Universidad de las Ciencias Informáticas
Facultad 9

Implementación de los subsistemas de Administración y Transmisión de la Plataforma de Televisión Informativa PRIMICIA.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autor:

Heikel Villar Matos

Tutor:

Ing. Jorge Daniel Olivares Tamayo

Ciudad de La Habana, diciembre de 2008
"Año 50 de la Revolución"

“El conocimiento nos hace responsables”.

Che.

Dedicatoria

A mi abuela…

A mi madre…

A mi familia…

A los amigos de siempre…

Agradecimientos

Le agradezco a mi familia por su apoyo.

Al chino por ser más que tutor, amigo.

A todos los miembros del proyecto por su colaboración.

A todos mis compañeros por su ayuda.

Resumen

El uso de aplicaciones web crece diariamente, gracias a su potencia y al aumento de la conectividad de alta velocidad entre usuarios y sistemas. Cada vez es más habitual encontrar usuarios que utilizan aplicaciones web como el correo electrónico, lectores de noticias o incluso herramientas para la creación de documentos.

La implementación y el empleo de aplicaciones web es la solución a los diversos problemas que afrontan numerosas empresas a la hora de automatizar sus procesos. Algunas de las empresas involucradas en esta evolución cuentan con redes de televisión interna que le permite visualizar las informaciones a través de sus telerreceptores.

Actualmente existen muchas empresas, que aunque presentan una red de televisión interna, no cuentan con un sistema automatizado, que les permita realizar un proceso de trasmisión y gestión de información de interés propio de la institución, así nace la necesidad de crear una Plataforma Web de Televisión Informativa, que permita administrar y trasmitir, noticas y recursos multimedia en las redes de televisión interna.

En el presente trabajo, se plasmará todo el proceso de implementación y desarrollo de la plataforma PRIMICIA, la cual esta destinada, a trasmitir información en diferentes formatos.

Palabras claves: Implementación, información, aplicación web, televisión, PRIMICIA.

Índice

Índice de Tablas

Índice de Figuras

Introducción

En un principio la web era sencillamente una colección de páginas estáticas y documentos, para su consulta o descarga. El paso inmediatamente posterior en su evolución fue la inclusión de un método para elaborar páginas dinámicas que permitieran que lo mostrado tuviese carácter dinámico (es decir, generado a partir de los datos de la petición). Este método fue conocido como Interface de Acceso Común (*Common Gateway Interface*, en ingles abreviado CGI) y definía un mecanismo mediante el que se podía pasar información entre el servidor y ciertos programas externos. Los CGIs siguen utilizándose ampliamente; la mayoría de los servidores web permiten su uso debido a su sencillez. Además, dan total libertad para elegir el lenguaje de programación que se desea emplear.

El funcionamiento de los CGIs tenía un punto débil, cada vez que se recibía una petición, el servidor debía lanzar un proceso para ejecutar el programa CGI. Como la mayoría de los CGIs estaban escritos en lenguajes interpretados, como Perl o Python, o en lenguajes que requerían "*run-time environment*[1]", como Java o VisualBasic, el servidor se veía sometido a una gran carga. La concurrencia de múltiples accesos al CGI podía comportar problemas graves.

Por eso se empiezan a desarrollar alternativas a los CGIs que solucionaran el problema del rendimiento. Las soluciones llegan básicamente por 2 vías:

1) Se diseñan sistemas de ejecución de módulos mejor integrados con el servidor, que evitan la instanciación y ejecución de varios programas.

2) Se dota a los servidores de un intérprete de algún lenguaje de programación que permita incluir el código en las páginas de forma que lo ejecute el servidor, reduciendo el intervalo de respuesta.

Estas alternativas implicaron que se experimentara un aumento del número de arquitecturas y lenguajes que permiten desarrollar aplicaciones web. Todas siguen alguna de estas vías. Las más útiles son las que

[1] run-time-enviroment: Entorno de tiempo de ejecución, es una máquina virtual de estado que permite el uso de software para procesos de servicios o programas, mientras que un ordenador está encendido. Tal vez se refieren al propio sistema operativo o el software que corre debajo de ella.

permiten mezclar los 2 sistemas: un lenguaje integrado que permita al servidor interpretar comandos "incrustados" en las páginas HTML y, además, un sistema de ejecución de programas mejor enlazado con el servidor, que no implique los problemas de rendimiento propios de los CGIs. Una tecnología de éxito y una de las más utilizadas es el lenguaje PHP (*Personal Home Page*). Se trata de un lenguaje interpretado que permite la incrustación de HTML (Lenguaje de Marcas de Hipertexto, en inglés *HyperText Markup Language*) en los programas, con una sintaxis derivada de C y Perl. El hecho de ser sencillo y potente ha contribuido a hacer de PHP una herramienta muy apropiada para determinados desarrollos.

El uso de aplicaciones web crece diariamente, gracias a su potencia y al aumento de la conectividad de alta velocidad entre usuarios y sistemas. Cada vez es más habitual encontrar usuarios que utilizan aplicaciones web como el correo electrónico, lectores de noticias o incluso herramientas para la creación de documentos.

La implementación y el empleo de aplicaciones web es la solución a los diversos problemas que afrontan numerosas empresas a la hora de automatizar sus procesos. Cuando se lleva a cabo este proceso la empresa u organismo en cuestión está en busca de un aumento de su productividad o en la mejora de un determinado servicio.

El desarrollo que hoy ha alcanzado este tipo de sistemas ha modificado la manera en que se gestiona la información y la ha hecho cada vez más eficiente. En la actualidad la información es uno de los recursos más importantes con el que puede contar toda institución, es por eso que es muy usual escuchar que se está en la era de la información.

Hoy día muchas instituciones como aeropuertos, terminales y hoteles necesitan que se muestre información a una gran cantidad de personas utilizando para ello sus redes de televisión interna. Para realizar esta tarea es necesario que se cuente con todo un sistema informático que permita la automatización de los procesos de confección y visualización de todas estas informaciones. Hoy muchas de estas actividades son realizadas de forma manual, lo que trae como consecuencia que el proceso sea ineficiente y que no siempre se cuente con la inmediatez que se necesita. Para dar solución a esta problemática, se desarrolla la Plataforma de Televisión PRIMICIA, y se conceptualiza el proceso de gestión y trasmisión de noticias así como de información de interés para el personal de dichas empresas. El **problema** identificado para darle solución a la problemática en cuestión se determina como:

Ausencia de una aplicación que permita la Administración y Transmisión de la Plataforma de Televisión Informativa PRIMICIA.

A partir del problema planteado y teniendo como base los resultados arrojados del análisis y el diseño de la Plataforma se desea construir e implementar una aplicación que permita la administración automática del canal y la gestión de su contenido, así como la transmisión eficiente de las noticias. Proporcionándole de esta forma una herramienta que le permita realizar este proceso de manera más cómoda, ágil y productiva.

El **objeto de estudio** de esta investigación práctica está determinado por los Procesos relacionados con la Administración y Transmisión de la Plataforma de Televisión Informativa PRIMICIA.

El **campo de acción** en el cual estará enmarcada la investigación es la implementación de los procesos presentes en los subsistemas de Administración y Transmisión de la Plataforma de televisión Informativa PRIMICIA.

Se plantea como **objetivo general** de la investigación práctica: implementar los subsistemas de Administración y Transmisión de la Plataforma de Televisión Informativa PRIMICIA.

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

- Conceptualizar y enumerar los procesos de Administración y Transmisión de la Plataforma de Televisión Informativa PRIMICIA a automatizar.
- Comparar y valorar técnicas de programación y lenguajes empleados el desarrollo de aplicaciones de gestión.
- Caracterizar el empleo de patrones y estándares de codificación en aplicaciones web.
- Analizar los elementos significativos para la implementación que se obtuvieron durante el Análisis y Diseño.
- Esquematizar y modelar casos de prueba del sistema.

Se le dará cumplimiento a los anteriores objetivos específicos en orden cronológico a través de las siguientes **tareas**:

- Caracterizar los procesos relacionados con la Administración y Transmisión de la Plataforma de Televisión Informativa PRIMICIA.
- Identificar las distintas técnicas de programación existentes a nivel nacional e internacional.
- Valorar las tendencias y metodologías relacionadas con las técnicas de programación actuales, así como de las plataformas de desarrollo que la soportan.
- Definir un estándar de codificación.
- Implementar el 50% de los casos de uso críticos del sistema.
- Desarrollar casos de prueba que certifiquen el correcto funcionamiento de los algoritmos implementados.

Capítulo 1
Tendencias actuales

1.1 Introducción

En este capítulo se hace una breve referencia de las técnicas actuales de programación, tendencias, así como algunas técnicas y metodologías utilizadas en el contexto de la programación web, haciendo énfasis en las utilizadas para la confección de la aplicación que da origen al desarrollo de este trabajo.

1.2 Aplicación Web

Una aplicación web o webapp (del inglés, Web Application) es una aplicación a la que se accede mediante un navegador web a través de una red como Internet. Es una aplicación de software de computadora que se codifica en un navegador apoyado en lenguajes (tales como HTML, JavaScript, CSS, PHP, Java, etc.) y dependen de un navegador web (como Internet Explorer, Mozilla, Netscape) para hacer que la aplicación sea ejecutable.

Ventajas:

- Basta con una conexión a una red como Internet o una intranet para acceder a estas y disponer de sus funcionalidades.
- Permite el desarrollo de aplicaciones multiusuario bajo el concepto de compartición de información debido a que se accede a través de Internet.
- Los clientes no requieren de grandes especificaciones de hardware en sus ordenadores para acceder a este tipo de aplicaciones.
- La capacidad de actualizar y mantener aplicaciones web sin distribuir e instalar software en miles de equipos clientes, lo que constituye una razón clave para su popularidad.

Desventajas:

- Existe dependencia entre la calidad de la conexión a la red y la comunicación constante con el servidor de la aplicación.
- El servidor debe soportar la conexión concurrente de múltiples usuarios para que la ejecución de sus funciones sea de manera fluida, por lo que debe reunir condiciones de hardware óptimas.

1.3 Arquitectura Cliente-Servidor.

Los sistemas cliente/servidor han evolucionado junto con los avances de la informática personal, en la ingeniería del software basada en componentes, con las nuevas tecnologías de almacenamiento, comunicación a través de redes, y tecnología de bases de datos mejoradas. Un servidor es una computadora que lleva a cabo un servicio que normalmente requiere mucha potencia de procesamiento. Un cliente es una computadora que solicita los servicios que proporciona uno o más servidores y que también lleva a cabo algún tipo de procesamiento por si mismo. (1)

La arquitectura Cliente/Servidor es la integración distribuida de un sistema en red, con los recursos, medios y aplicaciones que, definidos modularmente en los servidores, administran, ejecutan y atienden las solicitudes de los clientes; todos interrelacionados física y lógicamente, compartiendo datos, procesos e información. Se establece así un enlace de comunicación transparente entre los elementos que conforman la estructura. (2)

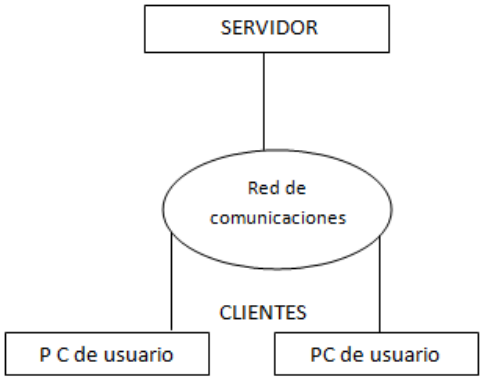


Figura 1 Esquema de la arquitectura Cliente/Servidor

Ventajas:

- Una arquitectura cliente-servidor permite que las funciones y responsabilidades de un sistema informático se distribuyan entre varios equipos independientes, los cuales se conocen el uno al otro sólo a través de una red. Esto posibilita una mayor facilidad de mantenimiento. Por ejemplo,

es posible sustituir, reparar, mejorar, o incluso cambiar la ubicación de un servidor, mientras que sus clientes no se ven afectados por ese cambio.

- Todos los datos se almacenan en los servidores, los cuales poseen controles de seguridad mucho mayor que la mayoría de los clientes. Los servidores presentan un mejor control de acceso y recursos, para garantizar que sólo los clientes con los permisos adecuados pueden acceder y cambiar los datos.
- Dado que el almacenamiento de datos está centralizado, las actualizaciones a los datos son mucho más fáciles de administrar.
- Funciona con múltiples ordenadores clientes de diferentes capacidades.
- Todos los clientes disponen de una misma y única interfaz presentada por el servidor.

Desventajas:

- La congestión del tráfico en la red constituye el principal inconveniente de este esquema cliente-servidor. A medida que el número de simultáneas solicitudes de clientes a un determinado servidor aumenta, el servidor puede ser muy sobrecargado.
- En virtud de cliente-servidor, en caso de que un servidor falle, las solicitudes de los clientes no puede ser cumplidas.

1.4 Paradigmas de programación.

1.4.1 Programación orientada a objetos.

Es un estilo de programación en que cada programa es visto como un objeto, se forma por una serie de componentes, autocontenidos que cooperan para realizar las acciones de la aplicación completa. La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar. (3)

La programación orientada a objetos introduce conceptos, que superan y amplían los ya hasta el momento conocidos. Entre otros se destacan los siguientes:

Objeto: entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos).

Clase: definiciones de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ellas.

Herencia: las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que reimplementarlo. Esto suele hacerse habitualmente agrupando los objetos en clases y estas en árboles o enrejados que reflejan un comportamiento común. (3)

Existen numerosos lenguajes de programación orientados a objetos dentro de los cuales se destacan los siguientes:

- C++
- C#
- C++
- Delphi
- Java
- Perl
- PHP (a partir de su versión 5)
- Python
- Ruby
- VB.NET

1.5 Lenguajes de Programación.

1.5.1 Lenguajes del lado del cliente.

1.5.1.1 HTML

HTML son las siglas de HyperText Markup Language (del español, Lenguaje Marcado de Hipertexto).

