UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 6

Implementación del Subsistema de Configuración de la Plataforma de Televisión Informativa, PRIMICIA.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN INFORMÁTICA

AUTOR: Osmany Torres Leyva

TUTOR: Ing. Carlos de Jesús Andrés González

La Habana, Junio de 2011

Año 53 de la Revolución

"Lo principal es la sabiduría; adquiere sabiduría,
y con todo lo que obtengas adquiere inteligencia."
Proverbios 4:7 (LBLA)

DEDICATORIA

Dedico este trabajo en primer lugar a mi mamá, mi papá y mi hermano que sin saber nada de informática, fueron capaces de educarme en el amor por el estudio, por el trabajo, por la competencia y mi auto-superación, haciéndome sentir que todo lo que hasta hoy he hecho para ellos vale muchísimo. Estas tres personas son modelos para mí en lo que a profesionalidad, seriedad y compromiso se trata. Amo mi profesión porque ellos aman las suyas y se entregan a lo que hacen. De la mejor manera que me han instruido ha sido con el ejemplo y no tengo nada que reprocharles en la educación que me han dado. Para ellos como para mí, lo más importante es la familia; no la institución que representa, sino los lazos que forma. Hemos sabido por instinto y amor, enfrentar cada proceso que nos ha sobrevenido y en lo que a mis estudios respecta, han sido de un estímulo sin igual.

Dedico este trabajo en segundo lugar pero también muy especialmente, a Ana Liz que usó durante todo el desarrollo de este trabajo, su capacidad para animarme con las palabras siempre correctas, las que necesité oír para vencer estrés y desánimo. Su apoyo durante mi carrera universitaria ha sido tan especial que este trabajo existe en gran medida gracias a ella. La tarea que tuvo de levantar mi autoestima y comportarse como mi ayuda idónea en su momento, la cumplió excelentemente y el resultado presentado aquí es también gracias a esto.

AGRADECIMIENTOS

Agradezco la ayuda prestada en el desarrollo y elaboración no solo de este trabajo sino de la aplicación que documenta:

A Dios que ha dicho de mí que soy un vencedor y al creerle he podido sobreponerme al cansancio, la dificultad y las pruebas que en el camino que acaba con este trabajo han aparecido.

A mis padres y mi hermano que creen en mí y me han apoyado incondicionalmente en todo.

A Ana Liz porque espiritualmente, soportó mi peso como las columnas de los templos antiguos soportaban toda su estructura.

Al proceso que ha hecho posible que la educación en este país sea gratuita y que exista un centro educacional de tan alto nivel como el que es hoy la Universidad de las Ciencias Informáticas.

A Carlos que me ha ayudado en todo.

A Ruber que creyó en mí para formar parte desde el segundo año del proyecto que es hoy PRIMICIA.

A todos los integrantes de este proyecto que aportaron a mis conocimientos y carácter y para los que espero también haber aportado algo.

Al grupo ANK en el cual fuimos escuderos unos de otros y encontré amigos verdaderos.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro de Geoinformática y Señales Digitales de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____ _____

Osmany Torres Leyva Ing. Carlos de Jesús Andrés González

OPINIÓN DEL TUTOR

El tutor del presente Trabajo de Diploma considera que durante su ejecución el estudiante mostró las cualidades que a continuación se detallan:

Durante el período de implementación demostró una alta independencia en las tareas desarrolladas, mostrando una alta creatividad y originalidad a la hora de interactuar con las tecnologías, que sirvieron de base para la consecución del Subsistema de Configuración. Ha puesto de manifiesto una alta responsabilidad y laboriosidad, cumpliendo tanto con las tareas de su trabajo de diploma como la tareas asignadas en el proyecto PRIMICIA donde se ha desempeñado de manera ejemplar.

En cuanto a los resultados obtenidos el autor mostró una gran receptividad ante las críticas y sugerencias efectuadas, así como la capacidad para apropiarse de los conocimientos que necesitaba para lograr la mayor calidad en los resultados alcanzados. Permitiendo obtener un subsistema capaz de acoplarse al producto y configurar las funcionalidades del mismo. El documento elaborado cumple con los requisitos establecidos, se plasma de manera concisa y con una excelente calidad, la respuesta a los objetivos trazados para la implementación. Los resultados alcanzados presentan un adecuado nivel científico, proporcionándole al producto PRIMICIA un grupo de funcionalidades que permitirán la adaptación de la plataforma a diferentes entornos de trabajo.

Por todo lo anteriormente expresado, considero que el estudiante ha cumplido los objetivos propuestos, venciendo cada una de las tareas previstas para su trabajo de diploma, estando apto para ejercer como Ingeniero Informático; y propongo que se le otorgue al Trabajo de Diploma la calificación de **5 puntos**.

Ing. Carlos de Jesús Andrés González

RESUMEN

La Plataforma de Televisión Informativa PRIMICIA, es un proyecto en desarrollo que existe gracias a la integración cada vez más profunda entre la informática y las comunicaciones. La televisión es el medio de comunicación que más rápido ha evolucionado desde que surgió como concepto alrededor del año 1884, con la invención del Disco de Nipkow[1]. Con el contemporáneo desarrollo de las tecnologías de la información y las comunicaciones, muchos procesos que se ejecutan en el ámbito televisivo se han ido automatizando. La plataforma PRIMICIA ejemplifica lo anteriormente dicho, al ser un sistema que permite gestionar y transmitir noticias en diferentes formatos, a través de una red de televisión y haciendo uso de medios informáticos.

A partir de numerosos inconvenientes detectados en el proceso de configuración de PRIMICIA, producto que se mantiene en constante evolución en cada ciclo de desarrollo, se determinó la ausencia de un conjunto de funcionalidades que permitan la configuración del producto. Guiado por la metodología de desarrollo de software RUP, se construyeron los principales artefactos para obtener el código fuente de un Subsistema de Configuración que satisface los problemas actuales de la plataforma, quedando plasmados en el presente trabajo.

PALABRAS CLAVES

PRIMICIA, Configuración, Subsistema, Funcionalidades, Implementación.

[1] *Dispositivo mecánico que permite analizar una escena de manera ordenada.*

DATOS EN INGLÉS

Title: Implementation of the Configuration Subsystem for the Informative Television Platform, PRIMICIA.

Author: Osmany Torres Leyva.

Tutor: Ing. Carlos de Jesús Andrés González.

Co-Tutor: Ing. Rafael Lorente Miranda.

The deepening integration between television and computing, specifically in the automation of the processes that takes place in this environment, is the area where the PRIMICIA Informative Television Platform project is developed. Television is the communication mean that has developed faster since it emerged as a concept around the year 1884 with the invention of the Nipkow Disk[2]. With the contemporary development of information and communications technologies, many processes running in television are been automated and the PRIMICIA platform exemplifies this, as it is a system to manage and deliver news in different formats, through a television network and using information technologies.

After several inconveniences found in the process of configuring PRIMICIA, it's been determined the absence of a set of features that allow a correct setting up of the product. Guided by the software development methodology RUP, major artifacts were built for the implementation of a configuration subsystem that solves the current problems of the platform; they are included in this work.

[2] *Mechanical device that allows to analyze a scene in an ordered way.*

FIGURAS

TABLAS

INDICE

INTRODUCCIÓN

El concepto de las tecnologías de la información y las comunicaciones (TIC)[3] es relativamente nuevo. Este ha hecho posible el englobe de un conjunto de herramientas que por su integración, hacen más fácil su aplique a los diferentes sectores de la sociedad. Un ejemplo claro de la integración mencionada lo representa la revolución que ocurre actualmente en el mundo de la televisión, uno de los medios de comunicación más difundidos a nivel mundial. La televisión es susceptible a constantes actualizaciones que van desde sus procesos, hasta las herramientas que se emplean para llevar imágenes en movimiento de un lugar a otro.

"La informática ha hecho posible que a través de la televisión se transmitan materiales audiovisuales de mayor calidad, con nuevos valores añadidos de interactividad y procesos de gestión de medias más eficientes" **(1)**.

La transmisión de un canal de televisión involucra muchos procesos que, gracias a la integración de esta con la informática, han sido poco a poco automatizados, especialmente en el área de la televisión informativa. La Plataforma de Televisión Informativa PRIMICIA, desarrollada en el Centro GEySED[4] de la Universidad de las Ciencias Informáticas (UCI), *"es un ejemplo de las ventajas que brinda la alianza creada entre la informática y la televisión, para el proceso de transmisión y gestión de noticias en diferentes formatos"* **(1)**.

Este sistema es el resultado de la generalización de un conjunto de soluciones informáticas, programadas a la medida de los clientes que se interesaron en la idea de poseer un canal de teletexto[5] para su centro o televisora. Sus principales antecedentes son las aplicaciones informáticas: **Señal 3**, sistema informativo creado para la red de televisión interna de la UCI que permite mantener informada

[3] *Conjunto de tecnologías que permiten la adquisición, producción, almacenamiento, tratamiento, comunicación, registro y presentación de informaciones en forma de voz, imágenes y datos contenidos en señales de naturaleza acústica, óptica o electromagnética.*

[4] *Geoinformática y Señales Digitales.*

[5] *Sistema de transmisión de textos escritos por medio de la televisión (DRAE).*

a la comunidad universitaria que se aloja en su residencia; **TVEnergía**, sistema personalizado creado con el propósito de mantener informados a trabajadores y visitantes de la sede central del Ministrerio del Poder Popular para la Energía y Petróleos de Venezuela (MENPET) y **Señal ACN**, canal de teletexto desarrollado para la Agencia de Información Nacional cubana y dedicado a transmitir informaciones en diferentes formatos, a colaboradores cubanos laborando fuera del país y para los habitantes de las zonas de silencio de la geografía cubana. El estudio de las soluciones mencionadas anteriormente, permitió la definición de un producto informático desarrollado en código y plataforma libres. Este producto es capaz de proveer un sistema informático para la redacción y edición de noticias, además de su posterior transmisión a través de un canal de televisión.

Aunque se defina la plataforma PRIMICIA como producto, esto contrasta con el hecho de que actualmente no permite ser configurado a conveniencia del usuario final. El nivel de configuración que permita un producto informático es importante, *"debido a las diferentes incompatibilidades que puedan existir entre el flujo de trabajo planteado por la empresa que lo produce y lo que realmente necesita el usuario que adquiere el software"* **(1)**. Con el propósito de convertir PRIMICIA en un producto que pueda ser adecuado a las necesidades de los clientes que lo adquieran y teniendo como antecedente trabajos de análisis y diseño, elaborados en el proyecto de desarrollo de la plataforma, se ha definido el siguiente **problema de la investigación:** ¿cómo obtener la capacidad operacional que permita la configuración de la Plataforma de Televisión Informativa PRIMICIA, a partir de los diseños elaborados con anterioridad? Como **objeto de estudio** se tomaron en consideración, los procesos relacionados con el funcionamiento de la Plataforma de Televisión Informativa PRIMICIA y su **campo de acción** fue enmarcado, en las funcionalidades de configuración de la Plataforma de Televisión Informativa PRIMICIA.

Se planteó como **objetivo** de este trabajo implementar un subsistema que permita la configuración de la Plataforma de Televisión Informativa PRIMICIA. Como guía para dar cumplimiento al objetivo propuesto, se definieron las siguientes **tareas de la investigación**:

1. Caracterizar el rol de Implementador a partir del uso de la metodología de desarrollo seleccionada.
2. Caracterizar el uso de los lenguajes de programación, herramientas y tecnologías a utilizar.
3. Caracterizar el diseño propuesto en trabajos anteriores.

4. Describir el estándar de codificación seleccionado para la implementación de las funcionalidades.
5. Implementar las funcionalidades diseñadas.
6. Documentar los principales algoritmos codificados.
7. Validar la solución propuesta.

Teniendo en cuenta lo anterior, se plantea la siguiente **idea a defender**: si se logra implementar el subsistema de configuración de PRIMICIA, se garantizará que la plataforma cuente con las funcionalidades necesarias que permitan su configuración.

Para cumplir con las tareas de la investigación, se emplearon los siguientes métodos científicos:

Métodos teóricos.

- **Analítico – Sintético:** Aportó la posibilidad de buscar la esencia de los fenómenos, los rasgos que caracterizan y distinguen los procesos relacionados con la configuración de PRIMICIA. Dentro de la investigación ha permitido la extracción de los elementos más importantes que se relacionan con la plataforma.
- **Inductivo – Deductivo:** Permitió llegar a un grupo de conocimientos generalizadores acerca de los posibles elementos configurables en la plataforma, así como determinar aquellos procesos más específicos implicados en los nuevos cambios que se necesitan para una correcta gestión de la configuración.

Métodos empíricos.

- **Observación**: Este método ha sido utilizado en distintos momentos de la investigación para recoger la información de cada uno de los conceptos o variables definidas en la idea a defender.
- **Entrevista**: Este método fue utilizado para conocer diferentes aspectos que no quedaron bien definidos en trabajos investigativos realizados con anterioridad, para lo cual fueron necesarias entrevistas con los principales desarrolladores de PRIMICIA, con el objetivo de profundizar en los elementos y problemática a tratar en la presente investigación.

CAPÍTULO 1. Establecimiento de la base teórica. Caracterización de las tecnologías a emplear.

1.1. Introducción

En este capítulo se abordan los principales conceptos asociados a la investigación, así como la caracterización de la metodología de desarrollo de software escogida para la documentación del trabajo. Se define la situación problemática que da origen a la investigación y un análisis del estado del objeto de estudio. Así mismo, se caracterizan y evalúan diseños previos que se tomarán como base para dar respuesta al problema enunciado.

1.2. Conceptos asociados al dominio de la investigación.

A continuación, se definen algunos conceptos que son esenciales para el mejor entendimiento del domino del problema y la investigación presente:

Configuración

Según el diccionario de la Real Academia Española (RAE) en su vigésima segunda edición:

"Disposición de las partes que componen una cosa y le dan su peculiar forma y propiedades anejas." **(2)**

En el ámbito de las ciencias informáticas sin embargo, esta palabra es muy usada para denotar la capacidad con que es posible manipular las propiedades y elementos de un sistema informático. De hecho, aunque no está registrada en la RAE, ha surgido en muchos grupos de desarrollo de sistemas informáticos y foros de Internet, la palabra *configurabilidad*. Esta palabra es empleada para expresar cuán configurable es un sistema informático, es decir, cuán opcional es y con cuánta facilidad pueden cambiarse sus opciones definidas de antemano. Para la presente investigación, por su ajuste a la actualidad informática, se respetará el concepto de **configuración** definido en la investigación del Ing. Carlos de Jesús Andrés:

“el conjunto de elementos que determinan la composición de un programa o sistema informático, son opciones que generalmente se establecen cuando se instala un software, pero también pueden implantarse en cualquier otro momento”. **(1)**

Rol

La RAE define esta palabra como *“función que alguien o algo cumple”* **(2)** y en la metodología RUP se trata como *“definición del comportamiento y la responsabilidad de un individuo o grupo de individuos que trabajan juntos como un equipo, en el contexto de una empresa”* **(3)***.* Puesto que el ámbito en el que se desarrolla la presente investigación está precisamente vinculado con la metodología RUP, todo el trabajo se regirá guiado por el segundo de los conceptos enunciados.

Artefacto

De las acepciones encontradas en la RAE, la que más se adecúa a los propósitos de esta investigación, es la que se encuentra bajo la categoría de despectivo: “*Máquina, mueble, y en general, cualquier objeto de cierto tamaño”* **(2)***.* Esta acepción refleja un poco el sentido con el que se emplea en el mundo informático y en especial, en el ámbito de RUP, pues con esta palabra se denomina a los ficheros, documentos, diagramas y en general a casi cualquier cosa que “*1) una tarea produce, modifica o utiliza, 2) define un área de responsabilidad, 3) está sujeto al control de versión”* **(3)***.* Para esta investigación se empleará el concepto definido por RUP.

Subsistema

La RAE no recoge definiciones en su última edición para esta palabra; sin embargo, es ya muy común en todos los ámbitos informáticos. Un subsistema es una forma de encapsulamiento, organización y agrupación de elementos, ficheros y clases. Según RUP, un subsistema es un “*Elemento de modelo que tiene la semántica de un paquete, ya que puede contener otros elementos de modelo, y una clase, ya que tiene comportamiento (…) Un subsistema ejecuta una o más interfaces, que defina el comportamiento que puede llevar a cabo*” **(3)***.*

1.3. Caracterización de las funcionalidades y procesos de PRIMICIA. Análisis de la situación problemática.

La Plataforma de Televisión Informativa PRIMICIA es un sistema diseñado para la transmisión cíclica de noticias en formatos de texto (teletexto), texto-imagen, imagen y video. Cada noticia va siempre acompañada de un fondo musical. La plataforma está basada en dos aplicaciones: el Subsistema de Administración, desarrollado empleando tecnologías web, está encargado de administrar el contenido de cada noticia que se almacena en una base de datos. Por su parte el Subsistema de Transmisión, desarrollado con el uso de tecnología para aplicaciones de escritorio, transmite a través de un canal televisivo las noticias de cualquiera de los tipos mencionados. Aunque estas aplicaciones están divididas, trabajan como un solo sistema para transmitir noticias. Esto se realiza utilizando como medio una red de televisión y haciendo uso de elementos informáticos. En estos subsistemas se ejecutan un conjunto de procesos que determinan el resultado final de la plataforma. A continuación una breve descripción de las funcionalidades según el subsistema a que pertenecen:

Subsistema de Administración:

- **Gestión de los usuarios del sistema**, permite adicionar y eliminar usuarios, así como establecer y modificar los permisos de acceso en el sistema.
- **Gestionar las secciones temáticas del canal**, permite establecer el orden de las secciones, horario en que serán mostradas y habilitarlas o deshabilitarlas.
- **Funcionalidades para la redacción de noticias,** según los formatos definidos para las pantallas; la publicación de las noticias teniendo en cuenta fecha de inicio y fin de la publicación; la gestión de las noticias del canal que permitan modificar, eliminar y archivar las noticias; la administración del archivo de noticias del canal que permitan reutilizar y eliminar las mismas.
- **Gestionar Recursos Multimedia,** posibilita el almacenamiento, administración y reproducción de recursos multimedia como imágenes, música y video.
- **Funcionalidades para la creación y administración de cintillos informativos o infocintas,** la administración de los cintillos establece el orden de prioridad de muestra y la habilitación o des-habilitación de los mismos.

