Universidad de las Ciencias Informáticas
Facultad 4

Componentes Tree, Drag and Drop y Web Editor de CRODA bajo las pautas del marco de trabajo Xalix

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores:

Cleydi García Díaz

Dalianne Yenisil Zamora Miclín

Tutor: Ing. Osvaldo Ernesto Stable Vilches

Co-tutor: Ing. Carlos Arturo Rondón Quintas

La Habana, junio 2016

"Año 58 de la Revolución"

Declaración de autoría

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

_____	_____
Firma del autor	Firma del autor
Cleydi García Díaz	Dalianne Yenisil Zamora Miclín
_____	_____
Firma del tutor	Firma del tutor
Ing. Osvaldo Ernesto Stable Vilches	Ing. Carlos Arturo Rondón Quintas

"La educación es el arma más poderosa que puedes usar para cambiar el mundo."

Nelson Mandela.

Agradecimientos

De Cleydi:

En especial agradezco a mis padres por cuidarme, aconsejarme, amarme y sobre todas las cosas estar siempre ahí cuando más los necesitaba. Gracias por todo el esfuerzo que han hecho en convertirme en la persona que soy. Los amo.

A mi hermana Claudia por todo el amor que me ha brindado y por ser un ejemplo a seguir a lo largo de mi vida. Gracias por existir, estoy muy orgullosa de ser tu hermana. Te amo.

A mi sobrino Ronaldo, que aunque no ha nacido, el solo hecho de saber que me va a brindar el privilegio de ser su tía, me ha dado mucha fuerza y alegría.

A mi abuela Angelina por ser una madre, un padre, una hermana, una amiga, por ser la gran persona que es y estar siempre pendiente de mí. Te amo.

A mí cuñado Silvio que es más que un hermano para mí. Gracias por aconsejarme y protegerme. Te quiero mucho.

A Yunier Martell Espinosa por haber sido una persona muy importante en mi vida y haberme apoyado incondicionalmente en las pruebas de ingreso a la universidad. Estés donde estés, siempre tendrás un lugar en mi corazón.

A toda mi familia por brindarme su cariño, en especial a mi tía Ana Gloria por ser como una gran amiga para mí.

A mi novio Yunierkis por escucharme, apoyarme y tener mucha paciencia conmigo. Estoy muy feliz de haberte conocido. Te quiero mucho.

A mi amiga y compañera de tesis Dalianne por ser tan incondicional conmigo. Te quiero mucho.

A mi amiga Dailenis por no haberme defraudado nunca. Gracias por tu amistad.

A todos mis amigos de Artemisa por haberme brindado tan buenos momentos a lo largo de mi vida. Gracias por esos grandes recuerdos.

A mis amigos de la universidad, en especial a Darysleidi, Zuniet, Neidy, Reinier y Roilandy.

A mis compañeros de la Mini-UCI Yaritza, Yinelis, Yeni, Lizandra, Roylan y Pedro.

A mi amigo Mony por ser una gran persona. Te quiero mucho.

A mis compañeros de aula, en especial a Frank, Felo y Marco.

A Yunior por brindarme su ayuda incondicional en el desarrollo de este trabajo.

A todos los profesores que he tenido a lo largo de mi vida como estudiante.

A todos, MUCHAS GRACIAS, este sueño no hubiera sido posible sin su apoyo.

De Dalianne:

Le agradezco a mi madre por haberme dado la vida y apoyarme en todo momento. Gracias por guiar mis pasos y hacer papel de madre, padre y amiga. "TE AMO".

A mi abuelo Julio que en los años que estuvo a mi lado me brindó su amor incondicional, gracias por ocupar el lugar de padre en todo momento. Aunque no esté presente hoy día y no puedas disfrutar de verme, quiero agradecerle por lo que fue, es y será para mí. "TE NECESITO".

A mi abuela Gisela por darme el placer de compartir mis tristezas y mis alegrías junto a ella, por todos sus consejos para poder llegar donde estoy. "GRACIAS POR EXISTIR".

A mi tía Reina por ser como una madre y ayudarme cuando más lo necesitaba.

A Raulito por cuidar de mi madre, ayudarla y apoyarla mientras me encontraba ausente.

Le agradezco a toda mi familia por estar pendiente de mí y brindarme muchas fuerzas para seguir adelante.

A mi compañera de tesis Cleydi por ser mi amiga en las buenas y en las malas. "Gracias por tu cariño incondicional".

A mi amiga Anisleidis por hacer papel de hermana y brindarme su amistad sincera. "Te quiero mucho".

A mi novio Ariel porque en tan poco tiempo supo hacerme reír cuando no tenía ganas, por apoyarme, comprenderme y secar mis lágrimas de una forma inteligente y cariñosa. Quisiera haberte conocido antes.

A mis amigos Yunior y Pedro por sin pensarlo dos veces siempre estar dispuestos a ayudarme cada vez que los necesitaba.

A todas las personas que de una forma u otra formaron parte de mi vida en estos años de universidad, en especial a: Lianne, Danger, Jorge, Yunierkis y Felo.

Dedicatoria

De Cleydi:

La familia no es algo que se escoge, simplemente la recibes al nacer, pero, si me hubiera tocado elegir, sin duda escogería la mía, no puedo pensar en una de la que me sienta más orgullosa. Dedico esta tesis a mi madre, mi padre, mi hermana y mi abuela. Gracias por haberme guiado, gracias por ser quienes son. Los amo.

Resumen

Con el constante desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) se ha logrado un notable avance en el ámbito educativo. La Universidad de las Ciencias Informáticas (UCI) cuenta con un Centro de Tecnologías para la Formación (FORTES) que brinda servicios y productos encargados de dar soporte a la formación aplicando el uso de las TIC. Este centro cuenta con la herramienta de autor web CRODA que se basa en la creación de recursos educativos estandarizados con fines formativos. La herramienta cuenta con los componentes Tree, Drag and Drop, Web Editor, entre otros que garantizan su correcto funcionamiento. Hoy en el centro FORTES surge un marco de trabajo denominado Xalix. CRODA en su actual versión 3.0 no logra acoplarse al marco de trabajo creado para dicho centro. El objetivo de esta investigación se basa en desarrollar los componentes Tree, Drag and Drop y Web Editor bajo las pautas del marco de trabajo Xalix, para satisfacer las principales funcionalidades de CRODA en su versión 3.0. Para lograrlo se realizó un estudio de la herramienta y del marco de referencia, se utilizaron las herramientas y tecnologías que este marco contempla en el desarrollo del Análisis, Diseño, Implementación y Prueba y como resultado se obtuvieron los componentes Tree, Drag and Drop y Web Editor bajo las pautas del marco de trabajo Xalix, con las características necesarias para satisfacer las principales funcionalidades de CRODA y lograr su reutilización en otros proyectos de software.

Palabras claves: CRODA, Tree, Drag and Drop, Web Editor, Xalix.

Índice de contenido

Índice de figuras

Índice de tablas

INTRODUCCIÓN

Desde los orígenes de la didáctica como disciplina pedagógica que se encarga del estudio de las técnicas y métodos de enseñanza, hasta la actualidad, son incontables los recursos utilizados por los educadores para alcanzar una mayor calidad en el proceso de enseñanza-aprendizaje. Esto ha propiciado que con la incorporación de las Tecnologías de la Información y las Comunicaciones (TIC) se haya logrado un notable avance en el ámbito educativo.

La Universidad de las Ciencias Informáticas (UCI) cuenta con el Centro de Tecnologías para la Formación (FORTES) que brinda servicios y productos encargados de dar soporte a la formación aplicando el uso de las TIC, en este centro se realizan importantes acciones con el objetivo de apoyar el proceso docente con el uso de plataformas. Dentro de las mismas, se encuentran en desarrollo las plataformas educativas Moodle y ZERA como LMS (Learning Management System) o Sistema de Gestión de Aprendizaje, RHODA como ROA (Repositorio de Objetos de Aprendizaje) y CRODA como herramienta de autor web.

CRODA cuenta con funcionalidades que permiten generar objetos de aprendizaje (OA) y diseños de aprendizaje (DA) como recursos educativos descritos por metadatos. Tiene como objetivo facilitar la creación de estos recursos haciendo uso de los estándares SCORM (Sharable Content Object Reference Model), que no es más que un conjunto de especificaciones que permiten crear OA estructurados y LOM (Learning Object Metadata) o Metadatos para Objetos de Aprendizaje que es un modelo de datos usado para describir un OA (1). Estos estándares garantizan la interoperabilidad en aplicaciones como ROA y LMS y son soportados en la herramienta con estructuras de datos XML (eXtensible Markup Language) o Lenguaje de Marcado Extensible que permiten almacenar datos en forma legible.

CRODA tiene un total de tres versiones, en la última de ellas se definen 7 módulos que garantizan el correcto funcionamiento de la aplicación con la incorporación de nuevas funcionalidades que brindan grandes mejoras con respecto a versiones anteriores. En un conjunto de estos módulos se evidencia un componente Tree (Árbol), que permite mostrar y modificar los tipos de información organizada jerárquicamente en forma de árbol. Para garantizar las funcionalidades de lectura y escritura de la herramienta, este componente maneja estructuras XML.

CRODA cuenta además con un componente Web Editor (Editor Web) que permite editar los elementos de información de un OA. El mismo contribuye con la asociación y reproducción de

recursos multimedia, su referencia y visualización. Como resultado se obtienen recursos web en formato HTML (Hypertext Markup Language) o Lenguaje de Marcado de Hipertexto.

El componente Drag and Drop (Arrastrar y Soltar) que propone CRODA para garantizar una de sus principales funcionalidades permite la creación de DA reutilizables, haciendo uso de la especificación IMS-LD. Esta especificación es un lenguaje de modelado educativo que tiene como objetivo definir formalmente una estructura semántica para anotar los procesos de enseñanza-aprendizaje y así convertirlos en entidades reutilizables entre diferentes cursos y aplicaciones (2). Teniendo en cuenta esto el componente ofrece un grupo de elementos ya definidos, con un conjunto de políticas de diseño previamente establecidas. Además, facilita el trabajo de los usuarios ya que con la utilización de la tecnología Drag and Drop se muestra una interfaz cómoda y flexible.

En cada una de sus versiones la aplicación muestra funcionalidades que la hacen más actualizada y cuenta con arquitecturas acordes con el desarrollo y las necesidades del centro donde se produce. Hoy en el centro FORTES se realizan un conjunto de transformaciones con el objetivo de homogenizar y facilitar el desarrollo de sus productos, ejemplo de esto es el surgimiento de un marco de trabajo denominado Xalix, el cual es una estructura de soporte definida, en la que un proyecto de software puede ser organizado y desarrollado.

Este marco de trabajo, incluye una base de tecnologías para el desarrollo y define una estructura y un mecanismo de integración para los componentes creados. Adopta una arquitectura basada en componentes, donde cada elemento puede desacoplarse y evolucionar independiente uno del otro y son esas partes las que conforman los productos genéricos una vez ensambladas.

CRODA en su versión liberada 3.0 se desarrolla sobre Symfony2, JQuery, PostgreSQL y Bootstrap, cumpliendo a través de ello lo necesario para formar parte de la estrategia marcaria y la cartera de productos de la UCI. Pero aun con estos avances sus componentes Tree, Drag and Drop y Web Editor no logran acoplarse directamente al marco de trabajo Xalix ya que no se ajustan de forma independiente a la arquitectura. Además, no se encuentran bajo el sistema de integración creado para dicha arquitectura.

Teniendo en cuenta lo anteriormente planteado se define como **problema de investigación:** ¿cómo contribuir con el desarrollo de los componentes Tree, Drag and Drop y Web Editor de CRODA bajo las pautas del marco de trabajo Xalix?

Se define como **objeto de estudio** las pautas del marco de trabajo Xalix.

El **campo de acción** está enfocado a los componentes Tree, Drag and Drop y Web Editor bajo las pautas del marco de trabajo Xalix.

Para darle solución al problema planteado se define como **objetivo general** de la investigación: desarrollar un componente Tree, Drag and Drop y Web Editor bajo las pautas del marco de trabajo Xalix para satisfacer las principales funcionalidades de CRODA.

Teniendo en cuenta el objetivo general se definen los siguientes **objetivos específicos:**

- Construir el marco teórico que sustenta la investigación.
- Diseñar los componentes Tree, Drag and Drop y Web Editor bajo las pautas del marco de trabajo Xalix.
- Implementar los componentes diseñados.
- Validar la solución de los componentes implementados.

Métodos de investigación:

Métodos Teóricos:

- **Analítico –sintético:** se utiliza al descomponer el problema de investigación en elementos por separado y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la propuesta de solución.
- **Modelación:** se utiliza para la realización de modelos en los capítulos 2 y 3.

Métodos Empíricos:

- **Entrevista:** se utiliza para obtener información relacionada con el marco de trabajo Xalix.

Estructura capitular:

Capítulo I Fundamentación teórica:

En este capítulo se hace un estudio de investigaciones anteriores para llegar a una conclusión actual y así conformar el marco teórico, en el cual se aborda todo lo relacionado con los conceptos principales de la presente investigación. Además, se selecciona la metodología de desarrollo de software más adecuada y las herramientas y tecnologías que se ajusten al problema planteado.

Capítulo II Descripción de la propuesta de solución:

Este capítulo contiene una descripción de la propuesta de solución, donde primeramente se presenta el modelo de dominio con el objetivo de comprender el ámbito donde se desarrollan los componentes. Se especifican los requerimientos, descritos posteriormente por historias de usuario y se muestran los artefactos generados por el flujo de trabajo Análisis y Diseño.

Capítulo III Implementación y prueba:

Este capítulo se enmarca en describir como están implementados los componentes y en validar el correcto funcionamiento de los mismos.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

Introducción:

En este capítulo se abordan los elementos teóricos que sustentan la investigación, relacionados con la herramienta de autor web CRODA en su versión 3.0 y el marco de trabajo Xalix. Entre los principales temas se tratan los objetos y diseños de aprendizaje como recursos educativos. Además, se presenta una descripción de la metodología utilizada para guiar el proceso de desarrollo del software, así como las herramientas y tecnologías a utilizar.

1.1 Recursos educativos

En el contexto económico y social actual, universidades de todo el mundo persiguen la excelencia y la calidad, al tiempo que compiten en la localización de recursos y tratan de rentabilizar aquellos de los que ya disponen. El desarrollo y la promoción de los recursos educativos están motivados por el deseo de poner freno a la mercantilización del conocimiento y de proporcionar una alternativa o mejora al paradigma educativo.

Un recurso educativo es un recurso distribuido de forma gratuita y libre de restricciones para modificar, combinar y reutilizar. El mismo está compuesto por contenidos educativos que pueden ser cursos completos, materiales para cursos, materiales multimedia, objetos de aprendizaje (OA) o cualquier otro recurso con fines educativos; todos con una alta pertinencia y certificación de calidad (3). Las plataformas educativas garantizan la accesibilidad y la disponibilidad de los recursos educativos.

La paulatina incorporación de las TIC a la docencia universitaria en los últimos años ha permitido pensar en un modelo de trabajo que precisamente busque la optimización de los recursos educativos. El paradigma de dicho modelo, desarrollado a finales del pasado milenio y caracterizado por sus especiales condiciones para la reutilización, se ha venido a denominar OA.

Un OA es un recurso educativo digital relativamente pequeño y auto-contenido, con una marcada intención formativa, compuesto por uno o varios elementos de información, con un objetivo, descrito con metadatos y con un comportamiento secuenciado que asegure el correcto enlace entre los elementos de su estructura didáctica; concebido para diferentes poblaciones según sus contextos socioculturales para lograr su reutilización e interoperabilidad en varios

entornos b-learning, que contribuya a la transmisión de conocimiento y la formación de valores tanto en los profesores, diseñadores, como en los usuarios finales (3).

Los OA son el recursos educativo por excelencia en la docencia universitaria actual, pero es válido aclarar que los diseños de aprendizaje (DA) son muy importantes para todo el proceso de enseñanza-aprendizaje, ya que son una descripción de un escenario pedagógico y describen la experiencia de todo el proceso educativo.

Se refieren a la actividad de diseñar unidades de aprendizaje, actividades de aprendizaje o entorno de aprendizaje. La idea básica de los DA es que representan un vocabulario que los usuarios de cualquier enfoque pedagógico entienden, y en la que los diseños existentes puede ser traducidos (4).

Hay varios factores que influyen en las decisiones sobre los DA, incluyendo las metas del aprendizaje, las características de los alumnos, su comodidad con el proceso de aprendizaje y con los demás estudiantes, su familiaridad con el contenido, la magnitud del cambio esperado, el entorno laboral de los educadores y los recursos disponibles para apoyar el aprendizaje (4). Los DA aumentan la eficacia docente y los resultados para los estudiantes, integrando teorías, investigaciones y modelos de aprendizaje humano para lograr los resultados previstos.