El lenguaje HTML es un estándar reconocido en todo el mundo y cuyas normas define un organismo sin ánimo de lucro llamado World Wide Web Consortium, más conocido como W3C. Como se trata de un

estándar reconocido por todas las empresas relacionadas con el mundo de Internet, una misma página HTML se visualiza de forma muy similar en cualquier navegador de cualquier sistema operativo. (4)

El propio W3C define el lenguaje HTML como "un lenguaje reconocido universalmente y que permite publicar información de forma global". Desde su creación, el lenguaje HTML ha pasado de ser un lenguaje utilizado exclusivamente para crear documentos electrónicos a ser un lenguaje que se utiliza en muchas aplicaciones electrónicas como buscadores, tiendas online y banca electrónica. (4)

1.5.1.2 JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. (5)

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java. Legalmente, es una marca registrada de la empresa Sun Microsystems, (5)

El código Javascript puede ser integrado dentro de nuestras páginas web. Para evitar incompatibilidades el World Wide Web Consortium (W3C) diseñó un estándar denominado DOM (en inglés Document Object Model, en su traducción al español Modelo de Objetos del Documento). (5)

Ventajas:

- Lenguaje de scripting seguro y fiable.
- Los script tienen capacidades limitadas, por razones de seguridad.
- El código Javascript se ejecuta en el cliente.
- La mayoría de los navegadores en sus últimas versiones interpretan código Javascript.

Desventajas:

- Código visible por cualquier usuario.
- El código debe descargarse completamente.
- Puede poner en riesgo la seguridad del sitio, con el actual problema llamado XSS (significa en inglés Cross Site Scripting renombrado a XSS por su similitud con las hojas de estilo CSS).
- No es un lenguaje orientado a objetos, el mismo no dispone de herencias.

1.5.1.3 CSS

El impulso de los lenguajes de hojas de estilos se produce a partir del auge y el crecimiento del uso de Internet y el incremento del lenguaje HTML para la creación de documentos electrónicos. La competencia entre los navegadores y la falta de un estándar para la definición de los estilos afectaban la creación de documentos con la misma apariencia.

En 1995, el W3C decidió apostar por el desarrollo y estandarización de CSS (del inglés, Cascading Style Sheets) y lo añadió a su grupo de trabajo de HTML. A finales de 1996, el W3C publicó la primera recomendación oficial, conocida como “CSS nivel 1”.

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas. (6)

El lenguaje CSS se utiliza para definir el aspecto o propiedades de cada elemento del contenido en una página web, tales como el color, tamaño, tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, etc.

1.5.2 Lenguajes del lado del servidor

1.5.2.1 PHP

PHP (acrónimo de Hypertext Preprocessor,) es un lenguaje "del lado del servidor" (esto significa que PHP funciona en un servidor remoto que procesa la página Web antes de que sea abierta por el navegador del

usuario) especialmente creado para el desarrollo de páginas Web dinámicas. Puede ser incluido con facilidad dentro del código HTML.

PHP es un lenguaje de script interpretado en el lado del servidor que no necesita ser compilado para ejecutarse. Para su funcionamiento necesita tener instalado Apache o IIS (Internet Information Server) con las librerías de PHP. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas.

Figura 2 Esquema de representación del funcionamiento del PHP

Ventajas:

- Muy fácil de aprender.
- Se caracteriza por ser un lenguaje muy rápido.
- Soporta en cierta medida la orientación a objeto. Clases y herencia.
- Es un lenguaje multiplataforma: Linux, Windows, entre otros.
- Capacidad de conexión con la mayoría de los manejadores de base de datos: MySQL, PostgreSQL, Oracle, MS SQL Server, entre otras.
- Capacidad de expandir su potencial utilizando módulos.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Incluye gran cantidad de funciones.
- No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

Desventajas:

- Se necesita instalar un servidor web.
- Todo el trabajo lo realiza el servidor y no delega al cliente. Por tanto puede ser más ineficiente a medida que las solicitudes aumenten de número.
- La legibilidad del código puede verse afectada al mezclar sentencias HTML y PHP.
- Dificulta la organización por capas de la aplicación.

1.6 Tecnologías

1.6.1 AJAX

El término AJAX se acuñó por primera vez en el artículo “Ajax: A New Approach to Web Applications” publicado por Jesse James Garrett el 18 de Febrero de 2005. Hasta ese momento, no existía un término normalizado que hiciera referencia a un nuevo tipo de aplicación web que estaba apareciendo.

El término AJAX es un acrónimo de Asynchronous JavaScript + XML, que se puede traducir como “JavaScript asíncrono + XML”.

El artículo define AJAX de la siguiente forma:

> “Ajax no es una tecnología en sí mismo. En realidad, se trata de la unión de varias tecnologías que se desarrollan de forma autónoma y que se unen de formas nuevas y sorprendentes.” (7)

Las tecnologías que forman AJAX son:

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.

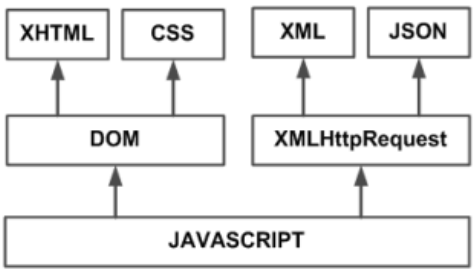


Figura 3. Tecnologías agrupadas bajo el concepto de AJAX.

AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano.

Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de AJAX mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor.

1.7 Tendencias actuales de la Programación Web.

1.7.1 ¿Qué es un Framework?

Los frameworks son similares a las bibliotecas de software, ya que son abstracciones reutilizables de código definidos en una API[1]. A diferencia de las bibliotecas, el control del flujo del programa no es dictado por el usuario que ejecuta determinada acción, sino por el propio framework. Esta inversión de control es su principal característica.

Un framework es una miniarquitectura que proporciona la estructura genérica y el comportamiento para una familia de abstracciones a lo largo de un contexto de metáforas que especifican su colaboración y es usado en un dominio dado. (8)

[1] API (**A**pplication **P**rogramming **I**nterface, en español, Interfaz de Programación de Aplicaciones): es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

El framework codifica el contexto en una clase de máquina virtual mientras hace las abstracciones abiertas. Un framework no es generalmente una aplicación específica. Una aplicación, en cambio puede ser construida con uno o más frameworks insertando dicha funcionalidad. Una posible definición de framework es: conjunto de clases cooperantes que hace reusable un diseño para una clase específica de software. Un framework proporciona una guía arquitectónica para dividir el diseño en clases y definir sus responsabilidades y colaboraciones. Un framework dicta la arquitectura de la aplicación. (8)

Un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas. (9)

El bajo costo de aprendizaje de los fundamentos de PHP, le dio al lenguaje una reputación como el "idioma de aficionados". Poco a poco la situación cambió con la llegada de los frameworks MVC y otros frameworks RAD[1] y las bibliotecas o librerías diseñados para facilitar y acelerar el desarrollo. ROR[2] (Ruby on Rails) trajo una nueva forma de ver el desarrollo Web: más rápido, más sencillo, más eficaz. Unos pocos años después de la aparición de ROR, salieron a la luz un gran número de frameworks de desarrollo rápido.

1.7.2 Patrón MVC.

MVC (model-view-controler o modelo-vista-controlador) es una arquitectura que integra tres niveles de diseño siguientes:

- El modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

[1] **RAD** (Rapid Aplication Development, en español, Desarrollo Rápido de Aplicaciones)
[2] **ROR** (Ruby on Rails**)** es un framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby, siguiendo el paradigma de la arquitectura Modelo Vista Controlador (MVC).

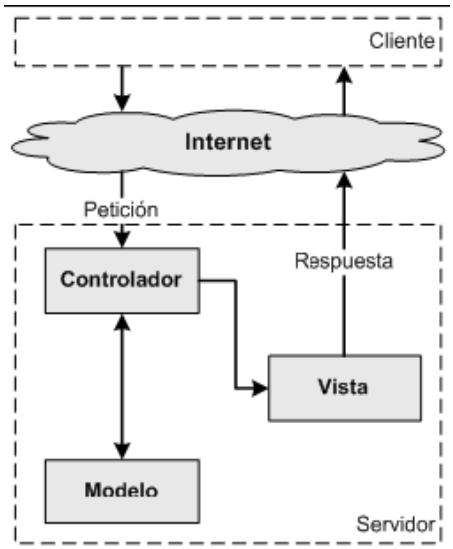


Figura 4. El patrón MVC

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación. (9)

1.7.3 Framework basados en MVC.

Debido a las ventajas que proporciona las características de la arquitectura MVC, se han desarrollado un gran número de frameworks bajo este concepto de funcionamiento, la siguiente tabla muestra un conjunto de frameworks de este tipo.

Web Application Component Toolkit	Ambivalence	Core Enterprise PHP
Limb PHP Web Application Framework	Aukyla PHP Framework	FastFrame
EZ Publish	Binarycloud	Fusebox

LogiCreate	Biscuit	FuseLogic
Mojavi	Bitweaver	Kumbia
Navigator	Booby	InterJinn
Phrame	CakePHP	Ismo
ZNF	Castor	Medusa
Achievo ATK	Cgiapp	Nexista
Akelos Framework	CodeIgniter	P4A
AModules3	Copix	PHP on Trax
PHPulse	PhpMVC	Popoon
Prado	Qcodo	rwfphp
Seagull	Sitellite	SolarPHP
sQeletor	Studs	TaniPHP
Tigermouse	Zend Framework	Wolfden CMF
Zephyr Framework	Zoop Framework	Symfony

Tabla 1. Frameworks MVC.

En mayo del 2008 la empresa francesa *Clever Age*[1] publicó en su "*Livre Blanc: Frameworks PHP pour l'entreprise*" (en español, Libro blanco: Framework PHP para la empresa) un estudio minucioso y exhaustivo, mostrando los puntos fuertes y débiles de los frameworks más utilizados por la comunidad de desarrolladores donde analizan y comparan los siguientes: Symfony, CakePHP, CodeIgniter y Zend Framework. En el resultado de esta comparación quedó el Symfony en primer lugar. El estudio realizado se basa en el método *QSOS*[2] el cual fue diseñado para evaluar el software libre de manera objetiva y está centrado en torno a los criterios para estimar los riesgos asumidos por el usuario.

[1] **Clever Age:** es una empresa de consultoría especializada en arquitectura técnica, diseño e integración de sistemas flexibles basados en las nuevas tecnologías.
[2] **QSOS:** del francés *Qualification et de Sélection de logiciels Open Source*. En español Calificación y Selección de Software Open Source.

1.7.3.1 Symfony

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

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows.

Características de Symfony.

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 aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

Symfony se caracteriza además por la automatización de los elementos comunes presentes en un proyecto web, dentro de los cuales se listan:

- La capa de presentación utiliza plantillas y layouts que pueden ser creados por diseñadores HTML sin ningún tipo de conocimiento del framework. Los *helpers* incluidos permiten minimizar el código utilizado en la presentación, ya que encapsulan grandes bloques de código en llamadas simples a funciones.
- Los formularios incluyen validación automatizada y relleno automático de datos ("repopulation"), lo que asegura la obtención de datos correctos y mejora la experiencia de usuario.
- La gestión de la caché reduce el ancho de banda utilizado y la carga del servidor.
- La autenticación y la gestión de credenciales simplifican la creación de secciones restringidas y la gestión de la seguridad de usuario.
- El sistema de enrutamiento y las URL limpias permiten considerar a las direcciones de las páginas como parte de la interfaz, además de estar optimizadas para los buscadores.
- Los listados son más fáciles de utilizar debido a la paginación automatizada, el filtrado y la ordenación de datos.
- Las interacciones con Ajax son muy fáciles de implementar mediante los helpers que permiten encapsular los efectos JavaScript compatibles con todos los navegadores en una única línea de código.

Symfony incorpora varios entornos de desarrollo diferentes e incluye varias herramientas que permiten automatizar las tareas comunes de la ingeniería del software:

- Las herramientas que generan automáticamente código han sido diseñadas para hacer prototipos de aplicaciones y para crear fácilmente la parte de gestión de las aplicaciones.
- El framework de desarrollo de pruebas unitarias y funcionales proporciona las herramientas ideales para el desarrollo basado en pruebas ("test-driven development").
- La barra de depuración web simplifica la depuración de las aplicaciones, ya que muestra toda la información que los programadores necesitan sobre la página en la que están trabajando.
- La interfaz de línea de comandos automatiza la instalación de las aplicaciones entre servidores.
- Es posible realizar cambios "en caliente" de la configuración (sin necesidad de reiniciar el servidor).

- El completo sistema de log permite a los administradores acceder hasta el último detalle de las actividades que realiza la aplicación.

1.8 Conclusiones

En este capitulo se realizó un breve estudio de las técnicas actuales de programación, tendencias de la programación web, así como algunas técnicas y metodologías utilizadas para la confección de la aplicación. Se realizó un estudio de los framework basados en MVC y se definieron algunas de las ventajas que brinda el framework escogido para la implementación.

Capítulo 2
Descripción de la solución propuesta

2.1 Introducción

En este capítulo se hace una breve descripción del diseño del sistema a implementar haciendo referencia a los principales artefactos que constituyen la base para el flujo de implementación, como son los diagramas de clases y el modelo de implementación; además del estándar de codificación a emplear.

2.2 Descripción del diseño propuesto

La aplicación está compuesta por dos subsistemas: Administración y Transmisión. El primero se encarga de la gestión del canal y el segundo de la conexión la BD de forma automática y la presentación de la información.

El subsistema de Administración está compuesto por seis módulos:

- **Seguridad:** Controla el acceso de los usuarios al sistema y permite la gestión de los mismos.
- **Redacción:** Permite la redacción de las noticias y la gestión de las que no estén publicadas.
- **Medias:** Permite la administración del archivo de recursos multimedia, en este paquete se suben al sistema las medias que van a ser utilizadas en la confección de las noticias, se modifican y eliminan de acuerdo a los intereses de las instituciones.
- **Editorial:** Permite la gestión de las secciones temáticas, las infocintas promocionales y las noticias que ya han sido publicadas, además permite que se administre el archivo de noticias del canal.
- **Gestión de Señal del Canal:** Permite la gestión de la señal del canal.
- **Reportes:** Da la posibilidad de generar reportes que ayuden a controlar la actividad de los redactores.

El subsistema de Administración está compuesto por un solo modulo:

- **Contenido:** se encarga del control de la salida de la información a transmitir, así como del formato y presentación de las noticias.