- **Generación de reportes sobre la actividad del sistema,** los reportes se realizarán sobre la actividad de los trabajadores del sistema, permite realizar búsquedas de noticias publicadas atendiendo distintos criterios como fecha de publicación, temática, palabras claves y título. Ofrece facilidades para la impresión de los reportes y la exportación de los reportes a formato digital.
- **Administración de la señal del canal,** que permite cambiar entre la señal del canal y la televisión en vivo de un canal externo.

Subsistema de Transmisión:

- **Generar una cartelera,** del ciclo de transmisión mostrando para cada noticia la sección temática y el titular, en el orden que se visualizarán.
- **Visualizar noticias,** compuestas por pantallas de tipo texto, texto-imagen, imagen y video.
- **Reproducir fondo musical,** mientras se muestran las noticias, excepto cuando se muestre un video.
- **Mostrar comentarios,** que oriente al televidente acerca de lo que está observando en las pantallas de tipo imagen.
- **Mostrar cintillos informativos o infocintas,** para promocionar eventos de última hora o acontecimientos de gran importancia.
- **Mostrar información adicional a la noticia,** la fecha, hora, tiempo restante de la pantalla y titular de la próxima noticia y sección temática.
- **Transmitir televisión en vivo,** proveniente de una señal externa.
- **Mostrar patrón del canal,** cuando este se encuentre fuera de servicio.

Las relaciones que existen entre los diferentes procesos que se ejecutan dentro de la plataforma PRIMICIA, se describen de forma gráfica en la siguiente figura. Las saetas indican la dirección en que fluye el trabajo dentro de la plataforma.

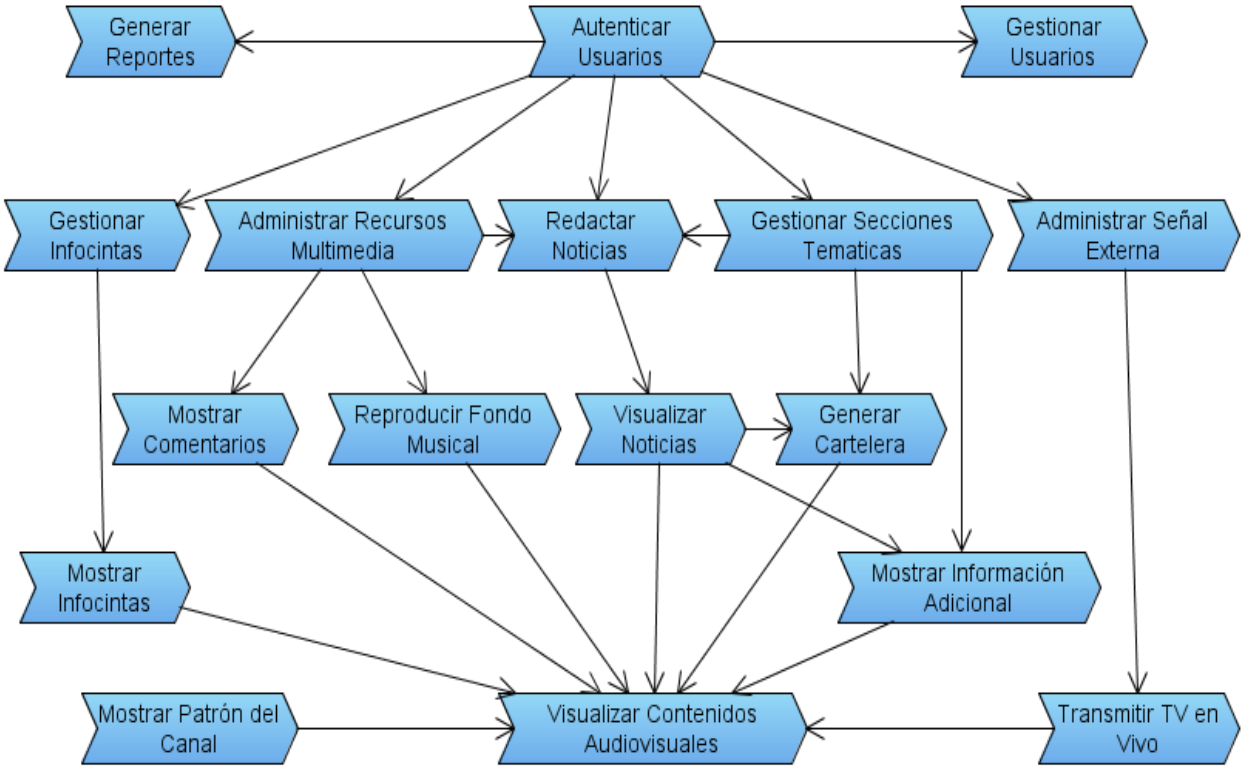

Figura 1: Diagrama de los procesos generales de PRIMICIA

Aunque algunos de estos procesos pueden trabajar independientes de los demás, como el de Gestionar Usuarios o el de Generar Reportes, entre la generalidad de ellos existe cierto grado de interdependencia que permite alcanzar el buen funcionamiento de la plataforma. La mayoría de estos procesos, sin embargo, pueden calificarse de “rígidos” cuando se trata de la posibilidad de la configuración de los mismos o de la influencia del usuario final sobre ellos. Actualmente, cuando se necesita adecuar la aplicación a los requerimientos de un nuevo cliente, el producto vuelve a pasar por las manos del equipo de desarrollo con el objetivo de personalizarlo accediendo directamente al código fuente. Esto significa que la plataforma no es lo más general posible y que no cuenta con las funcionalidades que permitan su configuración.

1.4. Caracterización de diseños previos.

Los problemas mencionados en el epígrafe anterior, fueron identificados en investigaciones previas. Durante el estudio realizado en el trabajo de diploma, “Conceptualización de la automatización de la personalización y configuración de PRIMICIA” del Ing. Levián Lara Gómez, se identificaron debilidades en la plataforma que como resultado de la investigación, fueron mitigadas y conceptualizadas como nuevas mejoras al sistema. Algunas de estas debilidades fueron:

- Deficiencia en la gestión de los roles de usuario en el subsistema de administración de la plataforma, donde todo este proceso es estático y no se permite modificar o crear nuevos roles que tengan acceso a funcionalidades deseadas y permitan variar el flujo de trabajo impuesto por la aplicación.
- Deficiencia en la gestión de las funcionalidades con que cuenta el sistema. Este tipo de sistema gestor de contenidos debería permitir la habilitación e inhabilitación de las funcionalidades con que cuenta a conveniencia del usuario final.
- Deficiencia en la gestión de las secciones temáticas[6]. PRIMICIA no permite hoy la adición de nuevas secciones o la modificación de las que el sistema traiga por defecto **(4)**.

Gracias a un estudio de los sistemas de administración y su especialización en sistemas de gestión de contenidos, al término de la investigación del Ing. Levián Lara Gómez, se logró la identificación de los procesos y elementos configurables de la plataforma PRIMICIA. Además, se conceptualizaron mejoras funcionales que permiten la configuración de los procesos presentes en el sistema. En el trabajo antes mencionado, se realizó un estudio de factibilidad que arrojó como resultado, que las mejoras que se le proponen a la plataforma son viables y tangibles tanto desde el punto de vista operacional como económico. También se concluyó que las nuevas mejoras que se proponen, dotarán a la plataforma PRIMICIA de un nivel de flexibilidad tal, que permitirá elevar las posibilidades de su elección por parte del cliente ante una posible competencia.

[6] *Forma de agrupamiento de las noticias por tema para su organización y visualización.*

La investigación antes referida fue analizada y empleada como base para el “Diseño del Subsistema de Configuración de la Plataforma de Televisión Informativa, PRIMICIA”, trabajo de diploma del Ing. Carlos de Jesús Andrés González. En su investigación, luego de un estudio profundo del diseño de la plataforma PRIMICIA, se corrobora igualmente que a raíz de la baja flexibilidad que presenta el sistema al configurar sus procesos, cada vez que sea necesario instalarlo en un nuevo entorno, es necesario que el equipo de desarrollo rediseñe y modifique el código fuente en todos los niveles de su arquitectura, eficientemente explicada en el trabajo antes mencionado. Aún cuando los cambios puedan ser pocos, se hace alto el costo en tiempo cada vez que se quiere obtener una personalización de la plataforma PRIMICIA, lo que puede provocar la pérdida de mercados y clientes.

Gracias al trabajo de conceptualización realizado anteriormente, el Ing. Carlos de Jesús pudo concentrarse en la obtención de los diagramas de clases de acuerdo con la metodología de desarrollo de software seleccionada en su pesquisa. Esto mismo permitió que se obtuviera la descripción de nuevas funcionalidades, que permiten tener una visión general del nuevo subsistema que se espera obtener al término de la presente investigación.

1.5. Caracterización de las tecnologías a emplear.

1.5.1. Metodología de desarrollo.

Una metodología de desarrollo de software es un “*conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software”* **(3)**.

Varios son los tipos y las metodologías a tener en cuenta cuando se va a elegir una para su aplicación en el proceso de desarrollo de un software, lo cual está en dependencia del tipo de proyecto que se esté desarrollando o planee desarrollar. Las metodologías de desarrollo de software se clasifican por lo general en dos grandes grupos, metodologías robustas y metodologías ágiles. Esta clasificación viene dada por la aplicación más eficiente de las diferentes metodologías a procesos de desarrollo que dependen en gran medida de los factores tiempo, economía y recursos humanos, entre otros. Por lo general, en proyectos grandes o con cronogramas holgados, se opta por metodologías robustas que se erigen sobre una extensa documentación de todas las actividades que se ejecutan dentro del proceso

de desarrollo. Las mismas cuentan con grandes volúmenes de artefactos a crear y mantener actualizados. En cambio, cuando el proyecto es pequeño o posee un cronograma apretado, es muy factible elegir una metodología ágil que basa su filosofía en *"desarrollar software que funciona más que conseguir una buena documentación"* **(5).**

Entre las metodologías robustas, podemos mencionar al Proceso Unificado de Desarrollo (RUP por sus siglas en inglés) y Métrica y Proceso Unificado Abierto (OpenUP por sus siglas en inglés). Como se había mencionado, estas metodologías arrojan una extensa documentación que es empleada para ejercer un estricto control sobre los procesos y ha demostrado ser muy efectiva su aplicación en proyectos de gran tamaño. Por otro lado, entre las metodologías ágiles se encuentran Programación Extrema (XP por sus siglas en inglés), Scrum y Marco de Trabajo para Soluciones Microsoft (MSF por sus siglas en inglés). En estas se antepone la obtención de soluciones prácticas a la exhaustiva documentación. Estas son la mejor elección cuando se trata de proyectos con entornos y requisitos ambiguamente definidos y/o con tendencia a cambiar varias veces durante el tiempo de vida del proyecto.

Por las características que ha presentado desde sus inicios el proyecto PRIMICIA tales como su envergadura, las condiciones legales de su concepción, el nivel de documentación requerido y las posibilidades de continuar su mejora en el futuro, la metodología RUP se presenta como la más indicada para la implementación del Subsistema de Configuración de la plataforma. RUP no solo cubre las necesidades presentadas por las características antes mencionadas, sino que de manera general sustenta el desarrollo del proyecto a través de un efectivo sistema de roles y responsabilidades bien distribuidas, garantizando que se generen todos los artefactos que permitirán la continuidad del trabajo en el proyecto. Además, habiendo sido esta misma la metodología empleada por el Ing. Carlos de Jesús en su "Diseño del Subsistema de Configuración de la Plataforma de Televisión Informativa, PRIMICIA", resulta muy conveniente realizar los siguientes pasos en la implementación del subsistema bajo su cobertura. El hecho de que los líderes y el equipo de desarrollo estén familiarizados con el modo de trabajo de RUP, por su empleo desde los inicios del proyecto, apoya la decisión de utilizarla para su aplicación en la presente investigación.

“El Rational Unified Process (RUP) es un Proceso de Ingeniería de Software. Provee un acercamiento disciplinado a la asignación de tareas y responsabilidades dentro de una organización de desarrollo. Su meta es asegurar la producción de software de alta calidad que satisfaga la necesidad de sus usuarios finales, dentro de límites de tiempo y presupuesto predecibles” **(3)**.

Esta metodología, más allá de enfocarse en la producción de grandes cantidades de documentos, enfatiza en crear y mantener modelos que son representaciones, lógicamente ricas, del sistema de software en desarrollo. En ella se efectúan un conjunto de actividades que transforman los requisitos de un usuario o cliente en un sistema de software. Una de las características por las que es ampliamente aceptada es por su empleo del Lenguaje Unificado de Modelado (UML por sus siglas en inglés) para la representación de sus artefactos. Además posee una amplia documentación tanto en idioma inglés como en español.

El ciclo de vida de RUP se identifica por tres características fundamentales. Estas son:

- **Guiado por casos de uso**: Un caso de uso es un fragmento de funcionalidad del sistema, una operación que este ejecuta y que proporciona al usuario un resultado importante. Se puede decir que un caso de uso responde a la pregunta “¿qué debe hacer el sistema (en una situación determinada) para cada usuario?”. RUP pone la mayor importancia en los casos de uso, ellos guían el proceso, pues su correcto modelado permitirán que su posterior implementación sea lo más sencilla posible **(3)**.
- **Centrado en la arquitectura:** La arquitectura involucra los aspectos estáticos y dinámicos más significativos del sistema, está relacionada con la toma de decisiones que indican cómo tiene que ser construido el sistema y ayuda a determinar en qué orden. La arquitectura se ve influenciada por la plataforma de software, sistema operativo, gestor de bases de datos, protocolos, consideraciones de desarrollo como sistemas heredados, entre otros **(3)**.
- **Iterativo e incremental:** La complejidad cada vez mayor de los sistemas que desarrollan en la actualidad hace que su desarrollo ya no se pueda ver como una línea recta. RUP propone que este se divida en miniproyectos, donde cada miniproyecto es una iteración dentro de la metodología que resulta en un incremento de los casos de uso implementados, actualización de la documentación que se genera y el crecimiento del producto final **(3)**.

Así entonces, RUP está dividido en 4 fases durante las cuales se transita por 9 flujos de trabajo, como se muestra en la Figura 2. No es objetivo de este epígrafe la explicación de cada fase y cada flujo; sin embargo, es necesario decir que dentro de cada una de ellas se activan o desactivan los diferentes roles de trabajo que propone la metodología, en dependencia de las funciones que se deban ejecutar.

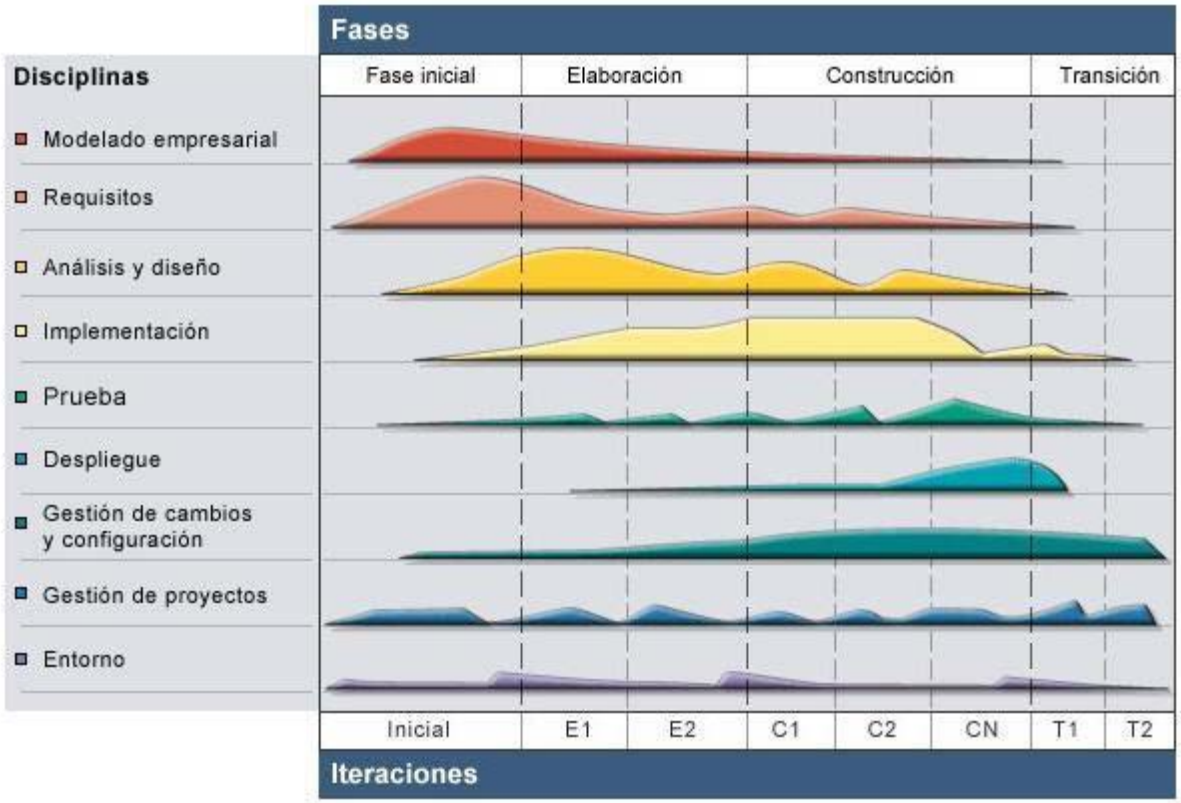

Figura 2: Arquitectura global de la metodología RUP

Los roles que propone RUP para organizar el trabajo y las responsabilidades del proceso se agrupan en conjuntos. Estos son: Analistas, Desarrolladores, Gestores, Producción y soporte, Roles generales y Verificadores **(6).** Por la cantidad de roles que se agrupan dentro de estos conjuntos, no se especificarán todos en esta investigación, pero sí será detallado el rol de Implementador que se inserta dentro del conjunto Desarrolladores. Este rol genera y modifica artefactos que son muy útiles para documentar y fundamentar la implementación de un sistema.

