Universidad de las Ciencias Informáticas

Facultad 4

Gestor de apariencia para la colección “El Navegante” en su versión multiplataforma.

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

Autores:

Maday Bárbara Recio Silva

Carlos Tarrau González

Tutores:

Ing. Jorge Martínez Padrón

Ing. Jorge Martínez García

Co-tutora:

Msc. Ivonne Burguet Lago

Ciudad de La Habana, Cuba

Junio, 2012

DECLARACIÓN DE AUDITORÍA

Declaramos ser autores de este trabajo y autorizamos a la Facultad (4) y a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ _____del 2012.

_____ _____

Firma del Autor: Firma del Autor

Maday Bárbara Recio Silva Carlos Tarrau González

_____ _____

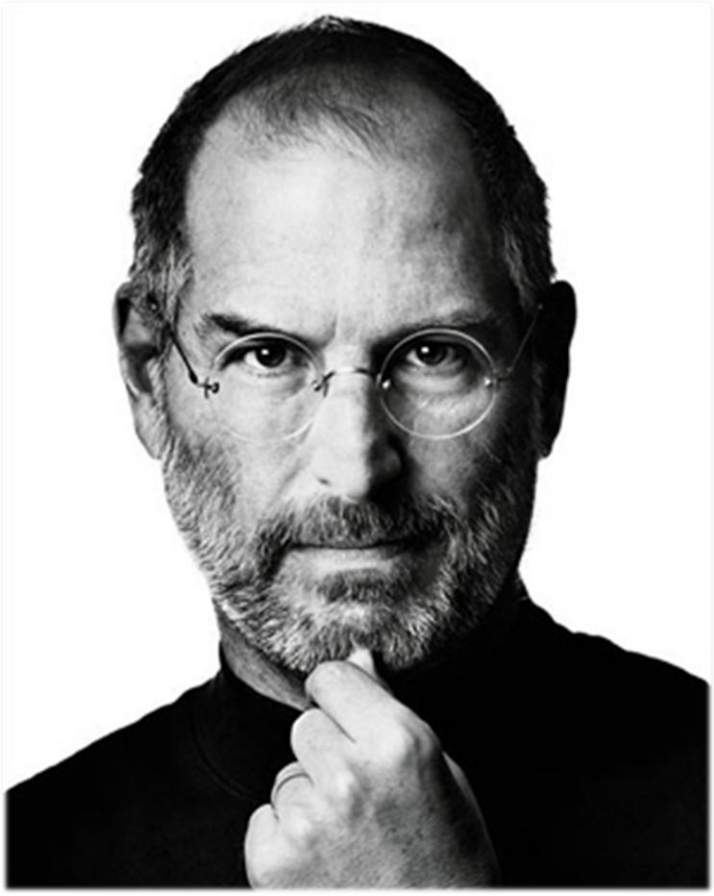
Firma del tutor: Firma del tutor:

Ing. Jorge Martínez García Ing. Jorge Martínez Padrón

PENSAMIENTO

"Es difícil diseñar productos centrándose en el público objetivo. Muchas veces, la gente no sabe lo que quiere hasta que se lo enseñas."

Steve Jobs



AGRADECIMIENTOS

De Maday:

A mis padres por ser la razón de mi ser, por siempre apoyarme, darme consejos, guiarme, por enseñarme lo que está correcto, darme las fuerzas que me faltan cuando me he sentido agotada a mitad de camino, por enseñarme que cometer errores y caer le pasa a todos; lo difícil es levantarse. Gracias a ellos, por ser mi base de inspiración he logrado evadir los obstáculos que se me han presentado. Agradezco y aprecio por sobre todas las cosas, todo el esfuerzo que han realizado por mí, para que yo cumpla este sueño. Gracias a los dos por acompañarme en esta trayectoria.

A mi hermana por ser lo mejor de mi vida y lo que más quiero, por permitirme ser un ejemplo en su vida, por demostrarme su cariño.

A toda mi familia, en especial a mi tía Betty por apoyarme, ayudarme con mis trajines, a mi tía y tío (Baby y Elio) porque siempre me tienen presentes y se preocupan por mí, a mis abuelos Toño y Toña por siempre estar pendientes y complacerme en mis ñoñerías, a mi padrastro por su atención y esfuerzo de cada día por la familia, a mi tío Gera y su familia, a mis primos y primas cercanos, a mi prima Liudmila que ha estado siguiendo mis pasos siempre en contacto conmigo desde que nos conocimos aquí en la universidad, a mi prima Odalys, Yainie, a mi tía Lourdes por su preocupación, a mi madrastra y su familia por también estar pendientes y brindarme su apoyo.

A mis amigas Dorys, Bleydis, Yailin, Arelis, Lanneis y mi amigo Hector que han sido los más cercanos y han estado siempre a mi lado, dándome ánimos, apoyo, ayuda, por compartir buenos y malos momentos.

A mi compañero de tesis por su esfuerzo, paciencia, dedicación y apoyo siempre que lo necesitaba, también por soportarme.

A mis compañeros de aula de todos los cursos, a mis amigos de Nuevitas en especial Angel, Baby y Yuni, a mis amigas distantes, en especial a Raylith por ayudarme mucho a comprender las cosas de la vida y por siempre darme ánimos, ayudarme en todo y preocuparse por mí.

A mis compañeros de universidad, a todos los que conocí en determinados momentos de la carrera y que me permitieron compartir momentos inolvidables con ellos. A todas estas personas siempre las llevaré presente y nunca las olvidaré.

A todos los profesores que me impartieron clases o trabajaron junto conmigo en el proyecto, a ellos por enseñarme cosas tanto de la materia como de la vida, en especial a Aliuzka, Syed, Osdalme, Yusimy, Yolanda, Thaymi, Lizandra y otros que aunque no los mencione en estos momentos hicieron un gran trabajo conmigo y siempre los llevaré presentes.

A mis mascotas Luna, puchuchu y Tea.

AGRADECIMIENTOS

A todas las personas que aportaron su ayuda y estuvieron juntos conmigo en este camino, q me apoyaron y ayudaron para que pudiera estar donde estoy en estos momentos.

A la Universidad de las Ciencias Informáticas por darme la oportunidad de ser parte de ella y permitir que me eduque y convierta en una profesional.

AGRADECIMIENTOS

De Carlos:

A mis padres por darme vida y luz durante toda mi vida, por ser parte de mis triunfos, por ser parte de mis puños a la hora de tumbar muros, por darme guía, fe, apoyo a toda costa para lograr mis metas, por hacerme cada día un hombre de bien. Mi padre ejemplo a seguir y firmeza, mi madre espíritu de familia y amor, este trabajo final es el resultado de andar junto a mis padres, siendo ellos la guía de mi camino, convirtiéndome en un profesional de la vida que puede avanzar por la misma sin dudar de su andar.

A mi novia Daymaris por darme todo su amor en los momentos difíciles de la carrera, por darme apoyo cuando todo se volvía oscuro, gracias a ella he podido encontrar la razón del esfuerzo universitario y la satisfacción de un logro bien merecido. A ella le debo mucho ya que esta tesis es su primera victoria y me la sede a mí. Con su luz me he levantado y superado los obstáculos de la vida, llegando a ser su caballero a seguir.

A mi familia, mis abuelos, tíos, primos, incluyendo los que están en el cielo, que desde muy pequeño creyeron en mí y cada uno dio una pizca de su corazón para que pudiera llegar a donde he llegado y lograr alcanzar el futuro con victorias. Gracias por ser esa enorme familia que siempre he tenido, por unirse a la hora de apoyarme y educarme, por darme cariño y amor desde cada hemisferio que se encuentren.

A mi compañera de tesis Maday que ha sido compañera de trabajo desde hace varios años, que sabe entenderme, que me apoya en todo, que me enseña y educa para ser cada día más noble y eficaz, por ser ese ángel que ennoblece el corazón de los hombres, gracias a ella me he sentido confiado para poder lograr mi trabajo final.

A mis amigos de toda la vida que siempre han estado en las buenas y en las malas tomándome como una persona merecedora de éxitos: Lena, Diana, Bruno, Osvaldo, Ángel, Leandro, Ariam, Pedro Pablo, Ulises y más.

A mis hermanos que desde primer año estamos juntos enfrentando vientos y mareas logrando nuestro principal objetivo: Alberto y Pablo, que me han adoptado como un familiar y me han apoyado en las buenas y en las malas.

A mi prima adoptiva Liset, por ser ejemplo, alegría y amor en esta escuela, que desde un principio me dio apoyo, me ayudó en todo y nunca dudó en darme una mano amiga para hacerme cada día mejor.

A mi equipo de baile Tecktonik por ayudarme a ser alguien en esta escuela, y poder alcanzar uno de mis más añorados sueños de mi vida, desempeñarme en un escenario como artista.

AGRADECIMIENTOS

Al equipo de kikimboll de la facultad por darme completa confianza tanto en los juegos deportivos como en la vida cotidiana, por dejar que las guiara a ser un mejor equipo.

A todos los que de un modo u otro han influido en mi vida durante el tránsito por la Universidad y siempre han estado al tanto de mí. En fin a todas aquellas personas que quiero y me quieren les agradezco.

Ambos:

Al colectivo de profesores y estudiantes del proyecto Multisaber-Navegante por brindarnos tiempo y esfuerzo propio para poder lograr este trabajo final, destacando a: Liana que desde el primer año de Carlos fue fuerte con él y le hizo entender que la vida no es juego, enseñándole que cada día hay que esforzarse más para poder vencer las metas enmarcadas. A Osdalme por ser buen profesor y amigo estando siempre pendiente de la tesis y de otras tareas, Osmany, Hector, Arisbel, Angel, Lara, Katia, Nersa, Ana, Rosa, Licet, Jacobo, Ismael, Cao y otros profesores de este equipo, por nunca escatimar a la hora de ayudarnos, por aportar ideas y ayuda en este trabajo. A los estudiantes Yolanda, Ernesto, Ramiro, Yordy, Mayara, Jorge, Rajiv, Hector, Arelis por nunca dudar en ayudarnos a cualquier hora, por compartir parte de su tiempo con nosotros.

A los profesionales Yaismel, Osvaldo y Yusdel por ser guía y apoyo de cómo realizar esta tesis, por regalarnos tiempo a la hora que siempre hizo falta sin dudar, por siempre tenerlos como guía y ejemplo para lograr alcanzar la meta primordial y por dar la mayor confianza de cada uno de que un hombre si puede alcanzar las estrellas solo con esfuerzo y superación.

A nuestros tutores, sostén de la colección "El Navegante" por siempre estar ahí presentes para ayudarnos a avanzar, por encaminarnos para poder lograr buenos resultados y poder convertirnos en ingenieros de verdad.

DEDICATORIAS

Dedico esta tesis principalmente a mis padres, a mi hermana, a toda mi familia, amigos y a todos aquellos que se han sabido ganar mi cariño y confianza, a mi compañero de tesis por juntos formar un eslabón e ir evadiendo los obstáculos teniendo presente de que paso a paso con amor, esfuerzo y sacrificio se alcanza el objetivo.

Maday

A mis padres Minerva y Carlos, por haberme apoyado más que en mi carrera en todos los años vividos logrando convertirme en un hombre hecho y derecho, a mi novia Daymaris por regalarme todo su amor y vida convirtiéndose en un perfecto bastón consejero, amiga, hermana y compañera, a mi familia completa por siempre creer en mi a pesar de mis caídas, a mi compañera de tesis Maday por dejarme formar parte de su vida para convertirnos en un gran equipo, a mis hermanos de universidad que me dan aliento en cualquier situación , a todos aquellos profesores y educandos que han dejado en mi un pedazo de su sabiduría para poderme realizarme como un profesional. A todos aquellos que de una forma u otra han hecho posible lo que soy hoy.

Carlos

RESUMEN

Las tecnologías de la información y las comunicaciones, han logrado un creciente auge en diversas esferas de la vida humana incluyendo el ámbito educacional. Es por ello, que el desarrollo de software educativo ha arribado resultados satisfactorios en cuanto a la enseñanza y el aprendizaje, debido a eso, la Universidad de las Ciencias Informáticas y en especial la facultad cuatro se expande en esta área; donde se encuentra el proyecto Colecciones de software educativo Multisaber y "El Navegante". Actualmente en ese proyecto se está desarrollando la colección "El Navegante" en su versión multiplataforma. Una de las necesidades que se presenta en esta colección, es la agilización del cambio de apariencia de los diez productos que la conforman. Por ello, los desarrolladores dedican gran parte de su tiempo en el cambio de colores e imágenes; dado que existen muchos estilos CSS que están desorganizados, con nombres que tienden a confundir y crear dudas. Para dale solución a esta necesidad, el presente trabajo tiene como objetivo desarrollar un componente que agilice la gestión de apariencia de la versión multiplataforma de la colección "El Navegante". Para cumplir ese objetivo y adecuándose a la arquitectura definida en el proyecto, se seleccionó la metodología *Rational Unified Process* (RUP) para apoyar a la solución del problema y generar todos los artefactos necesarios, los cuales permitirán posteriores modificaciones. Se identificaron las funcionalidades, lo que permitió el desarrollo de un componente integrado al panel de administración de la colección, que permite realizar cambios de apariencia en un instante.

Palabras claves: agilización, apariencia, colección "El Navegante", componente, estilos CSS, gestión, multiplataforma, software educativo.

Índice

Índice de tablas

Índice de figuras

Introducción

El mundo de hoy se caracteriza por el vertiginoso progreso tecnológico. Las Tecnologías de la Información y las Comunicaciones (TIC) están presentes en todas las esferas de la vida humana, incluyendo el ámbito educacional. Durante la última década, prácticamente la mayoría de los países han implementado programas nacionales destinados a capacitar a los docentes en la utilización de las TIC. También los países latinoamericanos han realizado un gran esfuerzo en este sentido.

En el caso de Cuba, a pesar de las difíciles condiciones económicas, tecnológicas y de comunicaciones impuestas por el bloqueo de los Estados Unidos que obstaculizan seriamente la política cubana de uso social e intensivo de las TIC, se ha decidido adoptar como opción de desarrollo inicial el uso de sus escasos recursos de conectividad y medios técnicos, priorizando principalmente la esfera de la educación. La introducción progresiva de las TIC en el Ministerio de Educación (MINED) ha incrementado las potencialidades de aprendizaje en la niñez y la juventud, ya que con el uso de las nuevas tecnologías en dicha esfera se facilita el trabajo a los estudiantes en la adquisición más fácil de un mayor conocimiento; por ello la creación de software educativos ha tenido una evolución significativa, los cuales cubren plenamente las áreas del currículo escolar en los diferentes niveles de enseñanza, sobre la base de la concepción de "Hiperentornos de aprendizaje". Estos fueron definidos por el Msc. César Labañino como "sistema informático basado en tecnología hipermedia que contiene una mezcla o elementos representativos de diversas tipologías de software educativo" (1).

El ministerio de la educación (MINED) crea la colección "El Navegante", dirigida a la Secundaria Básica como apoyo al Profesor General Integral (PGI), para desarrollar habilidades cognoscitivas en los estudiantes de dicha enseñanza. Esta colección poseía restricciones en cuanto a su uso y distribución, al desarrollarse con herramientas propietarias. Por lo que en la Universidad de Ciencias Informáticas (UCI), en convenio con el MINED y específicamente en el proyecto "Colecciones de software educativo Multisaber y El Navegante"; se está desarrollando una versión multiplataforma de dicha colección que sea flexible, eficiente, segura y organizada de manera genérica para futuras posibilidades comerciales.

La actual versión multiplataforma de la colección "El Navegante" está compuesta por diez productos, inspirados en una concepción integradora de los contenidos del nivel secundario.

Tabla 1: Productos de la colección "El Navegante"

No.	Nombre del software	Asignaturas	Grados
1.	Elementos Matemáticos	Matemática	7, 8, 9
2.	El fabuloso mundo de las palabras	Español-Literatura	7, 8, 9
3.	Encuentro con el pasado	Historia Antigua, Medieval y Geografía	7
4.	GeoClío	Historia Moderna, Historia Contemporánea y sus espacios geográficos	8
5.	Por los senderos de mi Patria	Historia de Cuba y Arte cubano	9
6.	Aprende construyendo	Educación Laboral y Dibujo básico	9
7.	Rainbow	Inglés	7, 8, 9
8.	Informática Básica	Computación	7, 8
9.	EducArte	Educación artística	7, 8, 9
10.	La naturaleza y el hombre	Física, Química, Biología y Geografía	7, 8, 9

Cada uno de estos productos está integrado por seis módulos conforme a la estructura genérica de un hiperentorno de aprendizaje. Actualmente se encuentra en desarrollo el primer producto, llamado "Elementos Matemáticos", para el que se definieron una serie de pautas generales en cuanto a la apariencia gráfica que son comunes para toda la colección; así cada producto puede ser personalizado dentro de un diseño coherente. Pero gestionar la apariencia de cada uno de estos productos puede resultar complejo debido a que la arquitectura de la colección no define adecuadamente este proceso. Los

desarrolladores incorporados que comienzan a interactuar con la colección tienen que estudiarse todos los estilos elaborados para adaptarse a las pautas antes definidas. Como solución al problema anterior se podría pensar en impartir capacitación a cada nuevo desarrollador incorporado al proyecto, lo que representa un atraso para el cronograma de actividades definidas a cumplir en un plazo determinado, inclusive, la mano de obra de desarrolladores estaría siempre atareada con el proceso de las gestiones de plantillas. Otra limitación existente es que a partir de la definición del documento de arquitectura no existe homogeneidad de los estilos, ya que en el proyecto se trata que estos sean lo más comunes posible; pero finalmente no se cumple con lo establecido y todos los desarrolladores no se guían estrictamente por las pautas definidas, como es el caso de las capas, los identificadores, entre otras.