2.3 Descripción de las clases u operaciones necesarias.

Las acciones constituyen el corazón de la aplicación, puesto que contienen toda la lógica de la misma, estas utilizan el modelo y definen variables que posteriormente se utilizarán para mostrar los contenidos en las vistas.

El nombre del método de la acción siempre es execute + Xxxxxxx + (), donde la segunda parte del nombre es el nombre de la acción con la primera letra en mayúsculas.

La clase que representa las acciones de un módulo se encuentra en el archivo *actions.class.php*, en el directorio actions/ de cada uno de los módulos.

A continuación se relacionan las clases pertenecientes a cada uno de los módulos de la aplicación y una breve descripción de las principales funciones que realiza cada una:

Nombre de la clase: editorialActions	
Tipo de Clase: controladora	
Responsabilidades	
Nombre	**Breve descripción**
executeGestionarSección()	Esta función permite establecer el orden en que serán transmitidas las secciones temáticas.
executeListadoInfocintas()	Este método lista todas las infocintas que han sido creadas, y muestra además la fecha y hora en que serán transmitidas.
executeEliminarInfocinta()	Eliminar una infocinta.
executeModificarInfocinta()	Modificar una infocinta.
executeCrearInfocinta()	Crear una infocinta.
executePublicarNoticia()	Lista todas las noticias creadas dando la opción de publicarlas.
executeListadoNoticias()	Muestra todas las noticias que han sido creadas por el usuario que ha sido logueado.
executeArchivarNoticia()	Permite archivar la notica.
executeAdministrarArchivo()	Muestra todas las noticias que han sido archivadas dando la posibilidad de publicarlas o eliminarlas.
executeSacarAire()	Cambia el estado de transmisión la noticia, si se encuentra transmitiéndose,

	se puede terminar la transmisión de esta noticia.

Tabla 2 Descripción de la clase editorialActions

Nombre de la clase: gestionActions	
Tipo de Clase: controladora	
Responsabilidades	
Nombre	**Breve descripción**
executeCambiarSenal()	Esta función permite cambiar que señal se va transmitir a través del canal, puede ser la propia señal de la plataforma o un canal externo.
executeProgramarCambio()	Permite agregar cambios de señal que se quiera transmitir en determinada fecha y hora.
executeListadoCambios()	Muestra todos los cambios de señal que se han realizado.
executeModificarCambio()	Permite modificar un cambio, puede ser un cambio de fecha o señal.
executeEliminarCambio()	Elimina un cambio programado ya existente.

Tabla 3 Descripción de la clase gestionActions

Nombre de la clase: mediasActions	
Tipo de Clase: controladora	
Responsabilidades	
Nombre	**Breve descripción**
executeAdicionarMedias()	Permite agregar un recurso multimedia (puede ser imagen, música o video).
executeAdicionarGenero()	Permite agregar un género para la clasificación de los recursos de tipo música.
executeListarMedias()	Muestra un listado de todos los recursos multimedia que están presentes en el banco de archivos.
executeBuscarRecurso()	Brinda la posibilidad de buscar uno o más recursos teniendo en cuenta algunos criterios de búsqueda como el nombre del recurso.
executeEliminarMedias()	Permite eliminar un recurso.

executeVisualizarMedias()	Permite realizar una pre-visualización de un recurso multimedia.
executeModificarMedias()	Posibilita cambiar los datos de determinado recurso multimedia.

Tabla 4 Descripción de la clase mediasActions

Nombre de la clase: redaccionActions	
Tipo de Clase: controladora	
Responsabilidades	
Nombre	**Breve descripción**
executeRedactarNoticia()	Permite redactar o crear una noticia.
executeListadoNoticia()	Lista todas las noticias que han sido redactadas.
executeMostrarNoticia()	Muestra una determinada noticia.
executeEliminarNoticia()	Elimina una noticia.
executeModificarNoticia()	Permite modificar los datos de una noticia.
executeVerModificaciones()	Muestra todas las modificaciones o cambios que se le han realizado a determinada noticia.

Tabla 5 Descripción de la clase redaccionActions

Nombre de la clase: reportesActions	
Tipo de Clase: controladora	
Responsabilidades	
Nombre	**Breve descripción**
executeReporteRedac()	Muestra la cantidad, nombre y fecha de las noticias que se han elaborado por cada uno de los redactores
executeExportarR()	Convierte los datos generados en el reporte y los exporta a un fichero PDF.
executeReporteTematicos()	Muestra la cantidad de noticias y la fecha y nombre de cada una de las noticias que han sido redactadas por secciones temáticas.

Tabla 6 Descripción de la clase reportesActions

Nombre de la clase: seguridadActions	
Tipo de Clase: controladora	
Responsabilidades	
Nombre	**Breve descripción**
executeIndex()	Controla la entra de los usuarios al sistema a través del login del mismo.
executeListadoDeUsuarios()	Muestra todos los usuarios del sistema.
executeModificarUsuario()	Permite cambiar el rol y nivel de acceso de determinado usuario.
executeEliminarUsuario()	Elimina un usuario.
executeRegistrarUsuario()	Permite agregar usuarios al sistema así como establecer su rol.
executeLogout()	Cierra la sesión del usuario que se encuentra logueado en el momento de ejecutar esta función.

Tabla 7 Descripción de la clase seguridadActions

Nombre de la clase: contenidoActions	
Tipo de Clase: controladora	
Responsabilidades	
Nombre	**Breve descripción**
executeControlador()	Es la función que se encarga de controlar el flujo principal del proceso de transmisión.
TransmitirCanal()	Transmite la propia señal interna de la propia plataforma.
TransmitirCanalesExternos()	Esta función se encarga de transmitir señales de televisión externas.
MostrarVideoPresentacion()	Muestra el video promocional de presentación del canal.
MostrarNoticias()	Esta función es la encargada de mostrar todas las noticias presentes en la BD y listas para la transmisión.
MostrarCartelera()	Muestra una lista con el orden en que las noticias serán mostradas en el ciclo de transmisión.

Tabla 8 Descripción de la clase contenidoActions

2.4 Diagramas de Clases del Diseño.

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Normalmente contiene la siguiente información:

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos.
- Información sobre los tipos de los atributos.
- Navegabilidad.
- Dependencias.

A diferencias del modelo conceptual, un diagrama de este tipo contiene las definiciones de las entidades del software en vez de conceptos del mundo real. El UML no define concretamente un elemento denominado “diagrama de clases del diseño” sino que se sirve de un término más genérico “diagrama de clases”. (10)

Los diagramas de clases de diseño que se presentan a continuación representan una aproximación al sistema a desarrollar y sirven además de apoyo para la implementación de la solución.

2.4.1. Subsistema de Administración.

2.4.1.1. Módulo Seguridad.

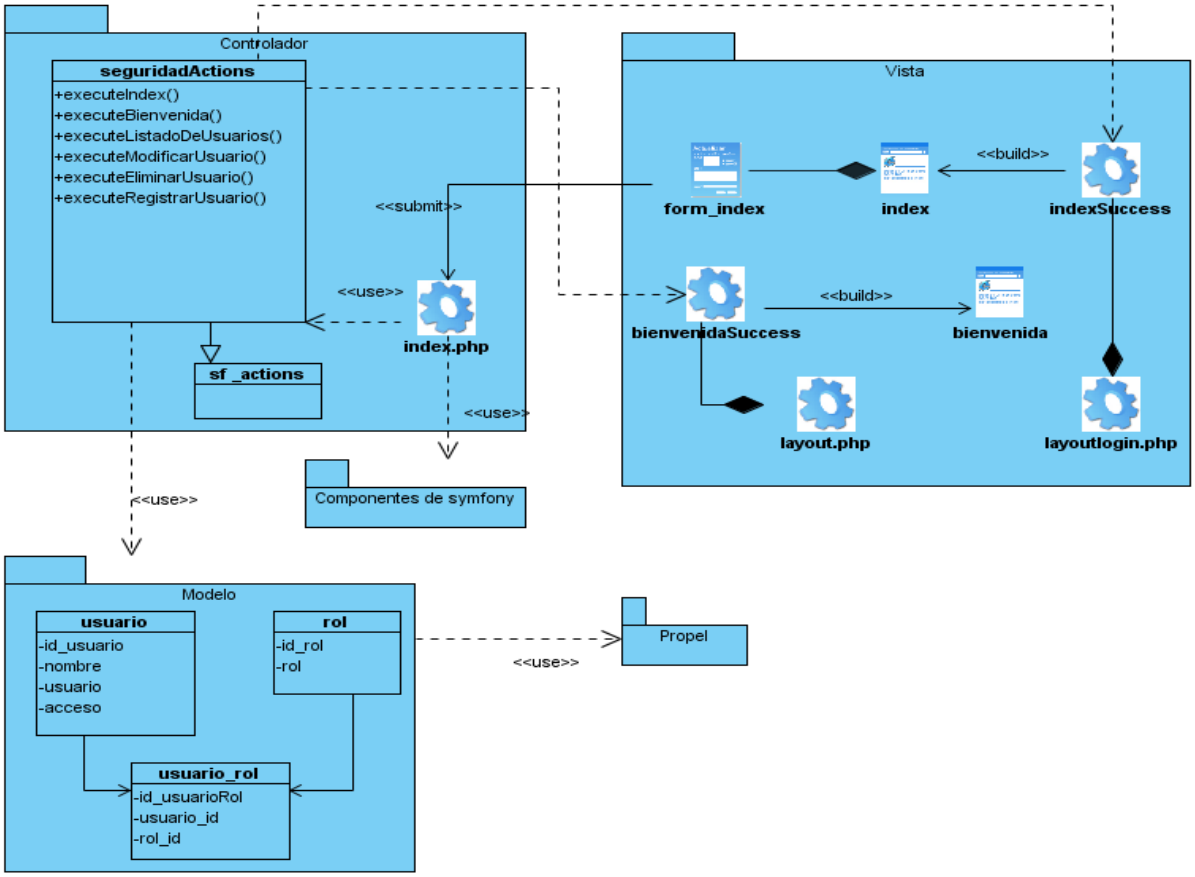


Figura 5 Diagrama de Clases: CU Autenticar Usuario.

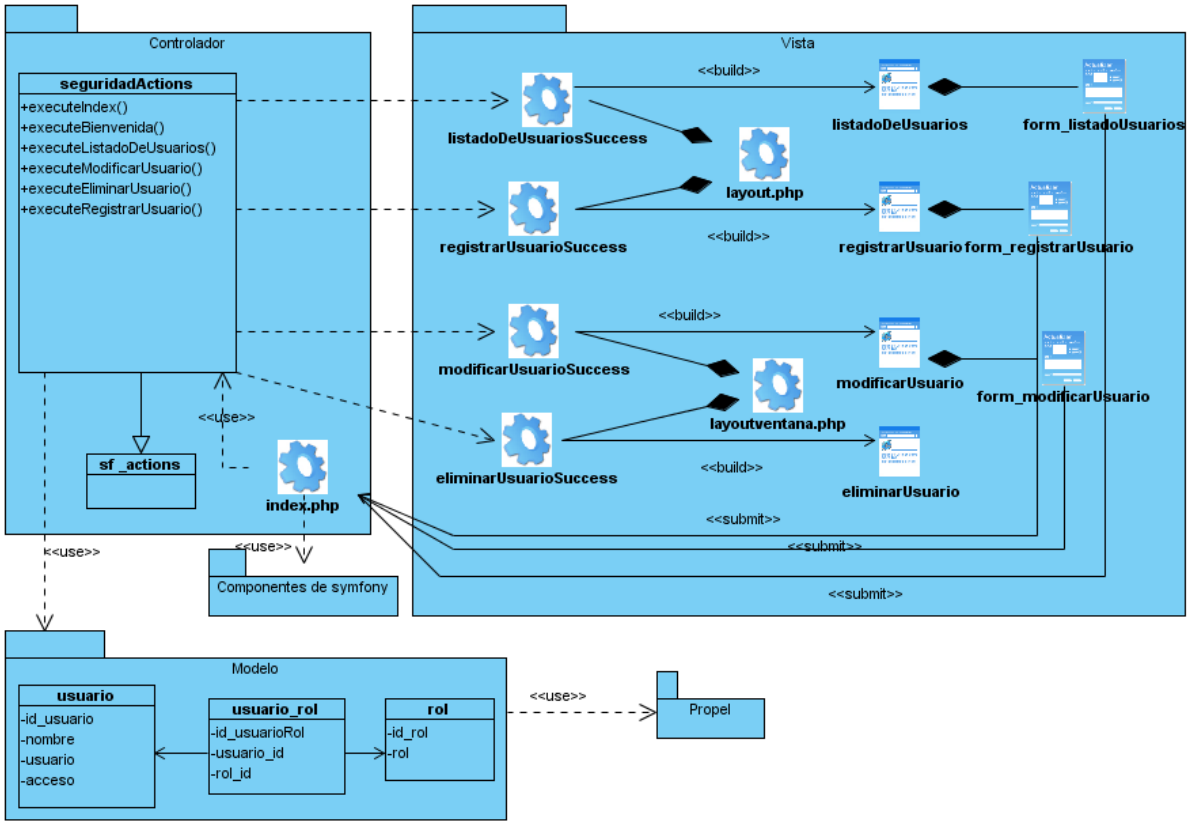


Figura 6 Diagrama de Clases: CU Gestionar Usuario.