Lograr un aprendizaje significativo en los estudiantes requiere de educadores altamente capacitados, que no sólo impartan clases, sino que también contribuyan a la creación de nuevas metodologías, materiales y técnicas. De ahí la importancia de utilizar estos recursos educativos, cuyo objetivo es fungir como facilitadores y potencializadores de la enseñanza que se quiere significar.

1.2 Herramienta de autor web CRODA

Una herramienta de autor es una aplicación que disminuye el esfuerzo a realizar por los usuarios, ofreciéndoles indicios, guías, elementos predefinidos, ayudas y una interfaz amigable para crear cualquier tipo de recurso educativo. Este tipo de herramienta brinda cierto grado de protagonismo a personas con conocimientos básicos de informática. La misma se caracteriza por ser fácil de utilizar y permite crear un entorno de aprendizaje dinámico bajo iniciativas propias, basta con poner un poco de empeño y estar motivado a ampliar conocimientos.

La herramienta de autor web CRODA en su versión 3.0 se basa en la creación de recursos educativos estandarizados con fines formativos, logrando su interoperabilidad para su

utilización en aplicaciones como ROA y LMS. La herramienta cuenta con funcionalidades que permiten generar OA y DA, descritos por metadatos, que no son más que datos que describen otros datos y es una información relativa a los propios datos que facilitan su catalogación. Además, aporta facilidades para el trabajo colaborativo por parte de los usuarios creadores durante la confección de estos recursos y mantiene el control y la seguridad necesaria de sus funcionalidades y sus usuario (1). Para la creación de recursos educativos estandarizados, esta herramienta hace uso de los estándares SCORM (Sharable Content Object Reference Model) y LOM (Learning Object Metadata) o Metadatos para Objetos de Aprendizaje, estos estándares son soportados con estructuras XML (eXtensible Markup Language) o Lenguaje de Marcado Extensible.

SCORM es un conjunto de estándares y especificaciones que permiten la creación de OA estructurados. Un paquete SCORM es un bloque de material web empaquetado de una manera que sigue el estándar SCORM de objetos de aprendizaje. Estos paquetes pueden incluir páginas web, gráficas, programas Javascript, presentaciones Flash y cualquier otra cosa que funcione en un navegador web. El módulo SCORM permite cargar fácilmente cualquier paquete SCORM estándar y lo convierte en parte de un curso (5).

LOM es un modelo de datos usado para describir un OA y otros recursos digitales similares encargados de apoyar el aprendizaje. Este estándar tiene como propósito ayudar a reutilizar los OA y facilitar su internacionalidad. El mismo es reconocido usualmente en el contexto de sistemas de aprendizaje on-line (6).

XML no es un lenguaje en particular, sino un sistema que permite definir lenguajes de acuerdo a diferentes necesidades. Facilita la organización, representación, almacenamiento, transmisión y etiquetado de información a través de medios de comunicación digital. Contiene un formato de texto simple, muy flexible y originalmente diseñado para cumplir con los retos de la publicación electrónica a gran escala (7).

Esta estructura desempeña un papel muy importante en el intercambio de una amplia variedad de datos en la web. Permite jerarquizar, estructurar y describir los contenidos dentro del propio documento, así como la reutilización de partes del mismo (7). A partir de un documento XML se pueden generar archivos en otros formatos, de esta forma la información puede ser presentada de una manera visual para su lectura por las personas y el XML sólo quedaría para ser entendido por los programas.

CRODA 3.0 define 7 módulos que garantizan el correcto funcionamiento de la aplicación. A continuación se muestra una tabla con estos módulos y una breve descripción de los mismos:

Módulos de CRODA	**Descripción**
PatternBundle	Brinda funcionalidades que permiten la creación de patrones de objetos de aprendizaje.
ContentBundle	Brinda funcionalidades que permiten la creación de recursos educativos.
AdminBundle	Brinda funcionalidades que permiten la administración del sistema
LearningDesignBundle	Brinda funcionalidades que permiten la creación de diseños de aprendizaje.
MainBundle	Encargado de las funcionalidades del módulo llamado Principal.
MetadatoBundle	Brinda funcionalidades que permiten la creación de los metadatos.
ColaborationBundle	Brinda funcionalidades que permiten la actividad colaborativa entre los usuarios.

Tabla 1: Descripción de los módulos de CRODA v3.0.

El módulo PatternBundle contiene un componente Tree que permite definir la estructura de los patrones de OA y el módulo MetadatoBundle contiene un componente Tree que permite definir la estructura de los metadatos.

El módulo ContentBundle contiene un componente Tree que permite definir la estructura de los recursos educativos y un componente Web Editor que permite editar los elementos de información de un OA.

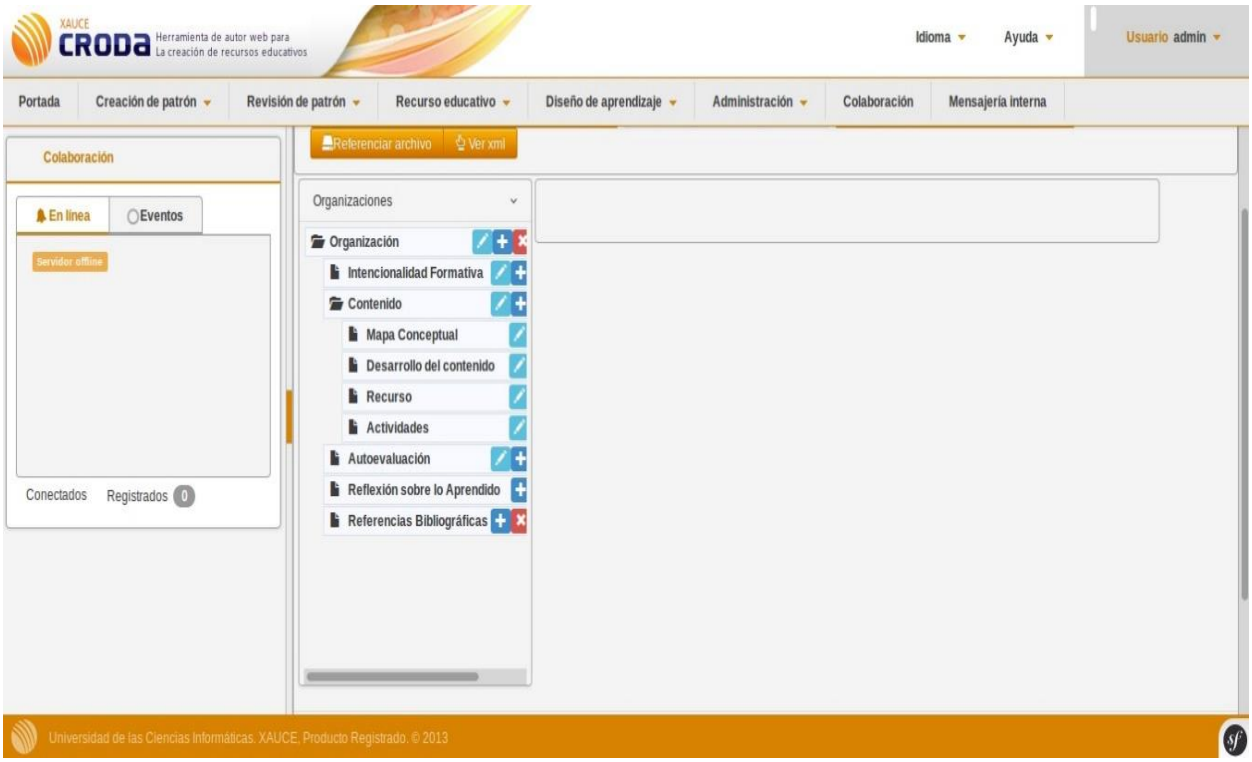


Ilustración 1: Componente Tree del módulo ContentBundle en CRODA v3.0.

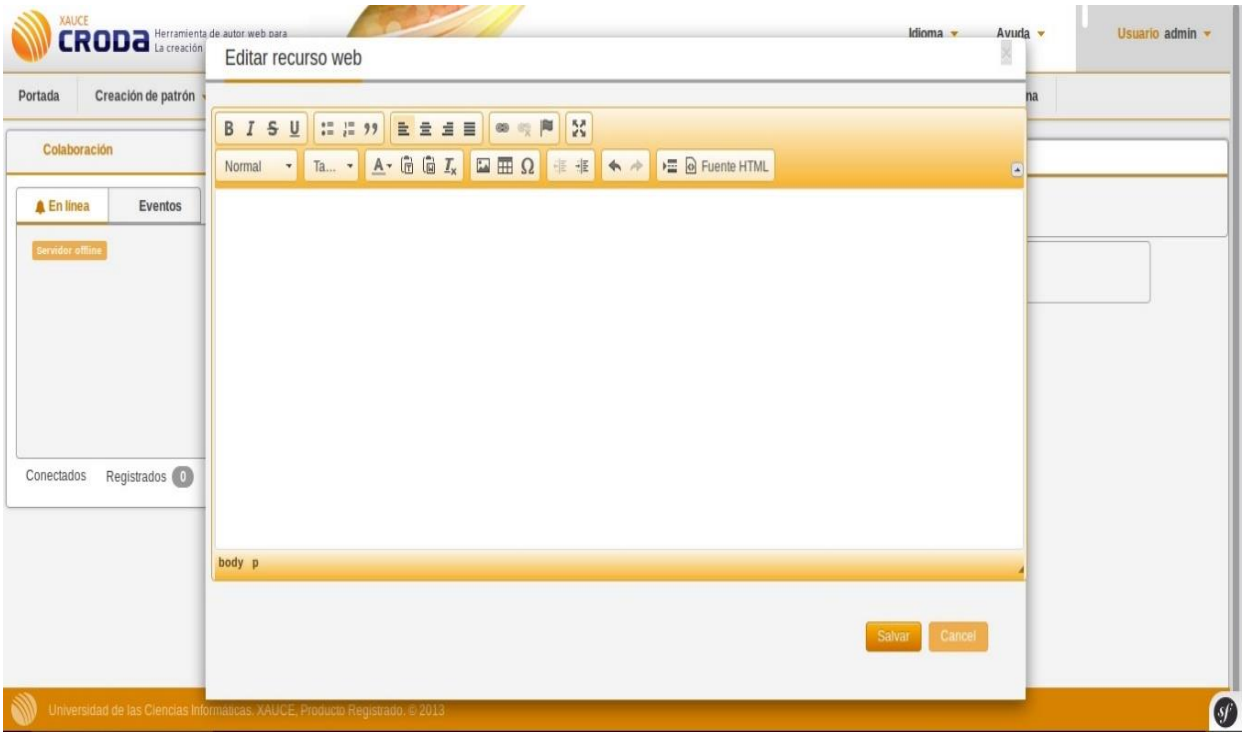


Ilustración 2: Componente Web Editor en CRODA v3.0.

El módulo LearningDesignBundle contiene un componente Tree que permite definir la estructura de los DA y un componente Drag and Drop que permite la creación de DA.

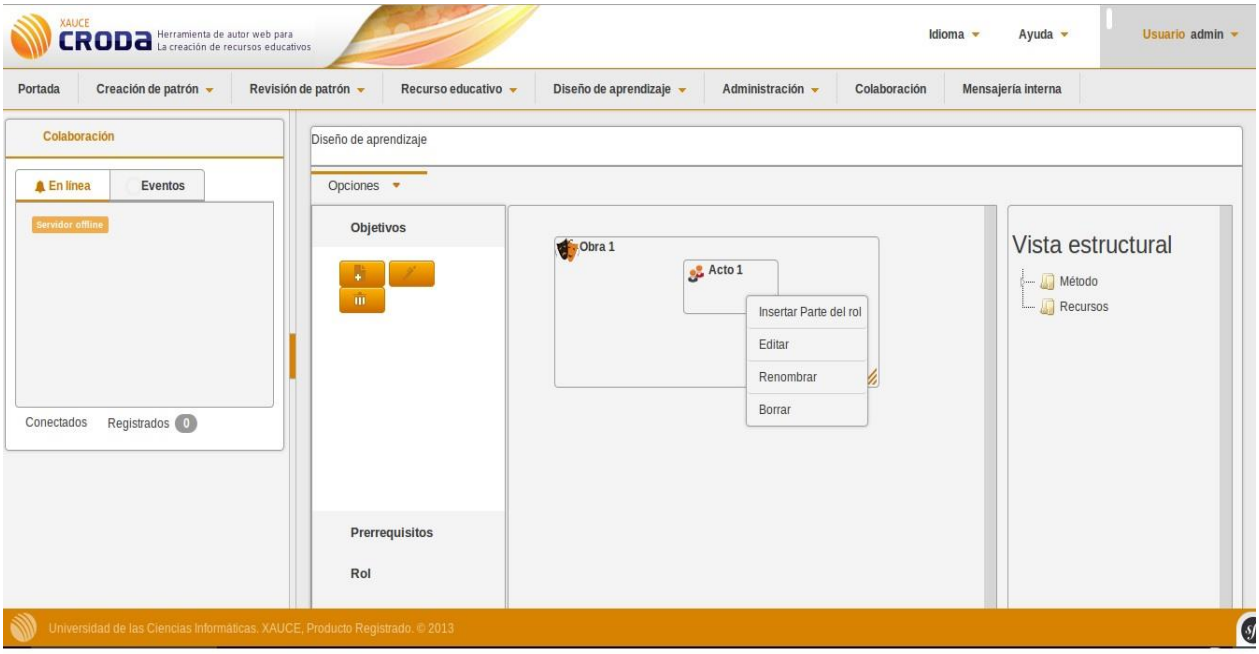


Ilustración 3: Componente Drag and Drop en CRODA v3.0.

Teniendo en cuenta que la herramienta de autor web CRODA utiliza estructuras XML para soportar los estándares SCORM y LOM, es imprescindible la persistencia de un componente Tree que maneje estructuras XML. Este componente muestra la información estructurada jerárquicamente en forma de árbol y le brinde facilidades de modificación al usuario con funcionalidades de lectura y escritura. Permite además la conversión a otras estructuras de datos como JSON (JavaScript Object Notation) o Notación de Objetos JavaScript, ya que para su funcionamiento interno se utiliza la librería JavaScript del lado del cliente JTree.

CRODA se encarga de crear OA como una de sus principales funcionalidades. Los OA están compuestos por uno o varios elementos de información y son auto-contenidos, por lo que es necesaria la persistencia de un componente Web Editor que permita editar estos elementos de información con la asociación y reproducción de recursos multimedia, su referencia y visualización. Como resultado se obtienen recursos web en formato HTML.

La herramienta utiliza la especificación IMS-LD (IMS-Learning Design) para la creación de DA reutilizables. Esta especificación es un lenguaje de modelado educativo que tiene como

objetivo definir formalmente una estructura semántica para anotar los procesos de enseñanza-aprendizaje y así convertirlos en entidades reutilizables entre diferentes cursos y aplicaciones (2). El principal potencial de IMS-LD es que permite que sean los propios educadores, que son los que realmente entienden las necesidades y objetivos finales de las aplicaciones educativas, los que lideren el desarrollo y mantenimiento de dichas aplicaciones.

Los DA describen la experiencia de todo el proceso educativo. La persistencia de un componente Drag and Drop que garantice la creación de DA haciendo uso de la especificación IMS-LD es imprescindible para el correcto funcionamiento de CRODA. Este componente les facilita el trabajo a los usuarios ya que se muestra una interfaz cómoda y flexible, ofreciéndoles un grupo de elementos ya definidos, con un conjunto de políticas de diseño previamente establecidas.

La herramienta de autor web CRODA se encuentra en desarrollo en la Universidad de las Ciencias Informáticas (UCI), específicamente en el Centro de Tecnologías para la Formación (FORTES) que brinda servicios y productos encargados de dar soporte a la formación aplicando el uso de las TIC. En su versión 3.0 se desarrolla sobre Simfony2, Jquery, PostgreSQL y Bootstrap, cumpliendo a través de ello lo necesario para formar parte de la estrategia marcaria y la cartera de productos de la UCI.

Partiendo de requisitos previos ya presentes en versiones anteriores, en su versión 3.0 define nuevos requisitos que permiten la incorporación de otras funcionalidades, como:

- Soporte para Internacionalización del sistema.
- Mejorar interoperabilidad con ROA (Repositorio de Objetos de Aprendizaje).
- Optimización del mecanismo de instalación del sistema.
- Diseño gráfico propuesto en la estrategia marcaria da la UCI.
- Diseño con enfoque adaptativo.

A pesar de los avances que se han logrado en esta herramienta, no es posible la reutilización de los componentes Tree, Drag and Drop y Web Editor en otros proyectos de software, ya que no existe una solución estándar de los mismos y están contenidos en los 7 módulos definidos en la herramienta, es decir, no se pueden desacoplar independientemente para conformar un producto genérico.