1.6. Rol Implementador.

"El rol del implementador es responsable de los componentes de desarrollo y de prueba, de acuerdo con los estándares aprobados por el proyecto, para la integración en subsistemas más grandes. Cuando los componentes de prueba, como controladores o fragmentos para simulación, deben crearse para dar soporte a las pruebas, el implementador también es responsable del desarrollo y las pruebas de los componentes de prueba y los subsistemas correspondientes" **(6).**

Según RUP, el Implementador es el que se encarga de desarrollar los componentes de software y de efectuar las pruebas de desarrollador para conseguir la integración en subsistemas más grandes, de acuerdo con los lineamientos asumidos en el proyecto.

Para este rol, RUP propone las siguientes habilidades:

- Conocimiento del sistema o aplicación que se somete a prueba.
- Familiaridad con las herramientas de prueba y de automatización de prueba.
- Habilidades de programación **(6)**.

Gracias a estas habilidades, al Implementador se le puede asignar la responsabilidad de implementar una parte estructural del sistema, como las clases de un subsistema definidas por un diseñador, o una parte funcional como la ejecución de casos de uso o sus características.

La siguiente figura muestra de manera visual las actividades que ejecuta y los artefactos que genera el Implementador, usando UML como lenguaje de representación visual:

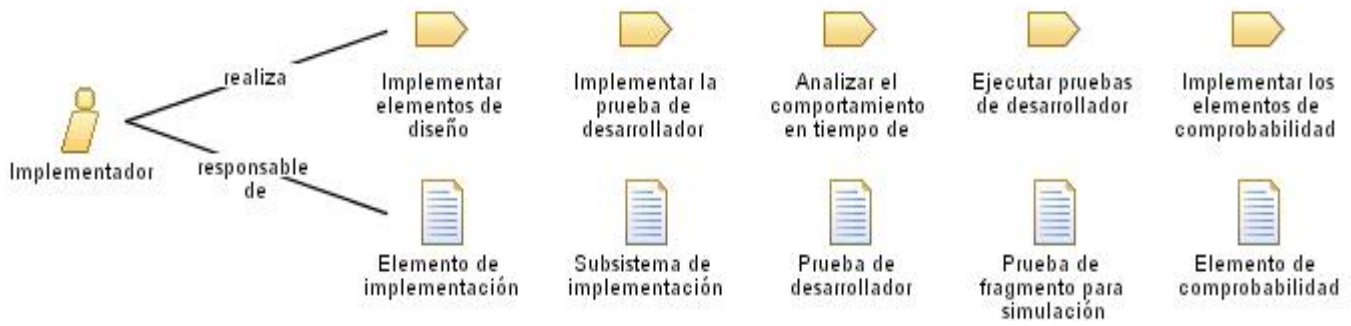

Figura 3: Actividades y artefactos del rol Implementador (6)

En la siguiente tabla se describen en detalle los elementos de la figura anterior para el rol Implementador según Proceso Unificado de Desarrollo:

ACTIVIDAD QUE REALIZA	ARTEFACTO QUE GENERA O MODIFICA
Implementar elementos del diseño: Se describe cómo producir una implementación para una parte del diseño, por ejemplo, una clase o una entidad de base de datos. El resultado son archivos nuevos o modificados de datos y de código fuente, que se conocen generalmente como elementos de implementación.	**Elemento de implementación:** Componentes físicos que forman una implementación, que conforman archivos y directorios. Incluyen los archivos de código de software (origen, binario o ejecutable), los archivos de datos y los archivos de documentación, como los archivos de ayuda en línea.
Implementar la prueba del desarrollador: Se describe cómo crear un conjunto de pruebas para comprobar que el componente funciona correctamente antes de que se realicen más pruebas formales en él.	**Prueba del desarrollador:** Artefacto que abarca las pruebas de unidad[7], parte de las pruebas de integración[8] y algunos elementos de las pruebas del sistema[9].

[7] *Prueba detallada de la estructura interna de un módulo de software.*

[8] *Prueba para obtener grado de integración de los elementos desarrollados.*

[9] *Pruebas que van desde la funcionalidad del sistema implementado hasta su usabilidad pasando por otras características.*

ACTIVIDAD QUE REALIZA	ARTEFACTO QUE GENERA O MODIFICA
Analizar el comportamiento en tiempo de ejecución: Se describe cómo analizar el comportamiento de un componente durante su ejecución para identificar las mejoras que se pueden realizar.	**Resultado de la prueba:** Resume el análisis de uno o más registros de prueba y solicitudes de cambio, proporcionando una valoración detallada de la calidad de los elementos de destino de la prueba y el estado del esfuerzo de prueba.
Ejecutar pruebas de desarrollador: Se describe cómo ejecutar y evaluar un conjunto de pruebas diseñadas para validar que el componente funciona correctamente antes de que se realicen más pruebas formales en el componente.	**Registro de prueba:** Contiene la salida bruta capturada durante una ejecución única de una o más pruebas.
Implementar elementos de comprobabilidad: Se describe cómo implementar funcionalidad especializada para dar soporte a los requisitos específicos de la prueba.	**Prueba de fragmento para simulación:** Elemento de implementación especializado que se utiliza para efectuar pruebas, que simula un componente verdadero.

Tabla 1: Actividades y artefactos principales del rol Implementador (6)

Otros artefactos que modifica el Implementador:

- **Subsistema de Implementación:** consta de un conjunto de elementos de implementación. Estructura el modelo de implementación dividiéndolo en componentes más pequeños que se pueden integrar y probar separadamente.
- **Elemento de comprobabilidad:** Es un elemento de implementación especializado que realiza el comportamiento específico de la prueba al que da soporte el software.

De manera general, podemos resumir que el Implementador es el rol dentro de RUP que se encarga de llevar de diseño a código las clases y artefactos definidos, documenta todo lo implementado y realiza las primeras pruebas que sufre el sistema o los componentes creados **(6)**.

1.7. Tecnologías.

Lenguaje de Programación

PHP: *"Acrónimo de 'PHP: Hypertext Preprocessor', es un lenguaje 'Open Source' interpretado de alto nivel, especialmente pensado para desarrollos web y el cual puede ser incrustado en páginas HTML. La mayoría de su sintaxis es similar a C, Java y Perl y es fácil de aprender. La meta de este lenguaje es permitir escribir a los creadores de páginas web, páginas dinámicas de una manera rápida y fácil, aunque se pueda hacer mucho más con PHP"* **(7).**

Este es un lenguaje de programación interpretado[10] creado con el objetivo de construir páginas web dinámicas en las que exista procesamiento del lado del servidor. Este lenguaje permite a las aplicaciones web ejecutadas por un servidor, generar contenido de manera dinámica, como por ejemplo, información contenida en una base de datos para su alteración. PHP puede ser interpretado por la mayoría de las aplicaciones servidores web existentes y en casi todas las plataformas sin costo alguno. Posee soporte por *The PHP Group* y su versión más reciente es la 5.3.3 de julio del año 2010.

En el Capítulo 2 de "Diseño del Subsistema de Configuración de la Plataforma de Televisión Informativa, PRIMICIA", del Ing. Carlos de Jesús Andrés González, en la definición de los requerimientos no funcionales del nuevo subsistema a desarrollar se definió el uso del marco de trabajo (*framework*) Symfony de desarrollo en PHP. El Subsistema de Administración de la plataforma PRIMICIA fue desarrollado igualmente sobre el *framework* antes mencionado, con el objetivo principal de aprovechar la arquitectura Modelo-Vista-Controlador incorporado por este. Por esa misma razón y por la experiencia obtenida en iteraciones anteriores del desarrollo del producto PRIMICIA, se empleará para la implementación del Subsistema de Configuración el lenguaje de programación PHP y su integración con el *framework* Symfony.

[10] *Lenguaje de programación diseñado para ser ejecutado por medio de un intérprete (en el caso de PHP, a través de una aplicación servidora como Apache).*

Marco de trabajo

Symfony: Marco de trabajo diseñado para optimizar el desarrollo y el trabajo de aplicaciones web de gran envergadura, mediante el uso de las características que incorpora. Algunas de las más importantes son:

- Separa la lógica de negocio, el acceso a datos y la lógica de presentación gracias a la incorporación de la arquitectura Modelo-Vista-Controlador, como se mencionó anteriormente.
- Posee un gran núcleo de clases implementadas que facilitan el desarrollo de una aplicación web compleja gracias a la reducción del tiempo de realización.
- Automatiza las tareas más comunes permitiendo al desarrollador enfocarse en los aspectos físicos del desarrollo de la aplicación **(8)**.

Symfony fue desarrollado completamente sobre la versión 5 de PHP y según sus propios creadores, *"ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel"* **(8).** Este *framework* es multiplataforma, es decir, puede utilizarse en sistemas operativos tales como Microsoft y plataformas *nix (Unix, Linux, etc.). También es compatible con la mayoría de los sistemas gestores de bases de datos existentes, como MySQL, PostgreSQL y Oracle.

El *framework* Symfony, es soportado por SensioLabs **(9),** empresa francesa que se dedica al desarrollo de aplicaciones web de gran envergadura y al desarrollo y soporte del *framework*. Igualmente, se encarga del Mapeador Objeto-Relacional Doctrine. Algunas de las características principales de Symfony según su creador, el francés Fabien Potencier, que lo convierten en la mejor opción para su aplicación en el desarrollo del Subsistema de Configuración de PRIMICIA son:

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.

- Basado en la premisa de *"convenir en vez de configurar"*, en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de "*mejores prácticas"* y patrones de diseño para la web.
- Preparado para el desarrollo de aplicaciones empresariales y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo **(8)**.

El uso de Symfony es muy conveniente pues automatiza elementos que son comunes cuando se desarrolla una aplicación, como por ejemplo:

- La capa de internacionalización que incluye Symfony permite la traducción de los datos y de la interfaz, así como la adaptación local de los contenidos.
- La capa de presentación utiliza plantillas y *layouts*[11] que pueden ser creados por diseñadores HTML sin ningún tipo de conocimiento del *framework*.
- Los formularios incluyen validación automatizada y relleno automático de datos, lo que asegura la obtención de datos correctos y mejora la experiencia de usuario.
- La autenticación y la gestión de credenciales simplifican la creación de secciones restringidas y la gestión de la seguridad de usuario.
- Los listados son más fáciles de utilizar debido a la paginación automatizada, el filtrado y la ordenación de datos **(8)**.

La arquitectura y los patrones arquitectónicos que emplea Symfony, mejor detallados en el trabajo realizado por el Ing. Carlos de Jesús, dotan a los sistemas que lo emplean de una gran robustez y seguridad, especialmente para proyectos de la envergadura de la plataforma PRIMICIA e incluso más grandes. Además de ser desarrollado por la reconocida empresa SensioLabs, el *framework* cuenta con una gran comunidad internacional que ofrece soporte y solución a problemas que puedan surgir durante la codificación de una aplicación web.

[11] *Palabra anglosajona que se emplea en el ámbito del desarrollo de aplicaciones para referirse a archivos de presentación de los datos.*

Entorno de Desarrollo Integrado

NetBeans IDE: *"Es un entorno de desarrollo - una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas"* **(10)**. Este Entorno de Desarrollo Integrado (IDE por sus siglas en inglés), fue escrito usando Java como lenguaje de programación y este mismo lenguaje se emplea como base para la codificación de sus aplicaciones, pero es extensible a cualquier otro lenguaje incluyendo PHP. NetBeans IDE puede ser configurado para usar plenamente las potencialidades de *frameworks* como el propio Symfony, con un muy eficiente completamiento automático de código, generación de funcionalidades menores y ejecución de los comandos que brinda Symfony para la configuración de cualquier aplicación web.

NetBeans IDE es un producto de código abierto, multiplataforma y posee una gran comunidad que brinda soporte y soluciones. Además es desarrollado por la compañía Oracle Corporation, quien posee otros famosos proyectos como el gestor de Base de Datos Oracle y la aplicación servidora GlassFish. Debido a la alta integración que posee con el lenguaje PHP y el *framework* Symfony, los cuales son indispensables para la implementación del Subsistema de Configuración de PRIMICIA, NetBeans IDE es la mejor opción a elegir para la codificación de la aplicación.

Otras tecnologías

ORM Doctrine: *"(…) es un mapeador relacional de objetos (ORM) para PHP 5.2.3+ que se sienta sobre una poderosa capa de abstracción de base de datos (DBAL)"* **(11)**. Es una librería de clases encargada de proveer a los desarrolladores de aplicaciones web que usan PHP, de una capa que abstraiga la lógica del negocio en la aplicación, de la base de datos con que trabaja. Doctrine proporciona un sofisticado lenguaje orientado a objetos para el trabajo con datos persistentes y el acceso propiamente dicho a la base de datos denominado *Doctrine Query Language* (DQL). Este "*(…) provee a los desarrolladores de una poderosa alternativa al SQL (Structured Query Language) que mantiene la flexibilidad sin requerir duplicación innecesaria de código"* **(11)**

Doctrine es el ORM predefinido en Symfony desde la versión 1.3 del *framework* y gracias a la integración que existe entre ambos, permite que para el desarrollo de aplicaciones web se genere a

través de Doctrine la capa que corresponde al modelo en la arquitectura que emplea Symfony. Esta eficiente operación se realiza usando máxime dos simples comandos.

Doctrine tiene varias ventajas que son de gran importancia para implementación del Subsistema de Configuración de PRIMICIA como su integración con Symfony, la sencillez de su lenguaje DQL y el soporte que brindan tanto su comunidad como la empresa Sensio que patrocina el proyecto de su desarrollo.

Firebug: Es una extensión del navegador Mozilla Firefox desarrollada especialmente para su uso por desarrolladores web. Esta extensión contiene un paquete de utilidades que permiten analizar, editar, monitorear y depurar el código fuente, CSS, HTML y JavaScript de una página web de manera instantánea durante su visualización en el navegador. Firebug está encapsulado en forma de complemento para Mozilla Firefox, es Open Source, libre y de distribución gratuita.

Sus funcionalidades más relevantes son:

- **Inspecciona y edita HTML**: Firebug simplifica el encontrar elementos HTML escondidos en la página web. Una vez encontrado, Firebug brinda toda la información posible sobre ese elemento y permite la edición de su HTML en el instante.
- **Ajusta propiedades CSS**: La pestaña CSS de Firebug informa al desarrollador de todo lo necesario acerca del estilo definido en las páginas web, permitiendo hacer cambios durante la visualización de la página y ver sus efectos inmediatamente.
- **Monitorea la actividad en la red:** Firebug permite investigar, monitoreando el flujo de red, que páginas demoran en cargar y por qué razón, al permitir este análisis archivo por archivo.
- **Permite depurar código JavaScript:** Firebug incluye un poderoso depurador de JavaScript que permite pausar la ejecución en cualquier momento y ver el estado de variables y procesos.
- **Encuentra errores rápidamente:** A través de su consola, Firebug informa de inmediato y detalladamente acerca de errores en JavaScript, CSS y XML.
- **Explora el DOM:** El Document Object Model (DOM) es una gran jerarquía de objetos y funciones que permiten a través de JavaScript la manipulación de los elementos de una

página web. Firebug permite encontrar rápidamente objetos DOM y editarlos mientras la página es visualizada.

- **Ejecuta JavaScript "al vuelo":** Firebug permite interrumpir la ejecución de una página web y la introducción de nuevas funciones JavaScript para ver su resultado en el momento preciso de la visualización **(12)**.

Todas estas características logran que el complemento Firebug se convierta en una herramienta casi necesaria para el desarrollo de aplicaciones web que necesiten un control constante sobre la presentación y la consecuente depuración de esta. Desde las primeras iteraciones realizadas durante el desarrollo de PRIMICIA, este complemento ha sido empleado por los desarrolladores y ha probado ser muy útil.

1.8. Estándar de codificación.

Por políticas del centro GEySED, se centralizó el estándar de codificación de las aplicaciones que se desarrollan en el centro. El objetivo de esto es orientar la producción a la creación de componentes reutilizables. El mismo se describe a continuación:

Principios generales:

- Los nombres de cada uno de los elementos del programa deben ser significativos.
- No manejar en los programas más de una instrucción por línea.
- Declarar las variables en líneas separadas.
- Añadir comentarios descriptivos junto a cada declaración de variables, si es necesario.
- La mayoría de los elementos se deben nombrar usando sustantivos.
- Los atributos deben comenzar con letra minúsculas.
- Los métodos deben comenzar con letra mayúsculas.

Nombramiento de elementos

Clases, interfaces y archivos:

- Nombre sustantivo singular, con la primera letra en mayúscula y las demás en minúsculas.
- Si el nombre de la clase está comprimido en más de una palabra la primera letra debe de ser mayúscula. Letras mayúsculas seguidas no son permitidas. Ejemplo: ***"Usuario_CLASS"*** no es permitido mientras que ***"Usuario_Class"*** si lo es.
- Las clases deben comenzar con un prefijo alegórico al componente que correspondan, evitando nombres duplicados. Ejemplo: ***"Controles"*** no es un nombre permitido mientras que ***"Reproductor_Controles"*** si lo es para el componente Reproductor
- Las API[12] deben comenzar con un prefijo alegórico al componente que correspondan, evitando nombres duplicados y terminar en el sufijo ***_Component***. Ejemplo: ***"Controles"*** no es un nombre permitido mientras que ***"Reproductor_Controles_Component"*** si lo es.
- Las constantes usarán nombre sustantivo en mayúsculas. Para separar palabras se usará el guión bajo (_).
- Los paquetes emplearán nombre sustantivo singular en minúsculas
- Los identificadores de variables comenzarán siempre con la primera letra minúscula correspondiente a su tipo de dato. Para distinguir palabras dentro del nombre deberá emplearse un guión bajo (_). Ejemplo: ***formato_de_video***

Las funciones implementadas llevarán comentarios describiendo el objetivo de la función pero no la descripción del funcionamiento. Si se explicará el uso de los argumentos (parámetros) y los valores devueltos (de retorno). Estas reglas permitirán estandarizar el código escrito en los módulos, paquetes o componentes que hayan sido desarrollados por personas diferentes.