2.4.1.2. Módulo *Redacción*.

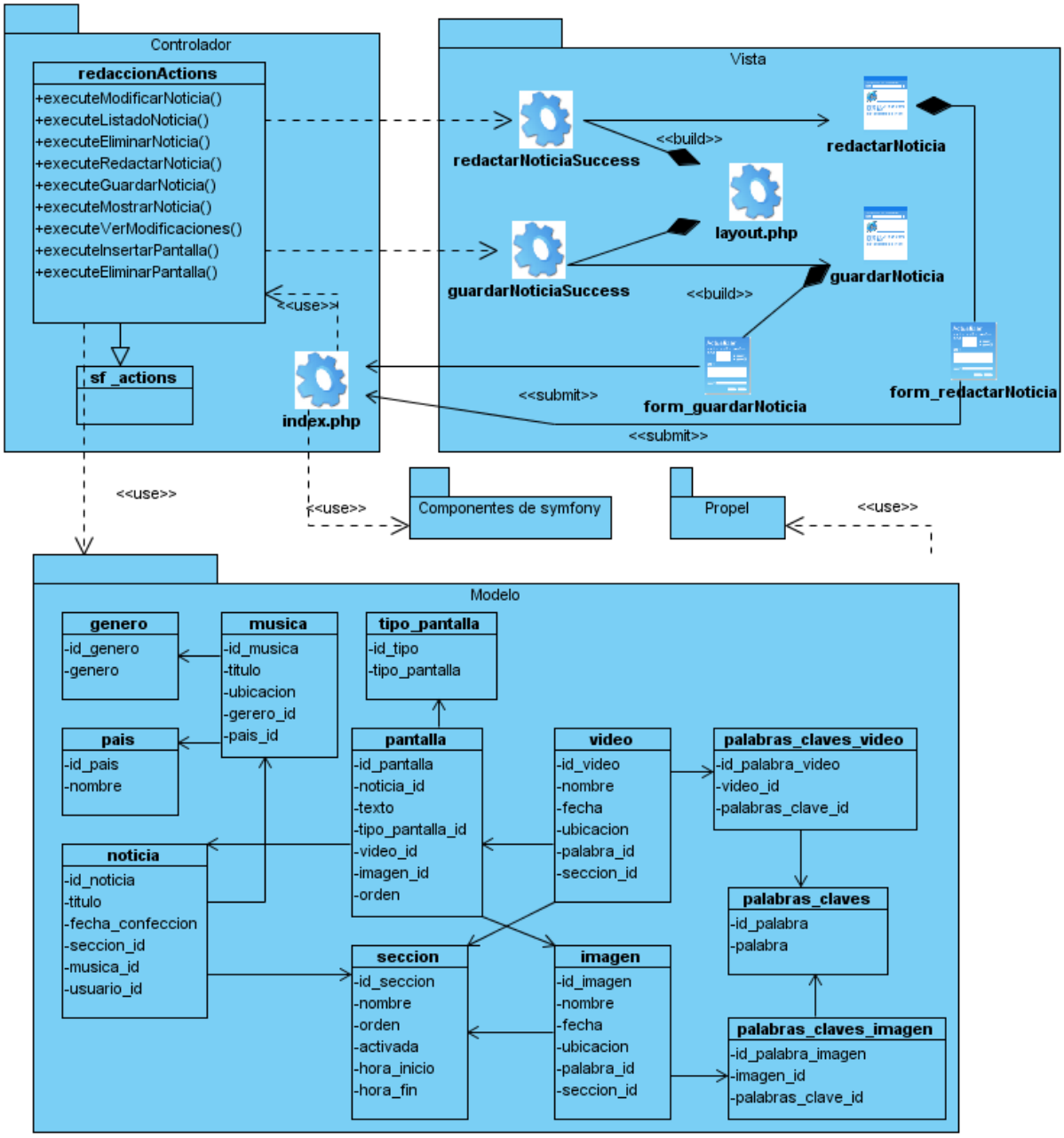


Figura 7 Diagrama de Clases: CU Redactar Noticia.

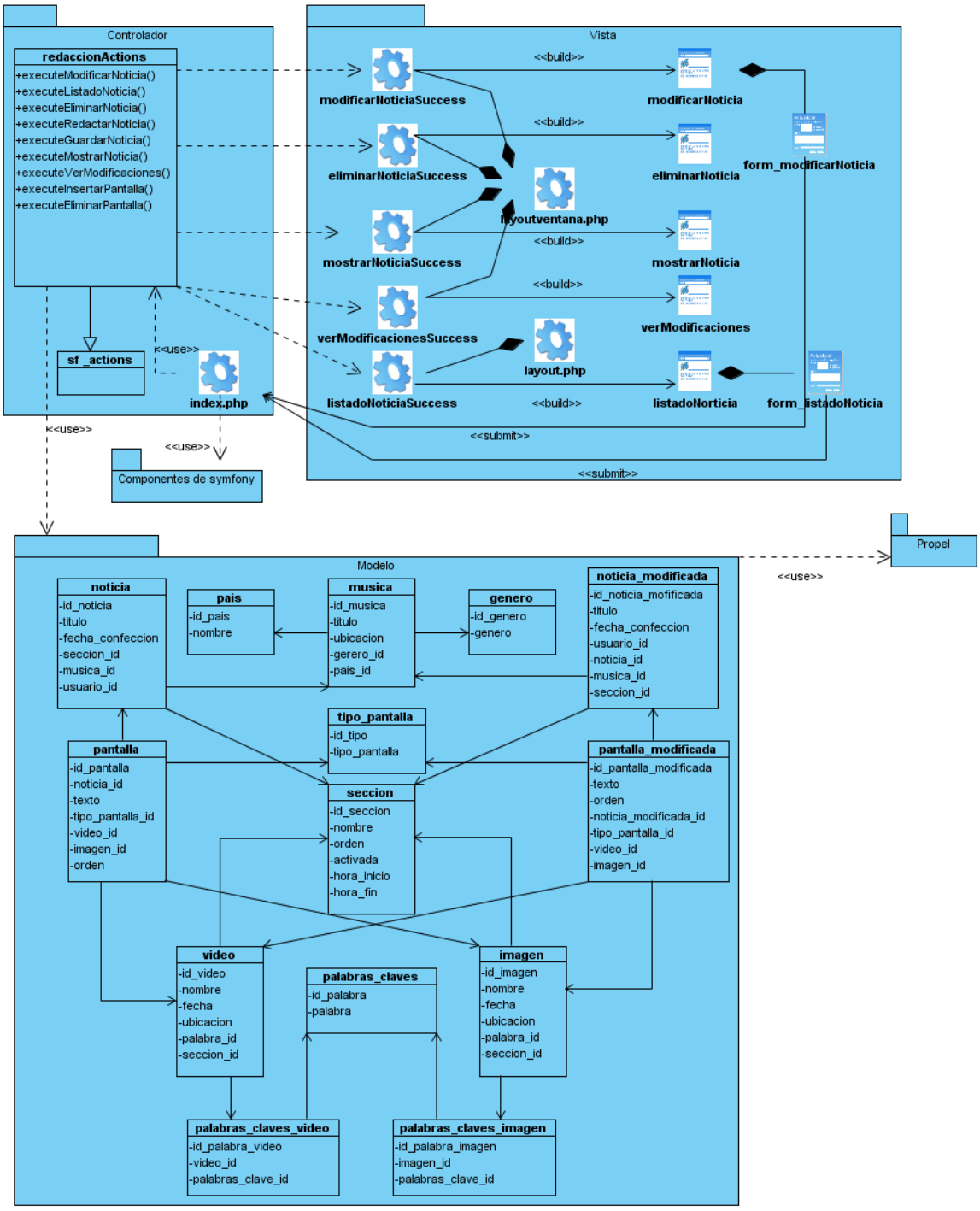


Figura 8 Diagrama de Clases: CU Gestionar Noticia redactada.

2.4.1.3. Módulo *Editorial*.

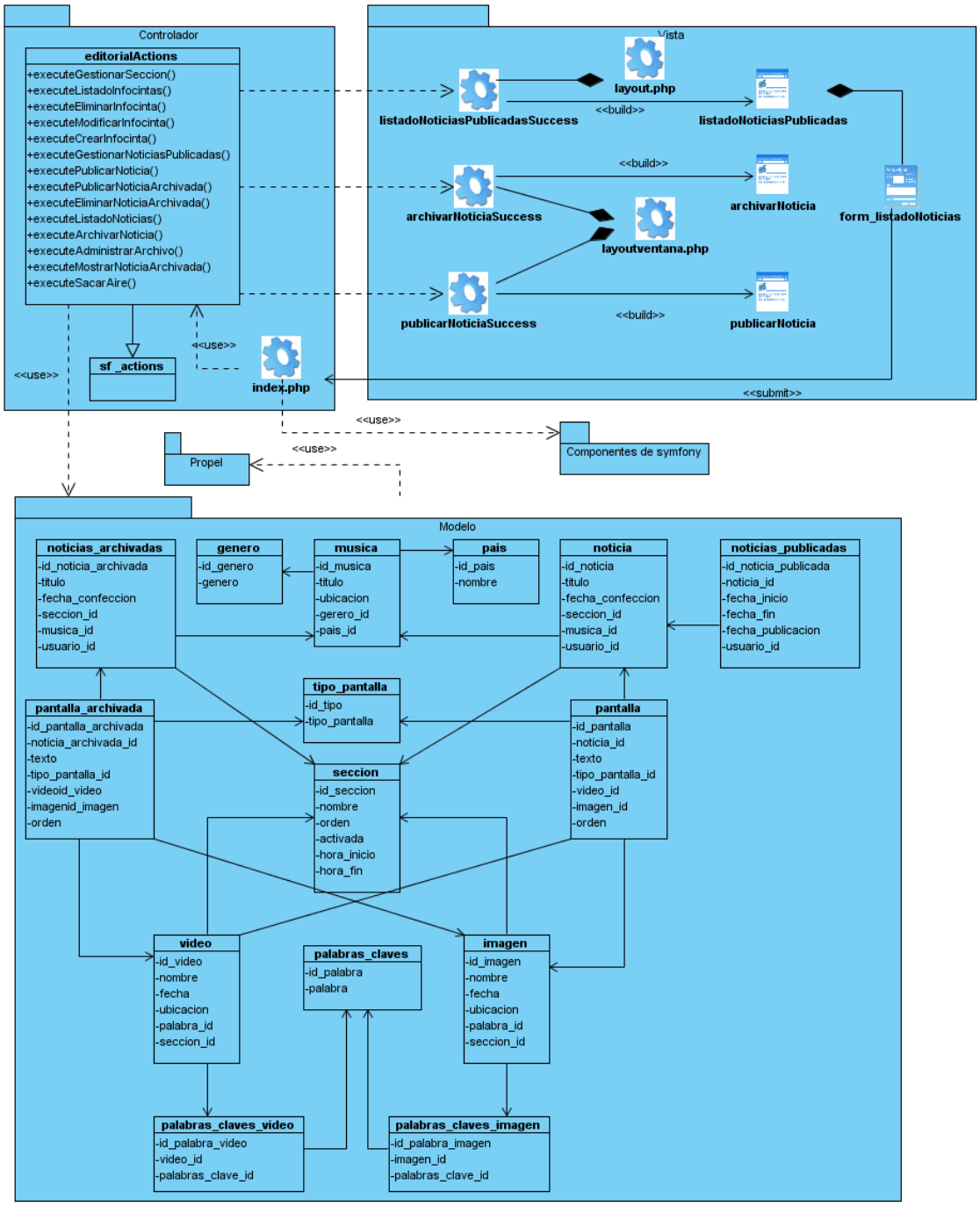


Figura 9 Diagrama de Clases: CU Publicar noticia.

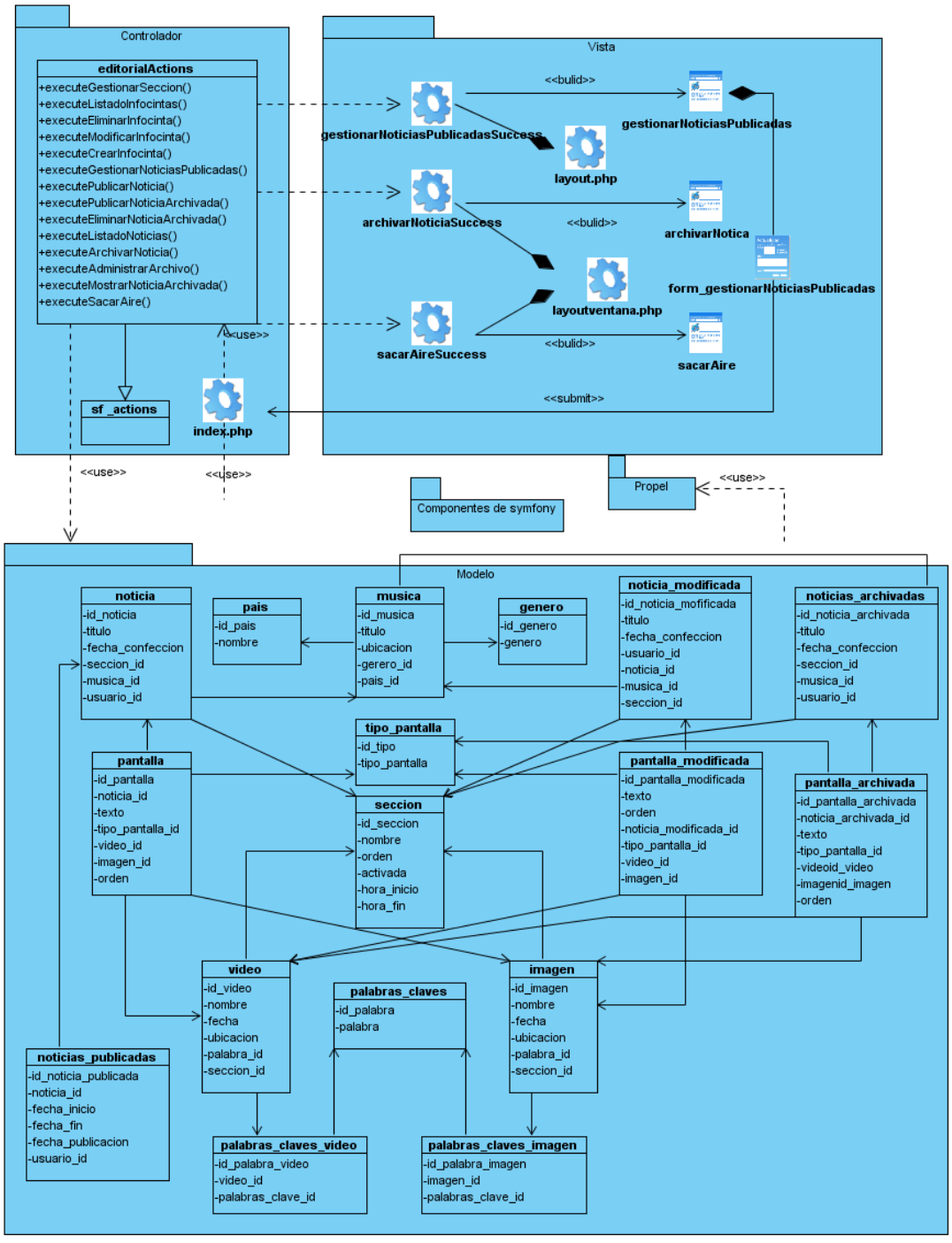


Figura 10 Diagrama de Clases: CU Gestionar Noticia Publicada.

