

# Universidad de las Ciencias Informáticas



*“Propuesta de Arquitectura de Software para el desarrollo de portales web orientados al Ajedrez.”*

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

Autor: **Eladio Eladiovich Avila Bagdasarova**

Tutor(a): **Ing. Sandra Gutiérrez Secades**

Co-tutor: **Ing. Alison Muñoz Capote**

Consultante: **MSc. Isbel Herrera del Sol**

Junio 2009

## DECLARACIÓN DE AUTORÍA

Declaro ser el único autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los \_\_\_\_ días del mes de \_\_\_\_\_ del año \_\_\_\_\_.

Eladio Eladiovich Ávila Bagdasarova

---

Firma del Autor

Ing. Sandra Gutiérrez Secades

---

Firma del Tutor

## **Agradecimientos**

A Dios por su maravillosa creación, y porque Dios es amor... gracias por aceptarme.

A mis padres, Eladio y Karina, por ser los mejores... son el espejo donde me miro, gracias por la educación que me brindan.

A mis segundos padres, Vladimir y Marisol, por su apoyo y comprensión.

A mis hermanos, Danilo y Martín, por la alegría que me proporcionan, los quiero mucho.

A mi familia, por el calor que siempre me transmiten... gracias por preocuparse.

A la revolución por permitir que mis padres se conocieran... por viabilizar mi superación hasta el límite del deseo, gracias por toda su obra.

A la Universidad de las Ciencias Informáticas por sustentar mis estudios y darme la oportunidad de la realización profesional y como estudiante.

A la dirección de la Facultad 5, por las oportunidades que me dieron, gracias por la confianza depositada.

A la Decana, Mayra, por ser como una madre y una amiga, mi admiración y respeto.

A la Secretaria Docente, Andrea, por su preocupación y su amistad.

A mis profesores, por guiar mi desarrollo como estudiante, gracias por el tiempo dedicado.

A mis tutores Sandra, Alison e Isbel, por responder siempre, gracias por su apoyo.

Al presidente de la Cátedra de Ajedrez, por tener siempre la puerta abierta, por ser mi amigo y consejero.

A mis amigos, por hacerlo todo más ameno.

## **Dedicatoria**

A la memoria de mis abuelos y bisabuelos, los que ya no están... los recuerdo con cariño.

## **Resumen**

En el presente trabajo se pretende diseñar una propuesta de Arquitectura de Software para el desarrollo de portales web orientados al ajedrez que facilite una adecuada comprensión del sistema a sus desarrolladores y que al mismo tiempo contribuya a construir portales (de ajedrez) con una alta aceptación. Para ello se toman ideas de los principales Sistemas de Gestión de Contenidos (CMS por su sigla en inglés) de código abierto que existen en la actualidad; con el objetivo de comprender su arquitectura, funcionamiento y protocolo para gestionar la información. Además se estudian los principales portales de ajedrez que existen dentro y fuera de Cuba, de forma tal que la arquitectura a realizar maneje los componentes necesarios exclusivamente para el ajedrez digital.

## **Palabras Claves**

Software Libre, Ajedrez, Arquitectura de Software, CMS.

## Tabla de contenido

<b>Agradecimientos</b> .....	3
<b>Dedicatoria</b> .....	3
<b>Resumen</b> .....	4
<b>Introducción</b> .....	11
<b>Capítulo 1 Fundamentación teórica</b> .....	17
<b>1.1 Introducción</b> .....	17
<b>1.2 Portal Web</b> .....	17
1.2.1 Estándares Web.....	18
1.2.2 Web 2.0 vs Web 1.0 .....	18
<b>1.3 Ajedrez</b> .....	20
1.3.1 Notación portable de juego.....	21
<b>1.4 Portales de ajedrez a nivel internacional</b> .....	23
1.4.1 Internet Chess Club.....	23
1.4.2 ICCF (International Correspondence Chess Federation) .....	23
1.4.3 ChessGames.Com .....	24
1.4.4 World Chess Federation.....	24
1.4.5 Corus Chess .....	25
1.4.6 ChessBase.....	25
<b>1.5 Portales de ajedrez a nivel nacional</b> .....	25
1.5.1 Portal “Infodrez”.....	25
<b>1.6 Resultado del análisis a los principales portales de ajedrez a nivel internacional</b> .....	26
1.6.1 Propuesta de módulos con que podría contar un portal orientado al ajedrez.....	26

1.6.2 Propuesta de Componentes con que debería contar un portal orientado al ajedrez .....	29
<b>1.7 Contenido Web</b> .....	30
<b>1.8 Gestión de Contenidos</b> .....	31
<b>1.9 Sistemas de Gestión de Contenidos</b> .....	31
1.9.1 ¿Qué es un CMS?.....	32
<b>1.10 Software Libre</b> .....	34
1.10.1 El Software Libre en Cuba.....	35
<b>1.11 Calidad de Software</b> .....	37
1.11.1 Calidad y Arquitectura de Software .....	37
<b>1.12 Arquitectura de Software</b> .....	37
1.12.1 ¿Por qué es necesaria la Arquitectura de Software? .....	38
1.12.2 Estilos en la Arquitectura de Software .....	39
1.12.3 Patrones de diseño.....	44
1.12.4 Patrones arquitectónicos .....	45
1.12.5 Estilos y patrones arquitectónicos y de diseño .....	48
1.12.6 Representación de la Arquitectura de Software .....	49
1.12.7 El Modelo de “4 + 1” Vistas de la Arquitectura de Software .....	50
1.12.8 El Rol del Arquitecto de Software .....	51
<b>1.13 Metodologías de desarrollo de software</b> .....	53
1.13.1 Metodologías pesadas .....	53
1.13.2 Metodologías ágiles.....	58
1.13.3 Comparación entre las metodologías ágiles y las pesadas.....	62
1.13.4 Fundamentación de la Metodología a utilizar.....	63
<b>1.14 Proceso Unificado de Desarrollo y Arquitectura de Software</b> .....	63

1.14.1 Casos de uso y arquitectura .....	63
1.14.2 Línea base de la arquitectura .....	66
1.14.3 Descripción de la arquitectura .....	67
1.14.4 Vistas de la arquitectura según RUP .....	67
<b>1.15 Selección de CMS para el estudio .....</b>	<b>68</b>
1.15.1 Joomla!.....	69
1.15.2 Drupal.....	70
1.15.3 e107 .....	70
1.15.4 Comparación entre los CMS estudiados.....	71
1.15.5 Elección de un CMS para el ajedrez.....	73
<b>1.16 Plataforma LAMP.....</b>	<b>73</b>
1.16.1 Sistema operativo Debian GNU/Linux .....	74
1.16.2 Servidor Web Apache.....	75
1.16.3 Gestores de Bases de Datos.....	76
1.16.4 Lenguaje de programación PHP .....	77
<b>1.17 Lenguajes Utilizados.....</b>	<b>78</b>
1.17.1 HTML .....	78
1.17.2 XML.....	80
1.17.3 XHTML.....	81
1.17.4 JavaScript .....	82
1.17.5 CSS.....	83
1.17.6 Lenguaje Unificado de Modelado .....	84
<b>1.18 Herramientas de soporte .....</b>	<b>86</b>
1.18.1 ab (Apache Benchmark).....	86

1.18.2 Herramientas CASE .....	86
<b>1.19 Conclusiones.....</b>	<b>87</b>
<b>Capítulo 2 Línea Base de la Arquitectura.....</b>	<b>88</b>
<b>2.1 Introducción .....</b>	<b>88</b>
<b>2.2 Arquitectura Propuesta .....</b>	<b>88</b>
<b>2.3 ¿Por qué Drupal?.....</b>	<b>89</b>
<b>2.4 Arquitectura de Drupal .....</b>	<b>92</b>
2.4.1 Patrones arquitectónicos .....	96
2.4.2 API de Drupal.....	98
2.4.3 Creación de Módulos.....	99
2.4.4 Creación de Temas .....	99
2.4.5 Código de Drupal .....	101
2.4.6 Patrones de diseño utilizados en Drupal.....	103
2.4.7 Editores de texto .....	104
<b>2.5 Prototipo Ejecutable de Arquitectura .....</b>	<b>105</b>
2.5.1 Sitio Oficial del Ajedrez Cubano .....	105
<b>2.6 Componentes imprescindibles dentro de la arquitectura propuesta .....</b>	<b>105</b>
2.6.1 Gestor de Cuadros Sinópticos.....	105
2.6.2 Visor de Partidas .....	107
2.6.3 Gestor de Diagramas .....	107
<b>2.7 Vistas de la Arquitectura propuesta .....</b>	<b>107</b>
2.7.1 Vista de Casos de Uso .....	107
2.7.2 Vista Lógica.....	108
2.7.3 Vista de Implementación .....	109



2.7.4 Vista de Despliegue.....	110
<b>2.8 Conclusiones.....</b>	<b>111</b>
<b>Capítulo 3 Descripción de la Arquitectura .....</b>	<b>112</b>
<b>3.1 Introducción .....</b>	<b>112</b>
<b>3.2 Representación arquitectónica .....</b>	<b>112</b>
<b>3.3 Objetivos y Restricciones Arquitectónicas .....</b>	<b>113</b>
3.3.1 Apariencia o Interfaz Externa .....	113
3.3.2 Usabilidad .....	113
3.3.3 Rendimiento .....	113
3.3.4 Soporte.....	113
3.3.5 Portabilidad, Escalabilidad, Reusabilidad .....	114
3.3.6 Hardware.....	114
3.3.7 Software .....	115
3.3.8 Seguridad.....	115
3.3.9 Confiabilidad, Integridad, Fiabilidad.....	116
3.3.10 Legales.....	116
3.3.11 Redes.....	116
<b>3.4 Vistas .....</b>	<b>116</b>
<b>3.5 Calidad .....</b>	<b>116</b>
<b>3.6 Conclusiones.....</b>	<b>116</b>
<b>Capítulo 4 Evaluación de la Arquitectura.....</b>	<b>117</b>
<b>Resumen.....</b>	<b>117</b>
<b>4.1 Introducción .....</b>	<b>118</b>
<b>4.2 Estudio del estado del arte.....</b>	<b>118</b>

<b>4.3 Propuesta de Procedimiento</b> .....	125
<b>4.4 Evaluación de la arquitectura propuesta</b> .....	127
<b>4.5 Conclusiones</b> .....	128
<b>Conclusiones</b> .....	129
<b>Recomendaciones</b> .....	130
<b>Referencias Bibliográficas</b> .....	131
<b>Bibliografía Consultada</b> .....	136
<b>Anexos</b> .....	137
<b>Glosario</b> .....	140

## Introducción

A raíz de las palabras pronunciadas por el Comandante en Jefe Fidel Castro Ruz en la Simultánea Gigante de Ajedrez realizada en la Plaza de la Revolución el 7 de Diciembre de 2002 incentivando la masificación del ajedrez:

*“Masificar el ajedrez, te aseguro, colocaría a este país con mucha más capacidad de pensar, más eficiente; es como saber una asignatura básica...”* (1)

La UCI (Universidad de las Ciencias Informáticas) ha dado incontables pasos pro a la realización de este sueño.

Inicialmente se elaboró un convenio de colaboración entre el ISLA (Instituto Superior Latinoamericano de Ajedrez) y la UCI el cual se firmó en mayo del 2003, que permitió entre otros aspectos la constitución de la Cátedra Honorífica de Ajedrez “Remberto A. Fernández González” de la UCI en abril del 2004. En el seno de dicha institución la comunidad ajedrecística fue creciendo, y surgió la necesidad de crear un ambiente más amplio donde interactuar. Fiel a una de las tareas fundamentales de la UCI como lo es la informatización de la sociedad, la cátedra lanzó un concurso para la realización de un portal donde los internautas pudieran jugar al ajedrez online e informarse del acontecer. Fue así como un proyecto de inteligencia artificial de la Facultad 3 desarrolló UCIdrez; su mérito fundamental fue dar el primer paso en la búsqueda de una solución web para el juego ciencia, pero lamentablemente carecía de una visión a largo plazo y pronto resaltaron las inconformidades por parte de los usuarios. UCIdrez se alejaba de cumplir con las normas y estándares internacionales para el diseño y desarrollo de aplicaciones web (W3C), se utilizó para su confección la tecnología ASP.NET (no es una tecnología multiplataforma puesto a que solo puede ser montada en Servidores Windows).

En marzo del 2006, la facultad 10 de la UCI realizó la total migración hacia el sistema operativo GNU/Linux lo que afectó su desarrollo en materia de ajedrez, dado a que el principal punto de contacto entre los trebejistas de la universidad –UCIdrez-, no era compatible con navegadores web libres como Mozilla. Al percatarse de la situación, el líder del proyecto Unicornios (Ing. Abel Meneses Abad) se dirigió a la Cátedra Honorífica de Ajedrez “Remberto A. Fernández González” en búsqueda de una solución, donde descubre que UCIdrez era una aplicación sin equipo de soporte técnico y observa además el descontento que producía para la cátedra, por la gran cantidad de fallas que este tenía (2). A raíz de este

crucial encuentro se fomentó el desarrollo de un equipo de estudiantes que valorara la situación técnica e investigara junto a los miembros de la cátedra las mejores opciones para tener una plataforma de ajedrez visible desde cualquier sistema operativo; fue así como surgió el proyecto “Infodrez” (cuyo objetivo en la actualidad consiste en informatizar el ajedrez en Cuba). Luego de un intenso trabajo se alcanzó la primera versión del “Juego Online”: módulo que establece y maneja un protocolo para garantizar que se efectúe una partida entre dos estaciones de trabajo, aprovechando además para la confección de un nuevo portal, las bondades del CMS “e107”; en su conjunto, constituyen la web oficial de la cátedra honorífica de ajedrez. Después de su puesta en práctica se detectaron algunas inconformidades que dificultan la integridad de esta aplicación web:

- ✓ Se pudo identificar que existe poca información en los contenidos de las noticias dado a que el gestor no permite la creación de tablas de forma dinámica (con los datos de los torneos), ni la inserción de un visor en el cual los usuarios puedan ver gráficamente las partidas publicadas.
- ✓ Como un CMS está hecho para soportar una cantidad de contenido diversa y gigantesca, surge el inconveniente de que una gran parte de su cuerpo no es aplicable al ajedrez, por lo que se tiene espacio de memoria utilizado innecesariamente, trayendo como problema asociado la demora para ser cargado desde una estación en otra red (siendo en Cuba esto un problema considerable por el limitado ancho de banda existente).

En el mes de Junio del 2005, en documento titulado “Alianza Estratégica UCI – INDER” se proponen una serie de proyectos, donde el proyecto número 6 “Desarrollo Integral del Ajedrez” coordinado por el MsC. Gerardo Lebreo Zarragoitía (Rector del ISLA), cuyo objetivo es *“Desarrollar en colaboración con la UCI, servicios y productos informáticos, y realizar eventos y actividades, que favorezcan dimensionar y sustentar, nacional e internacionalmente, el Programa de Masificación del Ajedrez”* fue aprobado por ambas partes. (3)

Entre sus principales acciones se plantean:

- ✓ Apoyar integralmente la Cátedra de Ajedrez de la UCI en sus proyecciones de trabajo y de desarrollo del Ajedrez en este Centro.
- ✓ Desarrollar servicios informáticos y productos de Ajedrez que contribuyan al desarrollo de esta disciplina y que puedan tener un valor comercial.
  - Software de enseñanza y de entrenamiento deportivo.

- Zonas de juego.
- Cursos en formato digital (multimedia).
- Libros en formato digital (multimedia).
- ✓ Convertir al Instituto Superior Latinoamericano de Ajedrez (ISLA) en una universidad virtual, capaz de desarrollar cursos y entrenamientos a distancia en tiempo real.
- ✓ Capacitar a especialistas del Ajedrez, profesores, entrenadores y árbitros.
- ✓ Garantizar el entrenamiento deportivo ajedrecístico en el Alto Rendimiento y en el desarrollo de niños y jóvenes talentos, acorde con el desarrollo contemporáneo de esta disciplina.
  - Desarrollo de concentrados deportivos.
  - Desarrollo de entrenamientos específicos.
- ✓ Realizar investigaciones para determinar la influencia de la práctica del ajedrez en el pensamiento lógico y en los resultados de los estudiantes de la UCI.

En carta fechada 23 de mayo del 2006, el presidente del INDER (Instituto Nacional de Deportes, Educación Física y Recreación) Christian Jiménez Molina agradece al Rector de la UCI Melchor Félix Gil Morell el apoyo dado por la universidad a la preparación del Equipo Olímpico de Ajedrez y expone respecto al desarrollo futuro del juego ciencia y el papel de la UCI en esa dirección: *“Considero que la Universidad de las Ciencias Informáticas puede ser líder en el país en el Ajedrez electrónico...”* (4)

Dándole seguimiento a la alianza estratégica UCI – INDER, en el marco del convenio de colaboración ISLA – UCI se tomaron, entre otros, los siguientes acuerdos: (5)

- ✓ Rediseñar acorde a las más potentes herramientas informáticas, la página oficial del ISLA.
- ✓ Diseñar sitios Web: Grandes Maestros, Arbitraje y Torneos de Cuba (Eventos Internacionales y Nacionales).

Al inicio de la investigación en la red cubana existían tres sitios de ajedrez:

1. Sitio del Ajedrez en Cuba: <http://www.cuba.cu/ajedrez/>
2. Sitio del Torneo Capablanca in Memoriam: <http://www.inder.cu/capablanca/>
3. Sitio de la Federación Cubana de Ajedrez por Correspondencia: <http://www.inder.cu/fecap/>

En un análisis estructural de los mismos fueron clasificados como Web 1.0:

Es un tipo de Web estática; con tecnología asociada: HTML, GIF. La frecuencia de actualización de los contenidos es limitada, debido a que dependen para su mantenimiento de una única persona, aquella que tiene acceso al FTP, lo que produce gran descontento para los usuarios visitantes así como para los directivos del ajedrez cubano.

Dada la situación anterior el proyecto "Infodrez" (cuyo principal cliente es el ISLA) de conjunto con la Cátedra Honorífica de Ajedrez "Remberto A. Fernández González", dándole cumplimiento a los acuerdos planteados, decidió abrir una línea para el desarrollo de portales orientados al ajedrez.

¿Cómo lograr que los desarrolladores de estos portales tengan una idea clara de lo que están desarrollando?

Un sistema software grande y complejo requiere una arquitectura para que los desarrolladores puedan progresar hasta tener una visión común. Un sistema software es difícil de abarcar visualmente porque no existe en un mundo de tres dimensiones. Es a menudo único y sin precedente en determinados aspectos. Suele utilizar tecnología poco probada o una mezcla de tecnologías nuevas. Tampoco es raro que el sistema lleve a sus últimos límites la tecnología existente. (6)

Uno de los principios de las metodologías modernas de desarrollo de software es priorizar la definición, el diseño, la implementación y la evaluación de la arquitectura del software (en lo adelante AS), que es como se conoce al esqueleto o estructura del sistema. (7)

La esencia del principio de priorizar la arquitectura es dedicar los mínimos esfuerzos a garantizar la corrección de las partes más importantes, costosas e indefinidas del sistema, y cuyo prototipo permita una demostración tangible de la viabilidad del proyecto. (7)

Se necesita una arquitectura para: (6)

- ✓ Comprender el sistema.
- ✓ Organizar el desarrollo.
- ✓ Fomentar la reutilización.
- ✓ Hacer evolucionar el sistema.
- ✓

En (6) se define la arquitectura como el *"conjunto de decisiones significativas acerca de la organización de un sistema software, la selección acerca de los elementos estructurales a partir de los cuales se compone el sistema, y las interfaces entre ellos, junto con su comportamiento, tal y como se especifica en las*

*colaboraciones entre esos elementos, la composición de estos elementos estructurales y de comportamiento en subsistemas progresivamente mayores, y el estilo arquitectónico que guía esta organización: estos elementos y sus interfaces, sus colaboraciones y su composición. La arquitectura del software se interesa no sólo por la estructura y el comportamiento, sino también por las restricciones y compromisos de uso, funcionalidad, funcionamiento, flexibilidad al cambio, reutilización, comprensión, economía y tecnología, así como por aspectos estéticos.”*

La definición “oficial” de AS se ha acordado que sea la que brinda el documento de IEEE Std 1471-2000, adoptada también por Microsoft:

*La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución.*

(8)

Es evidente la necesidad de la representación arquitectónica en los sistemas de software, en primer lugar las representaciones arquitectónicas elevan el nivel de abstracción, facilitando la comprensión de sistemas complejos. En segundo lugar hace que aumente la posibilidad de reutilizar los distintos componentes o elementos de software y que se pueda reutilizar hasta la misma arquitectura.

Lo antes expuesto llevó a definir el siguiente **problema científico** para la presente investigación:

¿Cuál es la Arquitectura de Software apropiada para el desarrollo de portales web orientados al ajedrez que permita una adecuada comprensión del sistema software a sus desarrolladores y que al mismo tiempo contribuya a la confección de portales web (para el ajedrez) con una alta aceptación?

Para dar respuesta a esta interrogante, se definió como **objeto de estudio**: Arquitecturas de plataformas web; y como **campo de acción**: Arquitecturas de plataformas web para los contenidos del ajedrez.

Como **idea a defender** se vaticina que si se define una arquitectura teniendo en cuenta los principales componentes que presentan los portales de ajedrez y centrada en atributos de calidad como portabilidad, escalabilidad, seguridad y rendimiento, se obtendrá una arquitectura mediante la cual se logren construir portales (para el ajedrez) con una alta aceptación.

Con el propósito de resolver el problema planteado, se trazó como **objetivo general** de este trabajo diseñar una AS teniendo en cuenta los principales componentes que presentan los portales de ajedrez y centrada en atributos de calidad como portabilidad, escalabilidad, seguridad y rendimiento. De aquí se derivan **objetivos** más **específicos** como son:

1. Describir la arquitectura del sistema a través de las diferentes vistas de la arquitectura según la metodología utilizada.
2. Determinar los estilos y patrones arquitectónicos que se emplearán durante el proceso de desarrollo y diseño del sistema.
3. Definir componentes y funcionalidades reusables que agilicen el desarrollo de la aplicación.
4. Evaluar la arquitectura propuesta.

Para dar cumplimiento a los objetivos se plantean un conjunto de **tareas**:

1. Investigación de las diferentes propuestas arquitectónicas a partir de consultar la bibliografía especializada.
2. Comparación de las distintas propuestas arquitectónicas consultadas, para obtener la más adecuada para el desarrollo de la aplicación.
3. Analizar los componentes fundamentales que presentan los portales de ajedrez más difundidos en la actualidad.
4. Identificación de los patrones a utilizar, fundamentación de los mismos y su aplicación.
5. Definir herramientas y tecnologías a utilizar para el desarrollo.
6. Creación de la documentación de la Arquitectura del Sistema.
7. Realizar un estudio sobre los métodos de evaluación de arquitecturas de software.



# Capítulo 1 Fundamentación teórica

## 1.1 Introducción

En el presente capítulo se realiza un estudio del estado del arte en materia de arquitectura de software. Se brinda además, una breve descripción de los estilos arquitectónicos existentes y más usados, la metodología, y herramientas a emplear para desarrollar la arquitectura de un sistema de software. Además se estudian los principales portales de ajedrez que existen dentro y fuera de Cuba, de forma tal que la arquitectura a realizar maneje los componentes necesarios exclusivamente para el ajedrez digital.

## 1.2 Portal Web

La idea de portal surgió en 1994 con el nacimiento del primer navegador denominado Netscape Navigator en un momento en el que muy pocos utilizaban Internet. Poco después, David Filo y Jerry Yang, dos estudiantes de la Universidad de Standford, empezaron a clasificar las direcciones de webs interesantes para tenerlas todas bien ubicadas. Desde un inicio se lo tomaron como un hobby personal pero en poco menos de un año y, viendo las grandes posibilidades de negocio, decidieron constituir Yahoo! (9)

Un *Portal* se entiende como “la página Web que agrega contenidos y funcionalidades, organizados de tal manera que facilitan la navegación y proporcionan al usuario un punto de entrada en la Red con un amplio abanico de opciones”. En ese punto de entrada, el usuario ve concentrados todos los servicios y productos que ofrece, de forma que le permite hacer cuanto necesita sin tener que salir de dicho *website*. Es una forma de captar clientes ya que, el objetivo empresarial de cualquier Portal es conseguir que su página genere lealtad entre los usuarios, en definitiva, maximizar el tiempo que permanece en sus páginas, antes de saltar a otro destino en la Red y asegurarse que vuelve de manera sucesiva. (10)

### Utilidad de los Portales

Los portales pueden ser aprovechados de muchas formas, todo depende de la utilidad que se le quiera dar. Generalmente recibe tres utilidades principales, estas son: Contenido, Comercio y Comunidad. (11)

- ✓ Ofrecer todo tipo de contenidos e informaciones.
- ✓ Comercialización de los contenidos.
- ✓ Formación de comunidades virtuales.

### **1.2.1 Estándares Web**

Un estándar es un conjunto de reglas normalizadas que describen los requisitos que deben ser cumplidos por un producto, proceso o servicio, con el objetivo de establecer un mecanismo base para permitir que distintos elementos hardware o software que lo utilicen, sean compatibles entre sí.

El W3C (World Wide Web Consortium), organización independiente y neutral, desarrolla estándares relacionados con la Web también conocidos como Recomendaciones, que sirven como referencia para construir una Web accesible, interoperable y eficiente, en la que se puedan desarrollar aplicaciones cada vez más robustas.

En la creación de las Recomendaciones del W3C participan sus Miembros (más de 400 organizaciones, distribuidas a lo largo de todo el mundo y de diversos ámbitos: grandes empresas de hardware o software, centros investigadores, universidades, administraciones públicas, entre otras), el Equipo del W3C, expertos invitados, y cualquier usuario de la Web que quiera mostrar su opinión. Todos ellos trabajan conjuntamente a través de un proceso basado en el consenso, la neutralidad y la transparencia de la información.

El resultado: más de 110 tecnologías desde 1996.

Algunos de los estándares Web más conocidos y ampliamente utilizados son: HTML, para definir la estructura de los documentos; XML, que sirve de base para un gran número de tecnologías; y CSS, que permite asignar estilos para la representación de los documentos.

La finalidad de los estándares es la creación de una Web universal, accesible, fácil de usar y en la que todo el mundo pueda confiar. Con estas tecnologías abiertas y de uso libre se pretende evitar la fragmentación de la Web y mejorar las infraestructuras para que se pueda evolucionar hacia una Web con la información mejor organizada. (12)

### **1.2.2 Web 2.0 vs Web 1.0**

El concepto Web 2.0 debe su origen a una tormenta de ideas entre los equipos de O'Reilly Media y MediaLive International a mediados de 2004, fortalecido por la primera Web 2.0 Conference en octubre de ese mismo año. Con el objeto de dar cierta entidad teórica al nuevo término y contrarrestar la confusión del momento, O'Reilly publicó en septiembre de 2005 lo que hasta hoy es la principal referencia bibliográfica del concepto. Se trata del artículo "What Is Web 2.0 Design Patterns and Business Models for the Next Generation of Software". (13)

Como muchos conceptos importantes, Web 2.0 no tiene una clara frontera, sino más bien, un núcleo gravitacional. (14)

El concepto original del contexto, llamado *Web 1.0* eran páginas estáticas HTML que no eran actualizadas frecuentemente. El éxito de las punto-com dependía de webs más dinámicas (a veces llamadas *Web 1.5*) donde los CMS (Sistemas de Gestión de Contenidos, Content Management System en inglés, abreviado CMS) servían páginas HTML dinámicas creadas al vuelo desde una actualizada base de datos. En ambos sentidos, el conseguir *hits* (visitas) y la estética visual eran considerados como unos factores muy importantes.

Los propulsores de la aproximación a la Web 2.0 creen que el uso de la web está orientado a la interacción y redes sociales, que pueden servir contenido que explota los efectos de las redes, creando o no webs interactivas y visuales. Es decir, los sitios Web 2.0 actúan más como puntos de encuentro, o webs dependientes de usuarios, que como webs tradicionales.

Según O'Reilly, siete son los principios constitutivos de las aplicaciones Web 2.0: la Web como plataforma; el aprovechamiento de la inteligencia colectiva; la gestión de la base de datos como competencia básica; el fin del ciclo de las actualizaciones de versiones del software; los modelos de programación ligera junto a la búsqueda de la simplicidad; el software no limitado a un solo dispositivo; y las experiencias enriquecedoras de los usuarios.

En general, cuando se menciona el término Web 2.0 se refiere a una serie de aplicaciones y páginas de Internet que utilizan la inteligencia colectiva para proporcionar servicios interactivos en red dando al usuario el control de sus datos.

Así, se puede entender como 2.0 -"todas aquellas utilidades y servicios de Internet que se sustentan en una base de datos, la cual puede ser modificada por los usuarios del servicio, ya sea en su contenido (añadiendo, cambiando o borrando información o asociando datos a la información existente), bien en la forma de presentarlos, o en contenido y forma simultáneamente."- (15)

### **Web 1.0**

Tipo de Web: estática.

Período: 1994 - 1997

Tecnología asociada: HTML, GIF.

Características: las páginas Web eran documentos estáticos que jamás se actualizaban.

## Web 2.0

Tipo de Web: Colaborativa.

Período: 2003 - hoy

Tecnología asociada: AJAX, DHTML, XML, SOAP

Características: los usuarios se convierten en contribuidores. Publican las informaciones y realizan cambios en los datos.

## 1.3 Ajedrez

Uno de los juegos de mayor rango en la formación de valores éticos y desarrollador del razonamiento lógico es, a no dudarlo, el Ajedrez. Sus orígenes se remontan muchos siglos atrás, y existen conjeturas diversas sobre la ubicación de sus comienzos, la más aceptada de las cuales lo establece en la India. En esencia, se trata de la porfía entre dos contendientes, cada uno de los cuales tiene un conjunto de piezas a su disposición (blancas uno, negras el otro) con movimientos preestablecidos, que deben mover simulando una batalla sobre un tablero de 64 casillas (8 x 8), con escaques de colores oscuros y claros, en turnos alternados, cumpliendo reglas registradas en un Reglamento, con el propósito de determinar un ganador.

Existe un reconocimiento universal acerca de que el aprendizaje del Ajedrez puede ser útil como forma de desarrollar el razonamiento. El Ajedrez es jugado tanto recreativa como competitivamente: en sesiones entre amigos, entre miembros de clubes, en torneos, en Internet, e incluso por correo (ajedrez por correspondencia).

Cuba se hizo notar a nivel internacional en la práctica de este deporte, con la figura de José Raúl Capablanca y Graupera, quien fuera el tercer campeón mundial, entre 1921 y 1927. Luego, con el triunfo de la Revolución, se creó un movimiento verdaderamente masivo de la práctica del Ajedrez en el país, liderado por el Comandante Ernesto Guevara, quien a pesar de sus múltiples obligaciones, encontró tiempo para mantenerse vinculado al juego ciencia, y vaticinó que algún día Cuba tendría muchos Grandes Maestros, lo que se ha cumplido con creces, pues ya suman más de veinte los titulados cubanos con el máximo pergamino de la Federación Internacional de Ajedrez (más conocida por FIDE, del acrónimo de su nombre en francés: *Fédération Internationale des Échecs*). Recientemente, el GM (Gran Maestro) Leinier Domínguez ha logrado superar la mítica cifra de los 2700 puntos de Rating Elo (el

sistema de puntuación Elo es un método para calcular la fuerza relativa de los jugadores de juegos como el ajedrez), lo que constituye uno de los resultados más relevantes del Ajedrez en Cuba en todos los tiempos.

### 1.3.1 Notación portable de juego

Notación portátil de juego (Del original en inglés: Portable Game Notation (.PGN)) es un formato de computadora para grabar partidas de ajedrez, tanto los movimientos como la información relacionada; la mayoría de los programas de ajedrez para computadora reconocen este formato que es muy popular como consecuencia de su fácil uso.

En lo sucesivo se hará referencia a este formato por sus siglas en inglés PGN.

El formato PGN está estructurado para una fácil lectura y escritura por usuarios humanos y para fácil análisis y generación por programas informáticos. Las jugadas están dadas en notación algebraica de ajedrez. Por lo general la extensión asignada a los archivos con este formato es ".pgn".

Existen dos subformatos dentro de la especificación del PGN, el formato de importación y el de exportación. El formato de importación describe información que ha sido preparada a mano y es intencionalmente flexible; un programa que pueda leer datos de un formato PGN debe ser capaz de manejar este formato flexible. El formato de exportación es en cambio estricto, describe la información generada bajo el control de un programa informático, algo así como una bonita impresión de un programa fuente compilado por una computadora. El formato de exportación generado por distintos programas debe ser exactamente equivalente byte por byte.

El código informático del formato PGN empieza con un conjunto de #pares de etiquetas (el nombre de la etiqueta y su valor), seguido de las jugadas (los movimientos del ajedrez con comentarios opcionales).

(16)

#### **Pares de etiquetas**

Cada par de etiquetas comienza con un "[", seguido del nombre de la etiqueta, el valor de la etiqueta encerrado en comillas dobles ("), y un "]" para cerrar.

Para almacenar la información en el formato PGN es necesario dar siete etiquetas, llamadas "STR" (del inglés Seven Tag Roster) que significa "lista de siete etiquetas". En el formato de exportación, las etiquetas STR deben aparecer antes que cualquier otro par de etiquetas. El orden es el siguiente:

(Los nombres de las etiquetas son en idioma inglés, se agrega entre paréntesis la traducción pero tome en cuenta que esto no es parte del formato).

