayoría de los servidores web. PHP también cuenta con soporte para comunicarse con otros servicios.(26)

1.4.5 Frameworks de desarrollo

Framework o Marco de Trabajo es una estructura compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un marco de trabajo

se puede considerar como una aplicación genérica incompleta y configurable a la que pueden añadirse las últimas piezas para construir una aplicación concreta. Los objetivos principales que persigue son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones. En general se puede definir como un conjunto de componentes que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas web. (27)

Fundamentación de los *frameworks* a utilizar

Entre los *framework* más empleados se encuentran JQuery, Bootstrap y Symfony. Para el desarrollo de esta aplicación serán utilizados JQuery v2.0 para el trabajo con el lenguaje JavaScript, Bootstrap v3.0 para el trabajo con CSS y como framework PHP Symfony 2.7.16. La elección de estos marcos de trabajo viene dada por sus características y porque son los utilizados en el proyecto al cual va dirigido el presente trabajo de diploma.

Symfony v2.7.16

Symfony es un proyecto PHP de software libre que permite crear aplicaciones y sitios web rápidos y seguros de forma profesional. Es un completo *framework* basado en el patrón Modelo Vista Controlador y que contiene un enorme conjunto de herramientas y utilidades. Ello permite la creación rápida de una aplicación web, sin descuidar ningún aspecto en la calidad del software además de ser multiplataforma. (28)

Otras características de Symfony son:

- su código, y el de todos los componentes y librerías que incluye, se publican bajo la licencia de software libre.
- la documentación del proyecto también es libre e incluye varios libros y decenas de tutoriales específicos.
- independiente del sistema gestor de bases de datos y sencillo de usar en la mayoría de los casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.

JQuery v 2.0

jQuery es un marco de trabajo para el lenguaje JavaScript puesto que implementa una serie de clases que permiten programar sin preocupación, ya que funcionan de exacta forma en todas las plataformas más habituales. Ofrece una infraestructura con la que se tiene mayor facilidad para la creación de aplicaciones complejas del lado del cliente y las obtenemos de manera gratuita, ya que el framework tiene licencia para uso en cualquier tipo de plataforma, personal o comercial.(29)

Entre sus características se destacan:

- es un producto serio, estable, bien documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización del framework.
- permite cambiar el contenido de una página web sin necesidad de recargarla mediante la manipulación del árbol DOM[6] y peticiones AJAX[7].
- utilidades varias como obtener información del navegador, operar con objetos, vectores y funciones para rutinas comunes.
- Interactividad y modificaciones del árbol DOM, incluyendo soporte para CSS 1-3.

Bootstrap v3.0

Bootstrap es un marco de trabajo HTML, CSS y JS más intuitivo, potente y popular para desarrollar proyectos móviles de primera respuesta en la web. Bootstrap hace que el desarrollo web sea más rápido y más fácil. Está hecho para gente de todos los niveles de habilidad, dispositivos de todas las formas y proyectos de todos los tamaños. Reduce de forma fácil y

[6] DOM (*Document Object Model* o Modelo de Objetos del Documento): es esencialmente una interfaz de plataforma que proporciona un conjunto estándar de objetos para representar documentos HTML, XHTML y XML.

[7] *AJAX* (*JavaScript asíncrono y XML*): permite que el cliente se comunique con el servidor a través de la solicitud, que es realizado en segundo plano sin recargar la página uniendo de manera efectiva la tecnología del cliente con la del servidor. (30)

eficiente los sitios web y aplicaciones con una sola base de código, desde teléfonos a tabletas, a escritorios con consultas de medios CSS, además de ser de código abierto.(31)

1.4.6 Sistemas de Mapeo Objeto Relacional

El **Mapeo Objeto-Relacional** (*Object-Relational Mapping*, o sus siglas ORM): es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional.(32)

Doctrine: es una herramienta que funciona como mapeador de objetos relacionales, para PHP (*Hypertext Preprocessor)* permitiendo obtener los datos solicitados de una forma eficaz ya que brinda varias alternativas de consultas a la base de datos ya sea de forma automática o escribiendo el dato a consultar.(32)

A continuación, las principales características:

- en su implementación contiene un lenguaje SQL propio del software llamado DQL (Doctrine Query Languaje) el cual entre sus funciones permite obtener objetos en sus consultas.
- soporta diferentes tipos de herencia que permitirán optimizar cada una de las operaciones.
- el sistema de archivo utilizado es YAML de igual forma se puede programar en código PHP.
- permite crear manualmente y automáticamente el modelo de base de datos a implementar.

1.6.7 Entorno de Desarrollo Integrado

Un **Entorno de Desarrollo Integrado** o **Entorno de Desarrollo Interactivo**, en inglés *Integrated Development Environment* (**IDE**), es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software. Un IDE consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador. (33)

Fundamentación del IDE a utilizar:

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. Es un producto libre y gratuito sin restricciones de uso. Es compatible con Sistemas Operativos *Windows, Linux* y Mac OS X.(34)

Para el desarrollo de esta solución se hará uso del IDE NetBeans v8.0 debido a que es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero ofrece soporte superior para desarrolladores de C / C ++ y PHP, proporcionando editores y herramientas completos para sus marcos y tecnologías relacionados. Además, el IDE tiene editores y herramientas para XML, HTML, PHP, Groovy, Javadoc, JavaScript y JSP, además viene con soporte para Symfony un gran *framework* MVC escrito en PHP, también cuenta con soporte para AJAX. (34) Es importante destacar que la elección de este entorno de desarrollo está sustentada fundamentalmente en que este es utilizando en el proyecto al cual va dirigida la presente investigación, además que los miembros del equipo de desarrollo tienen experiencia con el trabajo con el IDE.

1.4.8 Servidor Web

Servidor web o servidor HTTP[8]: es un programa informático capaz de ejecutar programas del lado del servidor, así como, establecer conexiones bidireccionales con cualquier cliente que se conecte a dicho servidor, generando respuestas dinámicas.(25)

Fundamentación del servidor web a utilizar

Existes varios servidores web, pero Nginx es uno de los más utilizados en dominios activos.

Servidor Web Nginx

Es un servidor web HTTP de código abierto que también incluye servicios de correo electrónico con acceso al *Internet Message Protocol* (IMAP) y al servidor *Post Office Protocol* (POP). Se trata de un servidor web/proxy completamente inverso que tiene como principal característica

[8] *Hypertext Transfer Protocol*,en español Protocolo de Transferencia de Hipertexto.

ser sumamente ligero, lo que lleva a su otro gran atractivo, su velocidad, lo que permite servir aplicaciones web con una velocidad muy superior a la de sus competidores más directos. Se puede decir entonces, que es un servidor web de alto rendimiento, ideal para realizar todo tipo de trabajos, ya sean profesionales o aficionados.(35)

Para el desarrollo de la solución planteada se decide utilizar el servidor web Nginx v 1.10 debido a que es multiplataforma y fácil de instalar, además de ser muy ligero. Es compatible con las aplicaciones web más populares. Además, es el servidor utilizado por los miembros del equipo de trabajo al cual va dirigido el presente trabajo de diploma.

1.5 Cliente de Base de Datos

Para administrar las instancias de PostgreSQL se hará uso de pgAdmin III debido a que es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia *Open Source.* Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita enormemente la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados.(36)

1.6 Patrones de arquitectura

Antes de hablar de patrones primero se debe centrar el análisis en el concepto de arquitectura de software, a partir de este se tendrá una visión más clara, para llegar a la definición de patrón de arquitectura y cómo emplearla en el desarrollo de la solución.

La arquitectura de software es un nivel de diseño que hace foco en aspectos más allá de los algoritmos y estructuras de datos de la computación; el diseño y especificación de la estructura global del sistema es un nuevo tipo de problema.(37) Además es un conjunto de patrones que proporcionan un marco de referencia necesario para guiar la construcción de un software, permitiendo a los programadores, analistas y todo el conjunto de desarrolladores del software compartir una misma línea de trabajo y cubrir todos los objetivos y restricciones de la aplicación.