1.3 Marco de trabajo Xalix

Como resultado de la entrevista realizada al Ing. Miguel Medina Ramírez, especialista del proyecto Zera, en el cual se encuentra en desarrollo el marco de trabajo Xalix, se obtuvieron los siguientes resultados:

El marco de trabajo Xalix extiende el concepto de bundle de Symfony2, tomándolos como plugin y creando un sistema de gestión de los mismos que permite su instalación y desinstalación. Cada vez que se adiciona un nuevo plugin a la aplicación se ejecutan un conjunto de acciones y luego se instala el plugin, al igual sucede en todo el proceso de desinstalación. Lo que marca la diferencia principal de Xalix con respecto a Symfony2 es todo este proceso de integración de los componentes al núcleo, ya que en este último a la vez que se adiciona un bundle no se puede desactivar, a no ser que se elimine el mismo. Cada plugin debe extender del bundle *(Xalix\InstallationBundle\Bundle\PluginBundle)*. Esta clase redefine el método *getConfiguration* que permite definir los archivos que contienen las rutas y las configuraciones de los plugin.

Diferencias de Xalix con respecto a Symfony2:

- El *appKernel* ya no almacena la referencia a los bundles de tercero, para ello se incorporó el fichero *bundles.ini* en el directorio *config* de *app*.
- Se adiciona el directorio *plugins* en la raíz de la aplicación donde existe el fichero *plugins.yml* que sirve como mecanismo de almacenamiento del estado de instalación de los plugins.
- Se elimina el fichero *config.yml* del directorio *app/config* y la configuración es controlada por cada plugin.

A continuación se muestra una tabla con los componentes que conforman la base del marco de trabajo Xalix y una breve descripción de cada uno:

Componentes de Xalix	**Descripción**
AdminGenerator	Brinda funcionalidades que generan CRUD básicos sobre las entidades del proyecto.
NomenclatorBundle	Brinda funcionalidades que gestionan los nomencladores.

WebInstalerBundle	Brinda funcionalidades que permiten la instalación del sistema vía web.
InstallationBundle	Brinda funcionalidades que permiten la instalación fácil del sistema.
DatabaseConfigBundle	Brinda funcionalidades que guardan toda la configuración del sistema en la base de datos.
CoreBundle	Núcleo de la arquitectura donde aparecen todas las configuraciones.
KernelBundle	Dependiente de CoreBundle.
MigrationBundle	Brinda funcionalidades que permiten la migración del sistema.
ExceptionBundle	Brinda funcionalidades que gestionan las excepciones personalizadas.
SeguridadBundle	Brinda funcionalidades que manejan la seguridad del sistema.
NotificationBundle	Brinda funcionalidades que permiten enviar notificaciones al usuario en tiempo real.
EasyAdminBundle	Brinda funcionalidades que permiten gestionar las entidades.

Tabla 2: Descripción de los componentes de Xalix.

En este marco los repositorios de las entidades deben estar ubicados en un directorio llamado *Repository* y no directamente dentro de *Entity* y los mensajes flash mostrados al usuario serán manejados por la clase *FlashMessageManager*.

Modo de instalación de la aplicación:

1. Descarga la aplicación desde la url *[https://repo-fortes.uci.cu/mooc/trunk/zera2/].*
2. Actualizar las dependencias del sistema con el comando *[php composer.phar update].*
3. Ejecutar el comando *[xalix:install].*

4. Eliminar la cache.

Xalix incluye la siguiente base de tecnologías para el desarrollo de sus productos:

- Framework de desarrollo: Symfony v2.7.x.
- Librería de JavaScript: JQuery v2.0 con JQuery UI v1.10.0.
- Librería de CSS: Bootstrap v3.0.
- Lenguaje de programación del lado del servidor: PHP: v5.4.x.
- Lenguaje de programación del lado del cliente: HTML v5.0.

Este proyecto utiliza las siguientes herramientas para guiar su proceso de desarrollo:

- Herramienta de modelado: Visual Paradigm v8.0.
- Entorno de desarrollo integrado: PhpStorm v8.0.
- Sistema gestor de base de datos: PostgreSQL v9.4.x.
- Servidor web: Apache v2.4.7.

La utilización del marco de trabajo Xalix es muy ventajosa ya que es una estructura de soporte definida en la que un proyecto de software puede ser organizado y desarrollado, busca homogenizar y facilitar el desarrollo de sus productos, adopta una arquitectura basada en componente donde cada elemento puede desacoplarse y evolucionar independientemente uno del otro, lo que conforma un producto genérico una vez ensamblado, incluye una base de tecnologías para su desarrollo y un mecanismo de integración para los componentes creados.

1.4 Metodología de desarrollo de software

Una metodología de desarrollo de software comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto de software, desde que surge la necesidad del mismo, hasta que cumplimos el objetivo para el que fue creado. La misma optimiza el proceso de desarrollo, define qué hacer, cómo y cuándo durante todo el desarrollo y mantenimiento de un proyecto (8).

Según la filosofía de desarrollo se pueden clasificar las metodologías en dos grupos: las metodologías tradicionales, que se basan en una fuerte planificación durante todo el desarrollo, y las metodologías ágiles, en las que el desarrollo de software es incremental, cooperativo, sencillo y adaptado (8). Desarrollar un buen software depende de un gran número de

actividades y etapas, donde el impacto de elegir la metodología para un equipo, en un determinado proyecto, es trascendental para el éxito del producto.

1.4.1 Metodologías ágiles

Este enfoque nace como respuesta a los problemas que puedan ocasionar las metodologías tradicionales y se basa en dos aspectos fundamentales, retrasar las decisiones y la planificación adaptativa. Basan su fundamento en la adaptabilidad de los procesos de desarrollo. Están especialmente preparadas para cambios durante el proyecto. Es un proceso menos controlado y con pocos principios. El cliente forma parte del equipo de desarrollo y este grupo contiene un número pequeño de integrantes trabajando en un mismo sitio. Genera pocos artefactos y existen pocos roles en el equipo de desarrollo (8).

Las metodologías ágiles ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan.

AUP-UCI

La metodología AUP (Agile Unified Process) o Proceso Unificado Ágil se basa en el desarrollo dirigido por pruebas, en el modelado, en la gestión de cambios ágil y en la refactorización de Base de Datos para mejorar la productividad. Contiene 4 fases que transcurren de manera consecutiva, las fases son: Inicio, Elaboración, Construcción y Transición. Define 7 disciplinas, de ellas 4 ingenieriles y 3 de gestión de proyectos (9).

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto, dígase equipo de desarrollo y recursos, exigiéndose así que el proceso sea configurable; se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Todo esto tomando como apoyo el modelo CMMI-DEV v1.3, el cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad.

De las 4 fases que propone AUP se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que llamaremos Ejecución y se agrega la fase de Cierre. A continuación se muestra una tabla con la descripción de cada fase definida en la metodología AUP-UCI (9).

Fases	Descripción
Inicio	Se llevan a cabo las actividades relacionadas con la planeación. Se realiza un estudio inicial de la organización cliente que brinda información fundamental acerca del alcance del proyecto. Se realizan estimaciones de tiempo, esfuerzo y costo para decidir si se ejecuta o no el plan.
Ejecución	Se ajustan los planes del proyecto, considerando los requisitos, la arquitectura, el modelado del negocio, el diseño, la implementación y la liberación del producto.
Cierre	Se analizan tanto los resultados del proyecto, como su ejecución y se realizan las actividades formales de cierre del proyecto.

Tabla 3: Fases de la metodología AUP-UCI.

Con el objetivo de lograr el objetivo general definido en la presente investigación se realizan en cada fase las siguientes tareas:

- Inicio: encuentro con el cliente para definir las características generales de los componentes Tree, Drag and Drop y Web Editor de la herramienta de autor web CRODA en su versión 3.0.
- Ejecución: descripción de una propuesta de solución teniendo en cuenta la elaboración de un modelo conceptual y la especificación de los requisitos funcionales y no funcionales. Ejecución de los flujos análisis, diseño e implementación.
- Cierre: proceso de prueba al producto final para validar el correcto funcionamiento del mismo.

AUP-UCI propone 7 disciplinas principales y cubre con las áreas de procesos que define CMMI-DEV v1.3 para el nivel 2, los procesos de gestión y soporte, que serían Gestión de la configuración, Planeación de proyecto y Monitoreo y control de proyecto. Para una mayor comprensión se muestra la siguiente tabla (9).

Disciplinas AUP-UCI	Objetivos de las Disciplinas AUP-UCI

Modelado de negocio	Comprender los procesos de negocio de una organización, para tener garantías de que el software desarrollado va a cumplir su propósito.
Requisitos	Desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto.
Análisis y diseño	Si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos y una descripción que sea fácil de mantener, para ayudar a la estructuración del sistema (incluyendo su arquitectura).
Implementación	A partir de los resultados del Análisis y Diseño se construye el sistema.
Pruebas internas	Verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales al ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible, componentes de prueba ejecutables para automatizar las pruebas.
Pruebas de liberación	Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
Pruebas de aceptación	Es la prueba final antes del despliegue del sistema. Verifica que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

Gestión de la configuración	Se cubren con las áreas de procesos que define CMMI-DEV v1.3 para el nivel 2.
Planeación de proyecto	
Monitoreo y control de proyecto	

Tabla 4: Disciplinas de la metodología AUP-UCI.

Para modelar el negocio en la disciplina Modelado de negocio AUP-UCI propone las siguientes variantes (9):

- Casos de Uso del Negocio (CUN).
- Descripción de Proceso de Negocio (DPN).
- Modelo Conceptual (MC).

A partir de las variantes anteriores y teniendo en cuenta que existen 3 formas de encapsular los requisitos, se condicionan cuatro escenarios en AUP-UCI para modelar el sistema en la disciplina Requisitos, los escenarios son (9):

- Escenario 1: proyectos que modelen el negocio con CUN solo pueden modelar el sistema con Casos de Uso del Sistema (CUS).
- Escenario 2: proyectos que modelen el negocio con MC solo pueden modelar el sistema con Casos de Uso del Sistema (CUS).
- Escenario 3: proyectos que modelen el negocio con DPN solo pueden modelar el sistema con Descripción de Requisitos por Proceso (DRP).
- Escenario 4: proyectos que no modelen el negocio solo pueden modelar el sistema con Historias de Usuario (HU).

Para guiar el proceso de desarrollo de la presente investigación se considera AUP-UCI como la metodología de referencia. Con el objetivo de tener una visión estructurada del negocio, en la disciplina Modelado de negocio se propone la realización de un Modelo Conceptual. Para modelar el sistema en la disciplina de Requisitos se propone utilizar el escenario 4 basado en Historias de usuario. Con el objetivo de ayudar a la estructuración del sistema se propone refinar los requisitos con la realización de la disciplina Análisis y diseño. Posteriormente se pasara a la construcción del sistema y por último se realizaran las pruebas necesarias al producto final.

1.5 Herramienta de modelado

Una herramienta de modelado no es más que una herramienta que se emplea para la creación de modelos de sistemas que ya existen o se desarrollaran, estos modelos permiten una visión futura del sistema y deben ser fáciles de entender.

Visual Paradigm. Versión 8.0

Visual Paradigm es una herramienta diseñada para modelar el sistema de información de negocio y gestionar los procesos de desarrollo. Permite el diseño de software con UML (Lenguaje de Modelado Unificado) y es multiplataforma. Ofrece a las empresas de software completos set de herramientas que necesitan para los requisitos de la captura, análisis de procesos, diseño de sistemas y diseño de base de datos (10).

El lenguaje de modelado UML es un lenguaje para especificar, construir, visualizar y documentar los artefactos de un sistema de software orientado a objetos. Un artefacto es una información que es utilizada o producida mediante el proceso de desarrollo de software (11).

Para el desarrollo de la investigación este lenguaje será utilizado para generar los siguientes artefactos ingenieriles:

- Modelo de dominio: permite identificar los principales conceptos del sistema y las relaciones que se establecen entre ellos.
- Diagrama de clases: tipo de diagrama que describe la estructura de un sistema a través de sus clases, ya sea en el análisis o en el diseño.
- Diagrama de secuencia del diseño: modela la interacción entre los objetos de un sistema a través del tiempo.
- Modelo de datos relacional: describe la estructura de la información que gestiona el sistema.
- Diagrama de componentes: divide el sistema de software en componentes y representa las relaciones entre ellos.

Con la utilización de Visual Paradigm como herramienta de modelado se pueden modelar las diferentes perspectivas del sistema y sus artefactos en cada fase. Permite la generación de código e ingeniería inversa, es decir, brinda la posibilidad de generar código a partir de los diagramas, así como obtener los diagramas a partir del código. Posee una interfaz gráfica enfocada a facilitar la interacción y el trabajo del usuario.

1.6 Lenguajes de desarrollo

Un lenguaje de programación es el conjunto de sentencias que sirven para decirle a una computadora qué es lo que tiene que hacer. Los lenguajes de programación pueden dividirse en dos grupos, los lenguajes del lado del servidor y los lenguajes del lado del cliente, donde la filosofía de la arquitectura Cliente/Servidor es válida para los dos, en los entornos web.

Del lado del servidor se encuentra PHP (Hypertext Preprocessor) o Procesador de Hipertexto que es un lenguaje reconocido, ejecutado e interpretado por el propio servidor y luego es enviado al cliente en un formato comprensible para él.

Del lado del cliente tenemos a Hypertext Markup Language (HTML) o Lenguaje de Marcado de Hipertexto, Cascading Style Sheets (CCS) u Hoja de Estilos en Cascadas y JavaScript, estos lenguajes basan su procesamiento en un cliente web, es decir, son interpretados por el navegador.

PHP. Versión 5.4.x

Es un lenguaje de programación de código abierto y una herramienta poderosa para hacer páginas web dinámicas e interactivas. Tiene soporte para una gran cantidad de sistemas operativos del mercado y para una gran variedad de sistemas gestores de base de datos, incluyendo PostgreSQL. Es compatible con casi todos los servidores utilizados en la actualidad, incluyendo Apache. Además es gratuito y fácil de aprender (12).

El uso de PHP es ventajoso ya que es simple para principiantes, pero a su vez, ofrece características avanzadas para los programadores profesionales. Se destaca por su potencia, alto rendimiento y escasez de consumo de recursos. Posee librerías que facilitan el trabajo con servicios web. Se caracteriza por el procesamiento de la información en formularios y por la manipulación de varios formatos, incluyendo XML.

DOMDocument (Document Object Model) es una librería PHP del lado del servidor que permite manipular documentos XML. Antes de poder utilizar sus funciones, transforma internamente el XML original en una estructura más fácil de manejar formada por una jerarquía de nodos interconectados en forma de árbol. El árbol generado representa los contenidos del archivo original y sus relaciones mediante las ramas del árbol que conectan los nodos (13).

La ventaja de emplear DOM es que le permite al programador disponer de un control muy preciso sobre la estructura del documento que está manipulando. Proporciona funciones que permiten añadir, eliminar, modificar y reemplazar cualquier nodo de cualquier documento de forma sencilla.

HTML. Versión 5.0

Es el lenguaje de marcado básico para la elaboración de páginas web. Está orientado fundamentalmente a la presentación de los datos, definiendo comandos y etiquetas que permiten delimitar la estructura lógica de un documento web. Establece la estructura y mayormente el contenido de un sitio web. Indica al navegador donde colocar cada texto, cada imagen o cada video y la forma que tendrán estos al ser colocados en la página (14).

A diferencia de otras versiones de HTML, los cambios en HTML5 comienzan añadiendo semántica y accesibilidad implícitas, especificando cada detalle y borrando cualquier ambigüedad, es decir, proporciona una compatibilidad mejorada para crear aplicaciones web que puedan interactuar de manera más sencilla y efectiva con el usuario y los servidores. Permite validar la información insertada por el usuario sin necesidad de JavaScript.

CSS. Versión 3.0

Es un lenguaje que se utiliza para definir el estilo de presentación que tendrá un documento HTML o XML. Se basa fundamentalmente en separar la estructura del documento de la presentación del mismo. La separación de los contenidos y su presentación es muy ventajosa ya que mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el documento en infinidad de dispositivos diferentes (14).

Se utilizará CSS en su versión 3.0 ya que permite a los desarrolladores mantener un mayor control sobre la apariencia de las páginas, lo que facilita su modificación. Gracias al uso de CSS3 somos mucho más dueños de los resultados finales de nuestra página web.

JavaScript. Versión 1.8

Es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se basa fundamentalmente en la creación de efectos especiales a los componentes visuales de una interfaz y en la creación de páginas web

dinámicas. Maneja objetos dentro de las páginas web y sobre esos objeto es posible definir diferentes eventos, dichos objetos facilitan la programación de páginas interactivas. Permite la verificación y procesamiento de los datos introducidos por el usuario antes de ser enviados al servidor (14).

