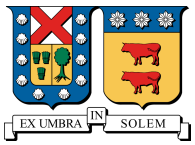
Universidad Técnica Federico Santa María
Departamento de Informática
Valparaíso – Chile



EVALUACIÓN DE TECNOLOGÍAS EMERGENTES PARA EL DESARROLLO DE APLICACIONES WEB

Memoria presentada como requerimiento parcial
para optar al título profesional de

INGENIERO CIVIL EN INFORMÁTICA

por

João Fuentes Pacheco

Comisión Evaluadora:
Raúl Monge (Guía, UTFSM)
(UTFSM)

AGOSTO 2013

Universidad Técnica Federico Santa María
Departamento de Informática
Valparaíso – Chile

TÍTULO DE LA MEMORIA:
EVALUACIÓN DE TECNOLOGÍAS EMERGENTES PARA EL DESARROLLO DE APLICACIONES WEB

AUTOR:
JOÃO FUENTES PACHECO

Tesis presentada como requerimiento parcial para optar al título profesional de **Ingeniero Civil en Informática** de la Universidad Técnica Federico Santa María.

Profesor Guía: _____
Raúl Monge

Profesor Correferente _____

Agosto 2013.
Valparaíso, Chile.

Autor

Resumen

Resumen ...

Abstract

Abstract ....

Índice de Contenidos

Glosario

UTFSM	Unversidad Técnica Federico Santa María
XML	Extensible Markup Language
PC	Personal Computer
ERP	Enterprise Resource Planning
SIW	Sistemas de Información Web
HTTP	HyperText Transfer Protocol
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
XHTML	eXtensible HyperText Markup Language
MIME	Multipurpose Internet Mail Extensions
URL	Uniform Resource Locator
CGI	Common Gateway Interface
PHP	Hypertext Pre-processor
ASP	Active Server Pages
IIS	Internet Information Server
AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
GPS	Global Positioning System
I/O	Input/Output
JS	JavaScript
NPM	Node Package Module

NoSQL	Base de Datos no Relacional
MVC	Modelo Vista Controlador
ORM	Object Relational Mapping
WWW	World Wide Web
SOA	Service Oriented Architecture
ESB	Enterprise Service Bus
DAL	Data Access Layer

Capítulo 1

Introducción

1.1 Contexto

La aparición de aplicaciones y sitios Web ha resultado en la creación y explotación de nuevos mercados conformados por servicios con los que antes sólo se podía soñar, algunos ejemplos son: sitios de comercio electrónico; *E-learning* [1]; redes sociales, incluyendo sus múltiples servicios de mensajería, juegos online y actualización de contenido en tiempo real; sitios de *streaming* [2]; tan sólo por nombrar algunas [12][3]. Esto conlleva a un importante crecimiento y evolución tanto de la metodología desarrollo como de las tecnologías utilizadas a la hora de crear una aplicación web.

[1] El E-learning es un modelo de formación a distancia que utiliza Internet como herramienta de aprendizaje. Este modelo permite al alumno realizar el curso desde cualquier parte del mundo y a cualquier hora. Con un PC y una conexión a Internet, el alumno realiza las actividades interactivas planteadas, accede a toda la información necesaria para adquirir el conocimiento, recibe ayuda del profesor; se comunica con su tutor y sus compañeros, o evalúa su progreso.[1]

[2] El término streaming hace alusión a una corriente continua, en este caso, de datos. Streaming corresponde a la distribución de contenido multimedia a través de una red (generalmente internet) de tal forma que el usuario consume el producto al mismo tiempo que se va descargando (por ejemplo Youtube). Este tipo de tecnología funciona mediante un búfer de datos que va almacenando el contenido descargado para luego mostrarse al usuario; a diferencia de la descarga de archivos, que requiere que el usuario descargue los archivos por completo para poder acceder a ellos.

[3] Página 1

El sitio Web es el medio más barato para darse a conocer rápidamente con un alcance, incluso, a nivel mundial. Esto es extensible no sólo a empresas que comercializan productos y servicios, o a profesionales autónomos, sino que también a personas u organizaciones que actúan sin ánimo de lucro, que intentan divulgar sus obras, inquietudes o ideas.

Al comienzo, los sitios Web ofrecían casi de forma exclusiva contenidos basados en texto y eran, por lo general, bastante estáticos; en la actualidad son sitios interactivos con abundancia de elementos multimedia.

La evolución, algo inherente al espiritu humano, se hace presente en todas y cada una de las áreas de investigación, tanto en las ciencias de la física y química que permiten la construcción de hardware más potente, como en los procesos y tecnologías de desarrollo de software, específicamente, y lo que es de interés para este documento, las tecnologías de Desarrollo Web.

1.2 Definición del Problema

Hoy en día las herramientas Web son fundamentales en todo ámbito de negocios, desde soluciones comerciales como carritos de compra, catálogos empresariales, e incluso complejos sistemas ERP, que consisten en una arquitectura de software para empresas que facilita e integra la información entre las funciones de manufactura, logística, finanzas y recursos humanos de la empresa.

Como se mencionó anteriormente, a la evolución de los servicios ofrecidos por las diversas empresas que pertencen al rubro, hay que sumar el continuo aporte y empoderamiento de los propios usuarios quienes crean y consumen contenido a tasas cada vez más altas.

Como ya ha ocurrido con la evolución desde páginas web estáticas a sitios web dinámicos, o la aparación de tecnologías asincrónicas, es natural esperar que esta evolución continue, después de todo la web evoluciona de acuerdo a las necesidades del usuario.

Es fundamental reconocer y probar las nuevas tecnologías que han ido apareciendo con el transcurrir de los años, de ese modo, poder evaluar de forma concreta y objetiva, en que contexto son de utilidad.

1.3 Objetivos

1.3.1 Objetivos Generales

El objetivo de esta memoria es realizar una evaluación de diversas tecnologías emergentes de desarrollo web y compararlas con una implementación de carácter tradicional bajo ciertos criterios. Para ello se deben cumplir los siguientes objetivos generales:

1. Explicar el avance de las tecnologías de desarrollo web, y cómo han evolucionado desde páginas web estáticas hasta el concepto de aplicaciones web.

2. Evidenciar empíricamente la diferencia de rendimiento entre tecnologías emergentes, utilizando como base de comparación, tecnología de carácter más tradicional.