Es considerada el nivel más alto en el diseño de la arquitectura de un sistema puesto que establecen la estructura, funcionamiento e interacción entre las partes del software.(38)

Patrón de arquitectura

Descripción de un problema particular y recurrente de diseño, que aparece en contextos de diseño específico y presenta un esquema genérico demostrado con éxito para su solución. El esquema de solución se especifica mediante la descripción de los componentes que la constituyen, sus responsabilidades y desarrollos, así como también la forma de cómo estos colaboran entre sí.(39)

Algunos ejemplos de patrones de arquitecturas son la programación por capas, arquitectura en pizarra, arquitectura dirigida por eventos, arquitectura orientada a servicios y Modelo Vista Controlador

Fundamentación del patrón seleccionado

Para el desarrollo de la solución planteada se seleccionó el patrón Modelo Vista Controlador (MVC) que es el utilizado por el framework Symfony seleccionado para el desarrollo de las funcionalidades. Además, la elección se fundamenta en lo siguiente:

El patrón MVC es un paradigma que divide las partes que conforman una aplicación en el Modelo, las Vistas y los Controladores, permitiendo la implementación por separado de cada elemento, garantizando así la actualización y mantenimiento del software de forma sencilla y en un reducido espacio de tiempo. A partir del uso de *frameworks* basados en el patrón MVC se puede lograr una mejor organización del trabajo y mayor especialización de los desarrolladores y diseñadores.(40)

Modelo: es la capa donde se trabaja con los datos, por tanto, contendrá mecanismos para acceder a la información y también para actualizar su estado. Los datos se tendrán habitualmente en una base de datos, por lo que en los modelos tendrán todas las funciones que accederán a las tablas.(41)

Vista: como su nombre hace entender, contienen el código de nuestra aplicación que va a producir la visualización de las interfaces de usuario, o sea, el código permitirá renderizar los estados de nuestra aplicación en HTML. En las vistas nada más se tiene los códigos HTML y PHP que permite mostrar la salida**.** En la vista generalmente se trabaja con los datos, sin embargo, no se realiza un acceso directo a éstos. Las vistas requerirán los datos a los modelos y ellas se generará la salida, tal como nuestra aplicación requiere.(41)

Controlador: contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento, realizar una compra, una búsqueda de información, etc. En realidad, es una capa que sirve de enlace entre las vistas y los modelos, respondiendo a los mecanismos que puedan requerirse para implementar las necesidades de nuestra aplicación. Sin embargo, su responsabilidad no es manipular directamente datos, ni mostrar ningún tipo de salida, sino servir de enlace entre los modelos y las vistas para implementar las diversas necesidades del desarrollo.(41)

El Controlador y el Modelo juntos residen del lado del servidor.

Entre sus características se destacan que es un patrón de diseño de *software* verdaderamente probado, que convierte una aplicación en un paquete modular fácil de mantener y mejora la rapidez del desarrollo. La separación de las tareas de la aplicación en modelos, vistas y controladores hace que la aplicación sea además, muy ligera de entender.

1.6 Conclusiones del capítulo

Con la realización de este capítulo se dieron a conocer un conjunto de elementos que marcaron el punto de partida para el desarrollo de un componente para la gestión de apuntes y el resaltado de texto de la Plataforma Educativa ZERA 2.0. Se realizó un estudio del estado del arte de los sistemas similares delimitando sus ventajas y desventajas. Se define la metodología de desarrollo de software AUP. Además, se identificaron los lenguajes herramientas y tecnologías HTML v5, CCS v3, JavaScript, AJAX y PHP v7.0; los frameworks Symfony 2.7.16, Bootstrap 3.0 y JQuery 2.0, y el ambiente de trabajo Netbeans v8.0 a utilizar.

Capítulo 2 Diseño

Para lograr un mejor entendimiento del componente que se desea desarrollar, por parte de los clientes y desarrolladores, es de vital importancia definir bien los procesos que intervienen en este. Se tiene como propósito exponer el contexto en que se desarrolla el problema que da origen a esta investigación. Mediante el modelo de dominio se describe la estructura que se desea representar en la herramienta a desarrollar. Describiendo los principales conceptos del entorno que serán objeto de análisis para la realización de la fase de Diseño. Se identifican los requisitos funcionales, no funcionales, las historias de usuario que incluyen a su vez los prototipos de interfaz de usuario, el modelo de dominio. Además, se definen los patrones de diseños, así como el modelo de entidad relación de la base de datos y el diagrama de despliegue.

2.1 Modelo de dominio

Un modelo del dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. Se utiliza con frecuencia como fuente de inspiración para el diseño de los objetos software.(42) El Modelo de Dominio ayuda a comprender los conceptos que utilizan los usuarios y están relacionados con los procesos que se desean desarrollar, por esta razón se hace necesaria su elaboración.

2.1.1 Conceptos del dominio

- **Plataforma Educativa ZERA 2.0**: plataforma educativa destinada a apoyar el proceso de enseñanza – aprendizaje
- **Estudiante**: persona que cursa estudios, tiene el propósito de aprender una materia y accede al hiperentorno en busca de conocimientos.
- **Docente:** persona responsable de educar, guiar y supervisar a los estudiantes en el proceso de enseñanza – aprendizaje.

- **Usuario:** representa la generalización de los usuarios Docente, Estudiante.
- **Apuntes:** proceso por el cual el usuario almacena información importante, acerca del contenido.
- **Resaltado:** proceso por el cual el usuario almacena y resalta información importante, acerca del contenido.
- **Recurso:** objeto o recurso que forma parte del contenido. Un recurso puede estar asociado a un apunte, y el apunte puede estar asociado a un recurso.
- **Página de contenido:** tiene como principal objetivo permitir a los docentes y estudiantes la navegación por cada uno de los contenidos de las materias a través de las páginas.

2.1.2 Diagrama del modelo de dominio

El presente modelo de dominio tiene como objetivo identificar las relaciones entre todas las entidades comprendidas en el ámbito del dominio del problema.

CAPÍTULO 2

DISEÑO

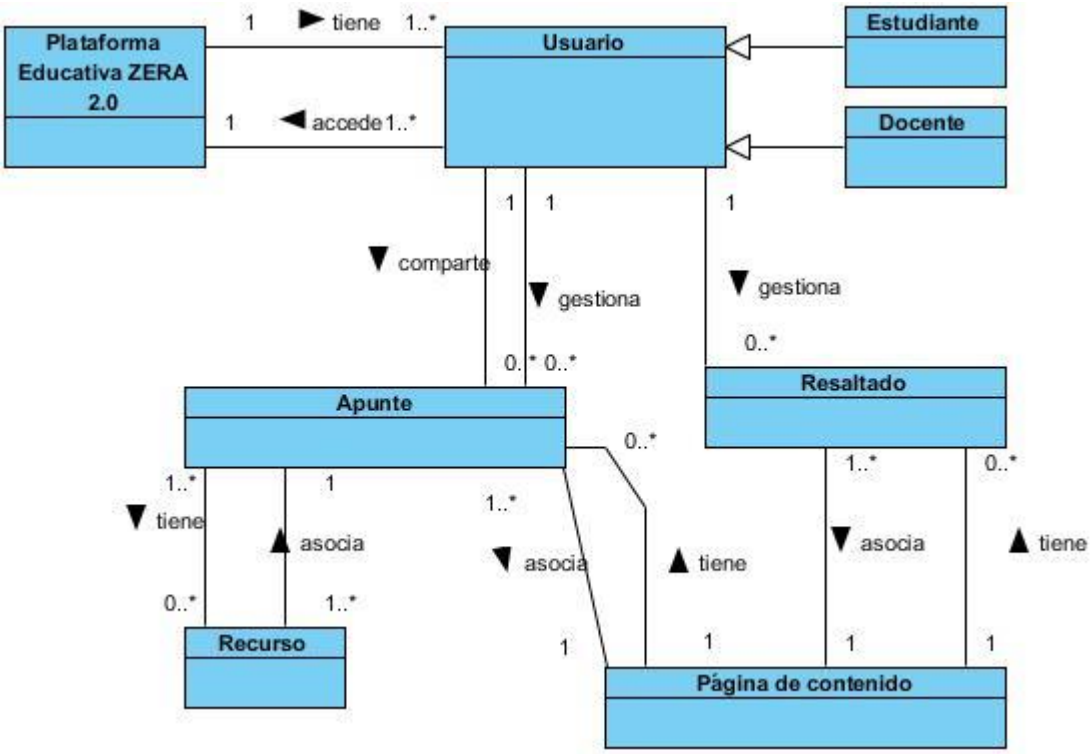


Imagen 1. Modelo del Dominio.