El uso de este lenguaje es muy favorable ya que responde a eventos en tiempo real, es posible cambiar totalmente el aspecto de la página al gusto del usuario, evitando tener en el servidor una página para cada gusto. En la creación de formularios dinámicos es muy útil ya que con HTML5 y CSS3 no es posible eliminar o crear elementos a partir de las interacciones del usuario. Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web.

JSON (JavaScript Object Notation) o Notación de Objetos JavaScript es un formato ligero ideal para el intercambio de datos y está basado en un subconjunto del lenguaje de programación Java Script. Este formato es simple de leer y escribir para humanos, mientras que para las máquinas es simple interpretarlo y generarlo (15).

JTree es una librería JavaScript del lado del cliente que permite la conversión a estructuras de datos como JSON. Esta librería se basa en un modelo de objeto independiente para sostener y representar los datos que en él se muestran. Representa un árbol jerárquico que se compone de nodos a los que se les puede vincular eventos o acciones dependiendo de las necesidades del cliente. Puede ser utilizado para mostrar rutas de directorios, ejemplo de esto es el explorador de carpetas o el árbol de directorios que nos permite navegar fácilmente (16).

1.7 Framework de desarrollo

Es un marco de trabajo con funcionalidades ya desarrolladas y probadas que pueden servir de base para la organización y desarrollo de software, facilitando la creación del mismo.

Symfony. Versión 2.7.x

Es un framework de desarrollo que permite crear aplicaciones y sitios web rápidos y seguros de forma profesional. Es un proyecto PHP de software libre. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en el proyecto. El núcleo es la pieza central del framework y es el responsable de inicializar la configuración de la aplicación y arrancar los bundles (17).

Este framework incluye varias herramientas gráficas y de consola para depurar fácilmente los errores que se produzcan en las aplicaciones. La seguridad es un parámetro muy importante para este proyecto, ya que antes de su lanzamiento una empresa independiente se encargó de una auditoria para garantizar la misma. Se trata del framework más popular en el mundo hispano. Anualmente se realiza una conferencia sobre Symfony organizada en España, este es el evento PHP más grande de ese país y la segunda conferencia más importante del mundo sobre este framework (17).

Dispone de un programa de certificación para validar los conocimientos de sus programadores. A pesar de que participan cientos de programadores de todo el mundo, las decisiones técnicas más importantes siempre las toman un reducido grupo de líderes, lo que garantiza que se mantenga la visión del proyecto y evita la descoordinación (17).

Symfony2 proporciona soluciones prefabricadas para los problemas más comunes, como son la atención de peticiones, formularios e interacción con base de datos. Ofrece una estructura clara y organizada a varios niveles, como la estructura de directorios y la separación por capas, proporciona una buena seguridad para las aplicaciones y cuenta con una vasta documentación tanto en internet, como en la UCI.

JQuery. Versión 2.0

Es una biblioteca rápida, pequeña y rica en funciones de JavaScript. Permite simplificar la manera de interactuar con los documentos HTML. Manipula el árbol DOM (Document Object Model) y maneja eventos, animaciones e interacciones con la técnica AJAX (Asynchronous JavaScript And XML), para el desarrollo web rápido. Crea efectos visuales y modifica los estilos CSS. Tiene una combinación de versatilidad y capacidad de ampliación. Posee un sistema de plugins que permite que sea extensible y utilizable (18). Contiene otras funcionalidades como JSON ya que es el framework JavaScript del lado del cliente.

Bootstrap. Versión 3.0

Es el framework más popular que existe en el desarrollo web para CSS. Es un marco frontal de extremo libre que utiliza lenguajes como CSS3 y HTML5 para el desarrollo rápido, impresionante y fácil de aplicaciones web. Permite crear interfaces que se adapten a los navegadores en diferentes dispositivos a distintas escalas y resoluciones. Se integra perfectamente con las principales librerías Javascript, por ejemplo JQuery (19).

JQuery y Bootstrap permiten realizar el diseño de una página de forma sencilla con la utilización de CCS 3 y HTML 5, lo cual ofrece un diseño sólido al sistema. Las aplicaciones creadas por estos frameworks se adaptan a los diferentes navegadores. Por tanto, se consideran los más apropiados para realizar la capa de presentación.

1.8 Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado, traducido del inglés Integrated Development Environment (IDE) es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Puede dedicarse en exclusiva a un sólo lenguaje, o bien poder utilizarse para varios (20).

PhpStorm. Versión 8.0

PhpStorm es un IDE perfecto para trabajar con Symfony. Entiende profundamente la estructura del código y apoya todas las características del lenguaje PHP para proyectos modernos y antiguos. Proporciona la mejor prevención de errores en marcha y una extraordinaria visión de lo que sucede en la aplicación a cada paso (21).

La utilización de este IDE es muy ventajosa, ya que se utiliza Symfony como framework de desarrollo y este es un proyecto PHP. Además, facilita la finalización de código inteligente, la configuración de formato de código extendido, es compatible con mezclas de idiomas, trata el código con cuidado y contribuye a la configuración global del proyecto con facilidad y seguridad.

1.9 Sistema Gestor de Base de Datos

Un Sistema Gestor de Bases de Datos (SGBD) es un software 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. Garantizan la seguridad e integridad de los datos.

PostgreSQL. Versión 9.4.x

El proyecto PostgreSQL tal y como lo conocemos hoy en día, empezó en 1996, aunque las bases y el trabajo en la que se asienta tienen sus comienzos en la década de los 70. Es un sistema de gestión de bases de datos relacional, con su código fuente disponible libremente y

utiliza un modelo cliente/servidor. Usa multiprocesos para garantizar la estabilidad del sistema, es decir, un fallo en uno de los procesos no afectará el resto. Funciona muy bien con grandes cantidades de datos y alta concurrencia de usuarios accediendo a la vez (22).

PostgreSQL tiene soporte profesional tanto de la comunidad, como de empresas especializadas. Es muy fiable y permite copias de seguridad en caliente. Es el sistema de código abierto más avanzado y potente del mercado y en sus últimas versiones ha demostrado su pertinencia frente a otras bases de datos comerciales (22). La utilización de PostgreSQL como SGBD es muy ventajosa ya que se está en presencia de un software con licencia libre.

1.10 Servidor web

Un servidor web es un software creado para cargar un archivo y servirlo a través de la red al navegador de un usuario.

Apache. Versión 2.4.7

Es el servidor web más utilizado a nivel mundial, de uso gratuito, muy robusto y se destaca por su seguridad y rendimiento. Es un proyecto de código abierto otorgando una transparencia y dando la posibilidad de conocer que es lo que realmente se está instalando. Corre sobre una multitud de plataformas y Sistemas Operativos. Es altamente configurable y de diseño modular, capaz de ampliar su funcionalidad y calidad de servicios. Otra particularidad propia de Apache y que está muy ligada a su pensamiento y filosofía libre, es que al ser tan popular y utilizado, es posible encontrar gran cantidad de documentos y ayuda en internet en todos los idiomas (23). Teniendo en cuenta todo lo antes expuesto, día a día, usuarios y servidores reiteran su confianza y renuevan la elección a este servicio.

Conclusiones parciales:

Como resultado del análisis realizado en este capítulo se logró concretar la base de conocimiento necesaria para dar soporte a la investigación. Con el estudio de la herramienta de autor web CRODA en su versión 3.0, se puede decir que los componentes Tree, Drag and Drop y Web Editor no se pueden reutilizar en otros proyectos de software. Teniendo en cuenta que se utiliza Xalix como marco de trabajo para la implementación de las nuevas funcionalidades, queda seleccionada la metodología de desarrollo de software AUP-UCI en su escenario 4 y se escogen las herramientas y tecnologías que este framework contempla.

CAPÍTULO 2. PROPUESTA DE SOLUCIÓN

Introducción

En el presente capítulo se describe la propuesta de solución que facilitará el cumplimiento del objetivo general definido en la presente investigación. La metodología AUP-UCI define en cada una de sus fases un grupo de artefactos que brindan una comprensión clara del negocio y el sistema a desarrollar, al tener la misma un enfoque ágil, permite que se elabore la documentación necesaria para facilitar la comunicación entre los desarrolladores y las partes interesadas. Se presenta además una descripción del patrón arquitectónico y de los patrones de diseño a utilizar en la solución.

2.1 Descripción de la propuesta de solución

Se propone desarrollar tres nuevos componentes Tree, Drag and Drop y Web Editor con el objetivo de satisfacer funcionalidades que brindan en CRODA, pero con la ventaja de que puedan ser reutilizados en otros proyectos de software. Para su desarrollo se utilizaran las pautas del marco de trabajo Xalix establecidas en el *Capítulo 1* y para visualizarlos, la plantilla base propuesta por dicho marco. EL componente Tree permitirá el manejo de cualquier estructura XML, lo que favorece a su reutilización. El componente Drag and Drop permitirá crear elementos con las políticas de diseño que el usuario desee, lo que proporciona un producto genérico; una vez creados los elementos, se podrán modificar los mismos. El componente Web Editor brindará funcionalidades para facilitar la creación de recursos web.

2.2 Modelo de dominio

Un modelo de dominio es un modelo conceptual en el que se representan los conceptos de las clases del dominio de un problema relacionadas entre sí (24). Proporciona una visión estructurada del negocio. Es un artefacto construido con reglas de UML, presentado como uno o más diagramas de clases. Es utilizado por el analista como un medio para comprender el sector al cual el sistema va a servir.

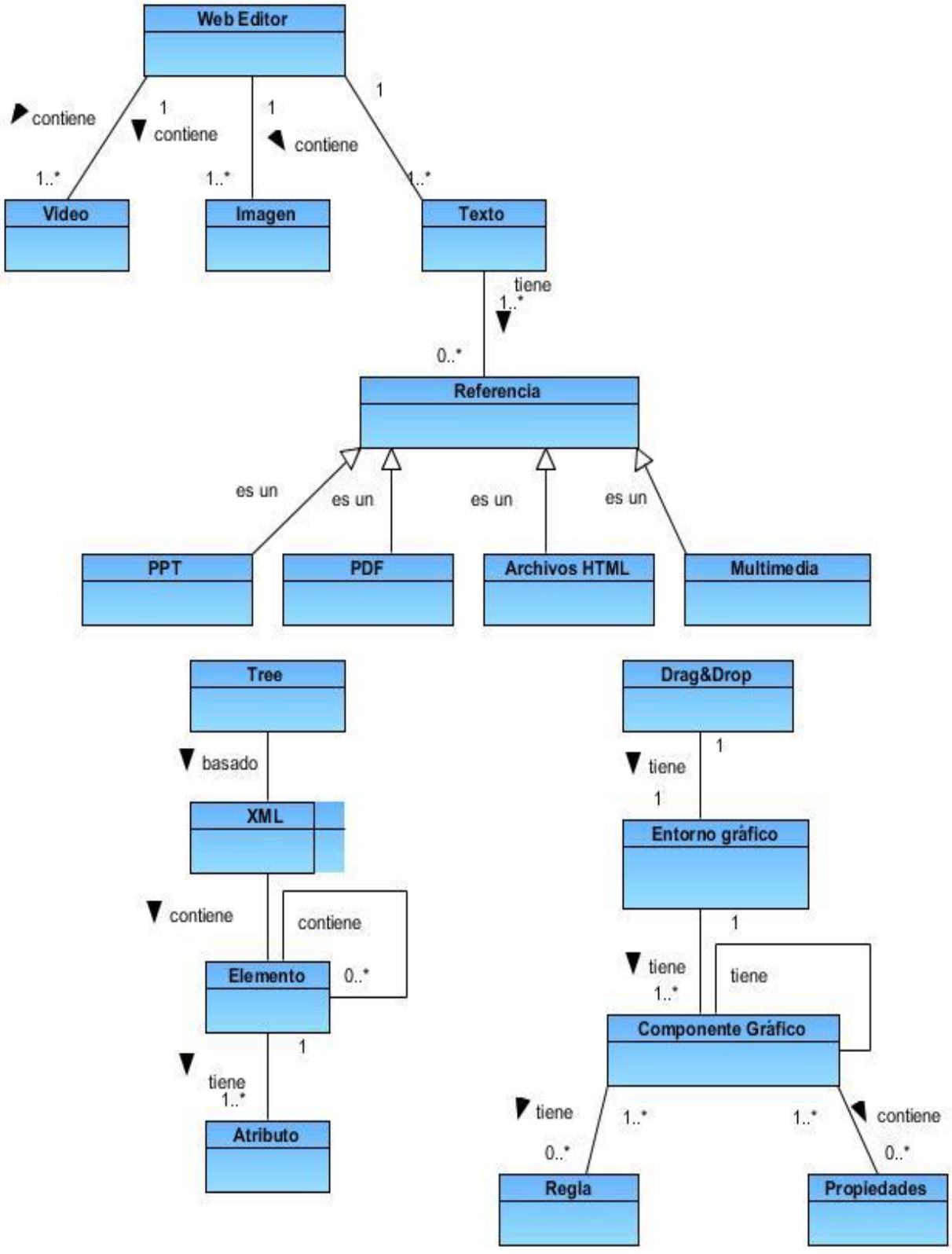


Ilustración 4: Modelo de dominio de la propuesta de solución.

Descripción de los conceptos del Modelo de Dominio:

Tree: componente que proporciona árboles iterativos, algunas funcionalidades que permite son: renombrar elemento, adicionar y eliminar elemento hijo.
XML: lenguaje de marcado en el que se basa el componente Tree para almacenar sus datos de forma legible.
Elemento: contenido del Tree, puede ser padre o hijo.
Atributo: características individuales que diferencian a un elemento de otro.
Drag&Drop: componente utilizado para graficar.
Entorno Grafico: lugar donde se interactúa con el componente Drag&Drop para realizar gráficos.
Componente Gráfico: artefactos que se utilizan para realizar los gráficos.
Regla: norma que debe cumplir cada componente para realizar los gráficos.
Propiedades: representa lo que puede realizar cada componente sobre sí mismo.
Web Editor: componente utilizado para la realización de recursos web.
Video: contenido que presenta el editor web.
Imagen: contenido que presenta el editor web.
Texto: contenido que presenta el editor web.
Referencia: representa el lugar de donde fue obtenida la información del texto.
PPT: forma en que puede estar referenciado un texto.
PDF: forma en que puede estar referenciado un texto.
Archivos HTML: forma en que puede estar referenciado un texto.
Multimedia: forma en que puede estar referenciado un texto.

2.3 Requisitos del sistema

La Ingeniería de Requerimientos cumple un papel primordial en el proceso de producción de software, ya que enfoca un área fundamental, como es la definición de lo que se desea producir. Su principal tarea consiste en la generación de especificaciones correctas que describan con claridad, sin ambigüedades, en forma consistente y compacta, el comportamiento del sistema; de esta manera se pretende minimizar los problemas relacionados al desarrollo del sistema (25).

Un requisito del sistema es una condición, necesidad, estándar o un documento formal de un usuario para resolver un problema o alcanzar un objetivo. Pueden dividirse en requerimientos funcionales y no funcionales.

Requisitos Funcionales

Un requisito funcional (RF) describe el funcionamiento del sistema, los servicios que debe proporcionar con todo detalle y como debe comportarse.

Requisitos funcionales del componente Tree:

- **RF 1** Visualizar un componente Tree.
- **RF 2** Cambiar de posición los elementos.
- **RF 3** Renombrar los elementos del Tree.
- **RF 4** Eliminar elementos del Tree.
- **RF 5** Adicionar elementos hijos.
- **RF 6** Sincronizar los cambios con el XML.

Requisitos funcionales del componente Web Editor:

- **RF 7** Visualizar un editor de recursos web.
- **RF 8** Cargar recurso.
- **RF 9** Insertar un texto.
- **RF 10** Poner el texto en negrita.
- **RF 11** Poner el texto en subrayado.
- **RF 12** Asignar tipología al texto.
- **RF 13** Asignar tamaño al texto.
- **RF 14** Asignar color al texto.
- **RF 15** Asignar color al fondo del texto.
- **RF 16** Alinear el texto a la izquierda.
- **RF 17** Alinear el texto a la derecha.
- **RF 18** Centrar el texto.
- **RF 19** Justificar el texto.
- **RF 20** Mover el texto a la izquierda.
- **RF 21** Mover el texto a la derecha.
- **RF 22** Limpiar formato de la letra.
- **RF 23** Adicionar viñetas al texto.
- **RF 24** Enumerar el texto.
- **RF 25** Insertar vínculos al texto.

- **RF 26** Insertar imagen.
- **RF 27** Insertar video.
- **RF 28** Insertar tabla.
- **RF 29** Limpiar todo el recurso.
- **RF 30** Poner el editor en pantalla completa.
- **RF 31** Acceder al manual de ayuda.
- **RF 32** Salvar el recurso.