1. Event (Evento): el nombre del torneo o de la competencia.
2. Site (lugar): el lugar donde el evento se llevó a cabo. Esto debe ser en formato "Ciudad, Región PAÍS", donde PAÍS es el código del mismo en tres letras de acuerdo al código del Comité Olímpico Internacional. Como ejemplo: "México, D.F. MEX".
3. Date (fecha): la fecha de inicio de la partida en formato AAAA.MM.DD. Cuando se desconocen los valores se utilizan "??".
4. Round (ronda): La ronda original de la partida.
5. White (blancas): El jugador de las piezas blancas, en formato "apellido, nombre".
6. Black (negras): El jugador de las negras en el mismo formato.
7. Result (resultado): El resultado del juego. Sólo puede tener cuatro posibles valores: "1-0" (las blancas ganaron), "0-1" (Las negras ganaron), "1/2-1/2" (Tablas), o "\*" (para otro, ejemplo: el juego está actualmente en disputa).

### Ejemplo

A continuación un ejemplo de un juego en formato PGN, esta partida es muy famosa y se llama "La inmortal":

[Event "Informal Game"]

[Site "London, England ENG"]

[Date "1851.07.??"]

[Round "-"]

[White "Anderssen, Adolf"]

[Black "Kieseritzky, Lionel"]

[Result "1-0"]

1. e4 e5 2.f4 exf4 3.Bc4 Qh4+ 4.Kf1 b5 5.Bxb5 Nf6 6.Nf3 Qh6 7.d3 Nh5 8.Nh4 Qg5 9.Nf5 c6 10.g4 Nf6 11.Rg1 cxb5 12.h4 Qg6 13.h5 Qg5 14.Qf3 Ng8 15.Bxf4 Qf6 16.Nc3 Bc5 17.Nd5 Qxb2 18.Bd6 Bxg1 19.e5 Qxa1+ 20.Ke2 Na6 21.Nxg7+ Kd8 22.Qf6+ Nxf6 23.Be7# 1-0

## 1.4 Portales de ajedrez a nivel internacional

Se seleccionaron para el análisis los principales portales de ajedrez que existen en la actualidad; para ello se tuvo en consideración el criterio de expertos, entre ellos el AN (Árbitro Nacional de Ajedrez) Isbel Herrera del Sol (Presidente de la Cátedra Honorífica de Ajedrez de la UCI).

### 1.4.1 Internet Chess Club

Sitio más popular de Juego Online: <http://www.chessclub.com/>

Fundado en 1995, el Internet Chess Club (ICC) fue pionero en ofrecer juegos online en la red. Sus raíces se remontan a un pequeño grupo de jugadores de ajedrez de la década de los 80. En 1995, después de estar algunos meses en internet, el número de usuarios había crecido de unos cientos a más de diez mil inscritos. Tras más de 10 años de actividad, el Internet Chess Club cuenta con más de 30.000 socios, lo que lo convierte en el mayor sitio de Internet con Servicios Premium de Ajedrez Online. Se realizan torneos periódicos restringidos para jugadores que paguen sus inscripciones donde participan jugadores con categoría de GM (Gran Maestro). Entre sus principales funcionalidades se tienen:

- ✓ Juego online en tiempo real.
- ✓ Juego en la modalidad de torneos.
- ✓ Juego contra motores de ajedrez.
- ✓ Multilenguaje.
- ✓ Observar partidas online de grandes maestros.
- ✓ Medición de fuerza por sistema de Elo.

El servidor está hospedado en los Estados Unidos, y se permite jugar una semana gratis, después de vencido el plazo es necesaria la compra del registro; solo los jugadores con título FIDE mayor que MF (Maestro Fide) pueden jugar de forma gratuita. Existen varios clientes para conectarse a él y jugar, los más famosos como el Fritz, no son libres.

El portal fue clasificado como Web 2.0.

### 1.4.2 ICCF (International Correspondence Chess Federation)

Sitio de la Federación Internacional de Ajedrez por Correspondencia: <http://www.iccf.com/>

Sitio oficial de la FIDE dedicado al ajedrez por correspondencia. Posee la funcionalidad del juego mediante email, además de gestionar el envío de tarjetas postales y cartas a cualquier lugar del mundo promoviendo el juego ciencia por correspondencia. Su sistema de rating Elo está aprobado por la FIDE como Elo oficial de ajedrez por correspondencia, teniendo Grandes Maestros y hasta campeones mundiales de ajedrez por correspondencia. Posee más de 100 000 miembros de 65 nacionalidades diferentes y se han realizado en el mismo 18 campeonatos mundiales de ajedrez por correspondencia. Entre sus principales funcionalidades están:

- ✓ Juego por correspondencia de partidas sueltas.
- ✓ Juego por correspondencia en la modalidad de torneos.
- ✓ Sistema de rating Elo oficial de la FIDE.

El servidor web está hospedado en los Estados Unidos, y como requisito de registro está pertenecer al club de forma oficial. Basado en PHP y como cliente se utilizan navegadores web.

El portal fue clasificado como Web 2.0.

#### **1.4.3 ChessGames.Com**

Sitio de base de partidas con filtro de selección: <http://www.chessgames.com/>

El sitio fue clasificado como Web 2.0. Cuenta con un filtro de selección para extraer las partidas de la base de datos, donde se muestran los siguientes campos: año, jugadores, color de piezas, número de movidas, apertura, resultado. Cuenta con un visor de partidas, el cual además se utiliza para publicar problemas online.

#### **1.4.4 World Chess Federation**

Sitio de la Federación Internacional de Ajedrez: [www.fide.com](http://www.fide.com)

El sitio fue clasificado como Web 2.0. En el mismo además de la publicación de noticias de actualidad ajedrecística, se brindan un grupo de servicios: consultar la base de datos de jugadores, consultar la base de datos de torneos, realizar cálculos de Elo (sistema de puntuación en el ajedrez, se denomina Elo en honor a su inventor: Árpád Élő), consultar el ranking de jugadores a nivel internacional.



### 1.4.5 Corus Chess

Sitio de uno de los torneos de ajedrez más prestigioso que se celebra anualmente:

<http://www.coruschess.com/>

El portal fue clasificado como Web 2.0. El mismo cuenta con las siguientes secciones: Historia, donde se guarda la historia de todas las ediciones del torneo relativa a los participantes y las partidas, incluye galería de imágenes; Estadísticas, donde se muestran estadísticas de los jugadores y las partidas a lo largo de todas las ediciones del torneo; Archivo, donde aparecen las últimas ediciones del torneo. Cuenta con un visor, y un filtro para la selección de las partidas. En la confección del sitio se empleó el lenguaje de programación PHP.

### 1.4.6 ChessBase

Principal página de noticias y de venta de productos online: <http://chessbase.com/>

La misma le da cobertura a los principales torneos de ajedrez que se celebran anualmente, cuenta con un gran número de colaboradores que la surten de reportajes, entrevistas e imágenes detalladas del acontecer ajedrecístico. Cuenta además con una tienda de productos online donde se ponen a la venta libros y programas de ajedrez del más alto nivel. Les da la posibilidad a los usuarios de visualizar las partidas a través de la web y promueve un producto para la gestión de bases de partidas que lleva su nombre.

## 1.5 Portales de ajedrez a nivel nacional

Al inicio de la investigación en la red cubana existían tres sitios de ajedrez:

1. Sitio del Ajedrez en Cuba: <http://www.cuba.cu/ajedrez/>
2. Sitio del Torneo Capablanca in Memoriam: <http://www.inder.cu/capablanca/>
3. Sitio de la Federación Cubana de Ajedrez por Correspondencia: <http://www.inder.cu/fecap/>

En un análisis estructural de los mismos fueron clasificados como Web 1.0.

### 1.5.1 Portal “Infodrez”

El portal “Infodrez” (<http://infodrez.uci.cu/>), Web de la Cátedra Honorífica de Ajedrez “Remberto A. Fernández González” de la UCI, consta de dos partes principales:

1. El CMS “e107” con los módulos que lo caracterizan: Gestión de usuarios, Gestión de noticias, Foro, entre otros.
2. El Módulo de Juego Online (principal atracción de la comunidad ajedrecística). Donde los usuarios pueden jugar partidas de ajedrez en tiempo real.

En un análisis realizado a este portal, fue clasificado como Web 2.0.

## **1.6 Resultado del análisis a los principales portales de ajedrez a nivel internacional**

Como resultado del análisis a los principales portales de ajedrez a nivel internacional se determinaron las siguientes tendencias:

- ✓ Para su confección se utilizan generalmente Sistemas de Gestión de Contenidos.
- ✓ Los mismos se orientan a una determinada rama del ajedrez (arbitraje, juego en línea, venta de productos, entre otras).
- ✓ Se clasifican como Web 2.0 en mayor medida.

### **1.6.1 Propuesta de módulos con que podría contar un portal orientado al ajedrez**

Como resultado del análisis de los componentes y temáticas de los principales portales de ajedrez que existen en la actualidad surge la propuesta de una serie de módulos con que podría contar un portal orientado al ajedrez.

#### **Módulo de Juego Online**

Módulo encargado del juego entre usuarios con respuesta en tiempo real y variantes de juegos como son torneos, simultáneas, match, juegos rápidos. Actualmente implementado y puesto en práctica en la UCI.

#### **Módulo de Arbitraje**

El objetivo de este módulo es brindarles información a los árbitros, así como las herramientas necesarias para su trabajo.

El módulo debe permitir mostrar los siguientes tipos de información:

- ✓ Reglamento vigente.
- ✓ Relación de árbitros (breve biografía incluyendo una foto).

- ✓ Elo actualizado (buscador de Elo).
- ✓ Torneos reportados.
- ✓ Cronograma oficial de eventos.
- ✓ Propaganda de eventos próximos a celebrarse.
- ✓ Artículos técnicos.
- ✓ Reclamaciones de Elo y título.
- ✓ Historia del arbitraje.
- ✓ Normas de títulos.
- ✓ Casos arbitrales.
- ✓ Títulos concedidos.

Además, debe contar con el apoyo de herramientas que permitan realizar las siguientes funcionalidades:

- ✓ Sorteo computarizado, emparejamientos y administración de torneos para los diferentes sistemas de juego.
- ✓ Cálculo del Elo y expectancia.
- ✓ Cálculo del Performance Rating

### **Módulo Visor de Torneos**

Módulo encargado de la gestión de Torneos. Debe presentar las siguientes funcionalidades:

- ✓ Crear un torneo con todos los datos generales del mismo.
- ✓ Gestionar jugadores.
- ✓ Ver partidas mediante un visor.
- ✓ Descarga de las partidas
- ✓ Cuadros sinópticos
- ✓ Directorio de ajedrecistas, árbitros y organizadores.
- ✓ Breve biografía de los jugadores.

- ✓ Galería de imágenes
- ✓ Pequeños artículos periodísticos sobre cada ronda.
- ✓ Incluir el historial de las ediciones pasadas.

### **Módulo de Estadísticas**

De semejante funcionalidad al sitio [www.chessgames.com](http://www.chessgames.com) y teniendo en cuenta las posibilidades estadísticas que brinda el programa **ChessBase** el cual permite cargar bases de partidas de jugadores profesionales y analizarlas. Lo novedoso de la aplicación sería la inclusión de una base de partidas con jugadores cubanos.

Debe permitir realizar búsquedas por las categorías que se muestran a continuación:

- ✓ Torneo
- ✓ Jugador, contrincante
- ✓ Fecha
- ✓ Edición
- ✓ Color de piezas
- ✓ Título
- ✓ Elo
- ✓ ECO (Sistema de clasificación de aperturas en ajedrez)
- ✓ Resultados
- ✓ Tipo de Torneo (Blitz, Rápido, Clásico)
- ✓ Categoría
- ✓ Número de Rondas
- ✓ Cantidad de movimientos
- ✓ Comentadas
- ✓ Premiada (belleza)
- ✓ Resultados de una o varias categorías seleccionadas por el usuario

Las principales funcionalidades que debe presentar son:

- ✓ Visor de partidas para ver los juegos.
- ✓ Descarga de las partidas.
- ✓ Gráfica de rendimiento estadístico de las opciones que el usuario seleccione (generar repertorio).
- ✓ Explorador de aperturas.
- ✓ Estudio de Contrarios

### **Módulo de Enseñanza**

Debe posibilitar el montaje de las diferentes vías de enseñanza existentes mediante el servicio Web, dentro de ellas:

- ✓ Multimedia.
- ✓ Herramientas Web que permitan la enseñanza-aprendizaje del ajedrez de una manera interactiva.

### **Módulo de Ajedrez por Correspondencia**

Producto que permita el desarrollo de la modalidad ajedrez postal, mediante una aplicación Web. Este módulo debe ser capaz de brindar las siguientes funcionalidades:

- ✓ Visor Web
- ✓ Secciones de información donde se puedan ubicar temas como historia y evolución del Ajedrez Postal, resultados históricos nacionales e internacionales, reglamento de la federación postal, estadística de los resultados individuales alcanzados (expectancia, performance rating y variación del Elo) e información sobre las preferencias del jugador (apertura o defensa).
- ✓ Base de datos de las partidas jugadas.
- ✓ Creación de torneos (temáticos y campeonatos) y gestión de los mismos.

### **1.6.2 Propuesta de Componentes con que debería contar un portal orientado al ajedrez**

Del análisis de los principales portales de ajedrez a nivel mundial y de sus puntos en común se desprenden los siguientes componentes necesarios para un portal orientado al Ajedrez:

**Visor de Partidas**

Componente mediante el cual se puedan visualizar partidas de ajedrez en la Web.

**Gestor de Cuadros Sinópticos**

Componente que permita crear, insertar, editar y eliminar cuadros sinópticos en un sitio web, con información y resultados de los torneos de ajedrez.

**Gestor de Diagramas**

Componente que permita la gestión de diagramas de ajedrez en un sitio web.

**1.7 Contenido Web**

Además de la funcionalidad y el diseño gráfico, un sitio Web no debe olvidar que la base de su éxito está en los contenidos que ofrezca a su audiencia. (17)

Los contenidos para la Web pueden ser gráficos, multimediales y textuales. Se consideran que estos son buenos contenidos para la web en cuanto:

- ✓ Están concebidos para Internet.
- ✓ Se ajustan a los objetivos del sitio.
- ✓ Dan respuesta a las necesidades y tareas del usuario.
- ✓ Tienen una buena producción.
- ✓ Respetan los estándares y la accesibilidad Web.

La claridad, pertinencia, eficacia y calidad de contenidos, inciden de una manera directa en:

- ✓ La asociación que el usuario hace de su experiencia particular en el sitio con la empresa. El principio es simple: así como es el sitio, es la empresa, sus servicios y productos. Es una variación de la máxima del saber popular que dice que las cosas se parecen a su dueño.
- ✓ La generación de confianza y credibilidad en los servicios o productos. Crear confianza es la conjugación de entregar un mensaje claro y conciso que coincida con las necesidades de los usuarios y la verdadera oferta del emisor del mensaje.
- ✓ El posicionamiento de su sitio Web en los principales motores de búsqueda. Finalmente, la gente busca contenido. (17)

## 1.8 Gestión de Contenidos

Inicialmente, con la escritura y la capacidad tecnológica de emplearla sobre tablas de arcilla, papiro y papel sucesivamente; los grupos humanos lograron colocar en el espacio y transferir en el tiempo, la información, los contenidos y el conocimiento. Con las capacidades tecnológicas actuales y como resultado de su estructura interior, en esa nebulosa a la que puede llamarse "La Red", la información, los contenidos y los conocimientos han alcanzado una tercera dimensión: la ubicuidad: la información, los contenidos y el conocimiento pueden estar accesibles desde todas partes al mismo tiempo. (18)

La gestión de los contenidos debe considerar que:

- ✓ Gestionar datos, información, contenidos y conocimiento, en su sentido práctico, es la misma cosa. Se trata, en definitiva, de manejar sus registros físicos; antes bajo el predominio de la tecnología del papel, ahora en las condiciones de la tecnología digital.
- ✓ La tecnología del papel obliga a gestionar contenidos, registrados en soportes físicos audiovisuales prácticamente a escala referencial con muy baja integración para el manejo de los contenidos primarios. La era digital aporta la posibilidad, como nunca antes, de gestionar el audio, el vídeo, la fotografía y los textos de manera totalmente integrada.
- ✓ La gestión de contenidos no puede obviar las necesidades de protección y seguridad que dichos contenidos requieren en correspondencia con la sensibilidad que implican para sus generadores o propietarios en función de sus intereses vitales.
- ✓ Los gestores de contenidos digitales, no importa su profesión de origen, requieren lograr un dominio "suficiente" sobre las tecnologías digitales para la creación, diseño, organización, procesamiento y diseminación de dichos contenidos. Como en la era del papel, un editor, un periodista, un bibliotecario, un documentalista, un analista e infinidad de ocupaciones más, conocían lo suficiente, tanto sobre las tecnologías que se asociaban al empleo del papel y de otros soportes físicos, como para emitir un juicio profesional.

## 1.9 Sistemas de Gestión de Contenidos

Realizar una web puede ser un trabajo complicado y muy laborioso si no se dispone de las herramientas adecuadas. En el pasado las herramientas eran básicamente editores que permitían generar una página, que evolucionaron para incorporar el control de la estructura de la web y otras funcionalidades, pero en general estaban enfocadas más a la creación que al mantenimiento. En los últimos años se ha

desarrollado el concepto de Sistema de Gestión de Contenidos (*Content Management Systems* o CMS). Se trata de herramientas que permiten crear y mantener una web con facilidad, encargándose de los trabajos más tediosos que hasta ahora ocupaban el tiempo de los administradores de las webs.

Teniendo en cuenta el ahorro que supone la utilización de estas herramientas, y el coste de desarrollarlas, sería lógico esperar que su precio fuera muy elevado. Eso es cierto para algunos productos comerciales, pero existen potentes herramientas de gestión de contenidos de acceso libre, disponibles con licencias de código abierto.

Los gestores de contenidos proporcionan un entorno que posibilita la actualización, mantenimiento y ampliación de la web con la colaboración de múltiples usuarios. En cualquier entorno virtual ésta es una característica importante, que además puede ayudar a crear una comunidad cohesionada que participe más de forma conjunta. (19)

### **1.9.1 ¿Qué es un CMS?**

Los Sistemas de Gestión de Contenidos (*Content Management Systems* o CMS) es un software que se utiliza principalmente para facilitar la gestión de webs, ya sea en Internet o en una intranet, y por eso también son conocidos como gestores de contenido web (*Web Content Management* o WCM). (19)

Consiste en una interfaz que controla una o varias bases de datos donde se aloja el contenido del sitio. El sistema permite manejar de manera independiente el contenido y el diseño. Así, es posible manejar el contenido y darle en cualquier momento un diseño distinto al sitio sin tener que darle formato al contenido de nuevo, además de permitir la fácil y controlada publicación en el sitio a varios editores.

James Robertson (2003) propone una división de la funcionalidad de los sistemas de gestión de contenidos en cuatro categorías: creación de contenido, gestión de contenido, publicación y presentación. (20)

#### **Creación de contenido**

Un CMS aporta herramientas para que los creadores sin conocimientos técnicos en páginas web puedan concentrarse en el contenido. Lo más habitual es proporcionar un editor de texto WYSIWYG, en el que el usuario ve el resultado final mientras escribe, al estilo de los editores comerciales, pero con un rango de formatos de texto limitado. Esta limitación tiene sentido, ya que el objetivo es que el creador pueda poner énfasis en algunos puntos, pero sin modificar mucho el estilo general del sitio web.



Hay otras herramientas como la edición de los documentos en XML, utilización de aplicaciones ofimáticas con las que se integra el CMS, importación de documentos existentes y editores que permiten añadir marcas, habitualmente HTML, para indicar el formato y estructura de un documento.

Un CMS puede incorporar una o varias de estas herramientas, pero siempre tendría que proporcionar un editor WYSIWYG por su facilidad de uso y la comodidad de acceso desde cualquier ordenador con un navegador y acceso a Internet.

Para la creación del sitio propiamente dicho, los CMS aportan herramientas para definir la estructura, el formato de las páginas, el aspecto visual, uso de patrones, y un sistema modular que permite incluir funciones no previstas originalmente.

### **Gestión de contenido**

Los documentos creados se depositan en una base de datos central donde también se guardan el resto de datos de la web, cómo son los datos relativos a los documentos (versiones hechas, autor, fecha de publicación y caducidad), datos y preferencias de los usuarios, la estructura de la web, entre otros.

La estructura de la web se puede configurar con una herramienta que, habitualmente, presenta una visión jerárquica del sitio y permite modificaciones. Mediante esta estructura se puede asignar un grupo a cada área, con responsables, editores, autores y usuarios con diferentes permisos. Eso es imprescindible para facilitar el ciclo de trabajo (*workflow*) con un circuito de edición que va desde el autor hasta el responsable final de la publicación. El CMS permite la comunicación entre los miembros del grupo y hace un seguimiento del estado de cada paso del ciclo de trabajo.

### **Publicación**

Una página aprobada se publica automáticamente cuando llega la fecha de publicación, y cuando caduca se archiva para futuras referencias. En su publicación se aplica el patrón definido para toda la web o para la sección concreta donde está situada, de forma que el resultado final es un sitio web con un aspecto consistente en todas sus páginas. Esta separación entre contenido y forma permite que se pueda modificar el aspecto visual de un sitio web sin afectar a los documentos ya creados y libera a los autores de preocuparse por el diseño final de sus páginas.

## **Presentación**

Un CMS puede gestionar automáticamente la accesibilidad del web, con soporte de normas internacionales de accesibilidad como WAI (Web Accessibility Initiative), y adaptarse a las preferencias o necesidades de cada usuario. También puede proporcionar compatibilidad con los diferentes navegadores disponibles en todas las plataformas (Windows, Linux, Mac, Palm...) y su capacidad de internacionalización le permite adaptarse al idioma, sistema de medidas y cultura del visitante.

El sistema se encarga de gestionar muchos otros aspectos como son los menús de navegación o la jerarquía de la página actual dentro del web, añadiendo enlaces de forma automática. También gestiona todos los módulos, internos o externos, que incorpore al sistema. Así por ejemplo, con un módulo de noticias se presentarían las novedades aparecidas en otro web, con un módulo de publicidad se mostraría un anuncio o mensaje animado, y con un módulo de foro se podría mostrar, en la página principal, el título de los últimos mensajes recibidos. Todo eso con los enlaces correspondientes y, evidentemente, siguiendo el patrón que los diseñadores hayan creado.

## **1.10 Software Libre**

Un programa es *software libre* si: (21)

- ✓ Usted tiene libertad para *ejecutar* el programa con cualquier propósito.
- ✓ Usted tiene la libertad para *modificar* el programa y adaptarlo a sus necesidades. (Para que esta libertad sea efectiva en la práctica, usted debe tener acceso al código fuente, porque modificar un programa sin disponer del código fuente es extraordinariamente dificultoso).
- ✓ Usted tiene la libertad para *redistribuir copias*, tanto gratis como por un canon.
- ✓ Usted tiene la libertad para *distribuir versiones modificadas* del programa, de tal manera que la comunidad pueda beneficiarse con sus mejoras.

**Estándares abiertos:** Son especificaciones técnicas, publicadas y controladas por alguna organización que se encarga de su desarrollo, las cuales han sido aceptadas por la industria, estando a disposición de cualquier usuario para ser implementadas en un software libre u otro, promoviendo la competitividad, interoperabilidad o flexibilidad. (21)

### 1.10.1 El Software Libre en Cuba

Sin duda alguna, el uso del Software Libre es sustentable en Cuba a partir de las ventajas que tiene con respecto a los del tipo propietario. Por esto, su aplicación como plataforma informática de trabajo adquiere una relevante significación que puede verse desde 3 ámbitos diferentes:

#### **Significado Político (21)**

- ✓ Desde un primer punto de vista, representa la no utilización de productos informáticos que demanden la autorización de sus propietarios (licencias) para su explotación. Es válido recordar que, en el presente Cuba se encuentra a merced de la empresa norteamericana Microsoft, que tiene la capacidad legal de reclamar a Cuba que no siga utilizando un sistema operativo de su propiedad, basado en leyes de propiedad industrial por las cuales también Cuba se rige; esto provocaría una interrupción inmediata del programa de informatización de la sociedad que como parte de la batalla de ideas está desarrollando el país, además pudiera implementarse una campaña de descrédito a la isla, abogando el uso de la piratería informática por parte de las instituciones estatales cubanas.
- ✓ Desde un segundo punto de vista, el Software Libre representa la alternativa para los países pobres, y es por concepción, propiedad social, si se tiene en cuenta que una vez que comienza a circular rápidamente se encuentra disponible para todos los interesados sin costo alguno o en su defecto a muy bajo costo.
- ✓ En tercer lugar, es desarrollado de forma colectiva y cooperativa, tanto en su creación como en su desarrollo, cuantitativa y cualitativamente, mostrando su carácter público y sus objetivos de beneficiar a toda la comunidad.

La posibilidad de usar, copiar, estudiar, modificar y redistribuir libremente el software como un bien social, que brinda esta plataforma, cumple los preceptos enunciados por la sociedad socialista cubana y está acorde con el tipo de economía socialista, donde el valor social está por encima de la ganancia.

#### **Significado Económico (21)**

- ✓ Su utilización no implica grandes gastos adicionales por concepto de cambio de plataforma de software, por cuanto es operable en el mismo soporte de hardware con que cuenta el país.
- ✓ La adquisición de cualquiera de sus distribuciones puede hacerse de forma gratuita, descargándolas

directamente de Internet o en algunos casos a muy bajos precios, se garantiza su explotación con un mínimo de recursos, por cuanto no hay que pagar absolutamente nada por su utilización (no requiere de licencia de uso, las cuales son generalmente muy caras), distribución y/o modificación.

- ✓ El uso del Software Libre desarrollado con Estándares Abiertos, fortalecerá la industria del software nacional, aumentando y fortaleciendo sus capacidades. Facilitará la reducción de la brecha social y tecnológica en el menor tiempo y costo posibles. Su uso en la Institución Pública y en los servicios públicos, facilitará la interoperabilidad de los sistemas de información del Estado, contribuyendo a dar respuestas rápidas y oportunas a los ciudadanos, mejorando la gobernabilidad.

### **Significado Tecnológico (21)**

- ✓ Permite su adaptación a los contextos de aplicación, al contar con su código fuente, lo cual garantiza un mayor porcentaje de efectividad, además de la corrección de sus errores de programación y la obtención de las actualizaciones y las nuevas versiones.
- ✓ Todas las mejoras que se realicen no tienen restricciones. De este modo, cualquier otra administración, empresa, institución u organismo se puede beneficiar de las mejoras introducidas.
- ✓ Se fomenta la innovación tecnológica del país. Al disponer del código fuente de la aplicación, se puede realizar el desarrollo de mejoras, en vez de encargarlas a empresas de otros países que trabajan con sistemas de licencia propietaria. De este modo, se contribuye a la formación de profesionales en nuevas tecnologías y al desarrollo local bajo planes estratégicos propios.
- ✓ Proceso de corrección de errores muy dinámico. Los usuarios del programa de todo el mundo, gracias a que disponen del código fuente del programa, pueden detectar los posibles errores, corregirlos, y contribuir con sus mejoras.
- ✓ Más dificultad para introducir código malicioso, espía o de control remoto. Debido a que el código es revisado por muchos usuarios que pueden detectar posibles puertas traseras.

El proyecto "Infodrez", tuvo sus inicios en la Facultad 10 de la UCI, donde siempre prevaleció entre sus miembros la filosofía del *software libre*, por este hecho y las motivaciones antes expresadas la presente investigación será enfocada hacia ese horizonte informático.

## 1.11 Calidad de Software

La Calidad de Software es la *“concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se espera de todo software desarrollado profesionalmente.”* (22)

### 1.11.1 Calidad y Arquitectura de Software

Una arquitectura software representa el diseño de alto nivel de la estructura del software y los principios y guías que garantizan su evolución en el tiempo. En la arquitectura los elementos se integran, siguiendo un estilo determinado, para satisfacer tanto los requisitos funcionales como los no funcionales del sistema, con tres beneficios importantes: se fortalece la reutilización de funcionalidad y estructuras de alto nivel, es posible realizar predicciones de calidad acerca del sistema que va a ser construido, es posible razonar acerca del impacto del cambio sobre los sistemas en operación. (23)

Uno de los objetivos importantes del área de estudio arquitecturas, es encontrar la relación entre atributos de calidad y arquitecturas de software. Las propuestas alrededor de este tema pretenden facilitar el razonamiento acerca de los atributos de calidad de un sistema ya construido (evaluación) o la predicción de la calidad de un sistema en construcción. Los atributos de calidad pueden ser divididos en dos categorías: Atributos de calidad observables en ejecución como rendimiento, seguridad, disponibilidad..., y atributos de calidad no observables en ejecución, como facilidad de modificación, portabilidad, reusabilidad, entre otros. Ambas categorías de atributos estarán directamente afectadas por las decisiones de arquitectura que se tomen.

## 1.12 Arquitectura de Software

La arquitectura de un software puede entenderse como aquella estructura del programa que cohesiona las funcionalidades más críticas y relevantes (necesarias para el sistema), y que sirve de soporte al resto de funcionalidades finales (necesarias para el usuario). Su especificación es ampliamente aceptada como el problema central de diseño de un sistema de software complejo.

*"La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución."*

(8)

### 1.12.1 ¿Por qué es necesaria la Arquitectura de Software?

A continuación se sintetizan las virtudes de la AS articulando las opiniones convergentes de los expertos:  
(24)

- ✓ **Comunicación mutua.** La AS representa un alto nivel de abstracción común que la mayoría de los participantes, si no todos, pueden usar como base para crear entendimiento mutuo, formar consenso y comunicarse entre sí. En sus mejores expresiones, la descripción arquitectónica expone las restricciones de alto nivel sobre el diseño del sistema, así como la justificación de decisiones arquitectónicas fundamentales.
- ✓ **Decisiones tempranas de diseño.** La AS representa la encarnación de las decisiones de diseño más tempranas sobre un sistema, y esos vínculos tempranos tienen un peso fuera de toda proporción en su gravedad individual con respecto al desarrollo restante del sistema, su servicio en el despliegue y su vida de mantenimiento.
- ✓ **Reutilización, o abstracción transferible de un sistema.** La AS encarna un modelo relativamente pequeño, intelectualmente tratable, de la forma en que un sistema se estructura y sus componentes se entienden entre sí; este modelo es transferible a través de sistemas; en particular, se puede aplicar a otros sistemas que exhiben requerimientos parecidos y puede promover reutilización en gran escala. El diseño arquitectónico soporta reutilización de grandes componentes o incluso de frameworks (marco de trabajo) en el que se pueden integrar componentes.
- ✓ **Evolución.** La AS puede exponer las dimensiones a lo largo de las cuales puede esperarse que evolucione un sistema. Haciendo explícitas estas “paredes” perdurables, quienes mantienen un sistema pueden comprender mejor las ramificaciones de los cambios y estimar con mayor precisión los costos de las modificaciones. Esas delimitaciones ayudan también a establecer mecanismos de conexión que permiten manejar requerimientos cambiantes de interoperabilidad, prototipado y reutilización.
- ✓ **Administración.** La experiencia demuestra que los proyectos exitosos consideran una arquitectura viable como un logro clave del proceso de desarrollo industrial. La evaluación crítica de una arquitectura conduce típicamente a una comprensión más clara de los requerimientos, las estrategias de implementación y los riesgos potenciales.

### 1.12.2 Estilos en la Arquitectura de Software

Mary Shaw y Paul Clements (25) describen los estilos arquitectónicos como un conjunto de reglas de diseño que identifican las clases de componentes y conectores que se pueden utilizar para componer el sistema o subsistema, junto con las restricciones locales o globales de la forma en que la composición se lleva a cabo. Los componentes, incluyendo los subsistemas encapsulados, se pueden distinguir por la naturaleza de su computación: por ejemplo, si retienen estado entre una invocación y otra, y de ser así, si ese estado es público para otros componentes. Los tipos de componentes también se pueden distinguir conforme a su forma de empaquetado, o dicho de otro modo, de acuerdo con las formas en que interactúan con otros componentes. El empaquetado es usualmente implícito, lo que tiende a ocultar importantes propiedades de los componentes. Para clarificar las abstracciones se aísla la definición de esas interacciones bajo la forma de conectores: por ejemplo, los procesos interactúan por medio de protocolos de transferencia de mensajes, o por flujo de datos a través de tuberías (*pipes*). Es en gran medida la interacción entre los componentes, mediados por conectores, lo que confiere a los distintos estilos sus características distintivas.

Cada estilo describe una categoría del sistema que contiene: un conjunto de *componentes* (por ejemplo, una base de datos, módulos computacionales) que realizan una función requerida por el sistema; un conjunto de *conectores* que posibilitan la comunicación, la coordinación y la cooperación entre los componentes; *restricciones* que definen como se pueden integrar los componentes que forman el sistema; y *modelos semánticos* que permiten al diseñador entender las propiedades globales de un sistema para analizar las propiedades conocidas de sus partes constituyentes (26).

#### **Taxonomía de estilos arquitectónicos**

Algunos de los principales estilos arquitectónicos que se usan en la actualidad están divididos por Clases de Estilos las que engloban una serie de estilos arquitectónicos específicos: (27)

#### **Estilos de Flujo de Datos**

Esta familia de estilos enfatiza la reutilización y la modificabilidad. Es apropiado para sistemas que implementan transformaciones de datos en pasos sucesivos. Ejemplares de la misma serían las arquitecturas de tubería-filtros y las de proceso secuencial en lote.