1.3.2 Objetivos Específicos

1. Investigar tecnologías existentes y emergentes para el desarrollo de aplicaciones web.

   (a) Revisar evolución histórica de las tecnologías asociadas a desarrollo de aplicaciones Web.
   (b) Realizar un levantamiento del estado del arte.

2. Investigar y evaluar cómo las tecnologías recientes pueden ayudar a desarrollar mejores aplicaciones Web y mejorar los procesos de desarrollo de software.
   (a) Definir criterios de clasificación general y selección de las tecnologías específicas a investigar
   (b) Aplicar los criterios anteriores para seleccionar un conjunto de tecnologías a investigar en mayor profundidad.
   (c) Evaluar en más detalle las tecnologías seleccionadas.
3. Definir una aplicación de prueba y desarrollar un prototipo aplicando las tecnologías seleccionadas que permita entender, integrar y evaluar estas tecnologías. Comparar con la aplicación de tecnologías más tradicionales.
   (a) Diseñar y desarrollar el prototipo aplicando diferentes tecnologías.
   (b) Evaluar cada uno de los casos de aplicación
4. Definir el tipo pruebas que midan rendimiento y como se aplicarán a los prototipos.
5. Hacer una evaluación global y obtener conclusiones.

1.4 Metodología de trabajo

El desarrollo de este trabajo se divide en etapas, las cuales son:

1. Definición de Objetivos y Estado del Arte: Esta etapa consiste en la definición de los Objetivos Generales y Específicos de esta memoria, se investiga la evolución histórica de las diversas tecnologías de desarrollo web, y se realiza un levantamiento del estado del arte algunas de las metodologías más importantes de desarrollo web.

2. Selección de Tecnologías y Ambiente de Desarrollo: Esta etapa consiste en la definición y aplicación de criterios de selección que permitan filtrar y escoger de entre las tecnologías investigadas en la etapa anterior. Por otra parte costa de la implementación de un ambiente de desarrollo necesario para realizar las diversas pruebas requeridas en los objetivos. Dicha implementación es necesaria para cada tecnología escogida.

3. Definición de pruebas y Experimentos: Esta etapa consiste en la definición las pruebas orientadas a medir el rendimiento de los prototipos construidos. Una vez definidas, se procederá a correr los experimentos en los prototipos, midiendo su desempeño.

4. Analisis y Resultados: En esta etapa se analizarán y compararán los resultados obtenidos en la etapa anterior, explicando cada uno de ellos.

5. Conclusiones: La última etapa consta de la obtención de conclusiones del trabajo realizado, tomando en cuenta todas las etapas anteriores.

1.5 Estructura del documento (En desarrollo de acuerdo a lo enviado al profesor)

En el capítulo 1 del presente documento se realiza una descripción del problema, contextualizando al lector y motivandolo con las consecuencias de su resolución. Además se presentan los objetivos que busca cumplir esta memoria.

En el capítulo 2 se define que es una aplicación web, y se explican los procesos de desarrollo de software necesarios para poder construirla; comenzando por los procesos de desarrollo de software en general, los cuales son los precursores de los procesos de desarrollo web.

En el capítulo 3 se realiza una revisión sobre la evolución histórica de las tecnologías

web. Se explican las características y las tecnologías utilizadas más populares en su época.

En el capitulo 4 se explica la importancia del uso de una arquitectura a la hora de desarrollar una aplicación. Son presentadas algunas de las más importantes y se toma la desición de cuál utilizar en este trabajo.

En el capítulo 5 se definen los criterios de selección bajo los cuales se escogerán las tecnologías a usar al momento de realizar las pruebas. Se divide la investigación en tecnologías tradicionales y emergentes.

Capítulo 2

Estado del Arte

2.1 Aplicaciones Web

Una aplicación Web corresponde a un software basado en internet, donde una gran población de usuarios realizan peticione remotas a través de un navegador web y esperan rpuestas de un servidor web [2][1]. Corresponde a una aplicación que se codifica en un lenguaje soportado por los navegadores web. Es decir las aplicaciones web corresponden al modelo cliente-servidor.

Una aplicación web se distingue por el uso de *hipermedia*, que conjuga tanto la tecnología hipertextual, como la multimedia. Si la multimedia proporciona una gran riqueza en los tipos de datos, el hipertexto aporta una estructura que permite que los éstos puedan presentarse y explorarse siguiendo distintas secuencias, de acuerdo a las necesidades y preferencias del usuario.

En la actualidad existe una gran variedad de aplicaciones Web, que van desde páginas sólo de carácter informativo hasta aplicaciones complejas que ofrecen diversidad de

[1]Página 90

servicios al usuario, ya sean interacción con otros usuarios a través de juegos en linea, mensajería y *streaming* entre otras.

Más allá del tipo de aplicación con el que se esté trabajando, es de suma importancia contar con un proceso de desarrollo de software, con el fin de estandarizar su proceso de creación. A continuación se exponen algunos de los procesos de desarrollo de software tanto general, como orientados al desarrollo de aplicaciones web

2.2 Procesos de Desarrollo de Software de carácter general

Desde el punto de vista de la ingeniería de software, es importante contar con los mecanismos adecuados (métodos y tecnologías de desarrollo, hardware acorde a la aplicaciones a desarrollar, entre otros) para que la creación de este tipo de aplicaciones satisfaga las necesidades tanto de los usuarios como de los clientes que contratan su desarrollo.

En cualquier caso, existen criterios universalmente aceptados acerca del desarrollo software. Por ejemplo, el modelo de proceso más adecuado para el desarrollo de software es un proceso iterativo e incremental, puesto que a diferencia de otros modelos de proceso, como el modelo en cascada, permite la obtención de diversas versiones del producto software antes de su entrega final, por ende permite su depuración y validación progresiva, lo que sin duda redundará en un software de mejor calidad. Además, con este tipo de proceso es posible añadir o modificar requisitos que no han sido detectados con anterioridad [12][2].

[2]Página 2

Dentro de los procesos de desarrollo de software más importantes, son destacables Modelo de cascada y el Modelo iterativo incremental.

Estas metodologías se conocen también como "tradicionales" e imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con la meta de conseguir uno más eficiente y predecible. Para ello, se hace un especial hincapié en la planificación total de todo el trabajo a realizar y una vez que esta todo detallado, comienza el ciclo de desarrollo del producto. Este planteamiento está basado en el resto de disciplinas de ingeniería, a pesar de que el software no pueda considerarse como la construcción de una obra clásica de ingeniería [12][3].