[12] *Aplication Program Interface (Interfaz de Aplicación de Programas).*

1.9. Conclusiones Parciales.

Se efectuó a lo largo de este capítulo un breve pero conciso estudio de las características fundamentales que presenta en la actualidad la plataforma PRIMICIA, aportando un análisis de los elementos teóricos que sirven como base para este trabajo. Fue posible constatar la baja flexibilidad que presenta la mencionada plataforma lo cual permitió plasmar el problema a que se enfrenta esta investigación y su punto de partida. A través del análisis de estudios previos, se decide tomar como base para la implementación de las nuevas funcionalidades de PRIMICIA, el trabajo de diploma del Ing. Carlos de Jesús Andrés González en el que se realizó el diseño y caracterización de las mismas. Además, se describieron las herramientas y tecnologías que se utilizarán para apoyar y dar respuesta al problema planteado haciendo especial énfasis en la metodología que se seguirá y el estándar de codificación.

CAPÍTULO 2. Implementación de la solución.

2.1. Introducción.

En este capítulo se brindarán los elementos que son imprescindibles para la comprensión de la solución que se propone. Se especificarán además, un grupo de modificaciones que se le administran al diseño propuesto en trabajos anteriores. Además, se expone un conjunto de diagramas que resultan básicos para el trabajo de un implementador según las características que propone RUP para dicho rol, declaradas en el capítulo anterior. Se representa igualmente, el código fuente de las principales funcionalidades implementadas y se explica el uso de determinadas herramientas que permiten la generación de código para flexibilizar el trabajo.

2.2. Especificación de los requisitos de software.

La metodología RUP define un requisito de software como:

"Especificación de un comportamiento del sistema observable externamente; por ejemplo, entradas al sistema, salidas del sistema, funciones y atributos del sistema, o atributos del entorno del sistema." **(6)**

Lo anterior significa que los requisitos son obligaciones, exigencias con las que el sistema que se implemente debe cumplir. Estas funciones y atributos se definen para que el usuario final se beneficie lo mayor y mejor posible del sistema en cuestión y delimitan su alcance y objetivo de construcción.

Con el objetivo de ofrecer un punto de partida, se listan en este subepígrafe los requisitos funcionales y no funcionales del software a implementar definidos en el trabajo de diploma del Ing. Carlos de Jesús Andrés González y tomados como referencia.

Requisitos funcionales

Los requisitos funcionales *"son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en*

situaciones particulares" **(13)**. Estos requisitos sencillamente dictan lo que el sistema debe hacer para dar respuesta a lo que el usuario necesita. El listado que se muestra a continuación fue directamente tomado del trabajo del Ing. Carlos de Jesús Andrés González.

R1 Permitir que se pueda establecer el idioma para la interfaz de usuario de la plataforma.

R2 Dar la posibilidad que se pueda establecer el tipo de autenticación a usar en la plataforma.

R3 Permitir que a través de la plataforma se gestione el nombre y los permisos para los roles de usuarios.

R3.1 Crear roles de usuarios.

R3.2 Modificar roles de usuarios.

R3.3 Eliminar roles de usuarios.

R3.4 Ver los permisos establecidos para cada uno de los roles de usuarios.

R4 Establecer el tipo de transmisión usada por la plataforma, ya sea por secciones temáticas, bloques de noticias o de manera continua.

R5 Gestionar las secciones temáticas usadas por la plataforma para la transmisión.

R5.1 Crear sección temática.

R5.2 Modificar sección temática.

R5.3 Eliminar sección temática.

R5.4 Ver el listado de las secciones temáticas habilitadas para la transmisión.

R6 Gestionar los bloques noticiosos usados por la plataforma para la transmisión.

R6.1 Crear bloques de noticias.

R6.2 Modificar bloques de noticias.

R6.3 Eliminar bloques de noticias.

R6.4 Ver el listado de los bloques de noticias habilitados para la transmisión.

R7 Gestionar los canales externos, con los que la plataforma puede enlazarse para cambiar su transmisión.

R7.1 Adicionar canal externo.

R7.2 Modificar canal externo.

R7.3 Eliminar canal externo.

R7.4 Ver el listado de los canales externos habilitados para enlazarse.

R8 Establecer los servicios web que serán usados para obtener las informaciones adicionales que se muestran durante la transmisión.

R9 Gestionar los sitios web que serán usados por el lector de noticias de la plataforma.

R9.1 Adicionar fuentes de información.

R9.2 Eliminar fuentes de información.

R9.3 Ver el listado de fuentes de información.

R10 Permitir que se puedan habilitar y deshabilitar los módulos y funcionalidades de la plataforma.

R11 Permitir que se puedan habilitar y deshabilitar los reportes que se instalen con la plataforma y seleccionar el formato de salida de los mismos. **(1)**

Todos estos requisitos fueron agrupados en casos de uso[13] que describen la manera en que deben ser resueltos. A continuación se presentan un conjunto de tablas resumen de estos casos de uso cuya descripción completa puede ser tomada del referenciado trabajo del Ing. Carlos de Jesús Andrés González.

[13] *Artefacto definido por RUP que define un conjunto de instancias de guión de uso, donde cada instancia es una secuencia de acciones que lleva a cabo un sistema que producen un resultado observable de valor para un actor concreto.*

CU-1	Determinar Idioma de la Interfaz de Usuario.
Actor	Administrador de Configuración.
Descripción	Permitirá cambiar el idioma de la interfaz de usuario para que la plataforma pueda ser usada por personas que hablen diferentes idiomas.
Referencia	R1

Tabla 2: Descripción del CUS Determinar Idioma de la Interfaz de Usuario.

CU-2	Determinar Tipo de Autenticación.
Actor	Administrador de Configuración.
Descripción	Dará la posibilidad de establecer el tipo de autenticación a usar en la plataforma ya sea local o por dominio.
Referencia	R2

Tabla 3: Descripción del CUS Determinar Tipo de Autenticación.

CU-3	Gestionar Roles de Usuarios.
Actor	Administrador de Configuración.
Descripción	Permitirá crear, modificar, eliminar y listar los roles de los usuarios de la plataforma, indicando las funcionalidades a las que puede tener acceso.
Referencia	R3, R3.1, R3.2, R3.3, R3.4

Tabla 4: Descripción del CUS Gestionar Roles de Usuario.

CU-4	Determinar Tipo de Transmisión.
Actor	Administrador de Configuración.
Descripción	Permitirá establecer el tipo de transmisión que realice la plataforma, ya sea por, secciones temáticas, bloques noticiosos o transmisión continua.
Referencia	R4

Tabla 5: Descripción del CUS Determinar Tipo de Transmisión.

CU-5	Gestionar Secciones Temáticas.
Actor	Administrador de Configuración.
Descripción	Permitirá que se pueda crear, modificar, eliminar y listar las secciones temáticas que se utilizarán para la trasmisión de las noticias.
Referencia	R5, R5.1, R5.2, R5.3, R5.4

Tabla 6: Descripción del CUS Gestionar Secciones Temáticas.

CU-6	Gestionar Bloques Noticiosos
Actor	Administrador de Configuración.
Descripción	Permitirá que se pueda crear, modificar, eliminar y listar los bloques de noticias que se utilizarán para la trasmisión de las noticias.
Referencia	R6, R6.1, R6.2, R6.3, R6.4

Tabla 7: Descripción del CUS Gestionar Bloques Noticiosos.

CU-7	Gestionar Canales Externos.
Actor	Administrador de Configuración.
Descripción	Permitirá que se pueda crear, modificar, eliminar y listar los canales de televisión externos con los que se puede enlazar la plataforma.
Referencia	R7, R7.1, R7.2, R7.3, R7.4

Tabla 8: Descripción del CUS Gestionar Canales Externos.

CU-8	Configurar Elementos de Información.
Actor	Administrador de Configuración.
Descripción	Permitirá establecer los servicios web que serán usados para obtener la información mostrada por los elementos de información adicional del canal.
Referencia	R8

Tabla 9: Descripción del CUS Configurar Elementos de Información.

CU-9	Gestionar Fuentes Web.
Actor	Administrador de Configuración.
Descripción	Permitirá adicionar, eliminar y listar los sitios web que servirán de agentes de información para el lector de noticias.
Referencia	R9, R9.1, R9.2, R9.3

Tabla 10: Descripción del CUS Gestionar Fuentes Web.

CU-10	Habilitar/Deshabilitar Módulos y Funcionalidades.
Actor	Administrador de Configuración.
Descripción	Permitirá que se puedan ocultar o visualizar los módulos y funcionalidades de la plataforma.
Referencia	R10

Tabla 11: Descripción del CUS Habilitar/Deshabilitar Módulos y Funcionalidades.

CU-11	Configurar Reportes.
Actor	Administrador de Configuración.
Descripción	Permitirá que se puedan ocultar o visualizar los reportes de la plataforma así como configurar el formato de salida de los mismos.
Referencia	R11

Tabla 12: Descripción del CUS Configurar Reportes.

Requisitos No Funcionales

Mientras que un requisito funcional especifica acciones que debe poder realizar un sistema sin tener en cuenta sus características físicas, un requisito no funcional describe solamente atributos, propiedades o cualidades que deben presentar el sistema y/o el entorno en que debe ser desplegado. Esto se traduce en que los requisitos no funcionales dicen cómo el sistema debe lucir, cuán rápido debe ser, cuán amigable, usable y otro conjunto de características que se clasifican en múltiples categorías. El Ing. Carlos de Jesús Andrés González refiere en su trabajo de diploma que en muchos casos, el éxito

comercial del producto informático está en gran medida relacionado con los requisitos no funcionales. Precisamente del trabajo antes mencionado es que son extraídos los que rigen la implementación del Subsistema de Configuración:

Requerimientos de apariencia o interfaz externa: La apariencia del módulo debe estar restringida por las mismas condiciones que PRIMICIA, haciendo uso de colores apropiados que denoten un producto profesional. El módulo debe contar con una interfaz amigable, intuitiva, interactiva y simple de usar.

Requerimientos de usabilidad: El sistema de forma general debe brindar gran facilidad de uso para personas con poca experiencia con las computadoras, pero con nivel calificado. Para trabajar con el Subsistema de Configuración se requiere de conocimientos mínimos en informática, televisión y uso de la web, ya que a través de este se gestiona el funcionamiento de toda la plataforma. Las funcionalidades deben ser claras y se debe mostrar la información de forma lógica y correctamente estructurada. El nuevo subsistema debe permitir una mejor configuración del sistema adaptándolo a las necesidades de la entidad donde se implante, y el cual reduzca los costos en tiempo y complejidad para estas operaciones con relación a la actualidad. Se requiere que pueda ser usado por personas que hablen diferentes idiomas.

Requerimientos de confiabilidad: Se debe garantizar un tratamiento adecuado de las excepciones y validación de las entradas del usuario.

Requerimientos de portabilidad: El sistema estará realizado para funcionar en los sistemas operativos GNU/Linux o Windows, garantizando de esta manera un producto multiplataforma.

Requerimientos de seguridad: La información manejada por el sistema y el uso del mismo estará protegida del acceso de personas no autorizadas. Solo se accederá a las opciones autorizadas según el rol que desempeñe. El Subsistema de Configuración solo debe ser accedido por un súper administrador. Solo se accederá al sistema de administración desde las computadoras autorizadas. Los usuarios autorizados tendrán garantizado el acceso pleno a la información, de acuerdo a los permisos que tengan establecidos.

La información manejada por el sistema será objeto de cuidadosa protección contra la corrupción o estado inconsistente. Deberán existir mecanismos de chequeo de integridad. Se deberán hacer copias de respaldo que puedan restaurar el sistema en caso de fallo crítico o pérdida total de la información. El sistema debe estar disponible los 7 días de la semana y las 24 horas del día, garantizando el buen funcionamiento de la Plataforma. De esta manera se debe garantizar que las modificaciones realizadas en la configuración de la plataforma se hagan en tiempo real y que no afecten en ningún sentido el correcto funcionamiento de la misma.

Requerimientos Legales: El Subsistema de Configuración estará sujeto a las mismas restricciones legales que el resto de la Plataforma de Televisión Informativa, PRIMICIA. Sus derechos de autor y otros están determinados por la empresa comercializadora del producto, ALBET S.A y la entidad desarrolladora, la UCI.

Requerimientos de Ayudas y Documentación en línea: El sistema debe presentar un manual de usuario donde indique al usuario los pasos que tiene que seguir para obtener una correcta configuración.

Requerimientos de Software: Se debe utilizar Apache como servidor web y PostgresSQL 8.4 o superior como sistema gestor de base de datos. El Subsistema de Administración y de Configuración deben ser accedidos a través del cualquier navegador, fundamentalmente Internet Explorer 7 o superior, o Mozilla Firefox.

Requerimientos de Hardware: Serán los mismos que para el servidor donde corra el Subsistema de Administración de la plataforma, el subsistema responde a un elemento que se debe integrar a dicho sistema, los requerimientos son los siguientes.

- Procesador: Dual-Core Xeon 2.33 GHz.
- Memoria RAM: 4Gb.
- Disco Duro: 500Gb.
- Tarjeta de Red: Ethernet Gigabit con 2 puertos.
- Tarjeta de Video: Capturadora Hauppauge WinTV PVR 350.

Restricciones en el diseño y la implementación: Para la modelación del sistema se utilizará el lenguaje UML y a través de la herramienta Visual Paradigm. Se requiere el uso de la arquitectura Modelo-Vista-Controlador implementada por el framework Symfony de PHP. La arquitectura debe soportar migrar la interfaz de usuario de forma rápida, para lograr visualizar cualquiera de los cambios que se produzcan.

2.3. Diagramas y herramientas generadoras de código.

Diagramas de Componentes

Los casos de uso anteriormente resumidos, describen de forma general las funcionalidades que los implementadores deben desarrollar. En este epígrafe se exponen un conjunto de diagramas que expresan visualmente, la relación física que existe entre los componentes[14] que son necesarios para la programación de los mencionados casos de uso. Estos diagramas son parte del artefacto de RUP Modelo de Implementación y ofrecen una vista de la estructuras de los componentes desarrollados y por desarrollar. Los componentes pueden ser ficheros, ejecutables, librerías, tablas de la base de datos o documentos. En este tipo de diagramas intervienen también los paquetes, que son una forma de organización o agrupamiento de diferentes archivos o clases que el desarrollador deberá usar.

[14] *Parte del sistema sustituible, casi independiente e importante que desempeña una función clara en el contexto de una arquitectura bien definida* **(13)**.