En el ámbito global los problemas antes descritos pueden generar una limitación comercial, ya que generalmente cada nuevo cliente requiere una personalización de la apariencia gráfica del producto, es aquí donde el tiempo y la mano de obra de los desarrolladores es crucial para manejar estos pedidos; tener que realizar los cambios de apariencia en las plantillas del proyecto resultaría tedioso y consumiría más tiempo de lo planificado para desarrollar otras tareas, cuando lo que pidió el cliente es tan solo un requisito no funcional de cambio de apariencia. Ante estas limitaciones se decidió desarrollar un componente integrado al panel de administración de la colección que agilice la gestión de la apariencia gráfica de los productos que la integran.

Teniendo en cuenta lo planteado anteriormente queda definido el siguiente **problema a resolver**:

- ✓ ¿Cómo agilizar la gestión de apariencia de la colección “El Navegante” en su versión multiplataforma?

Del mismo se deduce como **objeto de estudio:**

- ✓ La gestión de apariencia en sistemas informáticos.

Teniendo como **objetivo general**:

- ✓ Desarrollar un componente que agilice la gestión de apariencia de la colección “El Navegante” en su versión multiplataforma.

Enmarcado en el **campo de acción**:

- ✓ Gestor de apariencia de la colección “El Navegante” en su versión multiplataforma.

Para dar cumplimiento al objetivo general planteado se desglosa en los siguientes **objetivos específicos**:

- ✓ Generar toda la documentación según la metodología a emplear, la cual permitirá realizar posteriores cambios y/o desarrollar nuevas versiones del componente.
- ✓ Implementar un componente que agilice la gestión de apariencia de la colección “El Navegante” en su versión multiplataforma.

Como **idea a defender** se expone:

- ✓ Mediante el desarrollo del componente que agilizará la gestión de apariencia de los productos que conforman la colección “El Navegante” en su versión multiplataforma, se obtendrá un subsistema que posibilite realizar posteriores cambios de vistas personalizadas y/o desarrollar la gestión de apariencia de los productos de la colección sin ocupar mucho tiempo y mano de obra.

Con el fin de dar cumplimiento a los objetivos expuestos anteriormente se plantean las siguientes **tareas a realizar**:

1. Realización del estudio del estado de arte.
2. Generación de la fundamentación teórica de la investigación.
3. Identificación de los requisitos funcionales y no funcionales de la aplicación.
4. Investigación de los diferentes patrones arquitectónicos y de diseño existentes en la colección.
5. Selección de patrones arquitectónicos y de diseño a emplear en el componente a desarrollar.
6. Realización del análisis según la metodología a emplear.
7. Realización del diseño de acuerdo a la metodología a emplear.
8. Implementación de las funcionalidades necesarias, teniendo en cuenta los requisitos funcionales y no funcionales especificados.
9. Verificación del funcionamiento del gestor de apariencia de acuerdo a los requerimientos planteados.

Para el desarrollo de la presente investigación, se empleó la combinación de los **métodos de investigación**:

Métodos teóricos:

Analítico-sintético: este método será necesario para el estudio de las diferentes herramientas y tecnologías a utilizar, también para analizar la situación problémica existente y de ella obtener exactamente qué características debe poseer la solución propuesta.

Modelación: es justamente el proceso mediante el cual se crean modelos con vistas a investigar de la realidad. Los diagramas que se realicen, guiarán todo el proceso de desarrollo del componente así como la utilización del mismo para modelar el sistema en cuestión.

Métodos empíricos:

Observación: este método es el más utilizado en la investigación científica, debido a que es un procedimiento fácil de llevar a cabo y permite percibir directamente la evolución del trabajo realizado, estando pendientes a posibles retrasos o fallas en el cumplimiento de las tareas propuestas.

Entrevista: la entrevista es un método de gran utilidad, la cual, permitió la obtención de la información necesaria para la consolidación del conocimiento cualitativo de los problemas existentes. Estos problemas son los que afectan la agilización de la labor de los desarrolladores, en cuanto al cambio de las apariencias gráficas de los productos que componen la colección. Por ello se llevaron a cabo una serie de entrevistas con el objetivo de obtener diversos criterios de la situación que se tiene en estos momentos.

El presente trabajo está organizado con la siguiente **estructura capitular:**

Capítulo 1: Se realiza el marco teórico conceptual, donde se incluye el estudio del estado del arte y los fundamentos teóricos de la investigación relacionados con las tendencias actuales de la gestión de apariencias. Además, se hace un estudio de las tecnologías, herramientas y metodologías a utilizar en el desarrollo del componente.

Capítulo 2: Se definen las diferentes características del componente a desarrollar. También se identifican los requisitos funciones y no funcionales del sistema, así como la descripción de los mismos. Además, se elabora el diagrama de casos de uso (CU) del sistema.

Capítulo 3: Se describe la solución que se propone a partir de los diferentes modelos y diagramas presentes en el flujo de trabajo de Análisis y Diseño, como por ejemplo, los diagramas de clases del análisis y los diagramas de clases del diseño.

Capítulo 4: Se muestra la implementación de la solución propuesta a partir de la descripción de los diagramas de componentes y el diagrama de despliegue. Además se verifica el funcionamiento de este componente, para ver si cumple con los requisitos antes descritos.

Capítulo 1: Marco teórico conceptual

1.1 Introducción

En el presente capítulo se aborda el marco teórico conceptual de la investigación relacionada con las tendencias actuales, tanto nacional como internacional, de la gestión de apariencias. Se realizará un análisis de las tendencias actuales de la gestión de apariencias. Además, se hace un estudio de las tecnologías, herramientas y metodologías a utilizar en el desarrollo del componente, definidas por el equipo de proyecto al cual pertenece la colección.

1.2 Análisis de soluciones similares existentes

Hoy en día existen en el mundo varias aplicaciones que tienen integrado la opción de gestionar plantillas, dando la posibilidad al usuario de seleccionar una apariencia a su gusto y comodidad; pero son los sistemas de gestión de contenido (del inglés *Content Management System*, en lo adelante CMS) las principales aplicaciones que tienen integrada esta función; por ello se seleccionaron estos sistemas para su debido análisis. También se incluyó la plataforma ZERA que es un sistema de administración de aprendizaje (del inglés *Learning Management System*, en lo adelante LMS) como ejemplo de aplicaciones realizadas en Cuba.

1.2.1. Joomla

Joomla es un CMS potente y flexible que se utiliza para sitios web, desarrollado en lenguaje PHP sobre base de datos MySQL. Este sistema requiere de una base de datos MySQL y, preferiblemente, de un Servidor HTTP Apache. Entre sus características incluye la funcionalidad de gestión de plantillas, donde los contenidos van por un lado y el diseño por otro, es decir, en cualquier momento se puede modificar el diseño de todo (o parte) del sitio web sin alterar el contenido de los artículos publicados o la estructura de secciones y categorías (organización interna). Joomla, además permite la instalación de nuevas plantillas que se pueden descargar gratuitamente de Internet. (2)

Una de las ventajas más importantes es la rapidez con la que se puede hacer un sitio web y más aún para el que domina la materia; pero aun así es necesario tener conocimientos en el dominio de las imágenes y gráficos. (2)

1.2.2. Drupal

Es un CMS modular y muy configurable. Drupal es un programa de código abierto, con licencia GNU/GPL y escrito en PHP. Los temas en Drupal definen el aspecto visual de la web, incluyendo los colores, los tipos de letra y la ubicación del contenido mostrado en las páginas, entre otros detalles. En la práctica son un conjunto de ficheros que se encuentran en el directorio temas del servidor, y que incluyen plantillas con la estructura del sitio, hojas de estilo e imágenes, con el fin de conseguir una mayor separación entre el contenido, el control y la presentación. (3)

Una de las desventajas de Drupal es que el núcleo no está bien documentado, es decir, si existe abundante documentación online, pero esta adolece de muchas lagunas y funciones indocumentadas. Además, presenta una extrema lentitud, y la escalabilidad y el rendimiento dependen del uso de datos cacheados. Otro aspecto negativo es la necesidad de hacerlo todo respetando la arquitectura de Drupal; se debe seguir un procedimiento concreto, indicando qué se va a editar, declarando las funciones oportunas que se deben haber consultado previamente, para finalmente escribir una plantilla con el cambio deseado. (4)

1.2.3. Wordpress

Es una avanzada plataforma semántica de publicación personal orientada a la estética, los estándares web y la usabilidad. Este CMS permite que el usuario pueda cambiar el aspecto de su blog con solo pulsar un botón pues cuenta con un centenar de temas, los cuales se van incrementando en correspondencia con la necesidad de los usuarios. (5)

Este inteligente sistema permite tener todo el proceso bajo control, organizar adecuadamente la estructura y lograr una mayor versatilidad. Al estar todo mejor estructurado, las personalizaciones son sencillas y permiten tener un mayor control de los cambios. Se pueden utilizar ficheros de unos temas en otros, solo cambiando los estilos y clases CSS (del inglés *Cascading Style Sheets*), con los que se ahorra tiempo y se pueden reutilizar. (6)

1.2.4. ZERA:

La plataforma ZERA está dividida por dos módulos: Administración e Hiperentornos. Este último está integrado por temas e incluye un sistema de administración de plantillas que automatiza el trabajo de estas en la plataforma. Además, brinda la posibilidad de subir un tema y poder visualizar en la página principal del administrador de plantillas algunos detalles, como por ejemplo, una vista del tema, el nombre, una breve descripción y la fecha en la que fue subido a la plataforma. (7)

ZERA está muy bien organizada estructuralmente, donde cada módulo contiene una carpeta donde están archivados sus estilos, imágenes y código HTML, y otra donde se encuentran cada uno de los elementos comunes pertenecientes a todos los módulos; de este modo los estilos no se repiten (7).

A continuación se muestra una tabla donde se comparan las aplicaciones antes descritas teniendo en cuenta diversos aspectos necesarios para el desarrollo del componente en cuestión.

Tabla 2: Comparación de sistemas que gestionan plantillas

Sistemas	Características	Ventajas	Desventajas
Drupal	- El sistema de temas separa el contenido de la presentación permitiendo controlar o cambiar fácilmente el aspecto del sitio web. - Se pueden crear plantillas con HTML y/o con PHP. - Ha sido diseñado desde el principio para ser multiplataforma. Por otro lado, al estar implementado en PHP, es totalmente portable. - Los temas son bastante	- Fácil configuración de temas y una gran cantidad de plantillas (aunque todas muy parecidas). - Permite la compatibilidad hacia atrás con versiones previas de plantillas y otras extensiones.	- El mayor problema con las plantillas es, que los desarrolladores olvidaron completamente una de las claves que cualquier evaluador mira: el sistema de menú. Si se tiene un sistema profesional de menú en la web, parecerá que es de baja calidad. - En cuestiones de accesibilidad, Drupal parece llevarse todas las

	flexibles. Por ejemplo se puede declarar una vista diferente para un tipo de bloque en específico. Con esto se tiene más flexibilidad a la hora del diseño.		palmas dada su flexibilidad para desarrollar las plantillas y su uso intensivo de CSS y XHTML, la responsabilidad encaja del lado del administrador.
Joomla	- Es posible cambiar todo el aspecto del sitio web tan solo con un par de clics, gracias al sistema gestor de plantillas (*templates*) que utiliza. - Las plantillas son totalmente configurables, incluyendo los bloques de izquierda, centro y derecha. - Previsualizador de plantillas. Vea cómo se ve antes de publicarla.	- Desarrollar el diseño de una página es increíblemente sencillo, debido a que ofrece plantillas prediseñadas. Estas plantillas son editables. - Posee una buena cantidad de plantillas gratis. - Separa la presentación del contenido y marca semánticamente los documentos. - Permite separar el HTML del PHP en sus plantillas, lo cual agrega sencillez a la hora de recurrir a recursos	- Las plantillas suelen requerir pequeñas adaptaciones para acomodarlas a las necesidades del usuario. Si se cambia de plantilla se pierden las adaptaciones. - Existen plantillas muy buenas, pero también otras mal programadas. -Tiene una dependencia excesiva del JavaScript en su Panel de Administración. - Es un poco lento.

		foráneos.	
Wordpress	- No es necesario reconstruir todas sus páginas cada vez que actualiza su bitácora, o cambia algún detalle de la misma. Todas las páginas son generadas al utilizar la base de datos y las plantillas cada vez que su bitácora es solicitada por un visor. Esto significa que actualizar su bitácora, o su diseño es tan rápido como sea posible, y el espacio de almacenamiento requerido en el servidor es mínimo. - Separa el contenido y el diseño en XHTML y CSS. - Usa plantillas para generar las páginas dinámicamente. Puede controlar la presentación del contenido usando la herramienta Editor de Plantilla y las etiquetas de plantilla. - Cada instalación de wordpress viene con un editor que puede utilizar para editar	- Buena calidad: los prediseños o plantillas de wordpress han sido testados y diseñados por los mejores diseñadores del mundo, resultando en diseños de calidad y funcionalidad garantizada. - Disponibilidad de decenas de Plantillas listas para utilizar. - Dispone de un sistema de plantillas muy cómodo, completo e intuitivo. Cada bloque de información es separado por una plantilla diferente permitiendo modificar fácilmente cualquier parte de la web. - Este CMS dispone de muchos estilos CSS listos para utilizar en	- Algunas de las plantillas disponibles son de pago. -Muchos de los temas tienen tendencia similar o están repetidos. - Desde la plataforma de wordpress, la personalización de una plantilla nunca ha sido tan fácil.

	sus plantillas, de forma similar a como se usan los navegadores sin tener que preocuparse por la carga y descarga de archivos para editarlos. - Puede cambiar la apariencia de una bitácora utilizando los temas y estilos ya disponibles. También puede crear y compartir sus propios temas.	varios elementos.	
ZERA	-Posee un administrador de plantillas que permite la automatización del trabajo con los temas en la plataforma. - Brinda la posibilidad de subir temas y visualiza una vista del mismo con el nombre, descripción y la fecha en la que fue subido a la plataforma.	- Presenta una mejor organización, calidad y rapidez. - Tiene gran usabilidad y accesibilidad.	- Solo brinda la posibilidad de subir plantillas, y una vez subidas no se pueden modificar.

Después del análisis y la comparación realizados a estos sistemas se propone utilizar algunos elementos de ellos, como la estructura organizativa de ZERA y la posibilidad de modificar los temas como Joomla; pero en este caso solamente de imágenes y colores, ya que el resto del diseño de las plantillas de la Colección “El Navegante” son estándar para todos los productos. Se aplicará la ventaja que implementan todas las aplicaciones antes descritas de tener el contenido independiente y el diseño por otro, lo que permite modificaciones del diseño sin necesidad de alterar los contenidos.

1.3. Tecnologías, lenguajes y herramientas a utilizar

A continuación se describen las tecnologías y herramientas que se utilizarán para el desarrollo del gestor de apariencias, las mismas están definidas por la arquitectura del proyecto "Colecciones de software educativos Multisaber y El Navegante".

1.3.1. Metodología de desarrollo

La metodología a emplear para el componente a desarrollar va a ser Proceso Unificado de Desarrollo (del inglés *Rational Unified Process*, en lo adelante RUP) ya que está definida en la Colección "El Navegante", donde toda la información del software está bajo las instrucciones de la misma, precisamente la selección de esta metodología es la que hace posible el desarrollo de esta aplicación debido a las ventajas significativas que presenta en el desarrollo de la vida del software. (8)

Rational Unified Process (RUP)

RUP es una metodología del proceso de ingeniería de software que proporciona un enfoque disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo.

Esta metodología, basada en UML, proporciona todas las bases para llevar al éxito la elaboración del software. RUP proporciona disciplinas que incluyen artefactos que constituyen guías para documentar e implementar fácil y eficientemente todo el desarrollo del software. De manera general se encuentra organizado en fases que están relacionadas con determinados flujos de trabajo (Anexo 1).