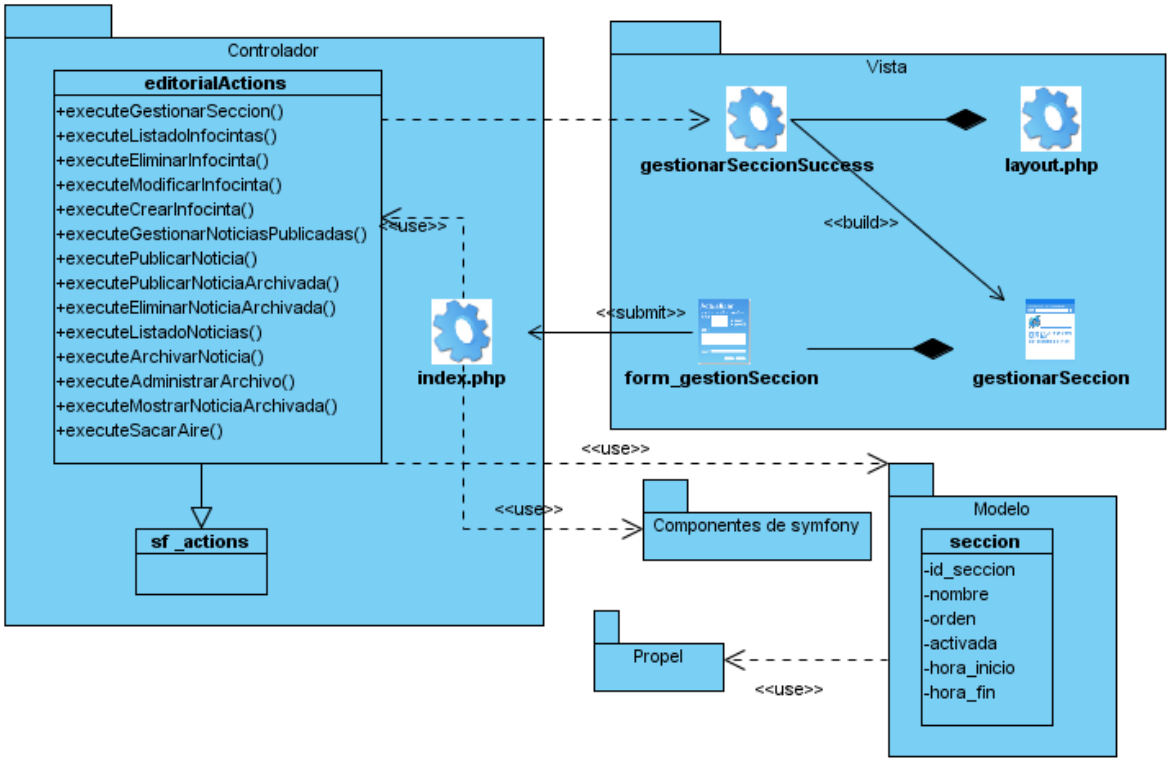


Figura 11 Diagrama de Clases: CU Gestionar Sección.

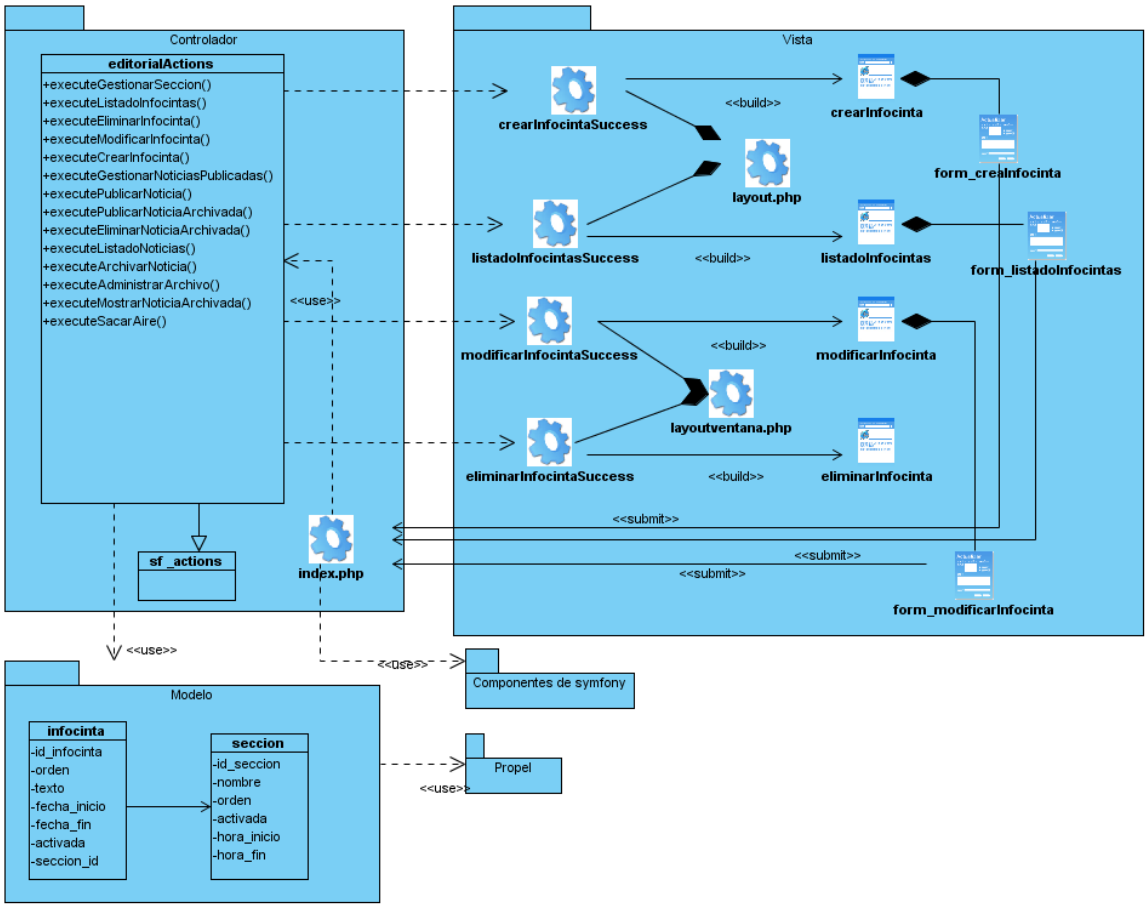


Figura 12 Diagrama de Clases: CU Gestionar Infocintas.

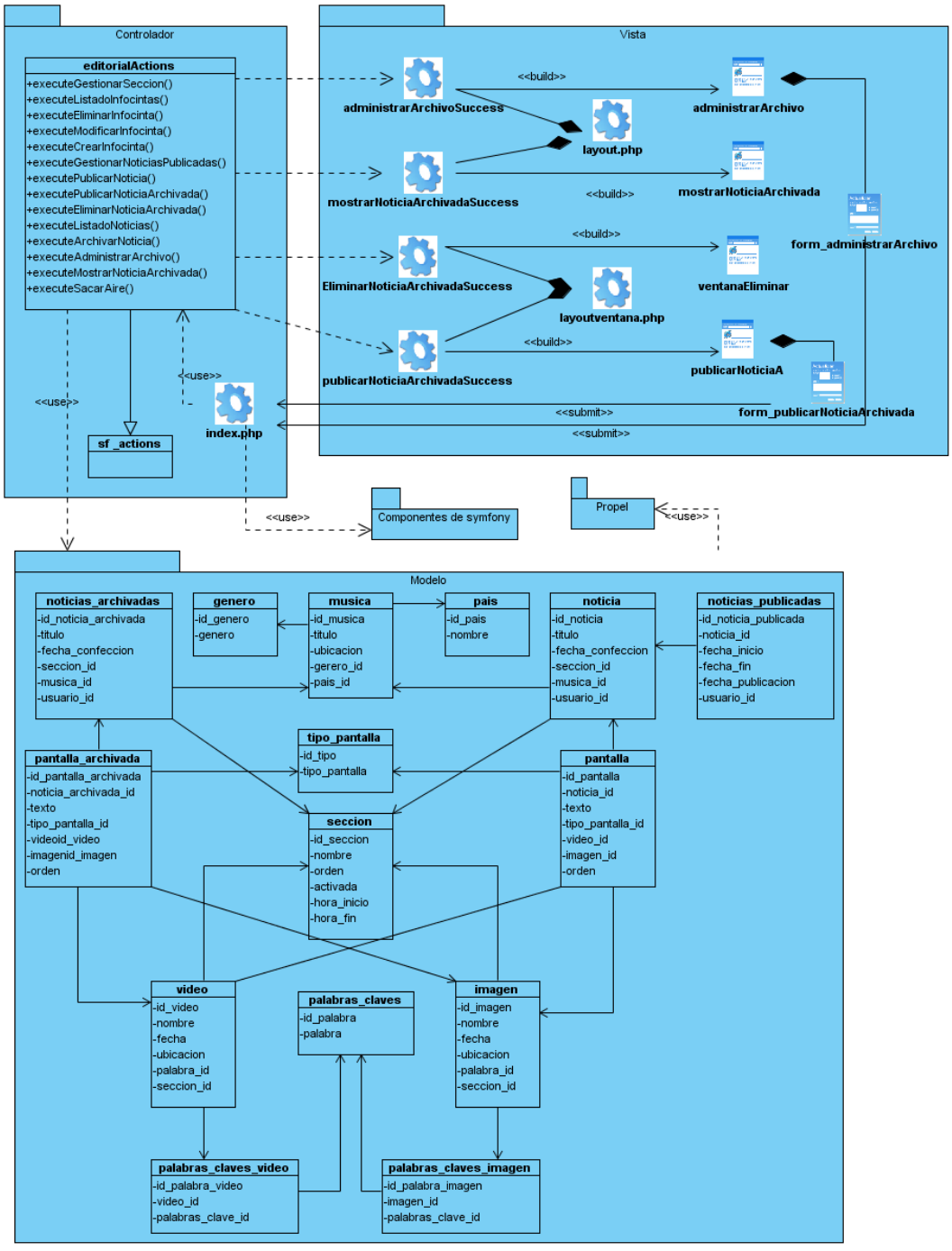


Figura 13 Diagrama de Clases: CU Administrar Archivo de Noticia.

2.4.1.4. Módulo *Medias*.

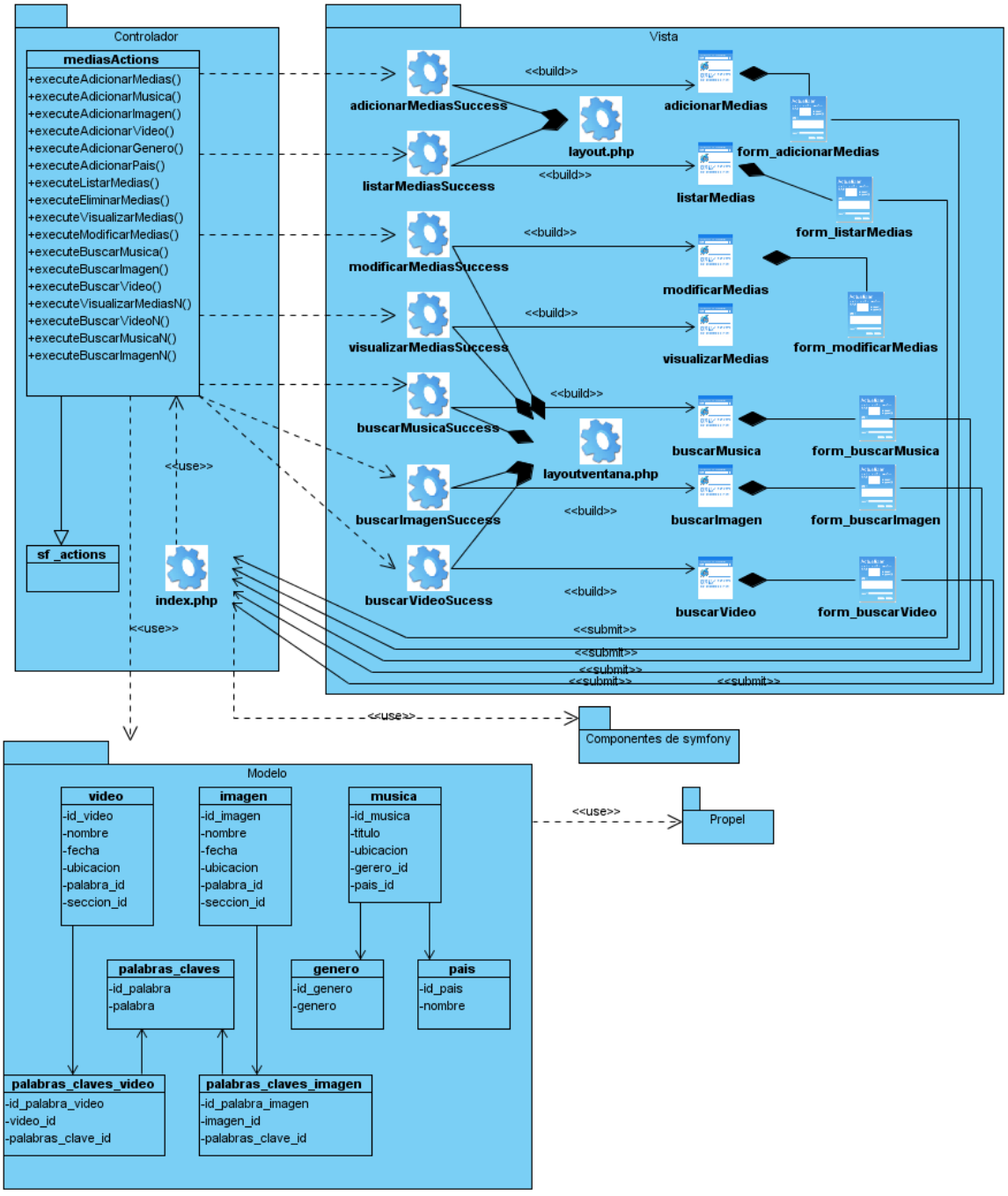


Figura 14 Diagrama de Clases: CU Administrar Archivo de Recursos Multimedia.