Una de sus principales deficiencias corresponde a que si existe un cambio, es posible que toda la planificación se venga abajo, es por ello que no son métodos adecuados cuando se trabaja en un entorno que pueda variar constantemente.

2.3 Procesos de Desarrollo de Aplicaciones Web

En los últimos años ha surgido un conjunto de métodos para desarrollar aplicaciones Web, los cuales presentan en forma explícita su modelo de proceso, es decir las actividades técnicas y gerenciales que son requeridas para el desarrollo de la aplicación.

Los métodos que se presentarán se han agrupado de acuerdo a su modelo de proceso y al contexto particular donde pueden ser aplicados. Es necesario aclarar que, ninguno de ellos guía al grupo de desarrollo en la construcción de aplicaciones Web para múltiples contextos, debido a su entorno multivariable. Entre los métodos más conocidos se encuentran [2][4]:

[3]Página 2.
[4]Páginas 90-92

2.3.1 Métodos ágiles para el desarrollo de aplicaciones

Estos procesos aportan como novedad, nuevos métodos de trabajo que apuestan por equilibrar la relación **proceso -esfuerzo**. En otras palabras, ni se pierden en la excesiva burocracia de los métodos tradicionales, ni apuestan por una ausencia total de procesos.

Los procesos ágiles son una buena elección cuando se trabaja con requisitos desconocidos y/o variables. De no existir requisitos estables, la posibilidad de utilizar un proceso "tradicional"no es recomendable. En estas situaciones, un proceso adaptativo se considera mucho más efectivo que un proceso predictivo. Los métodos ágiles para el desarrollo de software se caracterizan por poseer iteraciones cortas, pruebas continuas y frecuente replanificación basada en la realidad actual.

Dentro de las metodologías basadas en este enfoque destacan: Extreme Programming (XP), Open Source y Dynamic Systems Development Method (DSDM).

2.3.2 Métodos para el desarrollo de sistemas de información Web

Estos métodos se caracterizan por seguir una secuencia de fases y pasos requeridos para desarrollar un SIW, y asegurar la calidad del mismo. Cubren todo el ciclo de desarrollo de un SIW, emplean técnicas de análisis, de diseño orientado a objetos y utilizan un enfoque iterativo. Son comparables al proceso de desarrollo iterativo incremental, pues integra los siguientes procesos [11]:

- Análisis preliminar para sistemas de información Web
- Selección de lenguaje y desarrollo de la aplicación
- Implementación
- Mantenimiento

- Estándares y Documentación

Uno de los más conocidos corresponde a la Metodología para desarrollar Sistemas de Información Web (MIDAS).

2.3.3 Métodos para el desarrollo de aplicaciones hipermedia

La mayoría de estos métodos sólo cubren parcialmente el ciclo de desarrollo de las aplicaciones hipermedia, dando especial importancia al diseño. Por lo general utilizan dos técnicas en cualquier diseño de aplicaciones hipermedia: Modelo Entidad-Relación y técnicas de Orientación a Objetos.

Los más difundidos son: Aplicando modelos de proceso de software al desarrollo de aplicaciones hipermedia, The Object-Oriented Hypermedia Design Model (OOHDM) Relationship Management Methodology (RMM), Hypermedia Flexible Process Modeling (HFPM) y el Enfoque de Ingeniería de Lowe-Hall's.

2.3.4 Métodos para el desarrollo de sitios Web de aprendizaje (e-learning)

El objetivo de estos métodos es ayudar a los diseñadores de los cursos y profesores a desarrollar sitios Web de aprendizaje entendibles, por lo tanto se incorpora gran variedad de componentes organizacionales, administrativos, didácticos y tecnológicos, pues estos métodos consideran el elemento humano (alumnos, profesores, ayudantes, administradores), los recursos de aprendizaje basados en Web, otros recursos de aprendizaje (textos o guías) y la infraestructura tecnológica necesaria para desarrollar el proceso de aprendizaje. Cabe destacar que hacen énfasis en la reutilización de componentes para reducir el tiempo y costo de desarrollo.

2.3.5 Métodos para el desarrollo de aplicaciones de comercio electrónico: (e-commerce)

Estos métodos están basados en el reciclado e integración de componentes de software con el fin de lograr funcionalidades empresariales para aplicaciones de e-commerce: negociación, mediación; "workflow" interempresarial y notificaciones de eventos.

Uno de los más difundidos es el Marco de Referencia Basado en componentes para e-commerce.

Capítulo 3

Tecnologías utilizadas en el Desarrollo de aplicaciones Web

3.1 Un poco de historia

Antiguamente, la creación de un sitió web se limitaba sólo a escribir cada página directamente con mediante código HTML. Esta tarea es factible solamente en sitios cuyo contenido es limitado y sus actualizaciones son casi nulas, características propias de la *Web 1.0* es decir, las típicas que ostentaban los sitios durante la primera mitad de los años 90.

Posteriormente aparecen los lenguajes de desarrollo Web intentan facilitar las tareas de los creadores de aplicaciones, de manera que se automaticen los procesos, y permitan entrar al juego a los usuarios, quienes se mantenían pasivos hasta ese momento. Es decir, se crea la web 2.0.

Actualmente, y gracias al nacimiento de las redes sociales, el papel que juegan los usuarios en la web es cada vez más importante; tanto así que su uso ha permitido

realizar cambios tan drámaticos en la sociedad entre los que destacan por ejemplo, comunicarse con personas en casi cualquier parte del mundo o el auge de levantamientos populares en paises como Egipto.

Por lo tanto, mientras que con HTML sólo es posible crear sitios Web estáticos, utilizando lenguajes de desarrollo Web es posible crear sitios Web dinámicos. Se conoce con el nombre de sitio Web dinámico a aquel cuyo contenido se genera a partir de lo que un usuario introduce en un formulario web o, simplemente, formulario.

A continuación, se expondrá la evolución de las herramientas y el pensamiento a través del tiempo, la cual se ha categorizado en: *web 1.0, web 2.0, actualidad y evolución.*

3.2 Web 1.0

3.2.1 Características

La Web 1.0 o "web estática"(1991-2003) es la forma más básica que existe, con navegadores de sólo texto bastante rápidos. La aparición de HTML hizo que las páginas web fuesen más agradables a la vista. Paralelamente aparecen los primeros navegadores visuales tales como Internet Explorer y Netscape [3]. Su principal característica es que es de sólo lectura, es decir que para el usuario no es posible interactuar con el contenido de la página estando totalmente limitado a lo que el Desarrollador sube a ésta.