- ✓ Tubería y filtros

### **Estilos Centrados en Datos**

Esta familia de estilos enfatiza la integridad de los datos. Se estima apropiado para sistemas que se fundan en acceso y actualización de datos en estructuras de almacenamiento.

- ✓ Arquitecturas de Pizarra o repositorio

### **Estilos de llamada y retorno**

Esta familia de estilos enfatiza la modificabilidad y la escalabilidad. Son los estilos más generalizados en sistemas de gran escala.

- ✓ Arquitectura en capas
- ✓ Arquitecturas orientadas a objetos
- ✓ Arquitecturas basadas en componentes

### **Estilos de código móvil**

Esta familia de estilos enfatiza la portabilidad. Ejemplos de la misma son los intérpretes, los sistemas basados en reglas y los procesadores de lenguaje de comando.

- ✓ Arquitectura de máquinas virtuales

### **Estilos heterogéneos**

En este apartado podrían agregarse formas que aparecen esporádicamente en los censos de estilos, como los sistemas de control de procesos industriales, sistemas de transición de estados, arquitecturas específicas de dominios o estilos derivados de otros estilos.

- ✓ Sistemas de control de procesos
- ✓ Arquitecturas basadas en atributos

### **Estilos Peer-to-Peer**

Esta familia, también llamada de componentes independientes, enfatiza la modificabilidad por medio de la separación de las diversas partes que intervienen en la computación. Consiste por lo general en procesos independientes o entidades que se comunican a través de mensajes. Cada entidad puede enviar mensajes a otras entidades, pero no controlarlas directamente. Los mensajes pueden ser enviados a



componentes nominados o propalados mediante *broadcast*.

- ✓ Arquitecturas basadas en eventos
- ✓ Arquitecturas orientadas a servicios
- ✓ Arquitecturas basadas en recursos

### **1.12.2.1 Arquitecturas basadas en componentes**

La arquitectura basada en componentes consiste en una rama de la Ingeniería de Software (ISBC, Ingeniería de Software Basada en Componentes) en la cual se trata con énfasis la descomposición del software en componentes funcionales. Esta descomposición permite convertir componentes pre-existentes en piezas más grandes de software. (27)

Este proceso de construcción de una pieza de software con componentes ya existentes, da origen al principio de reutilización del software, mediante el cual se promueve que los componentes sean implementados de una forma que permita su utilización funcional sobre diferentes sistemas en el futuro.

Se debe entonces, para terminar de definir la arquitectura basada en componentes, saber que es un componente de software. Un componente de software se define típicamente como algo que puede ser utilizado como una caja negra, en donde se tiene de manera externa una especificación general, la cual es independiente de la especificación interna. (28)

De esta definición se presentan tres conceptos ligados con la definición de un componente:

- ✓ Interior del componente: es una pieza de software que cumple con un conjunto de propiedades y que se encuentra conformada como un artefacto del cual se espera que sea reutilizable.
- ✓ Exterior del componente: es una interface que cumple con un conjunto de propiedades y provee un servicio a los agentes humanos u otros artefactos de software.
- ✓ Relación interior-exterior: es la que define el proceso de relación entre el interior y el exterior del componente, a través de conceptos como especificación, implementación y encapsulación.

### **Elementos y estructura de la arquitectura basada en componentes**

De forma evidente se puede determinar que, el principal elemento de software dentro de una arquitectura basada en componentes son precisamente los componentes de software.

Existen 5 principios definidos por Clemens Szyperski (29), que definen a un componente de software como elemento de la arquitectura:

- ✓ Múltiple uso: se refiere al hecho de que un componente es escrito dentro de un contexto que permita que su funcionalidad sea útil en la creación de distintas piezas de software.
- ✓ Contexto no específico: en relación con la orientación conceptual de la especificación de un componente, debe estar planteada de una forma general que permita su adaptación en distintos sistemas, sin que el contexto tenga prioridad.
- ✓ Encapsulación: se refiere a la especificación interna oculta o no investigable a través de la interface. Así se protege que el resto de componentes y piezas de software dentro de un sistema, no se vean afectados por cambios en el diseño de uno de los componentes.
- ✓ Una unidad independiente de desarrollo con su propio control de versiones: este principio muy relacionado con la encapsulación, permite que un componente pueda ser desarrollado de manera independiente, cambiando el diseño o agregando nuevas funcionalidades, sin afectar significativamente el resto del sistema.

La estructura de la arquitectura basada en componentes contempla 3 partes:

1. El nombre de los componentes: el nombre de un componente debe ser la identificación de la funcionalidad y uso que tiene como software. Generalmente, los desarrolladores usan algún tipo de convención que facilite la identificación de componentes, especialmente, cuando se trabaja en proyectos de gran envergadura.
2. La interface de los componentes: es el área de intercambio (input-output) entre el interior y el exterior de un componente de software. La interface es quien permite acceder a los datos y funcionalidades que estén especificadas en el interior del componente (acceder funcionalmente, no a su especificación). Adicional a la interface se encuentra la documentación que muestra la información sobre cómo utilizar un componente.
3. Cuerpo y código de implementación: es la parte del componente que provee la forma (implementación) sobre la cual un fragmento del componente realiza sus servicios y funcionalidades. Este es el área que debe cumplir con el principio de encapsulación.

Dentro de la estructura de una arquitectura basada en componentes existen dos procesos fundamentales

para el desarrollo:

### **El ensamblaje de sistemas a partir de componentes de software**

Este proceso está compuesto por cuatro actividades:

1) Calificación de los componentes: comprende el estudio y análisis de qué tan adecuado es un componente para la construcción del sistema final. Esta evaluación se realiza sobre un conjunto de métricas que deben establecer los analistas y diseñadores de la arquitectura.

Además de esto, de acuerdo con Felix Bachman (30), durante esta actividad existen dos procesos relacionados con la certificación de los componentes:

I. El establecimiento de hechos sobre el componente para verificar que las propiedades que el componente posee están acordes con su especificación pública (documentada).

II. La validación de que los hechos establecidos sobre el componente son ciertos.

La razón por la cual se realiza este proceso de certificación es que existe un vínculo entre las propiedades certificadas de un componente y las del sistema final, es decir, la certificación es una herramienta para garantizar que las propiedades de la pieza de software final sean válidas.

2) Adaptación de los componentes: dado que un componente es desarrollado para cumplir con requerimientos específicos, es posible que esté orientado en cierta medida hacia el contexto en el que fue desarrollado. Por esta razón, se realiza un proceso de adaptación con el objetivo de minimizar la cantidad de conflictos.

Un componente puede ser adaptado entonces de tres maneras:

I. White-Box: cuando el componente debe ser reescrito para operar en conjunto con el resto de componentes del sistema.

II. Grey-Box: cuando el componente incorpora su propio API (Application Programming Interface).

III. Black-Box: cuando el componente no posee un API. Una interfaz completamente independiente es construida para acceder a los servicios del componente.

3) Ensamblaje de los componentes: es el proceso de integración de los componentes a través de la estructura mediante la cual fueron definidos. Esto incluye el modelo de software por componentes sobre el cual son escritos, por ejemplo: COM (Component Object Model), CORBA (Common Object Request

Broker Architecture), Enterprise JavaBeans y .NET.

4) Mantenimiento y evolución del sistema: consiste en el proceso de actualización de componentes, ya sea por requerimiento o por cambios de especificación. Estos cambios pueden ser la reescritura o sustitución de un componente. Por tal razón un componente trabaja como una unidad (conectable y desconectable) dentro de un sistema.

### **Reusabilidad**

Esta es una de las características más importantes en el desarrollo de sistemas bajo una arquitectura basada en componentes. Un componente de software debe ser diseñado de tal manera que pueda ser reutilizado en otros sistemas.

Este principio de reutilización del componente, requiere un esfuerzo extra por el equipo de desarrollo que se basa en:

- ✓ Una documentación completa de cada atributo y funcionalidad del componente.
- ✓ Una etapa de pruebas organizada y certera que certifique el correcto funcionamiento del componente.
- ✓ Una definición de comprobaciones precisa para el chequeo de cada parámetro de entrada (input) del componente.
- ✓ Un manejo de notificaciones de errores preciso, que advierta de la existencia de estos de una forma apropiada.
- ✓ Desarrollar teniendo en cuenta que el componente puede ser requerido para trabajar en muchos contextos muy diferentes unos de otros (tomar en cuenta la eficiencia, uso de memoria y recursos).

### **1.12.3 Patrones de diseño**

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. (31)

En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Se debe tener presente los siguientes elementos de un patrón: su

nombre, el problema (cuándo aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios).

Existen varios patrones de diseño popularmente conocidos, los cuales se clasifican como se muestra a continuación:

- ✓ Patrones Creacionales: Inicialización y configuración de objetos.
- ✓ Patrones Estructurales: Separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.
- ✓ Patrones de Comportamiento: Más que describir objetos o clases, describen la comunicación entre ellos.

#### 1.12.4 Patrones arquitectónicos

Los patrones de las arquitecturas se utilizan de una forma parecida a los patrones de diseño pero se centran en estructuras e interacciones de grano más grueso, entre subsistemas e incluso entre sistemas.

(6)

Los patrones arquitectónicos son plantillas para arquitecturas de software concretas, que especifican las propiedades estructurales de una aplicación y tienen un impacto en la arquitectura de subsistemas.

La selección de un patrón arquitectónico es, por tanto, una decisión fundamental de diseño en el desarrollo de un sistema de software. (32)

#### **Presentación-Abstracción-Control (PAC)**

El patrón de arquitectura PAC define una estructura jerárquica de agentes cooperativos. Cada agente es responsable de un aspecto específico de la funcionalidad de la aplicación y está compuesto por tres componentes: presentación-abstracción-control. Esta subdivisión separa los aspectos de HCI (Human-computer interaction, interacción persona-ordenador) de los agentes, del núcleo funcional de cada uno y de los mecanismos de comunicación con otros agentes. (33)

La **presentación** provee el aspecto visible del agente.

La **abstracción** provee el modelo de datos del agente y las operaciones para operar con estos datos.

El **control** conecta los componentes de presentación y de abstracción, y le agrega la funcionalidad necesaria para comunicarse con otros agentes.

La solución de este patrón organiza la estructura del sistema como una jerarquía de agentes según tres niveles: un agente de nivel superior, varios agentes de nivel intermedio y muchos agentes de nivel inferior. Cada agente es responsable de una funcionalidad específica de la aplicación y proporciona una interfaz de usuario especializada para ello. Véase la Figura 1.1. (33)

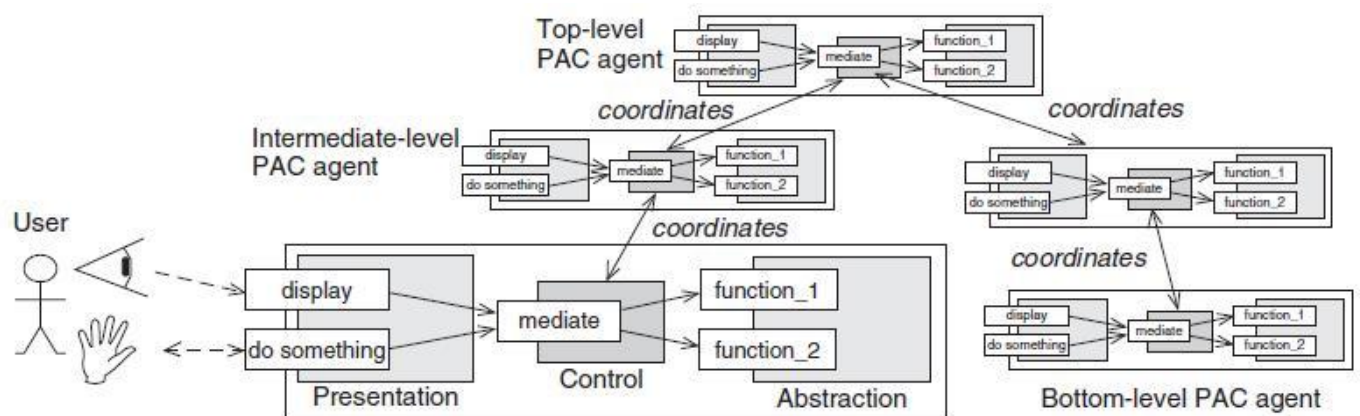


Figura 1.1 Estructura de una aplicación con PAC.

Los **agentes de nivel inferior** implementan la funcionalidad auto-contenida con la que los usuarios pueden interactuar, por ejemplo, administración, control de errores y manipulación de datos.

Los **agentes de nivel intermedio** coordinan múltiples agentes de nivel inferior relacionados entre sí, por ejemplo todas las vistas que visualizan un tipo particular de datos.

El **agente de nivel superior** provee el núcleo de la funcionalidad del sistema. Proporciona la funcionalidad básica que es compartida por todos los agentes (como el acceso a una base de datos).

Los agentes se conectan en la jerarquía a través de su componente de **control**.

Los usuarios interactúan con un agente a través de su presentación. Todas las peticiones de usuarios a las respectivas funciones en su abstracción son mediadas por el agente de control. Si un usuario necesita acceder a la acción o la coordinación de los demás agentes, median en esta petición los controles de estos agentes, ya sea hacia arriba o hacia abajo de la jerarquía, y de allí a sus abstracciones. (33)

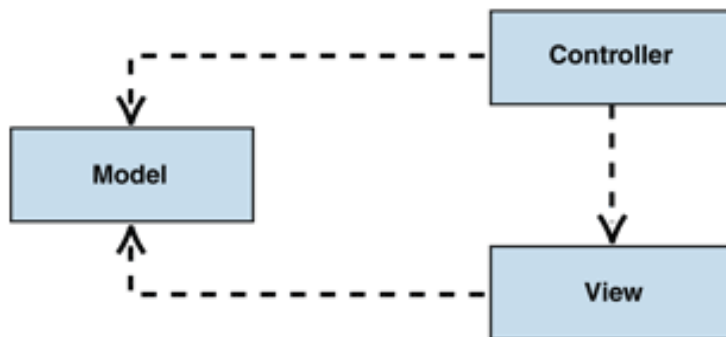
**Modelo Vista Controlador (MVC)**

Figura 1.2 Modelo pasivo.

Divide una aplicación interactiva en tres componentes. El **modelo** maneja el comportamiento y los datos del dominio de la aplicación, responde a las peticiones de información sobre su estado (que normalmente le hará la **vista**) y responde a las instrucciones para cambiar su estado (que normalmente le hará el **controlador**). La **vista** maneja la presentación de la información. El **controlador** interpreta las entradas del usuario e informa al **modelo** y/o a la **vista** para que cambien apropiadamente. Véase la Figura 1.2.

El controlador es el que efectúa los cambios de estado en el modelo y avisa a las vistas para que se actualicen. Las vistas accederán al modelo para consultar el nuevo estado que deben mostrar al usuario.

(27)

**MVC vs PAC**

MVC y PAC, abordan el problema de soporte a la variabilidad en cuanto a interfaces de usuario. Aunque ambos patrones están relacionados de varias maneras, no son necesariamente alternativas. En pocas palabras, MVC apoya la variabilidad dentro de una interfaz de usuario específica, mientras que PAC apoya el uso de múltiples, distintas interfaces de usuario y su variación independiente. Como la mayoría de los sistemas de software solo necesitan una interfaz de usuario, MVC suele ser siempre la primera elección. (33)

PAC, en contraste, (sólo) es útil si un sistema de software se divide en varios subsistemas en gran medida independientes pero a veces cooperantes, cada uno de los cuales sugiere su propio paradigma de interfaz de usuario. (33)

### 1.12.5 Estilos y patrones arquitectónicos y de diseño

#### Estilo arquitectónico

- ✓ Describen el esqueleto estructural y general para aplicaciones.
- ✓ Son independientes del contexto al que pueden ser aplicados.
- ✓ Cada estilo es independiente de los otros.
- ✓ Expresan técnicas de diseño desde una perspectiva que es independiente de la situación actual de diseño.
- ✓ Son una categorización de sistemas.
- ✓ Expresan componentes y sus relaciones.

#### Patrón arquitectónico

- ✓ Existen en varios rangos de escala, comenzando con patrones que definen la estructura básica de una aplicación.
- ✓ Partiendo de la definición de *patrón*, requieren de la especificación de un contexto del problema.
- ✓ Depende de patrones más pequeños que contiene, patrones con los que interactúa, o de patrones que lo contengan.
- ✓ Expresan un problema recurrente de diseño muy específico, y presenta una solución para él, desde el punto de vista del contexto en el que se presenta.
- ✓ Son soluciones generales a problemas comunes.
- ✓ Son una plantilla de construcción.

#### Patrón de diseño

- ✓ Esquema para refinar subsistemas o componentes.
- ✓ Expresan colaboraciones entre clases e instancias.

En la Figura 1.3 se puede apreciar el diferente nivel de abstracción entre estilos arquitectónicos, patrones arquitectónicos y patrones de diseño. Donde los estilos arquitectónicos poseen un mayor nivel de abstracción.



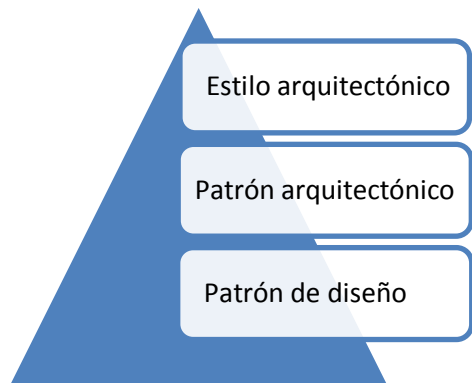


Figura 1.3 Estilos y patrones arquitectónicos y de diseño según nivel de abstracción.

### 1.12.6 Representación de la Arquitectura de Software

Alcanzar una arquitectura estable que dé garantías sobre la viabilidad del proyecto, se considera el punto de transición entre lo que se suele denominar la fase de ingeniería (definición del producto y su solución) y la fase de producción (construcción, integración, evaluación y entrega del producto). En la primera se toman las decisiones más importantes mientras que en la segunda se realizan los mayores gastos y esfuerzos. (7)

Desde un punto de vista de gestión del proceso de desarrollo, se pueden identificar dos dimensiones para manejar la arquitectura:

- ✓ descripción de arquitectura: subconjunto del modelo de diseño del software. Incluye elementos significativos de la arquitectura y excluye el diseño de las componentes básicas. Resuelve decisiones sobre qué desarrollar, qué reutilizar y qué comprar. Contiene notación ad hoc (textos y gráficos) necesaria para comprender los modelos. Debe incluir también los criterios de evaluación de la arquitectura.
- ✓ versión estable de arquitectura: subconjunto suficiente de componentes ejecutables que permiten demostrar lo antes posible que el método de solución (diseño, tecnología, tiempo y costes estimados) puede resolver satisfactoriamente la definición del producto (objetivos, alcance, rendimiento, beneficios, calidad).

Desde una perspectiva técnica, la arquitectura engloba requisitos críticos, decisiones de diseño, componentes de código fuente, y componentes ejecutables, información que debe ser modelada e incluida

en el documento de descripción de arquitectura. Este contexto de información puede ser representado a través de las siguientes vistas y diagramas UML:

- ✓ vista de casos de uso: describe los casos de uso críticos de la arquitectura; puede ser modelado estáticamente a través del diagrama de casos de uso.
- ✓ vista de diseño: describe los componentes del modelo de diseño significativos para la arquitectura, es decir, aquellos que aportan la estructura y funcionalidad básica del sistema; puede ser modelado estáticamente mediante diagramas de clases y objetos.
- ✓ vista de proceso: describe interacciones en tiempo de ejecución de los componentes de la arquitectura en un entorno distribuido, incluyendo distribución lógica de procesos, hilos de control, comunicación entre procesos; puede ser modelado estáticamente a través del diagrama de distribución.
- ✓ vista de componente: describe los componentes del conjunto de implementación (código fuente) significativos para la arquitectura desde la perspectiva de los programadores; puede incluir componentes de prueba, comerciales, de simulación, que luego pueden o no ser considerados en la vista de entrega; puede ser modelado estáticamente a través del diagrama de componente.
- ✓ vista de entrega: describe los componentes ejecutables de la arquitectura, incluyendo la correspondencia entre procesos lógicos y recursos físicos del entorno de explotación; puede ser modelado estáticamente a través del diagrama de distribución.

### 1.12.7 El Modelo de “4 + 1” Vistas de la Arquitectura de Software

La arquitectura del software se trata de abstracciones, de descomposición y composición, de estilos y estética. También tiene relación con el diseño y la implementación de la estructura de alto nivel del software.

Los diseñadores construyen la arquitectura usando varios elementos arquitectónicos elegidos apropiadamente. Estos elementos satisfacen la mayor parte de los requisitos de funcionalidad y performance del sistema, así como también otros requisitos no funcionales tales como confiabilidad, escalabilidad, portabilidad y disponibilidad del sistema.

En 1995 Philippe Kruchten propuso su célebre modelo “4+1”, vinculado al Rational Unified Process (RUP), que define cuatro vistas diferentes de la arquitectura de software: La vista lógica, que comprende las abstracciones fundamentales del sistema a partir del dominio de problemas. La vista de proceso: el

conjunto de procesos de ejecución independiente a partir de las abstracciones anteriores. La vista física: un mapeado del software sobre el hardware. La vista de desarrollo: la organización estática de módulos en el entorno de desarrollo. El quinto elemento considera todos los anteriores en el contexto de casos de uso. Lo que académicamente se define como AS concierne a las dos primeras vistas. El modelo 4+1 se percibe hoy como un intento de reformular una arquitectura estructural y descriptiva en términos de objeto y de UML. Con todo, las cuatro vistas de Kruchten forman parte del repertorio estándar de los practicantes de la disciplina. (34)

### **1.12.8 El Rol del Arquitecto de Software**

Los arquitectos diseñan estructuras que encajen con las necesidades humanas. Las estructuras pueden ser ensambladas con palos y piedras o software de computadora y hardware, pero el rol del arquitecto continúa siendo el mismo. Los arquitectos están la mayoría del tiempo escuchando los clientes, entendiendo a profundidad sus necesidades y recursos, investigando y documentando ordenadamente, creando una visión práctica de una estructura y creando un mapa de la misma. Así como se construye una estructura, el arquitecto interviene en favor del cliente, asegurando que el resultado sea fiel al plan y guiando la visión del resultado entre la tempestad de los cambios en el diseño, las crisis y las ambigüedades.

Abogar por el cliente es la piedra angular del rol del arquitecto. Para lograr el rol de un verdadero abogado, el arquitecto necesita un extenso repertorio de elementos de diseño en un aspecto de elección libre de cualquier atadura. Un arquitecto deja de ser un verdadero apoyo al cliente si se encuentra atado a un conjunto de tecnologías, herramientas o metodologías, restringiendo así las soluciones disponibles al cliente. Los estilos arquitectónicos individuales y las preferencias emergen y son impuestas, como siempre han sido, por el cliente, pero estas deben corresponder a los refinamientos y discernimiento de una mente entrenada, no a las elecciones forzadas por los límites de la educación o la experiencia. (35)

El término Arquitecto de Software se ha convertido en el título de moda en toda empresa de sistemas o con un área propia de sistemas, aunque es común que muchas de las tareas relevantes de un proyecto puedan ser perfectamente resueltas con desarrolladores experimentados, sin tener la necesidad de contratar un arquitecto. Muy frecuentemente se tiende a confundir estos dos perfiles, que son abismalmente diferentes. También es importante notar la diferencia entre los “gurúes tecnológicos” y los verdaderos arquitectos. Estas cuestiones aumentan la confusión existente sobre qué es un arquitecto y

cuáles se supone tendrían que ser sus responsabilidades.

**Mapa Conceptual (35)**

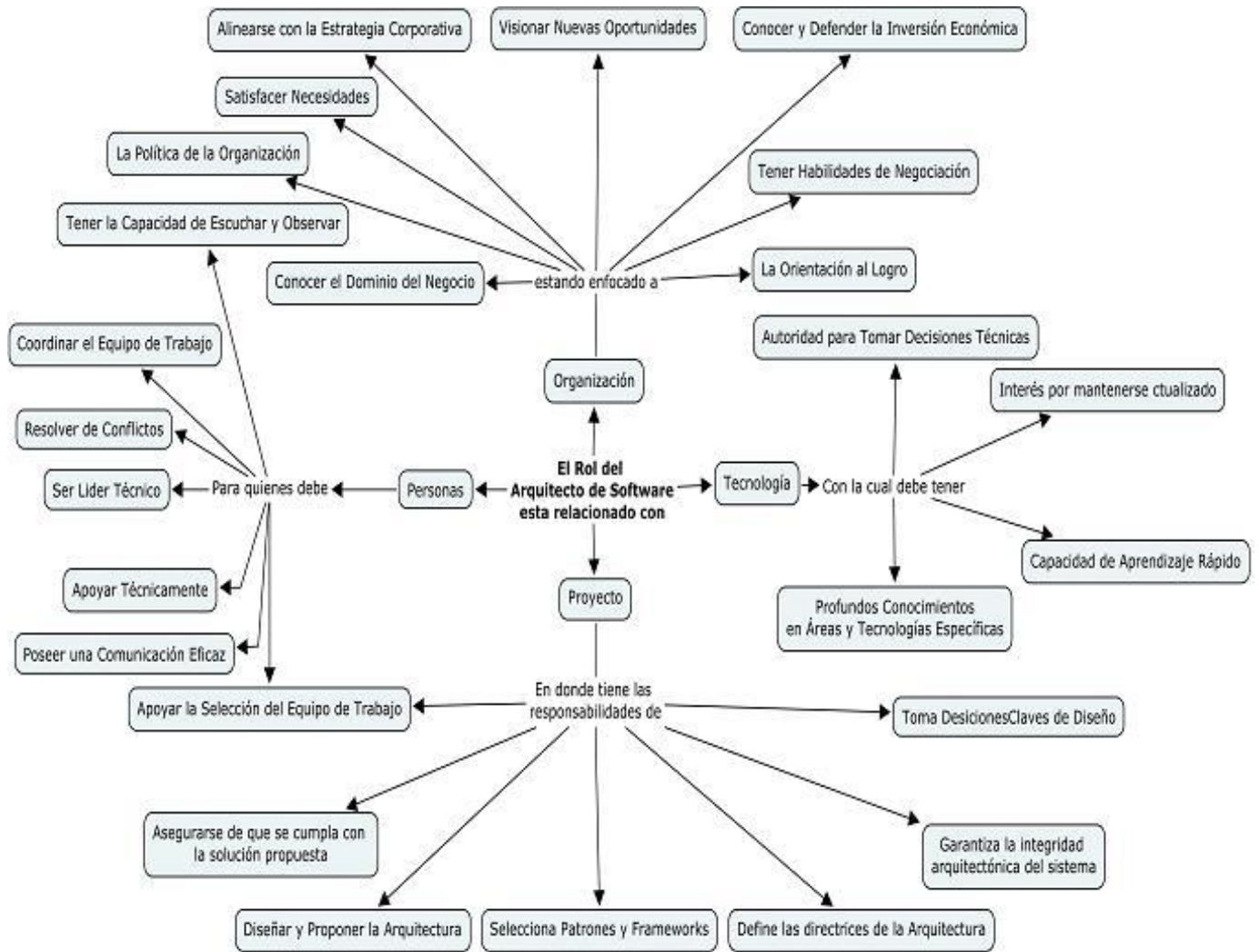


Figura 1.4 Rol del Arquitecto de Software

El rol del arquitecto es un rol crítico en los proyectos, se focaliza en la calidad de servicio y lidera el proceso de definición y la implementación de la arquitectura. El arquitecto reutiliza implementaciones de arquitecturas exitosas, frameworks y patrones de diseño y no es un diseñador en un proyecto. Un Arquitecto es un facilitador, no toma decisiones unilaterales, irracionales, evita riesgos en los proyectos y

agrega valor. Véase la Figura 1.4.

A diferencia de un programador, el Arquitecto de Software debe dominar la mayor cantidad de tecnologías de software y prácticas de diseño, para así poder tomar decisiones adecuadas para garantizar el mejor desempeño, reuso, robustez, portabilidad, flexibilidad, escalabilidad y mantenibilidad de las aplicaciones. Estas decisiones sobre la estructura y dinámica de la aplicación son plasmadas en una notación formal estandarizada como lo es UML.

Como base, el rol de los arquitectos suele comprender las tareas de: definición de las vistas de la arquitectura de una aplicación, dar soporte técnico-tecnológico a desarrolladores, clientes y expertos en negocios, conceptualizar y experimentar con distintos enfoques arquitectónicos, crear documentos de modelos y componentes y especificaciones de interfaces, validar la arquitectura contra requerimientos, suposiciones y además tener una dosis de estrategia y política, o sea, ser, en parte, un consultor.

### **1.13 Metodologías de desarrollo de software**

El desarrollo de software no es sin dudas una tarea fácil. Como resultado a este problema ha surgido una alternativa desde hace mucho: la Metodología. Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Lo hacen desarrollando un proceso detallado con un fuerte énfasis en planificar inspirado por otras disciplinas de la ingeniería.

Las metodologías ingenieriles han estado presentes durante mucho tiempo. No se han distinguido precisamente por ser muy exitosas. Aún menos por su popularidad. La crítica más frecuente a estas metodologías es que son burocráticas. Hay tanto que hacer para seguir la metodología que el ritmo entero del desarrollo se retarda.

En cualquier ámbito de ingeniería hay una fractura entre los responsables de analizar y definir los problemas (necesidades), y los expertos en proveer soluciones (tecnología). Las metodologías nacen para intentar solucionar este conflicto. Su propósito es establecer un contrato social entre todos los participantes en un proyecto para conseguir la solución más eficaz con los recursos disponibles. (36)

#### **1.13.1 Metodologías pesadas**

Hoy en día existen numerosas propuestas que inciden en distintas dimensiones del proceso de desarrollo. Un ejemplo de ellas son las propuestas tradicionales centradas específicamente en el control del proceso. Estas han demostrado ser efectivas y necesarias en un gran número de proyectos, sobre todo aquellos

proyectos de gran tamaño (respecto a tiempo y recursos).

### **1.13.1.1 Metodología RUP (Rational Unified Process)**

El Proceso Unificado Racional (*Rational Unified Process* en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software. Un proceso desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. Sin embargo, el Proceso Unificado es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto. (6)

El Proceso Unificado está basado en componentes, lo cual quiere decir que el sistema software en construcción está formado por componentes software interconectados a través de interfaces bien definidas.

El Proceso Unificado utiliza el Lenguaje Unificado de Modelado (Unified Modeling Language, UML) para preparar todos los esquemas de un sistema software. (6)

No obstante, los verdaderos aspectos definitorios del Proceso Unificado se resumen en tres frases clave – dirigido por casos de uso, centrado en la arquitectura, e iterativo e incremental.

**Caso de uso:** Una descripción de un conjunto de secuencias de acciones, incluyendo variaciones, que un sistema lleva a cabo y que conduce a un resultado de interés para un actor determinado. (6)

**Dirigido por casos de uso:** quiere decir que el proceso de desarrollo sigue un hilo –avanza a través de una serie de flujos de trabajo que parten de los casos de uso. Los casos de uso se especifican, se diseñan, y los casos de uso finales son la fuente a partir de la cual los ingenieros de prueba construyen sus casos de prueba. (6)

#### **Centrado en la arquitectura:**

El concepto de arquitectura de software incluye los aspectos estáticos y dinámicos más significativos del sistema. La arquitectura surge de las necesidades de la empresa, como las perciben los usuarios y los inversores, y se refleja en los casos de uso. Sin embargo, también se ve influida por muchos otros factores, como la plataforma en la que tiene que funcionar el software (arquitectura hardware, sistema

operativo, sistema de gestión de base de datos, protocolos para comunicaciones en red), los bloques de construcción reutilizables de que se dispone (por ejemplo, un marco de trabajo para interfaces gráficas de usuario), consideraciones de implantación, sistemas heredados, y requisitos no funcionales (por ejemplo, rendimiento, fiabilidad). La arquitectura es una vista del diseño completo con las características más importantes resaltadas, dejando los detalles de lado. (6)

Por tanto, los arquitectos moldean el sistema para darle una *forma*. Es esta forma, la arquitectura, la que debe diseñarse para permitir que el sistema evolucione, no sólo en su desarrollo inicial, sino también a lo largo de las futuras generaciones. Para encontrar esa forma, los arquitectos deben trabajar sobre la comprensión general de las funciones clave, es decir, sobre los casos de uso claves del sistema. Estos casos de uso pueden suponer solamente entre el 5 y el 10 por ciento de todos los casos uso, pero son los significativos, los que constituyen las funciones fundamentales del sistema. (6)

#### **Iterativo e incremental:**

El desarrollo de un producto software comercial supone un gran esfuerzo que puede durar entre varios meses hasta posiblemente un año o más. Es práctico dividir el trabajo en partes más pequeñas o mini-proyectos. Cada mini-proyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en el flujo de trabajo, y los incrementos, al crecimiento del producto. Para una efectividad máxima, las iteraciones deben estar controladas; esto es, deben seleccionarse y ejecutarse de una forma planificada. Es por esto por lo que son mini-proyectos.

La arquitectura proporciona la estructura sobre la cual guiar las iteraciones, mientras que los casos de uso definen los objetivos y dirigen el trabajo de cada iteración. (6)

#### **El Ciclo de Vida del Proceso Unificado (6)**

El Proceso Unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo constituye una **versión** del sistema. Véase la Figura 1.5.