2.4.1.5. Módulo *Gestión de Señal del Canal*.

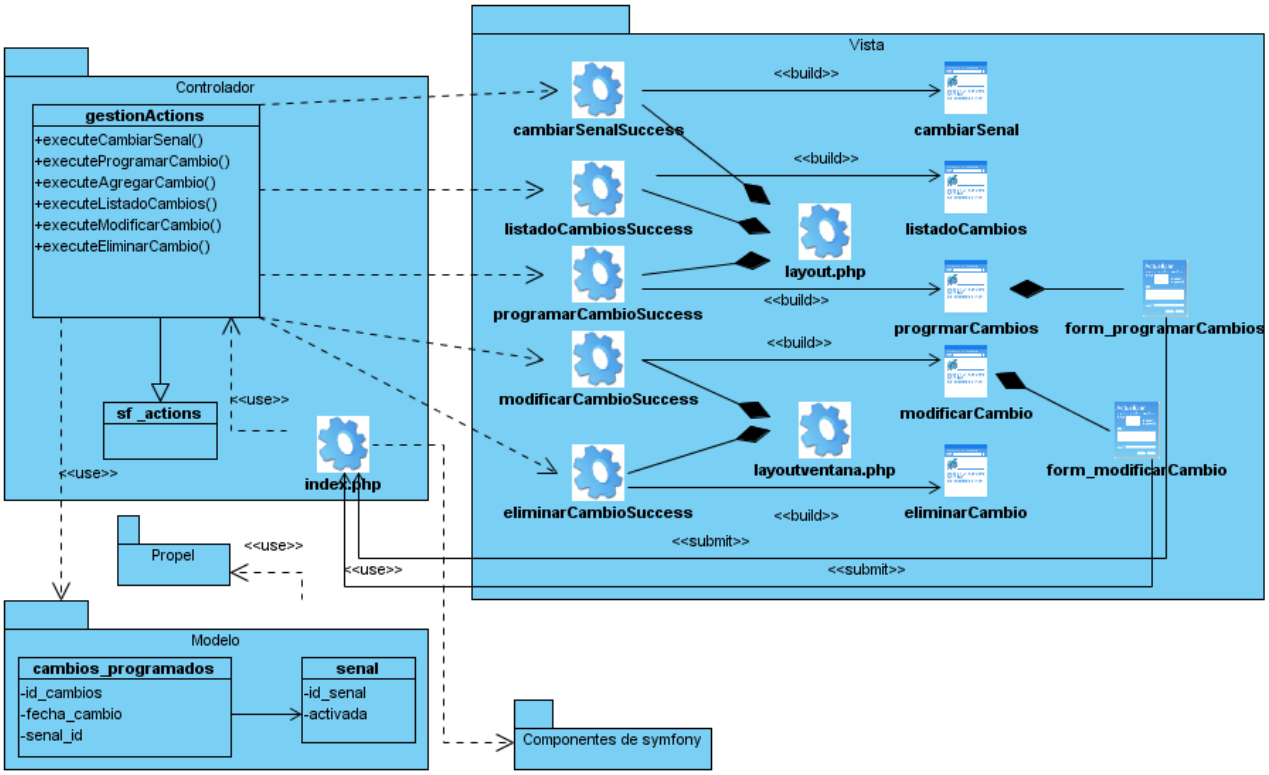


Figura 15 Diagrama de Clases: CU Gestionar Señal del Canal.

2.4.1.6. Módulo *Reportes*.

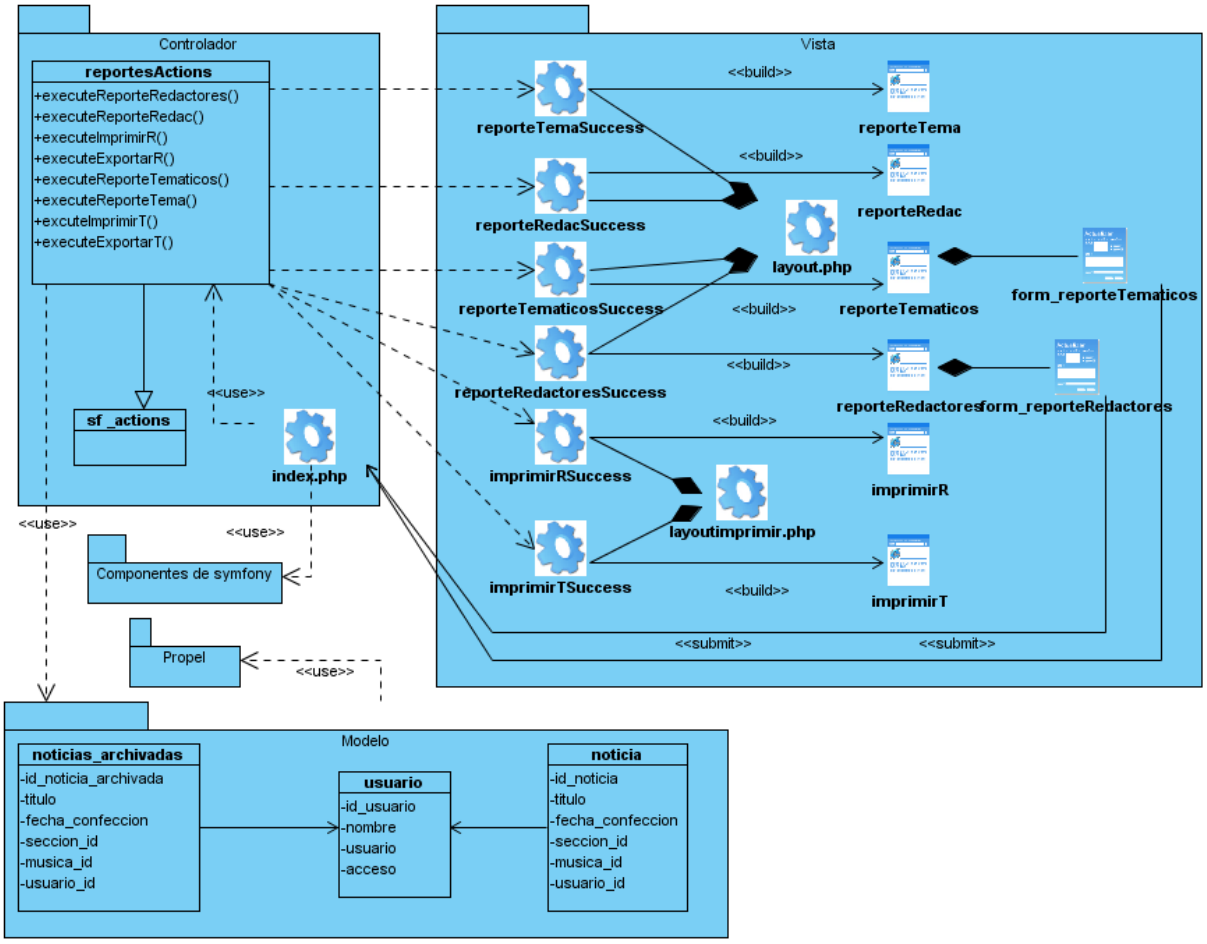


Figura 16 Diagrama de Clases: CU Gestionar Reportes.

2.4.2. Subsistema de Transmisión.

2.4.2.1. Módulo Contenido.

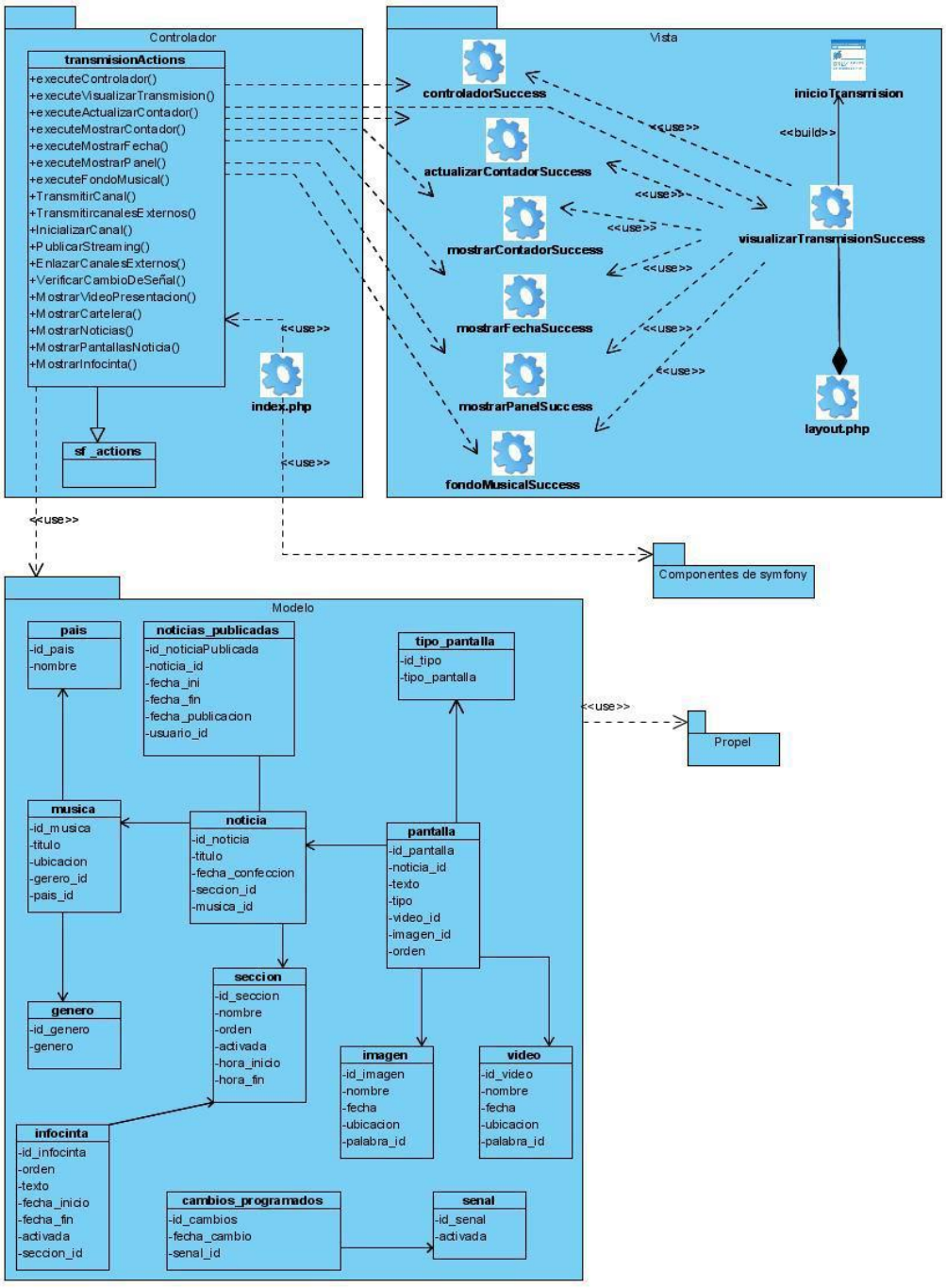


Figura 17 Diagrama de Clases: Subsistema Transmisión.

Los “componentes de Symfony” que se representan en los diagramas de clases anteriores, hacen referencias a los objetos que interactúan en una petición:

- **sfRequest** (objeto petición): almacena todos los elementos que forman la petición (parámetros, cookies, cabeceras, etc.)
- **sfResponse** (objeto respuesta): contiene las cabeceras de la respuesta y los contenidos. El contenido de este objeto se transforma en la respuesta HTML que se envía al usuario.
- **sfUser** (objeto sesión): esta clase dispone de un contenedor de parámetros que permite guardar cualquier atributo del usuario en el.
- **sfContext**: almacena una referencia a todos los objetos que forman el núcleo de Symfony y puede ser accedido desde cualquier punto de la aplicación.

2.5 Modelo de Implementación.

El modelo de implementación denota la implementación actual del sistema en términos de componentes y subsistemas de implementación. (11)

El modelo de implementación describe cómo se organizan los elementos de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros. (11)

Los componentes representan los elementos de un modelo dentro de un paquete, como son las clases en el modelo de diseño. Presentan estereotipos estándar como pueden ser:

- <<*executable*>> programas que se ejecutan en un nodo.
- <<*file*>> ficheros de datos o código fuente.
- <<*library*>> librerías estáticas o dinámicas.
- <<*table*>> tabla de base de datos.
- <<*document*>> documento.

En el siguiente modelo de implementación se representan componentes que hacen referencia a librerías y aplicaciones externas al propio framework de desarrollo utilizadas para la implementación de algunas

funcionalidades de la aplicación; tal es el caso de **LDAP**[1], para la autenticación mediante dominio, **TCPDF**[2] para la generación de reportes, **Propel** es framework **ORM**[3] para PHP que facilita la interacción con la Base de Datos, **VLC**[4] para la reproducción de archivos multimedia a través de un plug-in para el navegador web Mozilla y **cURL** es una herramienta para usar en un intérprete de comandos para transferir archivos con sintaxis URL.

