UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Desarrollo del sistema de gestión de contenidos de la colección El Navegante en su versión multiplataforma

Trabajo para optar por el título de Ingeniero en Ciencias Informáticas

Autores:

Ernesto de la Soledad García

Yolanda Mauri Pérez

Tutores:

Ing. José Ernesto Lara Rodríguez

Ing. Jorge Martínez Padrón

Ing. Angel Alberto Vazquez Sánchez

12 de junio de 2012

La Habana, Cuba

"Se puede hacer demagogia hablando del sexo de los ángeles o quizás de pintura abstracta; no de software".

Alfredo de Hoces (Fuckowsky).

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____
Yolanda Mauri Pérez
Tesista

_____
Ernesto de la Soledad García
Tesista

_____
Ing. José Ernesto Lara Rodríguez.
Tutor

_____
Ing. Jorge Martínez Padrón
Tutor

_____
Ing. Angel Alberto Vazquez Sánchez
Tutor

DEDICATORIA

De Yolanda Mauri Pérez:

***A René Ángel García Enríquez**, porque la vida no le permitió llegar hasta aquí, porque si la magia de este instante de me diera un único deseo, pediría que volviera a nacer y fuese mi amigo una vez más.*

***A mi Juli**, pues gracias a él todos los recuerdos que me llevo de la universidad están matizados con nuestro amor. Estoy feliz de estar a su lado por el modo en que nos involucramos el uno en las cosas del otro y la espontaneidad con que compartimos todo.*

***A mis padres Yolanda y Carlos Javier** que me dieron la vida. Ellos representan todo para mí, son ejemplo, guía e inspiración. Todos y cada uno de mis logros como estudiante han sido para corresponder a los sacrificios que han hecho para hacer mis sueños realidad. Papi y mami me he formado como un Ingeniero (igual que ustedes) y mi única expectativa es ser tan buen profesional como lo han sido ustedes a lo largo de sus años de trabajo.*

De Ernesto de la Soledad García:

***A la memoria de mis abuelos Alicia y Víctor**, por darme todo su amor mientras vivieron conmigo, por ser parte de lo que hoy soy, siempre los recordaré.*

***A mi mamá**, por ser mi cómplice y todo. Por ser siempre el ejemplo a seguir y la meta a alcanzar. Por ser la mejor mamá del mundo. Por apoyarme, creer en mí y guiarme por el camino correcto. A ti te lo debo todo, a ti te debo ser quien soy.*

RESUMEN

La colección El Navegante es un hiperentorno educativo que cuenta con diez productos y cada uno de ellos integra los subsistemas: General, Contenidos, Ejercicios, Juegos, Mediateca, Resultados y Maestro. Debido al cúmulo de recursos que se manejan en dichos módulos se hace muy compleja la tarea de administrar tanta cantidad de información. En este sentido el presente documento aborda el desarrollo de un panel de administración para la versión multiplataforma de El Navegante.

Esta herramienta (como principal resultado de la investigación) es una aplicación web con soporte de datos que centraliza las tareas de gestión de contenidos de El Navegante para los módulos Contenidos, Mediateca, Juegos y Ejercicios, además de la información de los servicios generales de la colección: ayuda contextual, efemérides y “sabías qué”. Como lenguajes de programación se utilizaron JavaScript y PHP, apoyado este último del framework Symfony. El proceso de desarrollo estuvo guiado por la metodología de RUP y fueron generados todos los artefactos correspondientes a los flujos de trabajo requisitos, diseño, implementación y pruebas.

Palabras claves: colección, gestión de contenidos, panel de administración.

ÍNDICE

INTRODUCCIÓN

Las Tecnologías de la Informática y las Comunicaciones (TIC) se insertan hoy a nivel mundial en cada uno de los procesos que involucran a la sociedad. Específicamente en Cuba el proceso docente-educativo es uno de los más beneficiados con los esfuerzos realizados por el gobierno en pos de informatizar el país.

Con el afán de crear soluciones propias y contextualizadas a los diferentes niveles de enseñanza en Cuba, a partir del año 2002 empezaron a desarrollarse las colecciones de software educativo Multisaber, El Navegante y Futuro para las enseñanzas primaria, secundaria básica y preuniversitaria respectivamente (1). Para ese entonces la experiencia cubana en la implementación de este tipo de software era prácticamente nula. Todavía hoy se trabaja para mejorar dichos productos de acuerdo con las tendencias actuales de esta rama, teniendo entre los objetivos principales evitar el uso de herramientas propietarias.

Entre las soluciones informáticas que se están desarrollando en el Centro de Tecnologías para la Formación (FORTES) perteneciente a la Universidad de las Ciencias Informáticas (UCI), se encuentra la versión multiplataforma de El Navegante; el responsable de la migración es el proyecto productivo Multisaber-Navegante.

El principal objetivo de esta nueva versión era migrar el producto a una aplicación web, implementada en su totalidad con software libre, ya que anteriormente había sido implementada con tecnologías propietarias y era una solución escasamente configurable y extensible. Al ser una aplicación de escritorio creada sólo para el sistema operativo Windows, requería de mucho esfuerzo para su despliegue puesto que debía ser instalada y configurada en cada estación de trabajo.

La versión multiplataforma de la colección El Navegante es un hiperentorno educativo que cuenta con diez productos que abarcan diferentes áreas del conocimiento que deben dominar los estudiantes de secundaria básica; a su vez cada uno de ellos integra los subsistemas: General, Contenidos, Ejercicios, Juegos, Mediateca, Resultados y Maestro (2).

El módulo General agrupa los servicios comunes de la colección tales como: impresión y búsqueda de contenido, manipulación de la mascota, consulta al perfil de usuario y créditos del producto. El módulo

Contenidos muestra la selección de los textos vinculados con la temática específica que aborda un software. Los módulos Ejercicios y Juegos, como lo indican sus respectivos nombres, relacionan una amplia gama de tipologías de juegos y ejercicios dedicados a comprobar los conocimientos adquiridos por los estudiantes. La Mediateca comprende diversidad de elementos audiovisuales del software agrupados según su formato: animaciones, diaporamas, sonidos, videos e imágenes; además de las personalidades y el glosario de términos. En Resultados se muestra el recorrido del estudiante en el software, mientras que al módulo Maestro sólo tienen acceso los profesores que guiarán a los estudiantes en el trabajo con el software; en él pueden ser gestionadas las noticias, los perfiles de los estudiantes y las recomendaciones metodológicas. En su conjunto todos estos subsistemas permiten organizar de forma modular tanto la información que se le brinda al usuario como la implementación del sistema.

Debido al cúmulo de información que manejan los hiperentornos educativos se hace muy compleja la tarea de administrar el contenido. Ante este conflicto fueron desarrollados en el proyecto varios sistemas para administrar el contenido de algunos módulos. A pesar de que dichos sistemas respondían a un mismo panel de administración, entre ellos existían un sinnúmero de elementos heterogéneos en cuanto al diseño de interfaz y la arquitectura de la información; además durante la implementación no se respetaron las precisiones de la arquitectura de software y por incumplimientos con el cronograma las funcionalidades no fueron programadas con calidad.

Los aspectos anteriores, junto a un conjunto de deficiencias identificadas por los usuarios del panel de administración tuvieron un impacto negativo en la calidad de la información gestionada. Entre las deficiencias encontradas se tiene que dicha herramienta no permitía gestionar los productos, y dentro de cada uno de ellos no podían vincularse todos los elementos (texto, imagen, video y audio) que componen los diferentes módulos; además tampoco permitía gestionar las efemérides, los “sabías que” ni las diferentes pantallas que componen la ayuda contextual de cada producto. Por otro lado es importante mencionar que el mecanismo de autenticación era muy débil pues todos los usuarios tenían los mismos privilegios dentro de la aplicación.

Tomando como punto de partida de esta investigación la situación descrita anteriormente, se identifica el siguiente **problema a resolver**: ¿Cómo mejorar la gestión de los contenidos que manejan los subsistemas de la colección de software educativo El Navegante en su versión multiplataforma? El

objeto de estudio es el desarrollo de software educativo, mientras que el **campo de acción** se enmarca en el desarrollo de aplicaciones que permitan la gestión de contenidos en software educativo.

El **objetivo general** de la investigación es desarrollar una aplicación informática que permita mejorar la gestión de los contenidos de los subsistemas de la colección de software educativo El Navegante en su versión multiplataforma. En concordancia con lo anterior se plantean los siguientes **objetivos específicos**:

- Investigar las tendencias actuales en aplicaciones para la gestión de contenidos en software educativo.
- Definir los requisitos funcionales y no funcionales que permitan mejorar la gestión de los contenidos de la colección de software educativo El Navegante en su versión multiplataforma.
- Implementar las funcionalidades que permitan mejorar la gestión del contenido de la colección de software educativo El Navegante en su versión multiplataforma.
- Probar las funcionalidades implementadas para mejorar la gestión del contenido de la colección de software educativo El Navegante en su versión multiplataforma.

Para dar cumplimiento a cada uno de los objetivos planteados se han definido las **tareas de investigación** que a continuación se listan:

- Estudiar las tendencias actuales de las aplicaciones de gestión de contenidos en software educativo que puedan tener impacto en la propuesta de solución.
- Identificar las limitaciones existentes en la gestión de contenidos de la colección de software educativo El Navegante en su versión multiplataforma, basándose en el estudio de los subsistemas que la componen.
- Identificar los requisitos funcionales y no funcionales de la aplicación.
- Definir la arquitectura que soporte la implementación de las funcionalidades que serán añadidas al sistema.
- Implementar las funcionalidades que den cumplimiento a los requisitos identificados.
- Probar las funcionalidades para detectar posibles errores.

Como **idea a defender** se tiene que si se logra mejorar el sistema de gestión de contenidos para la colección El Navegante en su versión multiplataforma, se agiliza el proceso de desarrollo logrando mayor eficiencia en el montaje y corrección de los contenidos.

Los **métodos científicos** fueron fundamentales en el desarrollo de la investigación puesto que contribuyeron a caracterizar el objeto de estudio (3) (4). A continuación se explica cómo fueron aplicados:

Métodos Teóricos

Analítico-sintético: Para desarrollar la investigación fue necesario analizar desde diferentes aristas los conceptos asociados al objeto de estudio y los sistemas similares. La síntesis de la información recopilada quedó reflejada en los resultados de la pesquisa.

Análisis histórico-lógico: Este método permitió analizar la trayectoria histórica de la colección de software educativo El Navegante en su proceso de migración a la versión multiplataforma y de los módulos que están presentes en ella.

Modelación: Fue de gran importancia para la investigación puesto que se aplicó en la representación gráfica de las características y funcionalidades del software desarrollado.

Métodos Empíricos

Observación: Fue utilizado en diferentes momentos de la investigación para agrupar elementos relacionados con la problemática en cuestión.

El documento que respalda la pesquisa está estructurado en cuatro capítulos que describen todo el proceso de investigación, diseño, implementación y prueba del software en cuestión:

1. **Fundamentación teórica:** Aborda los conceptos relacionados con el objeto de estudio y justifica la selección de las tecnologías, metodologías y herramientas que serán utilizadas en el desarrollo del sistema.
2. **Descripción del sistema:** Explica detalladamente el proceso de captura y refinamiento de los requisitos del software.
3. **Diseño del sistema:** Mediante los diagramas generados por la metodología de desarrollo utilizada se describe la estructura interna del sistema.
4. **Implementación y pruebas:** A partir de la documentación generada en el capítulo anterior se documenta el proceso de implementación y las estrategias de prueba a utilizar.

FUNDAMENTACIÓN TEÓRICA

Introducción

Un software educativo incorpora gran cantidad de contenido que, en su conjunto, conforman un recurso didáctico valioso para cualquier educador. Toda esta información puede ser gestionada a través de un sistema de gestión de contenidos o panel de administración. Las directrices en el desarrollo de paneles de administración varían de acuerdo con las necesidades de la organización que se beneficie con el producto final. En este capítulo se exponen los conceptos fundamentales relacionados con el desarrollo de aplicaciones que permiten la gestión de contenidos en software educativo, así como el estudio del arte sobre sistemas similares y el análisis para la selección de las tecnologías, metodologías y herramientas que serán utilizadas en el desarrollo del sistema.

Software educativo

Según (5) y (6), el software educativo se define como cualquier programa computacional cuyas características estructurales y funciones sirvan para apoyar el proceso de enseñar, aprender y administrar, es decir, un material de aprendizaje especialmente diseñado para ser utilizado en una computadora en los procesos de enseñar y aprender. Una definición más precisa sobre el término la ofrece (7): programas de computación que son elaborados con un sólo propósito y con características propias que determinan su carácter educacional.

Estos conceptos al igual que todos los que se relacionan con la informática, pueden ser replanteados a medida que se desarrolle esta área de la ciencia, por lo que de forma genérica puede expresarse que un software educativo es una aplicación o programa computacional que facilita el proceso de enseñanza aprendizaje (6). Como principal característica de este tipo de programa se identifican los elementos audiovisuales como imágenes, sonidos, videos o textos, para fungir como un recurso didáctico interactivo. La interactividad del mismo radica en la inmediatez con la que el estudiante percibe la respuesta a una acción realizada previamente.

Hiperentorno educativo

Un hiperentorno educativo o hiperentorno de aprendizaje es resultado de la fusión en una sola aplicación informática de varias tipologías de software educativo, díganse: libros electrónicos, juegos didácticos, sistemas expertos o simuladores; todo ello con la incorporación de tecnologías de hipermedia e hipertexto (6).

El hipertexto es la organización de determinada información en diferentes nodos conectados entre sí a través de enlaces. La tecnología multimedia es la que permite integrar varios medios como sonido, imágenes y textos, en una misma presentación. La hipermedia, por tanto, es la tecnología que permite estructurar la información de manera no lineal, a través de nodos interconectados por enlaces. La información presentada en estos nodos podrá integrar diferentes medios tales como: texto, sonido, gráficos y video (8).

Sistemas gestores de contenido

Con el crecimiento de la información que contienen los sitios web, el dinamismo que han alcanzado y la inclusión de elementos multimedia dentro de ellos, se hizo imprescindible la automatización de los procesos de gestión del contenido. Esta variante de aplicación informática significó un salto en la reutilización de los contenidos, permitiendo que sean publicados en diferentes formatos.

Un sistema gestor de contenidos o de gestión de contenidos o CMS[1] es un programa que permite generar una estructura de soporte para la creación y administración de contenidos principalmente en sitios web (9) (10). A esta definición también se le asocian otros términos como backend y panel de administración, de los cuales otros autores han puntualizado que son programas informáticos que permiten aumentar el volumen, la variedad, la complejidad y el control de los contenidos de un sitio web (11).

El uso de sistemas para administrar información tiene como principales ventajas que proveen de una interfaz que maneja de manera independiente el contenido del diseño (11) y permiten a varias personas acceder a editar el contenido al unísono. De acuerdo con el fin para el que sean desarrollados pueden alcanzar un alto grado de personalización para el diseño y las funcionalidades de los sitios web.

[1] *Del inglés Content Management System.*

En este sentido ha sido implementada una amplia gama de CMS que por su uso y funcionalidad responden a la siguiente clasificación (12) (13):

Clasificación	Ejemplos
Blogs	Word Press, bBlog, DotClear, Wordsmith
Gestores de foros	Vainilla, FluxBB, phpBB, Invision Board
Wikis	MediaWiki, DokuWiki, phpMyFAQ
Enseñanza	Moodle, Interact, Dokeos, DrupalEd
Gestores de comercio electrónico	PrestaShop, Magento, osCommerce, Virtuemart
Difusión de contenido multimedia	Piwigo, Zenphoto, Gallery, Coppermine
Propósito general	Joomla, Drupal, CMS Made Simple, DornCMS

Tabla 1 Clasificación y ejemplos de sistemas de gestión de contenidos.

Específicamente los CMS dedicados a la enseñanza se conocen además por el acrónimo LMS[2] o sistemas para la gestión del aprendizaje. Estos no solo implementan las funcionalidades básicas para la gestión del contenido, sino que además gestionan cursos, hacen un seguimiento detallado del trabajo del estudiante en la herramienta y permiten la comunicación vía correo, foro o chat, favoreciendo así el aprendizaje colaborativo (14).

Gestión de contenidos en hiperentornos educativos

El proceso de enseñanza-aprendizaje es muy complejo, ya que debe tener en cuenta factores objetivos y subjetivos para que los estudiantes adquieran el conocimiento de la mejor manera posible. Con el progreso de las TICs se ha logrado llevar a las aulas herramientas como los hiperentornos educativos y los LMS.