2.2 Propuesta de solución

El desarrollo del componente propuesto tiene como principal objetivo permitir a los docentes y estudiantes elevar las opciones de autogestión del aprendizaje. El componente a desarrollar proporcionará a los usuarios de la plataforma un grupo de funcionalidades que permitirán la interacción entre el usuario y el contenido. Entre las que se destacan: llevar a cabo un sistema de apuntes que se puedan compartir con otros usuarios además de asociarle un recurso, resaltar información que sea de importancia, lo cual traerá consigo un mejor entendimiento del contenido por parte del usuario.

2.3 Requerimientos del sistema

Los requerimientos del sistema son las condiciones o capacidades que debe cumplir el sistema para satisfacer el contrato establecido donde se debe mostrar todo lo que el sistema debe

hacer. Deben ser suficientemente entendibles como para que pueda llegarse a un acuerdo entre el cliente y los desarrolladores sobre qué debe y qué no debe hacer el sistema.

2.3.1 Requisitos funcionales

Los requisitos funcionales (RF) pueden ser una descripción de lo que un sistema debe hacer. Este tipo de requisito especifica algo que el sistema entregado debe ser capaz de realizar. Además son declaraciones de los servicios que proveerá el sistema y de qué manera éste reaccionará en situaciones particulares.(43)

RF1 Crear apuntes: permite crear un apunte tanto al estudiante como al docente.

RF2 Eliminar apunte: eliminar un apunte del contenido. Esta operación puede ser realizada tanto por el estudiante como por el docente.

RF3 Editar apunte: permite modificar los datos de un apunte.

RF4 Listar apuntes: visualiza un listado de apuntes, mostrando una síntesis del contenido del mismo.

RF5 Compartir apunte: permite compartir los apuntes de un usuario con otros en el mismo curso.

RF6 Asociar un recurso al apunte: asocia un recurso al apunte si el usuario es un profesor.

RF7 Ver detalles del apunte: muestra los detalles de un apunte en específico.

RF8 Resaltar textos en el contenido: permite al usuario resaltar el texto seleccionado.

RF9 Eliminar resaltado de textos: elimina un resaltado del sistema.

RF10 Modificar el resaltado de los textos: modifica el texto que con anterioridad ya fue resaltado.

RF11 Cambiar color de resaltado: cambia el color del resaltado según desee el usuario.

2.3.2 Requisitos no funcionales

Los requisitos no funcionales especifican algo sobre el propio sistema, y cómo debe realizar sus funciones. Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo, estándares, etc.(43)

Software:

- Sistema operativo: Distribución de CentOS última versión estable.
- Estos deberán contar con un navegador web moderno (Navegadores web: Firefox (v45.x en adelante), Chrome (v46.x en adelante), Safari (v10.x en adelante), navegadores de dispositivos móviles actualizados para acceder a la aplicación.
- Servidor de bases de datos relacional PostgreSQL 9.4.x
- Servidor de aplicaciones Ngnix: 1.10.x

Seguridad:

- Garantizar la protección de información de accesos no autorizados.
- Garantizar el acceso a las funcionalidades definidas para los usuarios de acuerdo a los roles que posean.

2.4 Historias de usuario

Las historias de usuario (HU) son técnicas utilizadas para especificar los requisitos del *software.* Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.(44)

Además, las historias de usuarios se centran en el valor que viene de usar el sistema en lugar de una especificación detallada de lo que el sistema debe hacer. Están concebidos como un medio para fomentar la colaboración.

A continuación, se muestra la historia de usuario del RF Crear apuntes. Las restantes se describen en el anexo 2.

Tabla 2. Historia de Usuario del RF Crear apuntes.

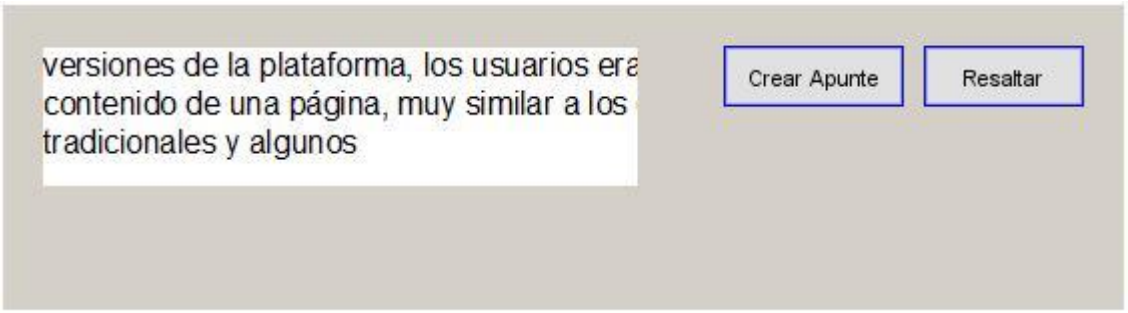
Número: 1	**Nombre del requisito:** Crear apuntes
Programador: Lisy Maday López Lugo	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 5 días
Riesgo en Desarrollo: Alta	**Tiempo Real:** 4 días
Descripción: Permitir crear nuevos apuntes asociados a un curso. **2- Acciones para lograr el objetivo (precondiciones y datos):** - Para crear un apunte, este debe estar asociado a una parte del contenido, se selecciona previamente la opción de crear apunte y seguidamente el texto al que pertenecerá el apunte. - Estar autenticado en el sistema con el rol estudiante o profesor **3- Comportamientos válidos y no válidos (flujo central y alternos):**	

4- Flujo de la acción a realizar:

- En caso de que el usuario seleccione la opción de Crear un apunte, el sistema dará la posibilidad de seleccionar el texto sobre cual se le realizara dicho apunte. Cuando el usuario selecciona correctamente el texto se creará un nuevo apunte y el sistema.

Observaciones

Prototipo de interfaz:



2.5 Patrones de diseño

Los patrones de diseño son soluciones para problemas típicos y recurrentes que se pueden encontrar a la hora de desarrollar una aplicación. Además que expresan esquemas para definir estructuras de diseño o sus relaciones.(45)

El framework Symfony utiliza en su implementación una serie de patrones de diseño, los cuales clasifican y describen formas de solucionar problemas específicos y comunes del diseño orientado a objetos.

2.5.1 Patrones GRASP

Los GRASP son patrones generales de *software* para asignación de responsabilidades *(General Responsibility Asignment Software Patterns,* por sus siglas en inglés*).* Como su nombre lo indica, estos patrones indican cual es la manera de asignar responsabilidades a objetos software. A continuación, se explicarán algunos de los patrones utilizados directamente en la solución.

Alta Cohesión: caracteriza a las clases con responsabilidades estrechamente relacionadas y asigna responsabilidades procurando que la cohesión sea lo más alta posible. (46) Se evidencia en las clases ***NotesManager* , *HighlightManager y NoteUtils*.**

Controlador: El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado.(46)Se ve evidenciado en las clases controladoras. ***NotesController HighligthContoller***

2.5.2 Patrones GOF

Singleton: el objetivo de este patrón es asegurarse de que de una clase solo existe una instancia y que esta es accesible, o, mejor dicho, ofrecer un punto de acceso a ella. Existen muchas clases para las cuales es importante tener una sola instancia que pueda ser utilizada en muchas partes diferentes del sistema . Algunos de los beneficios que aporta el patrón son: poder controlar el acceso a la instancia. Reduce el espacio de nombres ya que evita contaminarlo con variables globales. Permite refinar operaciones y la representación a través de la creación de subclases. Permite controlar fácilmente y sin apenas cambios el número de instancias que creamos.(47) Este patrón se ve evidenciado en las clases ***NotesManager, HighlightManager*** *y* ***NoteUtils.***

Factoría(*Factory*): Su propósito es definir una interface para crear objetos donde se delega la creación de las instancias de las subclases. Se encarga de centralizar el sitio donde se crean los objetos, normalmente donde se crean objetos de una misma familia.(47) Este patrón se evidencia en los formularios.