RUP presenta tres características esenciales que la definen como una metodología eficiente:

- ✓ Dirigido por casos de uso.
- ✓ Iterativo e incremental.
- ✓ Centrado en la arquitectura

1.3.2. Lenguaje de modelado

Unified Modeling Language (UML)

Lenguaje Unificado de Modelado (del inglés *Unified Modeling Language*, en lo adelante UML) es un lenguaje de propósito general para el modelado orientado a objetos, el cual combina diferentes

notaciones. Se prevé varias perspectivas de UML, ya que por ser un lenguaje de propósito general será un lenguaje de modelado orientado a objetos estándar predominante los próximos años, esto se basa en las siguientes razones: (8)

- ✓ Participación de metodólogo influyentes.
- ✓ Participación de importantes empresas.
- ✓ Aceptación del OMG como notación estándar.

Un diagrama es una representación gráfica de una colección de elementos de modelado, a menudo dibujada como un grafo con vértices conectados por arcos. (8)

UML, como parte de su modelo, está constituido por determinados diagramas (Anexo 2) que se describen a continuación: (8)

- ✓ Diagrama de casos de uso: modela la funcionalidad del sistema agrupándola en descripciones de acciones ejecutadas por un sistema para obtener un resultado.
- ✓ Diagrama de clases: muestra las clases (descripciones de objetos que comparten características comunes) que componen el sistema y cómo se relacionan entre sí.
- ✓ Diagrama de objetos: muestra una serie de objetos (instancias de las clases) y sus relaciones.
- ✓ Diagramas de comportamiento: este grupo está compuesto por determinados diagramas. Se describen a continuación:
    - Diagrama de estados: modela el comportamiento del sistema de acuerdo con eventos.
    - Diagrama de actividades: simplifica el Diagrama de Estados modelando el comportamiento mediante flujos de actividades.
    - Diagramas de interacción: Estos diagramas a su vez se dividen en dos tipos de diagramas, según la interacción que enfatizan:
    - Diagrama de secuencia: enfatiza la interacción entre los objetos y los mensajes que intercambian entre sí junto con el orden temporal de los mismos.

- Diagrama de colaboración: igualmente, muestra la interacción entre los objetos resaltando la organización estructural de los objetos en lugar del orden de los mensajes intercambiados.

✓ Diagramas de implementación: este grupo está compuesto por determinados diagramas. Se describen a continuación:

- Diagrama de Componentes: muestra la organización y las dependencias entre un conjunto de componentes.
- Diagrama de Despliegue: muestra los dispositivos que se encuentran en un sistema y su distribución en el mismo.

1.3.3. Herramienta CASE

Las herramientas de Ingeniería de Software Asistida por Computación CASE (del inglés *Computer Aided Software Engineering*, en lo adelante CASE) son un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores durante todos los pasos del ciclo de vida de desarrollo de un software. (9)

Visual Paradigm

Visual Paradigm para UML es una herramienta CASE que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El Visual Paradigm es capaz de exportar los diagramas de un modelo a otro con mucha facilidad. Además, permite modelar todos los tipos de diagramas de clases, así como generar código inverso, código desde diagramas y la documentación correspondiente. (10)

1.3.4. Lenguajes de Programación

En el desarrollo de aplicaciones web los lenguajes se dividen en dos grandes grupos: lenguajes del lado del servidor y lenguajes del lado del cliente.

✓ Los lenguajes del lado del servidor: son aquellos que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él.

- ✓ Los lenguajes del lado del cliente: son aquellos que pueden ser directamente "digeridos" por el navegador y no necesitan un pre-tratamiento.

Lenguaje de programación del lado del cliente

JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Además, es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos, en otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. La integración de JavaScript y XHTML es muy flexible, ya que existen al menos tres formas para incluir código JavaScript en las páginas web. (11)

Este lenguaje de programación presenta determinadas características que lo identifican: (12)

- ✓ Es simple, es decir, no hace falta tener conocimientos de programación para poder hacer un programa en JavaScript.
- ✓ Maneja objetos dentro de una página web y sobre ese objeto se pueden definir diferentes eventos. Estos objetos facilitan la programación de páginas interactivas, a la vez que se evita la posibilidad de ejecutar comandos que puedan ser peligrosos para la máquina del usuario, tales como formateo de unidades, modificar archivos, entre otros.
- ✓ Es dinámico, ya que responde a eventos en tiempo real, como presionar un botón, pasar el puntero del mouse sobre un determinado texto o el simple hecho de cargar la página en un tiempo. Con esto se puede cambiar totalmente el aspecto de una página al gusto del usuario, lo que evita tener en el servidor una página para cada gusto, hacer cálculos en base a variables cuyo valor es determinado por el usuario, entre otros.

Lenguaje de programación del lado del servidor

PHP 5

PHP (del inglés *Hypertext Pre-processor*) es un lenguaje script (no se compila para conseguir códigos máquina si no que existe un intérprete que lee el código y se encarga de ejecutar las instrucciones que contiene éste código), para el desarrollo de páginas web dinámicas del lado del servidor, cuyos

fragmentos de código se intercalan fácilmente en páginas HTML; debido a esto, además de ser de *Open Source* (código abierto), es considerado el más popular y extendido en la web. (13)

PHP es capaz de realizar determinadas acciones de una forma fácil y eficaz sin tener que generar programas implementados en un lenguaje distinto al HTML; esto se debe a que PHP ofrece un extenso conjunto de funciones para la explotación de bases de datos sin complicaciones. (13)

El principal objetivo de PHP 5 ha sido mejorar los mecanismos de Programación Orientada a Objetos (POO) para solucionar las carencias de las anteriores versiones. Este avance constituye un paso necesario para conseguir que PHP sea un lenguaje apto para todo tipo de aplicaciones y entornos, incluso los más exigentes.

PHP 5 presenta determinadas características que lo distingue del resto de los lenguajes de programación: ***(14)***

- ✓ Muy fácil de aprender.
- ✓ Es un lenguaje muy rápido.
- ✓ Soporta en cierta medida la orientación a objeto: clases y herencia.
- ✓ Es un lenguaje multiplataforma: Linux, Windows, entre otros.
- ✓ Presenta capacidad de conexión con la mayoría de los manejadores de base de datos: MySQL, PostgreSQL, Oracle, MS SQL Server, entre otras.
- ✓ Tiene capacidad de expandir su potencial utilizando módulos.
- ✓ Posee documentación en su página oficial, la cual incluye descripción y ejemplos de cada una de sus funciones.
- ✓ Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- ✓ Incluye gran cantidad de funciones.
- ✓ No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

1.3.5. Framework

Una plataforma de desarrollo (del inglés *Framework*) provee la infraestructura para el desarrollo de aplicaciones. De la misma forma que los cimientos y la estructura que sobre ellos se construye facilitan la construcción de edificios, las plataformas de desarrollo brindan un esqueleto que puede ser complementado con las partes específicas de una aplicación. (14)

Un framework es un esquema (esqueleto o patrón) para el desarrollo y/o la implementación de una aplicación. A continuación se muestran las ventajas que brinda el uso de un framework, las que se derivan de utilizar un estándar: (15)

- ✓ El programador no necesita plantearse una estructura global de la aplicación, sino que el framework le proporciona un esqueleto que hay que "rellenar".
- ✓ Facilita la colaboración. Cualquiera que haya tenido que enfrentarse con el código fuente de otro programador sabrá lo difícil que es entenderlo y modificarlo; por tanto, todo lo que sea definir y estandarizar va a ahorrar tiempo y trabajo a los desarrollos colaborativos.
- ✓ Es más fácil encontrar herramientas (utilidades, librerías) adaptadas al framework concreto para facilitar el desarrollo.

Existen dos tipos de framework: del lado del cliente y del lado del servidor. A continuación se explican cada uno de ellos.

Framework del lado del cliente

En la actualidad han surgido varias alternativas para el uso de aplicaciones AJAX para lograr la creación de aplicaciones web dinámicas del lado del cliente, ya que incorporan varios efectos visuales para obtener un mejor uso por parte del cliente. Estas alternativas han sido resultado del notable crecimiento de desarrollo basado en tecnología web a nivel mundial. (16)

En estos momentos existen varias librerías que cumplen con los requerimientos de usabilidad que el usuario necesita para lograr crear aplicaciones web dinámicas del lado del cliente, incorporando varios efectos visuales, entre estos se encuentran JQuery, Dojo y ExtJs, entre otros. (16)

JQuery 1.5

Este framework JavaScript ofrece una infraestructura con la que se tendrá mayor facilidad para la creación de aplicaciones complejas del lado del cliente. Con JQuery 1.5 se obtiene ayuda en la creación de interfaces de usuario, efectos dinámicos, aplicaciones que hacen uso de Ajax, etc. Cuando se programa JavaScript con JQuery tiene disponible una interfaz para programación que permitirá hacer cosas con el navegador que funcionará para todos los visitantes. Simplemente se deben conocer las librerías del framework y programar utilizando las clases, sus propiedades y métodos para la consecución de los objetivos trazados. (17)

Lo más importante de JQuery 1.5 es el aumento del rendimiento, ya que poco a poco JQuery se va transformando en una librería más potente y más ligera, lo que la convierte en ideal para todo tipo de proyectos web.

JQuery es un producto con una aceptación por parte de los programadores muy buena y un grado de penetración en el mercado muy amplio, lo que hace suponer que es una de las mejores opciones. Además, es un producto serio, estable y bien documentado.

Framework del lado del servidor

Del lado del servidor han surgido numerosos framework paralelamente al crecimiento existente de los framework del lado del cliente, entre los cuales se pueden mencionar: Symfony, CakePHP y CodeIgniter.

Anteriormente se especificó como lenguaje de desarrollo del lado del servidor a PHP, por las características que este presenta. Esta decisión tiene como resultado el uso de un framework implementado sobre el lenguaje PHP, para lograr mayor agilidad en el proceso de desarrollo.

Symfony 1.4

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Primeramente se puede asegurar que separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. También proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web. (18)

Este framework de desarrollo aumenta exponencialmente la productividad y ayuda a mejorar la calidad de las aplicaciones web aplicando todas las buenas prácticas y patrones de diseño que se han definido para la web. Cuenta además con una amplia documentación, miles de páginas, una amplia distribución de libros de forma gratuita y decenas de tutoriales.

Las características más comunes para el desarrollo de proyectos web están automatizadas en Symfony, tales como: (18)

- ✓ Fácil de instalar y configurar en la mayoría de plataformas.
- ✓ Independiente del sistema gestor de bases de datos.
- ✓ Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- ✓ Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
- ✓ Sigue la mayoría de las mejores prácticas y patrones de diseño para la web.
- ✓ Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- ✓ Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- ✓ Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

De manera general, las características fundamentales que se tuvieron en cuenta para la selección de este framework son que posee una amplia documentación y que es estable, ya que se le han aplicado una gran cantidad de pruebas funcionales y unitarias.

1.3.6. Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado (en inglés *Integrated Development Environment*, en lo adelante IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar

código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o puede dar cabida a varios de estos. (19)

NetBeans 7.0

NetBeans es un IDE que permite crear aplicaciones no solo en el lenguaje de programación Java, sino también en una gran variedad de lenguajes en los cuales se está creando el soporte. (20)

A continuación se muestran las nuevas características de NetBeans 7.0: (21)

- ✓ JDK 7
    - Soporte para el Project Coin.
    - Mejoras en el editor: autocompletado de código, pistas.
- ✓ WebLogic Server
    - Mejorada la velocidad y eficiencia al distribuir contenido a WebLogic.
    - Lista de aplicaciones y recursos en servidor consultable en tiempo real.
    - Integración con JSF y librerías del servidor.
- ✓ Oracle Database
    - Asistente de conexión simplificado.
    - Instalación guiada para el driver de JDBC.
    - Los procedimientos almacenados ya se pueden editar e instalar directamente.
- ✓ Java
    - Soporte para Maven 3.
    - Integración con JUnit 4.8.2 y varias mejoras en JUnit
    - En Javadoc ya se soportan las URLs remotas.
    - Mejorado el configurador para GridBagLayout.
- ✓ Lenguajes Web

- Soporte para edición en HTML5.
- Formateador de JSON.

✓ PHP

- Generador de PHPDoc.
- Refactoring de renombrado y de borrado seguro.
- PHP 5.3 – Soporte de aliases.
- C/C++
- Importación de proyectos usando los binarios existentes.
- Nuevo tipo de proyecto cuando los ficheros fuente están en un sistema remoto.

✓ General

- Wordwrap (ajuste automático de líneas) en el editor.
- Mejorada la integración del Profiler.

La comprobación de cambios externos en ficheros ahora es menos intrusiva cuando se cambia de tarea entre el IDE y otros programas.

1.4. Conclusiones

Luego de haber realizado un análisis de algunas de las aplicaciones existentes que realizan la gestión de temas o plantillas se pudo adquirir conocimientos sobre este tema y sus tendencias actuales, dichas aplicaciones simulan un proceso similar al que se desea desarrollar, por lo que se pudieron estudiar los aspectos positivos y negativos de las mismas para tomar lo mejor de ellas y tener en cuenta sus desventajas para tratar de mitigarlas.

En este capítulo se expusieron las principales características de las tecnologías, herramientas y lenguajes definidas por el equipo de dirección del proyecto en el producto al que se va a integrar el componente. Dicha selección aporta todo lo necesario para la creación del gestor de apariencias, teniendo en cuenta que sea una solución factible para el cliente.

Capítulo 2: Características del sistema

2.1. Introducción

En el presente capítulo se realiza una propuesta del sistema, a través de un modelo de dominio; en el que se describen los principales conceptos provenientes del negocio del componente a desarrollar. Se definen los requisitos funcionales y no funcionales, donde se exponen las características y cualidades que el sistema debe cumplir. Y basado en esos requerimientos se explica mediante los CU, la descripción de cada uno de ellos y el diagrama de CU del sistema (en el que se analizan los actores involucrados y las relaciones con los CU); la propuesta de solución del componente a desarrollar.

2.2. Modelo de dominio

En el estudio realizado al problema que se plantea, se determinó que con las condiciones existentes no es factible realizar un modelo de negocio pues sería necesario describir los procesos de negocio en términos de CU del negocio y actores del negocio que se correspondan con los procesos del negocio y los clientes, respectivamente. Por ello se hace necesario describir el problema de otra manera, realizándose así un modelo de dominio con el propósito de comprender y describir las clases más importantes dentro del contexto del sistema.

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las “cosas” que existen o los eventos que suceden en el entorno en el que trabaja el sistema. (22)

2.2.1. Análisis de los conceptos del dominio

Para lograr una mejor comprensión del diagrama de Modelo de Dominio es necesario realizar una breve descripción de las clases involucradas en dicho modelo:

- ✓ **Administrador:** Realiza las tareas de administración en la colección. Administra las plantillas a través del subsistema de gestión de plantillas.

- ✓ **Producto Multimedia:** Producto que combina diversos tipos de medias (textos, sonidos, imágenes o videos).
- ✓ **Plantilla:** Conjunto de archivos que contiene los estilos de una apariencia gráfica, incluye el diseño de fondo y combinaciones de colores de relleno e imágenes.

2.2.2. Diagrama del modelo de dominio

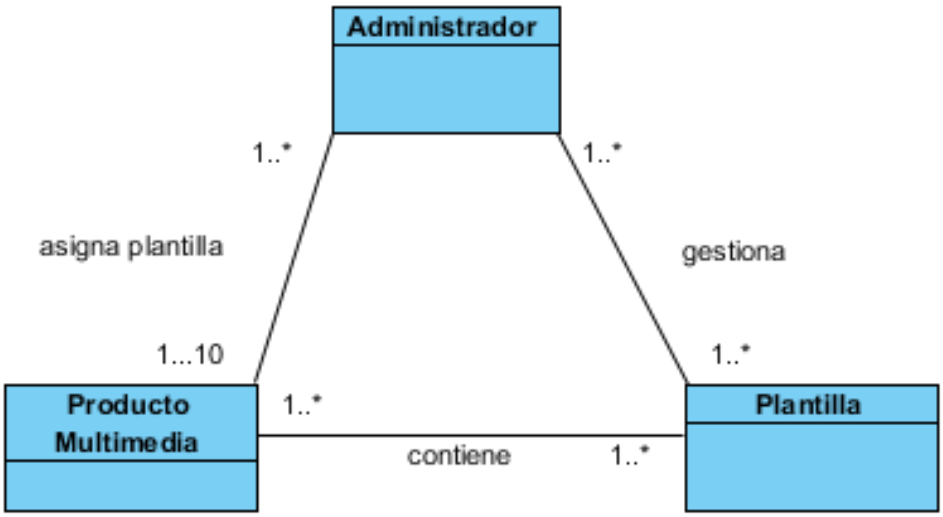


Figura 1: Modelo de dominio

2.3. Descripción del sistema propuesto

El componente Gestor de apariencia permite un manejo más eficaz de plantillas para cada producto, presentando una reestructuración de estilos e imágenes. Teniendo en cuenta la reutilización de los estilos, todos los elementos que tengan estilos redundantes se reagruparán y se reasignarán a los elementos con selectores de clase genéricos. De igual forma, todas las imágenes presentes en la colección tendrán un patrón de tamaño estructurado y las que estén repetidas, serán eliminadas. El sistema de carpetas donde se encuentran tanto los estilos e imágenes, se reagruparán en una serie de carpetas (CSS) en las cuales estarán los estilos del tema, y cada una con una carpeta (Imágenes) para las imágenes del tema. Esta organización agilizará la gestión de plantillas, obteniendo así una arquitectura homogénea de estilos y de ficheros a la hora de manipular la apariencia de cualquier producto de la colección, optimizando esta tarea en tiempo y trabajo.