3.2.2 Tecnologías de desarrollo

Si bien, la tecnología predominante en la web 1.0, es el código HTML, se debían realizar las transferencias necesarias de información entre cliente y servidor, información que principalmente correpsondía a hipertexto. Es por ello que se creó un protocolo, es decir una serie de reglas utilizadas por los computadores para poder realizar transferencias

de archivos de datos, sonido o imagen. Dicho protocolo corresponde a HTTP o acrónimo de *HyperText Transfer Protocol*.

Desde 1990, el protocolo HTTP es el más utilizado en Internet. Si bien su versión 0.9 sólo tenía la finalidad de transferir los datos a través de Internet, su versión 1.0 permite la transferencia de mensajes con encabezados que describen el contenido de los mensajes mediante la codificación MIME [1].

El propósito del protocolo HTTP es permitir la transferencia de archivos principalmente, en formato HTML. entre un navegador y un servidor web, mediante una cadena de caracteres conocida como dirección URL.

HTML corresponde al acrónimo de *HyperText Markup Language* y se trata de un conjunto de etiquetas que sirven para definir el texto y otros elementos que compondrán una página web. Básicamente este lenguaje indica a los navegadores cómo deben mostrar el contenido de dicha página web.

HTML se creó con el objetivo de divulgar información, principalmente texto y posteriormente texto con imágenes. Creado en 1986 por el físico nuclear Tim Berners-Lee [9], no se pensó que llegaría a ser utilizado para crear sitios de consulta con carácter multimedia. Sin embargo, pese a esta deficiente planificación, se han ido incorporando modificaciones con el tiempo, estos son los estándares del HTML[2] [8].

Desde el punto de vista del webmaster, la tarea de mantención de la página es relativamente simple, considerando la base de la web 1.0. Sin embargo, se convierte en

[1] *Multipurpose Internet Mail Extensions*: es un estándar que clasifica los recursos y provee información de cómo deben ser manejados. Mediante este estándar es posible la correcta manipulación e interpretación de diversos tipos de archivos por parte de los programas, en este caso, navegadores. Por ejemplo, gracias a MIME, los navegadores pueden abrir correctamente un archivo ".txtçomo un recurso de texto plano y no como un video u otro tipo.

[2] El estándar actual corresponde a HTML5

una tarea titánica en aquellos sitios con muchos contenidos y que incorporan frecuentemente novedades. Por ejemplo, si se quiere realizar uno o más cambios sobre algún elemento común a todas las páginas del sitio, se debe aplicar dicho cambio en todas las páginas, una por una, con lo que se convierte en un trabajo muy tedioso. Es en este punto que nace la necesidad de integrar al usuario a la faceta de creación y mantención de contenidos.

Como ya se ha mencionado, la tecnología preponderante es HTML, introducción de formularios y CGI. No obstante, también en esta etapa de desarrollo surgen lenguages de scripting para la Web, tales como : PHP y ASP.

- CGI [3]: Es posible calsificarlo como en el límite, pues corresponde a la primer intento de realizar el salto de contenidos estáticos a contenidos dinámicos. En las aplicaciones CGI, el servidor web pasa las solicitudes del cliente a un programa externo. La salida de dicho programa es enviada al cliente en lugar del archivo HTML tradicional. El cliente se encarga de interpretar esta salida.

- PHP: Una forma de generar el contenido dinámico, corresponde a que sea el servidor quien ejecute las secuencias de comandos para generar la página HTML. PHP [4] tiene la ventaja de ser gratuito y versátil, pues es soportado por la mayoría de los sistemas operativos y servidores; además de contar con múltiples herramientas de desarrollo como frameworks[5] donde destacan PHPCake o Symfony. Para utilizar PHP, el servidor Web debe entenderlo. Por lo general, las páginas Web que contienen comandos PHP utilizan la extensión ".php" en lugar de ".html". De todos modos, el cliente nunca ve el código PHP, sino los resultados que produce en código HTML.

- ASP.NET: Una alternativa es la que ofrece Microsoft para generar sitios Web

[3]Common Gateway Interface por sus siglas en inglés, o Interfaz de Puerta de Enlace Común.
[4]Hypertext Pre-processor por sus siglas en inglés.
[5]Framework es un concepto sumamente genérico, se refiere a "ambiente de trabajo, y ejecución".En general los framework son soluciones completas que contemplan herramientas de apoyo a la construcción (ambiente de trabajo o desarrollo) y motores de ejecución (ambiente de ejecución).

dinámicos, conjuntamente con su software servidor IIS [6]. Se trata de ASP [7], que desde su primera versión ha evolucionado hasta denominarse ASP.NET y estar dentro de la plataforma ".NET". Una de las principales ventajas de ASP.NET es la gran cantidad de lenguajes que soporta. ASP.NET constituye un entorno abierto en el que se puede combinar código HTML, scripts y componentes ActiveX del servidor para crear soluciones dinámicas y de calidad para la Web. Las páginas que utilizan esta tecnología tienen la extensión ".asp".

3.3 Web 2.0

3.3.1 Características

El término Web 2.0 está asociado a aplicaciones web que están desarrolladas para compartir información, pues su diseño está centrado en el usuario. Un sitio Web 2.0 está pensado para que los usuarios puedan interactuar y colaborar entre sí [3], tomando el rol de creadores de contenido generado por ellos mismos en una comunidad virtual, lo cual es diametralmente opuesto al concepto de pasividad del usuario, algo predominante en la web 1.0.

Por lo tanto, la Web 2.0 es una evolución del viejo concepto de cómo se usa la web, de manera unidireccional, como consumidores pasivos. El término, acuñado por Tim O'Reilly[8] en una conferencia del renacimiento y la evolución de la web, designa una nueva forma de servicios web basados en la participación de los usuarios, quienes conforman el motor básico del sistema de información.

Debido al aumento en la participación de los usuarios, nacen premisas como: "todos tienen algo que decir y todos pueden hacerlo" (Orihuela, 2006). El volumen de datos

[6]Internet Information Server, por sus siglas en inglés
[7]Active Server Pages, por sus siglas en inglés
[8]En [6], realiza interesantes comparaciones entre los elementos de la web 1.0 y la web 2.0

generados es tal, que se necesitan sistemas de filtrado, clasificación y organización de la información; sistemas que además, deben estar basados en la arquitectura de la participación y la inteligencia colectiva [5].