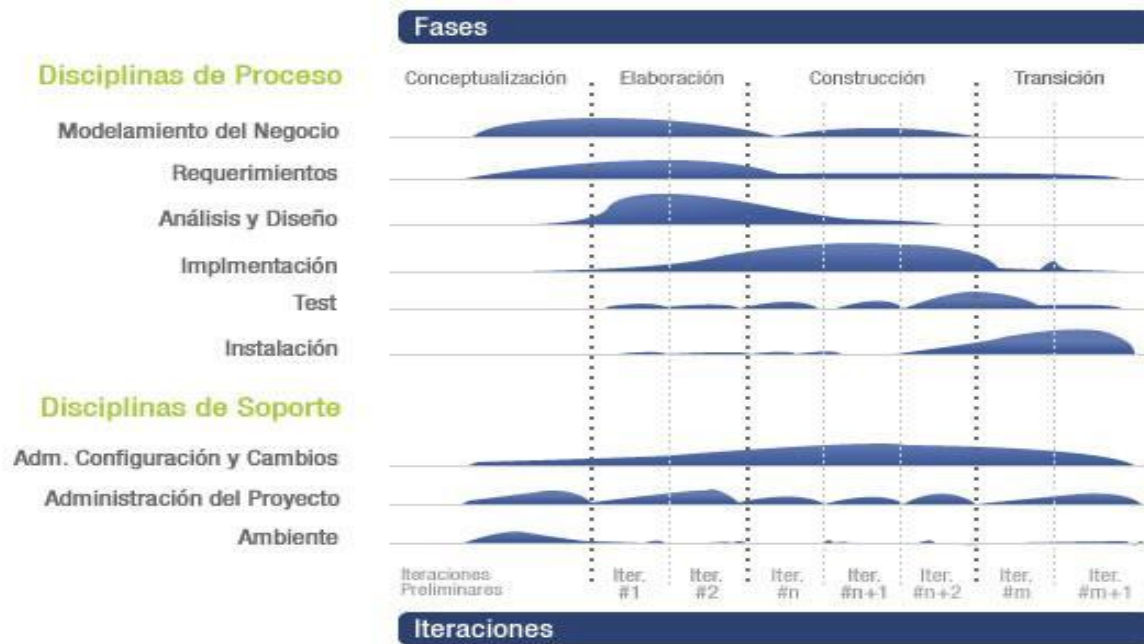


Figura 1.5 RUP en dos dimensiones.

**Fases**

Cada ciclo consta de cuatro fases: **inicio**, **elaboración**, **construcción**, y **transición**.

Cada fase se subdivide en **iteraciones**. En cada iteración se desarrolla en secuencia un conjunto de **disciplinas** o flujos de trabajos.

**Disciplinas**

Cada disciplina es un conjunto de actividades relacionadas (flujos de trabajo) vinculadas a un área específica dentro del proyecto total. Las más importantes son: **Requerimientos**, **Análisis**, **Diseño**, **Codificación**, y **Prueba**.

Los flujos de trabajo desarrollan modelos desde el modelo de casos de uso hasta el modelo de pruebas. Véase la Tabla 1.1.

Disciplina	Modelos
Análisis	Modelo de Análisis
Diseño	Modelo de Diseño - Modelo de Despliegue



Implementación	Modelo de Implementación
Prueba	Modelo de Prueba
Requisitos	Modelo de Casos de Uso

Tabla 1.1. Modelos del Proceso Unificado.

## **Hitos**

Cada fase finaliza con un hito. Cada hito se determina por la disponibilidad de un conjunto de artefactos, es decir un conjunto de modelos o documentos que han sido desarrollados hasta alcanzar un estado predefinido.

Los hitos tienen muchos objetivos. El más crítico es que los directores deben tomar ciertas decisiones antes de que el trabajo continúe con la siguiente fase. Los hitos también permiten controlar la dirección y progreso del trabajo. Al final se obtiene un conjunto de datos a partir del seguimiento del tiempo y esfuerzo consumidos en cada fase. Estos datos son útiles en la estimación del tiempo y los recursos humanos para otros proyectos.

## **Fase de Elaboración**

Durante la fase de **elaboración** se especifican en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura.

Las iteraciones en la fase de elaboración:

- ✓ Establecen una firme comprensión del problema a solucionar.
- ✓ Establecen la fundación arquitectural para el software.
- ✓ Establecen un plan detallado para las siguientes iteraciones.
- ✓ Eliminan los mayores riesgos.

El resultado de esta fase es la **línea base de la arquitectura**.

En esta fase se construyen típicamente los siguientes artefactos:

- ✓ El cuerpo básico del software en la forma de un prototipo arquitectural.
- ✓ Casos de prueba
- ✓ La mayoría de los casos de uso (80%) que describen la funcionalidad del sistema.
- ✓ Un plan detallado para las siguientes iteraciones.

La fase de elaboración finaliza con el **hito de la Arquitectura del Ciclo de Vida**.

Este hito se alcanza cuando el equipo de desarrollo y los stakeholders llegan a un acuerdo sobre:

- ✓ Los casos de uso que describen la funcionalidad del sistema.
- ✓ La línea base de la arquitectura
- ✓ Los mayores riesgos han sido mitigados
- ✓ El plan del proyecto

Al alcanzar este hito debe poder responderse a preguntas como:

- ✓ ¿Se ha creado una línea base de la arquitectura? ¿Es adaptable y robusta? ¿Puede evolucionar?
- ✓ ¿Se han identificado y mitigado los riesgos más graves?
- ✓ ¿Se ha desarrollado un plan del proyecto hasta el nivel necesario para respaldar una agenda, costes, y calidad realistas?
- ✓ ¿Proporciona el proyecto, una adecuada recuperación de la inversión?
- ✓ ¿Se ha obtenido la aprobación de los inversores?

### 1.13.2 Metodologías ágiles

En una reunión celebrada en febrero de 2001 en Utah-EEUU, nace el término “ágil” aplicado al desarrollo de software. En esta reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas. Varias de las denominadas metodologías ágiles ya estaban siendo utilizadas con éxito en proyectos reales, pero les faltaba una mayor difusión y reconocimiento.

Tras esta reunión se creó *The Agile Alliance*, una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue el Manifiesto Ágil, un documento que resume la filosofía “ágil”. (37)

#### **El Manifiesto Ágil (37)**

El Manifiesto comienza enumerando los principales valores del desarrollo ágil. Se valora:

- ✓ Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas.
- ✓ Desarrollar software que funciona más que conseguir una buena documentación.
- ✓ La colaboración con el cliente más que la negociación de un contrato.
- ✓ Responder a los cambios más que seguir estrictamente un plan.

Los valores anteriores inspiran los doce principios del manifiesto. Estos principios son las características que diferencian un proceso ágil de uno tradicional.

- I. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.*
- II. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.*
- III. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.*
- IV. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.*
- V. Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.*
- VI. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.*
- VII. El software que funciona es la medida principal de progreso.*
- VIII. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.*
- IX. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.*
- X. La simplicidad es esencial.*
- XI. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.*
- XII. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.*

### **1.13.2.1 eXtreme Programming (XP)**

La metodología Programación Extrema o *Extreme Programming* (XP), es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando

un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. (37)

Las Historias de Usuario: son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.

### **Prácticas XP**

La principal suposición que se realiza en XP es la posibilidad de disminuir la mítica curva exponencial del costo del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione. XP apuesta por un crecimiento lento del costo del cambio y con un comportamiento asintótico. Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación disciplinada de las prácticas que se describen a continuación: (37)

#### El juego de la planificación

Es un espacio frecuente de comunicación entre el cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración. Esta práctica se puede ilustrar como un juego, donde existen dos tipos de jugadores: Cliente y Programador. El cliente establece la prioridad de cada historia de usuario, de acuerdo con el valor que aporta para el negocio. Los programadores estiman el esfuerzo asociado a cada historia de usuario. Se ordenan las historias de usuario según prioridad y esfuerzo, y se define el contenido de la entrega y/o iteración, apostando por enfrentar lo de más valor y riesgo cuanto antes. Este juego se realiza durante la planificación de la entrega, en la planificación de cada iteración y cuando sea necesario reconducir el proyecto.

### Entregas pequeñas

La idea es producir rápidamente versiones del sistema que sean operativas, aunque obviamente no cuenten con toda la funcionalidad pretendida para el sistema pero si que constituyan un resultado de valor para el negocio. Una entrega no debería tardar más de 3 meses.

### Metáfora

En XP no se enfatiza la definición temprana de una arquitectura estable para el sistema. Dicha arquitectura se asume evolutiva y los posibles inconvenientes que se generarían por no contar con ella explícitamente en el comienzo del proyecto se solventan con la existencia de una metáfora. El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema.

### Diseño simple

Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. La complejidad innecesaria y el código extra debe ser removido inmediatamente.

### Pruebas

La producción de código está dirigida por las pruebas unitarias. Las pruebas unitarias son establecidas antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema. Los clientes escriben las pruebas funcionales para cada historia de usuario que deba validarse. En este contexto de desarrollo evolutivo y de énfasis en pruebas constantes, la automatización para apoyar esta actividad es crucial.

### Refactorización (*Refactoring*)

La refactorización es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios. La refactorización mejora la estructura interna del código sin alterar su comportamiento externo.

### Programación en parejas

Toda la producción de código debe realizarse con trabajo en parejas de programadores. Según Cockburn y Williams en un estudio realizado para identificar los costos y beneficios de la programación en parejas (38), las principales ventajas de introducir este estilo de programación son: muchos errores son detectados conforme son introducidos en el código (inspecciones de código continuas), por consiguiente la tasa de errores del producto final es más baja, los diseños son mejores y el tamaño del código menor (continua discusión de ideas de los programadores), los problemas de programación se resuelven más rápido, se posibilita la transferencia de conocimientos de programación entre los miembros del equipo, varias personas entienden las diferentes partes del sistema, los programadores conversan mejorando así el flujo de información y la dinámica del equipo, y finalmente, los programadores disfrutan más su trabajo. Dichos beneficios se consiguen después de varios meses de practicar la programación en parejas.

### 1.13.3 Comparación entre las metodologías ágiles y las pesadas

A continuación se muestra una comparación entre las metodologías ágiles y las metodologías pesadas. Véase la Tabla 1.2.

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparadas para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo de desarrollo).	Impuestas externamente.
Proceso menos controlado, con menos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (<10 integrantes) y	Grupos grandes y posiblemente distribuidos.

trabajando en el mismo sitio.	
Pocos artefactos.	Más artefactos.
Pocos roles.	Más roles.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

Tabla 1.2. Comparación entre las metodologías ágiles y las metodologías pesadas.

#### 1.13.4 Fundamentación de la Metodología a utilizar

Para documentar el presente trabajo se decidió utilizar RUP, dado a que es una metodología centrada en la arquitectura y propone una serie de artefactos bien definidos para manejar la AS.

### 1.14 Proceso Unificado de Desarrollo y Arquitectura de Software

En esta sección se aborda la AS según RUP.

#### 1.14.1 Casos de uso y arquitectura

Los casos de uso ayudan a llevar a cabo el desarrollo iterativo. Cada iteración excepto quizá la primera de todas en un proyecto, se dirige por los casos de uso a través de todos los flujos de trabajo, de los requisitos al diseño y a la prueba, obteniendo un incremento. Cada incremento del desarrollo es por tanto una realización funcional de un conjunto de casos de uso. En otras palabras, en cada iteración se identifican e implementan unos cuantos casos de uso.

Los casos de uso también ayudan a idear la arquitectura. Mediante la selección del conjunto correcto de casos de uso –los casos de uso significativos arquitectónicamente- para llevarlos a cabo durante las primeras iteraciones, se puede implementar un sistema con una arquitectura estable, que pueda utilizarse en muchos ciclos de desarrollo subsiguientes.

Si el sistema proporciona los casos de uso correctos –casos de uso de alto rendimiento, calidad y facilidad de utilización- los usuarios pueden emplearlo para llevar a cabo sus objetivos. Pero, ¿cómo conseguirlo? La respuesta, como ya se ha sugerido, es construir una arquitectura que permita implementar los casos de uso de una forma económica, ahora y en el futuro. (6)

Se esclarecerá cómo sucede esta interacción, observando primero qué influye en la arquitectura (véase la

Figura 1.6) y después qué influye en los casos de uso.



Figura 1.6 Existen diferentes tipos de requisitos y productos que influyen en la arquitectura, aparte de los casos de uso. También son de ayuda en el diseño de una arquitectura la experiencia de trabajos anteriores y las estructuras que se puedan identificar como patrones de la arquitectura.

La arquitectura no sólo se ve condicionada por los casos de uso arquitectónicamente significativos, sino también por los siguientes factores:

- ✓ Sobre qué productos software del sistema se quiere desarrollar, como sistemas operativos o sistemas de gestión de base de datos concretos.
- ✓ Qué productos de middleware (capa intermedia) se quieren utilizar.
- ✓ Qué sistemas heredados se quieren utilizar en el sistema.
- ✓ A qué estándares y políticas corporativas se debe adaptar.
- ✓ Requisitos no funcionales generales (no específicos de los casos de uso), como los requisitos de disponibilidad, tiempo de recuperación, o uso de memoria.
- ✓ Las necesidades de distribución especifican cómo distribuir el sistema.



La arquitectura se desarrolla en iteraciones de la fase de elaboración. La siguiente podría ser una aproximación simplificada. Comienza determinando un diseño de alto nivel para la arquitectura. Después se forma la arquitectura en un par de construcciones dentro de la primera iteración.

En la primera construcción, se trabaja con las partes generales de la aplicación (es decir, se selecciona el software del sistema, el middleware, los sistemas heredados, los estándares y las políticas de uso). Se decide qué nodos contendrá el modelo de desarrollo y cómo deben interactuar entre ellos. También se decide cómo manejar los requisitos generales no funcionales, así como la disponibilidad de estos requisitos. Con la primera pasada es suficiente para tener una visión general del funcionamiento de la aplicación.

En la segunda construcción, se trabaja con los aspectos de la arquitectura específicos de la aplicación. Se escogen un conjunto de casos de uso relevantes en cuanto a la arquitectura, se capturan los requisitos, se analizan, se diseñan, se implementan y se prueban. El resultado serán nuevos subsistemas implementados como componentes del desarrollo que soportan los casos de uso seleccionados. Pueden existir también algunos cambios en los componentes significativos de la arquitectura que se implementaron en la primera entrega (cuando no se pensó en términos de casos de uso). Los componentes nuevos o cambiados se desarrollan para realizar los casos de uso, y de esta forma la arquitectura se adapta para ajustarse mejor a los casos de uso. Entonces se elabora otra construcción, y así sucesivamente hasta terminar con las iteraciones. Si este final de las iteraciones tiene lugar en el final de la fase de elaboración, se habrá conseguido una arquitectura estable.

Cuando se tiene una arquitectura estable, se puede implementar la funcionalidad completamente realizando el resto de casos de uso durante la fase de construcción. Los casos de uso implementados durante la fase de construcción se desarrollan utilizando como entradas los requisitos de los clientes y de los usuarios (véase la Figura 1.7), pero los casos de uso están también influenciados por la arquitectura elegida en la fase de elaboración.

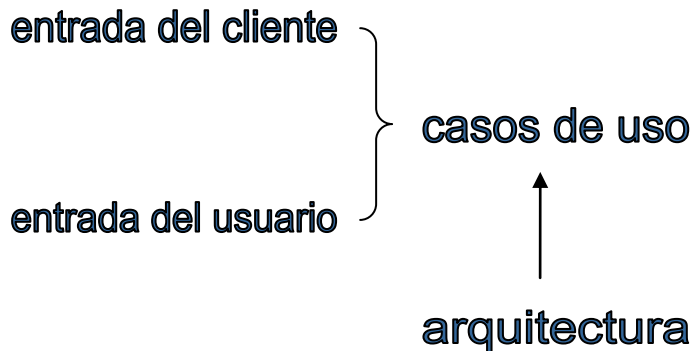


Figura 1.7 Los casos de uso pueden ser desarrollados de acuerdo a las entradas de los clientes y de los usuarios. No obstante, también se ven influenciados por la arquitectura seleccionada.

Así, por una parte, la arquitectura está influenciada por aquellos casos de uso que se quiere el sistema soporte. Los casos de uso conducen la arquitectura. Por otra parte, se utiliza el conocimiento de la arquitectura para hacer mejor el trabajo de captura de requisitos, para obtener casos de uso. La arquitectura guía los casos de uso (véase la Figura 1.8).

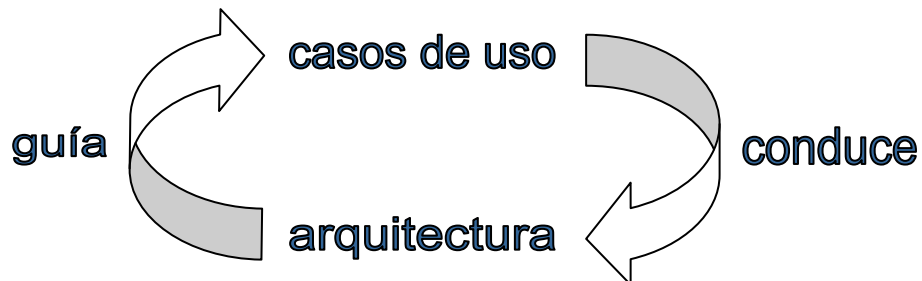


Figura 1.8 Los casos de uso conducen el desarrollo de la arquitectura, y la arquitectura indica qué casos de uso pueden realizarse.

### 1.14.2 Línea base de la arquitectura

La arquitectura se desarrolla mediante iteraciones, principalmente en la etapa de elaboración.

El resultado de la fase de elaboración es la **línea base de la arquitectura** – un esqueleto del sistema con pocos músculos de software.

Los casos de uso que son relevantes para la arquitectura son resumidamente aquellos que mitigan los mayores riesgos del proyecto, aquellos que son más importantes para el usuario, y aquellos que ayudan a cubrir todas las funcionalidades significativas.

Al final de la fase de elaboración se han desarrollado modelos del sistema que representan los casos de uso más importantes y sus realizaciones desde el punto de vista de la arquitectura.

Esta agregación de modelos es **la línea base de la arquitectura**. Es un sistema pequeño y delgado. Tiene las versiones de todos los modelos que un sistema terminado contiene al final de la fase de construcción. Incluye el mismo esqueleto de subsistemas, componentes y nodos que un sistema definitivo, pero no existe toda la musculatura. Es un sistema ejecutable.

### 1.14.3 Descripción de la arquitectura

La línea base de la arquitectura, es la versión interna del sistema al final de la fase de elaboración. El conjunto de modelos que describen esta línea base se denomina **Descripción de la Arquitectura**.

El papel de la descripción de la arquitectura es guiar al equipo de desarrollo a través del ciclo de vida del sistema.

La descripción de la arquitectura puede adoptar diferentes formas. Puede ser un extracto de los modelos que son parte de la línea base de la arquitectura, o puede ser una reescritura de los extractos de forma que sea más fácil leerlos.

La descripción de la arquitectura tiene cinco secciones, una para cada modelo: una vista del modelo de casos de uso, una vista del modelo de análisis (opcional / descartable), una vista del modelo de diseño, una vista del modelo de despliegue, y una vista del modelo de implementación.

### 1.14.4 Vistas de la arquitectura según RUP

La descripción de la arquitectura tiene cinco secciones, una para cada modelo. Tiene una vista del modelo de casos de uso, una vista del modelo de análisis (que no siempre se mantiene), una vista del modelo de diseño, una vista del modelo de despliegue, y una vista del modelo de implementación. No incluye una vista del modelo de prueba porque no desempeña ningún papel en la descripción de la arquitectura, y solo se utiliza para verificar la línea base de la arquitectura.

**La vista de la arquitectura del modelo de casos de uso**

La vista de la arquitectura del modelo de casos de uso presenta los actores y casos de uso más importantes (o escenarios de esos casos de uso). (6)

**La vista de la arquitectura del modelo de diseño**

La vista de la arquitectura del modelo de diseño presenta los clasificadores más importantes para la arquitectura pertenecientes al modelo de diseño: los subsistemas e interfaces más importantes, así como algunas pocas clases muy importantes, fundamentalmente las clases activas.

**La vista de la arquitectura del modelo de despliegue**

El modelo de despliegue define la arquitectura física del sistema por medio de nodos interconectados. Estos nodos son elementos hardware sobre los cuales pueden ejecutarse elementos software. Con frecuencia se conoce cómo será la arquitectura física del sistema antes de comenzar su desarrollo. Por tanto, se puede modelar los nodos y las conexiones del modelo de despliegue tan pronto como comience el flujo de trabajo de los requisitos.

**La vista de la arquitectura del modelo de implementación**

El modelo de implementación es una correspondencia directa de los modelos de diseño y de despliegue. Cada subsistema de servicio del diseño normalmente acaba siendo un componente por cada tipo de nodo en el que deba instalarse –pero no siempre es así-. A veces el mismo componente puede instanciarse y ejecutarse sobre varios nodos. Hay lenguajes que proporcionan construcciones para el empaquetado de los componentes, como es el caso de los JavaBeans. En otros casos, las clases se organizan en ficheros con el código que representa los diferentes componentes. (6)

**1.15 Selección de CMS para el estudio**

Dadas las funcionalidades que presentan los Sistemas de Gestión de Contenidos que posibilitan la creación y gestión de portales web de una forma rápida y eficiente, se decidió realizar un estudio detallado sobre una serie de ejemplares de estos sistemas.

La selección, implantación y puesta en marcha de una herramienta para gestión de contenidos es fruto de un estudio y de un análisis detallado de la organización que lo instala, de los objetivos de la misma, de los

procesos de trabajo y recursos de información que utiliza, y de los usuarios que van a usarlo. (39)

En consecuencia se seleccionaron para la investigación los principales Sistemas de Gestión de Contenidos de código abierto que tuvieran tecnologías en común con el Módulo de Juego Online (principal atracción de la comunidad ajedrecística). Para ello se tomaron los ganadores del concurso “2007 Open Source CMS Award” en la categoría: “Mejor CMS de código abierto escrito en lenguaje PHP” (40). Los ganadores fueron: Joomla!, Drupal y e107; en el año 2008 se ratificaron Drupal y Joomla!

### 1.15.1 Joomla!

#### **Requisitos del Sistema (41)**

Servidor de aplicación: Apache (recomendado), servidor con soporte para PHP.

Base de Datos: MySQL

Licencia: GNU GPL v2

Lenguaje de Programación: PHP

Servidor Web: Apache

Sistema Operativo: Multiplataforma

Joomla es el más popular CMS de código abierto. La etimología de Joomla proviene del swahili (Jumla) y significa "Todos Juntos". Este CMS proviene de Mambo, programa creado originalmente por la Compañía Australiana Miro. (42)

Organización del sitio web: Joomla está preparado para organizar eficientemente los contenidos de su sitio en secciones y categorías, lo que facilita la navegabilidad para los usuarios y permite crear una estructura sólida, ordenada y sencilla para los administradores.

Escalabilidad e implementación de nuevas funcionalidades: Joomla ofrece la posibilidad de instalar, desinstalar y administrar componentes y módulos, que agregarán servicios de valor a los visitantes de su sitio web.

En lo que refiere a seguridad, Joomla es uno de los pocos sistemas de gestión de contenidos que cuenta con la suficiente participación activa como para generar soluciones rápidas y precisas ante la presencia de bugs (vulnerabilidades) que vayan surgiendo.

### 1.15.2 Drupal

**Requisitos del Sistema:**

Servidor de aplicación: PHP 4.3.5+

Base de Datos: MySQL, PostgreSQL

Licencia: GNU GPL

Lenguaje de Programación: PHP

Servidor Web: Apache, IIS (Internet Information Server)

Sistema Operativo: Multiplataforma

Drupal es un sistema de gestión de contenido modular y muy configurable.

Es un programa de código abierto, con licencia GNU/GPL, escrito en PHP, desarrollado y mantenido por una activa comunidad de usuarios. Destaca por la calidad de su código y de las páginas generadas, el respeto de los estándares de la web, y un énfasis especial en la usabilidad y consistencia de todo el sistema. (43)

El diseño de Drupal es especialmente idóneo para construir y gestionar comunidades en Internet. No obstante, su flexibilidad y adaptabilidad, así como la gran cantidad de módulos adicionales disponibles, hace que sea adecuado para realizar muchos tipos diferentes de sitios webs.

### 1.15.3 e107

**Requisitos del Sistema:**

Servidor de aplicación: Apache (recomendado), IIS

Base de Datos: MySQL

Licencia: GNU GPL

Lenguaje de Programación: PHP, JavaScript, XML, XHTML 1.1

Servidor Web: Apache, IIS

e107 es un Sistema de Manipulación de Contenidos (CMS) escrito en PHP, que usa MySQL como base de datos. Es completamente gratuito y está en constante desarrollo. (44)

Desarrollado desde finales de 1998, e107 se ha hecho un espacio importante en el mundo de los CMS de libre distribución. No es de los más conocidos pero incorpora una serie de características muy depuradas

a la hora de la administración de contenidos que lo convierten en uno de los favoritos en muchos lugares del mundo.

#### 1.15.4 Comparación entre los CMS estudiados

##### Rendimiento

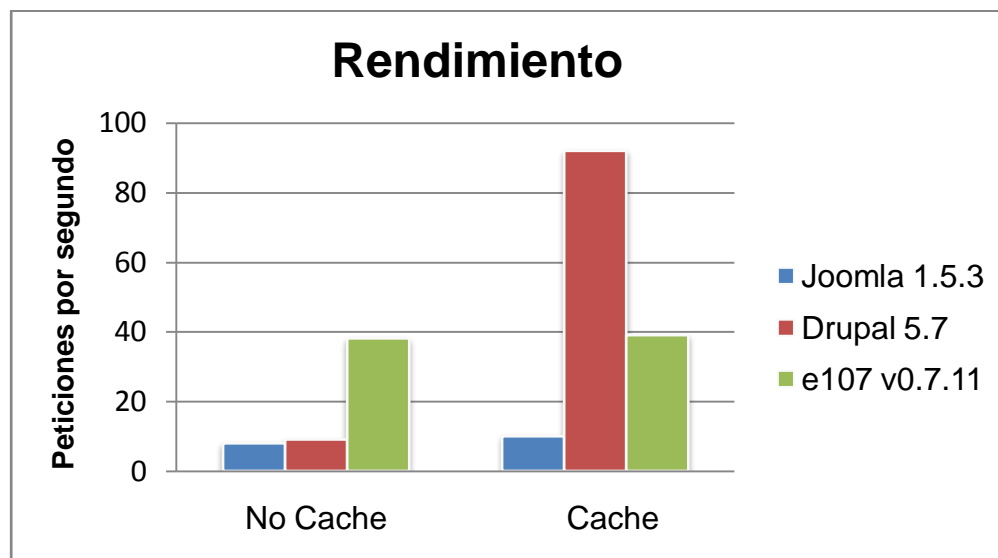


Figura 1.9 Benchmark: Joomla, Drupal, e107.

El caché avanzado, la replicación de bases de datos, balance de carga, caché de las páginas y exportación de contenido estático, son las características que más importancia se le concede a la hora de evaluar el rendimiento de un CMS. (45)

En un benchmark (técnica utilizada para medir el rendimiento de un sistema) realizado con la herramienta Apache Benchmark (ab) en una Pentium IV 3GHz con 512 MB de RAM y SO (Sistema Operativo) GNU/Linux. Software: Apache 2.2.3, PHP 5.2.0, MySQL 5.0.3, Drupal 5.7, Joomla 1.5.3 y e107 v07.11. Se obtuvieron los resultados que se muestran en la Figura 1.9.

Para un total de 1000 peticiones con una concurrencia de 5, con la caché desactivada el CMS e107 muestra un rendimiento superior ya que es capaz de servir correctamente un mayor número de peticiones por segundo; sin embargo, cuando la caché está activada Drupal alcanza un rendimiento que duplica lo alcanzado por e107.

Resultados similares se obtuvieron en (45).

### Total de líneas de Código Fuente (46)

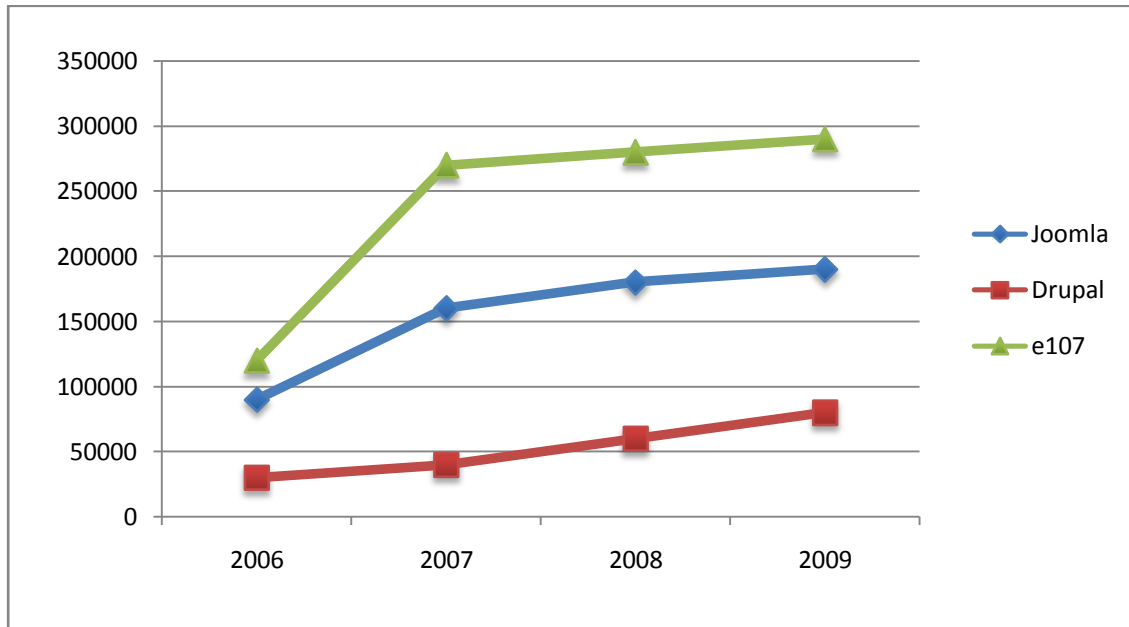


Figura 1.10 Total de líneas de código en los CMS: Drupal, Joomla! y e107, en el período 2006 - 2009.

En la figura se muestra el total de líneas de código fuente en el núcleo de los proyectos: Drupal, Joomla! y e107, en el período 2006 - 2009. Se puede apreciar cómo Drupal es el que menos líneas de código fuente presenta en su núcleo. Véase la Figura 1.10.

### Bases de Datos soportadas

Es conveniente que un CMS soporte más de una base de datos. Desde Enero del 2008 la empresa informática Sun Microsystems anunció la adquisición de MySQL, vaticinando que mantendrá un esquema de licenciamiento dual (47). Por su parte PostgreSQL es publicado bajo la licencia BSD (Berkeley Software Distribution) (48).

Tanto Joomla! como e107 soportan MySQL; Drupal soporta además PostgreSQL.



## Componentes orientados al ajedrez

### Joomla

Prince Clan Chess Club: permite hospedar partidas de ajedrez por correspondencia en una instalación de Joomla.

PC Chess: posibilita a los usuarios de Joomla jugar partidas de ajedrez.

### Drupal

Chess: módulo que permite a dos usuarios jugar una partida de ajedrez, implementado para las versiones 5.x y 6.x de Drupal. (49)

Chessboard Renderer: módulo que permite la inserción de diagramas de ajedrez en el sitio, implementado para la versión 5.x de Drupal. (50)

VChess: módulo que posibilita a los usuarios de Drupal jugar partidas de ajedrez. Cuenta con estadísticas por usuario. Actualmente en desarrollo para la versión 6.x de Drupal.

### 1.15.5 Elección de un CMS para el ajedrez

Dadas las ventajas que posee Drupal en cuanto a rendimiento, bases de datos soportadas, calidad del código fuente y estándares para la confección de sus módulos, se determinó que es el más apropiado dentro de los CMS de código abierto para adaptar sus potencialidades al mundo del ajedrez.

### 1.16 Plataforma LAMP

Las tecnologías a utilizar fueron seleccionadas en correspondencia con el CMS elegido:

Plataforma **LAMP**, con Gestor de Bases de Datos PostgreSQL como alternativa paralela a MySQL.

A finales del año 2000, los miembros del equipo de MySQL David Axmark y Monty Widenius visitaron al editor de O'Reilly Dale Dougherty y le hablaron de un nuevo término: LAMP. Al parecer era ya muy popular en Alemania, donde se empleaba para definir el trabajo conjunto con Linux, Apache, MySQL y uno de los siguientes lenguajes: Perl, Python o PHP. El término LAMP gustó tanto a Dougherty que empezó a promocionarlo desde la posición de extraordinaria influencia de su editorial en el mundo del software libre. (51)

LAMP representa la plataforma web del código abierto. Lo que es más importante, LAMP es la plataforma de elección para el desarrollo y despliegue de aplicaciones web de alto rendimiento. Es sólida y fiable, y si

Apache representa un indicador, a continuación, los sitios LAMP predominan (51). El origen de esta afirmación se puede observar en una encuesta realizada en Netcraft (52).

Algunas de las ventajas que se obtienen de utilizar LAMP son:

- ✓ Soporte a gran cantidad de arquitecturas, como son Intel y compatibles, SPARC, Mips y PPC (Macintosh).
- ✓ Código relativamente sencillo y con pocos cambios de una plataforma a otra.
- ✓ Parches generados en poco tiempo después de encontrarse un agujero de seguridad.
- ✓ Actualizaciones del software vía Internet.
- ✓ Posibilidad de incrementar los servicios y funciones desde el código fuente.