La propuesta del componente Gestor de apariencia tendrá dos secciones. La primera sección, “Asignación de plantilla”, presenta una serie de plantillas existentes para aplicarlas al producto deseado. La segunda sección, “Gestión de plantilla”, permitirá realizar una serie de cambios a dichas plantillas, incluyendo la creación de una nueva .Estos cambios se dividen en cuatro acciones:

- ✓ Crear plantilla: se creará una plantilla con características (color, imágenes) deseadas por el usuario, las cuales podrá observar en el instante como las va adquiriendo, esto de hace siempre siguiendo las pautas de diseño definidas por el componente.
- ✓ Editar plantilla: se cambiarán las características (color, imágenes) de una plantilla existente, igualmente en ese instante el usuario podrá visualizar las modificaciones que se le estén realizando.
- ✓ Ver plantilla le permitirá al usuario seleccionar una de las plantillas que existen y visualizarla a un tamaño estándar asequible.
- ✓ Eliminar plantilla: se eliminará la plantilla no deseada u obsoleta.

Dentro de esta segunda sección también se podrán visualizar las plantillas, tanto editadas como creadas, para que el usuario tenga noción de la apariencia que le dará a su producto.

2.4. Requerimientos del software

Un proceso de desarrollo de software no puede ser exitoso sin una descripción detallada, correcta y exhaustiva de los requerimientos, ya que es necesario saber qué debe hacer dicho sistema, además de tener bien claro cuáles son las expectativas del cliente con respecto al sistema a desarrollar. (23)

El propósito fundamental de la captura de los requisitos es guiar el desarrollo hacia el sistema correcto. En esta sección se especifican los requisitos funcionales y no funcionales definidos para satisfacer las necesidades del cliente.

2.4.1. Requerimientos Funcionales

Estos requisitos son las capacidades o funciones que el sistema debe cumplir. A continuación se plantean los mismos:

RF1 Mostrar plantillas existentes.

RF2 Asignar plantilla.

2.1 Permitir seleccionar una plantilla para aplicar a un producto determinado.

2.2 Aplicar la plantilla seleccionada al producto deseado.

RF3 Gestionar plantillas.

3.1 Mostrar opción para la creación de una nueva plantilla.

3.1.1 Permitir seleccionar opción para la creación de plantillas.

3.1.2 Mostrar ventana y el campo de texto para introducir el nombre de la plantilla.

3.1.3 Guardar el nombre de la plantilla.

3.1.4 Mostrar ventana de creación de plantillas.

3.1.5 Mostrar opciones para la selección de colores e imágenes.

3.1.6 Permitir seleccionar color para determinadas propiedades de la plantilla.

3.1.7 Permitir agregar imágenes a la plantilla según las pautas establecidas.

3.1.8 Permitir guardar la plantilla creada.

3.2 Mostrar opción de Gestionar plantilla seleccionada.

3.2.1 Permitir seleccionar opción para la gestión de la plantilla seleccionada.

3.2.2 Mostrar ventana con el nombre de la plantilla que se está gestionando.

3.2.3 Mostrar opción para la edición de plantilla.

3.2.4 Permitir seleccionar la opción para la edición de plantillas.

3.2.5 Mostrar opciones para la selección de colores e imágenes.

3.2.6 Permitir seleccionar color para determinadas propiedades de una plantilla.

3.2.7 Permitir agregar imágenes a la plantilla según las pautas establecidas.

3.2.8 Permitir guardar la plantilla editada.

3.2.9 Mostrar opción eliminar.

3.2.10 Permitir seleccionar la opción para eliminar la plantilla.

3.2.11 Permitir eliminar plantilla.

RF4 Mostrar vista previa de la plantilla.

2.4.2. Requerimientos No Funcionales

Son las propiedades o cualidades que el sistema debe tener para que sea más atractivo, usable, rápido y confiable. Estos representan las características del producto.

Seguidamente se muestran los requisitos no funcionales definidos por el proyecto, los cuales se pueden encontrar en el documento *"Especificación de requisitos de El Navegante"*(actualmente este documento está en proceso de actualización).

RNF1: Requisitos de usabilidad.

La utilización del componente por parte de los usuarios sin avanzados conocimientos de computación, no requiere de previa preparación, debido al diseño sencillo y estándar que tiene. Además de que será escrita en lenguaje español lo más sencillo y asequible posible para facilitar su entendimiento por parte de los usuarios.

RNF 2: Requisitos de fiabilidad.

La colección no está diseñada para grandes redes, debido fundamentalmente a la carga de recursos multimedia que tiene. La dimensión de la red donde se use deberá estar en correspondencia con las capacidades técnicas de la misma, y de las prestaciones que ofrezcan los equipos que la componen, ya que la velocidad y la estabilidad de conexión de los clientes y las prestaciones del o los servidores pudieran provocar fallos en las estaciones clientes y demora en la ejecución de los productos de la colección.

RNF 3: Requisitos de eficiencia.

Para la determinación de la eficiencia deberá tenerse en cuenta la forma de uso de la colección, ya que podrá ser local o distribuida en la red.

- Uso local:

  Requerimientos mínimos de hardware: 256 MB RAM, 9 GB de HDD, 500 MHz.

Tiempo de respuesta de una página: no mayor a 30 segundos.

- Uso en red:

  Velocidad de conexión mínima: 10 Mbit/s.

  Requerimientos mínimos de hardware: 128 MB RAM, 9 GB de HDD, 500 Mhz.

  Cantidad máxima de máquinas conectadas: 20

  Velocidad de respuesta máxima: 5 minutos (sobre todo para el caso de las páginas que posean grandes cargas de recursos multimediales).

RNF 4: Requisitos de soporte.

Se realizará transferencia tecnológica de la colección de software. La transferencia se llevará a cabo mediante la entrega de documentación y código fuente de los diez productos, lo que permitirá que se le puedan realizar mejoras y actualizaciones del software.

RNF 5: Requisitos de restricciones de diseño e implementación.

Los productos se implementarán como una aplicación web, garantizándose su funcionamiento solamente sobre el navegador Mozilla Firefox versión 10.0 y sistema operativo Ubuntu en su versión 10.0.4 o superior, de forma local en cada máquina.

La facilidad de desarrollo con software libre es debido a la amplia gama de herramientas disponibles para el desarrollo de aplicaciones web. La resolución de pantalla a la que deben visualizarse los productos de la colección es 1024 x 600 píxeles. Los contenidos de la aplicación estarán soportados sobre una base de datos en PostgreSQL. El framework seleccionado para el desarrollo es Symfony 1.4.3, siendo la versión utilizada compatible solamente con PHP 5. El estilo arquitectónico a emplear será el Modelo-Vista-Controlador y para la implementación de las clases de la vista se usará el framework de JavaScript jQuery.

RNF 6: Requisitos de Interfaz.

El diseño de las interfaces debe ser amigable y sencilla.

RNF 6.1 Interfaces de usuario.

Las interfaces se desarrollarán teniéndose en cuenta elementos de las aplicaciones con tecnología multimedia diseñadas para escritorio y las aplicaciones web, logrando equilibrar la eficiencia del software sobre la Web con la interactividad y el diseño visual que exige la aplicación.

RNF 6.2 Interfaces de hardware.

El sistema podrá interactuar solamente con una interfaz de hardware: la impresora.

RNF 6.3 Interfaces de software.

El sistema no tiene relación con otras interfaces de software.

RNF 6.4 Interfaces de comunicación.

El sistema puede ser desplegado sobre red LAN, MAN, o WAM siempre y cuando la velocidad de conexión sea mayor a 10 Mbits.

RNF 7: Requisitos de licencia.

En el desarrollo del software no podrán utilizarse componentes que no hayan sido liberados bajo licencias de software libre, se exceptúa la gestión de recursos gráficos y multimediales que componen el software, los cuales si podrán ser adquiridos bajo licencias propietarias. Deberá constar la licencia con la que fue obtenido cada recurso y herramienta empleada en el desarrollo del software. La aplicación debe ser liberada bajo alguna de las licencias de software libre.

RNF 8: Requisitos legales, de derecho de autor y otros.

Aunque la aplicación sea liberada bajo alguna de las licencias de software libre, deben respetarse las licencias y condiciones legales bajo las que fueron adquiridos cada uno de los recursos multimediales, por lo que deben ser liberados bajo distintas licencias, propietarias o no. Las medias a utilizar deben tener el permiso legal de sus autores y su aprobación para hacer uso de ellas. Debe mostrarse de cada media la fuente de donde fue extraída, entiéndase, la dirección del sitio y la fecha de consulta para el caso de las medias libres descargadas de Internet, y el nombre del autor para las medias adquiridas por otros medios legales.

RNF 9: Requisitos de Estándares aplicables.

Durante el proceso de desarrollo de software cada entregable es sometido a pruebas de liberación, en las que se evalúan las características de calidad definidas por la norma ISO/IEC 9126. Para el diseño de la Base de Datos se usará un convenio para el nombre de las tablas, así como para las vistas y funciones que sean necesarias implementar. La codificación estará basada en el estándar recomendado en el sitio oficial de PHP y disponible en http://pear.php.net/manual/en/standards.php. Podrán realizarse las adecuaciones que se crean necesarias con previo acuerdo entre las partes. Los contenidos serán almacenados utilizando el sistema de metadatos definidos por el estándar LOM, pretendiéndose en desarrollos futuros incorporar soporte total al estándar SCORM.

RNF 10 Requisitos de portabilidad.

El componente podrá ser utilizado bajo cualquier sistema operativo.

RNF 11: Requisitos de software.

Computadora Personal con navegador Mozilla Firefox 10.0.

2.5. Definición de los casos de uso

Los CU son fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. También se definen como procesos que responden a las funcionalidades definidas en los requerimientos funcionales. (24)

2.5.1. Actores del sistema

Son las personas (u otros sistemas) asociadas al cumplimiento de los requerimientos funcionales.

Tabla 3: Actores del sistema

Actor	Descripción
Administrador	Tiene todos los permisos necesarios para seleccionar y gestionar las plantillas que se le van a asignar a los productos.

2.5.2. Diagramas de casos de uso del sistema

Un diagrama de CU del sistema representa gráficamente a los procesos y su interacción con los actores.

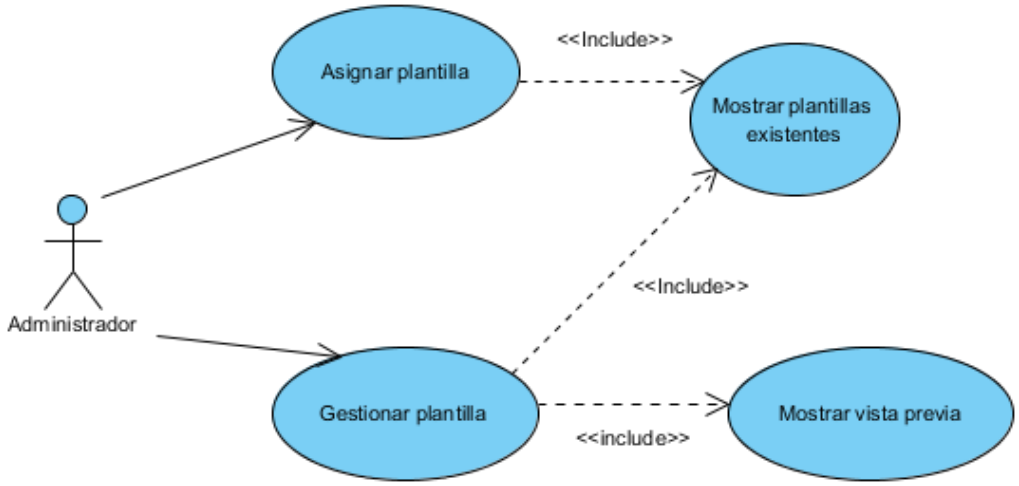


Figura 2: Diagrama de casos de uso del sistema

2.5.3. Descripción de casos de uso del sistema

A continuación se realiza la descripción de los CU presentes en el Diagrama de CU del sistema, donde se realiza un resumen de los mismos, se define su prioridad, la serie de actividades que se van a realizar por cada uno de ellos y sus acciones en caso de que no se realice el flujo básico. Se mostrará de manera simplificada, para verlo en su totalidad, remitirse al (Anexo 3).

Tabla 4: CU Mostrar plantillas existentes

Caso de Uso:	Mostrar plantillas existentes
Actores:	Administrador
Resumen:	El caso de uso inicia cuando el actor selecciona la opción gestionar plantillas o cuando selecciona la opción de editar producto donde aparece entre otras opciones la selección de plantillas. Luego el sistema le muestra el listado de plantillas existentes, finalizando así el caso de uso.

Tabla 5: CU Asignar plantilla

Caso de Uso:	Asignar plantilla
Actores:	Administrador
Resumen:	El caso de uso se inicia cuando el actor selecciona la opción de editar producto o la de crear producto donde aparece entre otras opciones la selección de plantillas. Posteriormente el sistema le dará la posibilidad al actor de seleccionar una de las plantillas existentes y aplicársela al producto, finalizando así el caso de uso.

Tabla 6: CU Gestionar plantilla

Caso de Uso:	Gestionar plantilla.
Actores:	Administrador
Resumen:	El caso de uso se inicia cuando el actor selecciona la opción de gestionar plantillas. El actor puede crear, editar, ver o eliminar las plantillas deseadas. En caso de que seleccione la opción crear, el sistema mostrará una ventana para introducir el nombre de la plantilla, posteriormente saldrá otra ventana que le brindará al usuario la posibilidad de seleccionar las características deseadas de la plantilla. En caso de que se seleccione la opción editar, el sistema mostrará una ventana con el nombre de la plantilla a editar y una serie de opciones para modificar la plantilla deseada. En el caso de ver y eliminar una plantilla, en la misma ventana del editar le brindará la posibilidad de realizar estas acciones, terminando así el caso de uso.

Tabla 7: CU Mostrar vista previa

Caso de Uso:	Mostrar vista previa
Actores:	Administrador
Resumen:	El caso de uso inicia cuando el actor está creando una plantilla o editando alguna de las plantillas existentes; donde el sistema le muestra al usuario en tiempo real los cambios efectuados en la apariencia de la plantilla, finalizando así el caso de uso.

2.6. Matriz de trazabilidad

La trazabilidad es el mecanismo que tiene como técnica separar y especificar correctamente los requisitos, controlar su evolución y soportar los cambios. Constituye un elemento de gran importancia ya que brinda mayor información para la comprensión del problema que se está tratando y apoya al control de las actividades y cambios en el software durante todo el ciclo de vida. (21)

En la matriz que aparece a continuación, un eje contiene los CU y otro los requerimientos, la trazabilidad relaciona cada requisito del sistema con su correspondiente CU.

CU1: Mostrar plantillas existentes.

CU2: Asignar plantilla.

CU3: Gestionar plantilla.

CU4: Mostrar vista previa.

Tabla 8: Matriz de trazabilidad

Requisitos\CU	CU1	CU2	CU3	CU4
1	x	x	x	
2	x	x		
2.1	x	x		
2.2		x		
3	x		x	

3.1	x		x	
3.1.1	x		x	
3.1.2			x	
3.1.3			x	
3.1.4			x	x
3.1.5			x	x
3.1.6			x	x
3.1.7			x	x
3.1.8			x	x
3.2	x		x	
3.2.1	x		x	
3.2.2			x	x
3.2.3			x	x
3.2.4			x	x
3.2.5			x	x
3.2.6			x	x
3.2.7			x	x
3.2.8			x	x
3.2.9	x		x	x
3.2.10	x		x	x
3.2.11	x		x	x
4			x	x

2.7. Conclusiones

Al concluir el presente capítulo se llegó a la conclusión de que mediante la representación del modelo de dominio, la elaboración de la propuesta de solución del sistema, la obtención de los requisitos funcionales y no funcionales se logró esclarecer el problema que se plantea, permitiendo una mejor visualización del mismo de acuerdo a las necesidades del cliente.

Toda la información expuesta y descrita en este capítulo, incluida en su mayoría en los artefactos obtenidos, permitió determinar la estructura y organización del componente a desarrollar, y de esta forma servir de apoyo para el análisis y el diseño a desarrollar posteriormente.

Capítulo 3: Análisis y Diseño

3.1. Introducción

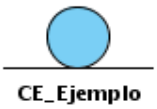
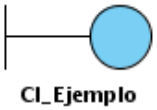
En este capítulo se elabora el análisis del sistema propuesto, donde se define el modelo del análisis y, a su vez, el diagrama de clases del análisis. Se elabora el diseño, donde se realizan los diagramas de interacción y los diagramas de clases del diseño. Se seleccionan y definen los patrones arquitectónicos y de diseño a emplear en el componente a desarrollar.