Cuba se ha insertado en esta experiencia implementando soluciones tanto para la exportación como para ser implantadas en el país. Estas soluciones han servido como antecedente a la versión multiplataforma de la colección de software educativo El Navegante. A continuación se detallan

[2] *Del inglés Learning Management System.*

algunos elementos sobre las herramientas para la administración del contenido en la colección de software educativo Multisaber en su versión multiplataforma.

Colección de software educativo Multisaber

La versión multiplataforma de la colección de software educativo Multisaber cuenta con catorce productos. Cada uno de ellos es un hiperentorno educativo con alto volumen información integrada en los siguientes módulos: Temas, Ejercicios, Juegos, Mediateca, Resultados y Maestro (15). La principal deficiencia que tiene el proceso de gestión de contenidos en este software es la utilización de tres herramientas: el panel de administración de Joomla para el contenido de los módulos Temas y Mediateca, junto a otras dos creadas por el equipo de desarrollo para gestionar las tipologías de los juegos y los ejercicios.

El sistema gestor de bases de datos que usa Multisaber es MySQL, por lo que el panel de administración de Joomla con algunas modificaciones se adaptó perfectamente a las necesidades de los módulos Temas y Mediateca. Esta alternativa se heredó de la versión anterior del software, desarrollada por el Ministerio de Educación (MINED).

Joomla como sistema gestor de contenido de propósito general no permite administrar las tipologías de los juegos y los ejercicios propias de Multisaber. Al evaluar la factibilidad de la incorporación de módulos a Joomla y la de implementar nuevos módulos aislados para realizar esta tarea, se determinó que la primera opción tendría un impacto negativo en el cumplimiento del cronograma, debido al excesivo gasto de tiempo que esto implicaba. De ahí que quedara la gestión de los ejercicios y los juegos en sistemas independientes.

Basado en esta experiencia se decidió integrar todos los módulos de gestión del contenido de la versión multiplataforma de la colección de software educativo El Navegante en una sola herramienta: un panel de administración. Para guiar el desarrollo de la solución que se propone es imprescindible el uso de una metodología para desarrollar software que controle estrictamente cada uno de los procesos que lleva a cabo el equipo de trabajo.

Metodología de desarrollo de software

Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto software. Dichas metodologías pretenden guiar a los desarrolladores al crear un nuevo software, pero

los requisitos de un software a otro son tan variados y cambiantes, que ha dado lugar a que exista una gran variedad de metodologías para la creación del software (16).

Estas tecnologías proporcionan una serie de elementos que permiten gestionar y controlar el proyecto, definir las herramientas a utilizar, dividir el proyecto en etapas y precisar qué hacer en cada una de ellas. Sin embargo no existe una metodología que sea genérica para el proceso de desarrollo de software ya que las condiciones objetivas y subjetivas del producto y del equipo de trabajo son muy variables.

Las metodologías orientadas al control riguroso de los procesos se clasifican como pesadas, mientras que las orientadas a la interacción con el cliente y el desarrollo incremental del software son las denominadas ágiles (16) (17) (18). Debido a los intereses del proyecto al que pertenece la solución que se implementa se decide el uso de la metodología tradicional RUP[3] para que el expediente de proyecto cuente con una documentación exhaustiva sobre el proceso de desarrollo.

Proceso Unificado de Rational

RUP es un marco genérico que puede especializarse para una variedad de tipos de sistemas, diferentes áreas de aplicación, tipos de organizaciones, niveles de aptitud y diferentes tamaños de proyectos (18). Esta metodología de desarrollo se ha hecho muy popular entre las tradicionales por el nivel de detalle de la documentación que genera sirviendo de base al equipo de trabajo para dar soporte a la solución implementada, además es adaptable a pequeños y grandes proyectos.

RUP divide el desarrollo del proyecto en fases y flujos de trabajo como se indica en la Figura 1 Fases y flujos de trabajo de RUP.:

[3] *Del inglés Rational Unified Process.*

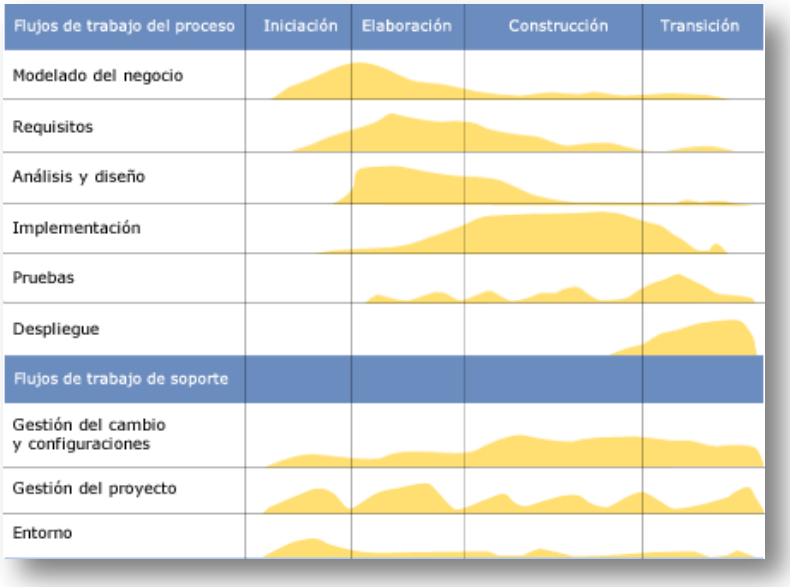


Figura 1 Fases y flujos de trabajo de RUP.

Según (18) y (19) las principales características de esta metodología de desarrollo son las que a continuación se explican:

Dirigido por casos de uso:

Un caso de uso es un segmento de funcionalidad que aporta un resultado concreto al usuario; constituye además una guía del proceso de desarrollo a través de sus fases y flujos (18). A grandes rasgos, identificar estos casos de uso es el punto de partida para el desarrollo del sistema y apoyándose en ellos se crean una serie de modelos de diseño e implementación que dan soporte a la solución.

El software en cuestión engloba gran cantidad de funcionalidades. Para llevarlas a cabo, fueron identificados y detalladamente documentados los casos de uso, además, los modelos obtenidos representan la realización de los mismos y son el resultado de cada flujo de trabajo.

Centrado en la arquitectura:

La arquitectura de un sistema software se describe mediante diferentes vistas del sistema en construcción. El concepto de arquitectura de software incluye los aspectos estáticos y dinámicos más significativos del sistema (18). Los casos de uso implementados fueron agrupados de acuerdo al impacto que tenían sobre la concepción del producto y en la arquitectura del mismo. Esta organización

del sistema que se logra a través de la arquitectura permitió al equipo de trabajo tener visión general del producto final para comprenderlo y desarrollarlo en tiempo y forma.

Iterativo e incremental:

Esta característica de la metodología fue primordial para arribar a un resultado concreto debido a que se dividió el proyecto en partes más pequeñas para implementar en cada iteración el conjunto de casos de uso que amplían la funcionalidad, disminuyendo así el impacto de los riesgos más significativos. Desarrollar por iteraciones permitió que tras cada una de ellas el equipo de trabajo pudiera ver resultados a corto plazo y hacer una planificación más realista de la iteración siguiente (18).

Lenguaje de Modelado Unificado

Como lenguaje de modelado RUP propone UML[4]. El Lenguaje Unificado de Modelado prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan (20). El uso de esta tecnología permitió especificar, visualizar, construir y documentar los artefactos generados durante cada flujo de trabajo. Es importante señalar que como lenguaje de modelado, UML indica qué debe hacerse y no cómo debe hacerse, por lo que es un error conceptual confundirlo con una metodología para desarrollar software.

Herramienta CASE

Las herramientas de ingeniería de software asistida por ordenador o herramientas CASE[5] son elementos indispensables para concebir una aplicación informática. El uso de las mismas reduce los gastos y el tiempo en el ciclo de desarrollo del software ya que permiten modelar fácilmente procesos y sistemas, con previo dominio de al menos un lenguaje de modelado soportado por la herramienta (21).

[4] *Del inglés Unified Modeling Language.*
[5] *Del inglés Computer Aided Software Engineering.*

Visual Paradigm

El proyecto al que pertenece la solución que se implementa utiliza la versión 3.5 de Visual Paradigm. Esta herramienta CASE proporciona abundantes tutoriales y demostraciones interactivas para la realización de proyectos bajo la metodología RUP, permitiendo modelar todos y cada uno de los diagramas que incorpora UML (21). La principal razón por la que fue seleccionada esta herramienta fue por la necesidad del uso de tecnologías libres durante el desarrollo del sistema para gestionar el contenido de la colección de software educativo El Navegante. Por otro lado es importante destacar que se integra con el entorno de desarrollo NetBeans7.0 que es el seleccionado por el equipo de trabajo para la codificación del software, además permite generar la documentación en diferentes formatos como: *.jpeg, .html, .pdf* (21).

Lenguajes de desarrollo

Un lenguaje de programación es una técnica estándar de comunicación que permite expresar las instrucciones que han de ser ejecutadas en una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen un lenguaje informático (22).

Partiendo de las características del software a desarrollar (aplicación web con soporte de datos), se hace necesaria la selección de un conjunto de lenguajes de desarrollo web que permitan cumplir con los requerimientos propuestos. Para ello se deben tener en cuenta un conjunto de variables que permitirán escoger el lenguaje de desarrollo idóneo para la implementación del software, como son: la familiaridad y conocimiento de las estructuras del lenguaje, soporte de los IDE[6] para el lenguaje escogido, así como el dominio de la herramienta escogida. La selección del lenguaje es una de las tareas más importantes realizadas por el equipo de trabajo durante el proceso de desarrollo del software porque influye en la decisión sobre el gestor de base de datos y sobre las herramientas para automatizar el proceso de desarrollo.

En la actualidad existe un gran número de lenguajes de programación que dan soporte a las tendencias actuales para las aplicaciones web entre las que se encuentran la utilización de AJAX[7], mayor interacción con el usuario, manejo y utilización de elementos multimedia (imágenes, videos, flash, ext.) y los requerimientos de cada uno de los escenarios de la aplicación. Estos lenguajes

[6] *Del inglés Integrated Development Environment.*
[7] *Del inglés Asynchronous JavaScript and XML.*

pueden dividirse en dos grandes grupos, los lenguajes del lado del cliente y los lenguajes del lado del servidor.

Lenguajes del lado del cliente

Los lenguajes del lado del cliente son aquellos que basan su procesamiento en el cliente web, o sea, no son compilados o interpretados por el servidor, sino que son devueltos por salida estándar en el proceso de una petición cliente-servidor, para ser interpretados por el navegador del usuario. Los criterios para la selección de las versiones más recientes de los lenguajes del lado del cliente para el desarrollo de la aplicación, estuvieron basados principalmente en el grado de conocimiento y habilidades del equipo de desarrollo con dichos lenguajes, la cantidad de material de consulta disponible sobre los mismos y en la popularidad de estos para su uso en aplicaciones web a nivel mundial.

Esta decisión de utilizar versiones actualizadas se tomó para dar cumplimiento a los requerimientos necesarios y lograr una aplicación acorde a las últimas tendencias y uso de estándares en la web; aclarando que solo se utilizarán componentes, características y funcionalidades de los lenguajes compatibles e interpretados por las versiones de los navegadores recomendados para el uso de la aplicación, logrando de esta forma un comportamiento y visión estándar del software en dichos navegadores.

HTML5

HTML5 es la actualización de HTML, el lenguaje en el que es creada la web. Definido formalmente por un cuerpo de normas internacionales conocido como World Wide Web Consortium (W3C), HTML5 consta de más de cien especificaciones que se relacionan con la nueva generación de tecnologías web. De hecho, HTML5 es un término aglutinador que describe un conjunto de especificaciones de HTML, CSS y JavaScript diseñado para permitir que los desarrolladores creen la nueva generación de sitios y aplicaciones web (23). Es una plataforma de código abierto desarrollado en términos de derechos de licencia libre (24).

La utilización de HTML5 como uno de los lenguajes para el desarrollo, responde a las necesidades de manipulación y maquetación de los elementos visuales de la aplicación como es el caso de los videos y los sonidos, utilizando para ellos las etiquetas provistas en sus especificaciones: <*video*> y <*audio*> que permiten a la aplicación una interacción más potente. La utilización de <*header*>, <*section*>,

<*nav*>, <*aside*>, entre otras etiquetas semánticas permite a los desarrolladores diseñar bajo estándares y buenas prácticas de programación web.

CSS3

Las hojas de estilo en cascada o CSS[8] son un mecanismo sencillo para añadir estilo (por ejemplo, fuentes, colores, espacios) a los documentos Web (25). El W3C es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores. En diferencia a CSS2, que fue una gran especificación que definía varias funcionalidades, CSS3 está dividida en varios documentos separados, llamados *módulos*. Cada módulo añade nuevas funcionalidades a las definidas en CSS2, de manera que se preservan las anteriores para mantener la compatibilidad.

Se decide utilizar CSS3 en la implementación ya que ofrece una gran variedad de nuevas formas de crear un impacto con los diseños, siendo además la mejor vía de separar los contenidos de la presentación; permite definir el aspecto de todos los contenidos, es decir, el color, tamaño y tipo de letra de los textos (25). Además mantiene el propósito del equipo de desarrollo de crear una aplicación que presente elementos de las tendencias actuales de diseño en la web.

JavaScript

JavaScript es un lenguaje de script multiplataforma basado en objetos. Es un lenguaje pequeño y ligero; no es útil como un lenguaje independiente, más bien está diseñado para una fácil incrustación en otros productos y aplicaciones, tales como los navegadores web. Dentro de un entorno anfitrión, JavaScript puede ser conectado a los objetos de su entorno para proveer un control programable sobre estos (26).

Este lenguaje, uno de los más populares entre los de la familia de script para el desarrollo de páginas web, permite agregarle más funcionalidad a las interfaces de usuario de la aplicación, haciéndolas más amigables. Además de estas características se decide utilizar este lenguaje, ya que permite validar los campos de los formularios y las acciones de los usuarios antes de enviar las peticiones al servidor; este método no es totalmente seguro, pero agrega un valor de funcionalidad importante al software, evitando las peticiones innecesarias al servidor. Entre otras ventajas de la utilización de este lenguaje que se aprovechan en el desarrollo del producto se encuentran las animaciones, las peticiones

[8] *Del inglés Cascading Style Sheets.*

mediante AJAX al servidor y la existencia de frameworks (o marco de trabajo) de desarrollo que agilizan y simplifican el trabajo con el mismo.

Lenguajes del lado del servidor

Los lenguajes del lado del servidor son aquellos que son procesados por el servidor y generan las páginas que son devueltas al cliente en cada petición. Además se encargan de manipular la información que persiste en la base de datos, la seguridad del sistema y en la mayoría de los casos realizan la lógica del negocio de la aplicación.

Para la selección del lenguaje del lado del servidor se realizó una comparación entre tres de los lenguajes más populares entre los desarrolladores web:

- **PHP5:** PHP[9] es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas web dinámicas, embebidas en páginas HTML y ejecutadas en el servidor. PHP no necesita ser compilado para ejecutarse. Para su funcionamiento necesita tener instalado Apache o IIS[10] con las librerías de PHP. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas (27).
- **Java:** Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros (28).
- **ASP.NET:** Este es una tecnología comercializado por Microsoft, y usado por programadores para desarrollar entre otras funciones, sitios web. ASP.NET es el sucesor de la tecnología ASP, fue lanzado mediante una estrategia de mercado denominada .NET. El ASP.NET fue desarrollado para resolver las limitantes que tenía su antecesor ASP. Creado para desarrollar web sencillas o grandes aplicaciones. Para el desarrollo de ASP.NET se puede utilizar C#, VB.NET o J#. Para publicar las páginas web necesita tener instalado IIS con el Framework .Net (27).

Estos lenguajes gozan de gran popularidad entre los grupos de desarrollo de aplicaciones web de gestión. Inicialmente se comenzó a realizar el filtro de la selección teniendo en cuenta las características que debe tener el lenguaje escogido para que pueda resolver los problemas

[9] *Del inglés PHP Hipertext Preprocessor.*
[10] *Del inglés Internet Information Services.*

planteados. Entre estas características se encuentran la posibilidad de conexión con los gestores de base de datos libres más utilizados (versiones libres de MySQL y PostgreSQL, siendo este último un requisito no funcional de la colección de software educativo El Navegante en su versión multiplataforma). Además debe incluir gran cantidad de funciones nativas que agilicen el proceso de desarrollo, ser un lenguaje orientado a objetos, libre y multiplataforma y contar con algún framework de desarrollo.