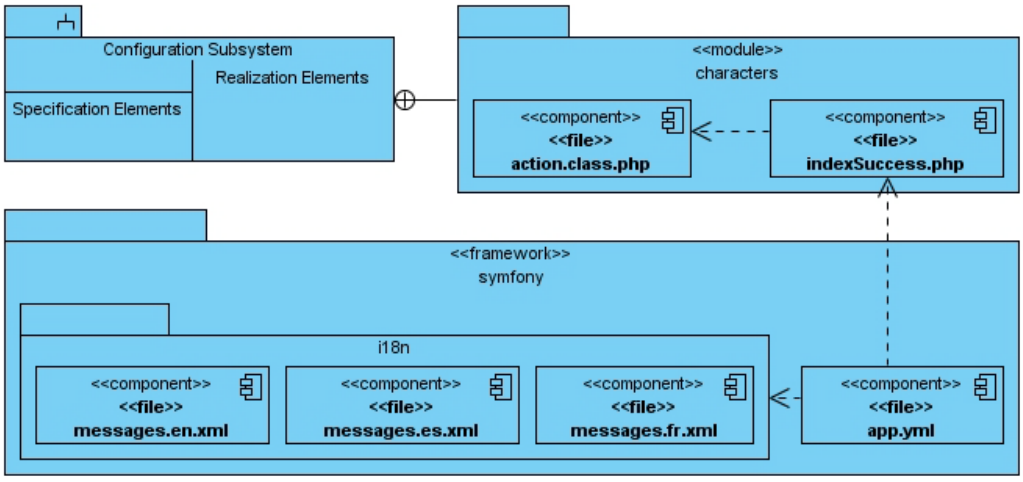

Figura 4: Diagrama de Componentes CU-1

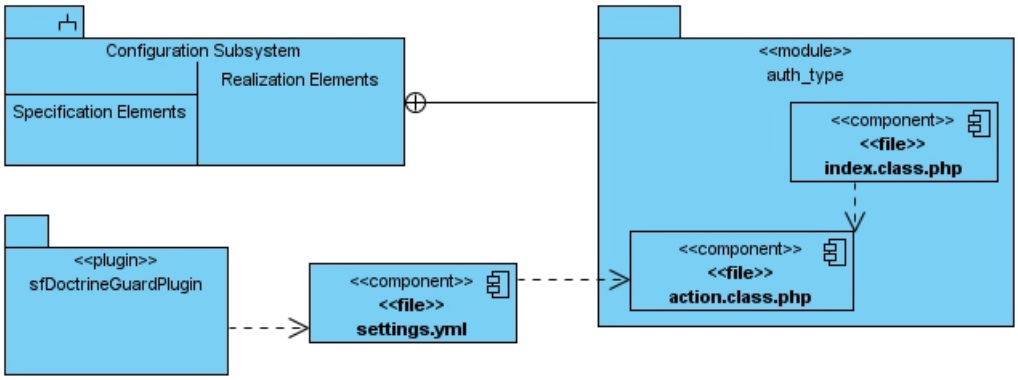

Figura 5: Diagrama de Componentes CU-2

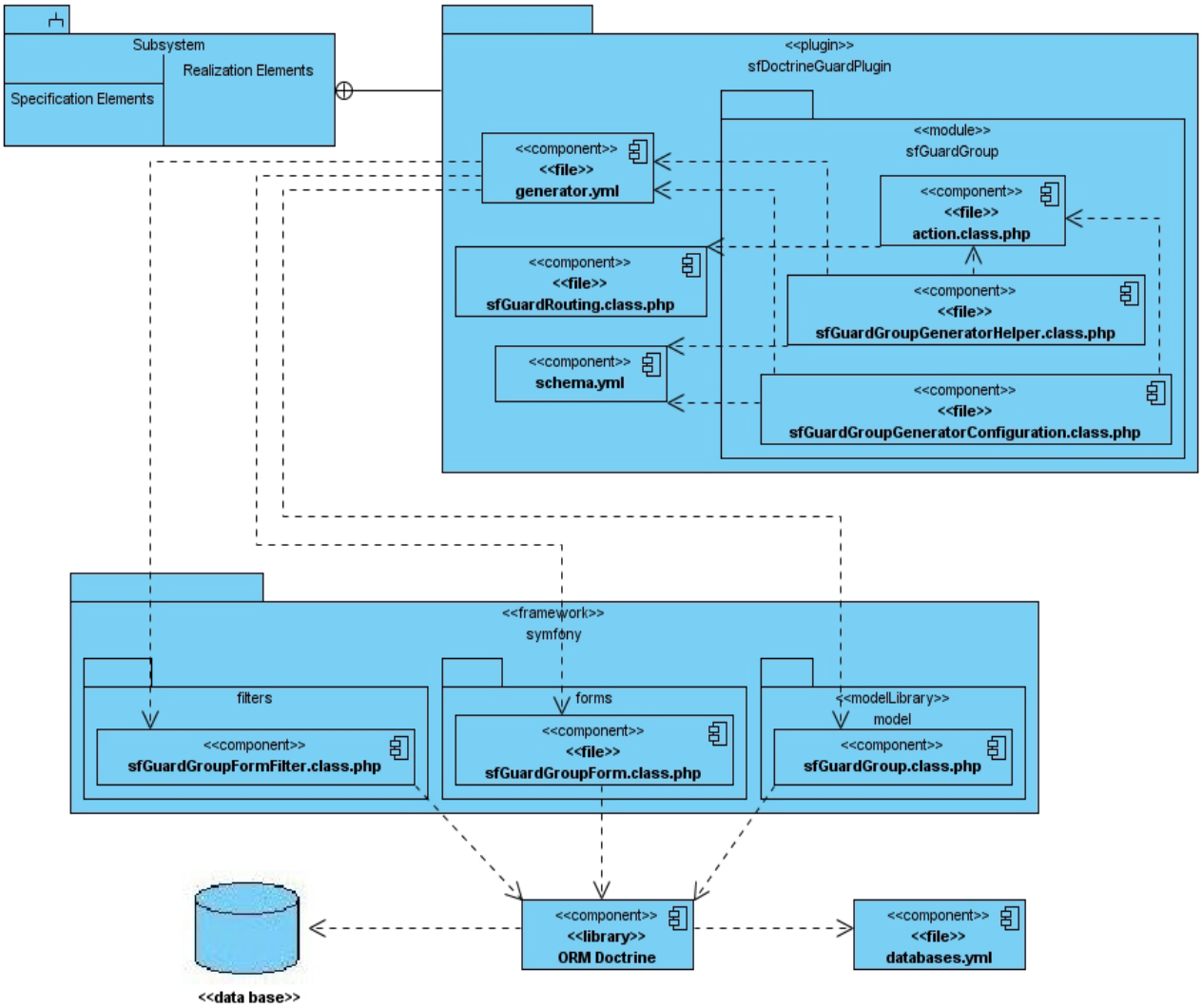

Figura 6: Diagrama de Componentes CU-3

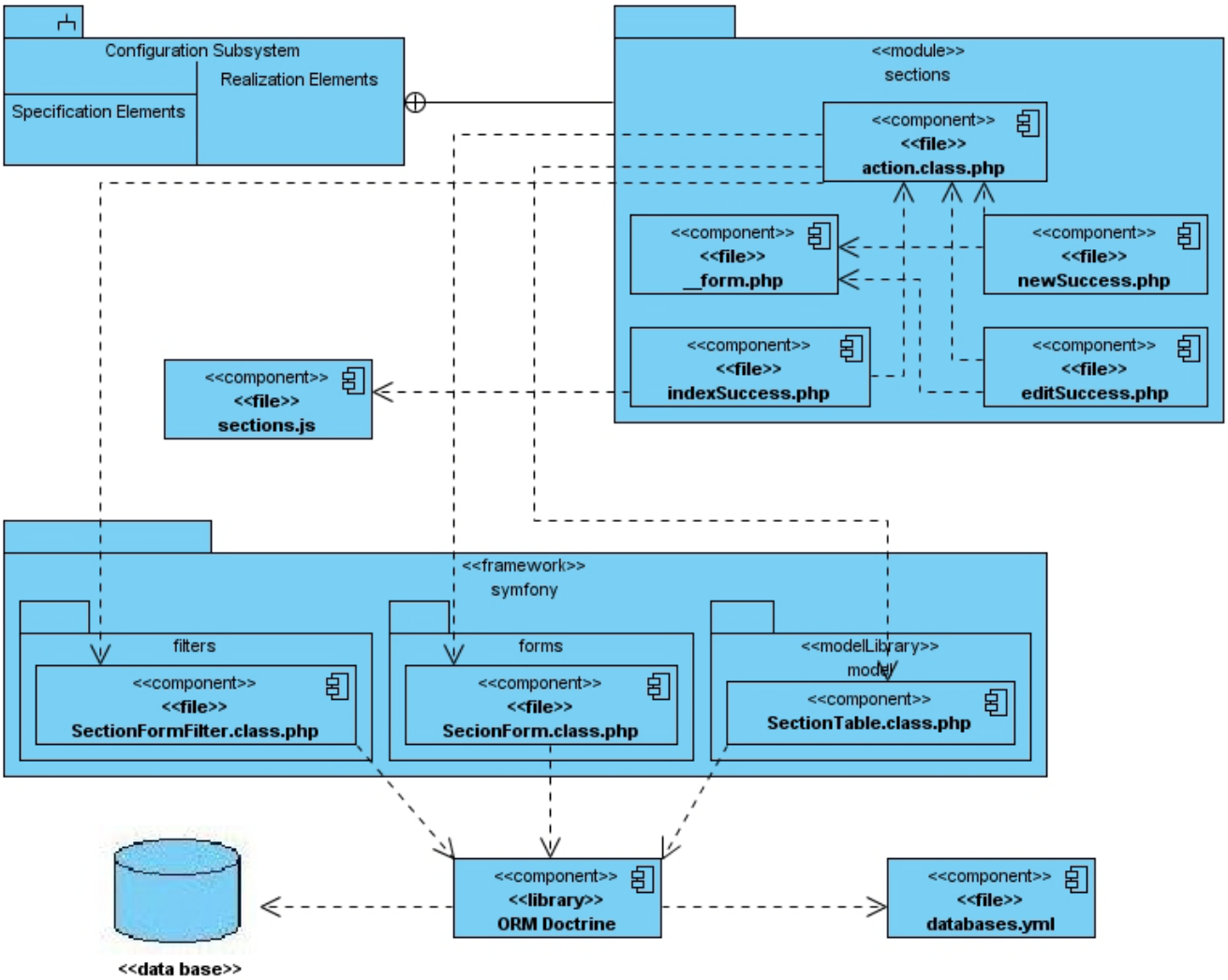

Figura 7: Diagrama de Componentes CU-5

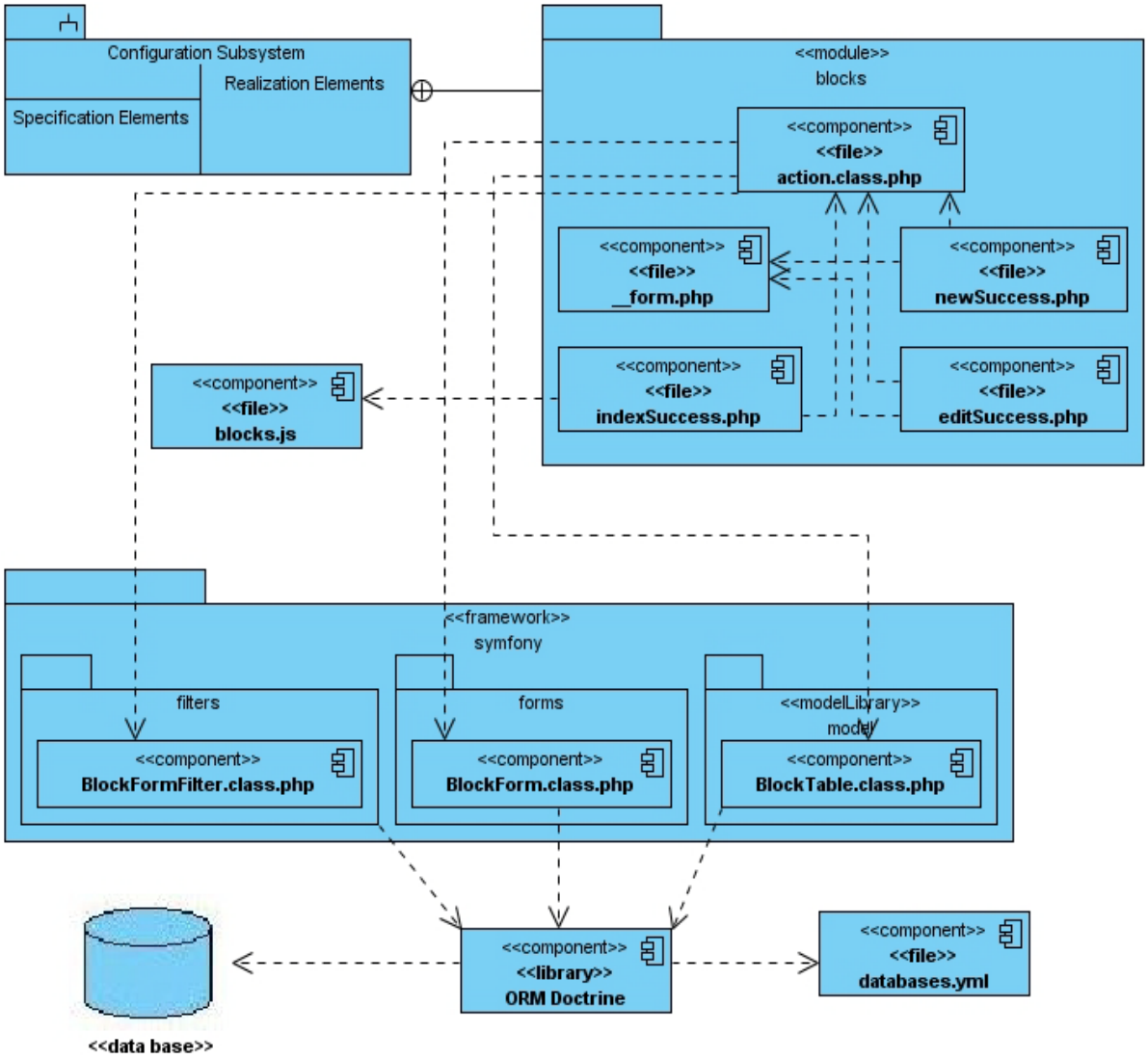

Figura 8: Diagrama de Componentes CU-6

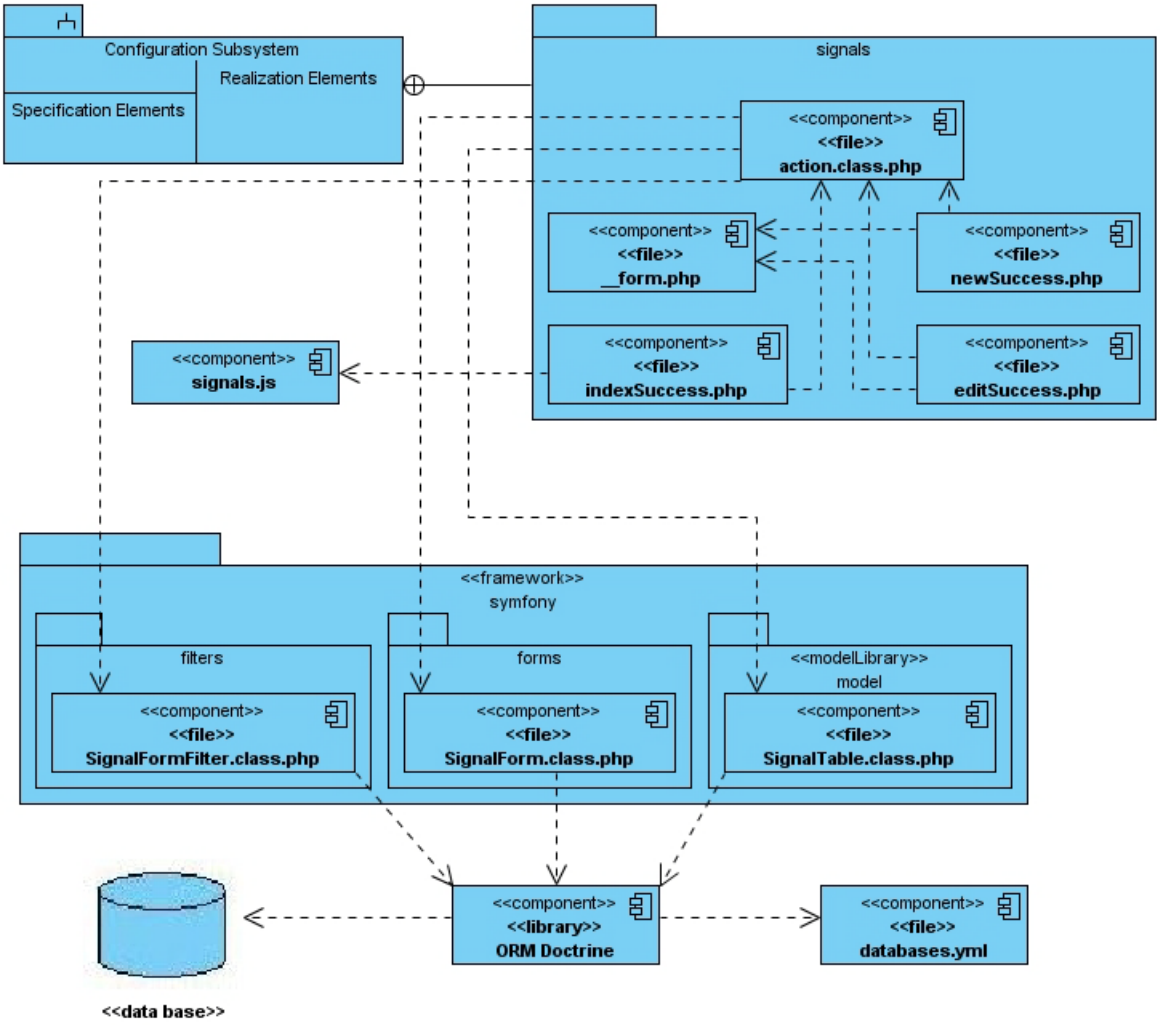

Figura 9: Diagrama de Componentes CU-7

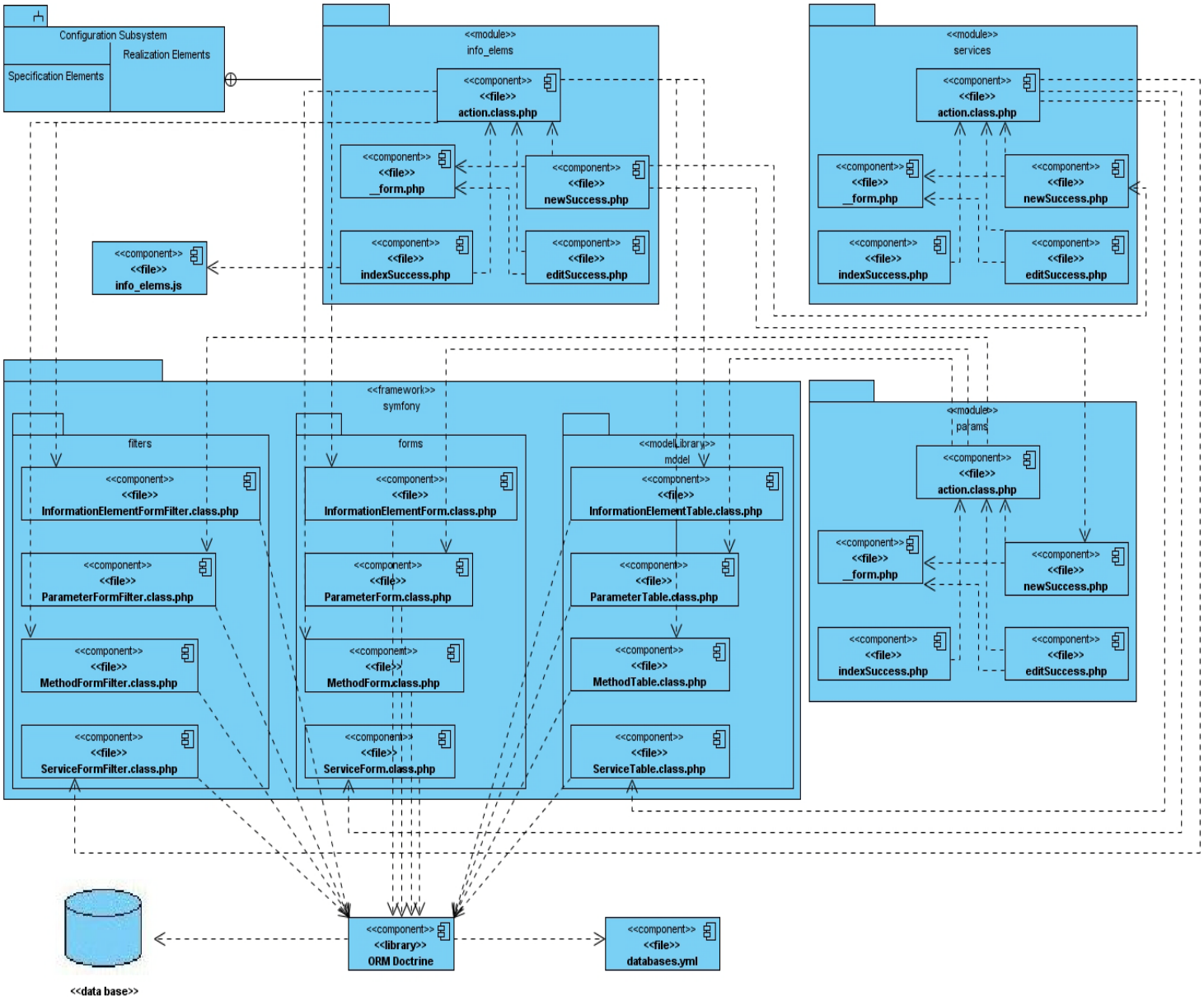

Figura 10: Diagrama de Componentes CU-8

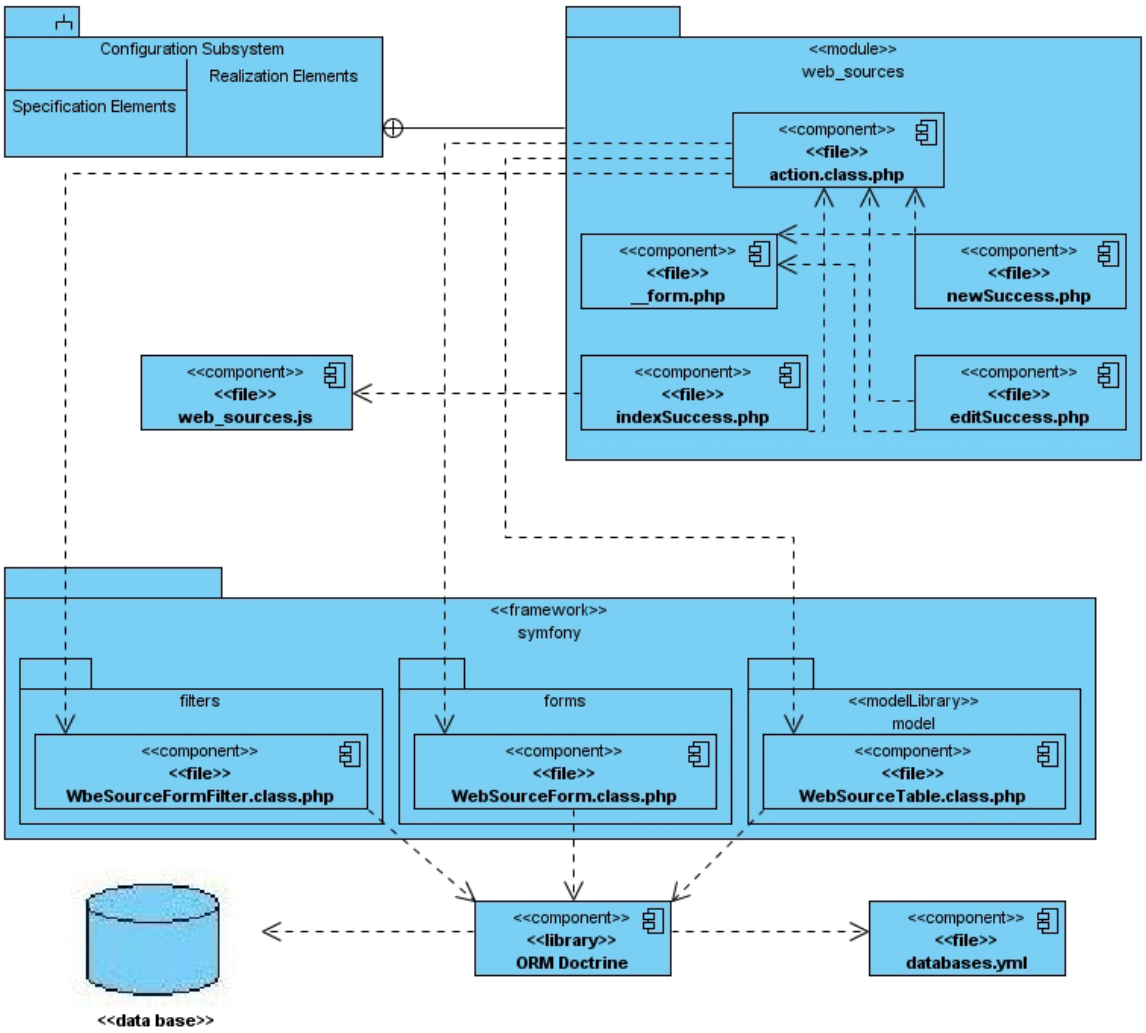

Figura 11: Diagrama de Componentes CU-9

Diagrama de Despliegue

El objetivo del diagrama de despliegue *"es capturar la descomposición del sistema en elementos que alojan el proceso"* (6)**.** Este diagrama permite realizar un esquema visual de la distribución física de los nodos y de los ficheros en los nodos. El siguiente diagrama ofrece una eficaz visualización de la plataforma PRIMICIA y el Subsistema de Configuración dentro de ella.

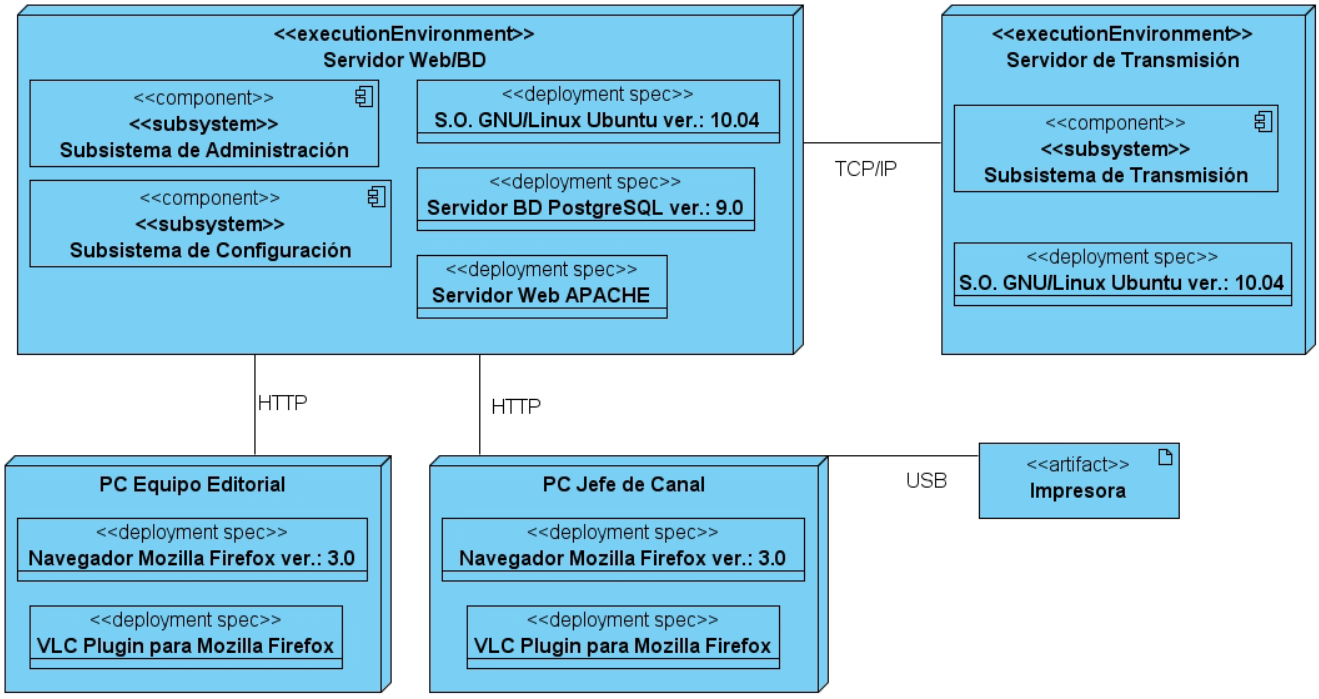

Figura 12: Diagrama de Despliegue de PRIMICIA.

Herramientas generadoras de código

Symfony emplea un sistema generador de elementos de ficheros y código que facilita en una gran medida el trabajo de los implementadores. Este sistema se activa a través de comandos que permiten la personalización de la generación. Para la implementación de varios casos de uso, en su mayoría funcionalidades que cumplen con las especificaciones del patrón GRASP CRUD, se emplea el comando:

```
doctrine:generate-module <nombre_aplicación> <nombre_módulo> <nombre_clase_modelo>
```

Este comando genera un módulo de administración de una tabla de la base de datos, empleando las clases del modelo generadas por otro comando del propio ORM Doctrine. El módulo que se genera respeta la estructura de carpetas usada por el *framework* Symfony y contiene cinco archivos con extensión *.php* que permiten listar, crear, modificar y eliminar elementos de la tabla que se usó para generar el módulo. Aunque facilita el trabajo, el programador ha de intervenir para personalizar esta

codificación estándar e incluir filtros y paginadores para los listados. También es necesario verificar y personalizar el procesamiento de los datos que se envían usando los formularios que también genera Symfony.

Con el mismo objetivo de ahorrar trabajo al implementador, se emplean *plugins* desarrollados por los equipos de trabajo de SensioLabs, empresa que se dedica al desarrollo del *framework* Symfony, los cuales tienen una excelente integración con el mencionado marco de trabajo. Ejemplo de estos *plugins* es el sfDoctrineGuardPlugin que se emplea para la seguridad de la aplicación y la administración de usuarios, roles y permisos o credenciales. Estos *plugins* se caracterizarán en epígrafes posteriores.

2.4. Actualizaciones al diseño

A continuación se describen algunos cambios realizados a los casos de uso propuestos por el Ing. Carlos de Jesús Andrés González en su trabajo “Diseño del Subsistema de Configuración de la Plataforma de Televisión Informativa, PRIMICIA”.

Caso de uso Determinar Idioma de la Interfaz de Usuario

En el referido trabajo se consideró que las acciones para cambiar el idioma de la interfaz de usuario de la aplicación, se restringirían solamente al Administrador de Configuración, rol que se le propone al usuario que poseerá los privilegios para realizar cambios a la aplicación. Para la implementación del Subsistema de Configuración, sin embargo, se permitirá a cualquier usuario de la aplicación cambiar el idioma de su interfaz, aunque se mantiene también el flujo de los eventos descritos en el caso de uso Determinar Idioma de la Interfaz de Usuario del trabajo del Ing. Carlos de Jesús Andrés González.

Caso de uso Determinar Tipo de Transmisión

En la primera versión desarrollada de la Plataforma PRIMICIA, la transmisión de las noticias se realizaba organizada por la sección temática a que pertenecían estas. Durante la concepción del Subsistema de Configuración, el Ing. Carlos de Jesús Andrés González introdujo para este sistema, los bloques noticiosos como una nueva forma de organizar la transmisión de las noticias.

Las secciones temáticas agrupan todas las noticias que tengan que ver con el tema de la sección; ej.: la sección “Deportivas” agrupa las noticias de ese corte que se redacten y de esa misma forma con cualquier otro tema. Esto provoca que las noticias siempre estén asociadas a una sección y que la transmisión de esas noticias sea de forma coherente, casi siempre con una pequeña presentación para cada sección. Los bloques noticiosos por su parte, se componen de secciones temáticas y noticias libres indistintamente. Estos bloques son más fáciles de manipular y programar; ej.: el bloque “Mañana” puede estar programado para transmitirse desde las 8:00 a.m. a las 12:00 m., y compuesto por las secciones “Deportivas”, “Nacionales” y “Económicas” más las noticias que se escojan de otras secciones que no se incluirán completamente en el bloque. En esta forma de organización, las secciones influyen en la forma de agrupar las noticias en el momento de transmitirlas.