CAPÍTULO 2

DISEÑO

2.6 Modelo de diseño

El modelo de diseño expande y detalla los modelos de análisis tomando en cuenta todas las implicaciones y restricciones técnicas. El propósito del diseño es especificar una solución que trabaje y pueda ser fácilmente convertida en código fuente y construir una arquitectura simple y fácilmente extensible.(48)

2.6.1 Diagramas de clases del diseño

Los diagramas de clases del diseño(DCD) muestran también las asociaciones entre las clases y los atributos entre ellas, además de mostrar las definiciones de las clases de *software* en lugar de los conceptos del mundo real.(49)

A continuación, se muestra el DCD de la historia de usuario Crear apuntes. Para ver el resto de los diagramas DCD remitirse al anexo 4.

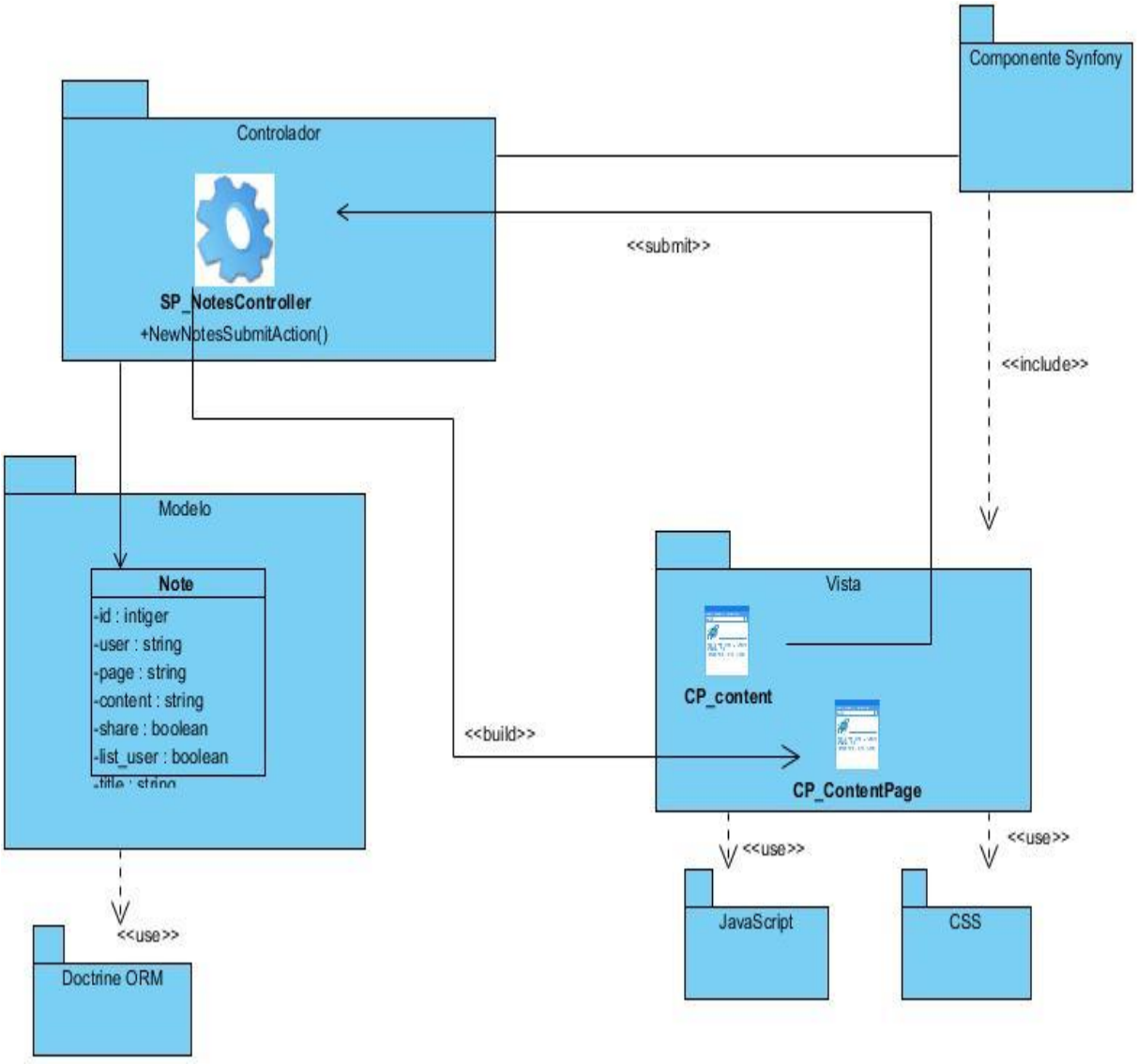


Imagen 2. DCD_HU_Crear Apuntes.

2.6.2 Diagramas de secuencia del diseño

Un diagrama de secuencias del diseño (DSD) muestra la interacción de un conjunto de objetos de una aplicación a través del tiempo, en el cual se indicarán los módulos o clases que formarán parte del programa y las llamadas que se hacen cada uno de ellos para realizar una

tarea determinada, por esta razón permite observar la perspectiva cronológica de las interacciones.(50)

A continuación, se muestra el DSD de la historia de usuario Crear apuntes. Para ver el resto de los diagramas DSD remitirse al anexo 5.

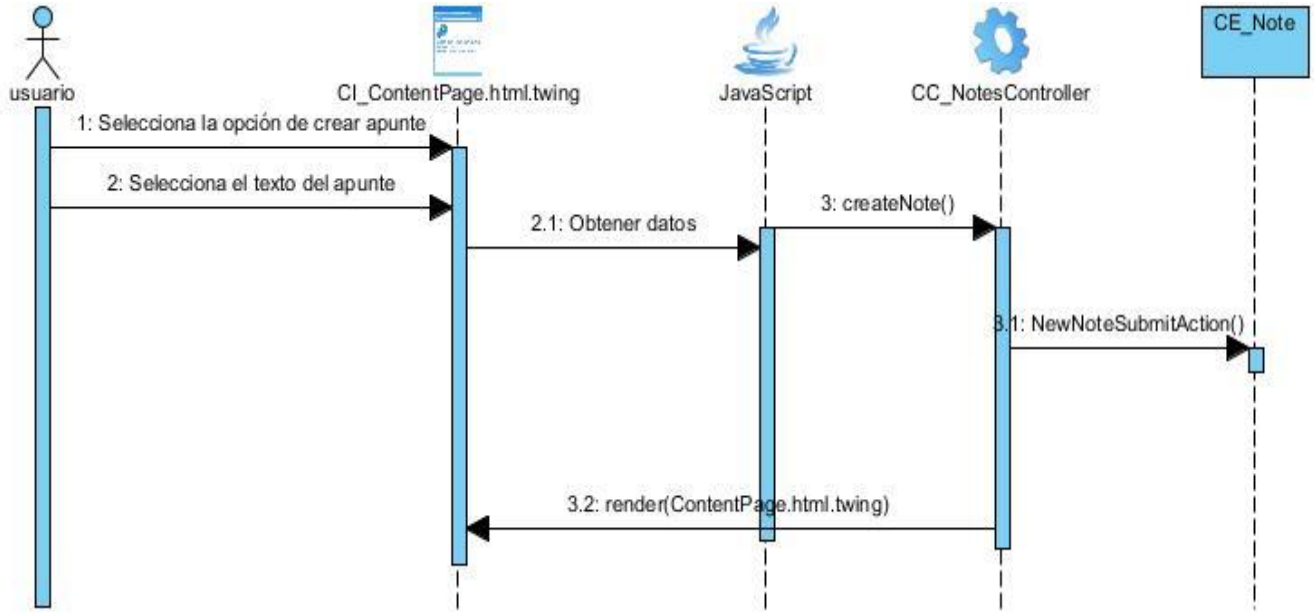


Fig 3. DSD_HU_CrearApuntes.

2.6.3 Diagrama de despliegue

El Diagrama de despliegue es un diagrama estructurado que muestra la arquitectura del sistema desde el punto de vista del despliegue de los artefactos del software en los destinos de distribución.(51)

CAPÍTULO 2

DISEÑO

Imagen 4.Diagrama de despliegue del sistema.

PC Cliente: PC desde donde se podrá visualizar e interactuar con la plataforma a través de un navegador web (previamente instalado en la máquina).

Servidor Web: PC servidor donde se encuentra desplegado el componente, al que se conectan los clientes por medio de sus estaciones de trabajo.