Luego de realizar un estudio sobre los diferentes lenguajes de programación escogidos para la comparación, se concluye que de una forma u otra los tres lenguajes de programación cuentan con las características fundamentales para el proceso de desarrollo del software, a pesar de la superioridad de unos sobre otros en algunos aspectos.

Como segundo paso de la comparación se tuvieron en cuenta criterios que podían incidir tanto en la calidad del producto final, como en el cumplimiento del cronograma. También fueron valoradas las necesidades del proceso de desarrollo y las capacidades de los programadores, aspectos que se muestran como un punto de vista subjetivo en la siguiente tabla:

Criterios	PHP5	JAVA	ASP.NET
Dominio del lenguaje y las tecnologías que engloba[11]	Sí	No	No
Referencia de desarrollos en el centro	Sí	Sí	No
Reutilización de componentes ya implementados en el proyecto	Sí	No	No
Legibilidad del código[12]	Sí	Sí	Sí
Frameworks de desarrollo multiplataforma	Sí	Sí	Sí

Tabla 2 Comparación entre PHP, Java y ASP.NET

Basado en la comparación anterior y teniendo en cuenta el domino de habilidades por parte de los desarrolladores se decidió utilizar como lenguaje del lado del servidor a PHP5, permitiendo poner en práctica los conocimientos adquiridos durante dos años desempeñando el rol de desarrollador en el proyecto Multisaber-Navegante. Además permite acoplarse a la arquitectura de la versión

[11] *Dígase configuración de servidor web o dominio de un framework de desarrollo para el lenguaje en cuestión.*
[12] *La legibilidad del código puede verse afectada al mezclar sentencias HTML y PHP.*

multiplataforma de El Navegante desarrollada con PHP5, logrando reutilizar componentes e integrar ambas aplicaciones.

PHP5 es el lenguaje de desarrollo utilizado por las aplicaciones similares analizadas en el estudio del estado del arte, es el lenguaje del lado del servidor más frecuente en proyectos pertenecientes al centro FORTES, siendo estos dos criterios de gran peso a favor de su selección ya que permiten acortar el período de desarrollo del software, de manera que la ganancia de tiempo supera con creces las desventajas del mismo.

Frameworks de desarrollo

En la actualidad es una tendencia en el desarrollo de software la utilización de frameworks (o marcos de trabajo). Utilizarlos acelera el proceso de desarrollo, reutilizar el código ya existente, promover buenas prácticas de desarrollo con el uso de patrones y mejorar la seguridad de la aplicación. Un framework se puede considerar como una aplicación genérica incompleta y configurable a la que se pueden añadir las últimas piezas para construir una aplicación concreta (29).

Un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas (29). Un framework web, por tanto, se puede definir como un conjunto de componentes que forman parte de un diseño reutilizable que facilita y agiliza el desarrollo de sistemas web (30).

Existen frameworks que permiten desarrollar diversos tipos de aplicaciones, desde videojuegos hasta aplicaciones médicas. Los frameworks se clasifican en dependencia de su principal funcionalidad, por tal motivo se decide que para el desarrollo de la aplicación debe utilizarse un framework para la interfaz de usuario o capa de presentación, uno para la modularización o capa de lógica de negocio y uno para la capa de acceso a datos.

Capa de presentación

Al ser JavaScript uno de los lenguajes del lado del cliente, se sobrentiende la utilización de un framework desarrollado para este lenguaje para la capa de presentación. Dando respuestas a las necesidades y requisitos de la aplicación se necesita seleccionar un framework que permita mostrar de forma dinámica los contenidos, aumentando la interactividad y experiencia del usuario.

Otros requisitos indispensables del marco de trabajo que se seleccione deben ser la manipulación del árbol de objetos dinámicos (DOM[13]), el manejo de eventos y animaciones fácilmente, así como la interacción con la tecnología AJAX. Además debe ser compatible con las principales versiones de los navegadores web más utilizados. Debe ser de código abierto, libre y gratis.

Teniendo en cuenta que los frameworks jQuery y Prototype presentan las características anteriores, se realizó una comparación subjetiva entre ellos, teniendo como premisas las necesidades, competencias y condiciones del equipo de desarrollo. Esta comparación arrojó como resultado el framework jQuery en su versión 1.5.1 debido a las ventajas que proporciona la utilización del mismo para el desarrollo de la aplicación:

- Ahorra líneas de código JavaScript escrito por los desarrolladores.
- Es fácil de usar y aprender, además de ser el framework JavaScript más conocido por los desarrolladores de la aplicación ya que se cuenta con experiencia de sistemas anteriores implementados por el equipo de trabajo utilizando dicho framework.
- Se integra con el lenguaje PHP y permite manejar JSON[14], XML[15] y CSS.
- Presenta soporte a extensiones o plugins, los cuales amplían las facilidades de uso y mejoran los tiempos de producción al permitir reutilizar componentes desarrollados por terceros.
- Posibilita usar jQuery UI[16] en su versión 1.8, una librería de código abierto de componentes de interfaz de usuario (con interacciones, widgets de todas las funciones y efectos de animación) basado en la librería jQuery. Cada componente está construido de acuerdo al evento impulsado por la arquitectura de jQuery (encontrar algo y manipularlo) y con diseños modificables, por lo que es fácil para integrar y ampliar en su propio código por desarrolladores de cualquier nivel (31).
- Existe una amplia documentación y comunidad de usuarios, aspectos estos muy importantes en la retroalimentación y rectificación de errores con el uso de esta herramienta.
- Permite mantener el propósito de acoplar la aplicación a la arquitectura de software definida para la versión multiplataforma de El Navegante, lo cual permitirá reutilizar componentes definidos por el equipo de trabajo.

[13] *Del inglés Document Object Model.*
[14] *Del inglés JavaScript Object Notation.*
[15] *Del inglés eXtensible Markup Language.*
[16] *Del inglés User Interface.*

Puede definirse a jQuery como una biblioteca de JavaScript rápida y concisa que simplifica el documento HTML: manejando eventos, animaciones, y las interacciones AJAX para el desarrollo web rápido. jQuery está diseñado para cambiar la forma en que se escribe JavaScript (32).

Capa de lógica de negocio

Debido a que la selección del lenguaje del lado del servidor es PHP5, se necesita un framework orientado a objeto desarrollado en este lenguaje y capaz de dar solución a las necesidades de la implementación del producto. En la actualidad existen gran cantidad de frameworks de desarrollo creados con PHP5 y que gozan de mucha popularidad a la hora de seleccionarlos para su uso en un proyecto determinado.

Entre estos frameworks puede citarse Symfony, CakePHP, Zend Framework y CodeIgniter. Para el desarrollo de la aplicación se seleccionó el framework Symfony en su versión 1.4.3 debido a la experiencia del equipo de desarrollo utilizándolo en el proyecto productivo Multisaber-Navegante y a las ventajas que representa su uso frente a los demás para resolver las necesidades y requisitos del proceso de desarrollo del software, además de contar con una amplia comunidad de usuarios, siendo esto importante para la retroalimentación, superación y reparación de errores.

Symfony

Symfony es un framework PHP que facilita el desarrollo de las aplicaciones web. Se encarga de todos los aspectos comunes como las configuraciones, validaciones, manejo de formularios, entre otros, dejando que el programador se dedique a crear las funcionalidades específicas de cada proyecto. Actualmente existe mucha información referente al trabajo con este framework, ya que están disponibles miles de páginas de documentación distribuidas en varios libros gratuitos y decenas de tutoriales (33).

Junto a las características anteriores, la principal ventaja de la utilización de Symfony es la integración que permite entre la aplicación desarrollada y la versión multiplataforma de El Navegante, permitiendo adaptarse a la arquitectura de dicho software, ya que Symfony considera un proyecto como un conjunto de servicios y operaciones disponibles bajo un determinado nombre de dominio y que comparten el mismo modelo de objetos (29).

Dentro del framework, las aplicaciones se agrupan de una forma lógica e independiente como un proyecto Symfony, permitiendo compartir el modelo de datos y las librerías. Al mismo tiempo están compuestas por módulos, los cuales representan una página web o un grupo de páginas con un

propósito relacionado. Los módulos almacenan las acciones que representan a cada una de las posibles operaciones; trabajar con estas es muy similar a trabajar con páginas de una aplicación web tradicional. A estas acciones se les puede hacer corresponder una vista específica o pueden retornar la respuesta en formato de texto.

Dichas características proporcionan una estructura al código fuente que permite escribir un código más legible para un futuro mantenimiento. Además, el patrón de arquitectura modelo-vista-controlador implementado por dicho framework, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web, logrando desarrollar un producto bajo buenas prácticas de desarrollo de aplicaciones web actuales.

Otras necesidades a resolver por este framework en el proceso de implementación son: la capacidad de lograr una aplicación de software libre instalable en las plataformas Unix, Linux y Windows; además de lograr una aplicación extensible, explotando al máximo el mecanismo de plugins. La aplicación desarrollada debe ser segura, y Symfony realiza gran parte del trabajo para lograr ese objetivo, protegiendo al sistema contra ataques de tipo XSS[17], CSRF[18], etcétera.

Los frameworks de ruta y formulario contenidos en Symfony ahorran gran parte del trabajo de implementación, logrando direcciones amigables y una correcta validación de los datos enviados por los usuarios. Además cuenta con un conjunto de líneas de comando que realizan tareas automatizadas ahorrando tiempo de implementación, por ejemplo, permite crear nuevos módulos perfectamente funcionales integrando interacción con la base de datos mediante CRUD[19] sin necesidad de crear código fuente.

La independencia de la base de datos utilizada, el soporte para la integración con un mapeador de objetos relacional (se logra a través de un potente framework de acceso a datos) y la integración con jQuery, son otras de las características que influyeron en la decisión de la utilización de dicho framework por parte del equipo de desarrollo.

Por último y no menos importante vale destacar el framework para pruebas con que cuenta Symfony, lo cual significa una potente herramienta utilizada por los desarrolladores, permitiendo encontrar errores y acortar el tiempo de desarrollo.

---

[17] *Del inglés Cross Site Scripting.*
[18] *Del inglés Cross Site Request Forgery.*
[19] *Del inglés Create, Retrieve, Update y Delete.*

Capa de acceso a datos

La utilización de un ORM[20] en el desarrollo de un software responde a la necesidad de tener una interfaz que traduzca la lógica de los objetos a la lógica relacional, o sea, poder acceder de forma efectiva a la base de datos desde un contexto orientado a objetos. En otras palabras, un ORM es una técnica de programación que permite convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, es decir, las tablas de la base de datos pasan a ser clases y los registros, objetos que podemos manejar con facilidad (34).

Los ORMs se sitúan en la parte superior de la capa de abstracción de la base de datos, lo que permite abstraerse del gestor utilizado. Además resuelven un conjunto de tareas comunes y de mantenimiento en la aplicación como son (35):

- **Reutilización:** La reutilización permite invocar a los métodos de un objeto de datos desde distintas partes de la aplicación e incluso desde diferentes aplicaciones.
- **Encapsulamiento:** La capa ORM encapsula la lógica de los datos pudiendo hacer cambios que afectan a toda la aplicación únicamente modificando una función.
- **Portabilidad:** Un ORM permite cambiar el gestor de base de datos en cualquier momento del desarrollo de la aplicación, sin que ocurran grandes cambios en la implementación. Esto se debe a que no se utilizan sintaxis propias del lenguaje de la base de datos actual, sino una sintaxis propia del ORM que es capaz de traducir a diferentes tipos de bases de datos.
- **Seguridad:** Los ORM suelen implementar mecanismos de seguridad que protegen la aplicación de los ataques más comunes como inyecciones SQL.
- **Mantenimiento del código:** Gracias a la correcta organización de la capa de datos, modificar y mantener el código del proyecto es una tarea sencilla.

Por tal motivo es necesaria la utilización de un ORM para el lenguaje PHP5 y que sea soportado por el framework Symfony 1.4.3.

Este último presenta soporte e integración con Propel y Doctrine (29), ambos con características similares y que resuelven las necesidades existentes en la aplicación ya expresadas anteriormente. Luego entonces, se decide Doctrine como ORM debido a que es el utilizado en el proyecto con el que

[20] *Del inglés Object Relation Mapper.*

se integra el presente trabajo, lo cual permitirá reutilizar el modelo de datos generado junto con sus funcionalidades. Además de ser rápido y eficaz, posee una amplia documentación y presenta soporte para migraciones. Doctrine implementa los métodos mágicos que facilitan aún más las consultas a la base de datos, permitiendo obtener información a partir de cualquier campo de una tabla.

Doctrine es un ORM para PHP 5.2.3 y posterior basado en YAML[21]. Además de todas las ventajas que conlleva un ORM, uno de sus puntos fuertes es su lenguaje DQL[22] inspirado en el HQL[23] de Hibernate. Soporta validación de datos en los modelos y relaciones entre ellos (34).

Entorno de desarrollo integrado

Un entorno de desarrollo integrado (IDE) es un programa compuesto por una serie de herramientas, tales como editor de texto, compilador, intérprete, depurador, sistema de ayuda para la construcción de interfaces gráficas de usuario, etc., que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos (36).

La selección de un IDE para la codificación, debe realizarse de forma que cumpla con las características que se listan a continuación:

- Debe ser un IDE con versión disponible para sistemas operativos Linux.
- Debe tener soporte y completamiento de código para los lenguajes definidos para la implementación del software.
- Debe tener soporte, integración y completamiento de código para los frameworks que se usarán para el desarrollo del software.
- Debe permitir crear nuevos proyectos y aplicaciones directamente desde el IDE.
- Debe tener integración con sistemas de control de versiones como SVN[24].
- Los requerimientos de hardware deben estar acorde a las condiciones de desarrollo presentes.

Los entornos de desarrollo analizados tras estas consideraciones fueron:

[21] *Del inglés YAML Ain't Markup Language.*
[22] *Del inglés Doctrine Query Language.*
[23] *Del inglés Hibernate Query Language*.
[24] *Abreviatura de Subversion.*

- **Eclipse 3.5.1:** Es un IDE de código abierto independiente de la plataforma, es una aplicación de cliente enriquecido, emplea plugins para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Eclipse es, en el fondo, únicamente un armazón sobre el que se pueden montar herramientas de desarrollo. La arquitectura de plugins permite, además de integrar diversos lenguajes, introducir otras aplicaciones (37).
- **NetBeans IDE 7.0:** En un entorno de desarrollo integrado libre y de código abierto para desarrolladores de software. Contiene todas las herramientas necesarias para crear aplicaciones profesionales de escritorio, empresariales, web y aplicaciones móviles con la plataforma Java, así como con C/C++, PHP, JavaScript y Groovy (2). Entre sus características se encuentra un sistema de proyectos basado en control de versiones y refactorización. Todas las funciones del IDE son provistas por plugins, al igual que Eclipse.

Finalmente se decide seleccionar a NetBeans IDE 7.0 como entorno de desarrollo integrado para utilizar durante la implementación de la aplicación. Esta selección está respaldada por los siguientes criterios: cumple con las características requeridas, de ser este IDE el utilizado en el desarrollo del proyecto productivo Colección de Software Educativo Multisaber-Navegante, además el equipo de desarrollo está familiarizado y tiene experiencia en el trabajo con el mismo.

NetBeans IDE 7.0 se destaca por brindar soporte nativo a frameworks como jQuery y Symfony (tanto para el completamiento de código como para las tareas desarrolladas mediante líneas de comando), los cuales fueron previamente seleccionados para el desarrollo de la aplicación; por otro lado brinda completamiento de código para los métodos mágicos que usa el ORM Doctrine.

Sistema gestor de base de datos

Un sistema gestor de base de datos (SGBD) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos.

Entre las características que debe presentar un SGBD se encuentra: abstracción de la información ahorrando detalles del almacenamiento físico de los datos al usuario, independencia de las aplicaciones que se sirvan de sus datos, redundancia mínima evitando repetir información,

consistencia, o sea, que todos los datos repetidos se actualicen de forma simultánea, seguridad frente a usuarios malintencionados o ataques que intenten destruir o acceder a la información, integridad al garantizar la validez de los datos almacenados, respaldo y recuperación, además de controlar el acceso concurrente a la información, que podría derivar en inconsistencias (2).

Para esta herramienta no se realizó ninguna comparación entre los diferentes SGBD disponibles debido a que la aplicación es un panel de administración para un software en desarrollo y por tanto debe trabajar sobre el gestor de base de datos definido en el documento de arquitectura de software de la versión multiplataforma de El Navegante: PostgreSQL en su versión 8.4 (2).