El hecho de que la Web 2.0 sea cualitativamente diferente de las tecnologías web revisadas en el apartado anterior, ha sido cuestionado por el creador de la World Wide Web Tim Berners-Lee, quien calificó en su tiempo al término como "tan sólo una jerga", pués tenía la intención de que la Web incorporase estos valores en el primer lugar.

3.3.2 Tecnologías de desarrollo

Ante la necesidad de empoderar a los usuarios, fueron surgiendo varios lenguajes de programación y tecnologías orientadas al desarrollo web necesarios para lograr la creación de sitios dinámicos.

Un sitio web dinámico se puede generar a través de secuencias de comandos en un servidor web; cuando el cliente web recibe la respuesta, la trata como una pagina HTML y la despliega. Un ejemplo de esto es cuando un usuario rellena los campos de un formulario y realiza el envío de la información, al momento de llegar al servidor dicha información se entrega a un programa o secuencia de comandos para que sea procesada, que por lo general corresponde a una interacción con una base de datos y la generación de una página HTML con información personalizada, la cual es reenviada al cliente.

Por lo general, los sitios web dinámicos, estan compuestos por la combinación:

- Plataforma del servidor web: Apache, Tomcat, entre otros.
- Gestor de base de datos, que por lo general es de carácter relacional: Oracle, PostgreSQL, Microsoft SQL Server, MySQL, entre otros.
- Lenguajes y tecnologías de programación web: Perl, PHP, JSP, JavaScript, entre

otros.

Dependiendo de las necesidades, se define la combinación adecuada.

Algunas de las tecnologías de la llamada web 2.0 son:

- JavaScript: A pesar de la gran potencia de las tecnologías anteriores, ninguna de ellas puede responder, por ejemplo, a los movimientos del ratón o interactuar de manera directa con los usuarios. Para lograr esto, es necesario tener secuencias de comandos embebidas en las páginas HTML, pero que a diferencia de, por ejemplo PHP, se ejecuten en la máquina cliente y no en el servidor. JavaScript es un lenguaje de scripts interpretado que se integra directamente en páginas HTML (a veces por modularidad se separa en ficheros con extensión ".js") y es interpretado, en su totalidad, por el cliente Web en tiempo de ejecución, sirviendo así para todos los sistemas operativos.
- AJAX: O *Asynchronous JavaScript And XML* es un término que engloba la utilización de varias tecnologías, para crear aplicaciones Web dinámicas que se ejecutan en el cliente. Entre estas tecnologías destacan JavaScript, XML, XHTML, HTML y CSS. Al realizar cambios sobre la misma página sin necesidad de recargarla, se consigue un notable aumento de interactividad y velocidad.

3.4 Actualidad

3.4.1 Características

La Web actual se caracteriza por premiar la creatividad de los usuarios, fomentar su participación y transmisión del conocimiento entre pares potenciando el sentido de comunidad. En esta nueva época la Web ya no es meramente informativa, sino que el usuario toma rol consumidor, intercambiador y creador de contenidos. Los sitios web se convierten en fuentes de contenido y expresión para los usuarios. Es aquí donde de fundan los actuales cimientos de la Web: redes sociales y sabiduría de las multitudes.

3.4.2 Tecnologías de desarrollo

Dentro de las tecnología utilizadas en esta web se encuentran:

- Redes Sociales: Las Redes Sociales son sitios web que permiten a las personas conectarse con sus amigos e incluso realizar nuevas amistades, a fin de compartir contenidos, interactuar y crear comunidades sobre intereses similares. Gracias a la alta tasa de conexión a internet y su posterior desarrollo como medio de comunicación de masas, ha posibilitado que las redes sociales puedan existir, además del espacio físico, en el espacio virtual. Esto facilita su extensión a lo largo de diferentes regiones y del mundo, superando para siempre el factor geográfico que muchas veces las limitaba. Cabe destacar que han sido (y son) un factor de cambio clave dentro de la sociedad actual, en ambitos tan distintos, que van desde llevar relaciones amorosas al plano virtual hasta ser el medio de comunicación primario a la hora de convocar manifestaciones.

- Mushup (Web Híbrida): Corresponde a aplicaciones webs que contiene a su vez aplicaciones webs. Consta por lo general, en el uso de las API de variadas compañías. Una situación recurrente es el uso de Google Maps, que corresponde a una aplicación web, dentro de páginas que se dedican al seguimiento de, por ejemplo, vehículos a través de un sistema GPS; lo que corresponde a otra aplicación web.

- QOOXDOO [9]: Qooxdoo es un framework universal de JavaScript que permite crear aplicaciones para una amplia gama de plataformas. Qooxdoo aprovecha las tecnologías web más modernas, como HTML5 y CSS3. Es de código abierto, está totalmente basado en clases y trata de aprovechar las características de orientación a objetos de JavaScript. Se basa completamente en los espacios de nombres (namespace) y no requiere grandes modificaciones a los objetos nativos de JavaScript para permitir una integración con otras bibliotecas y código de usuario personalizado [7]. Una aplicación típica de qooxdoo se crea mediante el uso de las herramientas de desarrollo integrado y el modelo de programación del

[9] http://qooxdoo.org/

lado del cliente basada en orientación a objetos de JavaScript. Algunas de sus características son:

1. Qooxdoo soporta una amplia gama de entornos de JavaScript, tales como navegadores convencionales [10] y móviles [11]
2. No necesita plugins (ActiveX, Flash, Silverlight)
3. Mantiene objetos nativos de JavaScript, con el fin de permitir una fácil integración con bibliotecas y código personalizado.
4. Al estar bajo el paradigma de la orientación a objetos, está basado en clases (en su totalidad). Además soporta clases abstractas.
5. Cuenta con soporte completo para programación basada en eventos
6. El desarrollo de aplicaciones qooxdoo es totalmente compatible con todas las plataformas, como Windows, todos los sistemas Unix (Linux), Mac OS X.
7. Cuenta con muchas aplicaciones de muestra y ejemplos.