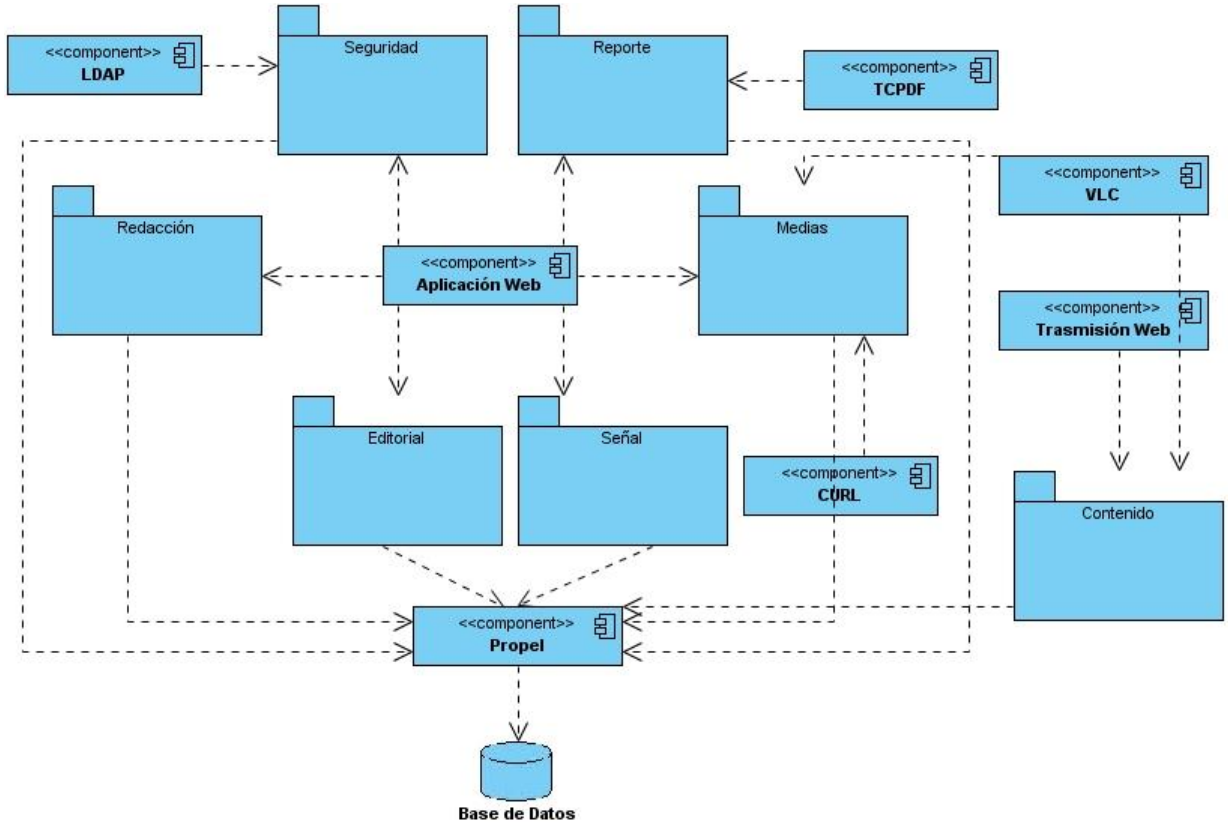


Figura 18 Modelo de Implementación.

[1] **LDAP** (Lightweight Directory Access Protocol, en español, Protocolo de Acceso a Directorios Ligeros) es un protocolo usado para acceder a "Servidores de Directorio".
[2] **TCPDF** es una clase de PHP5 para la generación de documentos PDF que no requiere extensiones externas.
[3] **ORM** (Object-Relational Mapping, en español, Mapeo Relacional de Objetos) es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional.
[4] **VLC media player** es un reproductor multimedia del proyecto VideoLAN.

2.6 Estilos de Programación.

El código desordenado es difícil de leer y las personas pueden perder interés si no son capaces de descifrar lo que el código intenta hacer. Resulta importante que los desarrolladores puedan entender el código rápidamente, lo que permite realizar modificaciones y mejoras en un periodo de tiempo reducido. El código es una forma de comunicación, y así como a alguien podría no querer leer un libro con errores ortográficos y mala puntuación, los programadores deben intentar escribir buen código de tal forma que sea fácil de entender y modificar por otros.

El estilo de programación se refiere a la forma en que se da formato al código fuente. Esto involucra la forma en que se ubican las llaves, se indenta el código y se utilizan los paréntesis, el uso de salto de líneas.

Indentación

Por indentación se entiende mover un bloque de texto hacia la derecha insertando espacios o tabuladores para separarlo del texto adyacente.

Para dar un ejemplo, a continuación se muestra un código no indentado, y uno correctamente indentado:

```
if($bandera){
$dominio = new ldap();
$this->nombre_completo = $dominio->Buscar($this->getRequestParameter('usuario'));
$this->check = true;}
else{
$this->mensaje = 1;}
```

El mismo código, pero aplicando indentación:

```
if($bandera)
{
        $dominio = new ldap();
        $this->nombre_completo = $dominio->Buscar($this->getRequestParameter('usuario'));
        $this->check = true;
}
else
{
        $this->mensaje = 1;
}
```

Existen varios estilos reconocidos por equipos que se dedican al desarrollo de software, a continuación se muestran algunos de ellos.

- Estilo K&R y BSD KNF

El estilo K&R es el más usado en el lenguaje C y PHP. Se trata de abrir la llave en la misma línea de declaración de la orden, indentando los siguientes pasos al mismo nivel que la llave y cerrando la llave en el mismo nivel que la declaración. Normalmente las tabulaciones en Windows son de 4 espacios, cuando las tabulaciones tienen 8 espacios se trata del estilo BSD o KNF, el código fuente del kernel de Unix y Linux está escrito en este estilo.

Se entiende mejor con el siguiente ejemplo:

```
function saludar($val){
    if ($val == 1) {
        echo "Hola!";
    } else {
        echo "Chao!";
    }
}
```

La ventaja de usar este estilo es que las llaves iniciales no necesitan ninguna línea extra para ellas solas, por lo que se ahorra espacio vertical de lectura.

La desventaja de este estilo es que las llaves de cierre si necesitan una línea exclusiva para ellas. Resulta difícil identificar el comienzo de las llaves de un bloque. Sin embargo es fácil identificar el comienzo de un bloque debido a la indentación a la derecha.

- Estilo Allman

Se trata de crear una nueva línea para las llaves, e indentar el código debajo de ellas. La llave de cierre tiene el mismo indentado que la de inicio.

```
function saludar($val)
{
    if($val == 1)
    {
        echo "Hola!";
    }
    else
```

```
    {
        echo "Chao!";
    }
}
```

Este estilo mantiene un código limpio y claro. Las llaves de inicio y fin coinciden en la misma columna, haciendo más fácil la identificación de cada bloque. Además, es más difícil olvidarse cerrar una llave cuando se identifica claramente la llave inicial.

La desventaja de este estilo, es que las llaves ocupan enteramente una línea, ocupando más espacio vertical de lectura. En monitores con resoluciones bajas (24 líneas) resulta difícil leer código fuente con este estilo aunque esto poco afecta en monitores con resoluciones más grandes.

Este es el estilo escogido para la codificación en la implementación de la aplicación a desarrollar, teniendo en cuenta además algunos estándares y reglas de codificación.

- Estilo GNU

Parecido al estilo Allman, se trata de poner las llaves en una nueva línea. La diferencia es que las llaves, deben estar indentadas por 2 espacios, y el código dentro de ellas debe estarlo por 2 espacios más.

```
function saludar($val)
{
    if($val == 1)
      {
        echo "Hola!";
      }
    else
      {
        echo "Chao!";
      }
}
```

Este estilo es usado por los proyectos de software GNU, los cuales formatean el código de algunas de sus aplicaciones a este estilo automáticamente, debido a que se ha declarado como un estándar dentro de esta comunidad.

2.7 Estándar de Codificación.

Un estándar de codificación es un conjunto de reglas de notación y nomenclatura, específicas de cada lenguaje de programación, que se usan y se siguen durante la fase de implementación (codificación) de una aplicación y reducen perceptiblemente el riesgo de que los desarrolladores introduzcan errores que no

son detectados por lo compiladores, reduciendo el tiempo y coste de las actividades de depuración y pruebas necesarias para la detección y corrección de los mismos. (12)

La elaboración y el empleo de estándares de codificación ofrecen numerosas ventajas a la hora de elaborar cualquier tipo de producto software y entre sus principales ventajas podemos mencionar:

- Se reduce la posibilidad de cometer errores.
- Se obtiene un código legible y comprensible.
- Mejora la comunicación entre los miembros del grupo de programadores del equipo de desarrollo.

Para la elaboración de un estándar de codificación se debe tener en cuenta, entre otros aspectos, las especificaciones y características del framework de desarrollo y el o los lenguajes de programación escogidos.

Comentarios, separadores, líneas y espacios en blancos		
Ubicación de comentarios	Inicio de una clase y al final de una línea o bloque de código	Describir brevemente el objetivo (que función realiza) de una clase previamente de su declaración se desea especificar la acción especifica de una línea de código, utilizar un breve comentario al final de esa línea.
Separador de instrucciones	Punto y coma.	Las instrucciones de código deben ser separadas al final de cada una de ellas y no debajo. Ej.: $pager->init(); $this->noticia_pager = $pager;
Líneas en blanco	Antes y después de cada función	Dejar una línea en blanco entre las declaraciones de cada función, se puede utilizar entre líneas cuando el bloque de código es extenso para dar claridad a su entendimiento.
Espacios en blanco	Entre operadores aritméticos y lógicos.	Usar un espacio en blanco entre los miembros de operaciones aritméticas y lógicas. Ej.: $criterio1 = new Criteria(); $result2 = NoticiaPeer::doSelect($criterio1);
Aspectos	Comentarios	Evitar el uso excesivo de comentarios, no hacerlo

generales		en forma de párrafo. Comentar de forma breve, precisa y concreta.
	Espacios en blanco	No usar espacios en blancos: - antes del punto y coma. - entre finales de líneas y comentarios.
Clases, objetos, funciones, atributos.		

Apariencia de clases y funciones	Teniendo en cuenta las especificaciones del framework.	Se establece usar la notación Camel Casing[1] para nombrar las clases y las funciones. Ej.:
		class redaccionActions extends sfActions
		{
		public function executeListadoNoticia()
		{
		***
		}
		}

Nombre de clases y objetos	Relacionados con el propósito de su función	Nombrar las clases de manera tal que con solo leerla se note su objetivo. Ej.:
		class seguridadActions extends sfActions
		{
		public function executeRegistrarUsuario(){}
		}

Apariencia de atributos	Letras minúsculas	Los atributos deben ser escritos en minúsculas y su nombre debe guardar relación con el valor que almacena. Ej.: $nombre = $u->getNombre(); $usuario = $u->getUsuario();
Aspectos generales	Nombre de clases, funciones y atributos.	Los nombres de estos miembros deben guardar estrecha relación con su objetivo, función y valor.
Variables y constantes		
Aspectos	Declaración de constantes y asignación a	Cada una de estas sentencias debe ser escrita una por cada línea. Ej.: $arreglo_senal = array(); $arreglo_senal[0] =" ";

[1] Camel Casing es una norma de notación que establece que las palabras compuestas debe ser escrita con mayúscula excepto la primera palabra. Ej: innerHtml.

generales	variables	
	Apariencia de constantes	Todas sus letras deben ser escritas en mayúscula. define('SF_APP', $app);
	Nombres de las variables y constantes	El nombre debe expresar su propósito con solo leerlas.

2.8 Conclusiones.

En este capítulo se realizó una breve descripción del diseño del sistema a implementar enfatizando los principales artefactos que constituyeron la base para el flujo de implementación, como son los diagramas de clases y el modelo de implementación; además se hace referencia a algunos de los estilos y estándar de codificación a emplear para la codificación.

Capítulo 3
Validación de la solución propuesta

3.1. Introducción.

La automatización de pruebas es una de las mayores ventajas de la programación desde el surgimiento de la orientación a objetos. En el desarrollo de las aplicaciones web, las pruebas aseguran la calidad de una aplicación aunque el desarrollo de nuevas versiones es muy activo. En este capítulo abarca todo lo relacionado con las herramientas y utilidades empleadas para la validación de la aplicación con el objetivo de garantizar el correcto funcionamiento de la misma.

3.2. Automatización de pruebas.

Para los desarrolladores de aplicaciones web es conocido el esfuerzo que supone probar correctamente una aplicación, resulta una tarea verdaderamente tediosa crear casos de prueba, ejecutarlos y analizar sus resultados. Debido a la variación constante de los requisitos de una aplicación y la refactorización continua del código es muy probable que aparezcan errores por lo que la automatización de pruebas es una recomendación útil para crear un entorno de desarrollo satisfactorio.

Este tipo de pruebas obligan a los programadores a crear pruebas en un formato estandarizado y muy rígido que pueda ser procesado por un framework de pruebas.

Las pruebas automatizadas pueden reemplazar la documentación técnica de la aplicación, ya que ilustran de forma clara el funcionamiento de la aplicación.

Las pruebas automatizadas comparan un resultado con la salida esperada para ese método. En otras palabras, evalúan “asertos” (del inglés, “assertions”, que son expresiones del tipo $a == 2. El valor de salida de un aserto es true o false, lo que determina si la prueba tiene éxito o falla. La palabra “aserto” es de uso común en las técnicas de automatización de pruebas.

3.3. Pruebas unitarias y funcionales.

Dependiendo del modelo de desarrollo de software empleado para el desarrollo de aplicaciones y su implementación estos dos tipos de pruebas pueden jugar un papel más o menos central, pero siempre importante.

En las pruebas unitarias se analiza una porción de código que pueda ser analizada de manera aislada, como por ejemplo funciones y métodos. A los que después de pasarle unos parámetros de entrada se debe obtener otros parámetros de salida claramente definidos, es decir, validan la forma en la que las funciones y métodos trabajan en cada caso particular. Las pruebas unitarias se encargan de un único caso cada vez, lo que significa que un único método puede necesitar varias pruebas unitarias si su funcionamiento varía en función del contexto.

Existen numerosos frameworks para crear pruebas unitarias, dentro de los que se destacan PhpUnit y SimpleTest. Symfony incluye su propio framework para pruebas llamado Lime, el cual se basa en la librería Test::More de Perl y es compatible con TAP, esto significa que los resultados de las pruebas se muestran con el formato definido en el “Test Anything Protocol”, creado para facilitar la lectura de los resultados de las pruebas.

Además de proporcionar soporte para pruebas unitarias, Lime tiene las siguientes ventajas:

- Las pruebas de Lime son fáciles de leer y sus resultados también lo son. En los sistemas operativos que lo soportan, los resultados de Lime utilizan diferentes colores para mostrar de forma clara la información más importante.
- El núcleo de Lime se valida mediante pruebas unitarias.
- Está escrito con PHP, es muy rápido y está bien diseñado internamente. Consta únicamente de un archivo, llamado lime.php, y no tiene ninguna dependencia.