3.2. Modelo de Análisis

Durante el análisis, se analizan los requisitos que se describieron en la captura de requisitos, refinándolos y estructurándolos. Todas estas actividades se realizan para conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema incluyendo su arquitectura.

El modelo del análisis contiene clases del análisis y sus objetos organizados en paquetes que colaboran entre sí. Estas clases se centran en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos y relaciones del dominio. Estas clases contienen atributos y entre ellas se establecen relaciones. RUP propone clasificar a las clases en:

Tabla 9: Estereotipos de clases del análisis

Nombre	Características	Representación
Entidad	Modelan información que posee larga vida y que es a menudo persistente.	
Interfaz	Modelan la interacción entre el sistema y sus actores.	

Control	Coordinan la realización de uno o unos pocos CU coordinando las actividades de los objetos que implementan la funcionalidad del CU.	 CC_Ejemplo

3.2.1. Diagramas de clases del análisis

A continuación se muestran los diagramas de clase del análisis (DCA) para los CU del sistema:

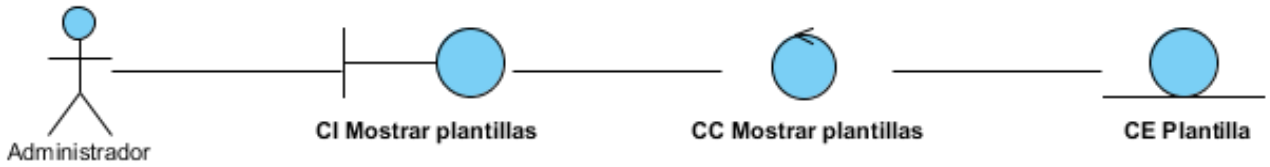


Figura 3: DCA Mostrar plantillas existentes

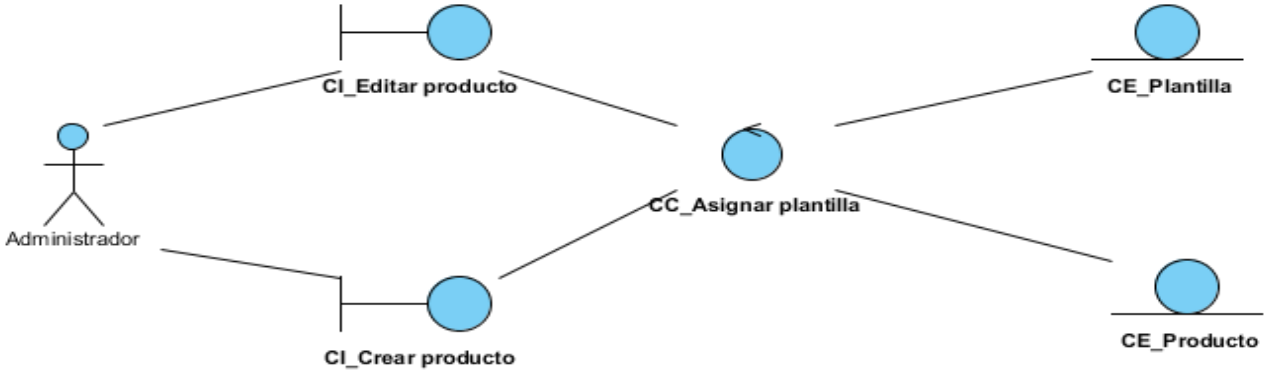


Figura 4: DCA Asignar plantilla

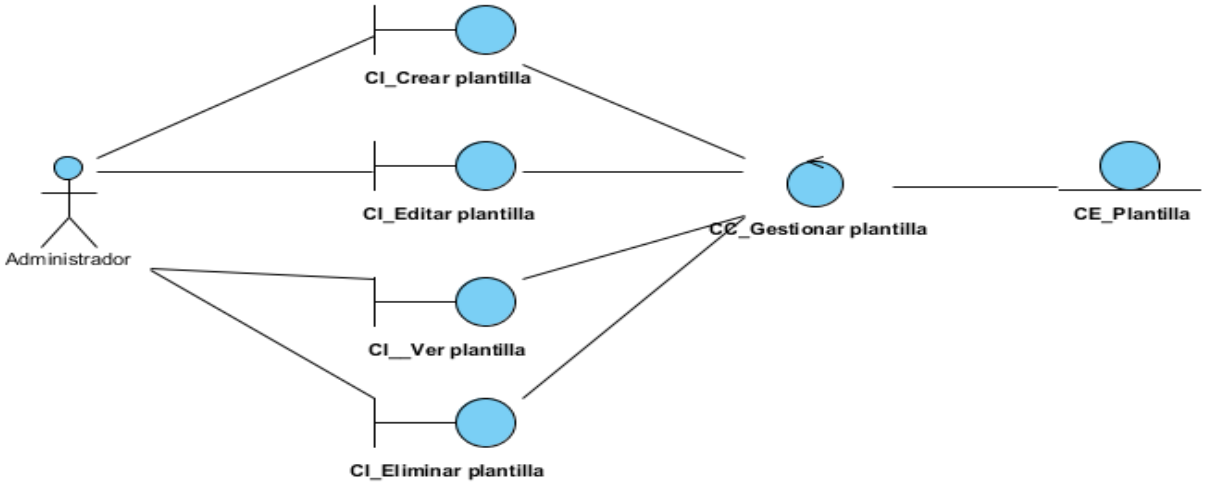


Figura5: DCA Gestionar plantilla

3.2.2. Diagramas de interacción

En estos diagramas se muestra un patrón de interacción entre objetos. Hay dos tipos de diagrama de interacción basados en la misma información, pero cada uno enfatizando un aspecto particular: Diagramas de Secuencia (dimensión temporal) y Diagramas de Colaboración (dimensión estructural). Ambos diagramas son isomorfos, es decir, se puede convertir de uno a otro sin pérdida de información. Los diagramas de interacción de manera general presentan determinadas características, a continuación se explican cada una de ellas:

Estos diagramas poseen ciertas características que lo identifican:

- ✓ Modelan el comportamiento dinámico del sistema; el flujo de control en una operación.
- ✓ Describe la interacción entre objetos; los objetos interactúan a través de mensajes para cumplir ciertas tareas.
- ✓ Las interacciones proveen un “comportamiento” y típicamente implementan un CU.

Diagramas de colaboración

Un diagrama de colaboración (en lo adelante DC) es una forma alternativa al diagrama de secuencia con el cual se muestra un escenario. Este tipo de diagrama muestra las interacciones entre objetos organizadas en torno a los objetos y los enlaces entre ellos. A continuación se muestran algunos de los

diagramas de colaboración correspondientes a la solución que se propone, donde se modela el comportamiento del sistema en función de los CU, el resto se pueden observar en el (Anexo 4):

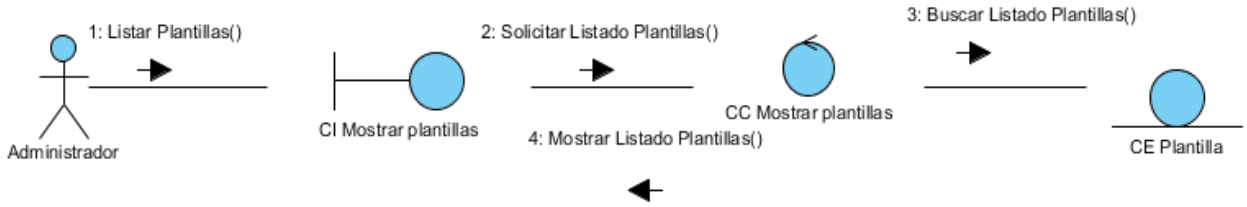


Figura 6: DC Mostrar plantillas existentes

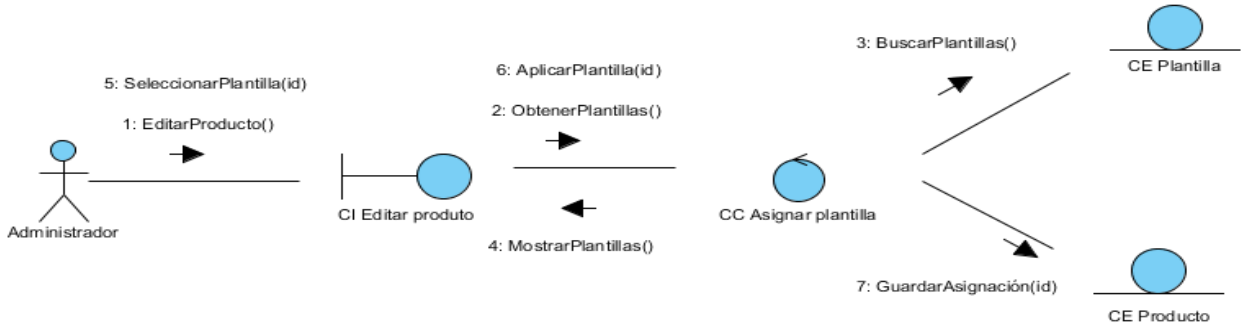


Figura 7: DC Asignar plantilla sección "Editar producto"

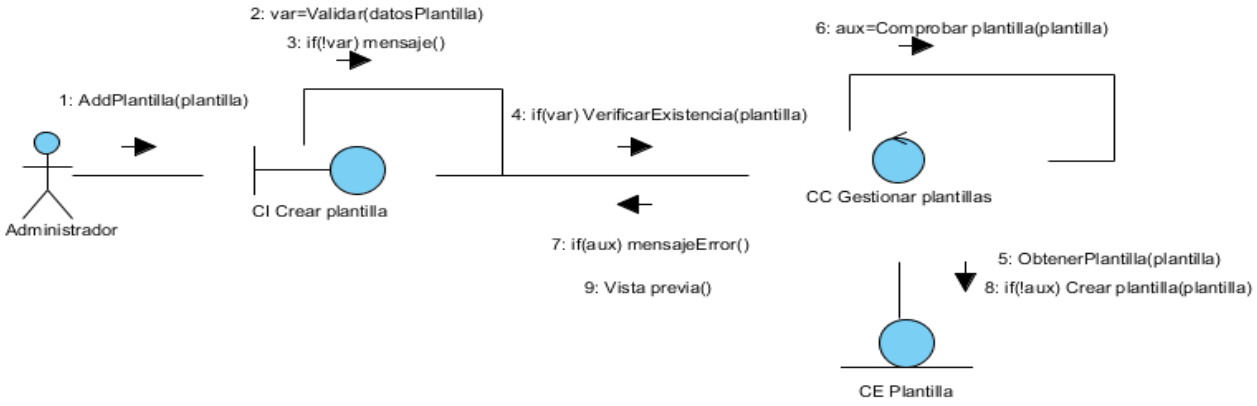


Figura 8: DC Gestionar plantilla sección "Crear plantilla"

Diagramas de secuencia

Los diagramas de secuencia (en lo adelante DS) son aquellos que destacan el orden temporal de los mensajes. A continuación se muestran algunos de los diagramas de secuencia correspondientes a la solución propuesta, el resto se puede encontrar en el (Anexo 5):

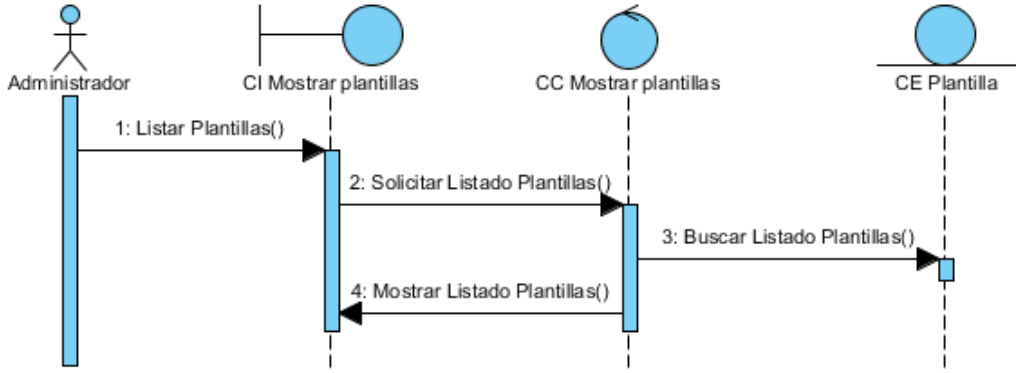


Figura 9: DS Mostrar plantillas existentes

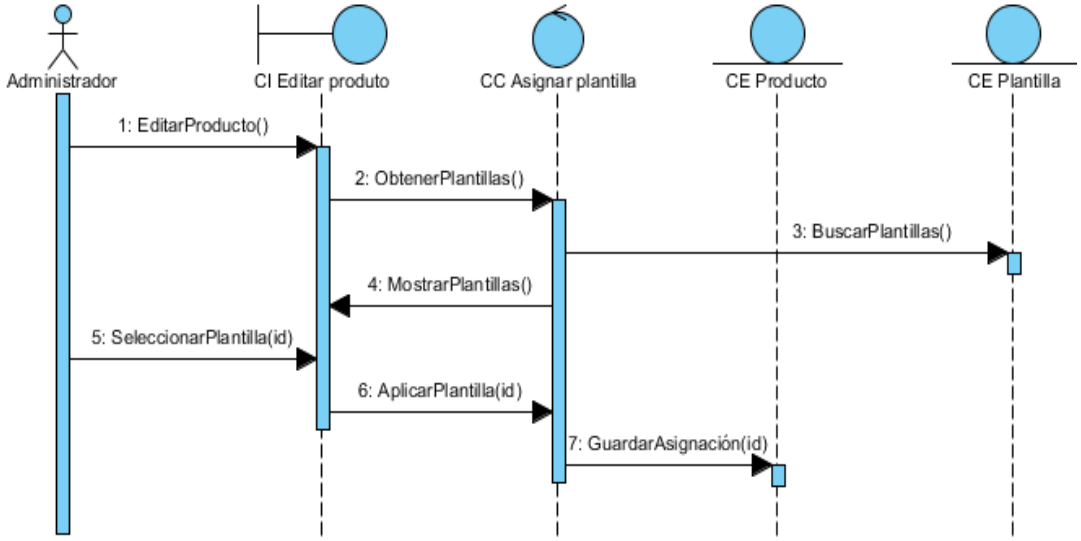


Figura 10: DS Asignar plantilla sección "Editar producto"

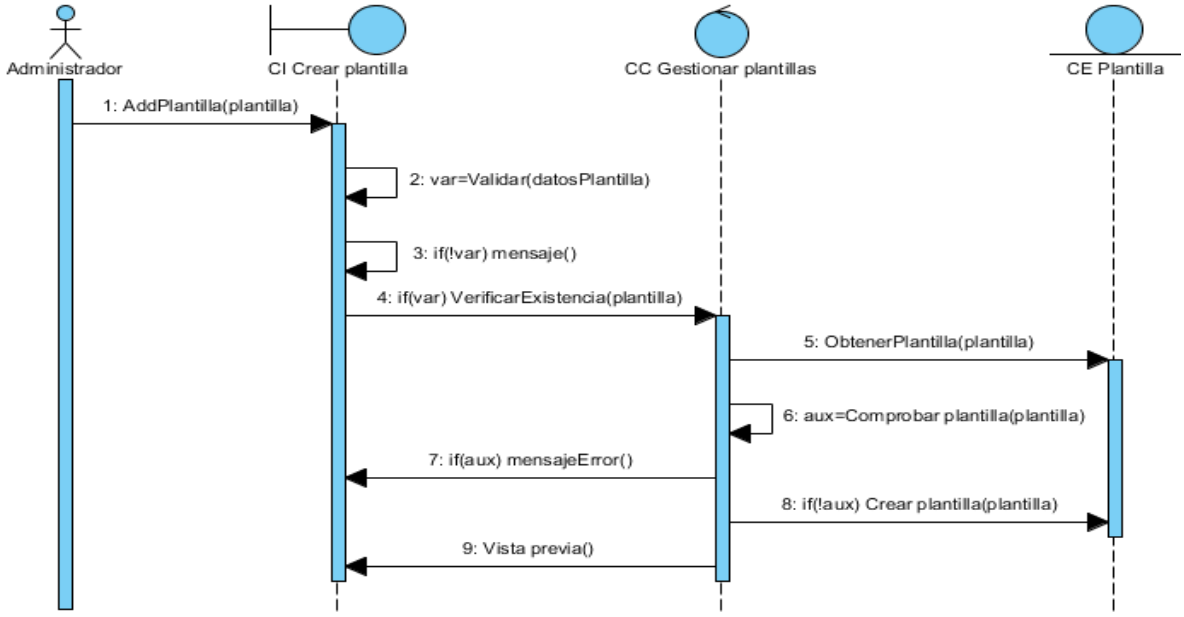


Figura 11: DS Gestionar plantilla sección "Crear plantilla"

3.3. Patrón arquitectónico

Symfony está basado en un patrón clásico del diseño web conocido como arquitectura Modelo-Vista-Controlador (en lo adelante MVC) (o en inglés, *model-view-controller*). MVC es un patrón de arquitectura de software encargado de separar la lógica de negocio de la interfaz del usuario y es el más utilizado en aplicaciones Web, ya que facilita la funcionalidad, mantenibilidad y escalabilidad del sistema, de forma simple y sencilla, a la vez que permite *"no mezclar lenguajes de programación en el mismo código".* (25)

MVC divide las aplicaciones en tres niveles de abstracción: (25)

- ✓ **Modelo**: representa la lógica de negocios. Es el encargado de conectar de forma directa a los datos con la base de datos, actuando como "intermediario".
- ✓ **Vista**: es la encargada de mostrar la información al usuario de forma gráfica y "humanamente legible".
- ✓ **Controlador:** es el intermediario entre la vista y el modelo. Es quien controla las interacciones del usuario solicitando los datos al modelo y entregándolos a la vista para que ésta lo presente al usuario.