PostgreSQL 8.4 es un sistema de gestión de base de datos objeto-relacional. Es compatible con gran parte del estándar SQL y ofrece muchas características modernas como consultas complejas, llaves externas, disparadores (triggers), vistas, integridad de las transacciones y control de recurrencia. Además de permitirle al usuario adicionar nuevos tipos de datos, funciones, operadores y lenguajes de procedimiento. Y debido a la licencia liberal, PostgreSQL puede ser usado, modificado y distribuido por cualquiera de forma gratuita para cualquier propósito, ya sea privado, comercial o académico (38).

Para acceder y manipular los datos almacenados durante el desarrollo de la aplicación y realizar tareas de mantenimiento, se decidió utilizar PgAdmin III como cliente del SGBD para PostgreSQL 8.4. Esta decisión no estuvo acompañada de ninguna comparación previa debido a que su utilización no es decisiva para el desarrollo del software, además existía conocimiento por parte del equipo de desarrollo sobre el trabajo con la herramienta.

Servidor web

Se decidió escoger como servidor web a Apache 2.0 ya que es el servidor web con licencia de software libre y de código abierto más usado, según estadísticas desde agosto de 1995 hasta febrero del 2011 (39).

Apache 2.0 contiene un conjunto de características y funcionalidades indispensables para que la aplicación desarrollada pueda funcionar correctamente. Entre estas características se encuentran:

- Soporta PHP5.
- Es altamente configurable, rápido, flexible, eficiente y adaptado a los nuevos protocolos web (40).

- Su condición de ser multiplataforma permite mantener este servidor web en todos los sistemas operativos en los que se despliegue el software.
- Es un servidor seguro que permite protección de ficheros.

Vale destacar que cuenta con una gran comunidad de usuarios que facilita encontrar solución tanto a errores del propio servidor como a errores en la configuración del mismo.

Conclusiones del capítulo

El análisis de conceptos asociados al objeto de estudio y de aplicaciones similares a la que se desarrolla en la investigación, arrojó como resultado que es necesario mejorar la gestión de contenidos en la versión multiplataforma de El Navegante. Estas mejoras deben estar sustentadas en el uso de tecnologías libres y para ello se hizo un estudio detallado de las principales tendencias en lenguajes, metodologías y demás herramientas para el desarrollo de aplicaciones web con soporte de datos. De ellos fueron seleccionados los que estaban en concordancia con la arquitectura previamente definida en el proyecto al que pertenece la solución.

DESCRIPCIÓN DEL SISTEMA

Introducción

El levantamiento de requisitos, la identificación de los casos de uso y la descripción de los mismos son tareas que tienen un impacto significativo en la caracterización del sistema que se desea implementar. Tomando como punto de partida el modelo de dominio, en este capítulo se relacionan detalladamente los requerimientos funcionales y no funcionales del software junto a las descripciones de los casos de uso del sistema.

Modelo del dominio

Un modelo de dominio asocia todas las entidades o conceptos que se manejan en el entorno en el que trabaja el sistema mediante un diagrama de clases UML (18). Son usados frecuentemente en ambientes donde no están bien definidos los procesos de negocio, siendo este el caso de la presente investigación.

Análisis de los conceptos del dominio

Colección: específicamente colección El Navegante, es el conjunto de productos (hiperentornos educativos) que abordan diferentes temáticas para apoyar el aprendizaje en estudiantes de secundaria básica.

Producto: es cada software que compone la colección El Navegante. Aborda un tema relacionado con una o varias áreas del conocimiento de los estudiantes de secundaria básica.

Ejercicio: relacionan una amplia gama de tipologías de ejercicios dedicados a comprobar los conocimientos adquiridos por los estudiantes en el software.

Juego: relacionan una amplia gama de tipologías de juegos que el estudiante puede realizar para profundizar algún contenido.

Medias: comprende todos los elementos audiovisuales del software clasificados de la siguiente forma: animaciones, diaporamas, sonidos, videos e imágenes; además de las personalidades.

Contenidos: comprende una selección de artículos vinculados con la temática que aborda el producto.

Ayuda: abarca los contenidos que podrán auxiliar a los estudiantes y maestros a interactuar correctamente con cada uno de los módulos y servicios de la colección.

Efemérides: conjunto de acontecimientos históricos que pueden ser de interés para el estudiante.

Usuario: persona encargada de gestionar el contenido en el sistema.

Administrador: persona que además de gestionar el contenido tiene privilegios para gestionar los productos de la colección, los usuarios, así como consultar las trazas de los mismos.

Traza: registro que se genera cuando cualquier usuario al realizar una acción sobre alguna entidad (adicionar, eliminar o modificar) del sistema.

Personalidad: elemento de la mediateca que relaciona información sobre un individuo relevante para el tema que aborde un producto determinado.

Glosario: relación de términos de un producto que pueden ser de dudoso significado para un estudiante.

Imagen: fotografía, retrato, dibujo, etc. que sirve de información visual para enriquecer los contenidos del producto.

Video: representación de una secuencia de imágenes sobre algún proceso o suceso relacionado con los contenidos del producto.

Sonido: archivo de audio para enriquecer los contenidos de la colección.

Diaporama: conjunto de imágenes que se reproducen una detrás de otra acompañadas de algún elemento de audio.

Animación: representación del movimiento de algún elemento que por definición es estático, también puede tener algún elemento de audio asociado.

Diagrama del modelo del dominio

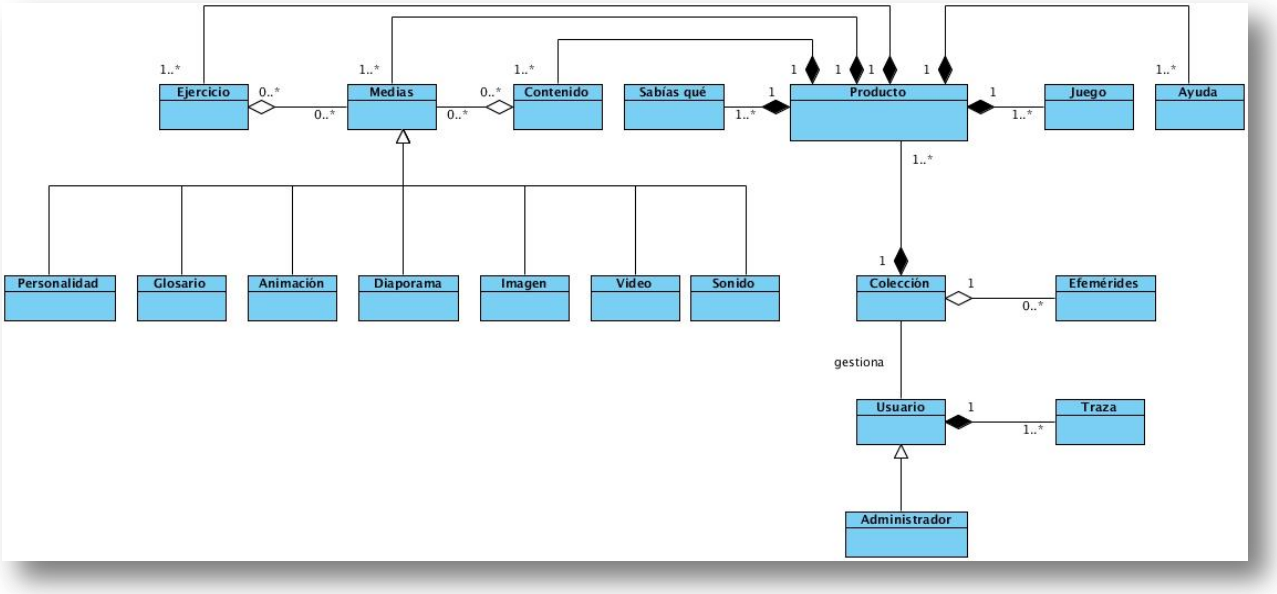


Figura 2 Modelo del dominio.

Especificación de requisitos

En numerosas ocasiones, durante el desarrollo de software, existen contradicciones entre el equipo de desarrollo y el cliente, debido a que no existe un entendimiento en cuanto a las condiciones y capacidades que debe cumplir el sistema. Esto viene dado porque las necesidades del cliente no son fáciles de discernir; lo cual obliga a contar con algún modo de capturar las necesidades del cliente y los usuarios de forma que puedan comunicarse fácilmente a todas las personas implicadas en el proyecto.

Entre las características de RUP se encuentra que es dirigido por caso de usos. Para obtener el modelo de caso de uso y además lograr un entendimiento total de las condiciones y características de la aplicación, los desarrolladores comienzan realizando una captura de los requisitos que el cliente solicita para el producto final.

La captura de requisitos es un flujo de trabajo cuyo propósito esencial es orientar el desarrollo hacia el sistema correcto. Esto se lleva a cabo mediante la descripción de los requisitos del sistema de forma tal que se pueda llegar a un acuerdo entre el cliente (incluyendo los usuarios) y los desarrolladores del sistema, acerca de lo que el sistema debe hacer y lo que no. Se entiende por requisito como la

condición o capacidad que debe cumplir un sistema. Teniendo en cuenta sus características, los requisitos se clasifican en: funcionales y no funcionales (18).

Un requisito funcional especifica una acción que debe ser capaz de realizar el sistema, sin considerar restricciones físicas; requisito que especifica comportamiento de entrada/salida de un sistema (18). Un requisito no funcional especifica restricciones físicas sobre un requisito funcional. Describe además propiedades del sistema, como restricciones del entorno o de implementación, rendimiento, dependencias de la plataforma, mantenibilidad, extensibilidad o fiabilidad (18).

Requisitos funcionales

RF .1: Registrar datos de una nueva imagen.

RF .1.1: Relacionar imagen con uno o varios temas del producto.

RF .2: Eliminar imagen.

RF .3: Modificar datos de una imagen.

RF .4: Ver datos de una imagen.

RF .5: Registrar datos de un nuevo video.

RF .5.1: Relacionar video con uno o varios temas del producto.

RF .5.2: Asociar una imagen al video.

RF .5.3: Asociar pasos al video.

RF .6: Eliminar video.

RF .7: Modificar datos de un video.

RF .8: Ver datos de un video.

RF .9: Registrar datos de un nuevo sonido.

RF .9.1: Relacionar sonido con uno o varios temas del producto.

RF .9.2: Asociar una imagen al sonido.

RF .10: Eliminar sonido.

RF .11: Modificar datos de un sonido.

RF .12: Ver datos de un sonido.

RF .13: Registrar datos de una nueva animación.

RF .13.1: Relacionar animación con uno o varios temas del producto.

RF .13.2: Asociar imagen a la animación.

RF .14: Eliminar animación.

RF .15: Modificar datos de una animación.

RF .16: Ver datos de una animación.

RF .17: Registrar datos de un nuevo diaporama.

RF .17.1: Relacionar diaporama con uno o varios temas del producto.

RF .17.2: Asociar imagen al diaporama.

RF .18: Eliminar diaporama.

RF .19: Modificar datos de un diaporama.

RF .20: Ver datos de un diaporama.

RF .21: Registrar datos de un nuevo término del glosario.

RF .21.1: Relacionar término con uno o varios temas del producto.

RF .21.2: Insertar imágenes y palabras calientes dentro del texto del significado del término del glosario.

RF .22: Eliminar término del glosario.

RF .23: Modificar datos de un término del glosario.

RF .24: Ver datos de un término del glosario.

RF .25: Registrar datos de una nueva personalidad.

RF .25.1: Relacionar personalidad con uno o varios temas del producto.

RF .25.2: Insertar imágenes y palabras calientes dentro del texto de la información sobre la personalidad.

RF .26: Eliminar personalidad.

RF .27: Modificar datos de una personalidad.

RF .28: Ver datos de una personalidad.

RF .29: Mostrar listado de imágenes.

RF .29.1: Filtrar listado de imágenes por los temas del producto.

RF .30: Mostrar listado de videos.

RF .30.1: Filtrar listado de videos por los temas del producto.

RF .31: Mostrar listado de sonidos.

RF .31.1: Filtrar listado de sonidos por los temas del producto.

RF .32: Mostrar listado de diaporamas.

RF .32.1: Filtrar listado de diaporamas por los temas del producto.

RF .33: Mostrar listado de animaciones.

RF .33.1: Filtrar listado de animaciones por los temas del producto.

RF .34: Mostrar listado de términos del glosario.

RF .34.1: Filtrar listado de términos del glosario por los temas del producto.

RF .35: Mostrar listado de personalidades.

RF .35.1: Filtrar listado de personalidades por los temas del producto.

RF .36: Autenticar usuario.

RF .37: Acceder al frontend de la aplicación desde cualquier pantalla.

RF .38: Registrar datos de un nuevo producto.

RF .39: Eliminar producto.

RF .40: Modificar datos de un producto.

RF .41: Ver datos de un producto desde cualquier pantalla.

RF .42: Mostrar el listado de los productos que componen la colección.

RF .43: Registrar datos de un nuevo usuario.

RF .44: Modificar datos de un usuario.

RF .45: Eliminar usuario.

RF .46: Mostrar el listado de los usuarios de la aplicación.

RF .47: Registrar datos de un nuevo “sabías qué”.

RF .48: Modificar datos de un sabías qué.

RF .49: Eliminar sabías qué.

RF .50: Ver datos de un sabías qué.

RF .51: Mostrar listado de los sabías qué de la colección.

RF .52: Registrar datos de una nueva efeméride.

RF .53: Modificar datos de una efeméride.

RF .54: Eliminar efeméride.

RF .55: Ver datos de una efeméride.

RF .56: Mostrar listado de efemérides.

RF .57: Registrar trazas del usuario en la aplicación.

RF .58: Mostrar listado de las trazas de los usuarios.

RF .59: Filtrar el listado de trazas.

RF .60: Ver datos de una traza.

RF .61: Mostrar árbol de contenidos de la ayuda.

RF .62: Adicionar un nuevo contenedor de ayuda.

RF .63: Modificar un contenedor de ayuda.

RF .64: Eliminar contenedor de ayuda.

RF .65: Ver contenedor de ayuda.

RF .66: Registrar un nuevo contenido de la ayuda.

RF .67: Modificar un contenido de la ayuda.

RF .68: Ver un contenido de la ayuda.

RF .69: Eliminar contenido de la ayuda.

Requisitos no funcionales

RnF 1: Requisitos de usabilidad.

El software será usado por el equipo de trabajo del proyecto Multisaber-Navegante. El mismo está compuesto por estudiantes de la carrera Ingeniería en Ciencias Informáticas y profesionales de la especialidad.

RnF 2: Requisitos de software.

La estación de trabajo donde se instale la aplicación debe contar con sistema operativo Linux o Windows, ya que estos han sido los dos escenarios en los que ha sido instalada y configurada la aplicación. Para publicarla se requiere del servidor web Apache 2.0 y como servidor de base de datos debe utilizarse PostgreSQL 8.4. El navegador web a utilizar debe ser Mozilla Firefox de la versión 10 en adelante, debido al uso de las nuevas etiquetas semánticas de HTML5 y de las reglas de CSS3.

RnF 3: Requisitos de hardware.

A pesar de ser una aplicación web con soporte de datos, el host donde estará alojada la misma debe contar con al menos 15 GB[25] disponibles de HDD[26], debido a que los elementos multimedia que se gestionen serán almacenados en directorios físicos dentro de la estructura de carpetas de la aplicación. Esta estación de trabajo debe contar con al menos 512 MB[27] de RAM[28] y con un microprocesador de 500 MHz[29] que son los requisitos para garantizar el funcionamiento correcto del servidor web.

RnF 4: Restricciones del diseño y la implementación.

---

[25] *Gigabytes.*
[26] *Del inglés: Hard Disk Drive.*
[27] *Megabytes.*
[28] *Del inglés: Random Access Memory.*
[29] *Mega Hertz.*

A la hora de la construcción y la codificación de la aplicación deben tenerse en cuenta los siguientes aspectos:

- Lenguaje PHP en su versión 5.3.
- Lenguaje JavaScript en su versión 1.5.
- Lenguaje HTML en su versión 5.
- Lenguaje CSS en su versión 3.
- IDE NetBeans 7.0.
- Framework Symfony 1.4.3.
- Framework jQuery 1.5.
- ORM Doctrine.

Los aspectos anteriormente mencionados están en concordancia con la necesidad del uso de tecnologías libres que, al mismo tiempo, permitan crear una aplicación web con soporte de datos que cumpla cabalmente con los requisitos especificados.

RnF 5: Requisitos de soporte.

Una vez desplegado el producto se impartirán talleres a los miembros del proyecto Multisaber-Navegante para asesorarlos en el trabajo con la herramienta. Las funcionalidades implementadas podrán ser mejoradas de acuerdo con las necesidades que puedan surgir.