El caso de uso Determinar Tipo de Transmisión, se diseñó con el propósito de permitir al Administrador de Configuración elegir entre una transmisión libre, una por secciones temáticas y una por bloques noticiosos. Por ser esta última una forma mucho más general pues brindará la posibilidad de construir escaletas de transmisión donde se define la duración, el momento y las noticias o secciones que se muestren; imprime a la programación de la transmisión una mayor flexibilidad y ya no es necesaria la funcionalidad definida en el caso de uso analizado.

Caso de uso Gestionar Roles de Usuario

En la versión actual de PRIMICIA, uno de los más graves problemas es la imposibilidad de gestionar a conveniencia los roles de usuario. Para la implementación del diseño propuesto en el trabajo del Ing. Carlos de Jesús Andrés González, se realizó el estudio de un *plugin* desarrollado en los laboratorios de SensioLabs para Symfony.

El sfDoctrineGuardPlugin brinda una eficiente gestión de la seguridad de la aplicación. Además provee de las funcionalidades necesarias para la autenticación segura y la asignación de responsabilidades o credenciales. Usando las propias herramientas del *framework* este *plugin* facilita la búsqueda, incorporación, edición y supresión de nuevos elementos usuario, permisos y roles que agrupan los permisos. Todas estas funcionalidades se acomodan directamente a lo diseñado por el Ing. Carlos de Jesús Andrés González en su trabajo de diseño. Puesto que las facilidades de este *plugin* se extienden

a otras áreas también necesarias de la aplicación, como el manejo de la autenticación y aseguramiento de páginas, se decide el empleo de este para la implementación del caso de uso analizado.

Caso de uso Habilitar/Deshabilitar Funcionalidades

En una aplicación desarrollada con Symfony, el acceso a las acciones o funcionalidades implementadas en cada módulo de esta, se maneja con la comprobación de credenciales definidas de antemano en un archivo de configuración de la aplicación denominado *security.yml*. Haciendo coincidir las credenciales que se definan en la aplicación con los permisos gestionados a través del sfDoctrineGuardPlugin, y logrando gestionar adecuadamente estos permisos, se resuelve el problema de habilitar o deshabilitar las funcionalidades para usuarios de la aplicación, que se corresponden precisamente con las acciones implementadas en los módulos. Por esta razón y puesto que lo explicado en esta sección se acomoda igualmente a lo diseñado por el Ing. Carlos de Jesús Andrés González, si se implementa como se propone el caso de uso Gestionar Roles de Usuario, se da respuesta al analizado en esta sección y se elimina su programación restando de esta manera tiempo a la codificación.

Nueva funcionalidad: Definir Cantidad de Caracteres

La transmisión de un canal de televisión puede realizarse en diferentes formatos que poseen así mismo diferentes propiedades. En los términos del Subsistema de Transmisión con que cuenta la plataforma PRIMICIA, esto quiere decir que en dependencia del formato de televisión que se emplee, variará considerablemente la cantidad de caracteres que puedan utilizarse para la confección de las noticias que se mostrarán a través de este.

Las noticias cuentan con varios elementos de texto a los que se hace necesario permitir su configuración, para cumplir con las exigencias de las normas de televisión a las que se hacía referencia anteriormente. Estos elementos son: título de la noticia, texto de las noticias de tipo texto, texto de las noticias de tipo texto-imagen y pie de página de las noticias de tipo imagen. Esto sucede también con la cantidad de texto de los cintillos informativos que se publiquen. Para la correcta configuración de estas variables es necesario brindarle la posibilidad al Administrador de Configuración, de suplir al sistema los valores necesarios para que se ajuste con la norma de televisión que se empleara en la

transmisión. Así se propone la funcionalidad Definir Cantidad de Caracteres que permitirá dicha configuración y se le define el siguiente diagrama de componentes:

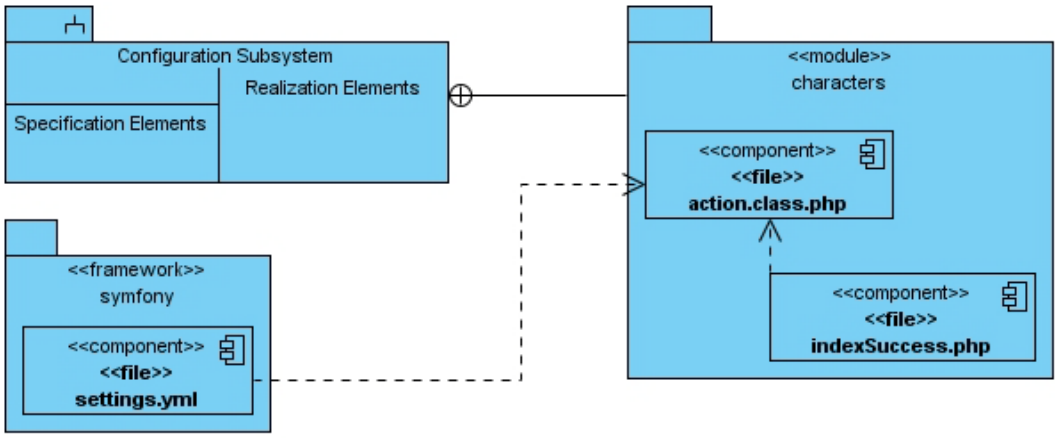

Figura 13: Definir Cantidad de Caracteres

2.5. Conclusiones Parciales

En este capítulo se expusieron los diagramas de componentes de los casos de uso del Subsistema de Configuración Estos diagramas son una gran herramientas del rol de Implementador definido por RUP, pues muestran los archivos de la aplicación luego de implementados así como sus dependencias de tiempo de ejecución. También se propuso el diagrama de despliegue, una vista general de la ubicación física de los subsistemas de PRIMICIA, y se plantearon un conjunto de modificaciones y actualizaciones al diseño realizado previamente por el Ing. Carlos de Jesús Andrés González, como consecuencia del estudio de las ventajas que brindan las herramientas seleccionadas para el desarrollo.

CAPÍTULO 3. Pruebas y validación de la solución.

3.1. Introducción.

Las pruebas son una parte importante del proceso de puesta en ejecución de una aplicación informática. En este capítulo se ofrecerá un rodeo sobre los principales tipos de pruebas y se especificarán las que se le aplican a la solución propuesta. También se ofrecerán elementos que permitirán validar esta solución y concluir si cumple con los requisitos detallados para ella.