Para elaborar una página sencilla son necesarios los siguientes componentes: (26)

- ✓ La capa del Modelo
    - Abstracción de la base de datos
    - Acceso a los datos
- ✓ La capa de la Vista
    - Vista
    - Plantilla
    - Layout
- ✓ La capa del Controlador
    - Controlador frontal
    - Acción

Symfony toma lo mejor de la arquitectura MVC y la implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo. Primeramente, el controlador frontal y el layout son comunes para todas las acciones de la aplicación. Se pueden tener varios controladores y varios layouts, pero solamente es obligatorio tener uno de cada uno. El controlador frontal es un componente que sólo tiene código relativo al MVC, por lo que no es necesario crear uno, ya que Symfony lo genera de forma automática. (23)

3.4. Patrones de Diseño

Un patrón es un par problema – solución que se puede aplicar en nuevos contextos, conteniendo consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos. Se puede afirmar que brindan soluciones a problemas recurrentes. (27)

Cada patrón describe un problema que ocurre una y otra vez en el entorno y describe también el núcleo de su solución, de forma que puede utilizarse un millón de veces sin hacer dos veces lo mismo. (Christoph Alexander). (28)

Los patrones de diseño a emplear son:

- ✓ **Patrones GRASP** (*General Responsibility Assignment Software Patterns*, en español Patrones de Principios Generales para Asignar Responsabilidades). (27)

Según Craig Larman: “Describen los principios fundamentales de diseño de objetos y la asignación de responsabilidades, expresados como patrones”. (27)

A continuación se describen los patrones GRASP utilizados:

Experto: Con la utilización de este patrón se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Se le debe asignar las responsabilidades a la clase que contenga la información necesaria para cumplir las tareas, o sea, la clase debe ser la experta en la información, alentando con ello definiciones de clases sencillas y más cohesivas, que son más fáciles de comprender y mantener.

Alta Cohesión: En cada clase Actions se definen las acciones para las plantillas, además estas colaboran con otras para realizar diferentes operaciones, o sea, en una Actions se utilizan diferentes funcionalidades estrechamente relacionadas entre sí, lo que proporciona un software flexible ante los cambios.

Controlador: Este patrón ayuda a definir quién será el responsable de atender un evento del sistema. Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente un controlador de fachada.

- ✓ **Patrones GOF** (*Gang of Four*, en español Banda de los Cuatro).

Los patrones GOF se dividen en tres grupos de patrones:

De creación: abstraen el proceso de creación de instancias.

Estructurales: se ocupan de cómo clases y objetos son utilizados para componer estructuras de mayor tamaño.

De comportamiento: atañen a los algoritmos y a la asignación de responsabilidades entre objetos.

Para la implementación de la solución propuesta se emplea el patrón:

Instancia única (en inglés *Singleton*): Es un patrón creacional que garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a ella. Se usa cuando debe haber exactamente una instancia de una clase y ésta debe ser accesible a los clientes desde un punto de acceso conocido. La única instancia debería ser extensible mediante herencia y los clientes deberían ser capaces de usar una instancia extendida sin modificar su código. Usa en las acciones disimiles métodos como getRequest () y getContext (), las cuales guardan una referencia a todos los objetos del núcleo de Symfony, estos métodos pueden ser accedidos desde la vista y desde el controlador, solo varía la forma de llamarlos.

3.5. Modelo de diseño

Es un modelo de objetos que describe la realización de los CU, centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Además, el modelo de diseño sirve de abstracción de la implementación y el código fuente.

En la siguiente tabla se muestran los estereotipos utilizados en los diagramas de clases del diseño:

Tabla 10: Estereotipos

Clases	Estereotipos	Función
Server Page [SP] (en español página servidora)		Representa la página web que tiene el código en el servidor.
Client Page [CP] (en español página cliente)		Representa la página web con formato HTML.
HtmlForm (en español formulario HTML).		Colección de elementos de entrada que son parte de una página cliente.

3.5.1. Diagramas de clases del diseño

A continuación se muestran los diagramas de clase del diseño (DCD) para los CU del sistema:

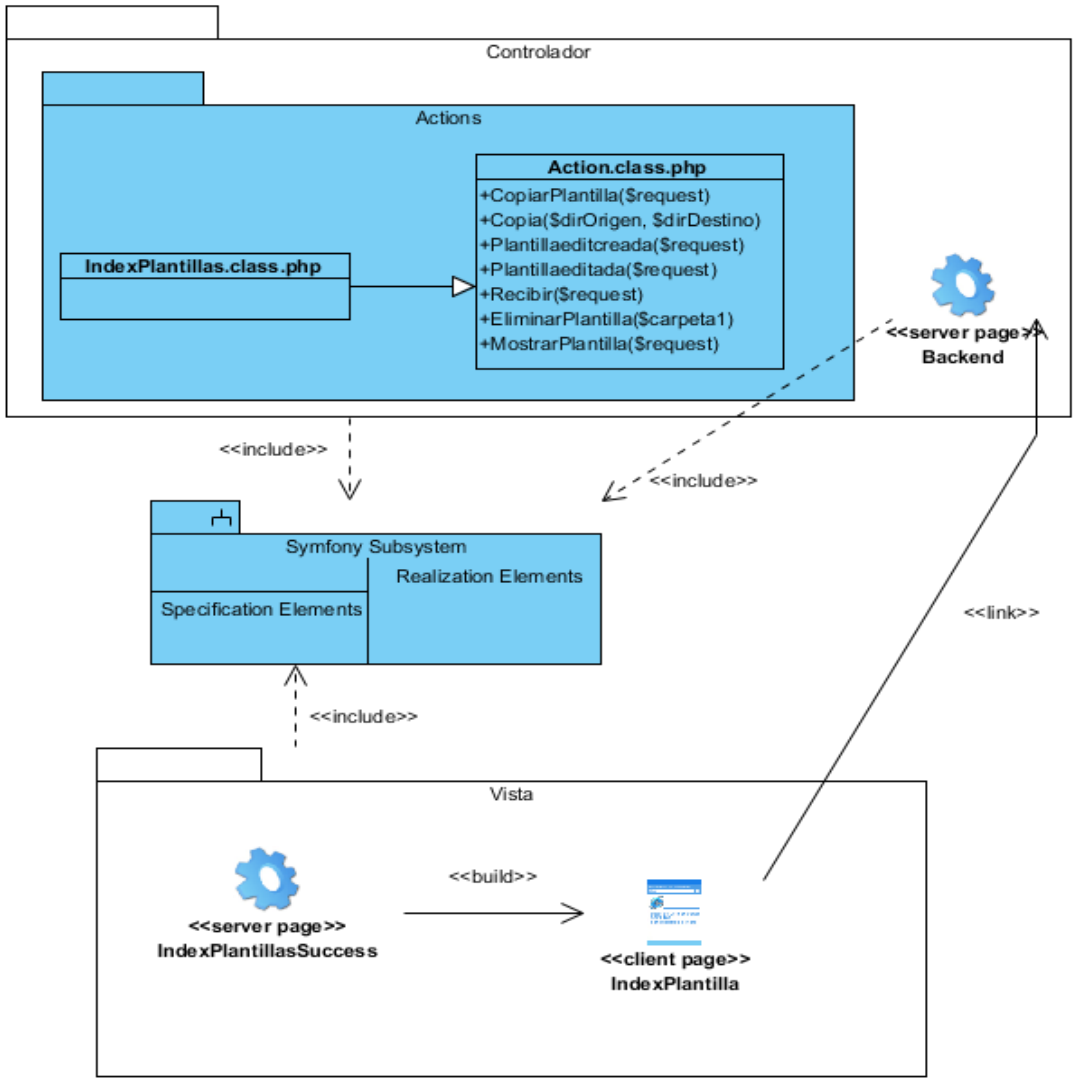


Figura 12: DCD Mostrar plantillas existentes

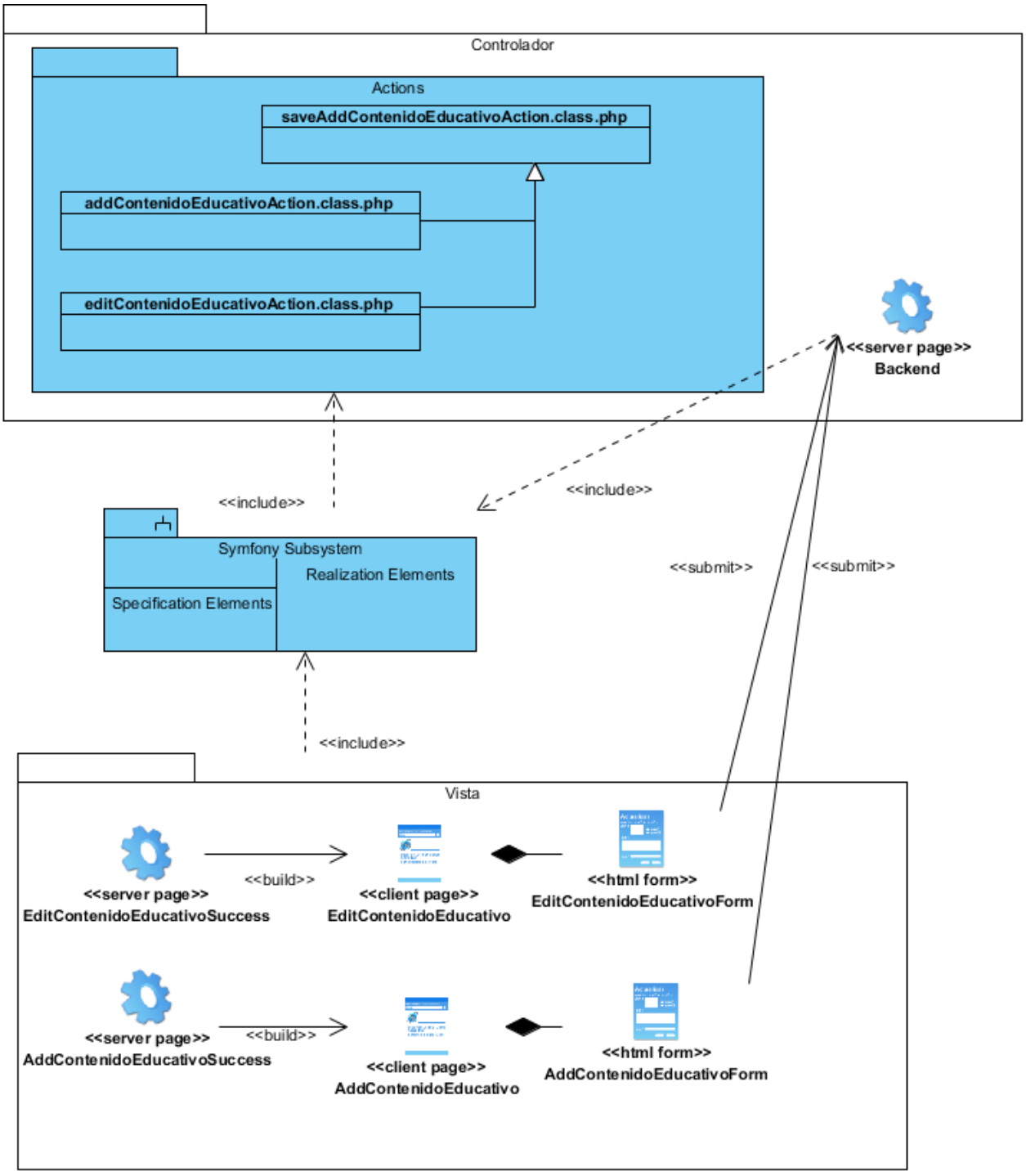


Figura 13: DCD Asignar plantilla

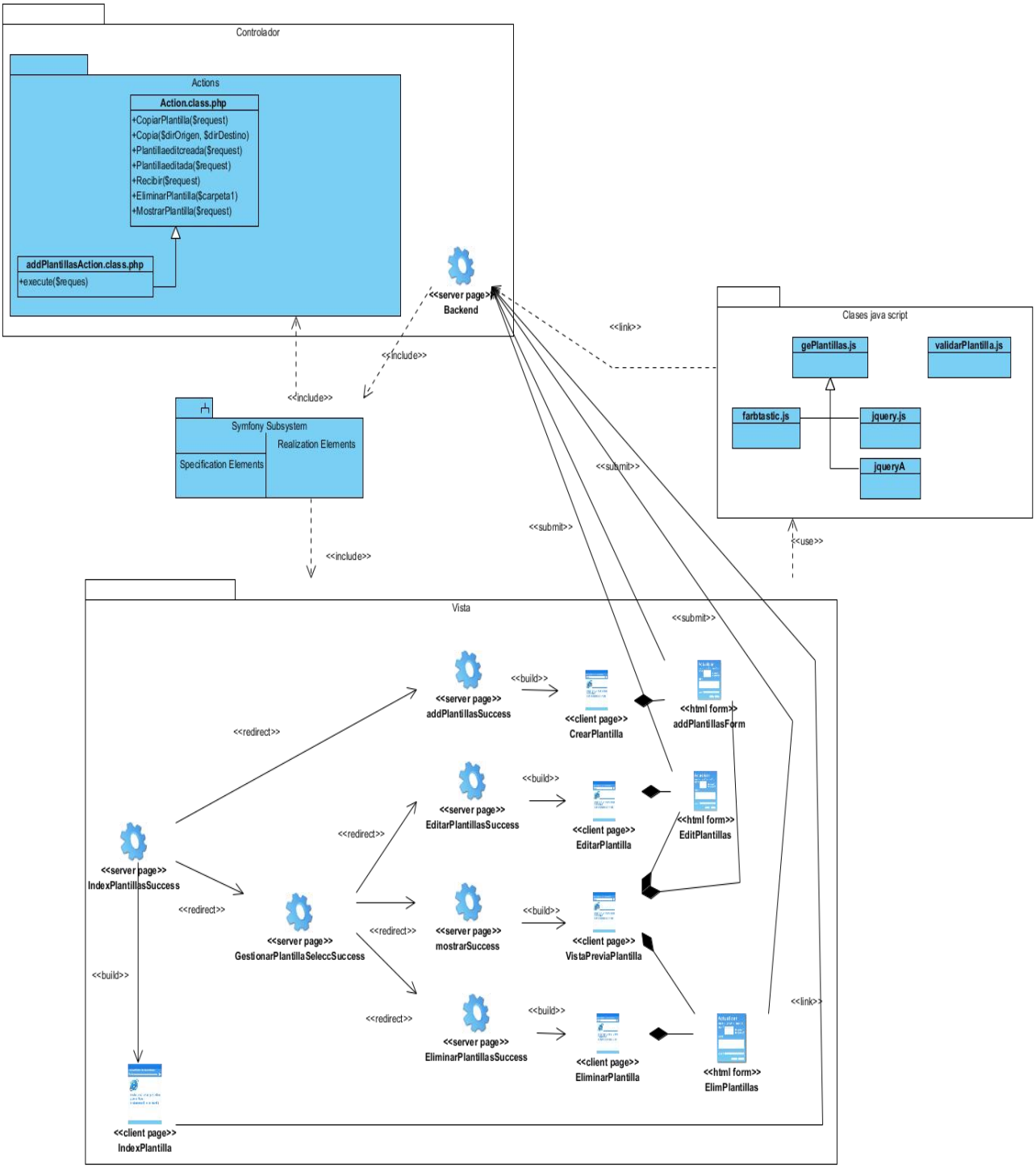


Figura 14: DCD Gestionar plantilla

3.6. Diagrama de despliegue

El modelo de despliegue se utiliza para capturar los elementos de configuración del procesamiento y las conexiones entre esos elementos. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos.

Dicho modelo consiste en:

- ✓ **Nodos:** elementos de procesamiento con al menos un procesador, memoria, y posiblemente otros dispositivos.
- ✓ **Dispositivos**: nodos estereotipados sin capacidad de procesamiento en el nivel de abstracción que se modela.
- ✓ **Conectores**: expresa el tipo de conector o protocolo utilizado entre el resto de los elementos del modelo.