- Node.js[12]: Es un entorno de programación basado en el lenguaje de programación Javascript, con I/O de datos y una arquitectura orientada a eventos. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como por ejemplo, servidores web. Las primeras encarnaciones de JavaScript vivían en los browsers, es decir en el frontend. Sin embargo, lo anterior es solo un contexto, pues JavaScript es un lenguaje "completo"; es decir, se puede usar en muchos contextos y alcanzar con éste, todo lo que se puede alcanzar con cualquier otro lenguaje "completo". Por ello Node.js realmente es sólo otro contexto: permite correr código JavaScript en el backend, fuera del browser. Para ejecutar el código JavaScript que se pretende correr en el backend, debe ser interpretado y ejecutado. Node.js se encarga de esta tarea haciendo uso de la Maquina Virtual V8 de Google; que por lo demás es el mismo entorno de ejecución para JavaScript

[10] Internet Explorer, Firefox, Opera, Safari, Chrome
[11] iOS, Android
[12] http://nodejs.org/

que Google Chrome utiliza. Además, Node.js viene con muchos módulos útiles [13], de manera que no hay que escribir todo de cero.

- V8 Engine [14]: Es un motor de JS desarrollado por Google. La ejecución de programas JS se realiza compilando el código, aumentando el desempeño respecto a Java ejecutado en lenguaje interpretado Bytecode [15]. Algunas de sus características son:

  1. Está escrito en C++ y es usado en Google Chrome.
  2. Está integrado en el navegador de internet del sistema operativo Android, al menos desde su versión Froyo.
  3. Corre en Windows (desde la versión XP en adelante), Mac OS X 10.5 (Leopard) y Linux en procesadores IA-32 y ARM.
  4. V8 puede funcionar de manera individual (standalone) o incorporada a cualquier aplicación C++.

- MongoDB [16]: MongoDB es un sistema de base de datos multiplataforma orientado a documentos, de esquema libre. Al estar escrito en C++ le confiere cierta cercanía a los recursos de hardware de la máquina, de modo que es bastante rápido a la hora de ejecutar sus tareas. Al ser NoSQL, hay que olvidarse de las tablas y las relaciones entre ellas. En MongoDB, cada registro o conjunto de datos se denomina documento. Los documentos se pueden agrupar en colecciones, las cuales se podría decir que son el equivalente a las tablas en una base de datos relacional (sólo que las colecciones pueden almacenar documentos con muy diferentes formatos, en lugar de estar sometidos a un esquema fijo). Es posible crear índices para algunos atributos de los documentos, de modo que MongoDB mantendrá una estructura interna eficiente para el acceso a la información por

[13] Usando Node Package Module o NPM por sus siglas en inglés.
[14] http://code.google.com/p/v8/
[15] Código intermedio entre el código fuente y el código máquina. Es una forma de salida utilizada para reducir la dependencia con respecto al hardware y facilitar su interpretación. En algunos casos existen traductores dinámicos o compiladores tiempo real que traducen el bytecode a código máquina justo antes de ejecutar el programa para mejorar la velocidad. Los bytecode suelen ser interpretados por programas que suelen llamarse máquina virtual[45].
[16] http://www.mongodb.org/

los contenidos de estos atributos [13]. Se abandona el enfoque relacional por bases de datos mas orientadas a objetos y de esta manera es como se procesa la información.

- Kendo [17]: Bajo el lema "El arte del desarrollo web", Kendo UI ofrece un completo abanico de posibilidades, siendo un Framework para la creación dinámica de modernas interfaces se vale de las virtudes de HTML5, CSS3 y jQuery para generar potentes elementos perfectamente compatibles con los navegadores más modernos como también para los dispositivos móviles más utilizados en la actualidad [15]. La compatibilidad de Kendo UI es uno de sus puntos fuertes, ya sea con modernos navegadores o sistemas operativos móviles:

  1. Internet Explorer 7 en adelante
  2. Firefox 3 en adelante
  3. Safari 4 en adelante
  4. Chrome
  5. Opera 10 en adelante
  6. Android 2.0 en adelante
  7. iOS 3.0 en adelante
  8. BlackBerry OS 6.0 en adelante
  9. webOS 2.2 en adelante

- Django [18] (Python): Django es un framework de desarrollo web basado en Python, cuya meta es un "desarrollo rápido, limpio y con un diseño pragmático". De acuerdo a la información otorgada por su sitio web, cuenta con un Object-Relational Mapping [19], que si bien, por defecto está configurada para operar una base de datos SQLite, puede modificarse para trabajar con una base de datos MySQL u otras. Se utiliza tanto para levantar un servidor web, como para crear el código fuente del sitio en construcción. Además cuenta con una gran comunidad de programadores que aportan soluciones a los diversos bugs que aparecen.

[17] http://www.kendoui.com/
[18] https://www.djangoproject.com/
[19] ORM, por sus siglas en inglés

- Ruby on Rails [20] (Ruby): Ruby on Rails es un framework de desarrollo web basado en Ruby. De acuerdo a su sitio web en español[46] corresponde a *"un conjunto de librerías, automatismos y convenciones destinados a resolver los problemas más comunes a la hora de desarrollar una aplicación web, para que el programador pueda concentrarse en los aspectos únicos y diferenciales de su proyecto en lugar de en los problemas recurrentes."*. Maneja el pradigma MVC y su filosofía es "Don't Repeat yourself". Permite, entre otras cosas, combinar leguaje de Ruby con HTML mediante archivos con extensión *html.erb*.

3.5 Evolución

3.5.1 Características

Actualmente se está viviendo otra revolución, términos como web semántica y web 3.0 o computación ubicua con web 4.0 son cada vez más comunes.

Utilizar números para marcar generaciones sucesivas de la web parece una buena idea cuando se comprueba el éxito que tuvo la denominación 2.0. ¿Cuáles serían los rasgos de esta futura web? Aquí se entra en un terreno difícil, puesto que no se trata de algo existente en la actualidad, sino de una especulación acerca de cómo se va encaminado la web ahora y en el futuro cercano. Una forma de solucionar el problema es lo que hacen algunos analistas y que consiste en identificar web 3.0 con Web Semántica. Otros analistas los ven de forma separada, donde esta nueva web tiene las siguientes características:

- Computación en la nube y Bases de datos no relacionales

[20] http://rubyonrails.org/

- Agentes de usuario[21] (como en la Web Semántica)
- Anchura de banda
- Mayor acceso a internet

Si bien los dos últimos puntos son de carácter técnico, sin duda están teniendo repercusiones sociales. A mayor ancho de banda se facilita la ejecución de aplicaciones multimedia, mientras que un mayor acceso a internet no implica sólo que hay mayor cantidad de conexiones, sino que es posible conectarse desde toda clase de aparatos electrónicos, como celulares, tablets e incluso vehículos. Esto a su vez incentiva la creación e investigación de nuevas tecnologías a la hora de desarrollar aplicaciones web, ya sea para estandarizar aspectos entre dispositivos, plataformas, sistemas operativos; o bien para personalizar cada aplicación con el fin de sacarle el máximo provecho en un dispositivo en particular.