Requisitos funcionales del componente Drag and Drop:

- **RF 33** Visualizar un componente que permita graficar.
- **RF 34** Adicionar elementos.
- **RF 35** Editar elementos.
- **RF 36** Redimensionar elementos.
- **RF 37** Mover los elementos.
- **RF 38** Eliminar elementos.
- **RF 39** Renombrar los elementos.

Requisitos no funcionales

Un requisito no funcional son restricciones que afectan los servicios o funciones del sistema (restricciones de tiempo, proceso de desarrollo o estándares), definen propiedades emergentes del sistema, pueden especificar una herramienta en particular, un lenguaje de programación o un método de programación.

Usabilidad

- El sistema debe poseer una interfaz fácil de utilizar para cualquier tipo de usuarios con conocimientos básicos de informática y manejo de ordenadores.
- El sistema debe permitir acceso al menú general desde cualquiera de sus páginas.

Apariencia

- El sistema debe tener una interfaz semejante a la plantilla base establecida para el marco de trabajo Xalix.

Soporte

- El sistema debe ejecutarse sobre cualquier navegador, siendo como mínimo compatible con:
    - Mozilla Firefox en su versión 7.0.
    - Opera en su versión 10.0.0.
    - Chrome en su versión 7.0.

Restricciones de diseño e implementación

- El framework de desarrollo que se utilizará es Symfony en su versión 2.7.x.
- Como lenguaje del lado del servidor se utilizará PHP en su versión 5.4.x.
- Como lenguaje del lado del cliente se empleará HTML en su versión 5.0
- Como IDE se empleará PhpStorm en su versión 8.0.
- Se empleará como SGBD PostgreSQL en su versión 9.4.x.
- Como servidor web se empleará Apache en su versión 2.4.7.

2.4 Historias de usuario

Una historia de usuario (HU) es una breve descripción de un requisito utilizando el lenguaje común del usuario (26). Es una forma rápida de administrar los requisitos sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Son utilizadas en las metodologías de desarrollo ágiles.

La propuesta de solución se modela con una historia de usuario por cada requisito funcional, ya que en el capítulo 1 de la presente investigación se selecciona la metodología de desarrollo de software AUP-UCI en su escenario 4. A continuación se presenta la HU Adicionar elementos. Para consultar el resto de HU remitirse al Anexo 2.

Número: 34	**Nombre del requisito:** Adicionar elementos
Programador: Dalianne Zamora Miclín Cleydi García Díaz	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: Alto	**Tiempo Real:** 2 días
Descripción:	

1- Objetivo:

Permitir adicionar elementos a un componente gráfico.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para adicionar elementos a un componente gráfico hay que:

- Visualizar el componente gráfico.

3- Comportamientos válidos y no válidos (flujo central y alternos):

-El campo nombre es obligatorio.

4- Flujo de la acción a realizar:

-El sistema permitirá insertar un elemento, seleccionando la opción Crear componente.

-Si no existen otros componentes creados, automáticamente se crea como un padre.

-Si existen otros elementos creados, puede seleccionar el padre que desea para el elemento creado.

Observaciones:

Prototipo de interfaz:

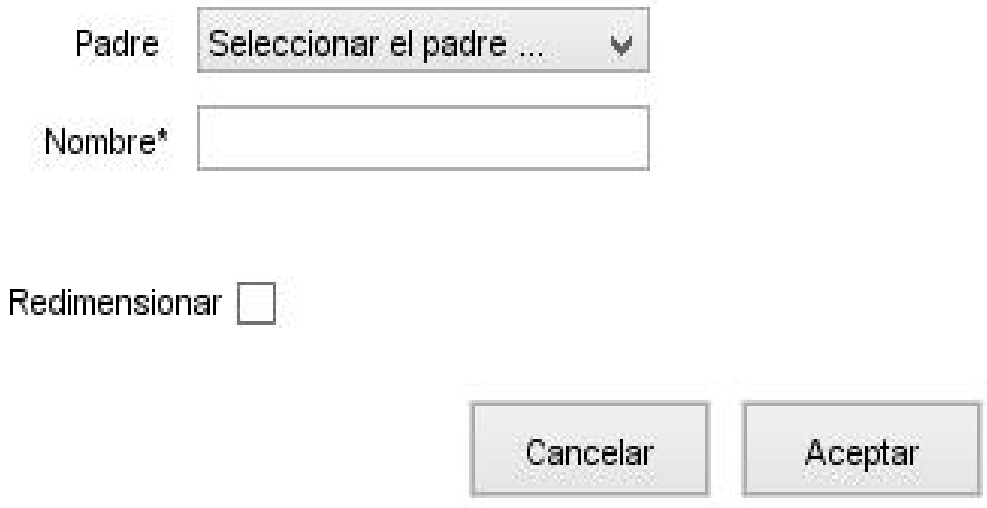


Tabla 5: HU Adicionar elementos.

2.5 Modelo del análisis

El modelo de análisis es la primera representación técnica de un sistema. Utiliza una mezcla de formatos en texto y diagramas para representar los requisitos del software, las funciones y el

comportamiento. De esta manera se hace mucho más fácil de comprender dicha representación, ya que es posible examinar los requisitos desde diferentes puntos de vista, aumentando la probabilidad de encontrar errores, de que surjan debilidades y de que se descubran descuidos.

Diagramas de clases del análisis

Un diagrama de clases del análisis representa la relación entre los actores y el sistema, este último se simboliza mediante clases de tipo interfaz, controladora y entidad; que pueden ser una abstracción del diseño del sistema. Este diagrama se representa mediante tres estereotipos que a continuación se presentan.

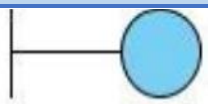
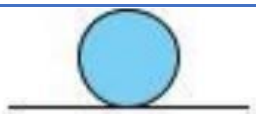
Estereotipos	Características
	**Clase Interfaz:** modela la interrelación entre el sistema y sus actores.
	**Clase Controladora:** modela los aspectos dinámicos del sistema, coordinando las acciones y flujos de control principales de los objetos que implementan las funcionalidades.
	**Clase Entidad:** modela información que posee larga vida y que es persistente en el sistema.

Tabla 6: Estereotipos del modelo del análisis.

A continuación se presenta el diagrama de clases del análisis del RF Adicionar elementos. Para consultar el resto de los diagramas de clases del análisis remitirse al Anexo 3.

Ilustración 5: Diagrama de clases del análisis del RF Adicionar elementos.

2.6 Modelo del diseño

El diseño es una representación significativa de la ingeniería de algo que se va a construir. Se puede hacer seguimiento basándose en los requisitos del cliente y al mismo tiempo la calidad se puede evaluar con el conjunto de criterios predefinidos para obtener un buen diseño. Deberá implementar todos los requisitos explícitos del modelo de análisis y ajustarse a todos los requisitos implícitos que desea el cliente (27).

Es una guía legible para aquellos que generan código y para aquellos que comprueban y consecuentemente dan soporte al software. Un diseño deberá presentar una estructura arquitectónica que se haya creado mediante patrones de diseño y contiene distintas representaciones de datos, arquitectura, interfaces y componentes.

En el diseño de esta investigación se realizaron diagramas de clases del diseño y diagramas de secuencia. A continuación se muestran los estereotipos utilizados para la elaboración de los mismos.

Estereotipos	Características
	**<<Client Page>>:** es una página web con formato HTML que le brinda al usuario toda la interfaz de la aplicación.
	**<<Form>>:** es una colección de elementos de entrada que están contenidos en la página cliente. Sus atributos son los elementos de entrada del formulario.
	**<<Server Page>>:** representa la página web que tiene código que se ejecuta en el servidor.
	**<<Entity>>:** representa la información que posee larga vida y que es persistente en el sistema.

Tabla 7: Estereotipos del modelo del diseño.

Diagramas de clases del diseño

Un diagrama de clases del diseño muestra las asociaciones entre las clases y las definiciones de las entidades de software, en lugar de los conceptos del mundo real. Cada clase muestra los métodos y atributos que la componen. Además, contiene las interfaces con sus operaciones. En este diagrama se puede observar qué clases conocen a otras clases o son partes de ellas, sin mostrar los métodos que las invocan (28). A continuación se muestran los diagramas de clases del diseño de los componentes Tree, Drag and Drop, y Web Editor:

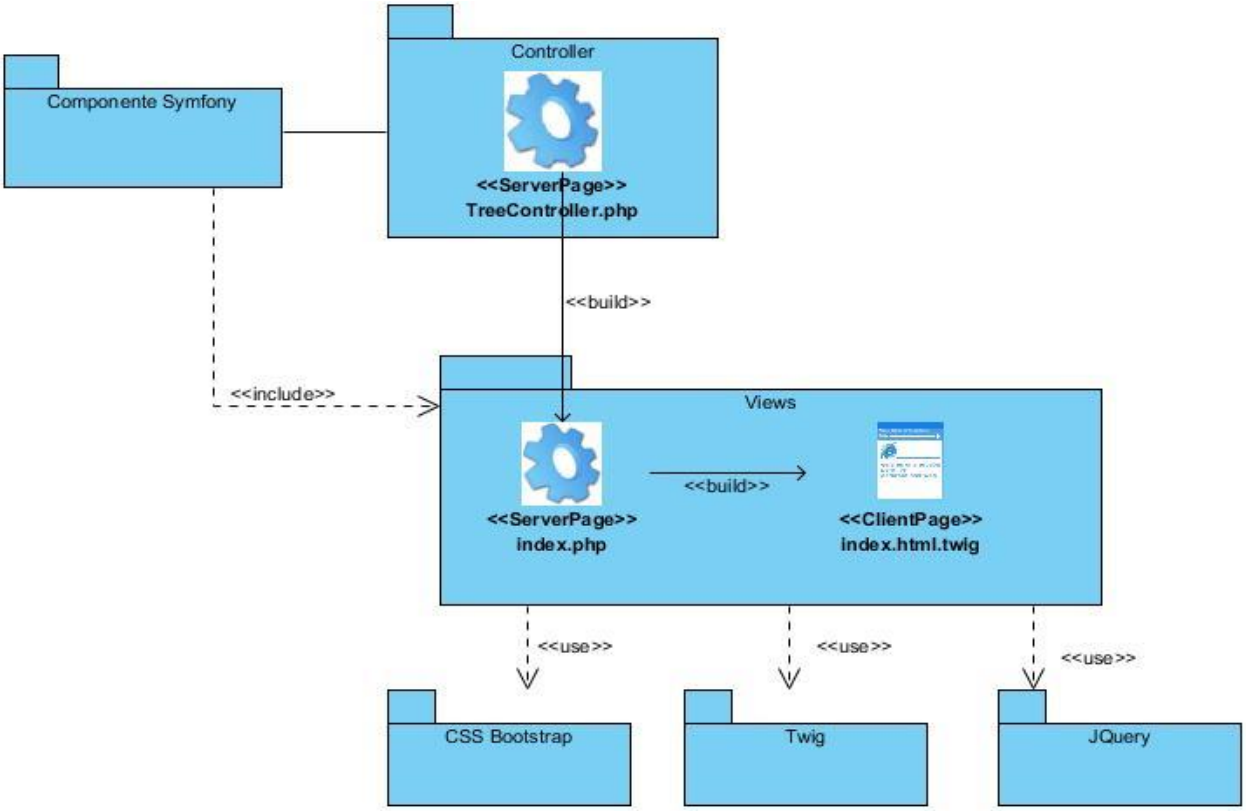


Ilustración 6: Diagrama de clases del diseño del componente Tree.

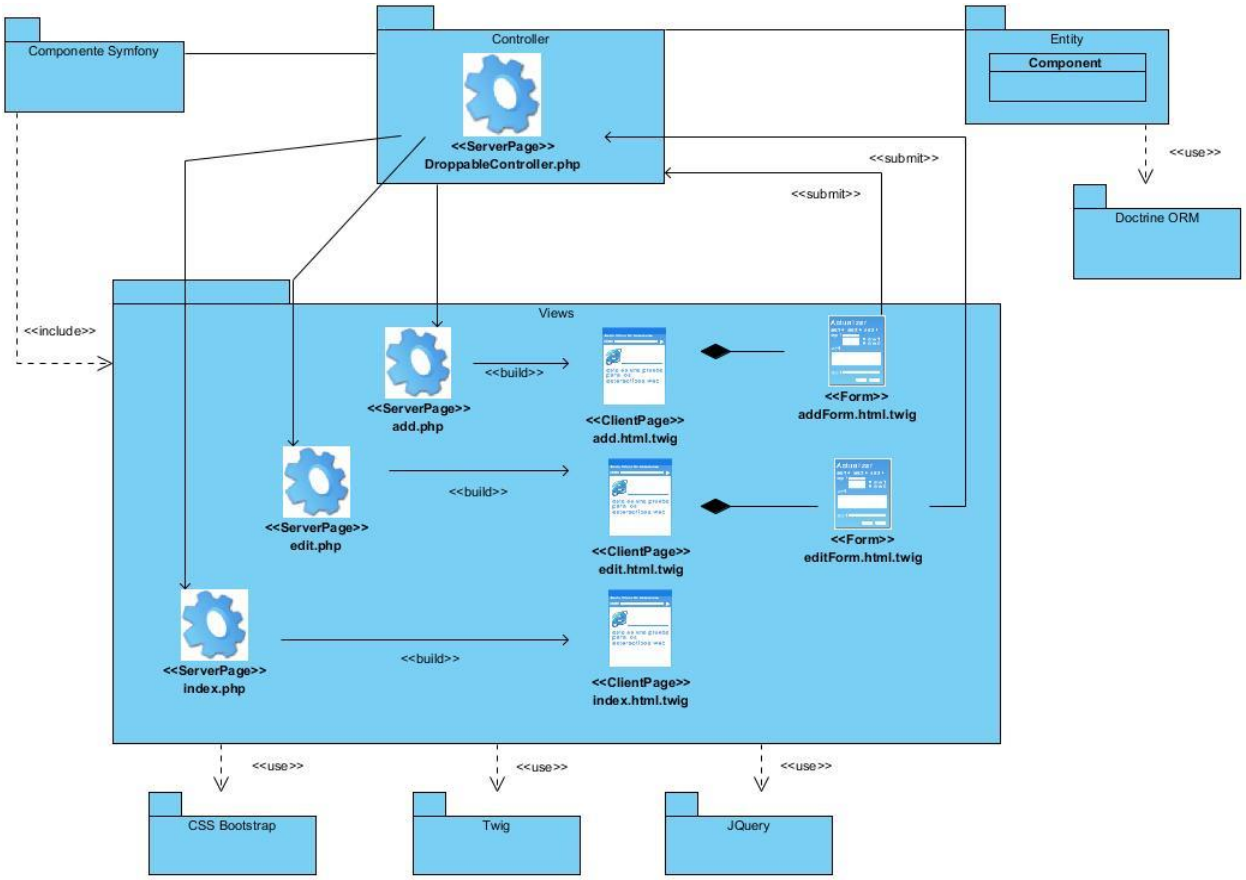


Ilustración 7: Diagrama de clases del diseño del componente Drag and Drop.

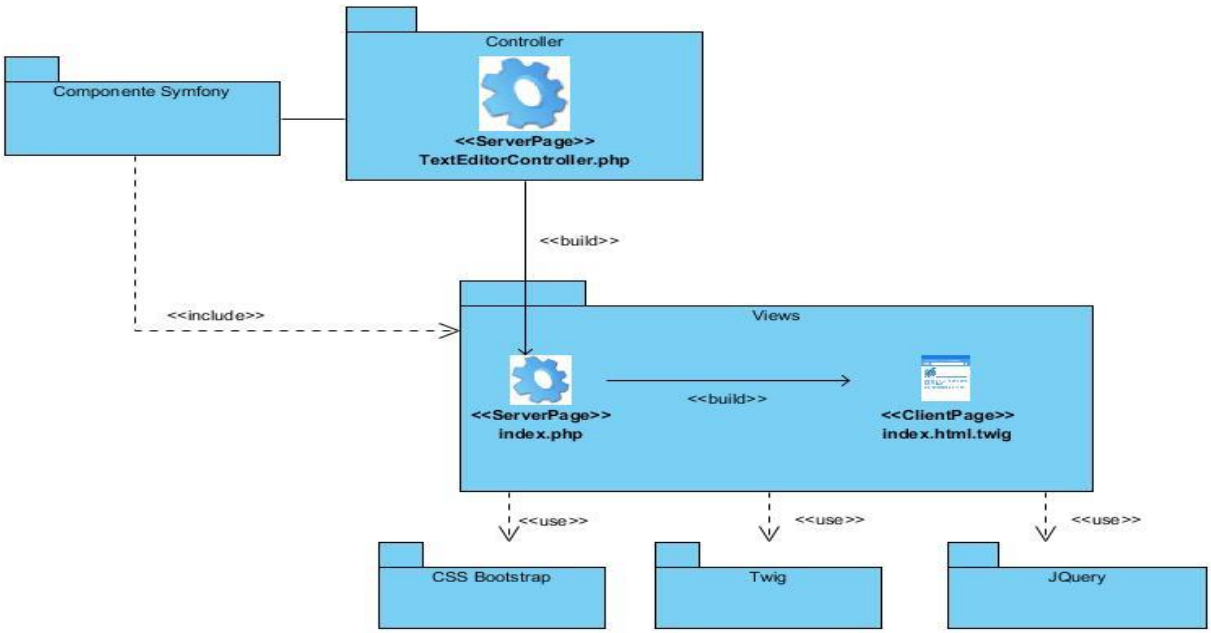


Ilustración 8: Diagrama de clases del diseño del componente Web Editor.