### **Software libre**

Todos los elementos que forman LAMP son software libre, de modo que disfrutan de las ventajas propias del mismo.

#### **1.16.1 Sistema operativo Debian GNU/Linux**

**Debian GNU/Linux** es la principal distribución Linux del proyecto Debian, que basa su principio y fin en el software libre.

Creada por el proyecto Debian en el año 1993, la organización responsable de la creación y mantenimiento de la misma distribución, centrado en el kernel Linux y utilidades GNU. Éste también mantiene y desarrolla sistemas GNU basados en otros núcleos (Debian GNU/Hurd, Debian GNU/NetBSD y Debian GNU/kFreeBSD).

Nace como una apuesta por separar en sus versiones el software libre del software no libre. El modelo de desarrollo es independiente a empresas, creado por los propios usuarios, sin depender de ninguna manera de necesidades comerciales. Debian no vende directamente su software, lo pone a disposición de cualquiera en Internet, aunque sí permite a personas o empresas distribuir comercialmente este software mientras se respete su licencia.

Al igual que todas las demás distribuciones de GNU/Linux comparte características que lo hacen muy popular.

Es muy robusto, estable y rápido: Ideal para servidores y aplicaciones distribuidas. A esto se añade que puede funcionar en máquinas con pocos recursos de hardware que no son de última generación.

Es libre: Esto implica no sólo la gratuidad del software, sino también que Linux es modificable y que Linux tiene una gran cantidad de aplicaciones libres en Internet. Todo ello arropado por la inmensa documentación que puede encontrarse en la Red.

No está restringido a personas con grandes conocimientos de informática: Los desarrolladores de Linux han hecho un gran esfuerzo por dotar al sistema de asistentes de configuración y ayuda, además de un sistema gráfico muy potente. Distribuciones Linux como Red Hat/Fedora tienen aplicaciones de configuración similares a las de Windows (53).

### 1.16.2 Servidor Web Apache

Apache es el servidor web por excelencia, con algo más de un 60% de los servidores de internet confiando en él. Entre sus características más sobresalientes están:

- ✓ Fiabilidad: Alrededor del 90% de los servidores con más alta disponibilidad funcionan con Apache.
- ✓ Gratuidad: Apache es totalmente gratuito, y se distribuye bajo la licencia Apache Software License, que permite la modificación del código.
- ✓ Extensibilidad: se pueden añadir módulos para ampliar las ya de por sí amplias capacidades de Apache. Hay una amplia variedad de módulos, que permiten desde generar contenido dinámico (con PHP, Java, Perl, Python...), monitorizar el rendimiento del servidor, atender peticiones encriptadas por SSL, hasta crear servidores virtuales por IP o por nombre (varias direcciones web son manejadas en un mismo servidor) y limitar el ancho de banda para cada uno de ellos. Dichos módulos incluso pueden ser creados por cualquier persona con conocimientos de programación.

Este potente y famoso servidor se basa en el pionero NCSA server, y surgió a partir de diferentes ampliaciones y parches para el mismo (de ahí su nombre, derivación de 'A patchy server'), cuyo desarrollo se estancó a mediados de 1994. Un grupo de administradores web pusieron en marcha una lista de correo y fundaron el Apache Group. Al año, Apache era el número 1 en la lista de Netcraft (52).

### 1.16.3 Gestores de Bases de Datos

A continuación se detallan las alternativas para la gestión de bases de datos en Drupal:

#### **Gestor de Bases de Datos MySQL**

La administración y gestión de la información es uno de los puntos clave del éxito en cualquier entidad empresarial. La informática aporta la tecnología que permite satisfacer la necesidad de control de esta información, pero las empresas no se conforman trabajando con aplicaciones o programas que amontonen la información de forma caótica. Los datos deben organizarse de acuerdo a un proceso previo que comprende el análisis y diseño del modelo de datos, así como la elección y posterior configuración del sistema que soportará la base de datos.

Existen diferentes arquitecturas para los sistemas de gestión de bases de datos, pero la más extendida, y la que más éxito ha tenido, es la arquitectura relacional. MySQL es un servidor de bases de datos relacionales muy rápido y robusto. Desde Enero del 2008 la empresa informática Sun Microsystems anunció la adquisición de MySQL, vaticinando que mantendrá un esquema de licenciamiento dual, GNU GPL o Uso comercial (47).

Este gestor se creó con la rapidez en mente, de modo que no tiene muchas de las características de los gestores comerciales más importantes, como Oracle, Sybase o SQL Server. No obstante, eso no ha impedido que sea el más indicado para aplicaciones que requieren muchas lecturas y pocas escrituras y no necesiten de características muy avanzadas, como es el caso de las aplicaciones web. MySQL está disponible para un enorme número de sistemas operativos.

MySQL AB estima que hay 4 millones de servidores MySQL instalados en el mundo, lo que significa aproximadamente el 20% del mercado. Entre sus clientes destacan Yahoo!, Cisco, NASA, Lucent Technologies, Motorola, Google, Silicon Graphics, HP, Xerox o Sony Pictures. Buena parte de su éxito se debe, sin duda, a formar parte de la tecnología LAMP.

#### **Gestor de Bases de Datos PostgreSQL**

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos de software libre, publicado bajo la licencia BSD.

Algunas de sus principales características son, entre otras:

### Alta concurrencia

Mediante un sistema denominado MVCC (Acceso Concurrente Multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo *commit*. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

### Amplia variedad de tipos nativos

PostgreSQL provee nativamente soporte para:

- ✓ Números de precisión arbitraria.
- ✓ Texto de largo ilimitado.
- ✓ Figuras geométricas (con una variedad de funciones asociadas)
- ✓ Direcciones IP (IPv4 e IPv6).
- ✓ Bloques de direcciones estilo CIDR.
- ✓ Direcciones MAC.
- ✓ Arrays.

Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indizables gracias a la infraestructura GiST de PostgreSQL. Algunos ejemplos son los tipos de datos GIS creados por el proyecto PostGIS.

## **1.16.4 Lenguaje de programación PHP**

Entre las muchas cosas que distinguen la web de los restantes medios de comunicación, está la capacidad de interacción. En este ámbito, las capacidades del HTML, Javascript y demás tecnologías de cliente son bastante reducidas. Una página realmente profesional no puede limitarse a mostrar información y disponer de formularios para conectarse con los usuarios. Esta necesidad se comprendió muy pronto y provocó el nacimiento del protocolo CGI que permite a los navegadores comunicarse con programas alojados en el servidor.

Con los años, no obstante, se comenzaron a percibir diversos problemas con respecto a los CGIs, entre los cuales el menor no era su complejidad. La popularidad de Javascript o Perl llevó a muchas cabezas pensantes a considerar el uso de los lenguajes de script para ejecutar tareas en el servidor. Así nacieron tecnologías como ASP, PHP, JSP o ColdFusion.

### **Diferencias de PHP con respecto a las demás alternativas**

- ✓ Es software libre, lo que implica menores costes y servidores más baratos que otras alternativas, a la vez que el tiempo entre el hallazgo de un fallo y su resolución es más corto. Además, el volumen de código PHP libre es mucho mayor que en otras tecnologías, siendo superado por Perl, que es más antiguo. Esto permite construir sitios realmente interesantes con sólo instalar scripts libres como PHP Nuke (weblog, comunidad o bitácora), osCommerce (comercio electrónico con capacidad multilingüe), eZ publish (sistema de gestión de contenidos), phpBB (foros de discusión) o phpMyAdmin (administración de base de datos MySQL).
- ✓ Es muy rápido. Su integración con la base de datos MySQL, también veloz, le permite constituirse como una de las alternativas más atractivas para sitios de tamaño medio-bajo.
- ✓ Su sintaxis está inspirada en C, ligeramente modificada para adaptarlo al entorno en el que trabaja.
- ✓ Su librería estándar es realmente amplia, lo que permite reducir los llamados 'costes ocultos', uno de los principales defectos de ASP.
- ✓ PHP es relativamente multiplataforma. Funciona en toda máquina que sea capaz de compilar su código, entre ellas diversos sistemas operativos para PC y diversos Unix. El código escrito en PHP en cualquier plataforma funciona exactamente igual en cualquier otra.
- ✓ El acceso a las bases de datos de PHP es muy heterogéneo, pues dispone de un juego de funciones distinto por cada gestor.
- ✓ PHP es suficientemente versátil y potente como para hacer tanto aplicaciones grandes que necesiten acceder a recursos a bajo nivel del sistema como pequeños scripts que envíen por correo electrónico un formulario relleno por el usuario.

### **1.17 Lenguajes Utilizados**

En esta sección se especifican los diferentes lenguajes utilizados a través del texto, a excepción del lenguaje PHP que ya fue especificado dentro de la plataforma LAMP.

#### **1.17.1 HTML**

El HTML, Hyper Text Markup Language (Lenguaje de marcación de Hipertexto) es el lenguaje de marcas de texto utilizado normalmente en la www (World Wide Web). Fue creado en 1986 por el físico nuclear Tim

Berners-Lee; el cual tomó dos herramientas preexistentes: El concepto de Hipertexto (Conocido también como link o ancla) el cual permite conectar dos elementos entre si y el SGML (Lenguaje Estándar de Marcación General) el cual sirve para colocar etiquetas o marcas en un texto que indique como debe verse. HTML no es propiamente un lenguaje de programación como C++, Visual Basic..., sino un sistema de etiquetas. HTML no presenta ningún compilador, por lo tanto algún error de sintaxis que se presente éste no lo detectará y se visualizará en la forma como éste lo entienda.

El entorno para trabajar HTML es simplemente un procesador de texto, como el que ofrecen los sistemas operativos Windows (Bloc de notas), UNIX (el editor vi o ed) o el que ofrece MS Office (Word). El conjunto de etiquetas que se creen, se deben guardar con la extensión .htm o .html

Estos documentos pueden ser mostrados por los visores o "browsers" de paginas Web en Internet, como Netscape Navigator, Mosaic, Opera y Microsoft Internet Explorer. (54)

HTML es un lenguaje de composición de documentos y especificación de ligas de hipertexto que define la sintaxis y coloca instrucciones especiales que no muestra el navegador, aunque si le indica cómo desplegar el contenido del documento, incluyendo texto, imágenes y otros medios soportados. HTML también le indica cómo hacer un documento interactivo a través de ligas especiales de hipertexto, las cuales conectan diferentes documentos -ya sea en su computadora o en otras- así como otros recursos de Internet, como FTP y Gopher. (55)

Los desarrolladores de navegadores dependen del estándar de HTML para programar el software que da formato y despliega documentos de HTML comunes. Los creadores de páginas utilizan el estándar para asegurarse de que sus documentos de HTML son correctos y efectivos. Sin embargo, las fuerzas comerciales han obligado a los desarrolladores a agregar a sus navegadores -Navigator e Internet Explorer, en particular- extensiones no estándares a fin de mejorar el lenguaje. Muchas veces, estas extensiones son la implementación de futuros estándares que aún se encuentran en debate. Las extensiones pueden predecir futuros estándares debido a que muchas personas las utilizan. (55)

El Consorcio World Wide Web (W3C) se formó con el propósito de definir las versiones estándares de HTML. Sus miembros son responsables de planear, someter a revisión y modificar el estándar, basados en la retroalimentación recibida por Internet, para satisfacer mejor las necesidades de las mayorías.

Más allá de HTML, el W3C tiene la completa responsabilidad de estandarizar cualquier tecnología relacionada con el World Wide Web; administran el estándar de HTTP, lo mismo que los relacionados con el direccionamiento de documentos en el Web. Y sugieren a los desarrolladores que se basen en

esquemas estándares para crear extensiones de las tecnologías Web existentes a fin de internacionalizar el estándar de HTML.

### 1.17.2 XML

Aunque los estándares visuales y de interface de usuario son una capa necesaria, no son suficientes para representar y administrar los datos. Hasta hace pocos años, Internet era un simple medio de acceso a texto e imágenes. No había ningún estándar establecido para la búsqueda inteligente, el intercambio de datos, la presentación adaptable ni para la personalización.

Internet debía ir más allá del establecimiento de un estándar de acceso y presentación de información, de forma de poseer un estándar para la comprensión de la información, una forma común de representar los datos para que el software pueda buscar, desplazar, presentar y manipular mejor los datos ocultos en una oscuridad contextual. HTML no puede hacerlo porque es un formato que describe la apariencia que debería tener una página Web, pero no representa datos. Por ejemplo, HTML:

- ✓ No ofrece una forma estándar para que un médico pueda enviar una receta a un farmacéutico.
- ✓ No habilita a un laboratorio médico para publicar datos estadísticos en un formato que puedan analizar todos los receptores.
- ✓ No describe un pago electrónico de forma que todos los receptores puedan descodificarlo y procesarlo.
- ✓ No ofrece una forma estándar de buscar documentos de pleitos sobre un tema determinado en bibliotecas legales.
- ✓ No especifica cómo se pueden transmitir los datos del catálogo de una empresa de forma que un comercial pueda trabajar fuera de línea, mostrar el catálogo a los clientes, recibir pedidos ni enviar dichos pedidos en un formato estándar.

Aunque HTML ofrece amplias facilidades de presentación, no ofrece ninguna forma basada en estándares para administrar los datos. Un estándar de representación de datos ampliaría Internet del mismo modo que el estándar de visualización HTML lo hizo hace pocos años. El estándar de datos sería el vehículo para las transacciones comerciales, la publicación de perfiles de preferencias personales, la colaboración automatizada y el uso compartido de bases de datos. Los historiales médicos, los datos de investigación farmacéutica, las hojas de piezas semiconductoras y los pedidos de compra se escribirían todos en el



mismo formato. Permitirá una gran variedad de nuevos usos, todos basados en una representación estándar para el desplazamiento de datos estructurados por toda la Web tan fácilmente como se desplazan las páginas HTML hoy en día.

En resumen, llegado a un punto en el que HTML dejó de servir para su función inicial, no le quedó más remedio al Consorcio World Wide Web (W3C) que la descripción de un nuevo subconjunto del SGML que sirva para describir contenidos de documentos, al que ha denominado XML, publicando las especificaciones de la versión 1.0 del mismo en 1998. (56)

XML (Lenguaje de Marcas Extensible, Extensible Markup Language) es un lenguaje de marcas que ofrece un formato para la descripción de datos estructurados, el cual conserva todas las propiedades importantes del antes mencionado SGML. Es decir, XML es un metalenguaje, dado que con él se puede definir un lenguaje propio de presentación y, a diferencia del HTML, que se centra en la representación de la información, XML se centra en la información en si misma. La particularidad más importante del XML es que no posee etiquetas prefijadas con anterioridad, ya que es el propio diseñador el que las crea a su antojo, dependiendo del contenido del documento. (56)

XML es un estándar interoperable que apoya la internacionalización, extensión, composición, y persistencia (porque el formato es abierto y también puede ser leído por humanos. XML está respaldado por un amplio abanico de estándares relacionados, entre ellos XSLT (para transformar contenido XML), XQuery (para realizar consultas sobre bases de datos XML); el Modelo de Objeto del Documento (para acceder a un entorno de programación), Esquema XML, y Firma XML y Encriptación XML. La interoperabilidad de XML lo ha convertido en la elección más adecuada para definir tanto el formato de los documentos (por ejemplo SVO o VoiceXML) como los servicios (los basados en SOAP y los basados en HTTP). (57)

### **1.17.3 XHTML**

XHTML es un vocabulario XML. Se construyó de acuerdo con las reglas de la Especificación XML 1.0; los documentos XHTML son documentos XML. XHTML, realmente, no es más que una reformulación de HTML4.01 de acuerdo a las reglas de XML. La siguiente lista describe lo que esta reformulación significa realmente para los usuarios de XHTML:

- ✓ Los documentos XHTML se deben adherir a un grupo de reglas más estricto que incluye poner siempre comillas alrededor de los valores de atributos y anidar los elementos correctamente.

- ✓ XML hace distinción entre mayúsculas y minúsculas; por lo tanto, XHTML también la hace. La Especificación XHTML 1.0 utiliza texto en minúscula para describir todos los elementos y atributos; por lo tanto, necesita hacer lo mismo en sus documentos XHTML.
- ✓ Debido a que los documentos XML están diseñados para separar el contenido del marcado, XHTML se aleja de elementos de formateo, como font, y se acerca al uso de hojas de estilo para organizar la visualización de sus documentos.

Para el W3C, XHTML 1.0 es el primer gran cambio efectuado a HTML desde que se publicó HTML 4.0 en 1997. Ofrece a las páginas web el rigor de XML, y es el pilar sobre el que trabaja W3C para crear estándares que proporcionen páginas web más fuertes en un rango siempre creciente de plataformas de navegadores que incluyen teléfonos móviles, televisores, coches, entre otros. (58)

#### **1.17.4 JavaScript**

JavaScript es un lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C.

Al igual que Java, JavaScript es un lenguaje orientado a objetos propiamente dicho, ya que dispone de Herencia, si bien ésta se realiza siguiendo el paradigma de programación basada en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad.

Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas web.

Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM.

El lenguaje fue inventado por Brendan Eich en la empresa Netscape Communications, que es la que desarrolló los primeros navegadores web comerciales. Apareció por primera vez en el producto de Netscape llamado Netscape Navigator 2.0.

En 1997 los autores propusieron JavaScript para que fuera adoptado como estándar de la European Computer Manufacturers' Association ECMA, que a pesar de su nombre no es europeo sino internacional, con sede en Ginebra. En junio de 1997 fue adoptado como un estándar ECMA, con el nombre de ECMAScript. Poco después también lo fue como un estándar ISO.

JScript es la implementación de ECMAScript de Microsoft, muy similar al JavaScript de Netscape, pero con ciertas diferencias en el modelo de objetos del navegador que hacen a ambas versiones con frecuencia incompatibles.

Para evitar estas incompatibilidades, el World Wide Web Consortium diseñó el estándar Document Object Model (DOM, ó Modelo de Objetos del Documento en castellano), que incorporan Konqueror, las versiones 6 de Internet Explorer y Netscape Navigator, Opera versión 7, y Mozilla desde su primera versión.

### 1.17.5 CSS

Las hojas de estilo en cascada (*Cascading Style Sheets*, CSS) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores.

La idea que se encuentra detrás del desarrollo de CSS es separar la *estructura* de un documento de su *presentación*.

#### Los tres tipos de estilos

CSS proporciona tres caminos diferentes para aplicar las reglas de estilo a una página Web:

1. **Una hoja de estilo externa**, que es una hoja de estilo que está almacenada en un archivo diferente al archivo donde se almacena el código HTML de la página Web. Esta es la manera de programar más potente, porque separa completamente las reglas de formateo para la página HTML de la estructura básica de la página.
2. **Una hoja de estilo interna**, que es una hoja de estilo que está incrustada dentro de un documento HTML. (Va a la derecha dentro del elemento <head>). De esta manera se obtiene el beneficio de separar la información del estilo, del código HTML propiamente dicho. Se puede optar por copiar la hoja de estilo incrustada de una página a otra, (esta posibilidad es difícil de ejecutar si se desea para guardar las copias sincronizadas). En general, la única vez que se usa una hoja de estilo interna, es cuando se quiere proporcionar alguna característica a una página Web en un simple fichero, por ejemplo, si se está enviando algo a la página web.
3. **Un estilo en línea**, que es un método para insertar el lenguaje de estilo de página, directamente, dentro de una etiqueta HTML. Esta manera de proceder no es excesivamente adecuada. El incrustar la descripción del formateo dentro del documento de la página Web, a nivel de código se convierte en una tarea larga, tediosa y poco elegante de resolver el problema de la programación de la página.

Este modo de trabajo se podría usar de manera ocasional si se pretende aplicar un formateo con prisa, al vuelo. No es todo lo claro, o estructurado, que debería ser, pero funciona.

### **Ventajas de usar las hojas de estilo**

Las ventajas de utilizar CSS (u otro lenguaje de estilo) son:

- ✓ Control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo.
- ✓ Los Navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web, con lo que aumenta considerablemente la accesibilidad. Por ejemplo, personas con deficiencias visuales pueden configurar su propia hoja de estilo para aumentar el tamaño del texto o remarcar más los enlaces.
- ✓ Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o incluso a elección del usuario. Por ejemplo, para ser impresa, mostrada en un dispositivo móvil, o ser "leída" por un sintetizador de voz.
- ✓ El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño (siempre y cuando no se utilice estilo en línea).

### **1.17.6 Lenguaje Unificado de Modelado**

Lenguaje Unificado de Modelado (UML, por su sigla en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

### **Diagramas**

En UML 2.0 hay 13 tipos diferentes de diagramas. Para comprenderlos de manera concreta, a veces es útil categorizarlos jerárquicamente

Los **Diagramas de Estructura** enfatizan en los elementos que deben existir en el sistema modelado:

- ✓ Diagrama de clases: es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos.
- ✓ Diagrama de componentes: Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes.
- ✓ Diagrama de objetos: Los diagramas de objetos modelan las instancias de elementos contenidos en los diagramas de clases. Un diagrama de objetos muestra un conjunto de objetos y sus relaciones en un momento concreto.
- ✓ Diagrama de estructura compuesta (UML 2.0): muestra la estructura interna de una clase y las *colaboraciones* que esta estructura hace posibles.
- ✓ Diagrama de despliegue: se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes.
- ✓ Diagrama de paquetes: muestra como un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones.

Los **Diagramas de Comportamiento** enfatizan en lo que debe suceder en el sistema modelado:

- ✓ Diagrama de actividades: representa los flujos de trabajo paso a paso de negocio y operacionales de los componentes en un sistema.
- ✓ Diagrama de casos de uso: es una especie de diagrama de comportamiento.
- ✓ Diagrama de estados: se usan para representar gráficamente máquinas de estados finitos.

Los **Diagramas de Interacción** son un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado:

- ✓ Diagrama de secuencia: es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema. Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada método de la clase.
- ✓ Diagrama de comunicación, que es una versión simplificada del Diagrama de colaboración (UML 1.x): Un diagrama de Comunicación modela las interacciones entre objetos o partes en términos de mensajes en secuencia. Los diagramas de Comunicación representan una combinación de información tomada desde el diagrama de Clases, Secuencia, y Diagrama de casos de uso

describiendo tanto la estructura estática como el comportamiento dinámico de un sistema.

- ✓ Diagrama de tiempos (UML 2.0): es una gráfica de formas de onda digitales que muestra la relación temporal entre varias señales, y cómo varía cada señal en relación a las demás.
- ✓ Diagrama de vista de interacción (UML 2.0): Muestra una cierta vista sobre los aspectos dinámicos de los sistemas modelados.

## 1.18 Herramientas de soporte

Herramientas para darle soporte a alguna metodología, teoría, paradigma, modelo o lenguaje.

### 1.18.1 ab (Apache Benchmark)

**ab** es una herramienta para la evaluación comparativa del rendimiento en un servidor Apache. Da una indicación del número de peticiones por segundo que la instalación de Apache es capaz de servir.

Es importante destacar que la herramienta **ab** no solicita imágenes ni archivos CSS, sólo son obtenidas las páginas HTML generadas dinámicamente (45).

### 1.18.2 Herramientas CASE

Las herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Sirven de apoyo en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

#### Visual Paradigm

El Visual Paradigm es una poderosa herramienta CASE de modelación visual. Utiliza UML para el modelado, permitiendo crear diferentes tipos de diagramas en un ambiente totalmente visual. Es muy sencillo de usar, fácil de instalar y actualizar. Genera código para varios lenguajes. Es una herramienta que puede ser utilizada en la creación de software libre, es nombrada por muchas bibliografías como la herramienta CASE por excelencia del software libre, siendo esta una de las características que dio lugar a

que se seleccionara para el diseño, sin dejar de mencionar que el “Visual Paradigm, además posee:

- ✓ Un entorno de creación de diagramas para UML.
- ✓ Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- ✓ Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- ✓ Capacidades de ingeniería directa e inversa.
- ✓ Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- ✓ Disponibilidad de múltiples versiones, para cada necesidad.
- ✓ Disponibilidad de integrarse en los principales IDEs.
- ✓ Disponibilidad en múltiples plataformas.

Por otra parte, posibilita la representación gráfica de los diagramas permitiendo ver el sistema desde diferentes perspectivas, como el de componentes, despliegue, secuencia, casos de uso, clase, actividad, entre otros. Además, se centra en cómo los componentes del sistema interactúan entre ellos, sin entrar en detalles excesivos, también, permite ver las relaciones entre los componentes del diseño y mejora la comunicación entre los miembros del equipo usando un lenguaje gráfico. Tiene disponible distintas versiones: Enterprise, Professional, Standard, Modeler, Personal y Community. Facilita licencias especiales para fines académicos.

## **1.19 Conclusiones**

A lo largo de este capítulo se han expuesto los principales aspectos dentro de la disciplina AS, se ha definido la Ingeniería de Software Basada en Componentes, y cómo la misma tiene una estrecha relación con la AS. Se analizaron algunos de los portales de ajedrez más difundidos en la actualidad, de donde se desprenden componentes imprescindibles (para un portal orientado al ajedrez) que representan puntos críticos dentro de la presente propuesta. Se realizó un análisis comparativo entre los principales CMS de código abierto que existen en la actualidad, donde Drupal resultó ser el que mejor se ajusta a las exigencias del juego ciencia. Se analizaron además los principales parámetros de calidad que se deben definir dentro de la arquitectura. Se integró la metodología de desarrollo con el objeto de estudio, así como una descripción detallada de las actividades a realizar por el arquitecto que conllevan a la realización de todas las tareas definidas en el presente trabajo.

# Capítulo 2 Línea Base de la Arquitectura

## 2.1 Introducción

La implementación, integración y prueba de la línea base de la arquitectura proporciona seguridad al arquitecto y a otros trabajadores de su equipo, por lo que comprender estos puntos es algo francamente operativo. Esto es algo que no puede obtenerse mediante un análisis y diseño “sobre el papel”. La línea base de la arquitectura de operación proporciona una demostración que funciona para que los trabajadores puedan proporcionar sus retroalimentaciones.

Al final de la fase de elaboración se han desarrollado modelos del sistema que representan los casos de uso más importantes y sus realizaciones, desde la perspectiva de la arquitectura. También se ha decidido con qué estándares se cuenta, qué software del sistema y qué middleware utilizar, qué sistemas heredados reutilizar y qué necesidades de distribución se tienen. Así se cuenta con una primera versión de los modelos de casos de uso, de análisis, de diseño y demás. Esta agregación de modelos es la línea base de la arquitectura; es un sistema pequeño y flaco. Tiene las versiones de todos los modelos que un sistema terminado contiene al final de la fase de construcción. Incluye el mismo esqueleto de subsistemas, componentes y nodos que un sistema definitivo, pero no existe toda la musculatura. No obstante contiene comportamiento y código ejecutable. El sistema flaco se desarrollará para convertirse en un sistema hecho y derecho, quizás con algunos cambios sin importancia en su estructura y comportamiento. Los cambios son menores porque al final de la fase de elaboración se ha definido una arquitectura estable; si no, la fase de elaboración debe continuar hasta que alcance su objetivo. (6)

## 2.2 Arquitectura Propuesta

Antes de especificar la propuesta de arquitectura se define el concepto de framework que estará estrechamente relacionado con la misma: (59)

**Framework:** es un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que se pueden añadir las últimas piezas para construir una aplicación concreta.

Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo, reutilizar



código ya existente y promover buenas prácticas de desarrollo como el uso de patrones.

### ¿Qué ventajas tiene utilizar un 'framework'?

Las que se derivan de utilizar un estándar; entre otras:

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

La utilización de un *framework* en el desarrollo de una aplicación implica un cierto coste inicial de aprendizaje, aunque a largo plazo es probable que facilite tanto el **desarrollo** como el **mantenimiento**.

Para el desarrollo de portales web orientados al ajedrez se propone utilizar el **CMS Drupal** como **framework de desarrollo**, demarcando un **estilo arquitectónico basado en componentes**.

## 2.3 ¿Por qué Drupal?

Drupal posee una serie de características que lo han llevado a ser por dos años consecutivos 2007 – 2008 el ganador del concurso "Open Source CMS Award". (40)

### Características generales (60)

Ayuda on-line Un robusto sistema de ayuda online y páginas de ayuda para los módulos del 'núcleo', tanto para usuarios como para administradores.

Búsqueda Todo el contenido en Drupal es totalmente indexado en tiempo real y se puede consultar en cualquier momento.

Código abierto El código fuente de Drupal está libremente disponible bajo los términos de la licencia GNU/GPL. Al contrario que otros sistemas de 'blogs' o de gestión de contenido propietarios, es posible extender o adaptar Drupal según las necesidades.

Módulos La comunidad de Drupal ha contribuido muchos módulos que proporcionan funcionalidades como

'página de categorías', autenticación mediante jabber, mensajes privados, bookmarks, entre otros.

Personalización Un robusto entorno de personalización está implementado en el núcleo de Drupal. Tanto el contenido como la presentación pueden ser individualizados de acuerdo a las preferencias definidas por el usuario.

URLs amigables Drupal usa el mod\_rewrite de Apache para crear URLs que son manejables por los usuarios y los motores de búsqueda.

### **Gestión de usuarios**

Autenticación de usuarios Los usuarios se pueden registrar e iniciar sesión de forma local o utilizando un sistema de autenticación externo como Jabber, Blogger, LiveJournal u otro sitio Drupal. Para su uso en una intranet, Drupal se puede integrar con un servidor LDAP.

Permisos basados en roles Los administradores de Drupal no tienen que establecer permisos para cada usuario. En lugar de eso, pueden asignar permisos a un 'rol' y agrupar los usuarios por roles.

### **Gestión de contenido**

Control de versiones El sistema de control de versiones de Drupal permite seguir y auditar totalmente las sucesivas actualizaciones del contenido: qué se ha cambiado, la hora y la fecha, quién lo ha cambiado, y más. También permite mantener comentarios sobre los sucesivos cambios o deshacer los cambios recuperando una versión anterior.

Enlaces permanentes (Permalinks) Todo el contenido creado en Drupal tiene un enlace permanente asociado a él para que pueda ser enlazado externamente sin temor de que el enlace falle en el futuro.

Objetos de Contenido (Nodos) El contenido creado en Drupal es, funcionalmente, un objeto (Nodo). Esto permite un tratamiento uniforme de la información, como una misma cola de moderación para envíos de diferentes tipos, promocionar cualquiera de estos objetos a la página principal o permitir comentarios -o no- sobre cada objeto.

Plantillas (Templates) El sistema de temas de Drupal separa el contenido de la presentación permitiendo controlar o cambiar fácilmente el aspecto del sitio web. Se pueden crear plantillas con HTML y/o con PHP.

Sindicación del contenido Drupal exporta el contenido en formato RDF/RSS para ser utilizado por otros sitios web. Esto permite que cualquiera con un 'Agregador de Noticias', tal como *NetNewsWire* o *Radio UserLand* visualice el contenido publicado en la web desde el escritorio.

## **Blogging**

Agregador de noticias Drupal incluye un potente Agregador de Noticias para leer y publicar enlaces a noticias de otros sitios web. Incorpora un sistema de cache en la base de datos, con temporización configurable.

Soporte de Blogger API La API de Blogger permite que un sitio Drupal sea actualizado utilizando diversas herramientas, que pueden ser 'herramientas web' o 'herramientas de escritorio' que proporcionen un entorno de edición más manejable.

## **Plataforma**

Independencia de la base de datos Aunque la mayor parte de las instalaciones de Drupal utilizan MySQL, existen otras opciones. Drupal incorpora una 'capa de abstracción de base de datos' que actualmente está implementada y mantenida para MySQL y PostgreSQL, aunque permite incorporar fácilmente soporte para otras bases de datos.

Multiplataforma Drupal ha sido diseñado desde el principio para ser multi-plataforma. Puede funcionar con Apache o Microsoft IIS como servidor web y en sistemas como Linux, BSD, Solaris, Windows y Mac OS X. Por otro lado, al estar implementado en PHP, es totalmente portable.

Múltiples idiomas y Localización Drupal está pensado para una audiencia internacional y proporciona opciones para crear un portal multilingüe. Todo el texto puede ser fácilmente traducido utilizando una interfaz web, importando traducciones existentes o integrando otras herramientas de traducción como *GNU ettext*.

## **Administración y Análisis**

Administración vía Web La administración y configuración del sistema se puede realizar enteramente con un navegador y no precisa de ningún software adicional.

Análisis, Seguimiento y Estadísticas Drupal puede mostrar en las páginas web de administración informes sobre *referrals* (enlaces entrantes), popularidad del contenido, o de cómo los usuarios navegan por el sitio.

Registros e Informes Toda la actividad y los sucesos del sistema son capturados en un 'registro de eventos', que puede ser visualizado por un administrador.

### Características de comunidad

Comentarios enlazados Drupal proporciona un potente modelo de comentarios enlazados que posibilita seguir y participar fácilmente en la discusión sobre el comentario publicado. Los comentarios son jerárquicos, como en un grupo de noticias o un foro.

Encuestas Drupal incluye un módulo que permite a los administradores y/o usuarios crear encuestas on-line totalmente configurables.

Foros de discusión Drupal incorpora foros de discusión para crear sitios comunitarios vivos y dinámicos.

Libro Colaborativo Esta característica es única de Drupal y permite crear un proyecto o "libro" a ser escrito y que otros usuarios contribuyan contenido. El contenido se organiza en páginas cómodamente navegables.