De todos modos y en definitiva, de eso trata la web 3.0, de páginas capaces de comunicarse con otras páginas mediante procesamiento de lenguaje natural y, es justo aquí cuando cobra sentido el nexo entre la Web Semántica y la web 3.0. Ésta es la principal interpretación que se hace de éste término.

3.5.2 Tecnologías de desarrollo

Algunas de las tecnologías de reciente aparición (y no tan recientes), de carácter libre y que se están aplicando en el área del desarrollo web, son:

[21] Es la aplicación cliente usada para un protocolo de red particular, generalmente WWW. Cuando se visita un sitio web, una cadena de texto es enviada al servidor con el fin identificar el agente de usuario (user-agent); dicha cadena forma parte de la petición HTTP. En general, el user-agent incluye información acerca del nombre de la aplicación y su versión, el sistema operativo que utiliza y el lenguaje [47]. Para más información, puede leer sobre de los agentes de software [48].

- Web Semantíca: A fines de la década de 1990, comenzó a idearse un nuevo cambio en la Web. Era un cambio a la vez más complejo y más profundo que el que ha representado la web 2.0. Se trataba del proyecto de la Web Semántica. A grandes rasgos, el objetivo fundacional de la Web Semántica consistió en desarrollar una serie de tecnologías que permitieran a las computadoras, a través del uso de agentes de usuarios parecidos a los navegadores actuales, no solo "entender" el contenido de las páginas web, sino además efectuar razonamientos sobre el mismo. La idea era conseguir que el enorme potencial de conocimiento encerrado en documentos como las páginas web pudiera ser interpretado por las computadoras de una forma parecida a como lo haría un ser humano.[14]

- Computación Ubicua: Corresponde al acceso a gran cantidad de información y a su procesamiento de forma independiente a la ubicación de los usuarios. Esto implica la existencia de una gran cantidad de elementos de computación disponibles en un determinado entorno físico y constituidos en redes. Los elementos están empotrados o embebidos en enseres, mobiliario y electrodomésticos comunes y comunicados en red [16]. Se suele asociar a términos como Computación Pervasiva e Inteligencia Ambiental. La base de la Computación Ubicua es: *"la vida no debe pasar atrás de un escritorio"*. Una persona necesita moverse constantemente para desarrollar las diversas tareas que debe completar. Es por ello que, la informática, debe ser considerada parte del ser humano sin que éste se dé cuenta que está ahí para solucionar muchas de sus necesidades, es decir, alcanzar un máximo nivel de trasnparencia para el usuario final [49].

Capítulo 4

Arquitectura

4.1 Definición

Una arquitectura de software corresponde a un patrón de referencia que brinda un marco necesario para guiar la construcción de software y establece la estrucutura de funcionamiento e interacción entre sus diversas partes [20]. Este concepto se refiere a *"la estructuración del sistema que, idealmente, se crea en las etapas tempranas de desarrollo"*[21].

Al crear un software, independientemente de la metodología que se utilice, es necesario cumplir una serie de pasos que preceden a su construcción:

- Requerimientos: Se enfoca a la captura y priorización de necesidades a satisfacer, ya sean de calidad, rendimiento y/o reestrictivas. Estos requerimientos son preponderantes e influencian la desición acerca de que arquitectura utilizar.
- Diseño: Corresponde a la fase central en relación con la arquitectura. Debe satisfacer todos los requerimientos y no solo utilizar tecnologías de moda.

- Documentación: La documentación corresponde a un factor crucial para comunicar un diseño de forma exitosa. Generalemente se utiliza la representación de varias de sus estructuras mediante el uso vistas. Una vista generalmente contiene un diagrama, además de información adicional, que apoya en la comprensión de dicho diagrama.

- Evaluación: Es de suma importancia evaluar el diseño una vez que éste ha sido documentado con el fin de identificar posibles problemas y riesgos. Evaluar (y validar) el diseño antes de codificar, disminuye el costo de corrección de errores.

Los grandes objetivos de una arquitectura de software son :

1. Servir como guía durante el proceso de desarrollo.
2. Definir y satisfacer los atributos de calidad.

Como dice Danny Thorpe:

"Programar sin una arquitectura en mente, es como explorar una gruta sólo con una linterna: no sabes dónde estás, dónde has estado ni hacia dónde vas" [18]

La importancia de contar con una buena arquitectura que se adecue a situaciones reales, donde las diversas herramientas serán probadas. Es por ello que la arquitectura debe ser la respuesta ante un problema, no una imposición. Por otra parte, el desarrollo de software dejó de ser, hace mucho tiempo, el trabajo de una o dos personas, pasando a ser equipos; por tanto es necesario facilitar la comunicación entre sus integrantes.

A la hora de diseñar un software, hay que tener en cuenta que la arquitectura a utilizar se describe utilizando varios tipos de modelos[22]:

- Estructurales: Se centran en la estructura coherente del sistema completo, en lugar de centrarse en su composición. Representan todo como una colección organizada de componentes.

- Frameworks: Identifican patrones de diseño repetibles, los cuales se pueden encontrar en aplicaciones similares.
- Dinámicos: Se centran en los aspectos del comportamiento dinámico de la arquitectura, indicando como la estructura o la configuración del sistema puede cambiar en función de eventos externos.
- De Procesos: Se enfocan en el diseño de los procesos del negocio que el sistema debe soportar.

Dependiendo de la situación, se debe tomar la desición de que tipo de arquitectura utilizar.

De entre las diversas arquitecturas, algunas de las más relevantes son:

- Orientada a Servicios
- Modelo Vista Controlador
- Cliente-Servidor
- Modelo de 3-Capas

4.2 Arquitectura Orientada a Servicios

Es una arquitectura de software que permite la creación y/o cambios de los procesos de negocio desde la perspectiva de tecnologías de la información de forma ágil, a través de la composición de nuevos procesos utilizando las funcionalidades de negocio que están contenidas en la infraestructura de aplicaciones actuales, utilizando protocolos estándar e interfaces convencionales [1] para facilitar el acceso a la lógica de negocios y la información entre diversos servicios[27].

[1] usualmente Web Services

Por lo general, en una empresa coexisten un sin número de aplicaciones, lo que conlleva a una serie de inconvenientes que aumentan el el tiempo y esfuerzo en que se responde a un requerimiento en particular.