Diagramas de secuencia del diseño

Un diagrama de secuencia del diseño es usado para modelar la interacción entre los objetos de un sistema a través del tiempo (29). Incluye las clases que se usan para implementar el escenario y los mensajes intercambiados entre los objetos. Muestra los objetos que intervienen en el escenario con líneas discontinuas verticales y los mensajes pasados entre los objetos como flechas horizontales.

A continuación se presenta el diagrama de secuencia del diseño del RF Adicionar elementos. Para consultar el resto de los diagramas de secuencia del diseño remitirse al Anexo 4.

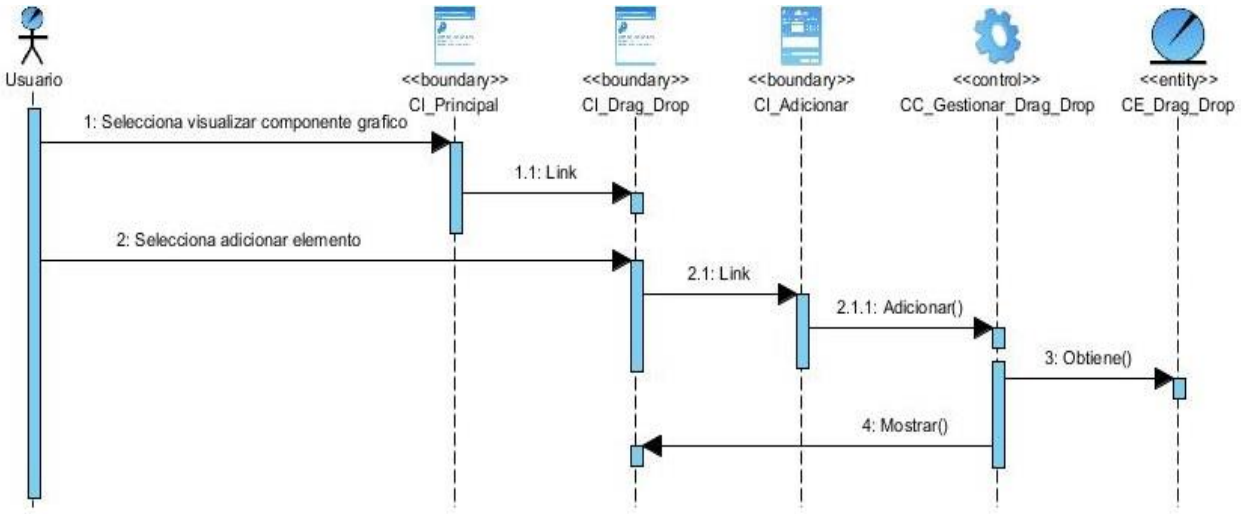


Ilustración 9: Diagrama de secuencia del RF Adicionar elementos.

2.7 Modelo de datos

Un modelo de datos relacional permite establecer interconexiones o relaciones entre los datos que están guardados en tablas. Este modelo de datos se utiliza para describir la estructura lógica y física de la información persistente que puede ser gestionada por el sistema. Es un conjunto de concepto que permiten describir a distintos niveles de abstracción la estructura de una base de datos (30). Este modelo de datos se utiliza en el componente Drag and Drop, para manejar la persistencia de sus datos.

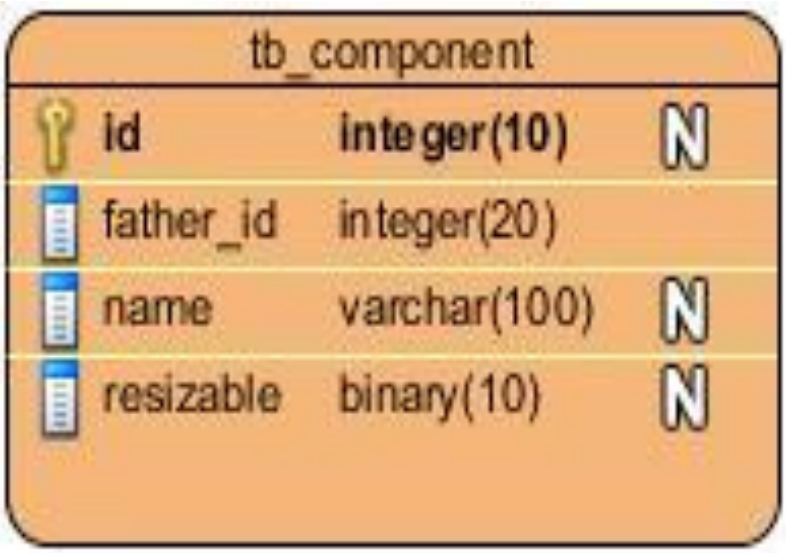


Ilustración 10: Modelo físico de la base de datos.

Nombre de la tabla: tb_component		
Descripción de la tabla*:* tabla donde se almacena toda la información de los DA creados por el componente Drag and Drop**.**		
Atributo	**Tipo**	**Descripción**
id	integer	Campo que contiene el identificador del objeto.
father_id	integer	Campo que genera la relación del objeto con el mismo.
name	varchar	Campo que contiene el nombre del objeto.
resizable	binary	Campo que identifica si un objeto se puede redimensionar o no.

Tabla 6: Descripción de la tabla tb_component de la base de datos.

Para mostrar la información de los recursos web creados por el componente Web Editor se utilizan documentos HTML que permiten definir de forma estricta la apariencia de la página.

Un documento HTML comienza con la etiqueta <html> y termina con </html>. Dentro del documento hay dos zonas principales, el encabezamiento y el cuerpo. El encabezamiento delimitado por las marcas <head> y </head>, que sirve para definir algunos valores válidos y el cuerpo delimitado por las etiquetas <body> y </body>, donde reside la información del documento. El elemento <title> contenido dentro del encabezamiento permite especificar el título de un documento HTML. Este título no forma parte del documento en sí, ya que no se ve en la pantalla principal, pero sirve como título de la ventana del programa que la muestra. Existen otros elementos que se engloban dentro del encabezamiento, pero para la estructura del lenguaje HTML en su nivel básico no son necesarios *(31)*. A continuación se muestra la estructura básica de un documento HTML:

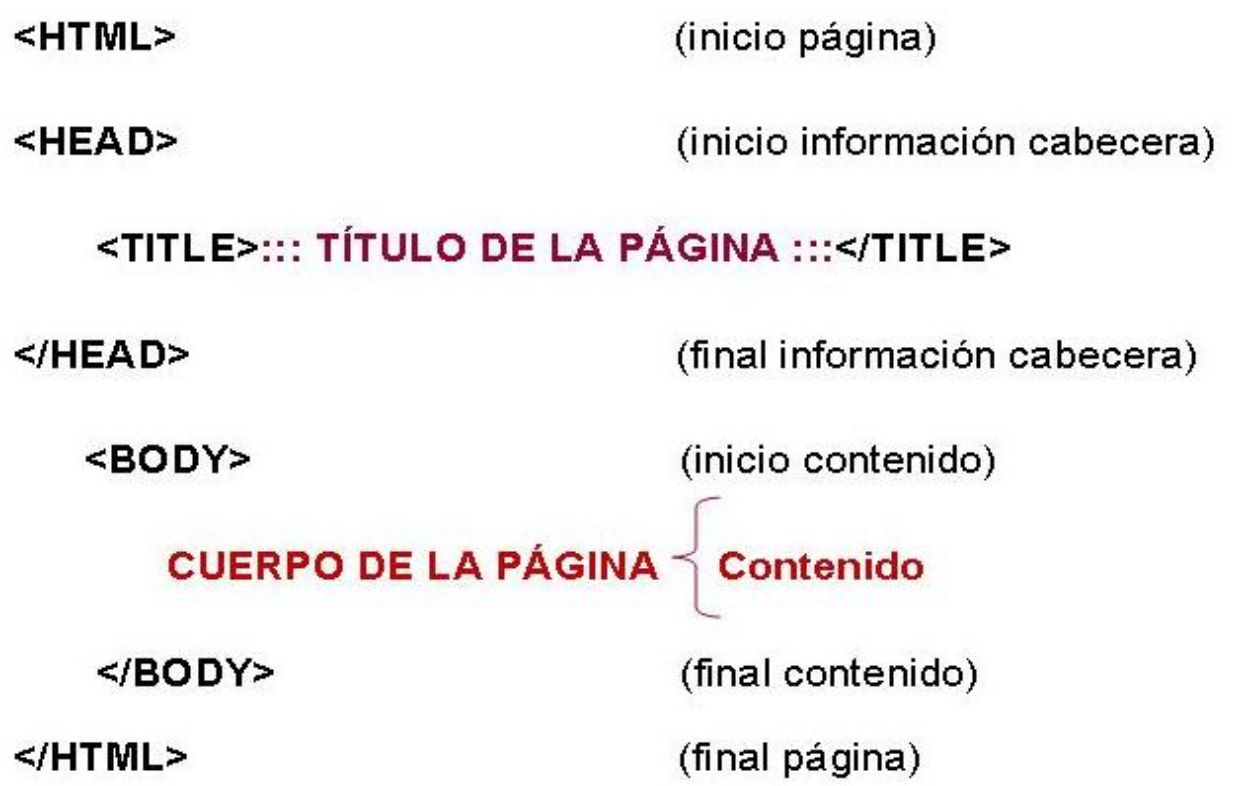


Ilustración 11: Estructura de un HTML.

El componente Tree manipula estructuras XML para su correcto funcionamiento, esta tecnología permite jerarquizar y estructurar la información de manera sencilla gracias al uso de etiquetas.

La diferencia fundamental entre HTML y XML es que el primero estaba orientado a la presentación de datos, mientras que el XML está orientado a los datos en sí mismos. Sin duda, esta diferencia es fundamental para los nuevos desarrollos de la web, donde es de suma importancia al contenido de los datos y su tratamiento y no sólo a su presentación.

2.8 Patrones de diseño

Un patrón de diseño brinda una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Permite entender cómo adaptarlo a la variante particular del problema donde se quiere aplicar y facilita la reutilización de diseños que han tenido éxito. Para anticiparse a los cambios en los requisitos hay que diseñarlo pensando en qué aspectos pueden cambiar (32).

Los patrones GRASP acrónimo de “General Responsibility Assignment Software Patterns”, tienen como objetivo fundamental orientar al diseñador de como asignar las responsabilidades a cada clase en diferentes circunstancias (32). Para el desarrollo de la propuesta de solución se tienen en cuenta los siguientes:

Experto: es el patrón básico de asignación de responsabilidades, se basa en asignar la responsabilidad a una clase que se encargará de ser experto de la información, la misma cuenta con la información necesaria para cumplir con la responsabilidad. Al aplicarlo permitirá identificar las clases que poseen la información requerida, para que luego estas puedan ser manipuladas por otras clases (32). El uso del mismo se puede observar en la clase entidad *Component*, la cual es experta en la información.

Creador: se le aplica la responsabilidad a una clase determinada de crear una o más instancias de otra (32). En Symfony2 las clases controladoras definen y ejecutan todas las acciones y en las acciones se crean instancias de las clases que representan las entidades. Ejemplo de este patrón se evidencia en la clase *DroppableController*, la cual crea instancias de tipo *Component*.

Controlador: sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación para aumentar la reutilización de código y a la vez tener un mayor control (32). Este patrón se evidencia en todas las clases controladoras.

Los patrones Gof (Pandilla de los Cuatro) acrónimo de “Gang of Four”, constituyen un catálogo de 23 patrones de diseño los cuales se clasifican en dependencia del propósito para los que hayan sido utilizados (33). En esta propuesta se tiene en cuenta el siguiente:

Factory Method: en español método de fabricación, centraliza en una clase constructora la creación de objetos de un subtipo de un tipo determinado, ocultando al usuario la diversidad de casos particulares que se pueden prever, para elegir el subtipo que crear (33). Un ejemplo de la utilización de este patrón se pone en práctica en la clase de entidad *Component* que es la encargada de crear un objeto de un tipo determinado.

2.9 Patrón arquitectónico

El patrón arquitectónico Modelo Vista Controlador (MVC) es una propuesta de diseño de software utilizada para implementar sistemas donde se requiere el uso de interfaces de usuario. Surge de la necesidad de crear un software más robusto, con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos. Su fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad (34). Este patrón en los últimos años ha ganado mucha más fuerza y seguidores, gracias a la aparición de numerosos framework de desarrollo web que lo utilizan como referencia para la arquitectura de sus aplicaciones.

El marco de trabajo Symfony2 basa su funcionamiento interno en la arquitectura MVC. Cuando el usuario solicita ver la portada del sitio, internamente sucede lo siguiente (35):

1. El sistema de enrutamiento determina qué controlador está asociado con la página de la portada.
2. Symfony2 ejecuta el controlador asociado a la portada. Un controlador es una clase PHP que contiene la lógica de negocio.
3. El controlador solicita al modelo los datos necesarios. El modelo no es más que una clase PHP especializada en obtener información, normalmente de una base de datos.

4. Con los datos devueltos por el modelo, el controlador solicita a la vista que cree una página mediante una plantilla y que inserte los datos del modelo. Una vista es la interfaz que utiliza el usuario para interactuar con la aplicación. En Symfony 2 la capa de la vista está formada principalmente por plantillas en PHP con código HTML.
5. El controlador entrega al servidor la página creada por la vista.

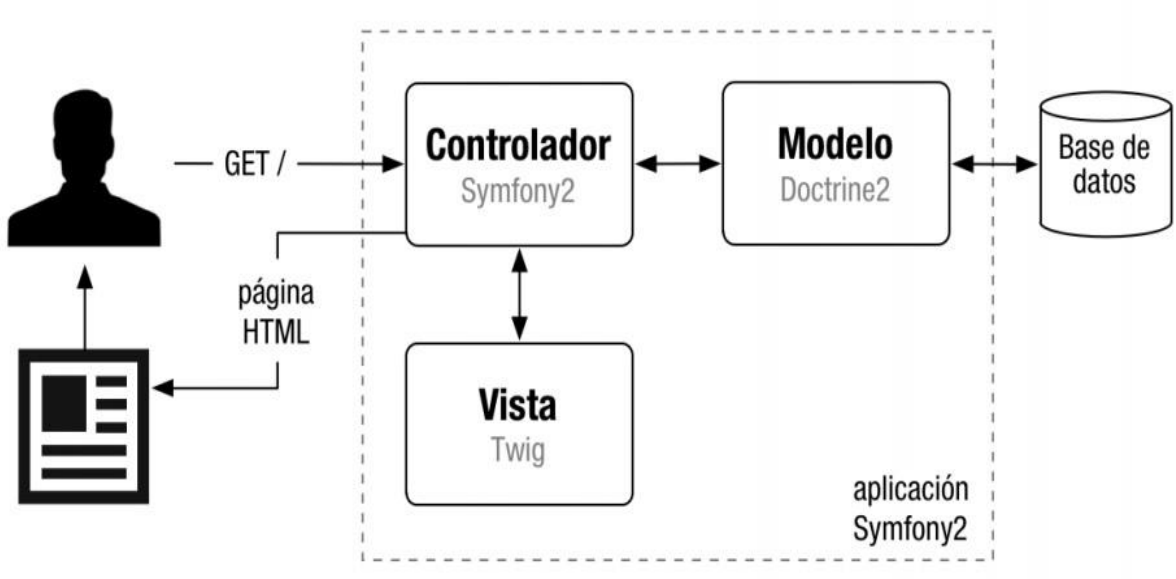


Ilustración 12: Arquitectura de Symfony2.

Básicamente el funcionamiento interno de la arquitectura MVC siempre es el mismo, el controlador manda y ordena, el modelo busca la información que se le pide y la vista crea páginas con plantillas y datos. Symfony2 toma lo mejor de esta arquitectura y lo realiza de modo que el desarrollo de aplicaciones sea rápido y sencillo. En el controlador se encuentran todas las acciones, las cuales son el núcleo de la aplicación, pues contienen toda la lógica. Estas acciones utilizan el modelo y precisan las variables para la vista. Al realizarse una petición web en una aplicación Symfony2, la URL define una acción y los parámetros de la petición.