3.2. Automatización y tipos de pruebas.

Para asegurar la validación de una aplicación informática e incluso la satisfacción del cliente que desee adquirirla, es muy importante la realización de pruebas concluyentes. Estas pruebas deben ser capaces de ofrecer información que permita a los desarrolladores y probadores decidir si la aplicación es funcional y si cumple con los requerimientos y el propósito para el que fue construida. De hecho, las pruebas que se realizan actualmente, ofrecen además parámetros que permiten analizar si la puesta en ejecución de una aplicación es viable en términos de rendimiento, soporte de carga y estrés, entre otros. El Proceso Unificado de Desarrollo, define una tarea importante para el rol de Implementador; esta es “Implementar las pruebas del desarrollador” y es definida como:

“En esta tarea se describe cómo crear un conjunto de pruebas para comprobar que el componente funciona correctamente antes de que se realicen más pruebas formales en él.” **(6)**

La definición antes dada indica que las pruebas del desarrollador, otro nombre para el implementador, pueden ser tan extensas como este quiera pero aplicada a todos los componentes de software que desarrolle. Estas pruebas serán tomadas luego para detallar experimentos más profundos antes de la liberación oficial del software desarrollado, como se especifica en sus objetivos:

- *“Implementar una o varias pruebas que permitan la validación de los componentes de software individuales mediante la ejecución física.*

- *Desarrollar pruebas que se puedan ejecutar conjuntamente con otras pruebas como parte de una infraestructura de prueba mayor."* **(6)**

Precisamente RUP define un nombre para las pruebas que realizan los desarrolladores, este es "Prueba de Unidades":

"La prueba de unidades es una prueba detallada de la estructura interna de un módulo de software." **(6)**

Un módulo o componente de software puede ser probado, según RUP, adoptando dos enfoques diferentes: de caja negra y de caja blanca.

"Un enfoque de prueba de caja blanca debe adoptarse para verificar la estructura interna de una unidad. Teóricamente, debe probar todas las vías de acceso a través del código, pero esto sólo es posible en unidades muy simples.

El objetivo de una prueba de caja negra es verificar la función y el comportamiento observable especificado de la unidad sin conocer cómo implementa la unidad la función y el comportamiento. Las pruebas de caja negra se centran y se basan en la entrada y la salida de la unidad." **(6)**

Estos dos enfoques son los más utilizados a nivel mundial por empresas desarrolladoras de software. Escribir casos de prueba (que son las descripciones de las pruebas a desarrollar), ejecutarlos y analizar los resultados es un trabajo bastante tedioso. Es por eso que numerosas empresas se han dedicado a la tarea de automatizar el proceso de prueba y hoy en día existen numerosos *frameworks* dedicados a esto entre los que destacan PhpUnit y SimpleTest. Symfony por su parte, integra su propio *framework* para pruebas llamado Lime el cual *"es basado en la librería de Perl* ***Test::More*** *y es compatible con TAP, lo que significa que el resultado de las pruebas es mostrado como se especifica en el Test Anything Protocol (TAP o Protocolo Prueba Cualquier Cosa en español), diseñado para una mejor lectura de las salidas de las pruebas"* **(14)**. Para Symfony, el enfoque de pruebas de caja blanca es llamado "prueba unitaria" y el de caja negra "prueba funcional".

3.3. Pruebas unitarias

"Las pruebas unitarias confirman que un componente de código unitario provee la salida correcta para una entrada dada" **(14)**. Lo anterior significa que estas pruebas validan cómo las funciones y métodos trabajan en cada caso particular, comprueban que las unidades de código escritas devuelvan el resultado esperado para una entrada de datos específica; o sea, para datos correctos, resultados correctos, para datos incorrectos, error. El mecanismo que Symfony utiliza para escribir e implementar casos para pruebas unitarias es simple. Las pruebas unitarias son archivos PHP cuyos nombres terminan en *Test.php* y se encuentran en el directorio *test/unit* del proyecto. A través de varios métodos de la clase *lime_test* es fácil escribir en código PHP las llamadas a las funcionalidades a probar, evidenciando con preguntas lógicas de respuestas *verdadero* o *falso* que el método trabaja como se espera.

Como se ha expuesto, el Subsistema de Configuración de PRIMICIA es una aplicación web de gestión, es decir, que trabaja mayormente sobre objetos generados desde las tablas de la base de datos que maneja. Symfony es capaz de generar automáticamente a través del comando `doctrine:generate-module <nombre_aplicación> <nombre_módulo> <nombre_clase_modelo>`, un módulo con todo lo necesario para realizar precisamente operaciones de gestión (listar, crear, modificar y eliminar) sobre los campos de una tabla de la base de datos. El comando antes mencionado fue ampliamente utilizado en la implementación del subsistema lo cual significa que gran parte del código escrito, proviene directamente de las factorías de los laboratorios SensioLabs, en donde ha sido ampliamente probado y verificado su funcionalidad. Por otro lado, los módulos *auth_type* y *global* poseen código casi puramente escrito por los desarrolladores del subsistema, específicamente, dentro de los archivos *actions.class.php* de sus respectivos módulos. En estos se encuentran dos funciones a las que se hace necesaria la aplicación de pruebas unitarias.

Caso de prueba para la función *executeModify* de la clase *auth_typeActions*, archivo *actions.class.php*:

Para realizarle pruebas a la acción en un ambiente controlado, fue necesario crear en el archivo *auth_typeTest.php*, una clase *prueba* que contiene el método *executeModify* tal y como se presenta en

la clase *auth_typeActions*. Luego para ejecutar la prueba se definieron los parámetros que se le introducen al método.

La función *executeModify* de las acciones del módulo *auth_type*, es la encargada de recibir y procesar la petición de cambiar y aplicar al sistema, una autenticación por base de datos local o por dominio. La petición que se le envía a esta acción es generada por la clase *sfWebRequest* de Symfony. Esta contiene la información que necesita la función, para cambiar en los archivos *app.yml* de las aplicaciones correspondientes a Administración y Configuración (*frontend* y *backend*), todo lo relacionado con el tipo de autenticación. Ella también tiene unas líneas dedicadas a cambiar la cultura en la que el usuario (manejado por Symfony a través de la clase *sfUser*) está visualizando o desea visualizar la interfaz de la aplicación.

Las siguientes figuras muestran dos salidas de la prueba, una realizada con datos de entrada correctos y otra realizada con datos de entrada incorrectos. Estas pruebas se ejecutaron en la consola del IDE Netbeans v6.9 empleando el comando de Symfony `test:unit auth_type` que permite correr las pruebas unitarias guardadas en el directorio *test/unit* del proyecto.

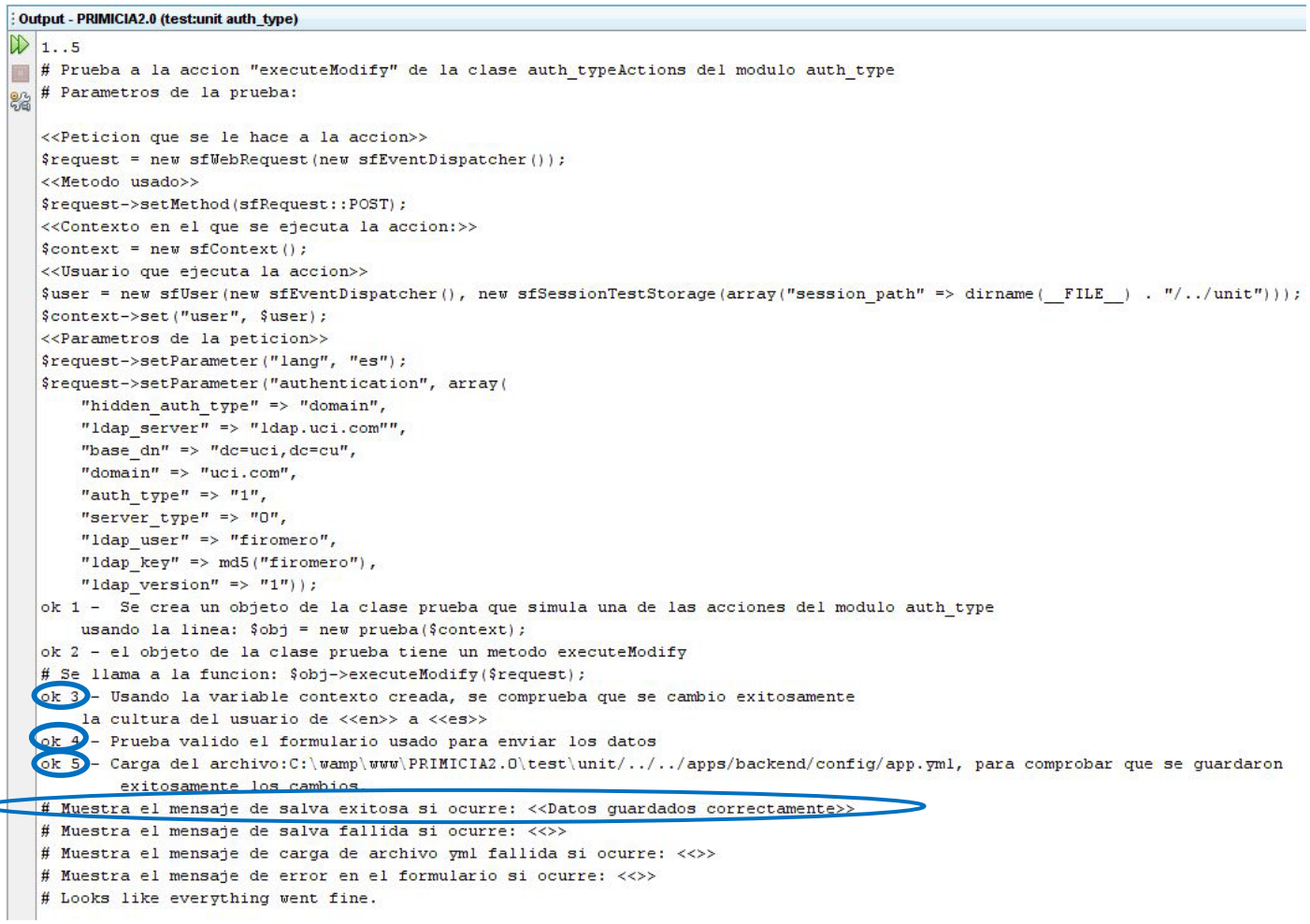

Figura 14: Prueba realizada al método *executeModify* exitosa

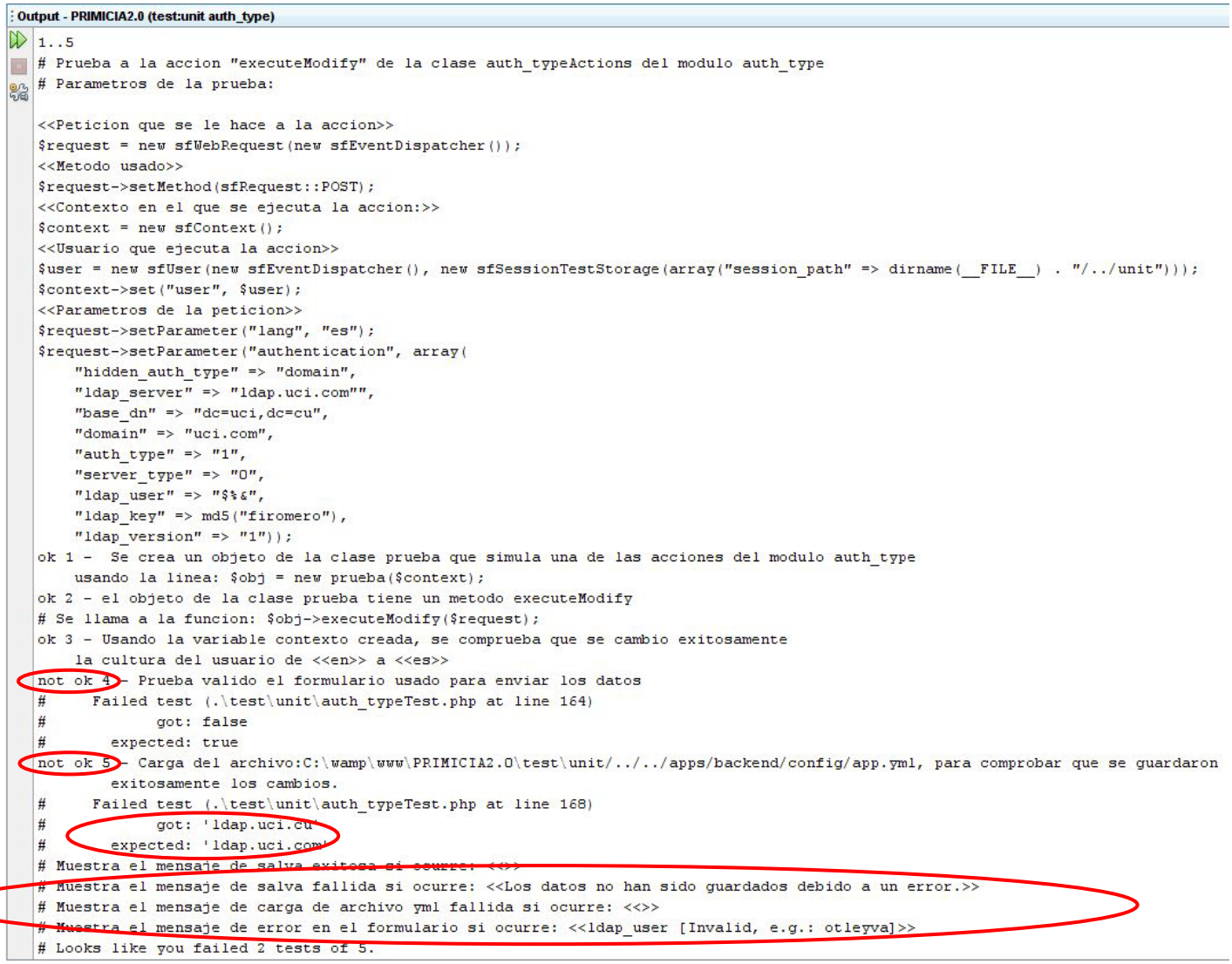

Figura 15: Prueba realizada al método *executeModify* fallida

MÉTODOS (PREGUNTAS)	DESCRIPCIÓN
`diag($mensaje)`	Muestra un mensaje de salida en la consola, pero no se cuenta como prueba.
`ok($prueba[, $mensaje])`	Prueba una condición y pasa si es verdadera.
`is($valor1, $valor2[, $mensaje])`	Compara dos valores y pasa si ambos son iguales (==).
`isnt($valor1, $valor2[, $mensaje])`	Compara dos valores y pasa si no son iguales.
`like($cadena, $expreg[, $mensaje])`	Prueba una cadena contra una expresión regular.
`unlike($cadena, $expreg[, $mensaje])`	Chequea que una cadena no pasa contra una expresión regular.
`cmp_ok($valor1, $operador, $valor2[,mensaje])`	Compara dos argumentos con un operador.
`isa_ok($variable, $type[, $mensaje])`	Chequea el tipo de un argumento.
`isa_ok($objeto, $clase[, $mensaje])`	Chequea la clase de un objeto.
`can_ok($objeto, $metodo[, $mensaje])`	Chequea la disponibilidad de un método para un objeto o clase.
`is_deeply($arreglo1, $arreglo2[, $mensaje])`	Chequea que dos objetos tengan los mismos valores.
`include_ok($file[, $mensaje])`	Valida que un archivo existe y que está debidamente incluido.
`fail([$mensaje])`	Falla siempre (útil para probar excepciones).
`pass([$mensaje])`	Pasa siempre (útil para probar excepciones).

Tabla 13: Métodos más comunes de un objeto de *lime_test* (14)

La tabla 13 muestra los métodos más comunes con los que cuenta el objeto de *lime_test* para probar funcionalidades. Para el caso de prueba que se analiza, se emplearon los métodos `diag`, `isa_ok`, `can_ok` e `is`. En las figuras 14 y 15 están señaladas las operaciones más importantes y sus correspondientes resultados.

3.4. Pruebas funcionales

"Las pruebas funcionales validan 'partes' de las aplicaciones. Ellas simulan una sesión de navegación, haciendo peticiones y chequeando elementos en las respuestas exactamente como se haría manualmente para validar que una acción hace lo que se supone que debe hacer" **(14)**. Las pruebas funcionales no tienen interés en comprobar cómo trabaja una unidad de código, sino en los resultados que devuelven varias unidades de código trabajando juntas. En estas pruebas se ejecutan los escenarios correspondientes a los casos de uso definidos.

Symfony trae también su mecanismo para ejecutar pruebas funcionales. El *framework* provee un objeto especial llamado *sfBrowser*, que actúa como un navegador conectado a una aplicación de Symfony sin necesidad de ejecutar un servidor o las tardanzas que impone el transporte HTTP. Igualmente se encuentra la clase *sfTestFunctional* diseñada especialmente para pruebas funcionales y que le añade al objeto *sfBrowser* algunos métodos de aserción o comprobación. Una prueba funcional sencilla escrita usando las posibilidades de Symfony, podría verse así:

```
$b = new sfTestFunctional(new sfBrowser());
$b->get('/foobar/edit/id/1');
$request  = $b->getRequest();
$context  = $b->getContext();
$response = $b->getResponse();

// Get access to the lime_test methods via the test() method
$b->test()->is($request->getParameter('id'), 1);
$b->test()->is($response->getStatuscode(), 200);
$b->test()->is($response->getHttpHeader('content-type'), 'text/
html;charset=utf-8');
$b->test()->like($response->getContent(), '/edit/');
```

Figura 16: Prueba funcional escrita usando Symfony (14)

La ejecución de esta en la consola, empleando el comando Symfony `test:functional backend ejemplo`, el cual ejecuta para la aplicación "backend" la prueba "ejemplo" que debe estar guardada en el directorio *test/functional/backend*; retornaría un resultado parecido a:

```
# get /foobar/edit/id/1
ok 1 - request parameter "id" is "1"
ok 2 - status code is "200"
ok 3 - response header "content-type" is "text/html"
ok 4 - response matches "/edit/"
1..4
```

Figura 17: Resultado de una prueba funcional escrita usando Symfony

Sin embargo, aún cuando podría ser provechoso ejecutar pruebas escritas en Symfony para probar aplicaciones escritas usando Symfony, existe una muy poderosa herramienta libre que agiliza mucho estos procesos pues permite tener interacción visual entre la prueba automática y la aplicación.