A continuación se muestra el modelo de despliegue:

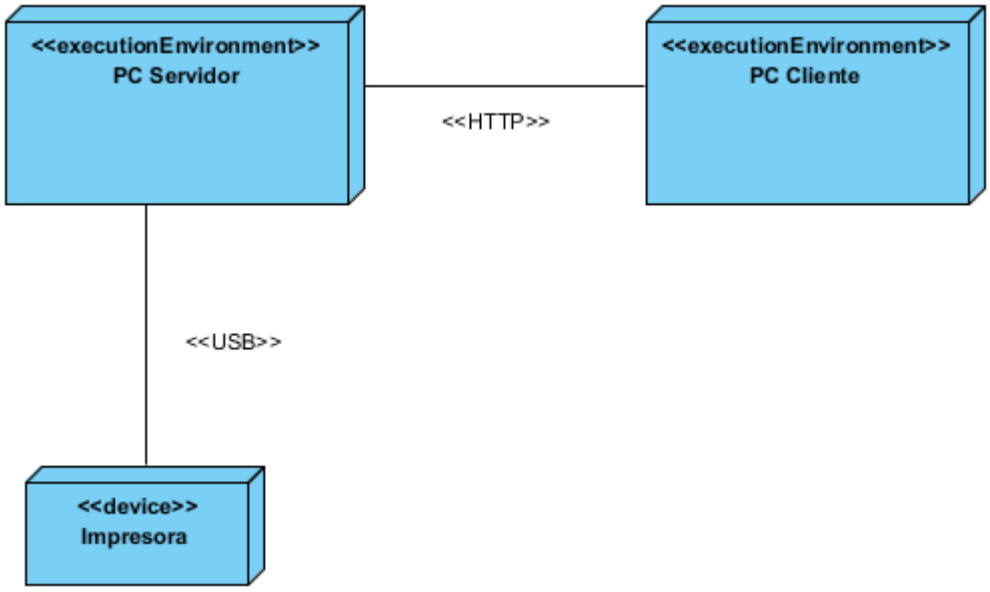


Figura 15: Diagrama de despliegue

Descripción de la funcionalidad y capacidad de los nodos:

- ✓ **PC Cliente:** ordenador cliente que se conecta a través de un navegador web al servidor central donde se encuentra la aplicación.
- ✓ **PC Servidor:** servidor central, el cual tiene instalado un servidor web donde hospeda todos los componentes necesarios para el funcionamiento del producto.
- ✓ **Impresora:** dispositivo utilizado por el sistema para posibilitar la impresión de documentos.

Descripción de elementos e interfaces de comunicación:

- ✓ **<<HTTP>>:** representa la conexión que se va a establecer entre una PC Cliente con el servidor central, donde se encuentra el servidor web.

3.7. Conclusiones

En este capítulo se incorporaron los diferentes artefactos pertenecientes al flujo de trabajo Análisis y Diseño con el objetivo de flexibilizar el trabajo y darle un punto de vista lógico y visible. Se analizó además la implementación del patrón MVC que realiza Symfony, los patrones de diseño utilizados, los cuales no permiten que el diseño y la implementación se observe de forma desorganizada ni engorrosa.

Capítulo 4: Implementación y prueba.

4.1. Introducción

En el presente capítulo se hace un análisis de las funciones que se tuvieron que ejecutar para lograr el desarrollo del componente. Se documenta el proceso de implementación de los elementos que fueron identificados en el diseño, este se describe a partir de los diagramas de componentes y despliegue, donde se puede observar la organización y las dependencias lógicas que existen entre los componentes del software. Además se explica de forma general la integración del componente a la colección y se procede a la verificación del funcionamiento del mismo.

4.2. Modelo de Implementación

El propósito de la fase de elaboración es el establecimiento de la línea base de la arquitectura del sistema para proporcionar una base estable al grueso del diseño y del esfuerzo de implementación en la fase de construcción. El flujo de trabajo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares.

4.2.1. Diagrama de componentes

El diagrama de componentes concibe un conjunto de elementos tales como, componentes, paquetes, subsistemas de implementación y sus relaciones. La idea principal de emplear este diagrama es para mostrar la estructura de nivel elevado del modelo de implementación enfatizado en los subsistemas de implementación y sus dependencias.

A continuación se presenta la distribución de los componentes del sistema:

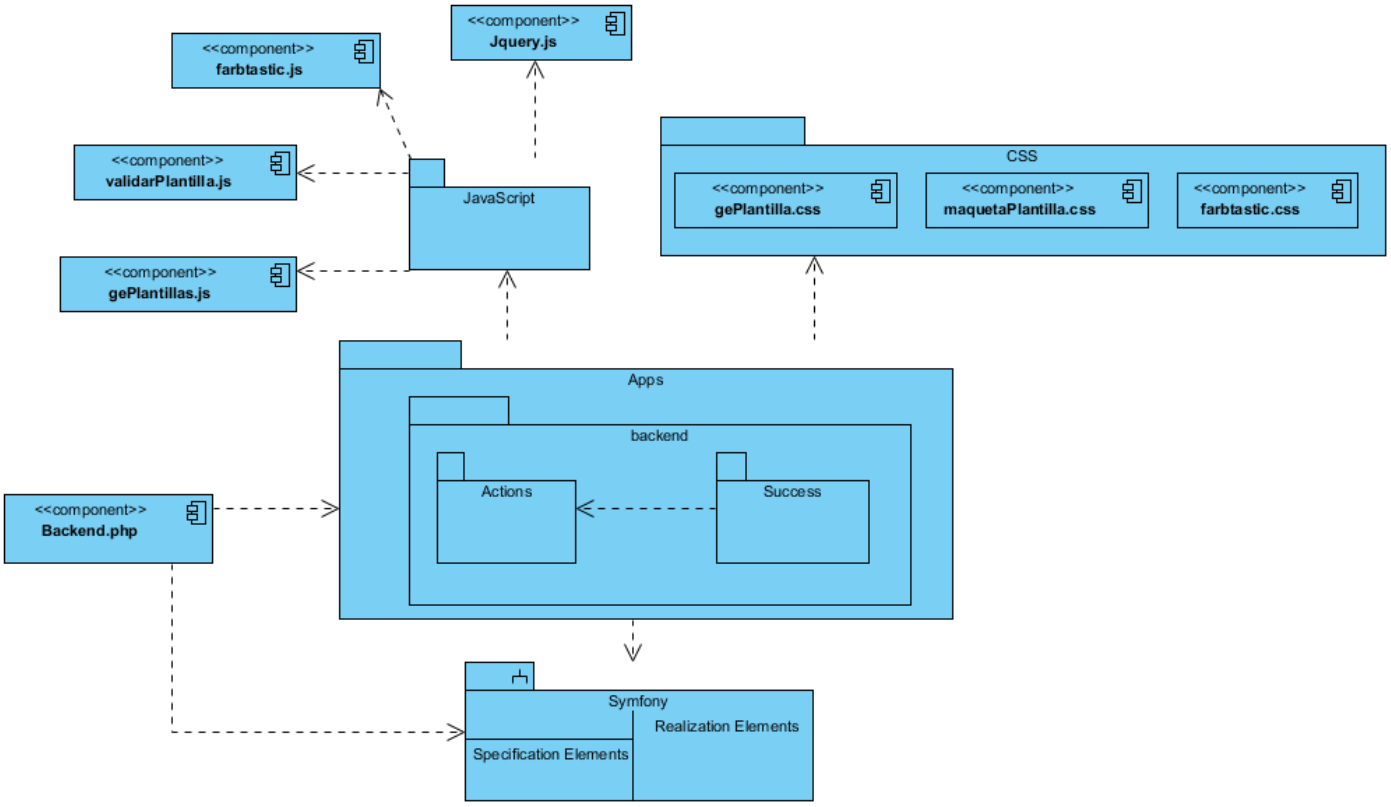


Figura 16: Diagrama de componentes

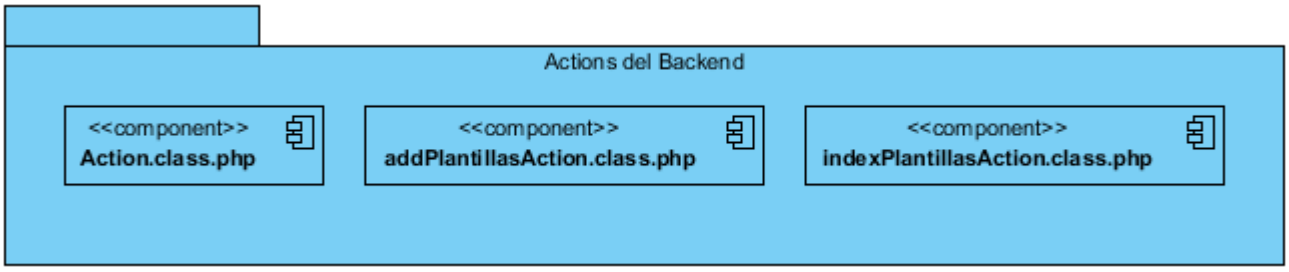


Figura 17: Paquete de actions del sistema

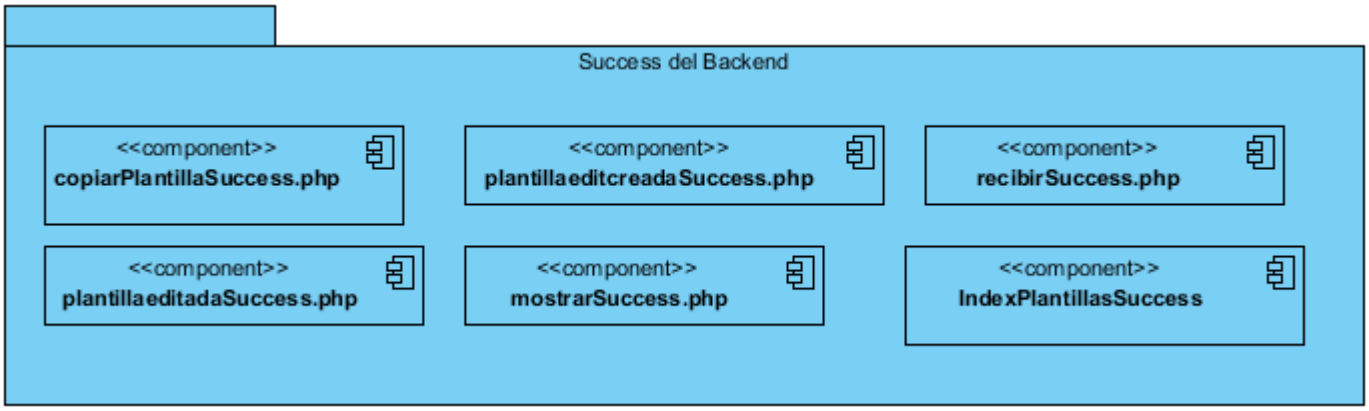


Figura 18: Paquete de success del sistema.

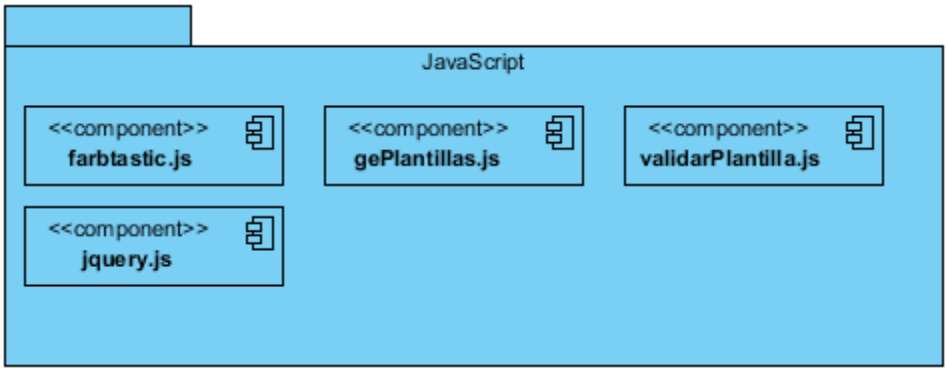


Figura 19: Paquete de archivos JavaScript

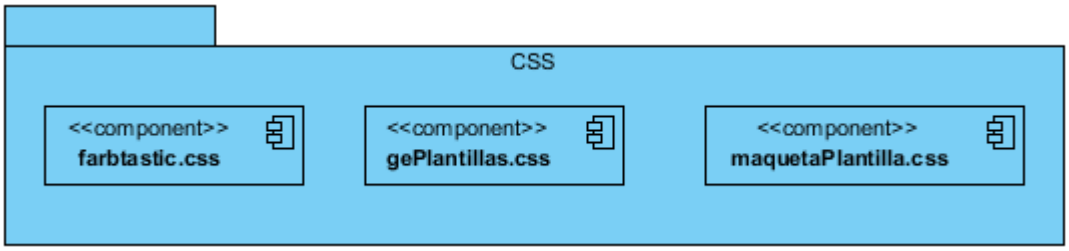


Figura 20: Paquete de archivos CSS

4.3. Verificación del funcionamiento del componente

La realización de las pruebas es un elemento crítico para garantizar la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación. Las pruebas son además una

actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente.

4.3.1. Niveles de prueba

Entre los niveles de prueba se encuentran el nivel de pruebas unitarias, nivel de pruebas de integración, nivel de pruebas del sistema y nivel de pruebas de aceptación. Una vez implementado el sistema fue sometido a los niveles de pruebas que a continuación se detallan, los cuales ayudaron a la detección de los errores existentes.

- ✓ **Pruebas unitarias:** Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de estos funcione correctamente por separado. El objetivo de las pruebas unitarias es aislar cada parte del programa y mostrar que las partes individuales son correctas. Proporcionan un contrato escrito que el fragmento de código debe satisfacer. Estas pruebas aisladas proporcionan cinco ventajas básicas: fomentan el cambio, simplifican la integración, documentan el código, separan la interfaz del código y hacen que los errores estén más acotados y sean fáciles de localizar.
- ✓ **Pruebas de integración:** Pruebas integrales o pruebas de integración son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias. Únicamente se refieren a la prueba o pruebas de todos los elementos unitarios que componen un proceso, hecha en conjunto, de una sola vez. Consiste en realizar pruebas para verificar que un gran conjunto de partes del software funcionan juntos.

4.3.2 Métodos de prueba

A continuación se explican los dos métodos de pruebas existentes:

Prueba de caja negra: Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software.

Prueba de caja blanca: Se basa en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que examinen que están correctas todas las condiciones y/o bucles para determinar si el estado real coincide con el esperado o afirmado. Esto genera gran cantidad de caminos posibles por lo que hay que dedicar esfuerzos a la determinación de las condiciones de prueba que se van a verificar.

Para verificar el funcionamiento del componente se utilizará las pruebas de caja negra ya que se centran principalmente en los requisitos funcionales del software.

Para desarrollar la prueba de caja negra existen varios métodos, entre ellos están:

(Pressman, 2005)

- ✓ **Método de partición de equivalencia:** divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba.
- ✓ **Método de análisis de valores límites:** este método prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- ✓ **Método de prueba de la tabla ortogonal:** es un método que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

El método seleccionado para aplicar las pruebas es el de partición equivalente, para la selección de este método se tuvo en cuenta que está orientado a los requisitos funcionales del software permitiendo que se creen juegos de datos para lograr probar a profundidad todas las funcionalidades del componente. Como se planteó anteriormente de este método se pueden derivar casos de pruebas que aseguren que se detecten errores de forma inmediata. Para el desarrollo de este método se identificaron los diseños de casos de pruebas.

4.3.3 Diseños de Caso de Prueba

Basada en la descripción de los CU de la investigación se generaron los diseños de casos de prueba correspondientes, con el objetivo de comprobar que cumple con las funcionalidades descritas. A continuación se muestra el diseño de casos de prueba elaborado para el caso de uso Asignar plantilla, el resto se encuentra en el Anexo 6.

Diseño de Caso de Prueba basado en el CU Asignar plantilla

Descripción general

El caso de uso se inicia cuando el actor selecciona la opción de editar producto o la de crear producto donde aparece entre otras opciones la selección de plantillas. Posteriormente el sistema le dará la posibilidad al actor de seleccionar una de las plantillas existentes y aplicársela al producto, finalizando así el caso de uso.

Condiciones de ejecución

Debe haberse generado el escritorio de trabajo del usuario autenticado, se accedió primeramente a editar plantilla o crear plantilla; y para aplicar una plantilla a un determinado producto, debe haberse seleccionado previamente la misma.