Servidor de BD: Servidor PostgreSQL en el que se encuentran almacenados todos los datos persistentes del sistema.

Servidor Media: Servidor encargado de almacenar todos los recursos de la plataforma.

Los protocolos utilizados son:

- **HTTP:** Protocolo de comunicación estándar-básico que se utiliza en las arquitecturas web. Se utiliza para la comunicación establecida entre el servidor web y las estaciones de los usuarios.
- **TCP/IP:** Protocolo mediante el cual se realiza la comunicación entre el servidor web y el servidor de bases de datos.
- **HTTPs:** Es un protocolo de aplicación basado en el protocolo HTTP, destinado a la transferencia segura de datos de Hipertexto, es decir, es la versión segura de HTTP.

2.6.4 Modelo de datos

El modelo de datos es una serie de conceptos que se utiliza para describir un conjunto de datos y las operaciones para manipularlos. Está compuesto por las entidades que conformarán las tablas de la base de datos que serán utilizadas por las funcionalidades a implementar.(52)

Descripción de las tablas creadas

PkT_resource. Tb_highlight: Es la tabla que almacena los datos de los resaltados realizadas por el usuario.

PkT_resource. Tb_notes: Es la tabla que almacena los datos de las notas realizadas por el usuario.

PKT_resource. Tb_r_sharenote: Es la tabla que posee la relación de las notas que pueden ser compartidas por los usuarios.

PKT_resource. Tb_r_resource_note: Es la tabla que posee la relación de las notas con los recursos.

PKT_resource. Tb_notebackup: Es la tabla que almacena los datos de los cambios que hace el administrador sobre una página de contenido.

Las tablas que no se especifican anteriormente son las tablas que son utilizadas por el sistema, pero de manera heredada.

A continuación, se muestra la relación de las tablas de la base de datos que serán utilizadas por el componente a implementar.

CAPÍTULO 2

DISEÑO

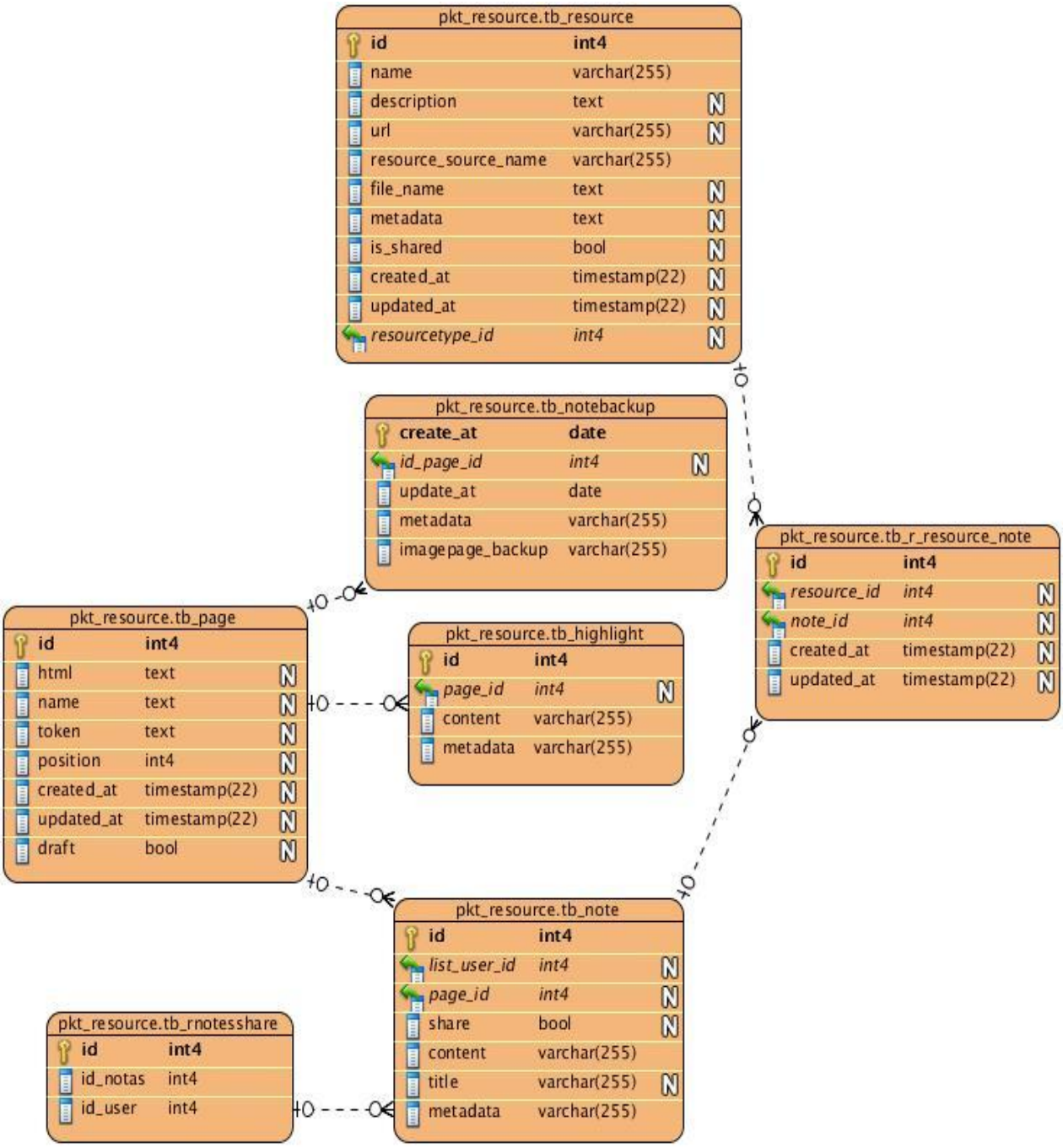


Imagen. 5. Modelo Entidad Relación.

2.7 Conclusiones del capítulo

El estudio realizado a los principales conceptos del negocio permite comenzar a desarrollar la propuesta de solución del componente a partir de determinar los artefactos necesarios para iniciar el diseño de la propuesta de solución. En este capítulo quedó definido el diseño del sistema, así como sus relaciones, describiendo las mismas en términos de diagramas de clase, enfocados al cumplimiento de los RF del sistema. Se realizó, además, el diseño de la base de datos, sentando las bases para la implementación del componente propuesto.

Capítulo 3 Implementación y Prueba

En este capítulo se documentó todo lo referente al flujo de trabajo de implementación. El cual tiene como propósito realizar el diagrama de componentes que describe los elementos físicos del módulo y sus relaciones, además de probar el componente desarrollado e integrarlo a un sistema ejecutable. Para poner a prueba el componente desarrollado se realizarán un conjunto de pruebas incluyendo además los resultados de las mismas, así como las validaciones realizadas al componente, para lo cual se hace necesario que el producto de software funcione como está previsto.

3.1 Modelo de implementación

En la implementación se parte del resultado del diseño y se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares. El propósito principal de la implementación es desarrollar la arquitectura y el sistema como un todo.

3.1.1 Diagrama de componentes

Los diagramas de componentes se usan para estructurar el modelo de implementación y describen los elementos físicos del módulo y sus relaciones. Organiza los subsistemas de implementación en capas y se utilizan para modelar la vista estática de un sistema. Muestran además, la organización y las dependencias lógicas entre un conjunto de componentes de software, sean estos componentes de código fuente, librerías, binarios o ejecutables.(53)

A continuación se presenta el diagrama de componentes propuesto para la solución:

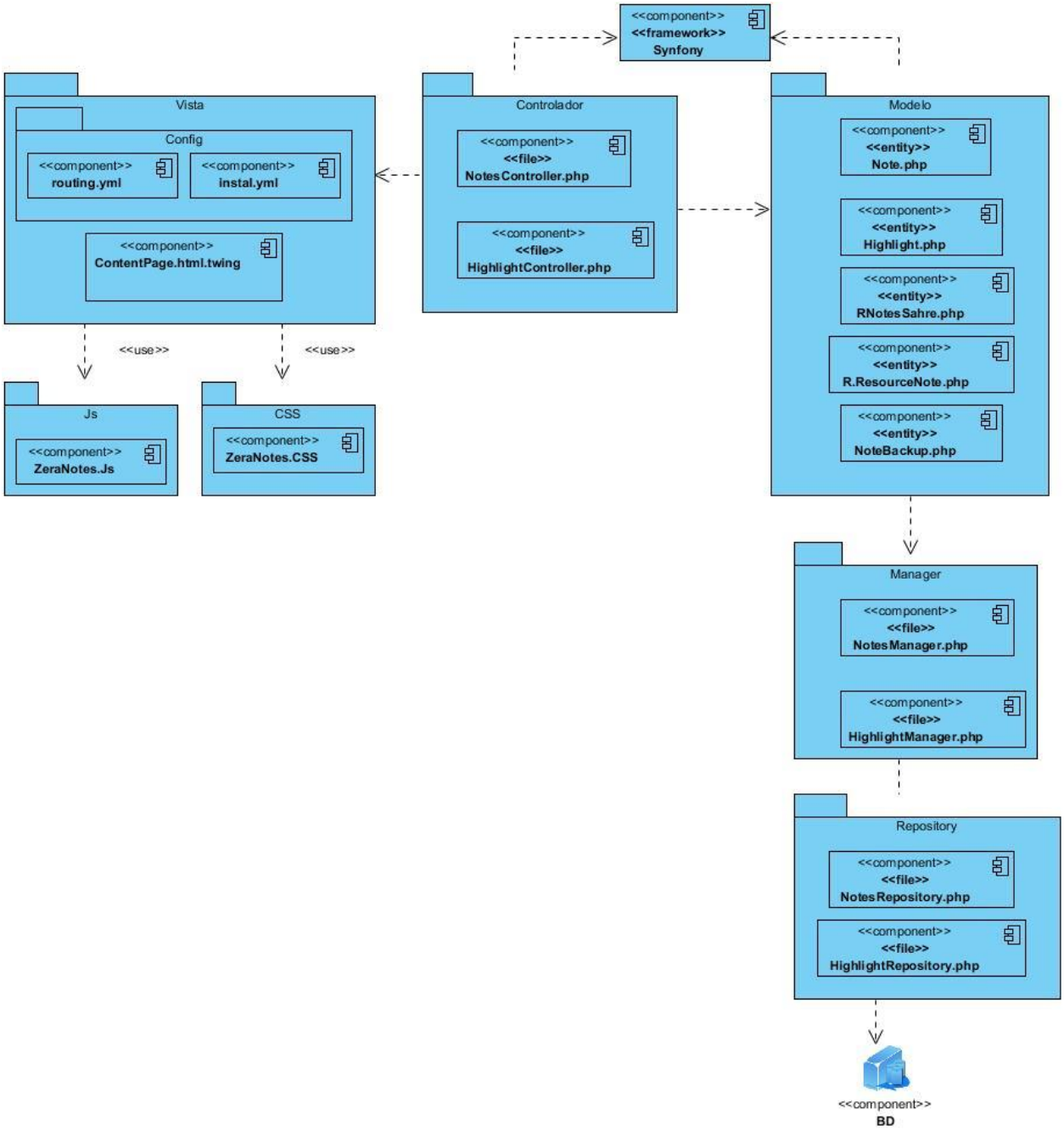


Imagen 6.Diagrama de componentes del sistema.

CAPÍTULO III

IMPLEMENTACIÓN Y PRUEBA

3.2 Código fuente

El código fuente es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para ejecutar dicho programa. Por tanto, en el código fuente de un programa está escrito por completo su funcionamiento. Estas líneas de texto están escritas en un lenguaje de programación específico y que puede ser leído por un programador. Debe traducirse a lenguaje máquina para que pueda ser ejecutado por la computadora o a bytecode para que pueda ser ejecutado por un intérprete. Este proceso se denomina compilación. (54)

3.2.1. Estándares de codificación

La forma de escribir código es propia de cada programador y completamente diferente a la forma de cualquier otro. De la forma usada depende la facilidad para entender el código y retomar ciertas partes realizadas por otros integrantes, así como la depuración de las mismas.

Los estándares de código, son parte de las llamadas buenas practicas o mejores prácticas, estas son un conjunto no formal de reglas, que han ido surgiendo en las distintas comunidades de desarrolladores con el paso del tiempo y las cuales, bien aplicadas pueden incrementar la calidad del código, notablemente. (55)

Para la propuesta de solución fueron utilizados diferentes estándares de codificación con respecto a los lenguajes de programación correspondiente, los cuales son los establecidos por el proyecto al cual va dirigido el presente trabajo de diploma.

3.2.2 Estándares de codificación utilizados

Java Script

- Se utiliza la notación camelCase para los nombres de variables y funciones.
- Todos los nombres comienzan con una letra.

- Siempre utilice la palabra reservada "var" para declarar variables.
- Los nombres de las variables globales escritos en *uppercase.*
- Los nombres de las constantes escritos en *uppercase*.
- Abre llave al final de la primera línea.

CSS

- La declaración CSS siempre termina en punto y coma y los conjuntos de declaraciones se colocan entre llaves.
- Colocar la llave que cierra en una línea nueva.
- Para nombres compuestos de una propiedad se utiliza el guión (-).
- Utilizar dos puntos más un espacio entre cada propiedad y su valor.
- Colocar la llave que abre en la misma línea que el selector.
- Un comentario CSS comienza con /* y termina con */. Los comentarios pueden abarcar varias líneas.

HTML

- Siempre declarar el tipo de documento en la primera línea el documento.
- El elemento <html> es la raíz del documento. Es el lugar recomendado para poner el lenguaje de la página.
- Cerrar todas las etiquetas html.
- Utilizar *lowercase* para el nombre de los atributos de las etiquetas.

- Para asegurar una interpretación apropiada y correcta indexación de los motores de búsqueda, tanto el lenguaje como la codificación de caracteres debe definirse lo antes posible en un documento.

PHP

- Utilizar la notación camelCase para los nombres de variables, funciones, métodos y argumentos
- Utilizar guiones bajos para nombres de opciones y parámetros .
- Utilizar namespaces para todas las clases.
- Sufije las excepciones con Exception.
- Sufije las excepciones con Interface.
- Prefije las clases abstractas con Abstract.
- Utilizar paréntesis cuando instancie una clase, sin tener en cuenta la cantidad de argumentos del constructor.

3.3 Pruebas de software

Las pruebas de *software* son un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación.(55) Esta actividad forma parte del proceso de control de calidad global. Las pruebas son básicamente un conjunto de actividades dentro del desarrollo de *software* y dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento del proceso de desarrollo.

Las pruebas se rigen por una serie de principios, una buena comprensión de estos facilitará el posterior uso de los métodos en un efectivo diseño de casos de prueba. Entre sus principales objetivos se encuentra detectar defectos en el *software*, comprobar la integración adecuada de

los componentes además de verificar que todos los requisitos se han implementado correctamente entre otros.

3.3.1 Niveles de prueba

Conjunto de pruebas que se le aplican al *software* en diferentes etapas del proceso de desarrollo y que son agrupadas en niveles para poder verificar y validar el producto o software. A continuación, se detallan los niveles de pruebas los cuales ayudaron a la detección de los errores existentes.

Pruebas de sistema: tienen como propósito fundamental ejercitar profundamente el sistema desarrollado, con el objetivo de verificar que se hayan integrado correctamente todos los elementos del sistema y que realizan correctamente las funciones descritas. Este tipo de pruebas estudia el producto completo para analizar defectos globales o para estudiar aspectos específicos de su comportamiento, tales como seguridad o rendimiento.(55)

Cada nivel de prueba engloba una técnica o tipo de prueba específica. A continuación, se detalla el tipo de prueba que ayudo a evaluar la funcionalidad del sistema.

Pruebas funcionales: se aseguran del trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados.(55)

Para la ejecución de las pruebas serán aplicadas por el propio equipo de desarrollo las pruebas de sistemas para tratar el componente como un todo y que todos los requerimientos estén implementados satisfactoriamente. Se decide aplicar como tipo de prueba las pruebas funcionales ya que asegura el apropiado trabajo de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados. Las pruebas funcionales se van a enfocar en los requisitos funcionales.

Pruebas de aceptación: las pruebas de aceptación son aquellas que son diseñadas por el propio equipo de desarrollo en base a los requisitos funcionales especificados, y ejecutadas por

el propio usuario de manera que este dé validez y conformidad al producto que se les está entregando en base a lo que se acordó inicialmente. De forma general las pruebas de aceptación pueden afrontarse mediante dos tipos de procedimiento para realizarlas: pruebas alfa y pruebas beta.