### Rendimiento y escalabilidad

Control de congestión Drupal incorpora un mecanismo de control de congestión que permite habilitar y deshabilitar determinados módulos o bloques dependiendo de la carga del servidor. Este mecanismo es totalmente configurable y ajustable.

Sistema de Cache El mecanismo de cache elimina consultas a la base de datos incrementando el rendimiento y reduciendo la carga del servidor.

## 2.4 Arquitectura de Drupal

### Pila de tecnología

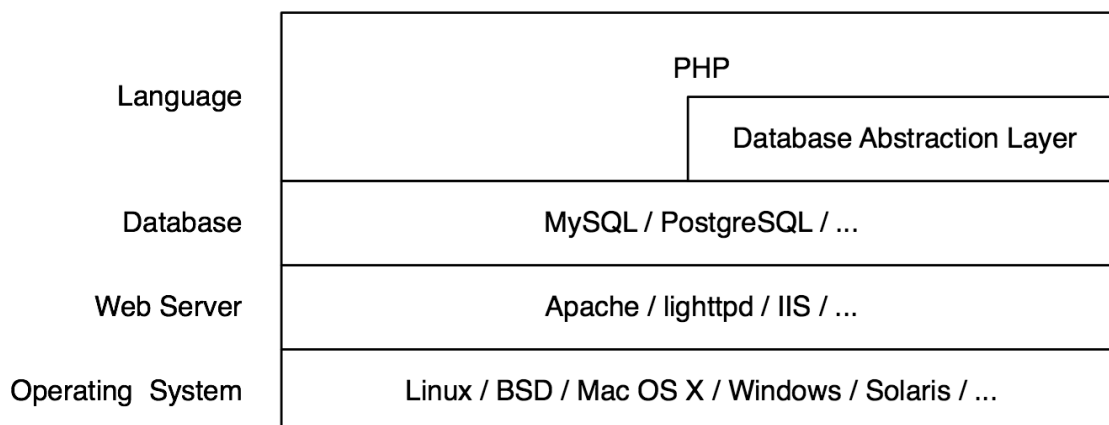


Figura 2.1 Pila de tecnología de Drupal.

Los objetivos de diseño de Drupal incluyen tanto la posibilidad de ejecutar bien en cuentas de alojamiento web de bajo costo, como ser capaz de escalar hasta sitios distribuidos. Estas metas se logran utilizando las tecnologías más populares, y medios cuidadosos de codificación. Véase la Figura 2.1. (61)

El Sistema Operativo está en un nivel tan bajo de la pila que Drupal no se preocupa mucho por este. Drupal se ejecuta con éxito en cualquier sistema operativo que soporte PHP.

El servidor web más ampliamente utilizado con Drupal es Apache, aunque se pueden utilizar otros servidores web (incluyendo Microsoft IIS). Por la larga historia de Drupal con Apache, Drupal se distribuye con *.htaccess* en su raíz que asegura la instalación de Drupal.

URLs limpias -es decir, carentes de signos de interrogación, ampersands, u otros caracteres extraños- se lograrían usando el componente *mod\_rewrite* de Apache.

Drupal implementa una delgada capa de abstracción de bases de datos para proporcionar a los desarrolladores la capacidad de soportar múltiples servidores de bases de datos fácilmente. La intención de esta capa es preservar la sintaxis y el poder de SQL en la medida de lo posible, mientras que se deja a Drupal controlar las piezas de consultas que deben estar escritas en forma diferente para diferentes servidores y proporcionar comprobaciones de seguridad básica. Las bases de datos más ampliamente probadas son MySQL y PostgreSQL.

Drupal está escrito en PHP. Todo el código del núcleo de Drupal se adhiere a un estricto estándar de codificación. (62)

### **Núcleo**

Un ligero marco constituye el núcleo de Drupal. El núcleo es responsable de proporcionar la funcionalidad básica que se utilizará para apoyar a otras partes del sistema.

El núcleo incluye código que permite que el sistema Drupal arranque cuando recibe una petición, una biblioteca de funciones comunes utilizadas con frecuencia por Drupal y los módulos que proporcionan la funcionalidad básica como la gestión de usuario, taxonomía y plantillas.

### **Interfaz Administrativa**

La interfaz administrativa de Drupal está estrechamente integrada con el resto del sitio y, de forma predeterminada, utiliza el mismo tema.

## Módulos

Drupal es un verdadero framework modular. La funcionalidad se incluye en los módulos, que pueden ser activados o desactivados (algunos módulos requeridos no se pueden desactivar). Se añaden características a un sitio web de Drupal, habilitando módulos existentes, instalando módulos escritos por miembros de la comunidad de Drupal, o escribiendo nuevos módulos. De esta manera, sitios web que no necesitan ciertas características pueden ejecutarse, mientras que los que necesitan más pueden añadir tantas funcionalidades como deseen. (61)

Tanto la adición de nuevos tipos de contenido, tales como recetas, entradas de blog, o archivos, y la adición de nuevos comportamientos, como la notificación por correo electrónico, publicación peer-to-peer, y la agregación se manejan a través de módulos. Drupal hace uso del patrón de diseño *inversión de control*, en el que la funcionalidad modular es llamada por el framework en el momento oportuno. Estas oportunidades para los módulos de desplegar su comportamiento se denominan *hooks*.

## Hooks

Los hooks (ganchos) pueden considerarse como eventos internos de Drupal. Los hooks permiten a los módulos enlazar con lo que está sucediendo en el resto de Drupal. (61)

Un hook es una función PHP que se llama `foo_bar()`, donde "foo" es el nombre del módulo (cuyo nombre de archivo por lo tanto, es `foo.module`) y "bar" es el nombre del hook. Cada hook tiene un conjunto definido de parámetros y un tipo de resultado especificado. (63)

La forma más común de introducirse en la funcionalidad del núcleo de Drupal es a través de la implementación de hooks en los módulos.

## Temas

Al crear una página web para enviar a un explorador, en realidad hay dos preocupaciones principales: el montaje de los datos adecuados y enmarcar los datos para la Web. En Drupal, la capa de temas es responsable de crear el código HTML que recibirá el explorador. Drupal alienta la separación entre el contenido y el formato.

## Nodos

Drupal llama nodos a las piezas que componen un sitio web, estas piezas pueden ser páginas, noticias, artículos, mensajes en un blog o foro, imágenes, predicciones de la bolsa de valores, ebooks y todo aquello que constituya contenido.

Se considera contenido a todo aquello que puede ser creado o modificado por los usuarios o los administradores del sitio. Un nodo es la unidad mínima de información en Drupal.

Los nodos también tienen un conjunto básico de propiedades de comportamiento que todos los demás tipos de contenido heredan.

## Bloques

Un bloque es información que se puede habilitar o deshabilitar en una ubicación específica en la plantilla de su sitio web. Por ejemplo, un bloque puede mostrar el número de usuarios activos actuales en su sitio. Los bloques normalmente se colocan en el lateral, cabecera o pie de la plantilla. Un bloque se puede establecer para mostrarse en nodos de cierto tipo, solo en la página principal, o según otros criterios.

## Archivos de Drupal

La comprensión de la estructura de directorios de una instalación predeterminada de Drupal le ayudará a depurar su sitio y enseñarle varias prácticas importantes, tales como donde deben residir los módulos y temas descargados y cómo tener diferentes perfiles en Drupal. Una instalación predeterminada de Drupal tiene la estructura que se muestra en la Figura 2.2. (61)



Figura 2.2 Estructura de carpetas predeterminada de una instalación de Drupal.

Detalles sobre cada elemento en la estructura de carpetas:

**includes:** contiene un conjunto de librerías que Drupal comúnmente usa.

**misc:** almacena imágenes disponibles para una instalación de Drupal, ficheros JavaScript y diversos iconos.

**modules:** contiene los módulos del núcleo, con cada módulo en su propia carpeta. Es mejor no tocar nada en esta carpeta (para agregar módulos adicionales es recomendable hacerlo en el directorio sites).

**profiles:** contiene diferentes perfiles de instalación para un sitio. Si hay otros perfiles además del perfil predeterminado en este subdirectorio, Drupal le pedirá qué perfil desea instalar para la primera instalación del sitio. El objetivo principal de un perfil de instalación es habilitar ciertos módulos del núcleo y contribuidos automáticamente. Un ejemplo sería un perfil de e-commerce que configura automáticamente Drupal como una plataforma de e-commerce.

**scripts:** contiene secuencias de comandos para comprobar la sintaxis, limpieza de código y manejo de casos especiales con cron. No se utiliza en el ciclo de vida de las peticiones de Drupal; estas son shell y secuencias de comandos de Perl.

**sites:** contiene las modificaciones a Drupal en forma de configuración, módulos y temas. Al agregar módulos a Drupal desde el repositorio de módulos contribuido o nuevos módulos desarrollados por usted, van dentro de sites/all/modules. Esto mantiene todas sus modificaciones a Drupal dentro de una sola carpeta.

**themes:** contiene los motores de temas y los temas predeterminados para Drupal.

**index.php:** página PHP que sirve todas las solicitudes de página en una instalación de Drupal. Las rutinas aquí despachan el control al manejador apropiado, que a continuación imprime la página adecuada. (61)

### 2.4.1 Patrones arquitectónicos

*“Drupal es una arquitectura PAC. De hecho es una buena arquitectura PAC. El sistema de menús actúa como el controlador. Acepta la entrada a través de una única fuente (HTTP GET y POST), enruta las solicitudes a las funciones auxiliares apropiadas, extrae los datos de la abstracción (nodos y, en Drupal 5, formas) y luego los empuja a través de un filtro para obtener una presentación (sistema de temas). Incluso tiene múltiples agentes PAC paralelos en forma de bloques que presionan los datos a un filtro común (page.tpl.php).” (64)*



Para una mejor comprensión de este patrón se especifica el sistema de menús de Drupal, así como el proceso que Drupal lleva a cabo para servir una petición y el proceso de asignación de devolución de llamada.

El sistema de menús de Drupal es complejo pero poderoso. El término “sistema de menús” es un poco inexacto. Tal vez sea mejor pensar en el “sistema de menús” según tres responsabilidades principales: la asignación de devolución de llamada, control de acceso y personalización de menú. Código esencial para el sistema de menús se encuentra en `includes/menu.inc`, mientras que el código opcional que permite características tales como la personalización de los menús está en `modules/menu`. (61)

### **Asignación de devolución de llamada**

Cuando un navegador web hace una solicitud para Drupal, da a Drupal una dirección URL. De esta información, Drupal debe averiguar qué código ejecutar y cómo manejar la solicitud. Esto se conoce comúnmente como enrutamiento. Drupal utiliza la última parte de la dirección URL, denominada la ruta de acceso. Por ejemplo, si la dirección URL es `http://example.com/?q=node/3`, la ruta de acceso a Drupal es `nodo/3`.

### **Asignación de direcciones URL a funciones**

El planteamiento general es el siguiente: Drupal pregunta a todos los módulos habilitados para proporcionar una matriz de elementos de menú. Cada elemento de menú consiste en un arreglo introducido por una ruta de acceso y que contiene alguna información sobre esa ruta. Una de las piezas de información que debe proporcionar un módulo es una página de devolución de llamada. Una devolución de llamada en este contexto es simplemente el nombre de una función PHP que se ejecutará cuando el navegador solicita una ruta determinada. Drupal recorre los siguientes pasos cuando recibe una solicitud:

1. Establecer la ruta de acceso de Drupal. Si la ruta de acceso es un alias para una ruta real, Drupal encuentra la ruta real y la utiliza en su lugar.
2. Drupal mantiene un seguimiento de qué rutas de acceso mapean con qué llamadas de retorno en la tabla de base de datos `menu_router` y mantiene un seguimiento de los elementos de menú que son

enlaces en la tabla de menu\_links. Se hace una comprobación para ver si las tablas menu\_router y menu\_links necesitan reconstruirse.

3. Averiguar qué entrada en la tabla de menu\_router se corresponde con la ruta de acceso de Drupal y construir un elemento enrutador que describe la devolución de llamada (callback) a ser llamada.
4. Cargar los objetos necesarios para pasar a la devolución de llamada.
5. Comprobar si el usuario tiene permiso para acceder a la devolución de llamada. Si no, se devuelve un mensaje de “acceso denegado”.
6. Localizar el título del artículo de menú y una descripción para el idioma actual.
7. Cargar los archivos que son necesarios (incluir).
8. Llamar a la devolución de llamada y devolver el resultado, que **index.php** a continuación pasa a través de theme\_page(), resultando en una página web finalizada.

### **Procesamiento de una petición**

La función de devolución de llamada es requerida para procesar y acumular los datos necesarios para completar la solicitud. Por ejemplo, si una solicitud de contenido como <http://example.com/q=node/3> se recibe, la dirección URL se asigna a la función node\_page\_view() en node.module. Posteriormente el procesamiento recuperará los datos de ese nodo de la base de datos y los pondrá en una estructura de datos. A continuación, es el momento de temas (presentación).

### **Presentación de los datos**

La presentación (tema) consiste en transformar en HTML los datos que se han recuperado, manipulado o creado. Drupal utilizará el tema que el administrador ha seleccionado para dar la apariencia correcta a la página web y enviará el HTML resultante al navegador web.

## **2.4.2 API de Drupal**

Las API de Drupal tienen gran importancia para el funcionamiento del CMS. Permite la posibilidad de la validación y de la ejecución de las funciones. Los elementos de las mismas pueden ser quitados, agregados o cambiados. Las API solo son reconocidas por Drupal y el usuario las implementa en dependencia de lo que se quiera realizar en el sitio, son todas las funciones de Drupal. Este ofrece numerosas facilidades a la hora de construir formularios, acceder a base de datos, e integrar el

funcionamiento de los módulos dentro del trabajo normal de Drupal, debido a esto, las API son muy utilizadas para la creación de nuevos módulos, lo que permite personalizar los sitios Web, añadiéndoles características no incorporadas en su distribución por defecto, o mejorando las existentes.

### 2.4.3 Creación de Módulos

Un módulo para Drupal consta de uno o más ficheros, el principal con extensión *.module* debe implementar una interfaz definida por el propio Drupal. Básicamente existen dos tipos de módulos: *de contenido* que son los que definen un nuevo tipo de contenido personalizado y la funcionalidad para su creación, edición y publicación y los módulos *funcionales*, estos últimos tienen disímiles propósitos dependiendo del objetivo para el cual fueron creados.

Para implementar un módulo es necesario crear un archivo PHP, y guardarlo con **nombremodulo.module** (aquí se incluiría la funcionalidad), otro con **nombremodulo.install** este sería para crear las tablas en la base de datos y un último archivo **nombremodulo.info** para ofrecer información acerca del módulo.

El sistema de módulos de Drupal se basa en el concepto de "hooks". Para ampliar Drupal, un módulo necesita simplemente implementar un hook. Cuando Drupal desea permitir la intervención de los módulos, determina los módulos que implementan un hook y llama a ese hook en todos los módulos habilitados que lo implementan. (63)

### 2.4.4 Creación de Temas

A fin de conseguir una mayor separación entre contenido, control y presentación, Drupal no incorpora directamente la gestión de temas dentro del núcleo del mismo, sino que delega esta gestión en módulos externos a los que accede a través de llamadas a funciones que éstos están obligados a implementar. Algunos de estos módulos a su vez son muy genéricos, y en vez de ofrecer un único tema, lo que ofrecen es funcionalidad para trabajar con ellos con plantillas. Este tipo de módulos reciben el nombre de *engine theme*, que traducido literalmente sería algo así como "motor de temas".

En la práctica, lo que hacen estos *engines* es extraer información de Drupal y dejarla en variables accesibles desde las plantillas. En las plantillas los valores de esas variables se "decoran" añadiéndoles las etiquetas HTML de las clases e identificadores definidos en ficheros CSS.

Por defecto, Drupal 5.7 viene con el motor de temas phptemplate, que se encarga de crear el esqueleto XHTML/HTML de la página.

PHPTemplate utiliza plantillas para definir la estructura global de las páginas y de ciertos tipos de contenido en particular.

Las plantillas son ficheros de texto con extensión `.tpl.php` que definen los elementos que deben aparecer en las páginas y su ubicación dentro de las mismas. Su "ubicación" dentro de las páginas entendidas como documentos de texto, como se verían si se abrieran con un editor de texto en vez de con un navegador. La ubicación "visual" de cada elemento, amén de su apariencia, se determina con ficheros `.css`.

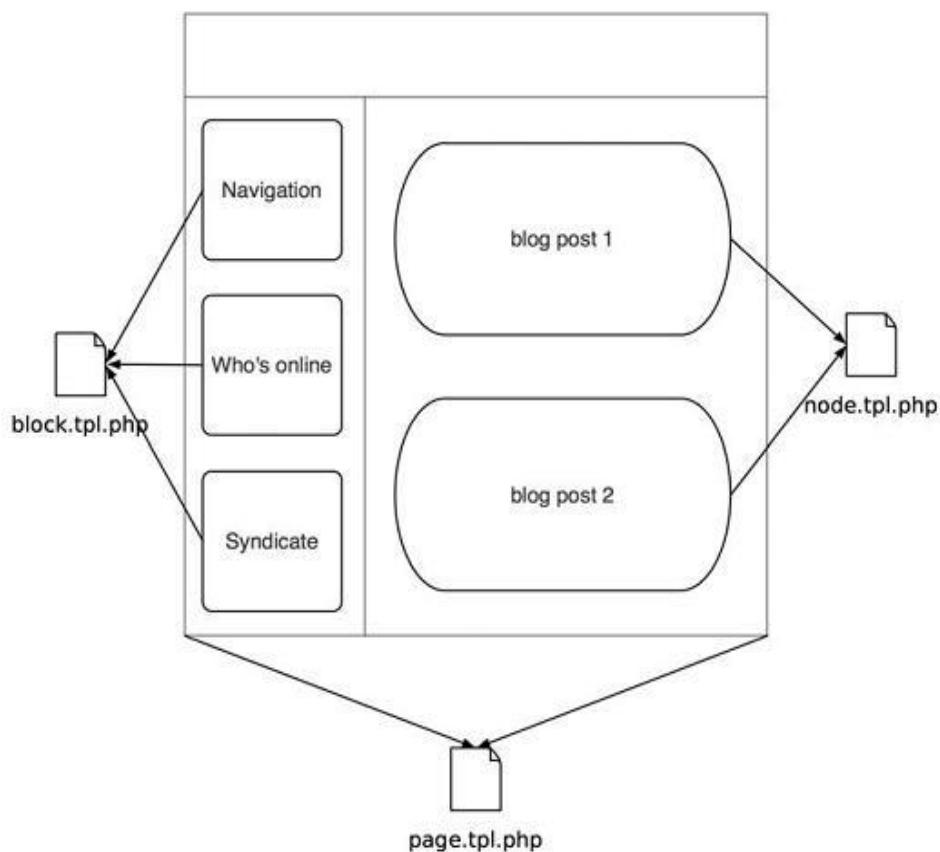


Figura 2.3 Otras plantillas se insertan dentro de la que abarca el archivo `page.tpl.php`

Los nombres de las plantillas son fijos y no pueden cambiarse. Los únicos nombres válidos son

page.tpl.php, node.tpl.php, block.tpl.php, comment.tpl.php y box.tpl.php, que se utilizan para definir las estructuras de las páginas, nodos, bloques, comentarios y contenedores, respectivamente. El único fichero obligatorio es el primero y cuando una plantilla concreta no existe, el *engine* toma una por defecto. Véase la Figura 2.3. (61)

**block.tpl.php** presenta el contenido para los bloques derecho y/o izquierdo de la página. Esta plantilla es opcional.

**box.tpl.php** imprime una caja simple del HTML alrededor de un elemento de la página.

**comment.tpl.php** define el HTML para un bloque del comentario.

**node.tpl.php** esta plantilla controla la exhibición de un nodo, y un resumen del nodo.

**page.tpl.php** esta plantilla define el esqueleto principal para la página, está compuesta por variables que ofrecen diversas facilidades y funcionalidades.

### 2.4.5 Código de Drupal

El núcleo de Drupal está meticulosamente escrito en PHP estilo procedural. El código sigue convenciones estrictas, todo archivo y función son documentados en el código fuente. Las APIs (del inglés **A**pplication **P**rogramming **I**nterface) son a menudo minimalistas, se mantienen breves y funcionales. Estos factores hacen el código fuente de Drupal más fácil de leer en muchos aspectos. Sin embargo, el minimalismo del código puede ser muy engañoso; simples herramientas y módulos se combinan para producir la sorprendente complejidad de este robusto sistema de gestión de contenidos. (65)

¿Por qué Drupal no es orientado a objetos?

En primer lugar el soporte de PHP para construcciones orientadas a objetos era mucho menos maduro en el momento de diseño de Drupal. Drupal fue construido en PHP 4, y la mayoría de las mejoras introducidas en PHP 5 se refieren a sus características orientadas a objetos. El proyecto Drupal se lanzó en el año 2000 y PHP 5 en Julio del 2004. (66)

El código Drupal está separado en *módulos*, cada uno de los cuales define sus propias funciones, que además manejan la inclusión de archivos. La idea es que se cargue la menor cantidad de código por petición para que resulte lo más eficiente posible. En Drupal las funciones son por lo tanto definidas dentro de otras funciones en tiempo de ejecución. PHP no permite este tipo de anidamiento con la declaración de clases, eso significa que la inclusión de ficheros que definan clases debe estar en el nivel más alto, y no

dentro de ninguna función, lo que lleva a código más lento (siempre incluyendo los archivos de definición de clases) o una gran cantidad de lógica en el archivo index.php principal. (66)

El CMS Drupal a pesar de que en su estructura no implementa clases, utiliza muchos términos de la Programación Orientada a Objetos (POO). Existen muchos componentes de Drupal que pueden ser catalogados como objetos, dadas sus funcionalidades específicas. Los más prominentes a ser considerados como objetos son: los módulos, los temas, los nodos y los usuarios. (66)

Los nodos son los bloques básicos para la construcción del contenido. Los métodos para operar estos objetos son definidos en node.module, usualmente llamado por la función node\_invoke().

Los objetos usuario reúnen la información sobre cada cuenta de usuario, información sobre el perfil y rastreo de sesión.

Módulos y temas muchas veces son utilizados para ocupar el rol de “controlador”. Cada módulo es un archivo fuente, pero también relaciona funciones comunes entre sí y sigue un patrón de definición de los hooks de Drupal.

Ahora como ejemplo de los paradigmas de orientación a objetos utilizados por Drupal, se tienen los siguientes:

### **Abstracción**

El Sistema de hooks de Drupal es la base de la abstracción de la interfaz. Un hook define las operaciones que pueden ser ejecutadas por un módulo. Si un módulo implementa un hook, entra en una forma específica para realizar una tarea en particular cuando el hook es invocado. El código de llamado no necesita saber nada más sobre el módulo o sobre la manera en que está implementado el hook para lograr el éxito del trabajo cuando el hook es invocado.

### **Encapsulamiento**

Como muchos otros sistemas OO (orientado a objetos), Drupal no tiene forma de limitar estrictamente el acceso a las funcionalidades internas de un objeto, pero más bien se basa en una convención para lograr esto. El código de Drupal es basado en torno a funciones que comparten un único espacio de nombres, este espacio de nombres se subdivide por el uso de prefijos. Siguiendo esta simple convención, cada módulo puede declarar sus propias funciones y variables sin preocuparse por conflictos con otros.

Esta convención delimita el API público de una clase, de su aplicación interna. Las funciones interiores se prefijan por un `_` (underscore) para indicar que no pueden ser invocadas por módulos externos. Por ejemplo la función `_user_categories()` es una función privada, mientras que `user_save()` es parte de la interfaz pública del objeto usuario y se puede llamar con la expectativa de que el objeto usuario se guardará en la base de datos (aunque el método de hacerlo es privado).

### **Polimorfismo**

Los nodos son polimórficos en el sentido clásico. Si un módulo tiene que mostrar un nodo, por ejemplo, puede llamar `node_view()` en ese nodo para obtener una representación HTML. Sin embargo, la representación real, dependerá de qué tipo de nodo se pasa a la función.

Además el nodo devuelto en este ejemplo puede ser alterado por el tema activo. Los temas son polimórficos de la misma manera, al tema se le pasa el mensaje “procesa este nodo” y responde de una manera diferente dependiendo de la implementación del tema activo, aunque la interfaz es constante.

### **Herencia**

Los módulos y temas pueden definir cualquier función que se quiera. Sin embargo, ambos se pensaron para heredar su comportamiento de una clase base abstracta. En el caso de los temas, el comportamiento de esta clase está determinado por las funciones en `theme.inc`; si un tema no sobrescribe una función definida allí, se utiliza la representación predeterminada de un componente de interfaz, pero el tema en su lugar puede prever su propia representación. Los módulos también tienen la posibilidad de sobrescribir todos los hooks de Drupal y pueden escoger cualquiera e implementarlo.

Drupal cuenta con un estándar de codificación (62), basado en “El estándar de codificación de PEAR (PHP Extension and Application Repository)” (67).

## **2.4.6 Patrones de diseño utilizados en Drupal**

### **Singleton (instancia única)**

La esencia del patrón consiste en garantizar que una clase solo tenga una instancia y proporcionar un punto de acceso global a ella. Pensando en los módulos y temas de Drupal como objetos, entonces estos siguen el patrón Singleton. Lo que diferencia un módulo Drupal de otro es el conjunto de funciones que

este contiene, por lo que debe ser pensado como una clase con una única instancia.

### **Decorator**

La esencia de este patrón responde a la necesidad de añadir dinámicamente funcionalidad a un objeto. Esto permite no tener que crear sucesivas clases que hereden de la primera incorporando la nueva funcionalidad, sino otras que la implementan y se asocian a la primera.

Drupal hace un uso extensivo de este patrón. El polimorfismo con el objeto nodo es un ejemplo claro, pero esto es solo una parte de la potencialidad del sistema de nodos de Drupal. Más interesante es el uso de `hook_nodeapi()` que permite a cualquier módulo extender el comportamiento de todos los nodos. Esta característica posibilita una amplia variedad de comportamientos que se añadirán a los nodos sin necesidad de crear subclases.

### **Observer**

El patrón Observador también conocido como "spider" define una dependencia del tipo uno-a-muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, el observador se encarga de notificar este cambio a todos los otros dependientes.

El patrón Observer también es muy utilizado en Drupal. Cuando es realizada una modificación a un vocabulario del sistema de taxonomía de Drupal, el hook `taxonomy` es llamado en todos los módulos que lo implementan. Al implementar el hook estos se han definido como *observadores* del objeto vocabulario; cualquier cambio que se realice a este, es notificado y actualizado en los dependientes.

## **2.4.7 Editores de texto**

El CMS Drupal cuenta con módulos para la integración de potentes editores de texto como son:

- ✓ FCKeditor: Editor HTML / WYSIWYG de código abierto (Open Source) que provee a la Web del poder de las aplicaciones de escritorio al estilo de editores como el Microsoft Word. Sin la necesidad de instalar ningún componente en la computadora del cliente.
- ✓ TinyMCE: Editor WYSIWYG para HTML de código abierto que funciona completamente en JavaScript y se distribuye gratuitamente bajo licencia LGPL.

Los mismos se pueden descargar desde el portal del proyecto:

<http://drupal.org/project/fckeditor>



## **2.5 Prototipo Ejecutable de Arquitectura**

Una arquitectura ejecutable es una implementación parcial del sistema, construida para demostrar algunas funciones y propiedades.

### **2.5.1 Sitio Oficial del Ajedrez Cubano**

Como uno de los principales resultados del presente trabajo, el proyecto “Infodrez” publicó recientemente de conjunto con el ISLA el nuevo Sitio Oficial del Ajedrez Cubano: [www.capablanca.co.cu](http://www.capablanca.co.cu), el cual constituye un prototipo ejecutable de la arquitectura propuesta.

Desarrollado en el CMS Drupal y con plataforma tecnológica LAMP; donde se publican noticias tanto nacionales (en mayor medida) como internacionales. Cuenta con servicio de Juego Online accesible desde cualquier parte del mundo, lo que ha hecho posible la interacción entre escuelas, centros de educación superior y Joven Clubs.

Lo que permitió demostrar lo antes posible que el método de solución (diseño, tecnología, tiempo y costes estimados) puede resolver satisfactoriamente la definición del producto (objetivos, alcance, rendimiento, beneficios, calidad).

## **2.6 Componentes imprescindibles dentro de la arquitectura propuesta**

En esta sección se presenta una propuesta de solución a los componentes imprescindibles dentro de la arquitectura planteada, con vistas a lograr un eficiente manejo de los contenidos relacionados con el ajedrez.

### **2.6.1 Gestor de Cuadros Sinópticos**

Dadas las facilidades que provee FCKeditor como editor de textos fue seleccionado para la gestión de cuadros sinópticos en el sistema propuesto.

#### **Características del editor de texto FCKeditor:**

- ✓ Generación de código XHTML 1.0
- ✓ Soporte CSS

- ✓ Incorporar formularios
- ✓ Formateo de Fuente
- ✓ Cortar, copiar, pegar
- ✓ Inserción de imágenes
- ✓ Creación de tablas
- ✓ Menús contextuales con botón derecho
- ✓ Entre otras características

**Soporte:**

**Navegadores**

- ✓ Internet Explorer
- ✓ Firefox
- ✓ Safari
- ✓ Ópera
- ✓ Netscape
- ✓ Camino

**Sistemas Operativos**

- ✓ Windows
- ✓ MacOS
- ✓ Linux

**Lenguajes de servidor**

- ✓ PHP
- ✓ Perl
- ✓ Python
- ✓ ASP.NET
- ✓ ASP
- ✓ ColdFusion
- ✓ Java
- ✓ Active-FoxPro
- ✓ Lasso

## 2.6.2 Visor de Partidas

Iniciadores de partidas (visores de ajedrez): son utilizados para almacenar y comenzar partidas. Parte de ellos están limitados a leer partidas en formato PGN, es decir son lectores PGN. (68)

Formato PGN: Notación Portátil de Juego (Del original en inglés: Portable Game Notation (.PGN)) es un formato de computadora para grabar partidas de ajedrez, tanto los movimientos como la información relacionada; la mayoría de los programas de ajedrez para computadora reconocen este formato que es muy popular como consecuencia de su fácil uso.

Se propone la implementación del mismo como un nuevo módulo para Drupal. Para ello se podría tomar como base la librería ltpgnviewer (69) utilizada en el desarrollo de un visor para el módulo de “Juego Online” perteneciente al proyecto “Infodrez”.

## 2.6.3 Gestor de Diagramas

Para la inserción de diagramas de ajedrez, donde se representen situaciones dadas en las partidas se propone la instalación del módulo de Drupal: Chessboard Renderer (el mismo puede ser descargado desde el sitio oficial del proyecto: drupal.org)

## 2.7 Vistas de la Arquitectura propuesta

Empleando las diferentes vistas para la arquitectura según RUP, se hará énfasis en el módulo por implementar: Visor de partidas (funcionalidad crítica dentro del sistema).

### 2.7.1 Vista de Casos de Uso

Vista arquitectónica del modelo de casos de uso: Vista de la arquitectura de un sistema abarcando los casos de uso significativos desde un punto de vista arquitectónico. (6)

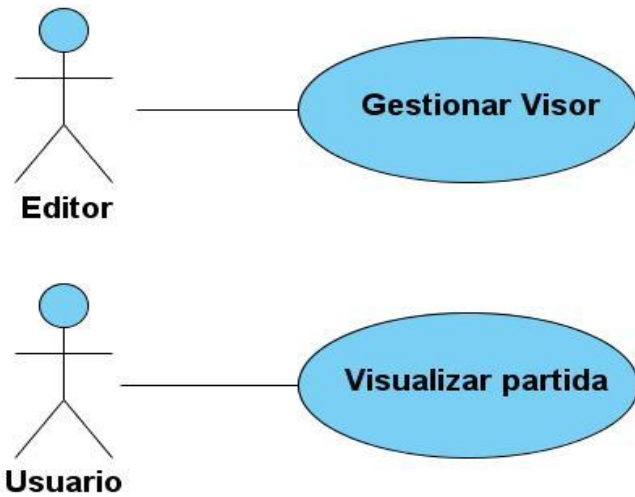


Figura 2.5 Casos de Uso Módulo Visor de Partidas.