Selenium IDE "*consiste en un framework de pruebas escrito completamente en JavaScript. La principal ventaja del uso de esta herramienta es que permite realizar una serie de acciones en la página de la misma forma que las haría un usuario normal. La ventaja de Selenium sobre el objeto sfBrowser de Symfony es que Selenium es capaz de ejecutar todo el código JavaScript de la página, incluidas las interacciones creadas con Ajax*" **(15)**. Las pruebas usando Selenium se escriben usando HTML por lo que cada prueba representa una página HTML en la que se muestra una tabla con 3 columnas: comando, destino y valor. Para escribir estas pruebas (trabajo tedioso si se realiza manualmente), es muy conveniente usar la extensión de Selenium IDE para el navegador Mozilla Firefox, que permite grabar todas las acciones realizadas sobre una página abierta con dicho navegador y guardarlas como una prueba.

Caso de Prueba para caso de uso *Determinar Tipo de Autenticación*, escenario *EC 2.1: Guardar selección Autenticación por Dominio exitosamente*:

Los diseños de casos de pruebas para cada caso de uso generan una documentación muy extensa por lo que no serán plasmados en este trabajo, sino que se entregarán a la dirección del proyecto. Las siguientes tablas muestran cómo debería desarrollarse solo uno de los escenarios definidos en el caso de prueba para el caso de uso en cuestión y cuál sería la matriz de datos de entrada correcta:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC 2: Autenticación por dominio	EC 2.1: Guardar selección Autenticación por Dominio exitosamente.	El usuario selecciona el tipo de autenticación “dominio”, introduce los datos referentes a la autenticación por LDAP correctamente y pulsa el botón Salvar.	Subsistema de Configuración, Menú Seguridad, opción Autenticación, campo de elección única Tipo de Aut., se introducen los datos en los campos servidor LDAP, DN base, dominio, tipo de servidor, usuario LDAP, clave LDAP y versión, botón Salvar.

Tabla 14: Descripción de escenario 2.1 de Caso de Prueba

ID de Escenario	Var 1 (Tipo de Autenticación)	Var 2 (Servidor LDAP)	Var 3 (DN Base)	Var 4 (Tipo de Servidor)	Var 5 (Usuario LDAP)	Var 6 (Clave LDAP)	Var 7 (Versión LDAP)	Var 8 (Dominio)	Respuesta del Sistema	Resultado de la Prueba
EC 2.1	*V/domain*	*V/192.168.1.2 o ldap.uci.cu*	*V/dc=uci, dc=cu*	*V/ActiveDirectory*	*V/otleyva*	*V/Y3shu4 M3schi4ch&*	*V/3*	*V/uci.cu*	*El sistema muestra el mensaje de confirmación **“Datos guardados correcta-mente”.***	*Satisfac-torio.*

Tabla 15: Matriz de datos de entrada a escenario 2.1

La imagen siguiente muestra cómo luce el caso de prueba en análisis desde el Selenium IDE:

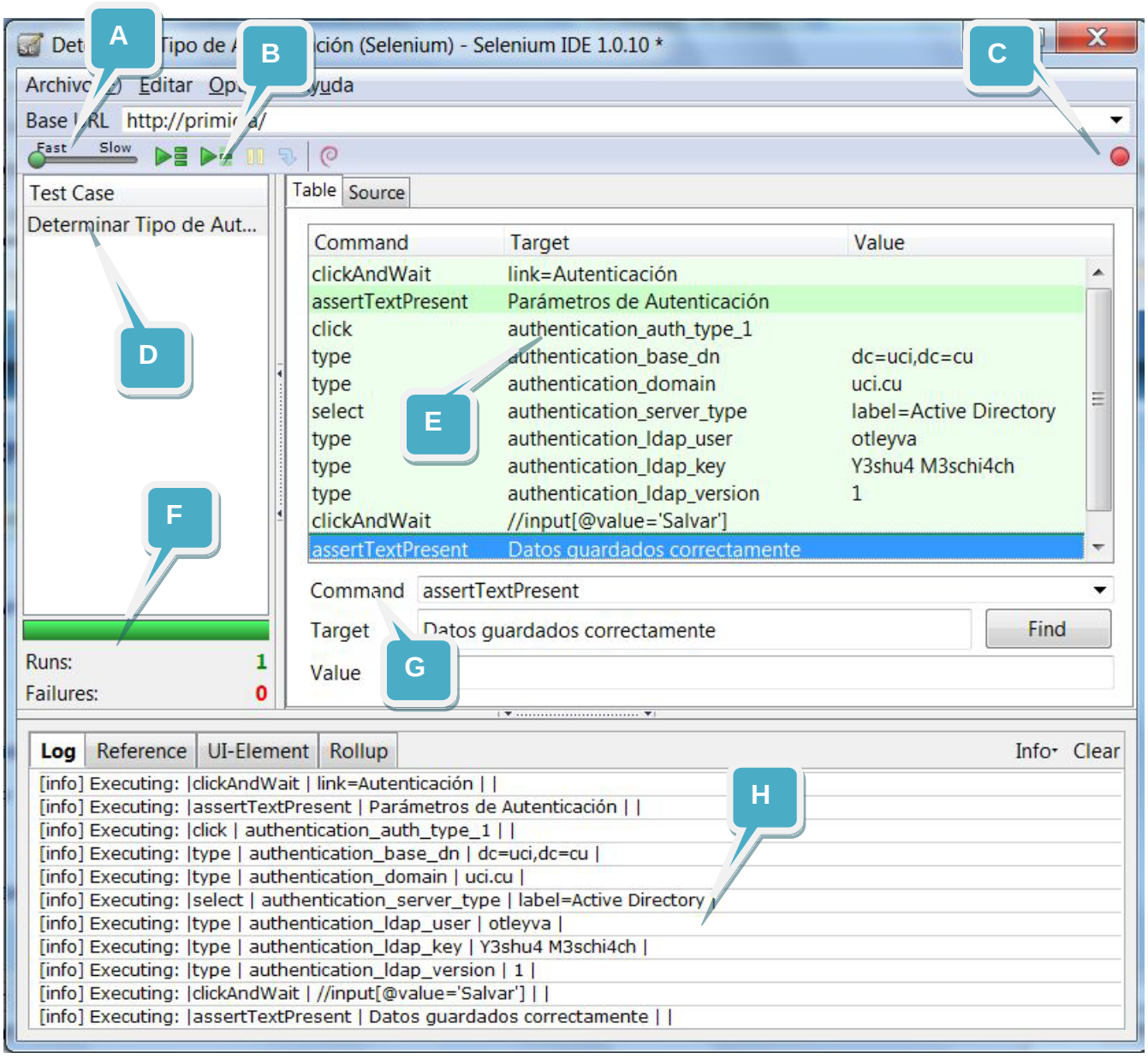

Figura 18: Elementos del Selenium IDE

Elementos

A. Determina la velocidad de ejecución para que el usuario pueda apreciar la prueba.

B. Permite escoger que prueba presente en la suite ejecutar, una a una o todas a la vez. Pausar o continuar las pruebas y realizarlas paso a paso.

C. Permite grabar los movimientos de una navegación.

D. Lista de pruebas en la suite.

E. Datos de la prueba.

F. Cantidad de pruebas realizadas: aceptadas (verde) y fallidas (roja).

G. Permite introducir comandos que se ejecutarán en la prueba.

H. Registro de los eventos y sucesos en la prueba.

3.5. Conclusiones Parciales

En este capítulo se presentaron los conceptos principales que se manejan en el ámbito de las pruebas del desarrollador, según la metodología utilizada en esta investigación. Asimismo, se demostraron los mecanismos que el *framework* Symfony utiliza para implementar los tipos de prueba que maneja (unitarias y funcionales). A través de ejemplos y el empleo de varias figuras y tablas, se presentaron las formas de trabajo de Symfony y especialmente de la herramienta para pruebas funcionales Selenium IDE.

Las pruebas que se le realizan a una aplicación informática, específicamente las pruebas funcionales, permiten comprobar que esta ejecuta correctamente las acciones para las que fue construida. La suite de pruebas creada con la herramienta Selenium IDE que se le aplicó al Subsistema de Configuración, demuestran que funcionalmente y operacionalmente esta aplicación web es factible. Esto significa que el subsistema cumple con los requisitos funcionales y no funcionales definidos en su diseño.

CONCLUSIONES

La Plataforma PRIMICIA es un producto informático en desarrollo sujeto a actualizaciones. Es por ello que uno de los principales aportes reales de esta investigación, fue la elaboración de un conjunto de diagramas y artefactos que cumplen con normas de la metodología de desarrollo RUP. Estos diagramas son de necesaria referencia en el futuro para posibles modificaciones dentro de la plataforma en versiones posteriores de la aplicación. Este aporte se ejemplifica en las propias actualizaciones a los diseños anteriores propuestas en este trabajo, donde se evidencia la importancia de documentar con artefactos los nuevos avances. Sin embargo, el más importante aporte es la elaboración de una aplicación web que permite la configuración directa de los subsistemas de Administración y Transmisión que componen la plataforma PRIMICIA.

La aplicación mencionada permite la personalización del producto informático poniéndolo a punto para enfrentar nuevos requisitos de nuevos clientes. El Subsistema de Configuración automatiza en gran medida ese proceso permitiendo ajustar la mayoría de esos parámetros desde la aplicación web, concebida de esta forma para que pueda ser accedida desde cualquier punto de una red. Este subsistema le otorga valor añadido al producto pues, al ponerlo también a disposición del cliente que adquiera el sistema, permite contar con una aplicación que es accesible y usable tanto por usuarios con bajo nivel de entendimiento en asuntos informáticos, como de nivel medio y avanzado.

Así entonces, se concluye que:

1. El Subsistema de Configuración de PRIMICIA dota a esta de un elevado nivel de personalización del producto informático, haciéndolo voluble a las exigencias y requisitos de la mayoría de clientes que quieran adquirirlo.
2. El impacto de este subsistema se debe medir en cuestiones de reducción de tiempo, esfuerzo y dinero cuando la empresa se enfrenta a la personalización del producto para clientes diferentes, representando un sustancial ahorro de estas variables.
3. Se le imprime un elevado nivel de profesionalidad al producto, lo cual no es más que una cualidad que ejemplifica la relevante capacidad y aplicación con que la empresa que desarrolla dicho producto, ejerce su actividad.

RECOMENDACIONES

Aunque los resultados obtenidos son satisfactorios, es recomendable seguir investigando con el propósito de identificar aún nuevos parámetros que se incluyan en la configuración y personalización de la plataforma. Es necesario para esto valerse de la experiencia obtenida de la interacción con clientes anteriores y nuevos, así como de las opiniones de todo el equipo que trabaja en las mejoras del producto para nuevas versiones. Estas personas son las que más cerca están de proveer el conocimiento necesario.

GLOSARIO

Aplicación informática: Programa preparado para una utilización específica, como el pago de nóminas, formación de un banco de términos léxicos, etc. **(2)**. Este tipo de programa es diseñado como herramienta para permitir a un usuario realizar una o varias acciones a través de un ordenador.

Aplicación web: Estas son especificaciones de las aplicaciones informáticas. Tiene como característica especial que los usuarios pueden utilizarla accediendo a un servidor web a través de Internet o una intranet mediante un navegador. Es una aplicación que se crea y codifica empleando lenguajes soportados e interpretados por los navegadores web

Arquitectura (de software): Son las guías y especificaciones con base a las cuales se debe construir una aplicación informática para dar solución a un problema. La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema y establece los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades.

Código fuente: Son las líneas de texto escritas en un lenguaje de programación determinado, que le dicen a la máquina que debe hacer y en qué momento a través del programa. En el código fuente está completamente descrito el funcionamiento de un programa.

Confiabilidad: Cualidad de confiable. *Fiabilidad*: Probabilidad del buen funcionamiento de algo **(2)**. En informática es la cualidad que describe a una aplicación informática en términos de manejo de errores y funcionamiento excepcional de esta.

Funcionalidades: Cualidad de funcional **(2)**. En informática, encapsulamiento de varias líneas de código fuente que representan la respuesta a un caso de uso específico, o resuelve un problema determinado. Como un proceso, recibe elementos (datos) de entrada y devuelve su correspondiente salida.

Implementación: Acción y efecto de implementar. *Implementar*: Poner en funcionamiento, aplicar métodos, medidas, etc., para llevar algo a cabo **(2)**.

Mapeador Relacional de Objetos (ORM): En informática, es una aplicación, librería de clases o *plugin* que empleauna técnica de programación para convertir objetos manipulados por un gestor de base de datos (tablas, campos, secuencias, etc.) a un lenguaje de programación orientado a objetos, utilizando un motor de persistencia. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional. Esto posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo).

Marco de Trabajo (*framework*): En términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar. En el desarrollo de software, un *framework* es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Personalización: Técnica que trata de eliminar la sobrecarga de información, mediante la adaptación de los contenidos para cada tipo de usuario. En la comercialización de productos informáticos, es la manera de poner a punto una aplicación informática para un nuevo cliente de acuerdo a los requerimientos de este.

Portabilidad: En informática, es la cualidad un *software* es descrito en términos de la reutilización de su código fuente al ser desplegado en plataformas diferentes. Es sencillamente, la independencia que posee una aplicación informática del sistema operativo.

Proceso: Conjunto de las fases sucesivas de un fenómeno natural o de una operación artificial **(2)**. En ingeniería de software, el conjunto de actividades que transforman artefactos de entrada en determinados artefactos de salida.

Producto: Cosa producida. Caudal que se obtiene de algo que se vende, o el que ello reditúa **(2)**. Un producto es cualquier cosa que se puede ofrecer a un mercado para satisfacer un deseo o una

necesidad. El producto es parte de la mezcla de marketing de la empresa, junto al precio, distribución y promoción, lo que conforman las 4 P’s.

Producto informático: Derivación de un proceso productivo que arroja como resultado una aplicación informática comercializable.

Sistema informático: Conjunto de partes interrelacionadas, hardware, software y de recurso humano que permite almacenar y procesar información. Puede usarse como sinónimo de aplicación informática.

Solución informática: Aplicación informática personalizada para ajustarse a los requerimientos de un usuario en específico y que resuelve un problema determinado.

Usabilidad: En informática, es la cualidad de un software que describe con qué facilidad con que las personas pueden utilizar una herramienta particular o cualquier otro objeto fabricado por humanos con el fin de alcanzar un objetivo concreto. La usabilidad se refiere a la claridad y la elegancia con que se diseña la interacción de los usuarios con un programa de ordenador o un sitio web.

BIBLIOGRAFÍA

1. Andrés, Carlos de Jesús. ***Diseño del Subsistema de Configuración de la Plataforma de Televisión Informativa PRIMICIA.*** **Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2010.**

2. Real Academia Española. **Diccionario de la Real Academia Española. [En línea] [Citado el: 29 de 01 de 2011.] http://buscon.rae.es/.**

3. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. ***El Proceso Unificado de Desarrollo de Software.*** **Madrid : Addison Wesley, 200. 84-7829-036-2.**

4. Gómez, Levián Lara. ***Conceptualización de la automatización de la personalización y configuración de PRIMICIA.*** **Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2009.**

5. H. Canós, José, Letelier, Patricio y Penadés, María del Carmen. ***Metodologías Ágiles en el Desarrollo de Software.*** **Valencia : Universidad Politécnica de Valencia, 2010.**

6. IBM Corporation. ***Rational Unified Process.*** **[IBM Corporation] 26 de Noviembre de 2010. Ayuda oficial en español de RUP.**

7. The PHP Group. **Manual de PHP.** ***Prefacio.*** **[En línea] The PHP Group, 2 de Diciembre de 2010. [Citado el: 2 de Diciembre de 2010.] http://docs.php.net/manual/es/preface.php.**

8. Potencier, Fabien y Zaninotto, François. ***The Definitive Guide to Symfony.*** **s.l. : Apress. 978-1590597866.**

9. Sensio. ***sitio Web de SensioLabs.*** **[En línea] Sensio, 2010. [Citado el: 1 de Diciembre de 2010.] http://www.sensiolabs.com/en.**

10. NetBeans Community. **Bienvenido a NetBeans y www.netbeans.org. *sitio Web de la Comunidad de NetBeans.* [En línea] 2010. [Citado el: 2 de Diciembre de 2010.] http://netbeans.org/index_es.html.**

11. Doctrine Project. **About the Doctrine Project. *sitio Web del Proyecto Doctrine.* [En línea] SensioLabs, 2010. [Citado el: 3 de Diciembre de 2010.] http://www.doctrine-project.org/projects/orm/1.2/docs/manual/introduction/en#introduction..**

12. Mozilla. **¿What is Firebug? *sitio Web de Firebug.* [En línea] Mozilla, 2010. [Citado el: 3 de Diciembre de 2010.] http://getfirebug.com/whatisfirebug.**

13. Sommerville, Ian. ***Ingeniería del Software. Séptima Edición.* Madrid : PEARSON EDUCATION S.A., 2005. 84-7829-074-5.**

14. SensioLabs. ***A Gentle Introduction to symfony.* s.l. : SensioLabs, 2011.**

15. Matos, Heikel Villar. ***Implementación de los subsistemas de Administración y Transmisión de la Plataforma de Televisión Informativa PRIMICIA.* Ciudad de La Habana : Universidad de las Ciencias Informáticas, 2008.**

16. IBM Corporation. **Rational Unified Process. *Best Practices for Software Development Teams.* [En línea] 2001. [Citado el: 28 de Octubre de 2010.] http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf..**