Conclusiones parciales

El modelo de dominio ofreció una visión estructurada del negocio y facilitó la identificación de los requerimientos. La definición de los requisitos funcionales y no funcionales del sistema establecieron las capacidades que debe cumplir la propuesta de solución, así como los diferentes servicios que brindará. Se generaron los artefactos correspondientes para facilitar el

entendimiento entre los desarrolladores y las partes interesadas. La arquitectura y los patrones de diseño definidos permitieron detectar problemas comunes y facilitan la organización del futuro desarrollo.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBA

Introducción

En el presente capítulo se tratan los elementos relacionados con los flujos de trabajo implementación y prueba. Para definir y organizar el código de la propuesta de solución se realiza una descripción del modelo de implementación. Además, se definen los estándares de codificación a tener en cuenta en la solución. Para garantizar el correcto funcionamiento de los requerimientos y así elevar la calidad del producto desarrollado, se explican las pruebas y los casos de prueba que se realizaron.

3.1 Modelo de implementación

El modelo de implementación describe como los elementos del diseño se implementan en términos de componentes. Representa cómo se organizan los componentes de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y de acuerdo con los lenguajes de programación utilizados. Describe la dependencia que tienen unos componentes con otros (36). Su objetivo fundamental es desarrollar la arquitectura y el sistema como un todo.

Diagramas de componentes

Un diagrama de componentes muestra la relación entre los componentes de un sistema, sus dependencias, su comunicación y ubicación (37). Provee una vista arquitectónica de alto nivel del software y ayuda a los desarrolladores a visualizar el camino de la implementación. A continuación se presentan presenta los diagramas de componentes de la solución propuesta:

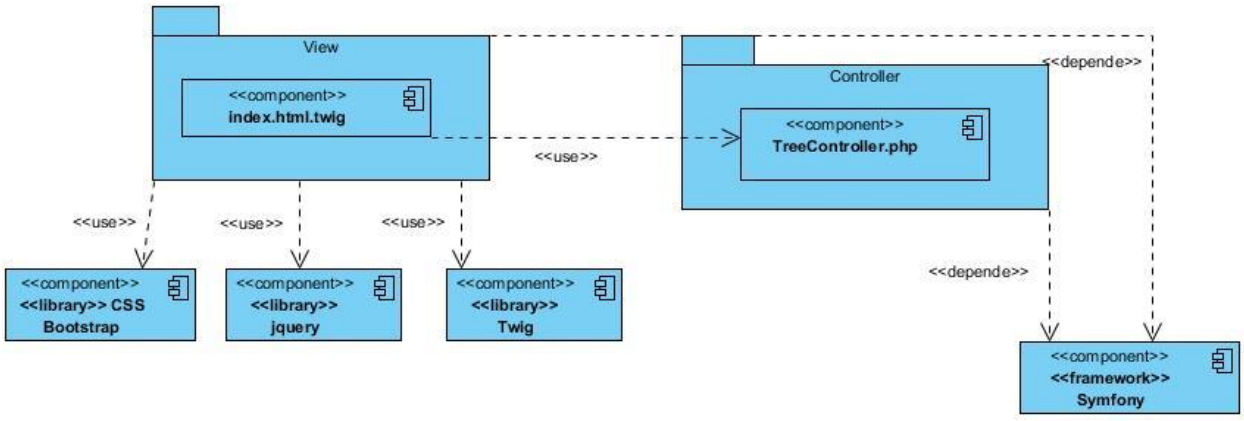


Ilustración 13: Diagrama de componentes del Tree.

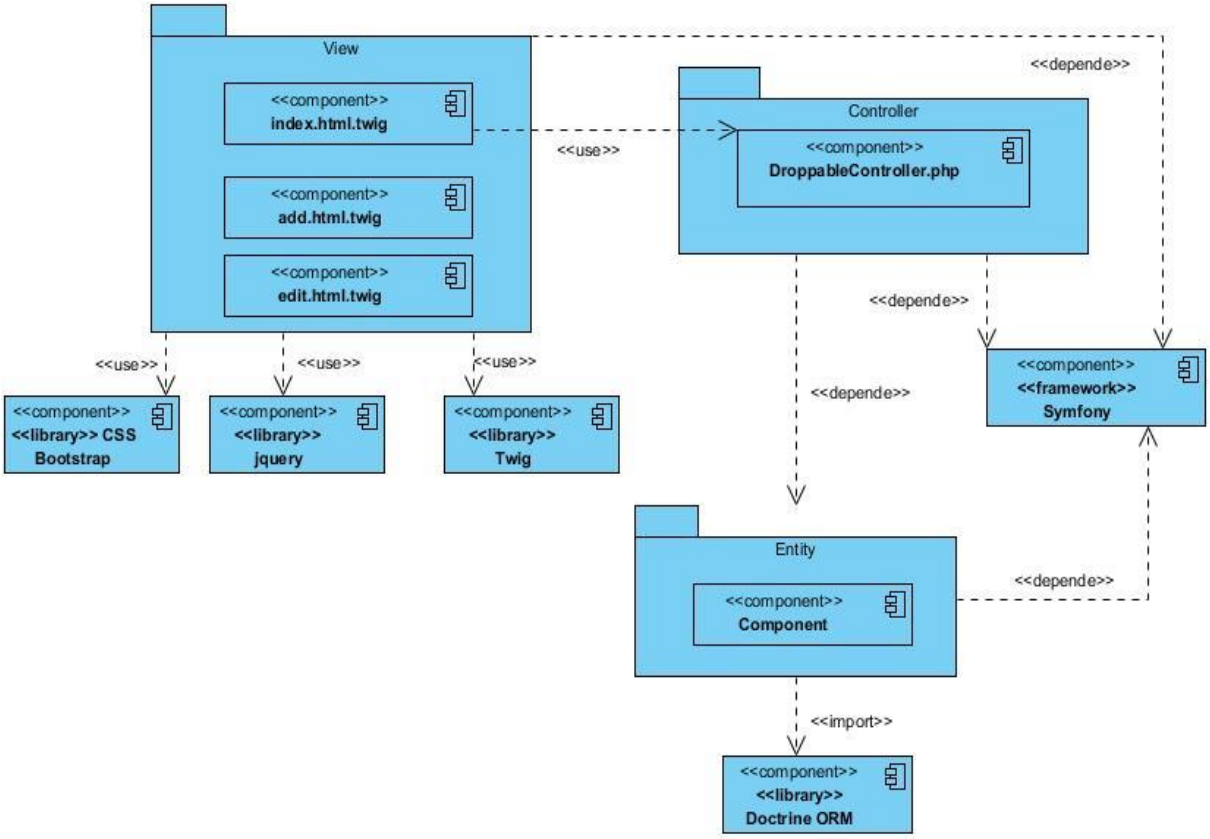


Ilustración 14: Diagrama de componentes del Drag and Drop.

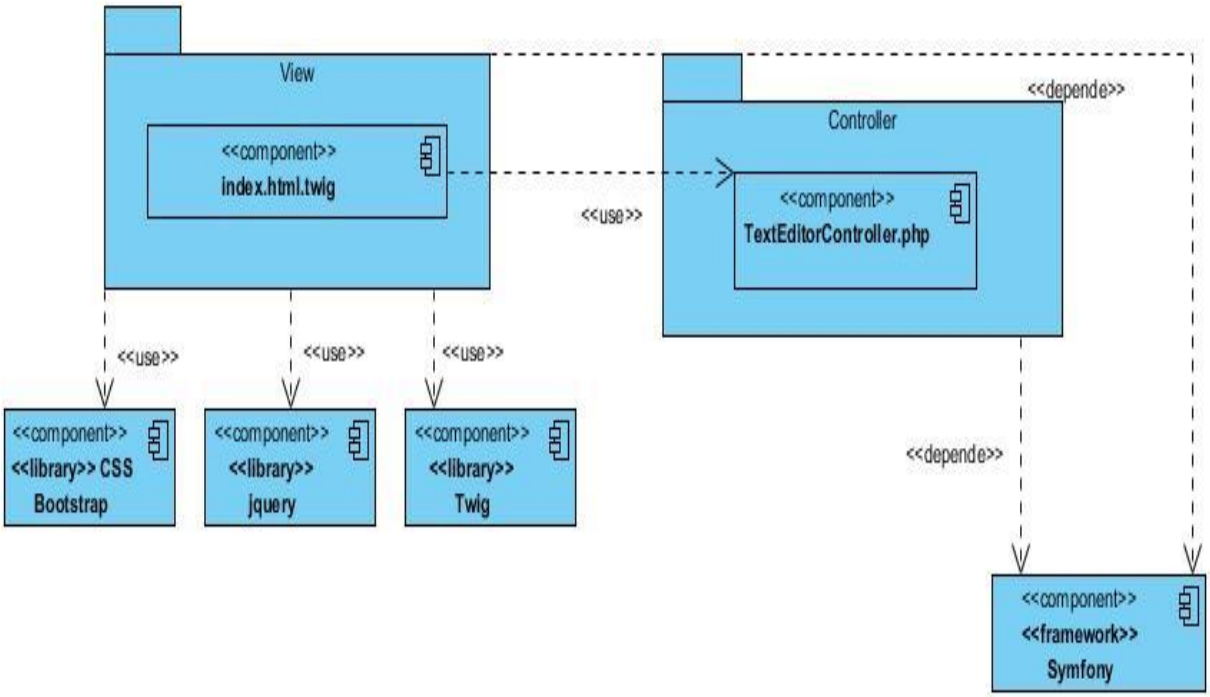


Ilustración 15: Diagrama de componentes del Web Editor.

3.2 Estándares de codificación

El propósito fundamental de los estándares de codificación es que el sistema en cuestión tenga una arquitectura y un estilo consistente, para que resulte fácil su entendimiento y mantenimiento. Para que un programador comprenda bien un sistema de software, influye directamente la organización y limpieza del código fuente (38).

Con el objetivo de ajustarse a los estándares utilizados en Symfony2 como framework seleccionado para el desarrollo del marco de trabajo Xalix, se siguen los estándares siguientes:

- Se añade un solo espacio después de cada delimitador coma.
- Se añade un solo espacio alrededor de los operadores (==*, &&*).
- Se añade una coma después de cada elemento del arreglo en un arreglo multilínea, incluso después del último.
- Se añade una línea en blanco antes de las declaraciones *return*, a menos que el valor devuelto solo sea dentro de un grupo de declaraciones (tal como una declaración *if*).
- Se usa llaves para indicar la estructura del cuerpo de control, independientemente del número de declaraciones que contenga.
- Se define una clase por archivo.
- Se declara las propiedades de clase antes que los métodos.
- Se declara los métodos públicos (*public*) primero, luego los protegidos (*protected*), y finalmente los privados (*private*).
- Se utiliza paréntesis cuando se instancien clases, independientemente del número de argumentos que tenga el constructor.
- Se utiliza mayúsculas intercaladas (sin guiones bajos) en nombres de variable, función, método o argumentos.
- Se usa guiones bajos para nombres de opción y nombres de parámetro.
- Utiliza espacios de nombres para todas las clases.
- Se prefija las clases abstractas con *Abstract*. En este caso hay que tener en cuenta que algunas de las primeras clases de Symfony2 no siguen esta convención y no se han rebautizado por razones de compatibilidad hacia atrás. No obstante, todas las nuevas clases abstractas tienen que seguir esta convención de nomenclatura.
- Se sufija las interfaces con *Interface*.
- Se sufija las características con *Trait*.
- Se sufija las excepciones con *Exception*.

- Se utiliza caracteres alfanuméricos y guiones bajos para los nombres de archivo.

El mantenimiento del código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores o mejorar el rendimiento. Establecer un estándar de codificación asegura que un equipo de programadores mantenga un código de calidad.

3.3 Prueba del software

La prueba de software es un elemento de un tema más amplio que usualmente se conoce como verificación y validación. La verificación se refiere al conjunto de tareas que garantizan que el software implementa correctamente una función específica. La validación es un conjunto diferente de tareas que aseguran que el software sigue los requerimientos del cliente (39). En conjunto representan una revisión de las especificaciones, del diseño y de la codificación.

Para validar la solución propuesta se escoge el nivel de prueba de sistema donde el software se prueba como un todo y se realizan pruebas funcionales, que no son más que pruebas de software que tiene como objetivo probar que los sistemas desarrollados cumplan con las funciones específicas para los cuales han sido creados. Para realizarlas se emplea el método de caja negra.

Pruebas de caja negra

Son aquellas que se realizan sobre la interfaz del software, donde los probadores no enfocan su atención en la estructura interna del programa. Pretenden demostrar que las funciones del sistema son operativas y al mismo tiempo se buscan errores en cada una, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene (39). Tienen como objetivo final garantizar que los requerimientos hayan sido cumplidos y que el producto sea aceptable.

Técnica de partición equivalente

Para aplicar el método de caja negra se utilizó la técnica de partición equivalente, la cual divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software. Descubre de forma inmediata un grupo de errores que de otro modo requerirían la ejecución de muchos casos antes de detectar el error genérico y permite examinar los valores validos e inválidos de las entradas existentes (40).

Diseño de Casos de Prueba

Al aplicar las técnicas de caja negra se procede a la elaboración de un conjunto de casos de prueba (CP) basados en historias de usuario. Un caso de prueba es un grupo de condiciones bajo las cuales se determinará si una característica del sistema es parcial o completamente satisfactoria. A continuación se presenta el CP propuesto para la HU Adicionar elementos. Para consultar el resto de los CP remitirse al Anexo 5.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 34.1 Adicionar elemento.	Seleccionar la opción Crear componente.	Muestra un formulario con los siguientes datos: -Padre -Nombre (es obligatorio) -Redimensionar -Aceptar -Cancelar	Drag&Drop/Crear componente
EC 34.2 Aceptar	Selecciona la opción Aceptar.	Muestra el componente insertado.	Drag&Drop/Crear componente/Aceptar
EC 34.3 Cancelar	Selecciona la opción Cancelar.	Regresa a la vista principal del Drag&Drop.	Drag&Drop/Crear componente/Cancelar

Tabla 7: CP de la HU Adicionar elementos.

Resultados de las pruebas funcionales

Con el fin de detectar la mayor cantidad de no conformidades posibles en las funcionalidades se realizaron tres iteraciones de prueba. A continuación se muestran las tablas que contienen las no conformidades identificadas por cada requisito funcional (RF) en cada iteración. Estas no conformidades se clasifican en Significativa y No significativa y su estado puede ser Resuelta o Pendiente.

Iteración 1				
	Requisito funcional	**Descripción**	**Clasificación**	**Estado**

1	Visualizar un componente Tree	No carga ningún XML.	Significativa	Resuelta
2		No carga todos los XML.	Significativa	Pendiente
3	Sincronizar los cambios con el XML	No guarda ningún cambio.	Significativa	Resuelta
4		No guarda todos los cambios.	Significativa	Pendiente
5	Insertar vínculos al texto	Cuando le das clic al vínculo no hace referencia a la dirección definida.	Significativa	Resuelta
6	Insertar imagen	Carga la imagen, pero no la visualiza.	Significativa	Resuelta
7	Insertar video	No carga ningún video.	Significativa	Resuelta
8	Adicionar elementos	Cuando se crea el elemento se le asigna su padre y al adicionarlo no adopta esta política de diseño.	Significativa	Resuelta
9	Editar elementos	No actualiza ningún cambio al editarlo.	Significativa	Resuelta
10		No actualiza el nuevo padre.	Significativa	Pendiente
11	Redimensionar elementos	No deja redimensionar los elementos al adicionarlos.	Significativa	Resuelta
12		No deja redimensionar los elementos al editarlos.	Significativa	Resuelta

Tabla 8: No conformidades detectadas en la Iteración 1.

Iteración 2				
	Requisito funcional	**Descripción**	**Clasificación**	**Estado**
1	Poner el editor en pantalla completa	No funciona.	Significativa	Pendiente
2	Salvar el recurso	No guarda los cambios de imagen.	Significativa	Resuelta
3		No guarda los cambios de video.	Significativa	Resuelta
4		No guarda los cambios de link.	Significativa	Resuelta
5	Adicionar elementos	El campo del nombre lo puedo dejar vacío.	No significativa	Pendiente
6	Editar elementos	El campo del nombre lo puedo dejar vacío.	No significativa	Pendiente
Pendientes de la Iteración Anterior				
	Visualizar un componente Tree	No carga todos los XML.	Significativa	Resuelta
	Sincronizar los cambios con el XML	No guarda todos los cambios.	Significativa	Resuelta
	Editar elementos	No actualiza el nuevo padre.	Significativa	Resuelta

Tabla 9: No conformidades detectadas en la Iteración 2.