### 2.7.2 Vista Lógica

Vista arquitectónica del modelo de diseño: Vista de la arquitectura de un sistema, abarcando las clases, subsistemas, interfaces y realizaciones de casos de uso del diseño que forman el vocabulario del dominio de la solución del sistema; vista que abarca también los hilos y procesos que establecen la concurrencia y mecanismos de sincronización del sistema; vista que aborda los requisitos no funcionales, incluyendo los requisitos de rendimiento y capacidad de crecimiento de un sistema. (6)

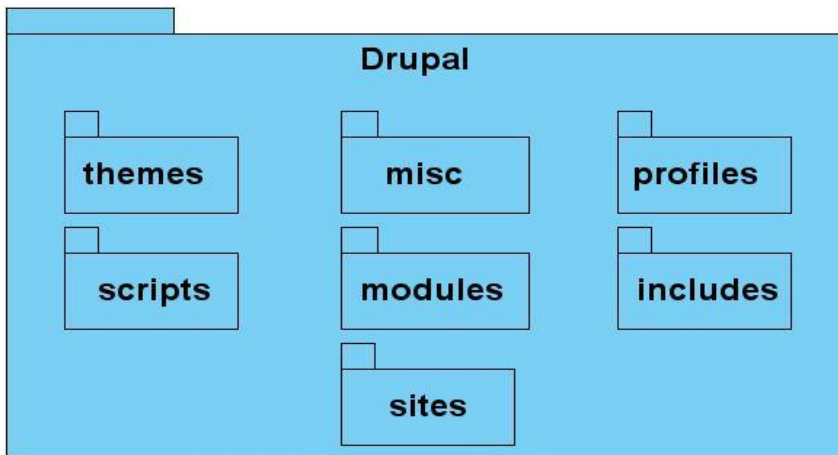


Figura 2.6 Elementos del diseño

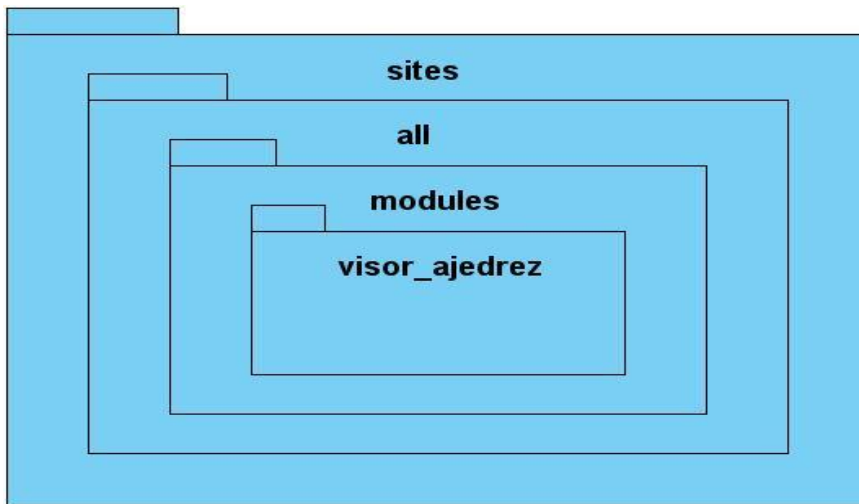


Figura 2.7 Estructura interna del paquete **sites**

### 2.7.3 Vista de Implementación

Vista arquitectónica del modelo de implementación: Vista de la arquitectura de un sistema, abarcando los componentes usados para el ensamblado y lanzamiento del sistema físico; vista que aborda la gestión de la configuración de las versiones del sistema, constituida por componentes independientes que pueden ser ensamblados de varias formas para producir un sistema ejecutable.

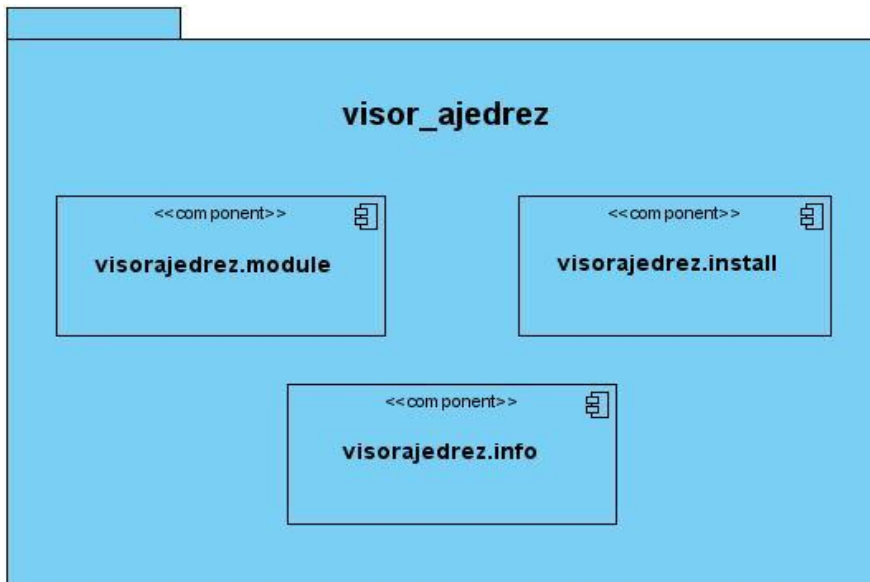


Figura 2.8 Diagrama de componentes del paquete **visor\_ajedrez**

## 2.7.4 Vista de Despliegue

Vista arquitectónica del modelo de despliegue: Vista de la arquitectura de un sistema abarcando los nodos que forman la topología hardware sobre la que se ejecuta el sistema; vista que aborda la distribución, entrega e instalación de las partes que constituyen el sistema físico. (6)

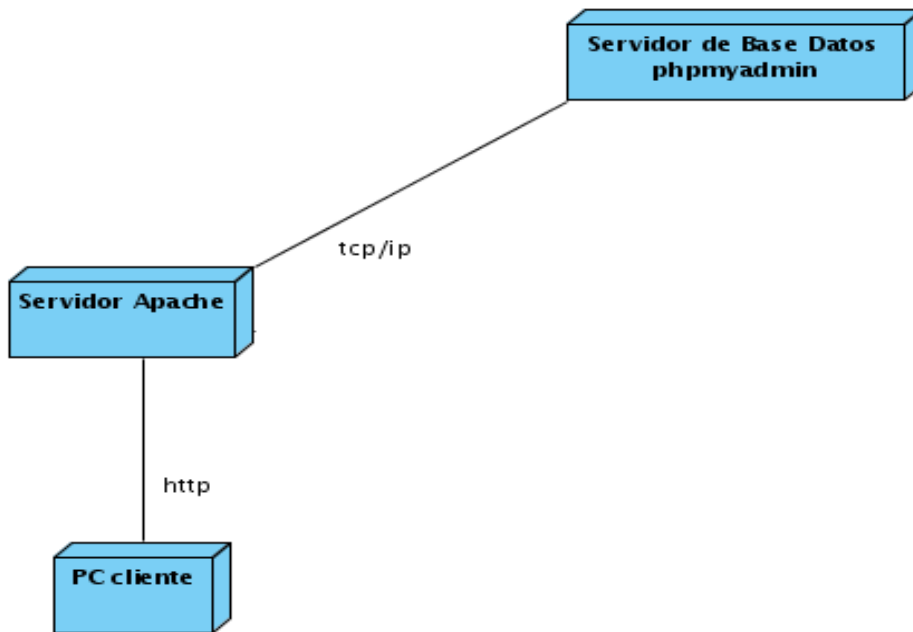


Figura 2.9 Diagrama de Despliegue.

### Descripción de Nodos

#### Nodo PC Cliente

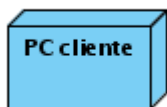


Figura 2.10 PC cliente

512 MB de Ram, Procesador de 3GHZ

### Nodo Servidor Apache

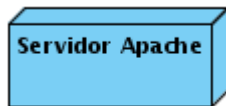


Figura 2.11 Servidor Apache

- ✓ Donde estará el Servidor Web

### Nodo Servidor de base de datos

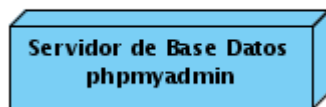


Figura 2.12 Servidor de base de datos

- ✓ Un servidor donde estará el SGBD MySQL

## **2.8 Conclusiones**

En el presente capítulo se definió el concepto de framework, que guarda estrecha relación con la arquitectura propuesta; donde fue elegido el CMS Drupal como framework de desarrollo para la integración de los componentes relativos al ajedrez especificados en el Capítulo 1. Se identificaron patrones arquitectónicos y de diseño utilizados por Drupal. Y se detalló una propuesta de solución para los diferentes componentes, donde se presentaron transparencias a través de las vistas de la AS.

# Capítulo 3 Descripción de la Arquitectura

## 3.1 Introducción

Este capítulo ofrece una visión general de la arquitectura del sistema, usando para esto diferentes vistas arquitectónicas para describir los aspectos más significativos del mismo. Se desea transmitir las decisiones más importantes que afectan o crean la arquitectura del sistema, además que sirva como referencia y le ofrezca un mayor entendimiento del mismo a las partes interesadas.

Para la realización de este capítulo se toma como guía la plantilla de Documento de Arquitectura de Software vigente en la Universidad de las Ciencias Informáticas.

### Propósito

Los propósitos que se persiguen con dicho documento son:

- ✓ Mejor comprensión del sistema.
- ✓ Organización del desarrollo.
- ✓ Fomento de la reutilización.
- ✓ Hacer evolucionar el sistema.

Debe capacitar a los desarrolladores, directivos, clientes, otros usuarios para comprender con suficiente detalle y para facilitar su propia participación.

### Alcance

Este documento influye en la toma de decisiones arquitectónicas con respecto a la línea de desarrollo de portales web orientados al ajedrez dentro del proyecto “Infodrez”.

## 3.2 Representación arquitectónica

Este documento presenta la arquitectura como una serie de vistas:

- ✓ Vista de Caso de Uso
- ✓ Vista Lógica
- ✓ Vista de Implementación
- ✓ Vista de Despliegue

Estas vistas están representadas en UML usando Visual Paradigm for UML 6.0 Enterprise Edition.



### 3.3 Objetivos y Restricciones Arquitectónicas

En esta sección se incluye una descripción de los requerimientos y objetivos del software que tienen un impacto significativo en la arquitectura.

#### 3.3.1 Apariencia o Interfaz Externa

- ✓ La interfaz a implementar debe ser sencilla para disminuir el tiempo de capacitación de los usuarios finales (principalmente aquellas personas que no son expertas en la rama de la informática).
- ✓ Por el uso diario y constante que tendrá el software, la interfaz debe ser agradable, que favorezca el estado de ánimo del cliente y que combine correctamente los colores, tipo de letra y tamaño y que los iconos estén en correspondencia con lo que representan.

#### 3.3.2 Usabilidad

- ✓ El sistema debe ser de fácil manejo para los usuarios que tengan niveles básicos sobre la computación o hallan trabajado con la Web y contar con una ayuda con instrucciones de tipo paso a paso, para entender el trabajo del sistema, así como un listado de definiciones para términos y acrónimos del mismo.

#### 3.3.3 Rendimiento

- ✓ La aplicación debe estar concebida para el consumo mínimo de recursos.
- ✓ Un total de 400 usuarios conectados de forma simultánea al servidor central en cualquier momento de tiempo dado.
- ✓ Debe completar las transacciones en un tiempo de 60 segundos.
- ✓ La latencia del sistema no debe ser mayor de 15 segundos.
- ✓ Los clientes no necesitarán más de 128MB de RAM, lo suficiente para ejecutar un navegador web.

#### 3.3.4 Soporte

##### Para el servidor de aplicaciones:

Se requiere que esté instalado PHP en su versión 4.3.5 o superior.

Servidor web Apache en su versión 2.x.

**Para el servidor de base de datos:**

Se requiere que esté instalado el gestor de base de datos MySQL 4.1 o superior; o en su defecto PostgreSQL 7.4 o superior.

**Para el cliente:**

Se requiere que esté instalado uno de los siguientes navegadores web: Internet Explorer 5.5 o superior, Mozilla Firefox 2.0.0.1 o superior, Opera 9 o superior, Safari 2 o superior.

Se requiere que los navegadores tengan habilitado el Javascript.

### **3.3.5 Portabilidad, Escalabilidad, Reusabilidad**

- ✓ El sistema será multiplataforma.
- ✓ La aplicación se construirá utilizando patrones (de diseño como los GRASP) y estándares internacionales de implementación, documentación y diseño (W3C, normas ISO), para facilitar su integración futura, con componentes desarrollados por cualquiera de las partes y garantizar posibilidades de mantenimiento ágil y seguro.
- ✓ Debido a los cambios en las condiciones económicas del país, las empresas cubanas toman decisiones continuas que cambian las condiciones en que se desarrollan los procesos, por lo que el sistema deberá implementar la forma de adaptarse ante el cambio de dichas condiciones.

### **3.3.6 Hardware**

Para las estaciones de trabajo:

- ✓ Se requiere tengan tarjeta de red.
- ✓ Se requiere tengan al menos 128 MB de memoria RAM.
- ✓ Se requiere al menos 100 MB de disco duro.
- ✓ Procesador 800 MHz como mínimo.

Para los servidores:

- ✓ Se requiere tarjeta de red.
- ✓ Se requiere tenga al menos 512MB de RAM.

- ✓ Se requiere al menos 40GB de disco duro.
- ✓ Procesador 2.0 GHz como mínimo.

### 3.3.7 Software

- ✓ En las computadoras de los clientes se garantizará versiones de Windows 2000 o superior, así como Linux y sus correspondientes distribuciones.
- ✓ En las computadoras de los clientes solo se requiere de un navegador (Internet Explorer versión 4.5 o superior, Mozilla Firefox versión 2.0.0.1 o superior, Opera 9 o superior, Safari 2.0 o superior).

### 3.3.8 Seguridad

- ✓ El sistema debe poder comunicarse usando un protocolo seguro (HTTPS).
- ✓ Los datos que no pueden viajar de forma transparente por la red, deben ser encriptados.
- ✓ Chequear si el usuario que está accediendo al sistema está autenticado y brindarle servicio de autenticación.
- ✓ Permitir que cuando se borre cualquier documento o información pueda existir una opción de advertencia antes de realizar la acción.
- ✓ Realizar auditoria a los principales eventos dentro del sistema, registrando al usuario, el tipo de usuario y los eventos efectuados.
- ✓ Un porcentaje de la seguridad corre por parte del lenguaje y Framework propuesto (PHP y Drupal respectivamente) y otra parte por los servidores (Apache, LDAP, PostgreSQL o MySQL).
- ✓ La asignación de roles a los usuarios y sus funcionalidades sobre el sistema se definirán desde el módulo de Administración.
- ✓ Se utilizará reglas o principios de la “programación segura” (diseño simple y abierto, separación de privilegios, control de acceso apropiado, validación de datos de entrada y salida, tratamiento de errores, utilización de criptografía, reutilización de código, control del flujo de datos, control de sobrecarga del búfer, control de inyección de código).
- ✓ Debe quedar constancia de quién, desde donde, y cuando se realizó una operación determinada en el sistema.

### 3.3.9 Confiabilidad, Integridad, Fiabilidad

- ✓ La información manejada por el sistema está protegida de acceso no autorizado y divulgación.
- ✓ La información manejada por el sistema será objeto de cuidadosa protección contra la corrupción de los datos y accesos indebidos.
- ✓ Debe garantizarse el resguardo de la información (imágenes, documentos), así como la grabación periódica (backups) de la Base de Datos, de forma tal que se posibilite la reinstalación del sistema y los datos, en caso de fallos en el sistema o en el hardware.

### 3.3.10 Legales

La mayoría de las herramientas de desarrollo son libres.

### 3.3.11 Redes

Para hacer más fiable la aplicación debe de estar protegida contra fallos de corriente y de conectividad, para lo que se deberán parametrizar los tiempos para realizar copias de seguridad.

## 3.4 Vistas

Las vistas fueron especificadas en la sección 2.7.

## 3.5 Calidad

Los aspectos concernientes a la calidad serán tratados en el Capítulo 4.

## 3.6 Conclusiones

En este Capítulo se apunta hacia las diferentes Vistas Arquitectónicas, permitiendo describir la arquitectura; se puede concluir, que las mismas son un subconjunto de modelos de diseño de software, incluyendo los elementos significativos de la arquitectura y excluyendo el diseño de los *componentes* básicos; resuelve a la hora de tomar decisiones sobre qué desarrollar y qué reutilizar, facilitándole un mejor desempeño y entendimiento al resto de los desarrolladores que tienen como tarea darle terminación al producto.

# Capítulo 4 Evaluación de la Arquitectura

## Resumen

En el presente capítulo se realiza un pequeño estudio del arte de la Evaluación de Arquitecturas de Software con el objetivo de brindar un panorama general sobre el tema. En el mismo se abordan temas como: ¿Qué es una evaluación? ¿Cuáles son los objetivos de evaluar una arquitectura? ¿Por qué evaluar una Arquitectura? ¿Cuándo realizar una evaluación a una Arquitectura? ¿Cuáles son sus características? ¿Quiénes participan en una evaluación de una Arquitectura? Así como técnicas empleadas a la hora de realizarle una evaluación a una Arquitectura, un breve análisis de algunos de los métodos de evaluación existentes y una propuesta de evaluación del LISI (Laboratorio de Investigación en Sistemas de Información), Universidad Simón Bolívar.

## Palabras claves

Calidad de Software: La Calidad de Software es *“la concordancia con los requisitos funcionales y de rendimiento establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado de forma profesional”*. (70)

Componente: *“Es una parte de una arquitectura de software claramente identificable, independiente a la aplicación en la que se utiliza y de otros componentes (partes), que describe y realiza funciones específicas y claras dentro del contexto de la arquitectura, puede ser modificado durante el diseño, posee una documentación clara que permite conocer sus características, atributos y comportamiento, reutilizable y su interoperabilidad con otros componentes (partes) no reduce el nivel de eficiencia de la arquitectura”* (70).

Escenario: Un escenario consta de tres partes: el estímulo, el contexto y la respuesta. El estímulo es la parte del escenario que explica o describe lo que el involucrado en el desarrollo hace para iniciar la interacción con el sistema. Puede incluir ejecución de tareas, cambios en el sistema, ejecución de pruebas, configuración, etc. El contexto describe qué sucede en el sistema al momento del estímulo. La respuesta describe, a través de la arquitectura, cómo debería responder el sistema ante el estímulo. Este último elemento es el que permite establecer cuál es el atributo de calidad asociado (71).

“Los escenarios describen una utilización anticipada o deseada del sistema, y típicamente se expresan en una frase” (72). “Los mismos representan una abstracción de los requerimientos más importantes de un sistema” (72).

#### 4.1 Introducción

La Arquitectura de Software (AS) de los Sistemas de Software (SS) a ser construidos, se convierte en un factor de importancia para lograr que estos tengan un alto nivel de calidad. Recuérdese que el poseer una buena AS es de suma importancia, ya que ésta es el corazón de todo SS y determina cuáles serán los niveles de calidad asociados al sistema (70).

No sirve de nada un sistema que no cumple con los atributos de calidad que se especificaron en los requerimientos no funcionales de los clientes. Por lo que diseñar una correcta arquitectura va a determinar el éxito o fracaso de un sistema de software, en la medida que esta cumpla o no con sus objetivos. Debido a esto para reducir tales riesgos, y como buena práctica de ingeniería, es recomendable realizar evaluaciones a la arquitectura. (71)

En este capítulo se explica el procedimiento para realizar Pruebas de Concepto a una Arquitectura de Software (Evaluar una Arquitectura), enfatizando en el por qué se hace necesaria, los beneficios que brinda, así como destacar algunos métodos de evaluación existentes y cuál de ellos se propone. En este caso, el **MECABIC (Método de Evaluación de la Calidad de Arquitecturas Basadas en la Integración de Componentes)**, cuyo objetivo principal es evaluar y analizar la calidad de este tipo de AS. El método es propuesto por Aleksander González, Marizé Mijares, Luis E. Mendoza, Anna Grimán, María Pérez, LISI (Laboratorio de Investigación en Sistemas de Información), Universidad Simón Bolívar.

El alcance que se persigue en el presente capítulo es dar a conocer un panorama general sobre evaluación de arquitecturas, proponer que el método elegido se aplique a la arquitectura de los módulos del proyecto “Infodrez” y aplicar el mismo a la presente propuesta de arquitectura.

#### 4.2 Estudio del estado del arte

##### Principales problemas en un proyecto causados por la Arquitectura

- ✓ Este falla en cumplir con la calidad necesaria.
- ✓ Este falla en intentar soportar las necesidades de negocio.

- Falta de compromiso del usuario para que el proyecto termine (problemas en la comunicación).
- ✓ El 50% de proyectos que se atrasan tienen problemas en la arquitectura.
- ✓ El 35% de los proyectos que exceden el costo de construcción tienen problemas en la arquitectura.

### ¿Qué es una Evaluación?

- ✓ Es un estudio de factibilidad que pretende detectar posibles riesgos, como así también buscar recomendaciones para contenerlos.
- ✓ La diferencia entre evaluar y verificar es que la evaluación se realiza antes de la implementación de la solución. La verificación es con el producto ya construido.

### Objetivos de Evaluar una Arquitectura

El **objetivo** de evaluar una arquitectura es saber si puede habilitar los requerimientos, atributos de calidad y restricciones para asegurar que el sistema a ser construido cumple con las necesidades de los stakeholders. (71)

La evaluación de una arquitectura de software es una tarea no trivial, puesto que se pretende medir propiedades del sistema en base a especificaciones abstractas, como por ejemplo los diseños arquitectónicos. Por ello, la intención es más bien **la evaluación del potencial de la arquitectura diseñada para alcanzar los atributos de calidad requeridos**. Las mediciones que se realizan sobre una arquitectura de software pueden tener distintos objetivos, dependiendo de la situación en la que se encuentre el arquitecto y la aplicabilidad de las técnicas que emplea. Algunos de estos objetivos son: *cualitativos, cuantitativos y máximos y mínimos teóricos*.

- ✓ La **medición cualitativa** se aplica para la comparación entre arquitecturas candidatas y tiene relación con la intención de saber la opción que se adapta mejor a cierto atributo de calidad. Este tipo de medición brinda respuestas afirmativas o negativas, sin mayor nivel de detalle.
- ✓ La **medición cuantitativa** busca la obtención de valores que permitan tomar decisiones en cuanto a los atributos de calidad de una arquitectura de software. El esquema general es la comparación con márgenes establecidos, como lo es el caso de los requerimientos de desempeño, para establecer el grado de cumplimiento de una arquitectura candidata, o tomar decisiones sobre ella. Este enfoque

permite establecer comparaciones, pero se ve limitado en tanto no se conozcan los valores teóricos máximos y mínimos de las mediciones con las que se realiza la comparación.

- ✓ La **medición de máximo y mínimo teórico** contempla los valores teóricos para efectos de la comparación de la medición con los atributos de calidad especificados. El conocimiento de los valores máximos o mínimos permite el establecimiento claro del grado de cumplimiento de los atributos de calidad.

### **Características de una Evaluación de Arquitectura**

- ✓ Es uno de los principales puntos de evaluación dentro del proyecto, ya que errores en ella, pueden traer que el proyecto fracase.
- ✓ Puede ser realizada por gente Interna o Externa al proyecto, aunque lo más interesante es que sea realizada por gente Externa (Mentores o Arquitectos del Área).

### **¿Por qué evaluar una Arquitectura?**

El propósito de realizar evaluaciones a la arquitectura, es para analizar e identificar riesgos potenciales en su estructura y sus propiedades, que puedan afectar al sistema de software resultante, verificar que los requerimientos no funcionales estén presentes en la arquitectura, así como determinar en qué grado se satisfacen los atributos de calidad. Cabe señalar que los requerimientos no funcionales también son llamados atributos de calidad. (71)

Un atributo de calidad es una característica de calidad que afecta a un elemento. Donde el término “característica” se refiere a aspectos no funcionales y el termino “elemento” a componente. (71)

### **¿Cuándo una Arquitectura puede ser evaluada?**

Es posible realizarla en cualquier momento según Kazman, pero propone dos variantes que agrupan dos etapas distintas: temprano y tarde (71).

- ✓ Temprana. No es necesario que la arquitectura esté completamente especificada para efectuar la evaluación, y esto abarca desde las fases tempranas de diseño y a lo largo del desarrollo.
- ✓ Tarde. Cuando ésta se encuentra establecida y la implementación se ha completado. Este es el caso general que se presenta al momento de la adquisición de un sistema ya desarrollado.



### ¿Quiénes participan en una Evaluación?

Generalmente las evaluaciones a la arquitectura se hacen por miembros del equipo de desarrollo, arquitecto, diseñador, entre otros. Sin embargo puede haber también situaciones en las que intervengan personas especialistas en el tema. Otro que también se interesa por los resultados de una evaluación es el cliente, ya que en dependencia de los resultados puede tomar decisiones de continuar o no con el proyecto (71).

### Técnicas de Evaluación

Existen un grupo de técnicas para evaluar que se clasifican en cualitativas y cuantitativas (71):

- ✓ Técnicas de cuestionamiento o cualitativas. Utilizan preguntas cualitativas para preguntarle a la arquitectura.
  - Cuestionarios. Abiertas. Temprana.
  - Checklists. Especifico del Dominio de la aplicación.
  - Escenarios. Especificas del Sistema. Arquitectura avanzada.
- ✓ Measuring techniques. Sugiere hacerle medidas cuantitativas a la arquitectura.
  - Utiliza métricas arquitectónicas, como acoplamiento, cohesividad en los módulos, profundidad en herencias, modificabilidad.
  - Simulaciones, Prototipos, y Experimentos.

Por lo regular, las técnicas de evaluación cualitativas son utilizadas cuando la arquitectura está en construcción, mientras que las técnicas de evaluación cuantitativas, se usan cuando la arquitectura ya ha sido implantada (71):

### ¿Por qué cualidades puede ser evaluada una Arquitectura?

A grandes rasgos, Bass establece una clasificación de los atributos de calidad en dos categorías (71):

Observables vía ejecución: aquellos atributos que se determinan del comportamiento del sistema en tiempo de ejecución. (Ejemplos: funcionalidad, desempeño, seguridad)

No observables vía ejecución: aquellos atributos que se establecen durante el desarrollo del sistema. (Ejemplos: portabilidad, reusabilidad, escalabilidad)

### ¿Cuáles son los beneficios de realizar una evaluación arquitectónica? (71)

- ✓ Financieros.
- ✓ Reúne a los stakeholders.
- ✓ Fuerza una articulación en las metas específicas de calidad.
- ✓ Fuerza una mejora a la documentación de la arquitectura.
- ✓ Mejora la arquitectura.
- ✓ Detección temprana de problemas.
- ✓ Validación de requerimientos y su prioridad.
- ✓ Recolecta los fundamentos y las decisiones arquitectónicas tomadas.

### ¿Qué resultados produce la evaluación de una Arquitectura?

Una vez que se ha efectuada la evaluación se debe elaborar un reporte. Que debe presentarse como un documento preliminar, con la finalidad de que se corrija por las personas que participaron en la evaluación. El contenido del reporte responde a dos tipos de preguntas (71):

- ✓ ¿Se ha diseñado la arquitectura más apropiada para el sistema?
- ✓ ¿Cuál de las arquitecturas propuestas es la más apropiada para el sistema a construir?

Además de responder estas preguntas, el reporte también indica el grado en que se cumplieron los atributos de calidad.

### ¿Cuáles son las salidas de una evaluación arquitectónica? (71)

- ✓ Lista priorizada de los atributos de calidad requeridos para la arquitectura que está siendo evaluada.
- ✓ Riesgos y no riesgos.

### Métodos de Evaluación de Arquitecturas

**ATAM** (*Architecture Trade-off Analysis Method*): está inspirado en tres áreas distintas: los estilos arquitectónicos, el análisis de atributos de calidad y el método de evaluación SAAM (*Software Architecture Analysis Method*). El nombre del método ATAM surge del hecho de que revela la forma en que una arquitectura específica satisface ciertos atributos de calidad, y provee una visión de cómo los atributos de

calidad interactúan con otros. El método se concentra en la identificación de los estilos arquitectónicos o enfoques arquitectónicos utilizados. Estos elementos representan los medios empleados por la arquitectura para alcanzar los atributos de calidad, así como también permiten describir la forma en la que el sistema puede crecer, responder a cambios, e integrarse con otros sistemas, entre otros (71). El método de evaluación ATAM comprende nueve pasos, agrupados en cuatro fases (Presentación, Investigación y Análisis, Pruebas y Reporte).

**Bosch (2000).** Que plantea que: “El proceso de evaluación debe ser visto como una actividad iterativa, que forma parte del proceso de diseño, también iterativo. Una vez que la arquitectura es evaluada, pasa a una fase de transformación, asumiendo que no satisface todos los requerimientos. Luego, la arquitectura transformada es evaluada de nuevo” (71). Este método consta de 5 pasos divididos en dos etapas.

**ADR (*Active Design Review*).** “ADR es utilizado para la evaluación de diseños detallados de unidades del software como los componentes o módulos. Las preguntas giran en torno a la calidad y completitud de la documentación y la suficiencia, el ajuste y la conveniencia de los servicios que provee el diseño propuesto” (71).

**ARID (*Active Reviews for Intermediate Design*).** ARID es un método de bajo costo y gran beneficio, el mismo es conveniente para realizar la evaluación de diseños parciales en las etapas tempranas del desarrollo (71). ARID es un híbrido entre ADR y ATAM. Se basa en ensamblar el diseño de los stakeholders para articular los escenarios de usos importantes y probar el diseño para ver si satisface los escenarios. Como resultado de la aplicación de dicho procedimiento se obtiene un diseño de alta fidelidad acompañado de una alta familiarización con el diseño de los stakeholders. El método consta de 9 pasos agrupados en dos fases (Actividades Previas y Evaluación) (71).

**Losavio (2003):** Es un método para evaluar y comparar arquitecturas de software candidatas, que hace uso del modelo de especificación de atributos de calidad adaptado del modelo ISO/IEC 9126. La especificación de los atributos de calidad haciendo uso de un modelo basado en estándares internacionales ofrece una vista amplia y global de los atributos de calidad, tanto a usuarios como arquitectos del sistema, para efectos de la evaluación. El método contempla siete actividades.

Comparación entre métodos de evaluación de AS. Véase la Tabla 4.3.

	<b>ATAM</b>	<b>SAAM</b>	<b>ARID</b>	<b>Bosch(2000)</b>	<b>Losavio(2003)</b>
Atributos de Calidad Contemplados	Modificabilidad Seguridad Confiabilidad Desempeño	Modificabilidad Funcionabilidad	Conveniencia del diseño evaluado	Seleccionados por el arquitecto, de acuerdo a la importancia sobre el sistema	Funcionabilidad Confiabilidad Usabilidad Eficiencia Mantenimiento Portabilidad
Objetos Analizados	Estilos Arquitectónicos, Documentación, Flujo de Datos y Vistas Arquitectónicas	Documentación, y Vistas Arquitectónicas	Especificación de los componentes	Estilos Arquitectónicos, Vistas Arquitectónicas, Patrones Arquitectónicos, Patrones de Diseño y Patrones de Idioma	Especificación de Atributos de Calidad
Etapas del Proyecto en las que se Aplica	Luego que el diseño de la arquitectura ha sido establecido	Luego que la arquitectura cuenta con funcionalidad ubicada en módulos	A lo largo del diseño de la arquitectura	Luego que el diseño de la arquitectura ha sido establecido	Luego que el diseño de la arquitectura ha sido establecido
Enfoques Utilizados	Árbol de Utilidad y lluvia de ideas para	Lluvia de ideas para escenarios y articular los	Revisiones de diseño, lluvia de ideas para	Análisis de perfiles (profiles).	Análisis y comparación de los

articular los requerimientos de calidad. Análisis arquitectónico que detecta puntos sensibles, puntos de balance y riesgos.	requerimientos de calidad. Análisis de los escenarios para verificar funcionalidad o estimar el costo de los cambios.	obtener escenarios.	resultados para las arquitecturas candidatas.
---	--	------------------------	--

Tabla 4.3 Comparación entre métodos de evaluación

### Conclusiones parciales

- ✓ El método ATAM evalúa con más profundidad, en relación con otros métodos, cuestiones referentes a la arquitectura, como son: los atributos de calidad.
- ✓ Hasta el momento se han presentado varios métodos de evaluación de arquitectura algunos más completos que otros, pero independientemente de esto todos tienen algo en común y es que utilizan la técnica de escenarios como vía de constatar en qué medida la arquitectura responde a los atributos de calidad requeridos por el sistema.

### 4.3 Propuesta de Procedimiento

Como resultado del análisis de la comparación de diferentes métodos, se observó que el Architecture Tradeoffs Analysis Method (ATAM), tiene un conjunto de pasos, un equipo de evaluación y un conjunto de salidas mejor definidas y no presenta ninguna restricción con respecto a la característica de calidad a evaluar. El MECABIC está inspirado en ATAM, aunque con el objetivo de facilitar su aplicación sobre Arquitecturas de Software Basadas en Componentes (ASBC) se incluyó en algunos de sus pasos un enfoque de integración de elementos de diseño, inspirado en la construcción de partes arquitectónicas adoptado por el Architecture Based Design (ABD). Además se propone un árbol de utilidad inicial basado en el modelo de calidad ISO-9126 instanciado para Arquitectura de Software propuesto por Losavio. La

adopción de este modelo por parte del MECABIC permite concentrarse en características que dependen exclusivamente de la arquitectura, y al ser un estándar facilita la correspondencia con características de calidad consideradas por los métodos estudiados. Los escenarios incluidos en este árbol son específicos para aplicaciones basadas en componentes. A continuación el método propuesto (70)

### A. Equipo de Evaluación

En el método MECABIC participan tres grupos de trabajo tal como se muestra en la Tabla 4.4.

Equipo	Definición	Fases en las que participan
Arquitectos	Responsables de generar y documentar una AS para el sistema estudiado.	Todas
Evaluador	Integrado por personas expertas en asuntos de calidad quienes guiarán el proceso de evaluación de la arquitectura.	Todas
Relacionados	Son las personas involucradas de alguna manera con el sistema: programadores, usuarios, gerentes, entre otros.	Fases 1, 3 y 4.

Tabla 4.4 Grupos participantes en el método MECABIC (70)

### B. Instrumentos y técnicas de evaluación

Para la especificación de la calidad se hace uso de un árbol de utilidad, el cual tiene como nodo raíz la “bondad” o “utilidad” del sistema. En el segundo nivel del árbol se encuentran los atributos de calidad. Las hojas del árbol de utilidad son escenarios, los cuales representan mecanismos mediante los cuales extensas (y ambiguas) declaraciones de cualidades son hechas específicas y posibles de evaluar. La generación del árbol de calidad incluye un paso que permite establecer prioridades. Para el análisis de la arquitectura se utiliza la técnica de escenarios, soportada por cuestionarios, para identificar lo que desean los participantes.