3.3.2 Métodos de prueba

Existen métodos de prueba independientemente del tipo que se utilice o el nivel en que se enmarquen estas técnicas. Estos métodos proporcionan distintos criterios para generar casos de prueba que propicien fallos en los programas, agrupándose en:

Pruebas de caja blanca: se basan en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del *software* proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones, denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba.(55)

Pruebas de caja negra: también denominadas pruebas de comportamiento, se centran en los requisitos funcionales de software, o sea las pruebas de caja negra permite al ingeniero obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa, más bien se trata de un enfoque que intenta descubrir diferentes tipos de errores.(55)

Para validar la propuesta de solución se empleará el método de caja negra para descubrir diferentes tipos de errores que se llevan a cabo sobre la interfaz del software y demostrar que las funciones del software son operativas.

3.3.3 Diseño de casos de prueba

El objetivo principal para el diseño de casos de prueba es derivar un conjunto de pruebas que tienen la mayor probabilidad de descubrir errores en el software. Con el propósito de comprobar

que todos los requisitos de una aplicación son revisados incluyendo un conjunto de entradas y resultados esperados.

A continuación, se presenta el diseño de casos de prueba propuesto para la HU Crear Apuntes. El resto de los artefactos de este tipo se encuentran en el Anexo 3.

Tabla 3. Diseño de casos de prueba propuesto para la HU Crear Apuntes

Descripción general Permitir crear un nuevo apunte en el sistema. **Condiciones de ejecución** El usuario debe estar autenticado en el sistema con el rol Profesor o Estudiante. Para Crear un apunte, este debe estar asociado a una parte del contenido, se selecciona previamente la opción crear el apunte y seguidamente el texto al que pertenecerá el apunte.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Crear apunte.	Selecciona la opción de crear un nuevo apunte.	El sistema queda a la espera de que el usuario seleccione el texto del apunte y devuelve el texto seleccionado como un apunte resaltado en otro color. Además, se actualiza le listado de apuntes.	Todos los cursos/Acceder/Crear apunte .

3.3.4 Resultados obtenidos

Resultados de las pruebas de sistema

A través del método de caja negra y apoyados en el diseño de CP se realizaron tres iteraciones de pruebas internas pertenecientes al nivel de sistema. Dichas pruebas fueron realizadas con el objetivo de detectar y corregir errores que impidieran el correcto funcionamiento de la solución.

Para realizar dicha prueba se hizo uso de la técnica de partición de equivalencia que es una técnica de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba. El diseño de casos de prueba para partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada.(56)

Para evaluar la solución se realizaron 3 iteraciones donde se probó el software íntegramente. A continuación, se presentan los resultados arrojados durante las diferentes pruebas aplicadas:

Tabla 4. Resultados arrojados durante las diferentes pruebas aplicadas

Iteraciones	**Cantidad de casos de prueba**	**No conformidades detectadas**			
		Alta	**Media**	**Baja**	**Total**
1	11	5	9	12	26
2	11	3	6	8	17
3	11	0	0	0	0

Resultado de las pruebas de aceptación

Se decide aplicar las pruebas alfa ya que en ellas se le entrega a un usuario final todo el producto terminado, junto a su documentación correspondiente para que este, en presencia del desarrollador y en entornos previamente preparados para el proceso de dichas pruebas, vaya informando de las inconsistencias y errores que se detecte.(56)

Se decide aplicar 3 iteraciones de las pruebas de aceptación, arrojando 5 no conformidades siendo corregidas al final de cada iteración.

Las no conformidades se clasificaron según su complejidad definiéndose de la siguiente forma:

- Alta son los errores en el código o errores de funcionalidad.
- Media son los errores de ortografía o de internalización.
- Baja son los errores de interfaz.

A continuación, se presentan los resultados arrojados durante las diferentes pruebas aplicadas:

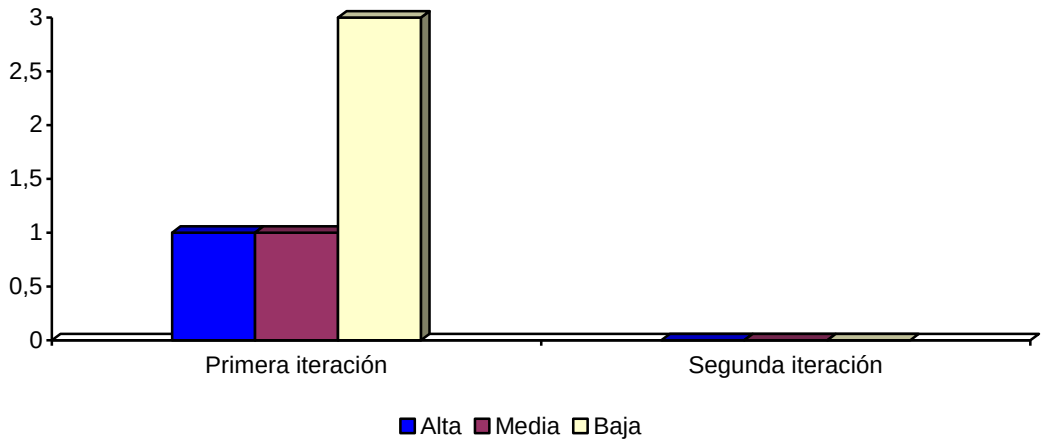


Imagen 7. Grafico que representa la relación de No conformidades(NC) detectadas y NC corregidas.

3.4 Conclusiones del capítulo

El proceso de implementación estuvo guiado por el diagrama de componentes el cual muestra la organización y las dependencias lógicas entre un conjunto de componentes de software. Además, las pruebas realizadas validaron que las funcionalidades desarrolladas satisfacen los requisitos especificados, dejando listo el software con vista a su implantación.

Conclusiones generales

Una vez concluida la investigación se arriba a las siguientes conclusiones:

- El estudio y análisis de las soluciones similares permitió comprender el funcionamiento de los componentes para la gestión de notas en los Sistemas Gestores del Aprendizaje y así justificar las herramientas, lenguajes y tecnologías seleccionadas para el desarrollo de la investigación.

- El componente se desarrolló siguiendo la metodología AUP-UCI y se utilizaron representaciones UML para la modelación de todas las fases del proyecto.

- Se obtuvo un componente que contribuye a elevar las opciones para la autogestión del aprendizaje y el trabajo colaborativo dentro de la Plataforma Educativa ZERA 2.0, mediante la gestión de apuntes y el resaltado de texto.

- Se demostró a partir de las 3 iteraciones de pruebas de sistema y de aceptación practicadas al software, que este satisface los requisitos funcionales definidos durante el flujo de trabajo.

Recomendaciones

Concluida la investigación y con el objetivo de perfeccionar la aplicación y lograr una mayor explotación de sus potencialidades se recomienda:

- Permitir exportar los apuntes realizados en algún formato compatible con herramientas de notas externas.
- Tener en cuenta este trabajo para proyectos futuros, siempre y cuando se cumpla con las normas de confidencialidad establecidas

Referencias bibliográficas

1. José Francisco García. Estado actual de los sistemas e-learning. 2005.

2. Susana Cabeza Vidal. Revisión de Recursos Educativos en la Plataforma Educativa ZERA. Universidad de las Ciencias Informáticas.

3. Rodríguez, Yerandy Manzo Guerra, Roxana Cañizares, González, Juan Pedro Febles. Plataforma Educativa ZERA: Modelo de adaptación de contenido sensible al contexto. 2015.

4. Centro FORTES. ZERA ,Plataforma Educativa. [en línea]. [Accedido 24 Noviembre 2016]. Disponible: https//eva.uci.cu/es/

5. Elena Villasana, Dorego Nallilda. Habilidades sociales en entornos virtuales del trabajo colaborativo. 2007. Vol. 10.

6. José Luis Rodríguez Dieguez. *Tecnología educativa. Nuevas tecnologías aplicadas a la educación. Marfil., 1995.*

7. Navarro. Uso intensivo de herramientas de colaboración en línea en educación superior. *6*. 2013.

8. Mónica Soler, Ranzani ,Pascual Solanas, Saura. Autogestión en el proceso de aprendizaje. El viaje Atiaca : *UNVEST2011*. Girona, Enero 2011.