Iteración 3				
	Requisito funcional	**Descripción**	**Clasificación**	**Estado**
1	Insertar video	No se reproduce el video.	Significativa	Resuelta
Pendientes de la Iteración Anterior				

	Poner el editor en pantalla completa	No funciona.	Significativa	Resuelta
	Adicionar elementos	El campo del nombre lo puedo dejar vacío.	No significativa	Resuelta
	Editar elementos	El campo del nombre lo puedo dejar vacío.	No significativa	Resuelta

Tabla 10: No conformidades detectadas en la Iteración 3.

Para obtener la cantidad total de no conformidades en cada iteración y su clasificación se realiza la siguiente tabla:

Requisitos funcionales	**Iteración 1**			**Iteración 2**			**Iteración 3**		
	NC	**S**	**NS**	**NC**	**S**	**NS**	**NC**	**S**	**NS**
Visualizar un componente Tree	2	2	-	-	-	-	-	-	-
Cambiar de posición los elementos	-	-	-	-	-	-	-	-	-
Renombrar los elementos del Tree	-	-	-	-	-	-	-	-	-
Eliminar elementos del Tree	-	-	-	-	-	-	-	-	-
Adicionar elementos hijos	-	-	-	-	-	-	-	-	-
Sincronizar los cambios con el XML	2	2	-	-	-	-	-	-	-
Visualizar un editor de recursos web	-	-	-	-	-	-	-	-	-
Cargar recurso	-	-	-	-	-	-	-	-	-
Insertar un texto	-	-	-	-	-	-	-	-	-
Poner el texto en negrita	-	-	-	-	-	-	-	-	-
Poner el texto en subrayado	-	-	-	-	-	-	-	-	-
Asignar tipología al texto	-	-	-	-	-	-	-	-	-
Asignar tamaño al texto	-	-	-	-	-	-	-	-	-
Asignar color al texto	-	-	-	-	-	-	-	-	-
Asignar color al fondo del texto	-	-	-	-	-	-	-	-	-
Alinear el texto a la izquierda	-	-	-	-	-	-	-	-	-
Alinear el texto a la derecha	-	-	-	-	-	-	-	-	-
Centrar el texto	-	-	-	-	-	-	-	-	-

Justificar el texto	-	-	-	-	-	-	-	-	-
Mover el texto a la izquierda	-	-	-	-	-	-	-	-	-
Mover el texto a la derecha	-	-	-	-	-	-	-	-	-
Limpiar formato de la letra	-	-	-	-	-	-	-	-	-
Adicionar viñetas al texto	-	-	-	-	-	-	-	-	-
Enumerar el texto	-	-	-	-	-	-	-	-	-
Insertar vínculos al texto	1	1	-	-	-	-	-	-	-
Insertar imagen	1	1	-	-	-	-	-	-	-
Insertar video	1	1	-	-	-	-	1	1	-
Insertar tabla	-	-	-	-	-	-	-	-	-
Limpiar todo el recurso	-	-	-	-	-	-	-	-	-
Poner el editor en pantalla completa	-	-	-	1	1	-	-	-	-
Acceder al manual de ayuda	-	-	-	-	-	-	-	-	-
Salvar el recurso	-	-	-	3	3	-	-	-	-
Visualizar un componente grafico	-	-	-	-	-	-	-	-	-
Adicionar elementos	1	1	-	1	-	1	-	-	-
Editar elementos	2	2	-	1	-	1	-	-	-
Redimensionar elementos	2	2	-	-	-	-	-	-	-
Mover los elementos	-	-	-	-	-	-	-	-	-
Eliminar elementos	-	-	-	-	-	-	-	-	-
Renombrar los elementos	-	-	-	-	-	-	-	-	-
TOTAL	**12**	**12**	**0**	**6**	**4**	**2**	**1**	**1**	**0**

Tabla 11: Resultados de las pruebas de caja negra por iteraciones.

A continuación se muestra un gráfico donde se puntualiza por iteraciones el total de no conformidades identificadas, el total de no conformidades resueltas y la cantidad de no conformidades pendientes.

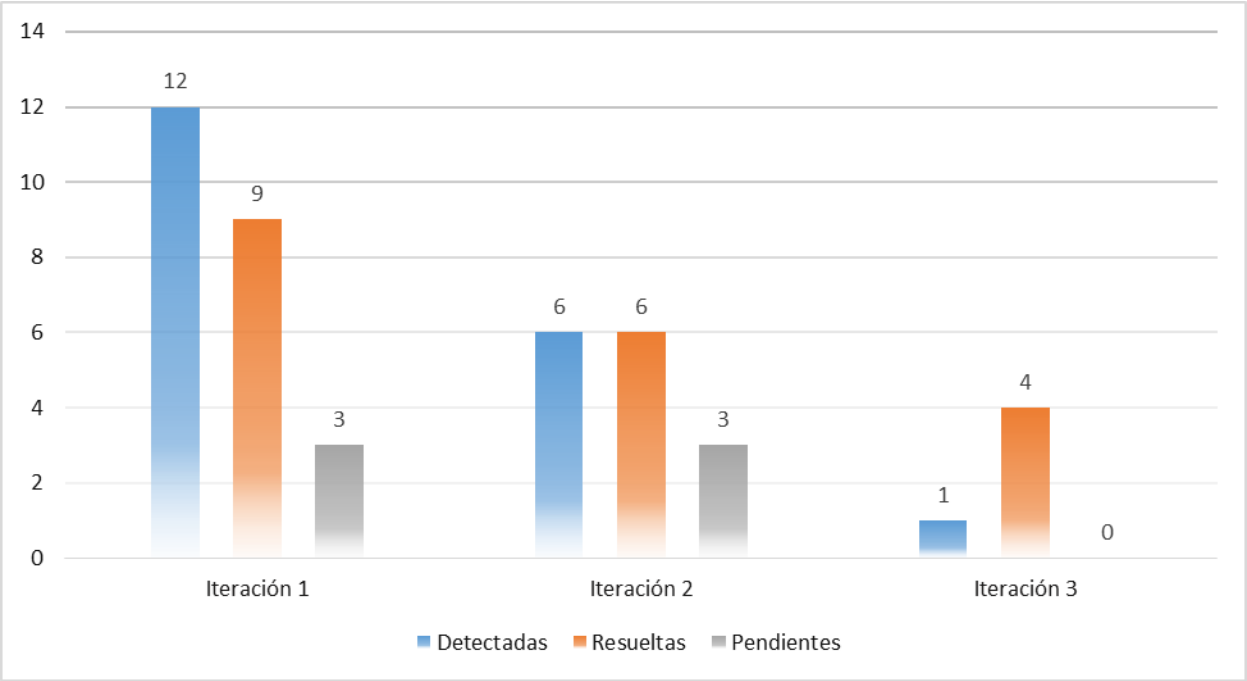


Ilustración 16: Resultados de las pruebas.

Conclusiones parciales

El modelado de una vista estática de la propuesta de solución mediante los diagramas de componentes ayudó al desarrollo de la implementación. La utilización de estándares de codificación facilitó la organización, el entendimiento y la limpieza del código fuente. La detección temprana de errores durante todo el proceso de pruebas funcionales con la realización del método de caja negra mejoró la calidad del producto final y garantizó que las funcionalidades sean las esperadas por el cliente.

CONCLUSIONES GENERALES

Con la realización del presente trabajo se arribaron a las siguientes conclusiones:

- El estudio del estado del arte sobre el marco de trabajo Xalix indicó el punto de partida de la solución, definiéndose las herramientas y tecnologías a utilizar.

- El análisis y diseño de la solución propuesta utilizando la metodología AUP-UCI generó un grupo de artefactos ingenieriles para futuros cambios y centró a los desarrolladores en el camino de la implementación.

- Las pruebas de software realizadas al producto final permitieron corregir un grupo de errores en un margen corto de tiempo, lo cual validó el correcto funcionamiento de la solución propuesta.

- El desarrollo de los componentes Tree, Drag and Drop y Web Editor bajo las pautas del marco de trabajo Xalix, satisfacen las principales funcionalidades de CRODA en su versión 3.0 y pueden ser reutilizables en otros proyectos de software.

RECOMENDACIONES

Teniendo en cuenta los resultados obtenidos en el presente trabajo se propones las siguientes recomendaciones:

- Personalizar el componente Tree para las especificaciones SCORM, LOM e IMS-LD.

- Personalizar el componente Drag and Drop para la especificación IMS-LD.

REFERENCIAS BIBLIOGRÁFICAS

1. FORTES - Centro de Tecnologías para la Formación. [online]. [Accessed 9 April 2016]. Available from: http://gespro.fortes.prod.uci.cu/login?back_url=http%3A%2F%2Fgespro.fortes.prod.uci.cu%2Fprojects%2Fredcentroxaucecroda30201410060944424713520 4%3Fjump%3Dwelcome

2. ADRIANA J. BERLANGA, FRANCISCO J. GARCÍA and JORGE CARABIAS. *IMS Learning Design: Hacia la Descripción Estandarizada de los Procesos de Enseñanza.* [online]. Available from: http://avellano.fis.usal.es/~aberlanga/files/Pubs/BerlangaetalSINTICE05-Pub.pdf

3. ROXANA CAÑIZARES GONZALEZ. *REPOSITORIO DE RECURSOS EDUCATIVOS PARA LAS INSTITUCIONES DE EDUCACIÓN SUPERIOR.* [online]. Available from: http://repositorio_institucional.uci.cu/jspui/handle/ident/3742/simple-search?query=roxana+ca%C3%B1isares

4. Learning Designs. [online]. [Accessed 10 April 2016]. Available from: http://learningforward.org/standards/learning-designs

5. SCORM - MoodleDocs. [online]. [Accessed 4 February 2016]. Available from: https://docs.moodle.org/all/es/SCORM

6. Learning Object Metadata - Wikipedia, la enciclopedia libre. [online]. [Accessed 4 February 2016]. Available from: https://es.wikipedia.org/wiki/Learning_Object_Metadata

7. XML. [online]. [Accessed 23 May 2016]. Available from: http://www.hipertexto.info/documentos/xml.htm

8. LABORATORIO NACIONAL DE CALIDAD DEL SOFTWARE. *INGENIERÍA DEL SOFTWARE: METODOLOGÍAS Y CICLOS DE VIDA* [online]. Available from: https://www.google.com.cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=10&sqi=2&ved=0ahUKEwihsvHB2tXLAhUHMyYKHR6KDcwQFghWMAk&url=https%3A%2F%2Fwww.incibe.es%2Ffile%2FN85W1ZWFHifRgUc_oY8_Xg&usg=AFQjCNENnl5-oTpq3s99afB4Lisiq4tJ3w&bvm=bv.117218890,d.dmo&cad=rja

9. TAMARA RODRÍGUEZ SÁNCHEZ. *Metodología de desarrollo para la Actividad productiva de la UCI.* 6 March 2015.

10. Visual Paradigm frequently asked questions. [online]. [Accessed 5 February 2016]. Available from: http://www.visual-paradigm.com/support/faq.jsp

11. Ingenieria de SoftwareUML - Monografias.com. [online]. [Accessed 29 March 2016]. Available from: http://www.monografias.com/trabajos5/insof/insof.shtml

12. PHP 5 Introduction. [online]. [Accessed 6 February 2016]. Available from: http://www.w3schools.com/php/php_intro.asp

13. Capítulo 4. DOM (Document Object Model) (Introducción a AJAX). [online]. [Accessed 10 June 2016]. Available from: http://librosweb.es/libro/ajax/capitulo_4.html

14. Lenguajes del lado servidor o cliente. [online]. [Accessed 6 February 2016]. Available from: http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html

15. JSON. [online]. [Accessed 10 June 2016]. Available from: http://www.json.org/json-es.html

16. CoDejaVu: Ejemplo JTree En Java. [online]. [Accessed 10 June 2016]. Available from: http://codejavu.blogspot.com/2013/11/ejemplo-jtree-en-java.html

17. ¿Qué es Symfony? [online]. [Accessed 6 February 2016]. Available from: http://symfony.es/pagina/que-es-symfony/

18. jQuery. [online]. [Accessed 9 April 2016]. Available from: https://jquery.com/

19. Bootstrap Get Started. [online]. [Accessed 9 April 2016]. Available from: http://www.w3schools.com/bootstrap/bootstrap_get_started.asp

20. Qué es un entorno de desarrollo integrado, IDE - Programacion Desarrollo. [online]. [Accessed 9 April 2016]. Available from: http://programaciondesarrollo.es/que-es-un-entorno-de-desarrollo-integrado-ide/

21. PhpStorm IDE :: JetBrains PhpStorm. [online]. [Accessed 14 May 2016]. Available from: https://www.jetbrains.com/phpstorm/

22. Sobre PostgreSQL | www.postgresql.org.es. [online]. [Accessed 6 February 2016]. Available from: http://www.postgresql.org.es/sobre_postgresql

23. Apache, el servidor Web más reconocido | Empresa y economía. [online]. [Accessed 29 March 2016]. Available from: http://empresayeconomia.republica.com/aplicaciones-para-empresas/apache-el-servidor-web-mas-reconocido.html

24. Modelo de Dominio | Tecnología y Synergix. [online]. [Accessed 10 April 2016]. Available from: https://synergix.wordpress.com/2008/07/10/modelo-de-dominio/

25. Ingeniería De Requerimientos Ingeniería De Software - Monografias.com. [online]. [Accessed 10 April 2016]. Available from: http://www.monografias.com/trabajos6/resof/resof.shtml

26. MARÍA PAULA IZAURRALDE. *Caracterización de Especificaciones de Requerimientos ene entornos Ágiles.* [online]. Available from: http://www.institucional.frc.utn.edu.ar/sistemas/lidicalso/pub/file/Tesis/Anteproyecto_Requerimientos_en_Metodolog%C3%ADas_Agiles.pdf

27. PRINCIPIOS DE DISEÑO | Mundo Kramer's Blog. [online]. [Accessed 22 April 2016]. Available from: https://mundokramer.wordpress.com/2011/05/19/principios-de-diseno/

28. Lección 39. Diagrama de Clases de Diseño. [online]. [Accessed 12 May 2016]. Available from: http://datateca.unad.edu.co/contenidos/200609/exeuml/leccin_39_diagrama_de_clases_de_diseo.html

29. Unidad 4: Modelo de diseño: 4.5 Diagramas de secuencia. [online]. [Accessed 12 May 2016]. Available from: http://rodrigoith.blogspot.com/2013/05/45-diagramas-de-secuencia.html

30. 2.1 DEFINICIÓN DE UN MODELO DE DATOS. | Tombasededatos's Blog. [online]. [Accessed 15 May 2016]. Available from: https://tombasededatos.wordpress.com/2010/08/28/2-1-definicion-de-un-modelo-de-datos/

31. HTML. [online]. [Accessed 23 May 2016]. Available from: http://www.hipertexto.info/documentos/html.htm

32. CRAIG LARMAN. *UML y Patrones*. [no date].

33. Patrones GoF. [online]. [Accessed 15 May 2016]. Available from: http://geektheplanet.net/5462/patrones-gof.xhtml

34. Qué es MVC. [online]. [Accessed 15 May 2016]. Available from: http://www.desarrolloweb.com/articulos/que-es-mvc.html

35. JAVIER EGUILUZ. *Desarrollo web ágil con Symfony2* [online]. [no date]. Available from: http://www.google.com.cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&cad=rja&uact=8&ved=0ahUKEwjRye-hvuTLAhWCWx4KHRS_Cd8QFgglMAI&url=http%3A%2F%2Fn3wton.net%3A2020%2Fget%2Fpdf%2F1092&usg=AFQjCNGG3iUL1CrXNd36zr0lvM5RLtg96g&bvm=bv.117868183,d.eWE

36. Unidad 5. [online]. [Accessed 3 May 2016]. Available from: https://es.scribd.com/doc/123267735/Unidad-5

37. Sparx Systems - Tutorial UML 2 - Diagrama de Componentes. [online]. [Accessed 12 May 2016]. Available from: http://www.sparxsystems.com.ar/resources/tutorial/uml2_componentdiagram.html

38. ERNESTO VLADIMIR PEREDA DÍAZ. *Pautas de codificación*.

39. ROGER S. PRESSMAN. *Ingenieria del software. Un enfoque practico* [online]. [no date]. Available from: http://eva.sepyc.gob.mx:8383/greenstone3/sites/localsite/collect/ciencia1/index/assoc/HASH015f/ceb375c1.dir/33040073.pdf

40. CARLOS BLANCO BUENO. *Ingeniería de software II* [online]. Available from: http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-software-ii/materiales/tema1-pruebasSistemasSoftware.pdf