### C. Fases

Este método consta de 4 fases:

1) En la primera fase, de *presentación*, se da a conocer el método entre todos los grupos, el sistema y su

organización, y finalmente se presenta cuál es la arquitectura que debe ser evaluada.

2) La segunda es de *investigación y análisis*, y en ella se determina de qué manera se va estudiar la arquitectura, se pide a los *stakeholders* que expresen de una manera precisa qué escenarios de calidad se desean y se analiza si la arquitectura es apropiada o se requieren modificaciones para que lo sea. En esta fase sólo participan el grupo evaluador y grupo de arquitectos.

3) La tercera fase es *de prueba*, consiste en la revisión de la segunda fase y en ella participan todos los grupos.

4) En la última fase se lleva a cabo la presentación de los resultados. En esta fase participan todos los grupos.

#### 4.4 Evaluación de la arquitectura propuesta

Para la evaluación de la arquitectura se toman como medidores los atributos de calidad que tributan al cumplimiento del **objetivo general** del presente trabajo. Para la selección de los escenarios a evaluar se toma en consideración el criterio de expertos (especialistas en calidad y arquitectos). Como resultado se obtiene el árbol de utilidad inicial propuesto por el método MECABIC. Véase la Tabla 4.5.

Característica	Escenario	Prioridad
Seguridad	El sistema detecta la actuación de un intruso e impide acceso a los componentes que manejen información sensible	(Bajo, Alto)
Rendimiento	El sistema o componente debe cumplir con sus funciones designadas en el transcurso de un tiempo indicado.	(Bajo, Alto)
Escalabilidad	El sistema debe ser capaz de cambiar su tamaño o configuración para adaptarse a las circunstancias cambiantes.	(Bajo, Alto)
Portabilidad	Los componentes pueden instalarse fácilmente en todos los ambientes donde debe funcionar.	(Bajo, Medio)

Tabla 4.5 Subconjunto del árbol de utilidad inicial propuesto por el método MECABIC.

A continuación se detallan las decisiones tomadas en cuanto a la prioridad (esfuerzo, riesgo) de los atributos de calidad evaluados:

Seguridad: el esfuerzo de satisfacer el escenario es **bajo** dado a que la instalación de Drupal cuenta con un módulo para la gestión de usuarios por roles, donde se administran los derechos de acceso a cada módulo y/o contenido para cada usuario según su rol. Excluir el escenario del árbol de utilidad constituye un **alto** riesgo dado a que cualquier intruso podría manipular información sensible dentro del sistema.

Rendimiento: en el benchmark realizado en la sección 1.15.4 a los principales CMS de código abierto se dedujo que Drupal cuenta con un rendimiento superior al de sus competidores gracias a la implementación de un mecanismo de caché avanzado, por lo que el esfuerzo de satisfacer este escenario es **bajo**. La exclusión del escenario conllevaría un **alto** riesgo ya que atentaría contra la paciencia de los usuarios del portal.

Escalabilidad: el esfuerzo de satisfacer el escenario es **bajo** dado a que Drupal presenta un mecanismo de caché avanzado, y un módulo para el control de la congestión del sistema (Throttle), que permiten disminuir la carga al servidor y aumentar el rendimiento. El riesgo que conlleva excluir el escenario es **alto** dada la necesidad que tiene un portal web de crecer en cuanto a usuarios que lo frecuentan.

Portabilidad: Drupal ha sido diseñado desde el principio para ser multi-plataforma. Puede funcionar con Apache o Microsoft IIS como servidor web y en sistemas como Linux, BSD, Solaris, Windows y Mac OS X. Por otro lado, al estar implementado en PHP, es totalmente portable; por lo que el esfuerzo de satisfacer el escenario es **bajo**. Excluir el escenario conllevaría un riesgo **medio** dado a que el sistema se vería limitado a una sola plataforma.

## 4.5 Conclusiones

Del análisis anterior se concluye que la arquitectura propuesta satisface los atributos de calidad seleccionados, por lo que los principales riesgos han quedado mitigados.



## Conclusiones

Durante el desarrollo del trabajo se le dio cumplimiento a los objetivos planteados en el diseño teórico:

- ✓ Se hizo un estudio detallado de los principales aspectos de la descripción arquitectónica.
- ✓ Se definió la metodología a utilizar, acorde con los objetivos definidos.
- ✓ Se realizó una valoración de las principales tareas que debe llevar a cabo el rol de arquitecto según la metodología de desarrollo RUP (Proceso Unificado de Desarrollo).
- ✓ Se definieron los principales estilos arquitectónicos, sus componentes, configuraciones y restricciones, impuestas por los requisitos de los clientes.
- ✓ Se realizó un estudio detallado de las principales categorías de patrones arquitectónicos, se fundamentó la elección de los mismos para su aplicación en el sistema.
- ✓ Dada las restricciones impuestas por algunos requisitos no funcionales del cliente y en función de los principales atributos de calidad de la Arquitectura de Software se definieron las principales tecnologías y herramientas a utilizar en el desarrollo del sistema.
- ✓ Se cumplió con cada uno de los aspectos de la descripción de la arquitectura propuesta por la metodología de desarrollo, y se propuso la implementación de componentes de software para facilitar la reutilización de los mismos.
- ✓ Se analizaron los portales de ajedrez más difundidos en la actualidad, de donde se extrajeron los componentes fundamentales que presentan.
- ✓ Se describió la arquitectura del sistema a través de las diferentes vistas de la arquitectura según la metodología utilizada.
- ✓ Se realizó un estudio sobre los métodos de evaluación de arquitecturas de software. Donde la arquitectura propuesta fue evaluada según el método seleccionado.

Lo que condujo al desarrollo de una propuesta arquitectónica teniendo en cuenta los principales componentes que presentan los portales de ajedrez y centrada en atributos de calidad como portabilidad, escalabilidad, seguridad y rendimiento; que contribuyó al desarrollo de un portal web para el ajedrez con una alta aceptación, como se pudo apreciar en una encuesta realizada a los usuarios del mismo (73).

## Recomendaciones

- ✓ El refinamiento constante de la arquitectura propuesta, durante el ciclo de desarrollo.
- ✓ Escalar hacia una SOA (Arquitectura Orientada a Servicios).
- ✓ Incorporar nuevas funcionalidades al sistema (módulos propuestos).
- ✓ Reutilizar la arquitectura para el desarrollo de portales web orientados al ajedrez.

## Referencias Bibliográficas

1. **Torres de Diego, Mario.** *Fidel y el Deporte*. Ciudad de La Habana : Editorial Deportes, 2007. ISBN 978-959-203-077-0.
2. "Infodrez: Portal de Ajedrez de la UCI". **Garrido, Raciél, Rodríguez, José y Meneses, Abel.** 4, La Habana : Revista de Software Libre de la UCI, 2007.
3. **UCI - INDER.** Alianza Estratégica UCI - INDER. Ciudad de La Habana : s.n., 2005.
4. **Jiménez Molina, Christian.** Carta de agradecimiento. Ciudad de La Habana : s.n., 2006.
5. **UCI - ISLA.** Acta de la Primera Reunión de Trabajo ISLA - UCI. 2007.
6. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software*. La Habana : Editorial Félix Varela, 2004. ISBN: 978-84-7829-036-9.
7. *Arquitectura del Software: Arte y Oficio.* **Mollineda, Ramón.** 6, Valencia : Actualidad TIC, 2005.
8. **IEEE Std 1471-2000.** *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*. New York : The Institute of Electrical and Electronics Engineers, Inc., 2000. ISBN 0-7381-2519-9 SS94869.
9. **Colderram, J.** IESE Business School. *Portales en la red, un negocio en revisión*. [En línea] 7 de Septiembre de 2001. [Citado el: 3 de Mayo de 2009.]  
[http://www.iese.edu/es/files/Art\\_Colderram\\_Portales\\_esp\\_tcm5-7536.pdf](http://www.iese.edu/es/files/Art_Colderram_Portales_esp_tcm5-7536.pdf).
10. **Vanessa, M<sup>a</sup>. Sánchez Arce y Saorín Pérez, Tomás.** *LAS COMUNIDADES VIRTUALES Y LOS PORTALES COMO ESCENARIOS DE GESTIÓN DOCUMENTAL Y DIFUSIÓN DE INFORMACIÓN*. Murcia : EDITUM, 2001.
11. **Jones, Paul.** Utilidad y Tipos de Portales. *¿Que es un Portal Web?* [En línea] 19 de Diciembre de 2008. [Citado el: 3 de Mayo de 2009.] <http://utilidadytiposdeportales.wordpress.com/>.
12. **W3C.** World Wide Web Consortium. *Guía Breve sobre Estándares Web*. [En línea] 10 de Abril de 2008. [Citado el: 3 de Mayo de 2009.] <http://www.w3c.es/Divulgacion/Guiasbreves/Estandares>.
13. **Cobo, Cristóbal y Pardo, Hugo.** *"Planeta Web 2.0. Inteligencia colectiva o medios fast food"*. Barcelona / México DF : Grup de Recerca d'Interaccions Digitals, 2007. ISBN 978-84-934995-8-7.
14. **O'Reilly, Tim.** *"What Is Web 2.0 Design Patterns and Business Models for the Next Generation of Software"*. [En línea] 30 de Septiembre de 2005. [Citado el: 9 de Abril de 2009.]  
<http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>.

15. **Ribes, Xavier.** *"La Web 2.0. El valor de los metadatos y de la inteligencia colectiva"*. [En línea] 2007. [Citado el: 9 de Abril de 2009.]  
<http://www.campusred.net/TELOS/articuloperspectiva.asp?idarticulo=2&rev=73>.
16. **Edwards, Steven J.** Internet Chess Club. *Portable Game Notation Specification and Implementation Guide*. [En línea] 12 de Marzo de 1994. [Citado el: 14 de Mayo de 2009.]  
<http://www.chessclub.com/help/PGN-spec>.
17. **Carvajal, Vera.** *Contenido Web: una prioridad aún no comprendida*. [En línea] 12 de Mayo de 2008. [Citado el: 9 de Abril de 2009.] <http://www.astrolabio.com.co/periodismo-digital/6-periodismo-digital/1-contenido-web-una-prioridad-aun-no-comprendida.html>.
18. **Arango Sales, Humberto.** *Gestión de contenidos: el homo sapiens desde la antigüedad hasta la era digital*. [En línea] 2003. [Citado el: 9 de Abril de 2009.]  
[http://bvs.sld.cu/revistas/aci/vol11\\_5\\_03/aci09503.htm](http://bvs.sld.cu/revistas/aci/vol11_5_03/aci09503.htm).
19. **Cuerda, Xavier y Minguillón, Juliá.** *Introducción a los Sistemas de Gestión de Contenidos (CMS) de código abierto*. [En línea] 29 de Noviembre de 2004. [Citado el: 9 de Abril de 2009.]  
<http://mosaic.uoc.edu/articulos/cms1204.html>.
20. **Robertson, James.** *So, what is a CMS?* [En línea] 3 de Junio de 2003. [Citado el: 9 de Abril de 2009.]  
[http://www.steptwo.com.au/papers/kmc\\_what/index.html](http://www.steptwo.com.au/papers/kmc_what/index.html).
21. **Grupo Unicornios.** *Mini-Guía de Migración a SWL*. Ciudad de La Habana : s.n., 2008.
22. **Pressman, Roger S.** *Ingeniería de Software - Un enfoque práctico*. s.l. : Mcgraw-hill, 2001. ISBN:8448132149.
23. **Universidad EAFIT.** *Universidad EAFIT*. [En línea] 4 de Septiembre de 2006. [Citado el: 9 de Abril de 2009.]  
<http://www.eafit.edu.co/EafitCn/investigacion/grupos/ingenieria/ingenieriaSoftware/planEstrategico.shtm>.
24. **Billy Reynoso, Carlos.** *Introducción a la Arquitectura de Software*. Buenos Aires : s.n., 2004.
25. **Shaw, Mary y Clements, Paul.** *A field guide to Boxology: Preliminary classification of architectural styles for software systems*. Carnegie Mellon University : s.n., 1996.
26. **Bass, Len, Clements, Paul y Kazman, Rick.** *Software Architecture in Practice*. Michigan : Addison-Wesley, 1998. ISBN 0201199300, 9780201199307.
27. **Reynoso, Carlos y Kicillof, Nicolás.** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft*. Buenos Aires : s.n., 2004.

28. **Veryard, Richard.** Madasafish. *Component-Based Development*. [En línea] 19 de Mayo de 1999. [Citado el: 4 de Junio de 2009.] <http://www.users.globalnet.co.uk/~rxv/CBDmain/cbdfaq.htm#Component-Based%20Development>.
29. **Szyperski, Clemens.** *Component-Based Software Engineering*. Karlsruhe, Germany : Springer, 2008. ISBN 978-3-540-87890-2.
30. **Bachmann, Felix y Bass, Len.** *Technical Concepts of Component-Based Software Engineering*. Pittsburgh, Pensilvania : Carnegie Mellon University, 2000. CMU/SEI-2000-TR-008.
31. **Tedeschi, Nicolás.** Microsoft Developer Network. *¿Qué es un Patrón de Diseño?* [En línea] 2005. [Citado el: 3 de Junio de 2009.] <http://msdn.microsoft.com/es-es/library/bb972240.aspx#authorbrief>.
32. **Pérez de Ovalles, María A. y Mendoza, Luis Eduardo.** *Arquitectura del Sistema de Software*. Caracas - Venezuela : Universidad Simón Bolívar, 2003.
33. **Buschmann, Frank, Henney, Kevlin y Schmidt, Douglas C.** *PATTERN-ORIENTED SOFTWARE ARCHITECTURE: A Pattern Language for Distributed Computing*. England : John Wiley & Sons Ltd, 2007. ISBN-13: 978-0-470-05902-9 (hbk).
34. **Kruchten, Philippe.** *4+1 view model of software architecture*. [En línea] 1995. [Citado el: 9 de Abril de 2009.] <http://philippe.kruchten.com/architecture/>.
35. **Cano, Raúl de Villa.** *El Rol del Arquitecto de Software*. Medellín : UNIVERSIDAD EAFIT, 2008.
36. **ATI.** *Asociación de Técnicos de Informática*. [En línea] 2009. [Citado el: 9 de Abril de 2009.] <http://www.ati.es/spip.php?article1136>.
37. **Letelier, Patricio y M<sup>a</sup> Carmen, Penadés.** *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)*. [En línea] 2003. [Citado el: 9 de Abril de 2009.] <http://www.willydev.net/descargas/masyxp.pdf>.
38. **Cockbun, A. y Williams, L.** *The Costs and Benefits of Pair Programming*. Salt Lake City : s.n., 2000.
39. **Tramullas, Jesús.** *Arquitectura y prestaciones de los sistemas de gestión de contenidos*. [En línea] 2006. [Citado el: 9 de Abril de 2009.] <http://eprints.rclis.org/13234/>.
40. **CMS Award.** *Open Source CMS Award Previous Winners*. [En línea] 2008. [Citado el: 9 de Abril de 2009.] <http://www.packtpub.com/open-source-cms-award-previous-winners>.
41. **CMS Matrix.** *CMS Matrix. The Content Management Comparison Tool*. [En línea] 2009. [Citado el: 9 de Abril de 2009.] <http://www.cmsmatrix.org/>.
42. **Reynoso, Gonzalo.** *todolinux.com. Tutorial de Joomla*. [En línea] 3 de Octubre de 2005. [Citado el: 9

de Abril de 2009.] <http://www.todolinux.com/ftp/manuales/joomla.tutos/A.1.htm>.

43. **Reyero, Jose A.** Drupal Hispano. *Sobre Drupal*. [En línea] 16 de Marzo de 2006. [Citado el: 9 de Abril de 2009.] <http://drupal.org.es/drupal>.

44. **e107.** e107.org. *e107*. [En línea] 2009. [Citado el: 9 de Abril de 2009.] <http://www.e107.org/news.php>.

45. **Matos Padilla, Reynier.** *Rendimiento entre Sistemas Gestores de Contenido (CMS)*. [En línea] 22 de Mayo de 2008. [Citado el: 9 de Abril de 2009.] <http://www.maestrosdelweb.com/editorial/rendimiento-entre-sistemas-gestores-de-contenido-cms/>.

46. **Ohloh.** Ohloh, the open source network. *Compare Projects*. [En línea] 2009. [Citado el: 9 de Abril de 2009.] <http://www.ohloh.net/p/compare>.

47. **Schwartz, Jonathan.** *Helping Dolphins Fly*. [En línea] 16 de Enero de 2008. [Citado el: 9 de Abril de 2009.] [http://blogs.sun.com/jonathan/entry/winds\\_of\\_change\\_are\\_blowing](http://blogs.sun.com/jonathan/entry/winds_of_change_are_blowing).

48. **PostgreSQL.** *License*. [En línea] 2009. [Citado el: 9 de Abril de 2009.]

<http://www.postgresql.org/about/licence>.

49. **Bailey, Eoin.** drupal.org. *Chess*. [En línea] 11 de Noviembre de 2008. [Citado el: 17 de Mayo de 2009.] <http://drupal.org/project/chess>.

50. **Drupal - Chessboard.** drupal.org. *Chessboard Renderer*. [En línea] 14 de Enero de 2008. [Citado el: 17 de Mayo de 2009.] <http://drupal.org/project/chessboard>.

51. **Dougherty, Dale.** *LAMP: The Open Source Web Platform*. [En línea] 26 de Enero de 2001. [Citado el: 9 de Abril de 2009.] <http://www.onlamp.com/pub/a/onlamp/2001/01/25/lamp.html>.

52. **Necraft.** *April 2003 Web Server Survey*. [En línea] Abril de 2003. [Citado el: 9 de Abril de 2009.] [http://news.netcraft.com/archives/2003/04/13/april\\_2003\\_web\\_server\\_survey.html](http://news.netcraft.com/archives/2003/04/13/april_2003_web_server_survey.html).

53. **Ciberaula.** *Ventajas e inconvenientes de Linux*. [En línea] 2009. [Citado el: 9 de Abril de 2009.] [http://linux.ciberaula.com/articulo/ventajas\\_inconvenientes\\_linux](http://linux.ciberaula.com/articulo/ventajas_inconvenientes_linux).

54. **Ravioli, Pablo.** Monografias.com. *Lenguaje de programación para paginas web HTML*. [En línea] 2007. [Citado el: 12 de Mayo de 2009.] <http://www.monografias.com/trabajos7/html/html.shtml>.

55. **Musciano, Chuck y Kemeddy, Bill.** *HTML La Guía Completa*. Mexico, D.F. : Litografica Ingramex, 1999. ISBN 970-10-2141-X.

56. **Amaya, Marcelo.** Monografias.com. *XML Lenguaje de Marcas Extensible*. [En línea] 2001. [Citado el: 13 de Mayo de 2009.] <http://www.monografias.com/trabajos7/xml/xml.shtml>.

57. **Jacobs, Ian.** World Wide Web Consortium. *El XML del W3C cumple 10 años*. [En línea] 12 de Febrero

de 2008. [Citado el: 13 de Mayo de 2009.] [http://www.w3c.es/Prensa/2008/nota080212\\_XML10](http://www.w3c.es/Prensa/2008/nota080212_XML10).

58. **Valentine, Chelsea y Minnick, Chris.** *XHTML Serie Práctica*. s.l. : Pearson Educacion, 2001. ISBN 8420530115.

59. **Sánchez, Jordi.** jordisan.net. *¿Qué es un 'framework'?* [En línea] 29 de Septiembre de 2006. [Citado el: 9 de Abril de 2009.] <http://jordisan.net/blog/2006/que-es-un-framework/>.

60. **Reyero, Jose A.** Drupal Hispano. *Características de Drupal*. [En línea] 17 de Julio de 2005. [Citado el: 9 de Abril de 2009.] <http://drupal.org.es/caracteristicas>.

61. **VanDyk, John K. y Westgate, Matt.** *Pro Drupal Development*. New York : Apress, 2007. ISBN-10 (pbk): 1-59059-755-9.

62. **Drupal - Coding Standars.** drupal.org. *Coding standards*. [En línea] 22 de Abril de 2009. [Citado el: 9 de Abril de 2009.] <http://drupal.org/coding-standards>.

63. **Drupal API.** api.drupal.org. *Hooks*. [En línea] 2009. [Citado el: 9 de Abril de 2009.] <http://api.drupal.org/api/group/hooks/5>.

64. **Garfield, Larry.** GarfieldTech. *MVC vs. PAC*. [En línea] 31 de Diciembre de 2006. [Citado el: 15 de Junio de 2009.] <http://www.garfieldtech.com/blog/mvc-vs-pac>.

65. **Butcher, Matt.** *Learning Drupal 6 Module Development*. Birmingham - Mumbai : Packt Publishing, 2008. ISBN 978-1-847194-44-2.

66. **Chaffer, Jonathan.** drupal.org. *Drupal programming from an object-oriented perspective*. [En línea] 13 de Febrero de 2008. [Citado el: 9 de Abril de 2009.] <http://api.drupal.org/api/file/developer/topics/oop.html>.

67. **PEAR.** PHP Extension and Application Repository. *Coding Standards*. [En línea] 26 de Abril de 2009. [Citado el: 9 de Abril de 2009.] <http://pear.php.net/manual/en/standards.php>.

68. **Schackportalen.** El Portal de Ajedrez. *Software de Ajedrez*. [En línea] 2009. [Citado el: 14 de Mayo de 2009.] <http://www.schackportalen.nu/Espanol/esramportal.htm>.

69. **Tautenhahn, Lutz.** Lutz Tautenhahn. *Welcome to the free LT-PGN-VIEWER*. [En línea] 2008. [Citado el: 8 de Junio de 2009.] <http://www.lutanho.net/pgn/pgnviewer.html>.

70. *Método de Evaluación de Arquitecturas de Software Basadas en Componentes (MECABIC)*.

**Aleksander González, Marizé Mijares, Luis E. Mendoza, Anna Grimán, María Pérez.** Colimia, Mexico : s.n., 2005, Vol. vol 1.

71. **Carrascoso Puebla, Yoan Arlet, Chaviano Gómez, Enrique y Céspedes Vega, Anisleydi.**

*Procedimiento para la evaluación de arquitecturas de software basadas en componentes*. Ciudad de La

Habana : UCI, 2008.

72. **Reinoso, Carlos Billy.** *Introducción a la Arquitectura de Software.* Buenos Aires : s.n., 2004.

73. **Web Capablanca.** Capablanca: Sitio Oficial del Ajedrez Cubano. *¿Qué opina usted sobre la calidad del nuevo sitio?* [En línea] 2008. [Citado el: 6 de Junio de 2009.] <http://capablanca.co.cu/?q=node/105>.

## Bibliografía Consultada

**Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* La Habana : Editorial Félix Varela, 2004. ISBN: 978-84-7829-036-9.

**VanDyk, John K. y Westgate, Matt.** *Pro Drupal Development.* New York : Apress, 2007. ISBN-10 (pbk): 1-59059-755-9.


*Método de Evaluación de Arquitecturas de Software Basadas en Componentes (MECABIC).* **Aleksander González, Marizé Mijares, Luis E. Mendoza, Anna Grimán, María Pérez.** Colimia, Mexico : s.n., 2005, Vol. vol 1.

**Pressman, Roger S.** *Ingeniería de Software - Un enfoque práctico.* s.l. : Mcgraw-hill , 2001. ISBN:8448132149 .

**Butcher, Matt.** *Learning Drupal 6 Module Development.* Birmingham - Mumbai : Packt Publishing, 2008. ISBN 978-1-847194-44-2.

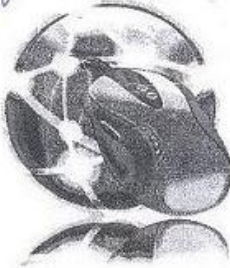


Anexos






# Relevante

*A: Propuesta de Arquitectura para un Gestor de contenidos para Ajedrez*



**VI Jornada Científica Estudiantil**  
**dado a los 7 días del mes de mayo de 2008**  
**“Año 50 de la Revolución”**

 Mayra Durán Benejam Decana Facultad 5	 Lazaro-Campoalegre Vera Secretario UJC	 Felipe Pérez Hernandez Presidente de la FEU
---	--	--

**“La ciencia es la verdadera sabiduría”**

Relevante VI JCE Nivel de Facultad.

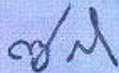
# Reconocimiento

**A** "Propuesta de Arquitectura para un gestor de contenidos orientado al ajedrez."

**Por haber obtenido Relevante**

Jornada  
Científica  
Estudiantil

*se hace camino al andar*



Msc. Melchor Gil Morel  
Rector UCI



Dr. Alina Ruiz Jhones  
Vice-Rector UCI

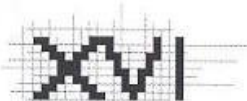
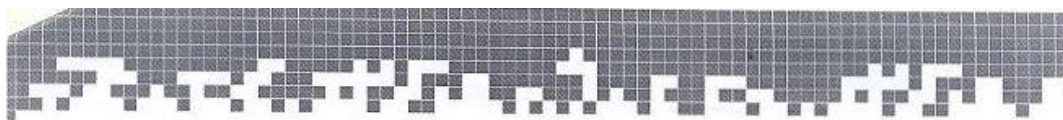


Reinier Tan Matamoro  
Presidente FEU UCI

Dado en La Habana a los 30 días del mes de Mayo de 2008  
Universidad de las Ciencias Informáticas

Relevante VI JCE Nivel UCI.





**Fórum de Ciencia y Técnica**

Segunda Etapa

Universidad de las Ciencias Informáticas

Se le otorga la categoría de premio

**MENCIÓN**

En el evento Universitario

**A** Eladio E. Avila Bagdasarova

Por la ponencia titulada:

"Propuesta de arquitectura para un gestor de contenidos orientado al Ajedrez"

"Avanzaremos con la más absoluta decisión en los propósitos del ahorro energético, motivados por la conciencia creciente que adquiere hoy nuestro pueblo".


**Fidel Castro Ruz**



Universidad de las Ciencias Informáticas

Melchor Gil Morell



  
Rector y Presidente Comisión Central

Alina Ruiz Jhones



  
Vicerrectora Primera

■ Dado en la Ciudad de La Habana a los 27 días del mes de junio de 2008

## Glosario

**API:** del inglés Application Programming Interface - 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 librería para ser utilizado por otro software como una capa de abstracción.

**Browser o Navegador:** aplicación para visualizar documentos WWW y navegar por Internet. En su forma más básica son aplicaciones hipertexto que facilitan la navegación por los servidores de navegación de Internet. Los más avanzados, cuentan con funcionalidades plenamente multimedia y permiten indistintamente la navegación por servidores WWW, FTP, Gopher, acceso a grupos de noticias, la gestión del correo electrónico, entre otras.

**Broadcast:** es un modo de transmisión de información donde un nodo emisor envía información a una multitud de nodos receptores de manera simultánea, sin necesidad de reproducir la misma transmisión nodo por nodo.

**Cache:** En informática, el término cache se aplica a un conjunto de datos duplicados de otros originales, con la propiedad de que los datos originales son costosos de acceder, normalmente en tiempo, con respecto a la copia en el cache. Cuando se accede por primera vez a un dato, se hace una copia en el cache; los accesos siguientes se realizan a dicha copia, haciendo que el tiempo de acceso medio al dato sea menor.

**Casos de Usos:** en ingeniería del software, es una técnica para la captura de requisitos potenciales de un nuevo sistema. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico.

**Cliente:** es un ordenador que accede a recursos y servicios brindados por otro llamado Servidor, generalmente en forma remota.

**Clúster:** conjunto de máquinas funcionando como unidad y trabajando juntas para tratar una única tarea.

**Encapsulamiento:** es una característica de la programación orientada a objetos. El encapsulamiento consiste en ocultar los detalles de la implementación de un objeto, a la vez que se provee una interfaz pública por medio de sus métodos permitidos. También se define como la propiedad de los objetos de permitir acceso a su estado solamente a través de su interfaz o de relaciones preestablecidas con otros objetos.

**FTP:** (File Transfer Protocol) es un protocolo de transferencia de ficheros entre sistemas conectados a una red TCP basado en la arquitectura cliente-servidor, de manera que desde un equipo cliente se puede conectar a un servidor para descargar ficheros desde él o para enviarle los archivos independientemente del sistema operativo utilizado en cada equipo.

**Hardware (soporte físico):** se utiliza generalmente para describir los artefactos físicos de una tecnología. Es el conjunto de elementos físicos que componen una computadora Disco Duro, CD-ROM, entre otros.

**Herencia:** es uno de los mecanismos de la programación orientada a objetos, por medio del cual una clase se deriva de otra de manera que extiende su funcionalidad. Una de sus funciones más importantes es la de proveer Polimorfismo y late binding.

**.htaccess:** fichero de configuración que se pone dentro de la estructura de directorios del sitio web y aplica directivas de configuración al directorio en el que está y a sus subdirectorios. A pesar de su nombre, este fichero puede contener cualquier tipo de directivas, no solo directivas de control de acceso.

**HTTP:** protocolo de transferencia de hipertexto (HTTP, HyperText Transfer Protocol) es el protocolo usado en cada transacción de la Web (WWW). El hipertexto es el contenido de las páginas web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceso a una página y la respuesta con el contenido.

**IDE (Integrated Development Environment):** Un entorno de desarrollo integrado, es un programa compuesto por un conjunto de herramientas para un programador. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un

compilador, un depurador y un constructor de interfaz gráfica GUI. Puede dedicarse en exclusivo a un sólo lenguaje de programación o bien, puede utilizarse para varios.

**IEEE:** corresponde a las siglas de The Institute of Electrical and Electronics Engineers, el Instituto de Ingenieros Eléctricos y Electrónicos, una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas. Es la mayor asociación internacional sin fines de lucro formada por profesionales de las nuevas tecnologías, como ingenieros eléctricos, ingenieros en electrónica, científicos de la computación e ingenieros en telecomunicación.

**Informática:** es la disciplina que estudia el tratamiento automático de la información utilizando dispositivos electrónicos y sistemas computacionales. Es la unión sinérgica del cómputo y las comunicaciones.

**ISO:** la Organización Internacional para la Estandarización o International Organization for Standardization (ISO), es una organización internacional no gubernamental, compuesta por representantes de los organismos de normalización (ONs) nacionales, que produce normas internacionales industriales y comerciales. Dichas normas se conocen como normas ISO.

**Microsoft:** (acrónimo de Microcomputer Software), es una empresa de Estados Unidos, fundada por Bill Gates y Paul Allen, los cuales siguen siendo sus principales accionistas. Dueña y productora de los sistemas operativos: Microsoft DOS y Microsoft Windows, que se utilizan en la mayoría de las computadoras del planeta.

**mod\_rewrite:** es un módulo del servidor web Apache. Este módulo usa un motor basado en re-escritura (basado en un analizador de expresiones regulares) para volver a escribir la URL solicitada sobre la marcha. Permite crear URL alternativas a las páginas dinámicas de forma que sean mas fáciles de recordar y también mejor indexadas por los buscadores. Para poder lograr esto es necesario tener acceso para editar los archivos de configuración .htaccess.

**OMG:** el Object Management Group (Grupo de Gestión de Objetos) es un consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos, tales como UML, XMI, CORBA.

**Paradigmas:** representa un enfoque particular o filosofía para la construcción del software.

**Polimorfismo:** se denomina a la capacidad que tienen objetos de diferentes clases de responder al mismo mensaje en programación orientada a objetos.

**Protocolo:** Conjunto de normas y procedimientos útiles para la transmisión de datos, conocido por el emisor y el receptor.

**RAM (Random Access Memory):** memoria de acceso aleatorio ó memoria de acceso directo.

**Servidor:** Una aplicación informática o programa que realiza algunas tareas en beneficio de otras aplicaciones llamadas clientes.

**Servicios:** es un conjunto de actividades que buscan responder a una o más necesidades de un cliente.

**Servicio WEB (en inglés Web service):** es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores.

**Sistema Operativo:** es un conjunto de programas destinados a permitir la comunicación del usuario con un computador y gestionar sus recursos de una forma eficaz.

**SOAP:** (siglas de Simple Object Access Protocol) es un protocolo estándar creado por Microsoft, IBM y otros, está actualmente bajo el auspicio de la W3C que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. SOAP es uno de los protocolos utilizados en los servicios Web.

**Software (soporte lógico):** los componentes intangibles de una computadora, es decir, al conjunto de programas y procedimientos necesarios para hacer posible la realización de una tarea específica.

**TCP/IP:** es un conjunto de protocolos de red que permiten la transmisión de datos entre redes de computadoras.

**URIs (Uniform Resource Locator):** es un localizador uniforme de recurso. Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.

**Versión:** en software, es un número que indica el nivel de desarrollo de un programa. Es habitual que una aplicación sufra modificaciones, mejoras o correcciones. El número de versión suele indicar el avance de los cambios.

**WYSIWYG (What You See Is What You Get):** Traducido: lo que ves es lo que obtienes, que aplicado a la edición significa trabajar con un documento con el aspecto real que tendrá. Editar una página de HTML en un editor que no sea WYSIWYG, implica trabajar con los códigos que indican el formato que tendrá el texto, sin ver el resultado final.

**WWW (World Wide Web):** es un sistema de documentos de hipertexto enlazados y accesibles a través de Internet. Con un navegador Web, un usuario visualiza páginas Web que pueden contener texto, imágenes u otros contenidos multimedia.