SC Asignar plantilla

Tabla 11: Escenarios para el CU Asignar plantilla

Escenario	Descripción	Mostrar	Respuesta del sistema	Flujo central
EC 1.1 selecciona la opción que le permite editar los productos o la de crear un nuevo producto.	Selecciona la opción que le permite editar los productos o la de crear un nuevo producto.	*V*	2. En ambos casos el sistema muestra una ventana con una serie de opciones entre las cuales aparece la opción de selección de plantilla, a partir de ahí muestra las plantillas (Ver CU Mostrar plantillas existentes).	Escritorio de Trabajo/Editar producto o Crear producto
EC 1.2 Selecciona la plantilla a aplicar al producto.	Selecciona la plantilla a aplicar al producto.	***Seleccionar plantilla*** *N/A*	Marca la plantilla seleccionada por el actor para aplicársela al producto deseado y permite: • Insertar • Limpiar • Cancelar	Escritorio de Trabajo/Editar producto o Crear producto

EC 1.3 Selecciona opción Insertar.	Selecciona opción Insertar.		Se asigna la plantilla al producto. El caso de uso termina.	Escritorio de Trabajo/Editar producto o Crear producto
EC 1.4 selecciona la opción Limpiar	selecciona la opción Limpiar		Desmarca la plantilla seleccionada. Permanece en esa ventana.	Escritorio de Trabajo/Editar producto o Crear producto
EC 1.5 selecciona la opción Cancelar	selecciona la opción Cancelar		Regresa a la vista anterior. El caso de uso termina.	Escritorio de Trabajo/Editar producto o Crear producto

Tabla 12: Descripción de variables para el CU Asignar plantilla

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Seleccionar plantilla	Campo de selección	si	Campo opcional que representa que contiene todas las plantillas existentes. Tipo seleccionable. Ejemplo: GeoClío, EducArte.

4.3.4. Resultados obtenidos

Durante el desarrollo del componente gestor de apariencia se realizaron pruebas unitarias para ir comprobando el funcionamiento del mismo. Estas no se planificaron ni se registraron sus resultados, fueron haciéndose a medida que la solución se desarrollaba. Para evaluar la solución se realizaron varias iteraciones donde se probó el software íntegramente. Prestando gran atención a las pruebas de integración con vista a probar funcionalidades que relaciona al componente con la colección "El Navegante". Se realizaron las pruebas a todos los casos de uso y arrojaron errores que se incorporaron al documento de no conformidades. Se realizó una primera iteración donde se detectaron diferentes tipos de

errores y se dieron solución a algunos de ellos, posteriormente se realizó otra iteración donde se pudo observar como los errores detectados en la primera iteración se erradicaron completamente.

Tabla 13: Resultado de las pruebas

Componente	CU	Iteración	NC	Cerrada(S)	No procede
Gestor de apariencia	4	1ra	6	4	0
		2da	2	2	0

4.4 Conclusiones

Se generaron diferentes artefactos como resultado de la implementación del componente, quedando confeccionada la vista estática del sistema. Finalmente con el empleo de las pruebas se pudo verificar el funcionamiento del sistema, para lograr esa finalidad se llevó a cabo el diseño de casos de pruebas definido por el método de prueba de caja negra; dichas pruebas arrojaron resultados que permitieron conocer los errores detectados al sistema, los mismos fueron erradicados satisfactoriamente dejando listo el software con vista a su implantación.

Conclusiones generales

Una vez terminado el trabajo se concluye que, mediante el empleo de la metodología RUP se obtuvieron los artefactos necesarios para la generación de la documentación del ciclo de vida completo del componente desarrollado, lo que permitirá efectuar posteriores cambios.

Además se desarrolló el componente que agiliza la gestión de apariencia de la colección "El Navegante" en su versión multiplataforma, dando respuesta a la necesidad que lo originó, permitiendo realizar cambios de apariencia en un instante y brindando comodidad y flexibilidad en el trabajo de los desarrolladores.

Recomendaciones

A partir del trabajo realizado se sugieren las siguientes recomendaciones:

- ✓ Incluir en el componente un reporte de las plantillas existentes como por ejemplo: el nombre del administrador que gestionó la plantilla, fecha de la última gestión con el tipo de acción realizada y otros aspectos que se deseen agregar para un mayor control de la gestión de plantillas.
- ✓ Permitir la creación de plantilla con una estructura que no necesariamente esté sujeta a las pautas establecidas por el diseño.

Referencias bibliográficas

1. **Labañino Rizzo, César.** *Maestría Ciencias de la Educación.* 2006.

2. *Castilla-La Mancha.* [En línea] 22 de octubre de 2008. [Citado el: 2 de febrero de 2012.] http://edu.jccm.es/joomla15/index.php/manuales-joomla-15x/videotutoriales-administracion/72-plantillas-en-joomla-15x.html.

3. **Mellado, Juan.** *inmensia.* [En línea] 22 de septiembre de 2005. [Citado el: 1 de febrero de 2012.] http://www.inmensia.com/articulos/drupal/personalizacion.html.

4. **Medín, Juan.** *Artículos.* [En línea] 2011. [Citado el: 3 de marzo de 2012.] http://juanmedin.com/posts/drupal-aspectos-positivos-negativos.

5. *Wordpress.* [En línea] [Citado el: 16 de enero de 2012.] http://es.wordpress.org/.

6. **Fernando.** *Ayuda Wordpress.* [En línea] [Citado el: 16 de enero de 2012.] http://ayudawordpress.com/5-razones-por-las-que-deberias-usar-wordpress-en-lugar-de-joomla/.

7. **Cáceres Días, Yanetsi y Herrera Sánchez, Noslén.** *Arquitectura de Presentación de la Plataforma Zera.* Cuba : s.n., 2011.

8. **Rueda Chacón, Julio César.** *APLICACIÓN DE LA METODOLOGÍA RUP PARA EL DESARROLLO RÁPIDO DE APLICACIONES BASADO EN EL ESTÁNDAR J2EE.* Guatemala : s.n., 2006.

9. **Informática, Sub-Jefatura.** *Herramientas Case.* Cuba : s.n. 875-99-OI-OTDETI-INEI.

10. **Díaz Gutiérrez, Jorge Antonio.** *Desarrollo de un IDE libre y multiplataforma para la creación de componentes visuales de ActionScript para Software Educativo: codeDraw.* Habana-Cuba : s.n., 2009.

11. **Eguíluz Pérez, Javier.** *Introducción a JavaScript.* 25 de marzo 2009.

12. Manual JavaScript. [En línea] [Citado el: 12 de enero de 2012.] http://www.uazuay.edu.ec/estudios/sistemas/lenguaje_iii/MAnualJavaScript/caracteristicas.htm.

13. Ciberaula. [En línea] 2010. [Citado el: 12 de enero de 2012.] http://php.ciberaula.com/articulo/introduccion_php.

14. **Carlos Blanco, Ignacio.** *Plataformas de desarrollo de aplicaciones Web orientadas a componentes reutilizables.* 2008.

15. **Sánchez, Jordi.** jordisan.net. [En línea] 29 de septiembre de 2006. [Citado el: 11 de enero de 2012.] http://jordisan.net/blog/2006/que-es-un-framework.

16. **Morgado Novo, Jorge y Carmenate Cabrera, Erianne.** *"Análisis, Diseño e Implementación del módulo Resultados de la Colección El Navegante en su versión multiplataforma".* Ciudad de La Habana : s.n., 2010.

17. **Alvarez, Miguel Angel.** desarrolloweb.com. [En línea] 25 de marzo de 2009. [Citado el: 11 de enero de 2012.] http://www.desarrolloweb.com/articulos/introduccion-jquery.html.

18. **Potencier, Fabien y Zaninotto, François.** *Symfony la guía definitiva.* 2008.

19. **Jaime.** *Entornos de Desarrollo Integrados.* 2009.

20. **Bautista Sánchez, Lioba y Gamez Rios, Blanca Estela.** *PROGRAMACIÓN WEB II.* RIOVERDE : s.n., 2009.

21. **xhidnoda.** *Supremacía Linux.* [En línea] 22 de abril de 2011. [Citado el: 4 de abril de 2012.] http://www.supremacialinux.com/las-nuevas-caracteristicas-de-netbeans-7-0/.

22. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* Madrid-España : DIGRAF, S. A., 2000. 84-7829-036-2.

23. **S. Pressman, Roger.** *Ingeniería del Software, un enfoque práctico.*

24. **Larman, Craig.** *UML y Patrones. Introducción al diseño orientado a objetos.* PRENTICE HALL, México : por Prentice Hall Hispanoamericana, S.A. ISBN: 970-17-0261-1.

25. **Bahit, Eugenia.** *El paradigma de la Programación Orientada a Objetos en PHP y el patrón de arquitectura de Software MVC.*

26. **Potencier, Fabien y Zaninotto, François.** *Symfony la guía definitiva.* 2008.

27. **Soriano, Ing. Fernando.** *Análisis y Diseño Orientado a Objetos.* s.l. : publicación de la Universidad de FASTA, 2006.

28. **Prieto, Félix.** *Patrones de diseño.* s.l. : públicación de la Universidad de Valladolid, 2008.

29. *AJAX Ya.* [En línea] [Citado el: 3 de marzo de 2012.] http://www.ajaxya.com.ar/temarios/descripcion.php?cod=8&punto=1.

30. CubaMinRex. [En línea] [Citado el: 21 de noviembre de 2011.] http://www.cubaminrex.cu/sociedad_informacion/Informacion_Gral.htm.

Bibliografía

1. **Larman, Craig.** *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* México : 2da edición.

2. —. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : D.R O 1999 por Prentice Hall Hispanoamericana, S.A. 1era edición. ISBN 0-13-748880-7.

3. **Revisado por Msc Pérez Martinto, Pedro Carlos.** *Teoría de Muestreo: población y muestra. Diseño experimental y métodos.* Habana : Publicaciones Universidad de las Ciencias Informáticas, 2006.

4. *Metodología de la investigación científica.* [Presentación PowerPoint] Habana : Producciones Universidad de las Ciencias Informáticas, 2012.

5. **Luna, Marcelo.** *Ingeniería de Requerimientos.* [En línea] 2007. [Citado el: 11 de abril de 2012.] http://www.nachomezzadra.com.ar/IngenieriaRequerimientos/MatrizTrazabilidad.html..

6. *BALUART.NET. Teconología entretenimiento y cultura.* [En línea] 19 de julio de 2010. [Citado el: 9 de mayo de 2012.] http://www.baluart.net/articulo/introduccion-a-los-patrones-de-diseno-con-php.

7. *Creación de formularios symfony.*

8. **Sanchez Ortega, Ivan.** *curso de php 5.*

9. *Manual de gestión de archivos en php.*

10. **Ramos Monso, Martín.** *Programación php.Tomo: sitios web dinámicos e interactivos.*

11. **Vazquez, Carlos y Ferrol, Marinno.** *Programación en php5 nivel básico.*

12. *Listas Dinámicas con PHP.*

13. *El tutorial Jobeet.*

14. *Symfony la guía definitiva.*

15. *Symfony la guía definitiva 1.2.*

16. **Eguíluz Pérez, Javier.** *Introducción a AJAX.*

17. **Álvarez, Miguel Angel.** *Manual de jQuery.*

18. —. *Trabajar con JSON desde PHP.*

19. **Eguíluz Pérez, Javier.** *Introducción a CSS.*

20. *Drupal Hispano.* [En línea] 11 de abril de 2005. [Citado el: 27 de mayo de 2012.] http://drupal.org.es/caracteristicas.

21. **P. Domíngues, Danilo.** *Danilo Domíngues P.Site.* [En línea] 13 de junio de 2008. [Citado el: 27 de mayo de 2012.] http://danilo04.accionasolutions.net/drupal-un-estudio-detallado/72.

22. **Rincón, Carlos.** *Blog de Carlos Rincón.* [En línea] 1 de marzo de 2009. [Citado el: 27 de mayo de 2012.] http://carlos.rinconsanchez.com/drupal-vs-joomla-una-comparativa-sincera-de-un-consultor-de-ibm.

23. *ite Instituto de Tecnologías Educativas.* [En línea] [Citado el: 27 de mayo de 2012.] http://www.ite.educacion.es/formacion/materiales/99/cd/modulo_01/caractersticas_de_joomla.html.

24. *webempresa.com.* [En línea] http://www.webempresa.com/creacion-web-joomla/caracteristicas-de-joomla.html.

25. *CMS Joomla.* [En línea] 26 de agosto de 2007. [Citado el: 27 de mayo de 2012.] http://cmsjoomla.blogspot.com/2007/08/caractersticas-de-joomla.html.

26. **R. Rancel, Mario.** *OGRAMAR.com.* [En línea] [Citado el: 27 de mayo de 2012.] http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=141:caracteristicas-generales-de-joomla-core-frontend-backend-componentes-vistas-plantillas-etc-cu00405a&catid=38:curso-qcreacion-web-con-joomla-nivel-iq&Itemid=152.

27. *Web profesional.* [En línea] 20 de febrero de 2009. [Citado el: 27 de mayo de 2012.] http://www.web-profesional.net/es/blog/52-joomla/108-ventajas-joomla.

28. *astrolabio.* [En línea] 20 de julio de 2008. [Citado el: 27 de mayo de 2012.] http://www.astrolabio.com.co/diseno-web/13-joomla-razones-para-usarlo-en-su-sitio-web.html.

29. **Darío.** *maestros del web.* [En línea] [Citado el: 28 de mayo de 2012.] http://www.maestrosdelweb.com/editorial/tips-para-mejorar-tu-desempeno-con-joomla/.

30. *Joomla Región capital.* [En línea] [Citado el: 28 de mayo de 2012.] http://joomlaiutfrp.wikispaces.com/Algunas+desventajas+de+Joomla.

31. **Ávila Dorado, Carlos Andrés.** *eduteka.* [En línea] 1 de septiembre de 2008. [Citado el: 28 de mayo de 2012.] http://www.eduteka.org/BlogsWordpress.php.

32. *WordPress.org.* [En línea] [Citado el: 28 de mayo de 2012.] http://codex.wordpress.org/es:WordPress_Features.

33. *poetify.* [En línea] [Citado el: 28 de mayo de 2012.] http://www.poetify.es/20-ventajas-de-wordpress/.

34. *INFORMATICAHOY.* [En línea] [Citado el: 28 de mayo de 2012.] http://www.informatica-hoy.com.ar/internet/WordPress-Joomla-Beneficios-y-desventajas.php.

35. *AltamiraWeb.* [En línea] [Citado el: 28 de mayo de 2012.] http://altamiraweb.es/las-ventajas-de-wordpress.

36. Revista on-line sobre diseño y programación web. *MAS PIXEL.* [En línea] [Citado el: 28 de mayo de 2012.] http://www.maspixel.com/2012/02/ventajas-de-usar-wordpress-como-gestor-de-contenidos/.

37. *HOSPEDAJE WEB HOY.* [En línea] 24 de mayo de 2012. [Citado el: 28 de mayo de 2012.] http://hospedajewebhoy.com/ventajas-wordpress/.

38. **juafemal.** *Agucho.* [En línea] [Citado el: 28 de mayo de 2012.] http://blog.espol.edu.ec/juanferjac/2009/08/27/ventanjas-y-desventajas/.

Glosario

Apariencia: conjunto de características que conforman el aspecto gráfico de una plantilla.

AJAX: son las siglas de **A**synchronous **J**avaScript **A**nd **X**ML. No es un lenguaje de programación sino un conjunto de tecnologías (HTML-JavaScript-CSS-DHTML-PHP/ASP.NET/JSP-XML) que nos permiten hacer páginas de internet más interactivas. (29)

Hipermedia: Hipertexto que no se limita a documentos escritos, sino también otros medios como imágenes, audio y vídeo.

Multiplataforma: Programa o aplicación que puede utilizarse sin inconvenientes en distintas plataformas de hardware y sistemas operativos.

Software educativo: Programa informático diseñado con la intención de ser utilizado en el contexto del proceso de enseñanza - aprendizaje. Se caracteriza por ser altamente interactivo, a partir del empleo de recursos multimedia, diccionarios especializados, explicaciones de experimentados profesores, ejercicios y juegos instructivos que apoyan las funciones de evaluación y diagnóstico.

HTTP: Protocolo de transferencia de hipertexto (del inglés *Hypertext Transfer Protocol*), es el método más común de intercambio de información en la red informática mundial (del inglés *World Wide Web*), mediante el cual se transfieren las páginas web a un ordenador.

URL: La sigla URL (en inglés *Uniform Resource Locator*) se refiere a la dirección única que identifica a un sitio web en Internet.

Web: Es un sistema de documentos de hipertexto y/o hipermedias enlazados y accesibles a través de Internet. Con un navegador web, un usuario visualiza páginas web que pueden contener texto, imágenes, videos u otros contenidos multimedia, y navega a través de ellas usando hiperenlaces.

HTML: Lenguaje de Marcado de Hipertexto, es el lenguaje autoritario para crear documentos en la World Wide Web. Define la estructura de un documento web usando etiquetas y atributos.

CSS: Lenguaje de hoja de estilo que permite que los autores y los usuarios asocien un estilo (las fuentes y espaciado) a los documentos estructurados (documentos HTML y aplicaciones XML).