RnF 6: Requisitos de rendimiento.

El tiempo de respuesta ante las peticiones no debe exceder los 3 segundos para garantizar la interactividad del usuario con el sistema.

Modelo de casos de uso del sistema

Los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Por tanto, establece un acuerdo entre clientes y el grupo de desarrollo sobre las condiciones y posibilidades (requisitos) que debe cumplir el sistema (18).

A continuación se muestra un listado de los casos de uso definidos para la implementación del sistema. Los mismos fueron identificados a partir de los requisitos funcionales del software y van

encaminados a solucionar las deficiencias encontradas en las herramientas de gestión de contenidos que anteriormente se utilizaban en el proyecto Multisaber-Navegante.

Casos de uso del sistema

1. Gestionar imagen.
2. Gestionar video.
3. Gestionar sonido.
4. Gestionar animación.
5. Gestionar diaporama.
6. Gestionar glosario.
7. Gestionar personalidad.
8. Listar media.
9. Buscar media.
10. Autenticar usuario.
11. Gestionar producto.
12. Listar producto.
13. Gestionar usuario.
14. Listar usuario.
15. Gestionar “sabías qué”.
16. Listar “sabías qué”.
17. Gestionar efemérides.
18. Listar efemérides.
19. Registrar traza.
20. Listar traza.
21. Buscar traza.
22. Ver traza.
23. Gestionar contenedor de ayuda.
24. Gestionar contenido de ayuda.

El diagrama de casos de uso del sistema puede ser consultado en el **¡Error! No se encuentra el origen de la referencia.**.

Descripción de casos de uso del sistema

El diagrama de casos de uso es insuficiente para comprender las funcionalidades asociadas a cada uno de ellos. Por tal motivo se debe realizar una descripción detallada de los mismos para que clientes y desarrolladores tengan una noción sobre el funcionamiento del software que se desea implementar. En las especificaciones de los casos de uso se relacionan una serie de eventos que tienen lugar durante la interacción entre los actores y el sistema.

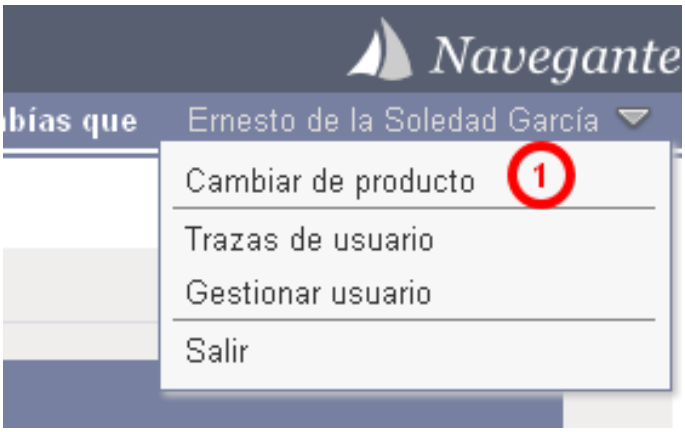
Seguidamente se muestra la especificación del caso de uso *Gestionar producto* perteneciente al módulo *General* del sistema para la gestión de contenidos en la colección El Navegante.

Caso de uso	Gestionar producto
Actores	Administrador (Inicia): Incluye, modifica o elimina producto. Usuario: Ve productos.
Resumen	El caso de uso se inicia cuando el actor Administrador selecciona la opción que le permite realizar una acción sobre un producto. El actor Usuario puede ver un producto y el actor Administrador puede además incluir, modificar y eliminar productos. En caso de que el actor Administrador seleccione la opción de incluir producto, el sistema mostrará la vista para incluir los datos de un producto. Si cualquier actor elige la opción de ver un producto, el sistema mostrará la pantalla principal del producto en cuestión. Si el actor Administrador elige la opción de modificar producto, el sistema mostrará los datos que pueden ser editables dentro de un producto, y una vez realizados los cambios, guardará las modificaciones. Si el actor Administrador elige la opción eliminar producto, el sistema mostrará una ventana de confirmación, en caso de aceptar dicha confirmación, el producto en cuestión será eliminado.
Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado. Para incluir, modificar o eliminar un producto el usuario autenticado debe tener el rol de administrador. Para ver, modificar o eliminar un producto, debe listar los productos (Ver CU Listar Producto) y seleccionar el producto sobre el que desea realizar la acción.
Referencias	RF 38, RF 39, RF 40 y RF 41.

Flujo básico	
Acciones del actor	**Respuesta del sistema**
1. El caso de uso se inicia cuando el actor selecciona la opción cambiar de producto.	
	2. Brinda la posibilidad de realizar las acciones: • Incluir un nuevo producto. • Ver un producto. Ver Sección 1: “Ver datos de producto”. • Modificar los datos de un producto. Ver Sección 2:”Modificar datos de producto”. • Eliminar un producto. Ver Sección 3: “Eliminar producto”.
3. Selecciona la opción de incluir un nuevo producto.	

	4. Brinda la posibilidad de introducir los datos del producto. • Nombre • Referencia • Descripción Y opcionalmente: • Imagen asociada Y permite: • Guardar los datos. • Cancelar la operación en cualquier momento. • Limpiar los datos de cada uno de los campos.
5. Introduce los datos del producto.	
6. Selecciona la opción de guardar los datos.	
	7. Valida los datos.
	8. Crea el producto.
	9. Muestra un mensaje de información "El producto se ha insertado exitosamente."
	10. El caso de uso termina.
Flujos alternos	
4.a El actor selecciona la opción de Cancelar	
Acciones del actor	**Respuesta del sistema**
	4. a.1 Elimina los datos creados.
	4. a.2 Regresa a la vista anterior.
	4. a.3 El caso de uso termina.
4.b El actor selecciona la opción de Limpiar	
Acciones del actor	**Respuesta del sistema**
	4. b.1 Borra los datos de los campos.
	4. b.2 Regresa al paso 5.
7. a Existen datos incompletos.	
Acciones del actor	**Respuesta del sistema**
	7. a.1 Muestra un indicador sobre los campos vacíos.
	7. a.2 Regresa al paso 5.
7. b Existen datos incorrectos.	
Acciones del actor	**Respuesta del sistema**
	7. b.1 Muestra un indicador sobre los campos incorrectos.
	7. b.2 Regresa al paso 5.
Sección 1: "Ver datos de producto"	

Flujo básico	
Acciones del actor	**Respuesta del sistema**
1. Selecciona la opción ver asociada a un producto.	
	2. Muestra la pantalla principal del producto con los datos asociados al mismo. Permite: • Salir de la vista actual.
3. Selecciona la opción de salir de la vista actual.	
	4. Muestra la vista anterior.
	5. El caso de uso termina.
Sección 2: "Modificar datos de producto" **Flujo básico**	
Acciones del actor	**Respuesta del sistema**
1. Selecciona la opción modificar asociada a un producto.	
	2. Muestra los datos editables del producto. Permite además: • Guardar los datos. • Cancelar la operación en cualquier momento. • Limpiar los datos modificados.
3. Modifica los datos que necesite y selecciona la opción de guardar los datos.	
	4. Valida los datos.
	5. Muestra el listado de productos añadidos hasta el momento.
	6. El caso de uso termina.
Flujos alternos	
2.a El actor selecciona la opción de Cancelar	
Acciones del actor	**Respuesta del sistema**
	2. a.1 Regresa a la vista anterior.
	2. a.2 El caso de uso termina.
2.b El actor selecciona la opción de Limpiar	
Acciones del actor	**Respuesta del sistema**
	2. b.1 Elimina los datos modificados y mensajes generados.
	2. b.2 Continúa en el paso 3.
4. a Existen datos incorrectos.	
Acciones del actor	**Respuesta del sistema**
	4. a.1 Muestra un indicador sobre los campos incorrectos.
	4. a.2 Regresa al paso 3.
Sección 3: "Eliminar producto" **Flujo básico**	

Acciones del actor	**Respuesta del sistema**
1. Selecciona la opción de eliminar asociada a un producto.	
	2. Muestra el mensaje de confirmación "¿Está seguro que desea eliminar este producto? Para confirmarlo teclee la palabra ELIMINAR en la caja de texto."
	3. Brinda la posibilidad de introducir el texto indicado y permite: • Eliminar el producto. • Cancelar la operación.
4. Introduce el texto y selecciona la opción eliminar.	
	5. Valida los datos.
	6. Elimina el producto.
	7. Muestra el listado con el resto de los productos registrados en el sistema.
	8. El caso de uso termina.
Flujos alternos	
3.a El actor selecciona la opción de Cancelar	
Acciones del actor	**Respuesta del sistema**
	3. a.1 Regresa a la vista anterior.
	3. a.2 El caso de uso termina.
5. b Existen datos incorrectos.	
Acciones del actor	**Respuesta del sistema**
	5. b.1 Muestra un mensaje de información "Debe escribir correctamente la palabra ELIMINAR".
	5. a.2 Regresa al paso 4.
Prototipos elementales de interfaz de usuario **IU 1. Interfaz para acceder a gestionar producto.**	
	
Eventos	

Control	Valor	Evento	Acción
Ítem del menú Cambiar de Producto.	1	Clic	Muestra IU 2 Pantalla gestionar producto.

IU 2. Pantalla gestionar producto.

Eventos

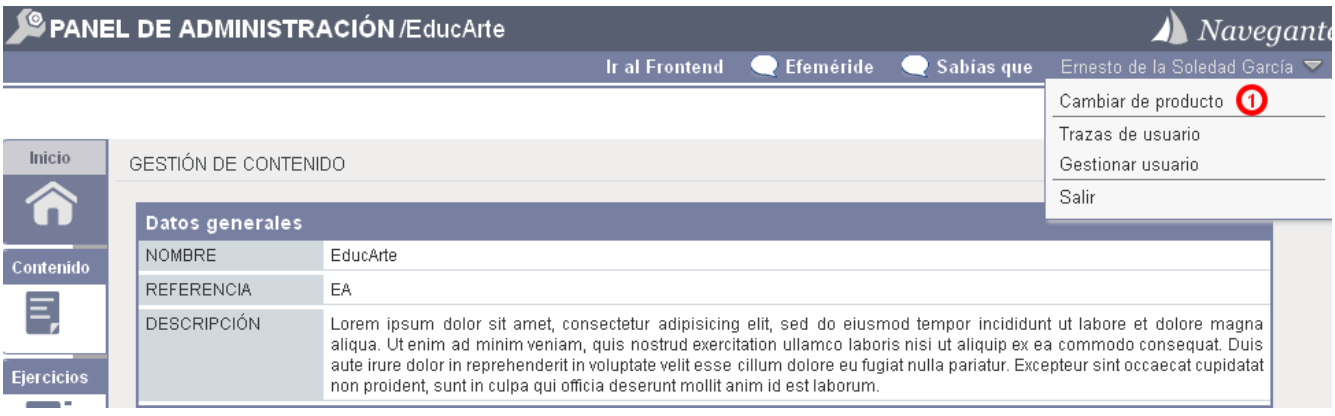
Control	Valor	Evento	Acción
Botón “Cree uno nuevo”	1	Clic	Muestra IU 3 Pantalla crear nuevo producto.
Hipervínculo “Entrar”	2	Clic	Muestra IU 4 Pantalla ver datos de producto.
Hipervínculo “Editar”	3	Clic	Muestra IU 5 Pantalla modificar datos de producto.
Hipervínculo “Eliminar”	4	Clic	Muestra IU 6 Pantalla eliminar producto.

IU 3. Pantalla crear nuevo producto.

Eventos

Control	Valor	Evento	Acción
Botón Insertar	1	Clic	Adiciona el producto. Limpia los campos y permite adicionar otro.
Botón Limpiar	2	Clic	Limpia los campos del formulario.
Botón Cancelar	3	Clic	Limpia los campos del formulario y cierra la ventana, mostrando la vista anterior actualizada.

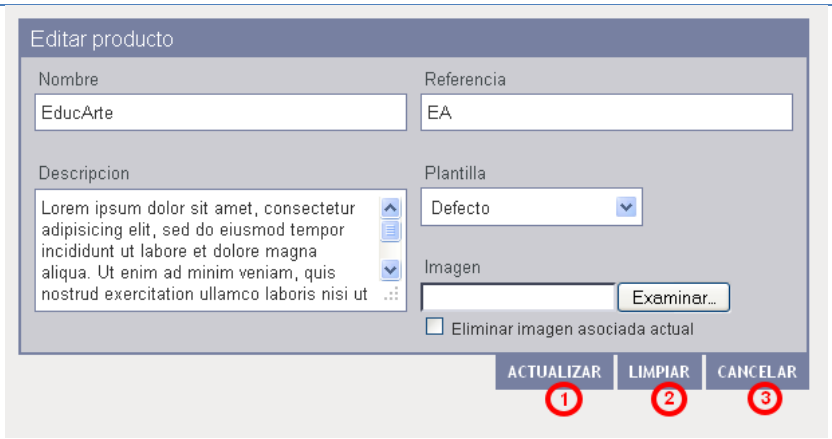
IU 4. Pantalla ver datos de producto.



Eventos

Control	Valor	Evento	Acción
Ítem del menú Cambiar de Producto.	1	Clic	Muestra IU 2 Pantalla gestionar producto.

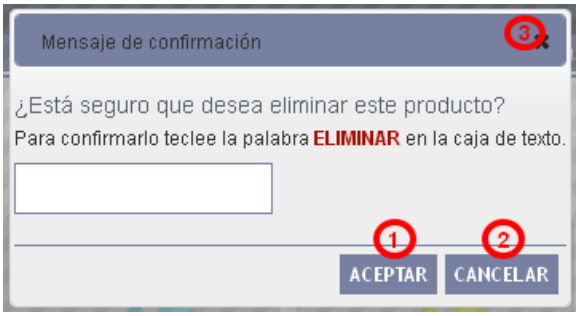
IU 5. Pantalla modificar datos de producto.



Eventos

Control	Valor	Evento	Acción
Botón Actualizar	1	Clic	Actualiza los datos del producto. Permite seguir modificando los datos.
Botón Limpiar	2	Clic	Actualiza los campos del formulario con los valores iniciales.
Botón Cancelar	3	Clic	Limpia los campos del formulario y cierra la ventana, mostrando la vista anterior actualizada.

IU 6. Pantalla eliminar producto.



Eventos

Control	Valor	Evento	Acción
Botón Aceptar	1	Clic	Elimina el producto. Cierra la ventana y muestra la vista anterior actualizada.
Botón Cancelar	2	Clic	Cierra la ventana y muestra la vista anterior.
Botón Cerrar	3	Clic	Cierra la ventana y muestra la vista anterior.

Tabla 3 Especificación del caso de uso gestionar producto.

Conclusiones del capítulo

La realización del modelo de dominio y la descripción de los conceptos asociados al mismo fueron actividades que contribuyeron a una correcta captura de requisitos, lográndose al mismo tiempo un entendimiento pleno entre desarrolladores y usuarios del producto. Como parte de la descripción del sistema también se identificaron y modelaron los casos de uso que guiarán el proceso de desarrollo del software; los cuales se describieron detalladamente de acuerdo a lo que plantea la metodología en uso.

DISEÑO DEL SISTEMA

Introducción

Según la IEEE[30] 610.12-90 el diseño del software es el proceso de definición de la arquitectura, componentes, interfaces y otras características de un sistema o componente que resulta de este proceso. Estas actividades son las que se describirán a lo largo de este capítulo, sirviendo al mismo tiempo para justificar la utilización de determinados patrones de diseño y algunos elementos importantes dentro del modelo de datos del sistema.

Arquitectura

Para el desarrollo del sistema se utiliza el patrón arquitectónico Modelo-Vista-Controlador (MVC) en el cual está basado el framework Symfony. Este patrón separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Además permite mantener el controlador y el modelo original para aplicaciones que se accedan desde navegadores estándar o móviles cambiando solamente la vista para cada dispositivo (29) (33). La separación de la vista y el modelo trae ventajas; es posible tener diferentes representaciones de la misma información, haciendo uso del mismo código dentro del modelo. Es posible además programar el código del modelo, abstrayéndose de la representación visual que se le dará a la información.

El patrón MVC define tres roles o niveles:

- **El modelo:** representa la información relacionada con el dominio de la aplicación, es decir, su lógica del negocio. Abstrae la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes del tipo de gestor de base de datos utilizado. El ORM Doctrine permite utilizar el lenguaje DQL en las clases del modelo generadas para manipular los contenidos de la base de datos. Dichas clases son generadas a partir del esquema de la base de datos escrito en XML o YAML.