En las pruebas funcionales se evalúa la aplicación como un todo y se comprueba que se alcanzan determinados resultados tras realizar una serie de acciones sobre la aplicación, validan partes de las aplicaciones. Estas pruebas simulan la navegación del usuario, realizan peticiones y comprueban los elementos de la respuesta, tal y como lo haría manualmente un usuario para validar que una determinada acción hace lo que se supone que tiene que hacer. En las pruebas funcionales, se ejecuta un escenario correspondiente a lo que se denomina un “caso de uso”.
Symfony dispone de un objeto especial, llamado sfBrowser, que actúa como un navegador que está accediendo a una aplicación, pero sin necesidad de utilizar un servidor web real. Este objeto permite el acceso directo a los objetos que forman cada petición (el objeto petición, el objeto sesión, el objeto

contexto y el objeto respuesta). Symfony también dispone de una extensión de esta clase llamada sfTestBrowser, que está especialmente diseñada para las pruebas funcionales y que tiene todas las características de sfBrowser, además de algunos métodos muy útiles para los asertos.

3.4. Herramientas para pruebas.

Las herramientas descritas anteriormente que incluye Symfony para realizar pruebas unitarias y funcionales no son suficientes para la mayoría de casos, sobre todo a la hora de probar las interacciones del lado cliente.

El principal inconveniente de estas técnicas es que no pueden simular el comportamiento de JavaScript. Si se definen interacciones muy complejas, como por ejemplo interacciones con Ajax, es necesario reproducir de forma exacta las pulsaciones de teclado que realiza el usuario y ejecutar los scripts de JavaScript. Normalmente, estas pruebas se hacen a mano, pero cuestan mucho tiempo y son propensas a cometer errores.

Una solución a estos problemas, es el empleo de ***Selenium*** que 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.

Las pruebas de Selenium se escriben en HTML, por lo que cada caso de prueba consiste en una página HTML, con una tabla de 3 columnas: comando, destino y valor.

La siguiente figura muestra un ejemplo de un caso de prueba donde se observan los elementos HTML presentes en un caso de prueba.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<title>Autenticarse</title>
</head>
<body>
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">Autenticarse</td></tr>
</thead><tbody>
<tr>
        <td>open</td>
        <td>/seguridad/index</td>
        <td></td>
</tr>
<tr>
        <td>type</td>
        <td>user</td>
        <td>hvillar</td>
</tr>
<tr>
        <td>type</td>
        <td>pass</td>
        <td>valor</td>
</tr>
<tr>
        <td>clickAndWait</td>
        <td>commit</td>
        <td></td>
</tr>
<tr>
        <td>assertTextPresent</td>
        <td>Bienvenido al sistema de administracion</td>
        <td></td>
</tr>
</tbody></table>
</body>
</html>
```

Figura 19 Caso de prueba.

Debido a que las pruebas de Selenium se crean con HTML, acaba siendo muy aburrido escribir todo ese código HTML. Existe una extensión de Selenium para Firefox que permite grabar todos los movimientos y acciones realizadas sobre una página y guardarlos como una prueba. Una vez realizados todos los movimientos y añadidos todos los comandos, se pueden guardar en un archivo HTML para añadirlo al conjunto de pruebas. Este conjunto o suite de pruebas se define en archivo *TestSuite.html,* el cual se encarga de guardar las referencias a cada uno de los archivos de pruebas guardados independientemente y mostrarlos en una lista.

```
<table id="suiteTable"     cellpadding="1"
       cellspacing="1"
       border="1"
       class="selenium">
    <tbody>
            <tr><td><b>Test Suite</b></td></tr>
        <tr><td><a href='./TestEjemplo.html'>Autenticarse</a></td></tr>
        <tr><td><a href='./TestEjemplo2.html'>Crear un Usuario</a></td></tr
        ...
    </tbody>
</table>
```

Figura 20 Estructura de TestSuite.html

La extensión de Firefox incluso permite ejecutar las pruebas de Selenium que se han creado con la extensión.

3.5. Diseño de pruebas.

Para validar el funcionamiento de la aplicación se realizan algunas pruebas funcionales apoyadas en el uso de Selenium debido a que permite probar el funcionamiento de varios formularios varias veces seguidas. Las pruebas que realiza son como las que haría cualquier usuario desde un navegador, con la ventaja de que las hace mucho mas rápido y evita el trabajo repetitivo de probar una y otra vez lo mismo "a mano".

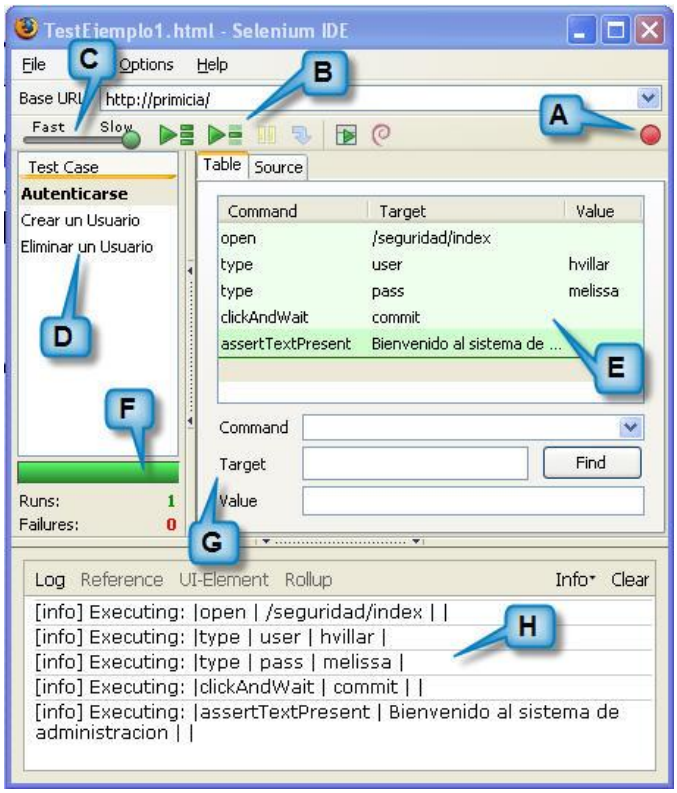
A continuación se relaciona uno de los casos de prueba implementados para la validación de la propuesta que consiste en chequear el funcionamiento de la autenticación del sistema.

Caso de prueba #1.

Caso de prueba.	Autenticar Usuario
Condiciones.	El usuario debe estar registrado en la base de datos como usuario del sistema para entrar al sistema. La contraseña de un usuario registrado debe corresponder con la del dominio.
Datos de entrada.	Usuario registrado y contraseña.
Resultados esperados.	1. En caso de ingresar la contraseña correcta para un usuario registrado, el sistema debe mostrar la página de bienvenida.
	2. Si se ingresa una contraseña incorrecta el sistema debe mostrar un texto de error y brindar la posibilidad de volver a ingresar sus datos.

Resultados obtenidos.	1. La prueba es satisfactoria. Al autenticarse con un usuario registrado en la base de datos y su contraseña del dominio correcta, el sistema se redirecciona hacia la pagina de bienvenida.
	2. La prueba es satisfactoria. Al autenticarse con un usuario registrado en la base de datos y su contraseña del dominio es incorrecta, el sistema muestra un mensaje de error diciendo que la contraseña es incorrecta y dala posibilidad de ingresar nuevamente los datos.
Observaciones.	Al realizar la prueba con la herramienta Selenium se obtuvo el resultado correcto.

En la figura que se muestra el resultado del Caso de prueba Autenticarse y sirve de ejemplo para relacionar los elementos presentes a la hora de ejecutar una prueba con la herramienta Selenium.



A: Graba los movimientos de la navegación.

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: Determinar la velocidad de ejecución de pruebas para que el usuario pueda apreciar la prueba.

D: Suite de Pruebas, lista todas las pruebas presentes en la suite.

E: Datos de la prueba presentes en el código de cada una.

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

G: Permita cambiar los valores de una prueba en tiempo de ejecución.

H: Log. Registra los eventos y sucesos de cada prueba.

Figura 21 Elementos de Selenium IDE.

3.6. Conclusiones.

En este capitulo se hizo referencia a los principales elementos relacionados con los tipos de pruebas concernientes a la fase de implementación y algunas de las herramientas utilizadas. Es necesario señalar que el caso de ejemplo mostrado anteriormente no es el único que se le realizó a la solución, el objetivo principal de esta sección es precisamente conocer como se manejan estos conceptos de pruebas en el contexto de la implementación de una aplicación.

Conclusiones

El desarrollo de esta plataforma brindó la posibilidad de documentar parte de las tareas propuestas para el desarrollo de la aplicación y la implementación de la Plataforma de Televisión Informativa Primicia. Se puede afirmar que con el desarrollo de esta investigación práctica:

- Se logró un incremento en los conocimientos acerca de algunas de las técnicas y tendencias de la programación actuales
- Se caracterizó el empleo de estándares de codificación
- Se modelaron y ejecutaron algunos casos de pruebas que certificaron el funcionamiento de la aplicación desarrollada
- Se implementaron de los subsistemas de Administración y Transmisión de la Plataforma de Televisión Primicia

De forma general se puede concluir que se cumplieron satisfactoriamente todos objetivos enunciados desde el inicio de la investigación práctica, por lo que se le dio solución al problema científico planteado.

Recomendaciones

Luego de terminada la investigación, se recomienda:

- Realizar un estudio de posibles mejoras para facilitar la adaptabilidad del sistema a las necesidades de diferentes clientes.
- La implantación de la aplicación en diferentes instituciones nacionales o internacionales.
- Optimizar la codificación de la aplicación y actualizar la versión del framework empleado.

Referencias Bibliográficas

1. **Pressman, Roger S.** *Ingenieria de software. Un enfoque practico.* s.l. : McGraw-Hill.

2. *Aplicación Web "Sistema de Rehabilitación Integral".* **Llanes Guerra, Néstor.** 2009.

3. **Liceaga, Ramón Castro.** *Informática VI Programación Orientada a objetos.* [ppt]

4. **Pérez, Javier Eguíluz.** *Introduccion a XHTML.* 2008.

5. **Pérez, Javier Eguíluz.** *Introducción a Javascript.* 2008.

6. **Pérez, Javier Eguíluz.** *Introduccion a CSS.* 2008.

7. **Pérez, Javier Eguíluz.** *Introduucion a AJAX.* 2007.

8. *Reutilización basado en la tecnología de objetos.* **Berrospi, Eloy Bernabé.**

9. **Potencier, Fabien y Zaninotto, François.** *Symfony, la guia definitiva.* 2007.

10. **Larman, Craig.** *UML y Patrones.Introduccion al análisis y diseño orientado a objetos.* s.l. : Prentice Hall, 1998.

11. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* s.l. : Addison Wesley, 1999.

12. **Rodríguez, Yaquelin Y. Morales y Remedio, Adisneisy C. Román.** *Propuesta de una Guía para estandarizar la codificación en la Universidad.* C.Habana : UCI, 2007.

Bibliografía

1. **Pressman, Roger S.** *Ingenieria de software. Un enfoque practico.* s.l. : McGraw-Hill.

2. *Aplicación Web "Sistema de Rehabilitación Integral".* **Llanes Guerra, Néstor.** 2009.

3. **Liceaga, Ramón Castro.** *Informática VI Programación Orientada a objetos.* [ppt]

4. **Pérez, Javier Eguíluz.** *Introduccion a XHTML.* 2008.

5. **Pérez, Javier Eguíluz.** *Introducción a Javascript.* 2008.

6. **Pérez, Javier Eguíluz.** *Introduccion a CSS.* 2008.

7. **Pérez, Javier Eguíluz.** *Introduucion a AJAX.* 2007.

8. *Reutilización basado en la tecnología de objetos.* **Berrospi, Eloy Bernabé.**

9. **Potencier, Fabien y Zaninotto, François.** *Symfony, la guia definitiva.* 2007.

10. **Larman, Craig.** *UML y Patrones.Introduccion al análisis y diseño orientado a objetos.* s.l. : Prentice Hall, 1998.

11. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* s.l. : Addison Wesley, 1999.

12. **Rodríguez, Yaquelin Y. Morales y Remedio, Adisneisy C. Román.** *Propuesta de una Guía para estandarizar la codificación en la Universidad.* C.Habana : UCI, 2007.

13. **Valdés, Damián Pérez.** Maestros del Web. *Los diferentes lenguajes de programación para la web.* [En línea] 2007. http://www.maestrosdelweb.com/principiantes/los-diferentes-lenguajes-de-programacion-para-la-web/.

14. **Pereda.** ¿Que es una Aplicacion Web? *WordPress.com.* [En línea] http://es.wordpress.com/tag/aplicacion-internet/.

15. **Olivares Tamayo, Jorge Daniel y Rey Almaguer, Bernardo.** *Desarrollo del Canal Informativo del Ministerio del Poder Popular para la Energía y Petróleo de Venezuela: Subsistema de Administración.* Ciudad de la Habana : s.n., 2008.

16. **Morales Rodríguez, Yaquelin y Román Remedio, Adisneisy.** *Propuesta de una Guía para estandarizar la codificación en la Universidad.* s.l. : UCI, 2007.

17. **Larman, Craig.** *UML y Patrones.* Mexico : Prentice Hall, 1999.

18. **Hristov, Alexander.** *Manual de Estilo de Programación.* s.l. : Planetalia, 2007.

19. **Hernández García, Ruber y Montaner Hernández, Yunior.** *Sistema Automatizado de Teletexto para la Plataforma de Televisión Digital Satelital Cubana.* Ciudad de la Habana : s.n., 2007.

20. **García, Ricardo Marmolejo.** *Aplicaciones Web con UML.*

21. **Bustio, Jose Andres Hernandez.** *Desarrollo del Canal Informativo del Ministerio del Poder Popular para la Energía y Petróleo de Venezuela: Subsistema de Transmisión.* Ciudad de la Habana : s.n., 2008.

22. **Colectivo de autores.** *Livre Blanc: Frameworks PHP pour l´enterprise.* 2008.