9. Ros Martínez de Lahidalga, Iker. Moodle, la plataforma para la enseñanza y organización escolar. 2008.

10. Guerra, Yerandy Manso, González, Roxana Cañizares y Rodríguez, Juan Pedro Febles. Plataforma educativa ZERA: modelo de adaptación de contenidos sensible al contexto. Digital Education Review. 2015. No. 27, p. 154–164.

11. Herrera, Mike. Iingeniería del software: Metodologías y ciclos de vida Laboratorio Nacional de Calidad del Software. [en línea]. [Accedido 14 Enero 2017]. Disponible: http://www.academia.edu

12. Canós, José H., Letelier, Patricio y Penadés, Mª Carmen. Metodologías ágiles en el desarrollo de software. Metodologías Ágiles en el Desarrollo de Software. 2003. P. 1.

13. Maguana, Aucancela, Javier, Carlos y Ponzo Molina, Tatiana Carolina. Desarrollo dirigido por test utilizando el *Framework Junit* en un Sistema Web de asignación de laboratorios computacionales de la ESPE aplicando la metodología AUP. 2011.

14. Jorge Sánchez. Apuntes Completos. Sistemas de Bases de Datos. 2015.

15. Martínez, Rafael. Sobre PostgreSQL. Recuperado el. 2010. Vol. 8.

16. José Enrique González. ¿Qué es UML? el lenguaje de modelado Unificado. 2008.

17. UCML.Herramienta CASE.¿Cómo incorporarlas con éxito en nuestra organización? [en línea]. [Accedido 5 Diciembre 2016]. Disponible: www.uclm.es/ab/educacion/ensayos/pdf/revista10/10_17.pdf.

18. Pressman Roger, S. Ingeniería del Software Un enfoque práctico, Sexta Edición McGrawHill. ISBN 970-10-54-73-3.

19. Leslie Wilson. *Comparative Programming Languajes.* Segunda edición. 1993. ISBN 0-201-56885-5.

20. Documentos de lenguajes de Programación del Lado Servidor. 2016.

21. Juan Diego Gauchat. *El gran libro de HTML5, CSS3 y JavaScript. 2012.*

22. Brigges, Owen, Champeon, Steven, Costello, Eric y Patterson, Matt. *Cascading Style Sheets*. 2003. ISBN 84-415-1497-6.

23. David FLangan y Anaya Multimedia (eds.). *JavaScript.* La Guía Definitiva. España, 2007. ISBN 978-84-415-2202-2.

24. *Home - Twig - The flexible, fast, and secure PHP template engine*. [en línea]. [Accedido 22 Enero 2017]. Available from: http://twig.sensiolabs.org/

25. Cobo, Ángel, Gómez, Patricia, Pérez, Daniel y Rocha, Rocío. PHP y MySQL. Tecnología para el Desarrollo de aplicaciones web, Ediciones Díaz de Santos, 5ta. ED, España. 2005.

26. PHP: ¿Qué puede hacer PHP? - Manual. [en línea]. [Accedido 14 Enero 2017]. Disponible: http://php.net/manual/es/intro-whatcando.php

27. Gutiérrez, Javier J. ¿Qué es un *framework* Web? Disponible*: http://www. lsi. us. es/~ javierj/investigacion_ficheros/Framework. pdf Accessed May*. 2014. Vol. 12.

28. Mejía Abad, Jhonatan Javier. Desarrollo de una Aplicación Web que Automatice el Control Docente en la Unidad Educativa Sucúa. 2016.

29. Álvarez, Miguel Ángel. Manual de jQuery. Recuperado el, 2010.

30. Cobo, Ángel. PHP y MySQL*:* Tecnología para el desarrollo de aplicaciones web. Ediciones Díaz de Santos, 2005. ISBN 84-7978-706-6.

31. CSS Bootstrap. [en línea]. [Accedido 15 Diciembre 2016]. Disponible: http://getbootstrap.com/css/

32. Amador, Leydi Johana Polo. Estudio comparativo de sistemas de mapeo objeto relacional desarrollados en plataformas *Open Source*. *CITECSA*. 2013. Vol. 3, no. 5, p. 74–83.

33. María Dolores Lozano Pérez, Isidro Ramos Salavert. Ingeniería del software y bases de datos: tendencias actuales*.* Entornos de Desarrollo Integrados. Universidad de Castilla La Mancha. ISBN 84-8427-077-7.

34. N*etBeans IDE - Overview.* [en línea]. [Accedido 8 Marzo 2017]. Disponible: https://netbeans.org/features/index.html

35. NEDELCU, Clément. *Nginx HTTP server second edition*. Packt Publishing Ltd, 2013. ISBN 1-78216-233-X.

36. pgAdmin: PostgreSQL *administration and management tools*. [en línea]. [Accedido 8 Marzo 2017]. Disponible: https://www.pgadmin.org/

37. Jacobson, IvarBooch, Rumbaugh, Grady, Jacobson, JamesIvar, Booch, Grady y Rumbaugh, James. El proceso unificado de desarrollo de software/*The unified software development process.* Pearson Educación, 2000. ISBN 84-7829-036-2.

38. DesarrolloWeb.com. Usabilidad y arquitectura del *software*. DesarrolloWeb.com [en línea]. [Accedido 14 Enero 2017]. Disponible: http://www.desarrolloweb.com/articulos/1622.php

39. Erica Camacho y Flavio Cardasco. Arquitectura de *Software.* 2004.

40. González, Yanette Díaz y Romero, Yenisleidy Fernández. Patrón Modelo-Vista-Controlador. *Revista Telem@ tica*. 2012. Vol. 11, no. 1, p. 47–57.

41. DesarrolloWeb.com. Qué es MVC. DesarrolloWeb.com [en línea]. [Accedido 14 Enero 2017]. Disponible: http://www.desarrolloweb.com/articulos/que-es-mvc.html

42. Larman, Diapositivas. Modelo Del Dominio. de UML y Patrones, Prentice Hall. 2003. P. 23.

43. Sommerville, Ian. Requerimientos del software. Ingeniería del Software, Pearson, Madrid. 2005. P. 110–111.

44. DUARTE, Ailin Orjuela and ROJAS, Mauricio. Las metodologías de desarrollo ágil como una oportunidad para la ingeniería del software educativo. *Avances en Sistemas e Informática*. 2008. Vol. 5, no. 2.

45. Cosmin, Puiu Ionut. Patrones de Diseño. *MoleQla:* revista de Ciencias de la Universidad Pablo de Olavide. 2016. No. 23, p. 36.

46. Departamento de lenguajes y sistemas informáticos. Patrones de Asignación de Responsabilidad(GRASP).

47. Ros, Miguel Zapata. Patrones en *elearning*. Elementos y referencias para la formación. Revista de Educación a Distancia. 2015. No. 27.

48. Mendoza Navarro, Javier. Diseño del sistema de tarjeta de crédito con UML. 2002.

49. Gramage, María Carmen Penades. Diagrama de clases. 2016.

50. Vidal, Cristian L, Schmal, Rodolfo F, Rivero, Sabino y Villarroel, Rodolfo H. Extensión del Diagrama de Secuencias UML (Lenguaje de Modelado Unificado) para el Modelado Orientado a Aspectos. Información tecnológica. 2012. Vol. 23, no. 6, p. 51–62.

51. Sarmiento, Johana. UML: Diagrama de Despliegue [en línea]. Bogotá: La Empresa [citado 20 septiembre, 2014]. Disponible en Internet URL: http://umldiagramadespliegue. blogspot. com.

52. *Atlantic International University*. Modelos de Datos. [en línea]. [Accedido 7 Marzo 2017].Disponible: http://www.aiu.edu/cursos/base/ 2.pdf.

53. James Rumbaugh, Ivar Jacobson, Grady Booch. El Proceso Unificado de Desarrollo de Software. Manual de Referencia*. S.l Addison Wesley*.

54. Booch, Grady, Rumbaugh James, Jacobson Ivar. El lenguaje unificado de modelado. Manual . 2009.

55. Roger S.Pressman. Ingeniería del Software*.* Un enfoque práctico. 5ta Edición. 2002.

56. Roger S.Pressman. Ingeniería del Software*.* Un enfoque práctico. 2009.