[30] *Del inglés: Institute of Electrical and Electronics Engineers.*

- **La vista:** transforma el modelo en interfaces de usuario permitiendo la interacción con el sistema. Solo se encarga de mostrar información. Esta capa está separada en tres partes: los elementos comunes a todas las páginas de la aplicación, denominada layout; las plantillas, encargadas de visualizar las variables definidas en el controlador; y la lógica de las vistas, definida a través de los slots y componentes (fragmentos de vistas que contienen lógica de aplicación y pueden ser reutilizables).
- **El controlador:** se encarga de procesar las peticiones realizadas desde el navegador realizando los cambios necesarios en el modelo o en la vista. Este se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comando, etc.).

Symfony divide esta capa en un controlador frontal y acciones. El controlador frontal es el único punto de entrada a la aplicación, carga la configuración y determina las acciones a ejecutarse. Este se encuentra ubicado en el directorio web del proyecto. Las acciones verifican la integridad de las peticiones, contienen la lógica de la aplicación y preparan los datos necesarios para actualizar la vista.

La capa del controlador contiene los objetos encargados de acceder a los parámetros de la petición, a las cabeceras de las respuestas y a los datos de sesión del usuario. Además brinda la posibilidad de ejecutar filtros antes o después de cada acción; según las comprobaciones sistemáticas realizadas en el filtro, puede o no ser modificado el procesamiento de la petición (29).

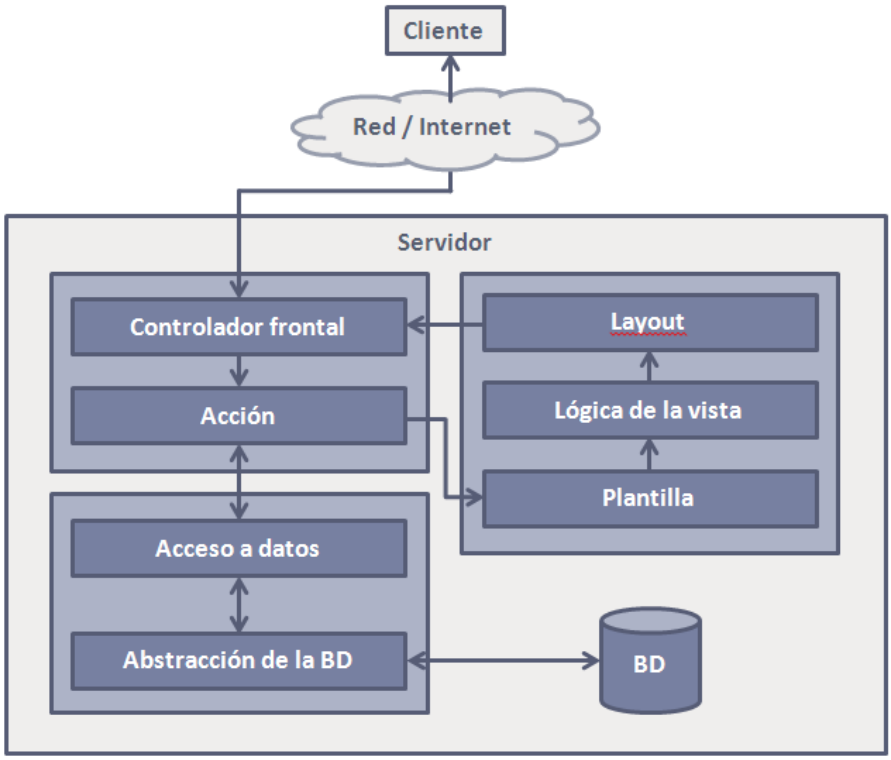


Figura 3 Patrón modelo-vista-controlados en Symfony.

Patrones de diseño

Un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos (41). El framework Symfony utiliza en su implementación un conjunto de patrones de diseño para dar solución a problemas específicos del diseño orientado a objetos durante el flujo de ejecución de una petición. Para el desarrollo del sistema no fue necesario utilizar patrones de diseño diferentes a los que utiliza el framework, por tal motivo se explicarán los utilizados directamente en la implementación de la aplicación.

Patrones GRASP

- **Experto:** este patrón es el principio básico de asignación de responsabilidades. Pretende asignar una responsabilidad al experto en información, o sea, la clase que cuenta con información necesaria para cumplir la responsabilidad, ya sea crear un objeto o implementar un

método (42). La utilización de este patrón dentro del sistema permite un diseño con mayor cohesión y se mantiene el encapsulamiento, ayudando a entender y mantener el código fácilmente. El patrón Experto fue utilizado en la capa de abstracción del modelo. Las clases generadas poseen un grupo de funcionalidades que facilitan el acceso y la manipulación de los datos de las entidades persistentes en la base de datos.

- **Creador:** en el desarrollo de un sistema orientado a objetos conviene contar con un principio general para asignar la responsabilidad de creación. En este sentido, el patrón Creador ayuda a identificar quién debe ser el responsable de crear o instanciar un objeto o clase en dependencia de la relación y visibilidad entre la clase creada y la clase creador (42). La ventaja de utilizar este patrón es eliminar la dependencia entre clases, facilitando el mantenimiento y la reutilización. Dentro del sistema este patrón se evidencia en las acciones del controlador, las cuales crean objetos del modelo o los formularios que representan las entidades.
- **Alta cohesión:** la información que almacena una clase debe ser coherente y debe estar relacionada con la clase, asignando responsabilidades con una alta cohesión (42). Dentro del sistema, este patrón fue utilizado en las acciones, las cuales contienen varias funcionalidades que están relacionadas. Un ejemplo de ello lo constituyen los diferentes “actions” que definen las acciones para las plantillas y colabora con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades de los mismos.
- **Bajo acoplamiento:** asigna una responsabilidad a una clase para mantener bajo acoplamiento, o sea, disminuir la dependencia entre clases, evitando que una modificación en alguna de ellas repercuta en gran medida en el resto, posibilitando además una mayor reutilización (42). Este patrón es utilizado por el framework Symfony, y por ende en el sistema, en las clases pertenecientes a las acciones que heredan únicamente de “sfAction” para alcanzar un bajo acoplamiento entre clases. Además, al no asociar las clases del modelo con las de la vista o el controlador, la dependencia entre las clases, en este caso, se mantiene baja.
- **Controlador:** asigna la responsabilidad del manejo de mensajes de los eventos del sistema a una o varias clases. Sirve como intermediario entre una determinada interfaz y la acción que debe ejecutarse, es decir, recibe los datos del usuario y los envía a las distintas clases según el método llamado (42). Este patrón aumenta la reutilización de código. Dentro del sistema el patrón Controlador es utilizado en las clases de la capa del controlador del patrón MVC, como son: “sfController”, “sfWebController”, “sfContex”, “sfAction” y las clases que heredan de la

misma (los “actions”), así como el index.php del ambiente (en el caso del sistema integrado al proyecto, “backend.php”).

Patrones GoF

- **Singleton:** patrón de tipo creación, a nivel de objetos. Garantiza que una clase solo tiene una única instancia, proporcionando un punto de acceso global a la misma (43). El objeto “sfContext” de Symfony sirve de mecanismo para interactuar con los objetos únicos del núcleo del framework, ya que almacena una referencia a todos estos objetos. Además la clase “sfRouting” que se encarga de enrutar todas las peticiones que de hagan a la aplicación implementa este patrón mediante el método “getInstance”.
- **Decorator:** patrón de tipo estructura, a nivel de objetos. Añade responsabilidades adicionales a un objeto dinámicamente (44). Symfony utiliza el patrón Decorator para la vista, el layout o plantilla global decora el contenido de la plantilla.

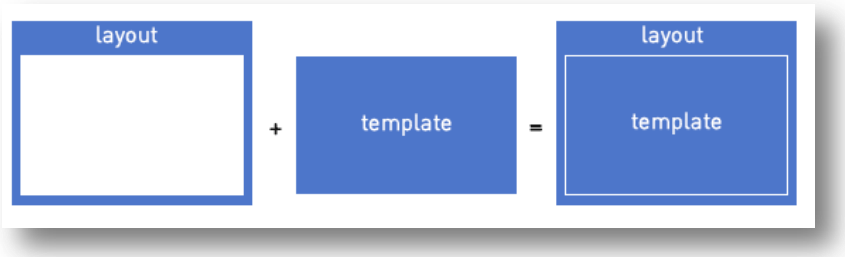


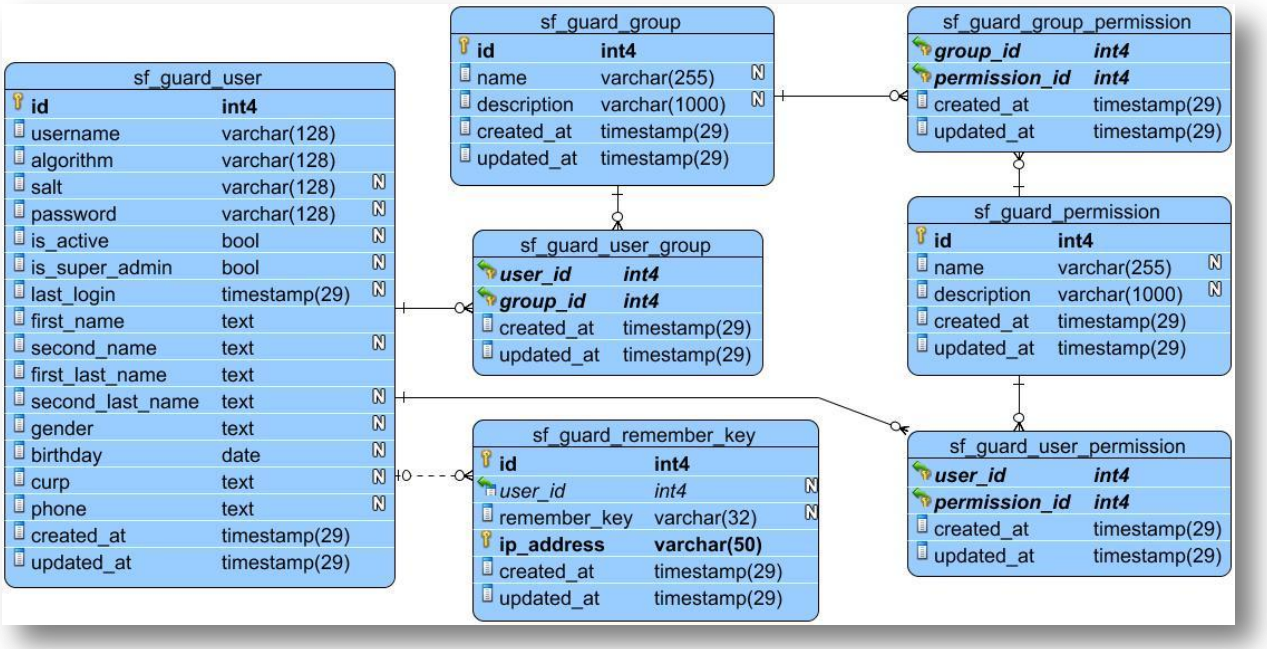
Figura 4 Patrón Decorator en Symfony.

Computación distribuida

- **Registry:** permite compartir datos y objetos en la aplicación sin la necesidad de preocuparse por conservar numerosos parámetros o hacer uso de variables globales (43). Dentro de la aplicación, el uso de este patrón se evidencia en la clase encargada de la configuración, ya que contiene todas las variables globales del sistema.

Seguridad en el sistema

Uno de los aspectos fundamentales en el desarrollo de aplicaciones web, es la confidencialidad, integridad y disponibilidad de la información. En este sentido, las aplicaciones web en ocasiones se



ven afectadas por las vulnerabilidades de la propia implementación del sistema. Por tal motivo, se desarrolló un conjunto de mecanismos, apoyados con los implementados por el framework Symfony, para garantizar una aplicación segura.

Dándole solución a un requisito funcional del sistema y apoyando la seguridad de los datos, uno de los mecanismos es el de autenticación al sistema, mediante usuario personal y contraseña. Cada usuario deberá autenticarse previamente para poder realizar cualquier acción sobre algún módulo del sistema. Para su implementación se utilizó el plugin de Symfony "sfGuardPlugin" donde los roles y credenciales de los usuarios se manejan mediante un conjunto de grupos de usuarios y permisos asociados o bien al usuario o bien al grupo de usuarios.

Figura 5 Modelo de datos del sfGuardPlugin.

Además este mecanismo se apoya en ficheros de configuración YAML que garantizan que para acceder a los módulos del sistema, el usuario, además de estar autenticado, debe tener las credenciales necesarias para los mismos. Garantizando de esta forma, la confidencialidad de los datos, donde los usuarios solo podrán accionar sobre la información que por el rol establecido en el sistema puede controlar. La aplicación también registra un conjunto de trazas sobre las acciones realizadas por cada usuario, lo que permite conocer en cada momento qué usuario manipuló un conjunto de datos determinados.

Las aplicaciones web pueden verse afectadas también por ataques CSRF (Cross-site Request Forgery), XSS (Cross Site Scripting) y SQL Injection. Ante estos ataques el framework Symfony implementa un conjunto de mecanismos muy eficaces, los cuales fueron utilizados en la implementación del sistema (29).

Los ataques XSS, aprovechan vulnerabilidades en el sistema de validación de los componentes HTML incrustados en una página web, provocando muchas veces la denegación de servicios y pérdida de información. Los ataques CSRF, aprovechan la confianza que tienen los sitios web en los usuarios, para ejecutar códigos maliciosos. Los ataques SQL Injection, explotan vulnerabilidades en los códigos de acceso a datos con el objetivo de obtener información almacenada o destruir bases de datos completas.

Para evitar los ataques XSS, Symfony establece estrategias de escape para las variables disponibles en las vistas evitando la ejecución de código JavaScript malicioso. En el caso de los ataques CSRF se establece la directiva “csrf_secret”, que puede ser configurada en el fichero “settings.yml”, a la misma se le asigna una palabra secreta; internamente el framework genera un hash a partir del token establecido, el cual puede ser comprobado en cada petición, incluyéndolo como valor en un widget de formulario o en los helpers para vínculos. En otro sentido, se estableció que todo el acceso a datos de la aplicación se realizaría mediante el uso de Doctrine, así como las consultas serían generadas por los métodos provistos por el ORM y las clases de acceso a datos del modelo, evitando ataques SQL Injection (29) (33).

Modelo del diseño

Antes de pasar a las particularidades del diseño en la presente investigación, se hace necesario justificar por qué no se realiza el análisis. Como un elemento determinante se tiene el hecho de que los clientes del producto son personas (estudiantes o profesionales) con conocimientos de informática, por lo que el proceso de captura de requisitos fue claro y explícito. Dentro de la metodología en uso, el análisis es un refinamiento de dichos requisitos para poder estructurar el software; de ahí que el equipo de desarrollo decidió ofrecer la visión general del sistema a través del estudio de los resultados del diseño y la implementación (18).

El modelo del diseño es un modelo de objetos que describe la realización física de los casos de uso y se centra en los requisitos funcionales y no funcionales. Las abstracciones del modelo del diseño tienen una correspondencia directa con los elementos físicos del ambiente de implementación. Los

objetivos del diseño de un caso de uso son: identificar las clases (subsistemas o interfaces) necesarias para llevar a cabo el caso de uso y definir los requisitos de las operaciones sobre dichos artefactos, permitiendo así reflejar con claridad las restricciones que deben ser cumplidas a la hora de la implementación (18). En la Figura 6 Diagrama de clases del diseño para el caso de uso Gestionar producto. se presenta el diagrama de clases del diseño para el caso de uso *Gestionar producto*; el mismo está estructurado de acuerdo con el patrón arquitectónico MVC y muestra la interacción con el ORM Doctrine, el framework de presentación jQuery y el resto de los elementos que provee el uso de Symfony.

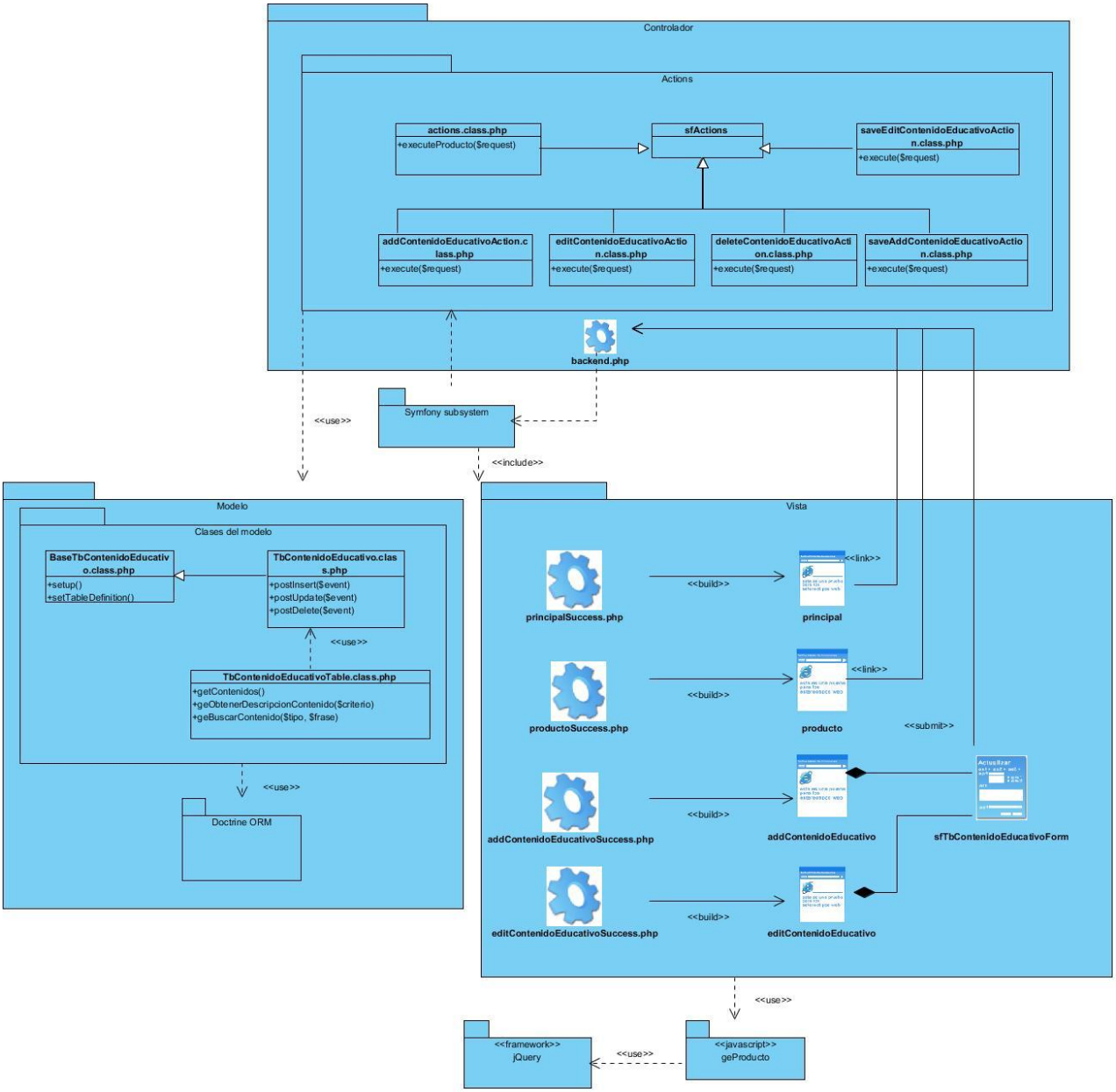
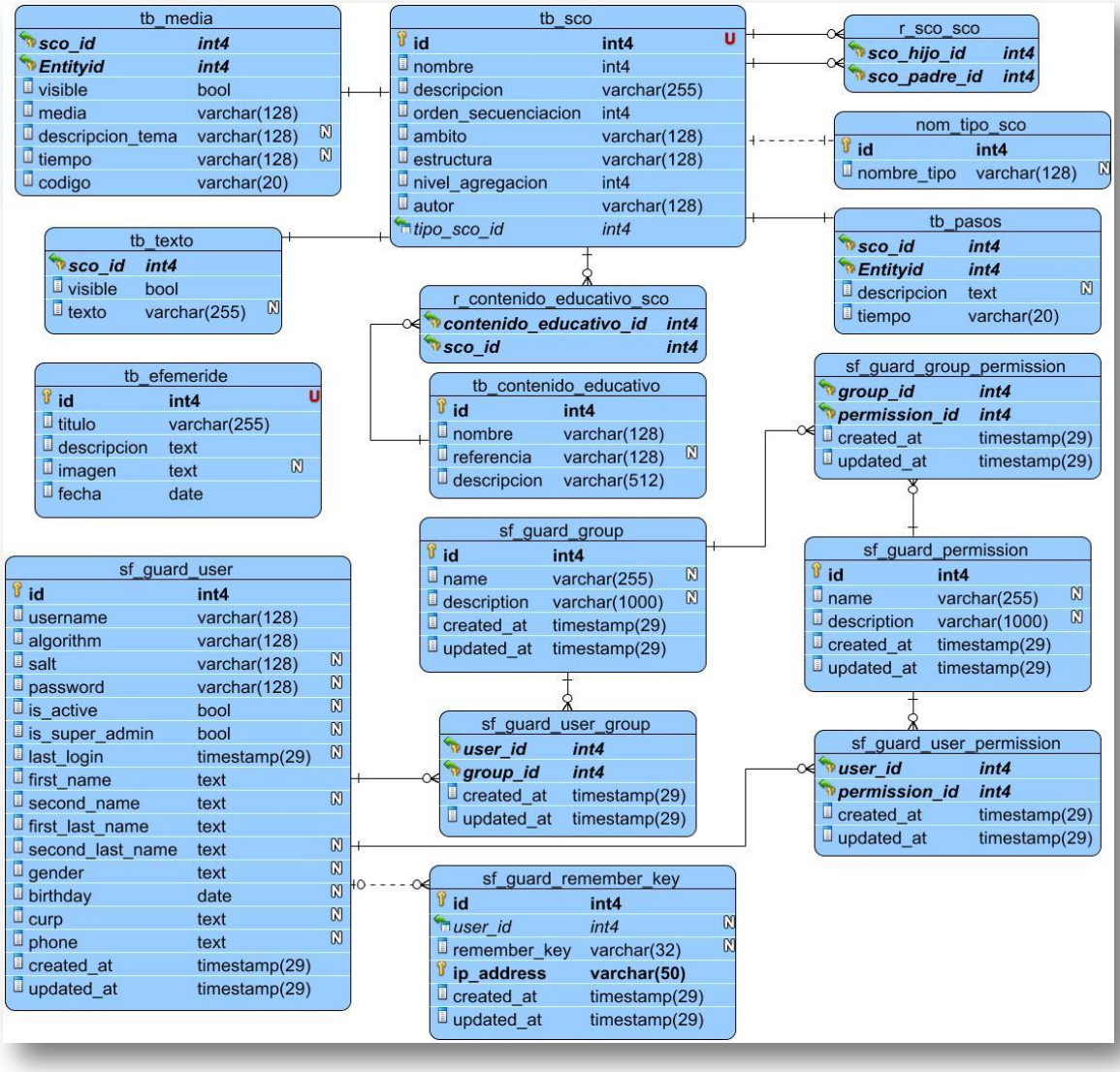


Figura 6 Diagrama de clases del diseño para el caso de uso Gestionar producto.

Diseño de base de datos

El modelo de datos contiene una representación física de los datos del sistema que deben persistir en la base de datos (34). Debido a que se implementa el panel de administración para la colección de



software educativo El Navegante ambas aplicaciones comparten el mismo modelo de datos; el cual fue concebido bajo las especificaciones del estándar que define los metadatos para objetos educativos.

Figura 7 Modelo de datos.

El estándar LOM[31] especifica un esquema conceptual de datos que define la estructura de una instancia de metadatos para un objeto educativo, o sea, describe las características relevantes del objeto educativo al que se aplica. Para este estándar, un objeto educativo se define como cualquier

[31] *Del inglés: Learning Object Metadata.*

entidad, digital o no, susceptible de ser usada en aprendizaje, educación o formación. El estándar asegura que las implementaciones de los metadatos de objetos educativos tendrán un alto grado de interoperabilidad semántica (45).

El diagrama de la Figura 7 Modelo de datos. modela solamente las entidades relacionadas con los módulos Contenidos, Mediateca y los servicios generales del panel de administración. En la tabla *tb_sco* se generalizan las diferentes clasificaciones de objetos educativos que se manejan en el sistema, como son los elementos de la Mediateca (imágenes, videos, audios, diaporamas, animaciones, palabras del glosario y personalidades) y los artículos del módulo Contenidos. Para el modelo de datos de la colección El Navegante se redefinió la estructura que propone el esquema base del estándar LOM; por lo que las características que se describen para una instancia de metadatos de un objeto educativo son: nombre, descripción, orden de secuenciación, nivel de agregación, ámbito, estructura y autor.

Modelo de despliegue

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo (18). La Figura 8 Modelo de despliegue. representa el hardware y los protocolos de comunicación que se requieren para el despliegue del sistema en cuestión.

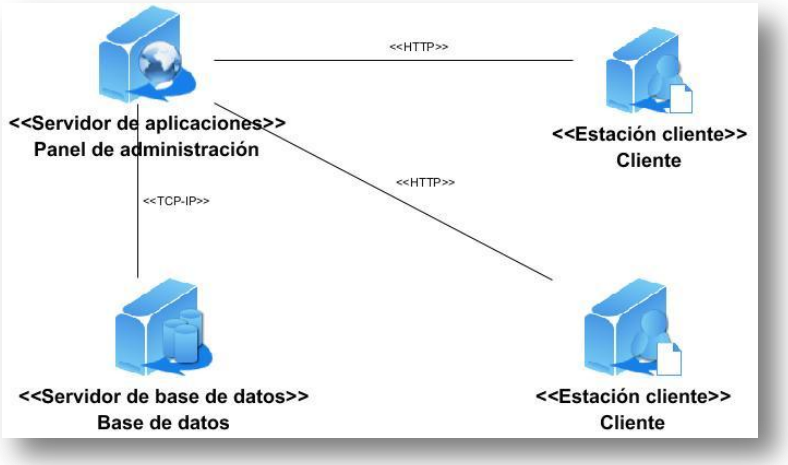


Figura 8 Modelo de despliegue.

Conclusiones del capítulo

A lo largo del capítulo fueron descritos todos los aspectos que tuvieron un impacto significativo en el diseño del software en cuestión. Es importante destacar que el framework de desarrollo Symfony propició el uso del patrón arquitectónico MVC y de una serie de patrones de diseño que, en su conjunto, garantizaron la reutilización de las funcionalidades y ponen de manifiesto el uso de buenas prácticas en la codificación del sistema. Además fueron representados en sus respectivos diagramas el modelo de diseño para uno de los casos de uso de la aplicación, el modelo de despliegue y el de base de datos, artefactos que son la entrada al flujo de trabajo implementación.

IMPLEMENTACIÓN Y PRUEBAS

Introducción

Los artefactos generados en el diseño son la principal entrada en la flujo de trabajo implementación. En esta etapa se describe el software en términos de componentes (ficheros de código fuente y binario, scripts, ejecutables y similares). En el presente capítulo se describen cada uno de dichos componentes y la estrategia de prueba utilizada para verificar el resultado de la implementación.

Modelo de implementación

La organización de los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en los lenguajes de programación utilizados es descrita en el modelo de implementación. Este modelo además describe la forma en que los elementos obtenidos del modelo de diseño (como las clases) se implementan en términos de componentes y las dependencias que presentan dichos componentes entre sí. El modelo de implementación es la entrada principal para las pruebas de software que siguen a la etapa de implementación (18).

Diagrama de componentes

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño (18). El diagrama de componentes se utiliza para modelar la vista estática de un sistema. Se encarga de mostrar los componentes de software y las relaciones lógicas entre ellos en un sistema, además de estructurar el modelo de implementación en términos de subsistemas de implementación.

A continuación se muestra el diagrama de componentes para la implementación del software:

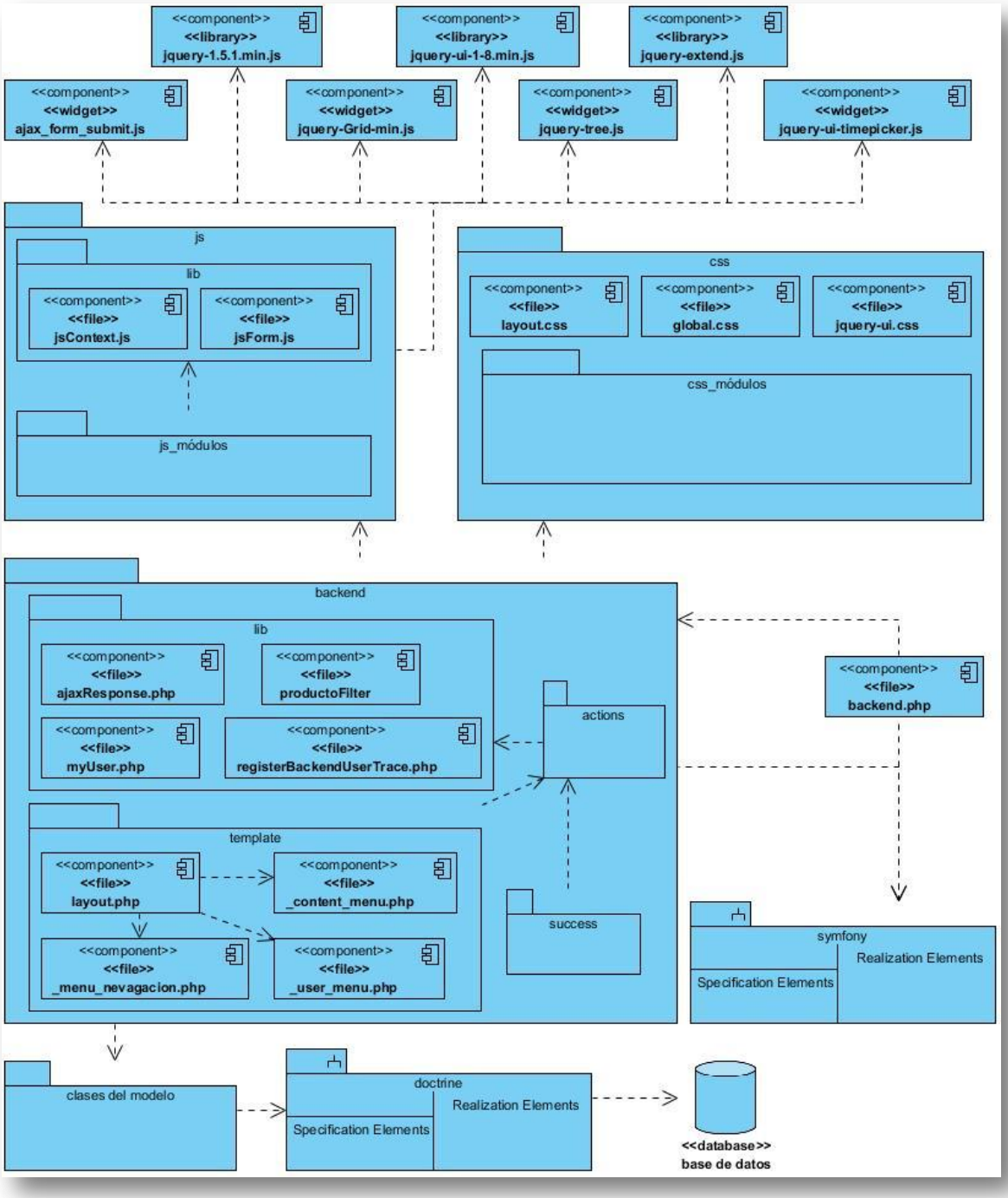


Figura 9 Diagrama de componentes.

Estándares de codificación

El proceso de codificación incluye el conjunto de reglas o normas usadas para escribir código, lo cual aporta eficiencia al proceso de desarrollo, logrando que los programas sean más comprensibles y de mayor calidad. Los aspectos fundamentales que se deben tener en cuenta a la hora de confeccionar un estilo de código son: notación, comentarios, indentación, espacios y líneas en blanco, longitud máxima de las líneas de caracteres, declaración de variables y constantes y parámetros de los métodos.

Los estándares definidos para el desarrollo de la aplicación han sido tomados, en su gran mayoría, de los definidos por la arquitectura de software del proyecto al cual pertenece el sistema (2). A continuación de enumeran algunos de ellos:

- Se utiliza el tabulador para la alineación, excepto en los archivos YAML donde se utilizan dos espacios.
- Cuando una expresión no cabe en una línea simple debido a su extensión se divide en más de una línea, después de una coma o un operador.
- Los comentarios se definen comenzando con los caracteres “/*” y terminando con “*/”.
- Se utiliza el estándar CamelCase para el nombre de los métodos y UpperCamelCase para el nombre de las clases.
- Las variables deben ser explícitas y sin abreviaturas.

Interfaces de usuario

La interfaz, además de permitir interactuar con la aplicación, brinda la primera impresión del usuario hacia el sistema. Por tal motivo se debe lograr una apariencia adecuada para que el usuario se sienta a gusto dentro del sistema. El diseño de una interfaz tiene en cuenta varios aspectos como son los relacionados con la tipografía, los colores, los gráficos, la navegación y la composición del sitio.

A continuación se brindan algunas de las características significativas de las interfaces de usuario; en el **¡Error! No se encuentra el origen de la referencia.** pueden ser consultadas algunas imágenes del software:

- La gama de colores utilizada en el diseño denota seriedad, además de ser diferente a las utilizadas en los productos de la colección, evitando parcializar al panel de administración con algún producto en específico.

- La arquitectura de la información es muy similar a la de la colección El Navegante. Lo anterior garantiza que si un usuario ha trabajado con alguna de las dos aplicaciones, está perfectamente capacitado para trabajar con la otra, siendo esta una vía de lograr uniformidad en el diseño.
- Se utilizan imágenes identificativas como vínculos para la navegación.
- La letra utilizada en todo el sistema es FreeSans.
- Los mensajes de error en los formularios son claros y dirigidos específicamente a los campos que lo requieren. Cualquier otro tipo de mensaje es mostrado en una barra inferior común para todas las vistas de la aplicación.

Estrategia de pruebas

Una vez descrito el software en términos de sus componentes se hizo necesario verificar el funcionamiento del mismo para garantizar calidad en el producto. Los objetivos de las pruebas son (18):

- Planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de integración y las pruebas de sistema.
- Diseñar e implementar pruebas creando los casos de prueba (especifican qué probar), procedimientos de prueba (especifican cómo realizar las pruebas), creando componentes de prueba para automatizar estos procesos.
- Realizar las pruebas y manejar los resultados de cada una de ellas sistemáticamente.

Dentro de este flujo de trabajo en el ciclo de desarrollo del panel de administración de la colección El Navegante se definieron niveles, tipos de pruebas, métodos y técnicas, para verificar que todos los requisitos fuesen implementados. Lo anterior no garantiza la ausencia de defectos en la aplicación pues las pruebas solo se limitan a demostrar la existencia de dichos defectos.

Niveles, tipos, métodos y técnicas de pruebas

Entre los niveles generales de las pruebas (18) que propone la metodología en uso, el seleccionado fue el de pruebas internas, y dentro de él, los de pruebas de integración y sistema. Los casos de prueba resultantes de esta estrategia están dirigidos a evidenciar la integración de cada componente y el funcionamiento del sistema como un todo. La siguiente tabla relaciona con más detalles la estrategia diseñada.

Niveles de pruebas	Tipos de pruebas	Métodos	Técnicas
Integración	Funcionalidad (función)	Caja negra	Partición de equivalencia
Sistema	Funcionalidad (seguridad)	Caja negra	Partición de equivalencia

Tabla 4 Niveles, tipos, métodos, y técnicas para la estrategia de pruebas

A la aplicación le fueron realizadas pruebas de funcionalidad (función y seguridad) que permitieron validar la implementación de funciones, métodos y casos de uso, así como verificar la integridad y disponibilidad de la información y el acceso a ella por parte de los actores deseados (18).

Las pruebas de caja negra son las que se llevan a cabo sobre la interfaz del software. A partir de los casos de prueba generados bajo este método se demostró que las funcionalidades del software son operativas, que las entradas se aceptan de la forma adecuada y que se produce el resultado correcto; fueron corregidos además algunos errores en la interfaz y en las funciones visibles al usuario. La técnica empleada fue la partición de equivalencia puesto que divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software. En el **¡Error! No se encuentra el origen de la referencia.** puede ser consultado el caso de prueba para el caso de uso “Gestionar producto”.

Entorno de las pruebas

De acuerdo con lo planteado en los requisitos no funcionales de hardware en el entorno de pruebas las computadoras utilizados tenían 1 GB de RAM, procesador Corel 2 Duo a 2.2 GHz y los sistemas operativos Debian 6.0, Ubuntu de la versión 10.4 a la 11.10 y Windows 7 SP 1, todos ellos con el navegador Mozilla Firefox con versión superior a la 3.5. Para la conexión entre los diferentes nodos el ancho de banda disponible fue de 100 Mbps.

Análisis de los resultados

Tras aplicar la estrategia de pruebas propuesta, se obtuvieron resultados satisfactorios que validan en cierta medida la calidad de la aplicación desarrollada. La Tabla 5 Resultados de las pruebas. muestra las no conformidades resultantes de tres iteraciones de pruebas:

	Iteración 1	Iteración 2	Iteración 3
Cantidad	20	11	4

Tabla 5 Resultados de las pruebas.

Las no conformidades encontradas en la primera iteración estuvieron estrechamente relacionadas con el sistema de almacenamiento de medias en los directorios físicos, conflictos con los usuarios de la colección y dificultades en los rangos de fechas de las efemérides. Debido al impacto que tenían en la calidad de los contenidos a gestionar, el equipo de trabajo se enfocó en solucionar dichas no conformidades, logrando en cada iteración de prueba disminuir el número de defectos en el software.

El resto de las no conformidades estuvieron vinculadas a los elementos del diseño de las interfaces y la arquitectura de la información en el software. Ante esta dificultad se decidió replantear estos elementos de acuerdo con la distribución del contenido en los productos de la colección, como se explica en el epígrafe Interfaces de usuario.

Conclusiones del capítulo

Uno de los resultados de este capítulo es modelo de implementación, donde quedó claramente reflejada la realización física de los elementos del diseño. Por otro lado se definió una estrategia de prueba para verificar el funcionamiento del software. El desarrollo y evaluación de cada caso, componente y procedimiento de prueba estuvo enfocado en los casos de uso que estaban asociados a los requisitos a de mayor impacto para el usuario. Las pruebas aplicadas al sistema dieron lugar a realizar mejoras sustanciales en la implementación de las capas de la lógica del negocio y presentación, logrando una apariencia más profesional, a fin con la colección El Navegante y con las herramientas de gestión de contenidos estudiadas.

CONCLUSIONES

Las mejoras realizadas al panel de administración de la colección El Navegante junto a las nuevas funcionalidades implementadas facilitan los procesos de gestión de contenidos. La aplicación permite crear dinámicamente los productos que integran la colección, dentro de cada uno de ellos en la mediateca podrán ser gestionados los objetos educativos de tipo multimedia (imagen, video, sonido, diaporamas y animaciones) además de los términos del glosario y la información sobre personalidades relacionadas con las temáticas que aborde el producto. Todos estos elementos pueden ser utilizados en la creación de artículos, ejercicios y juegos para enriquecer el contenido de cada uno de ellos.

Los cambios realizados al diseño de las interfaces de usuario se basaron en las no conformidades encontradas durante el período de pruebas. Apoyado en personal capacitado, el equipo de trabajo logró una buena arquitectura de información en el sistema y modificó la tipografía, los colores, los gráficos, la navegación y la composición del sitio.

La aplicación web con soporte de datos resultante de esta investigación centraliza las tareas de gestión de contenidos de El Navegante para los módulos Contenidos, Mediateca, Juegos y Ejercicios, además de la información de los servicios generales de la colección: ayuda contextual, efemérides y “sabías qué”, permitiendo agilizar el proceso de desarrollo logrando mayor eficiencia en el montaje y corrección de los contenidos.

RECOMENDACIONES

Flexibilizar el proceso de gestión de los productos, permitiendo al usuario seleccionar los módulos que desee incorporar al mismo, ya que en estos momentos todos tienen la misma estructura

Se recomienda además implementar una funcionalidad que permita exportar un producto, para evitar que el usuario dependa de la colección completa para utilizar uno de ellos.

Utilizar un servidor dedicado al almacenamiento de los elementos de la mediateca ya que actualmente se guardan en una dirección física en el servidor de aplicaciones.

Se recomienda al proyecto incorporar el panel de administración como valor agregado a la colección El Navegante, ya que deja a la entidad donde se despliegue la colección en completa libertad de incorporar el contenido que pueda ser más provechoso para sus estudiantes.

BIBLIOGRAFÍA

1. **Del Toro Rodríguez, Mario y Parra Vigo, Isel Bibiana.** *Modelo de diseño didáctico de hiperentornos de enseñanza-aprendizaje desde una concepción desarrolladora.* La Habana : s.n., 2006.

2. **Jorge Díaz, Anays y Cardona Marrero, Ediel.** *Arquitectura de la versión multiplataforma de la colección de software educativo El Navegante.* La Habana : s.n., 2010.

3. **Hernández León, Rolando Alfredo y Coello González, Saida.** *El proceso de la Investigación Científica.* La Habana : Editorial Universitaria del Ministerio de Educación Superior, 2011.

4. *El diseño metodológico de la investigación científica. Teoría de muestreo: población y muestra. Diseño experimental y métodos.* **Pérez Martinto, Pedro Carlos.** Universidad de las Ciencias Informáticas : s.n., 2010. Entorno Virtual de Aprendizaje.

5. **Ramos Pérez, Lourdes, y otros, y otros.** ¿Software educativo, hipermedia o entorno educativo? [En línea] 28 de Septiembre de 2008. Dispobible en: http://bvs.sld.cu/revistas/aci/vol18_4_08/aci61008.htm.

6. *Software educativos.* **Gómez Martínez, Freddy, Vidal Ledo, María y Ruiz Piedra, Alina.** 1, La Habana : Editorial de Ciencias Médicas, 2009, Vol. 24. ISSN 0864-2141.

7. IHMC-Institute for Human Machine Cogniition - CampServer. *El Software Educativo.* [En línea] 2011. Disponible en: http://cursa.ihmc.us/rid=1196862742453_516504673_8298/SOFTWARE_EDUCATIVO.pdf.

8. **Lamarca Lapuente, María Jesús.** Tesis Doctoral Hipertexto: El nuevo concepto de documento en la cultura de la imagen. [En línea] Universidad Complutense de Madrid, 19 de Noviembre de 2011. Disponible en: http://www.hipertexto.info.

9. *Gestión de Contenidos y TIC, La Web como alternativa de visibilidad de las organizaciones.* **Hidalgo, Darío , y otros, y otros.** 31, Buenos Aires : Universidad del Nordeste de Argentina, 2011, Vol. 1. ISSN 1669-6581.

10. *Virtualmend: un sistema gestor de contenidos de aprendizaje para la universalización de la educación superior.* **Domínguez Lobaina, Junior y Frezno Chávez, Caridad.** 3, La Habana : s.n., 2007, Vol. 16. ISSN.

11. **Martín Galán, Bonifacio, y otros, y otros.** *Gestión de Contenidos Web mediante herramientas de software libre.* Madrid : Universidad Carlos III.

12. *La integración de plataformas de e-learning en la docencia universitaria:enseñanza, aprendizaje e investigación con moodle en la formación inicial.* **Correa Gorospe, José Miguel.** 1, s.l. : RELATEC, Vol. 4. ISSN 1695-288X.

13. **OpenSourceCMS.com.** Open Source CMS Demos & Information. [En línea] 2012. Disponible en: http://www.opensourcecms.com.

14. *Estudio docente del Sistema Web de Apoyo a la Docencia en alumnado del Máster.* **Cambil Martín, Jacobo y Correa Rodríguez, M.** 2, Madrid : Scientia, 2007, Vol. 6. ISSN1989-7375.

15. **Álvarez Valdés, Ana María y Montes de Oca Rodríguez, Osmany.** *Desarrollo del módulo Ejercicios de la Colección Multisaber en su versión multiplataforma.* La Habana : Universidad de las Ciencias Informáticas, 2010.

16. **Carrillo Pérez, Isaías, Pérez González, Rodrigo y Rodríguez Martín, Aureliano David.** *Metodología de Desarrollo del Software.* 2008.

17. **Canós, José H., Letelier, Patricio y Penadés, María del Carmen.** *Metodologías Ágiles en el Desarrollo de Software.* Valencia : Universidad Politécnica de Valencia.

18. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* Wesley : Pearson Adisson, 2000.

19. **Galves, Jorge y Gómez Gallego, Juán Pablo.** *Fundamentos de la Metodología RUP.* Prereira : Universidad Tegnológica de Pereira, 2004.

20. **Popkin Software and Systems.** *Modelado de Sistemas con UML.*

21. **Sierra, María.** *Trabajando con Visual Paradigm for UML.* Cantabria : Universidad de Cantabria – Facultad de Ciencias.

22. *Conferencia de Historia de la Informática: Historia de los lenguajes de programación.* **UCI, Cátedra Historia de la Informática.** Ciudad de la Habana : s.n., 2008.

23. **Brandon Satrom.** MSDN Magazine. *Creación de aplicaciones con HTML5: lo que necesita saber.* [En línea] Agosto de 2011. [Citado el: 14 de Enero de 2012.] Disponible en: http://msdn.microsoft.com/es-es/magazine/hh335062.aspx.

24. **Freddy Vega, John y Van Der Henst, Christian.** *Guía de HTML. El presente de la web.* 2011.

25. **Eguíluz Pérez, Javier;.** Cascading Style Sheets home page. [En línea] 2 de Enero de 2009. Disponible en: http://www.librosweb.es/css_avanzado.

26. **Zeldman, Jeffery.** *Diseño con estándares web.*

27. **Damián Pérez Valdés.** Los diferentes lenguajes de programación para la web. [En línea] 2007. [Citado el: 14 de Enero de 2012.] Disponible en: http://www.maestrosdelweb.com/principiantes/los-diferentes-lenguajes-de-programacion-para-la-web/.

28. **Garcia, Javier.** *Aprenda Java como si estuviera en primero.* s.l. : Escuela Superior de Ingenieros Industriales, San Sebastian, 2005.

29. **Potencier, Fabien y Zaninotto, François.** *Symfony 1.2, la guía definitiva.* 2008.

30. **Valdés, Mayliuvis Esquijarrosa y Díaz, Ernesto Vladimir Pereda.** *Desarrollo del módulo iTopics de la plataforma educativa ZERA.* [PDF] La Habana : s.n., 2011.

31. **jQuery UI Team.** jQuery User Interface. [En línea] 2010. [Citado el: 15 de Enero de 2012.] Disponible en: http://jqueryui.com/about.

32. Sitio Oficial de jQuery. [En línea] 2010. [Citado el: 15 de Enero de 2012.] Disponible en: http://jquery.com/.

33. **Symfony.** Sitio oficial de Symfony en español. [En línea] 2012. [Citado el: 15 de Enero de 2012.] Disponible en: http://www.symfony.es/que-es-symfony/.

34. **SesioLabs.** *Symfony and Doctrine with sfDoctrinePlugin.* 2009.

35. **Mata, Manel Pérez.** TecnoRetales. *Qué es Doctrine ORM?* [En línea] 3 de Julio de 2009. [Citado el: 15 de Enero de 2012.] Disponible en: http://www.tecnoretales.com/programacion/que-es-doctrine-orm/.

36. Entornos de Desarrollo Integrado. [En línea] [Citado el: 15 de Enero de 2012.] Disponible en: http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf.

37. **The Eclipse Foundation.** Eclipse. [En línea] 2012. [Citado el: 15 de Enero de 2012.] Disponible en: http://eclipse.org/.

38. **PostgreSQL Global Development Group.** PostgreSQL 8.4.10 Documentation. *What is PostgreSQL?* [En línea] 2012. [Citado el: 16 de Enero de 2012.] Disponible en: http://www.postgresql.org/docs/8.4/static/intro-whatis.html.

39. **Netcraft Ltd.** Netcraft Ltd. [En línea] 2011. [Citado el: 16 de Enero de 2012.] Disponible en: http://news.netcraft.com/archives/2011.

40. **Hernández, Yadima Morales y Vega, Joel Tamayo.** *Desarrollo del módulo General de la Colección El Navegante.* Universidad de las Ciencias Informáticas, La Habana : PDF, 2011.

41. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 1999.

42. **Craig, Larman.** *UML y patrones.* s.l. : Prentice Hall, 2004. ISBN 978-8420534381.

43. **Erich, Gamma, y otros, y otros.** *Patrones de diseño.* s.l. : Pearson Educación, 2003. ISBN 9788478290598.

44. **Freeman, Eric y Robson, Elisabeth.** *Head First Design Patterns.* s.l. : O'Reilly Media, 2004. ISBN 978-0596007126.

45. **IEEE.** *Estándar para Metadatos de Objetos Educativos.* 2002. P1484.12.1-2002.

46. **Real Academia Española.** Diccionario de la lengua española - Vigésima segunda edición. [En línea] 07 de 2010. Disponible en: http://buscon.rae.es/draeI/.