Uno de los principales inconvenientes, es que, ante aplicaciones desarrolladas en lenguajes diferentes, no se pueda acceder desde una a otra para consultar un dato en particular[24].

Mediante la aplicación de la Arquitectura Orientada a Servicios (SOA)[2] pretende solucionar los problemas antes mencionados. Dentro de la arquitectura SOA la funcionalidad se implementa en pequeños componentes autónomos reutilizables denominados servicios.

SOA no es software o un lenguaje de programación, sino un marco de trabajo conceptual que permite a organizaciones unir los objetivos de negocio con su infraestructura de tecnologías de información, integrando los datos y la lógica de negocio de sus sistemas separados.[25]

La necesidad de tal marco se deriva de la evolución del software de negocio. Antes, los desarrollos de aplicaciones de negocio se concentraban en necesidades específicas: contabilidad, compras, planillas de sueldos. Cada aplicación se desarrollaba sin considerar a otros sistemas dentro de la empresa, pues las aplicaciones (de pequeña escala), se caracterizaban por ser auto suficientes; el cambio más grande es filosófico, ya que, en lugar de pensar en el diseño de aplicaciones individuales para resolver problemas especificos, SOA ve el software como un patrón que soporta todo el proceso del negocio. Cada elemento de un servicio es un componente que puede ser utilizado muchas veces a través de muchas funciones y procesos dentro y fuera de la empresa

[2]Service Oriented Architecture, por sus siglas en inglés

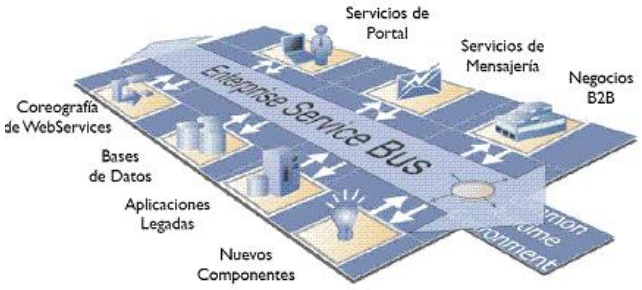


Figura 4.1: Ejemplo de ESB y como conecta diversos servicios. Fuente: [24]

Para lograr esto, se requiere de una infraestructura de comunicación, en este caso SOA utiliza ESB[3], correspondiente a una plataforma, cuya meta es la transformación de datos para conectar y coordinar la interacción de un número significativo de aplicaciones de empresas extendidas con integridad transaccional (Fig. 4.1).

La idea detrás de todo esto es que es más efectivo trabajar con servicios que con aplicaciones.

SOA define las siguientes capas de software [26]:

- Aplicaciones básicas, sistemas desarrollados bajo cualquier arquitectura o tecnología, geográficamente dispersos y bajo cualquier figura de propiedad.
- Exposición de funcionalidades, las cuales son expuestas en forma de servicios (webservices).
- Integración de servicios que facilita el intercambio de datos entre elementos de la capa de aplicaciones orientada a procesos de negocio.
- Composición de procesos, que define el proceso en términos del negocio y sus necesidades, y que varia en función del mismo.
- Entrega, donde los servicios son desplegados a los usuarios finales.

[3]Enterprise Service Bus

4.2.1 Principios SOA

Algunos principios importantes de esta arquitectura son [28][29]:

- **Deben ser reusables**: Los servicios deben ser diseñados y construidos pensando en su reutilización dentro de la misma aplicación, dentro del dominio de aplicaciones de la empresa o incluso dentro del dominio público para su uso masivo.

- **Deben proporcionar un contrato formal**: Los servicios desarrollados deben proporcionar un contrato en el cual figuren: el nombre del servicio, su forma de acceso, las funcionales que ofrece, los datos de entrada de cada una de las funcionalidades y los datos de salida. Así, los consumidores del servicio, accederán a través de las reglas establecidas por el contrato, logrando la independencia entre el consumidor y la implementación del propio servicio.

- **Deben tener bajo acoplamiento**: Los servicios tienen que ser independientes unos de otros. Para lograrlo, es necesario que, cada vez que se vaya a ejecutar un servicio, se accederá a él a través del contrato [4], logrando la independencia entre el servicio que se va a ejecutar y el que lo llama. De esta manera serán totalmente reutilizables.

- **Deben permitir la composición**: Los servicios deben ser construidos de tal manera que puedan ser utilizados para construir servicios genéricos de más alto nivel, el cual estará compuesto de servicios de más bajo nivel. Al trabajar con Servicios Web, se logra mediante el uso de los protocolos para orquestación [5] y coreografía [6] .

- **Deben de ser autónomos**: Los servicios deben tener su propio entorno de ejecución. De esta manera el servicio es totalmente independiente y es posible

[4] Para mayor información sobre lo que es un contrato en este contexto, puede visitar [53]

[5] "En servicios Web, la orquestación se refiere a un conjunto de actividades que conforman un proceso. Orquestación es la gestión, la organización de las interacciones que existen entre los servicios web de un proceso."[30]

[6] "La coreografía corresponde a la descripción y guía de un modelo global y define las colaboraciones entre cualquier tipo de aplicaciones. Es un protocolo de negocio que dicta las reglas de interacción que deben ser cumplidas por las entidades participantes."[30]

asegurar que podrá ser reutilizable desde el punto de vista de la plataforma de ejecución.

- **No deben tener estado**: Los servicios no deben guardar ningún tipo de información. La razón de esto corresponde a que si una aplicación está formada por un conjunto de servicios, y un servicio almacena algún tipo de información, es posible que se produzcan problemas de inconsistencia de datos. Es por ello que es necesario que servicio sólo contenga lógica, y que toda información esté almacenada en algún sistema independiente.

- **Deben poder ser descubiertos**: Los servicios deben poder ser descubiertos de alguna forma para que puedan ser utilizados, y evitar la creación de multiples servicios que proporcionen las mismas funcionalidades.

4.3 Arquitectura Modelo Vista Controlador (MVC)

Es un patrón de arquitectura de software que separa la lógica de negocio de la interfaz de usuario, facilitando la evolución por separado de ambos aspectos e incrementa reutilización y flexibilidad [31].

Su principal característica es que separa los datos de la aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Es encontrado frecuentemente en aplicaciones web, donde la vista es la página HTML (que vé el usuario) y el código es el que provee de datos dinámicos a la página; el modelo que corresponde al sistema de gestión de base de datos y lógica de negocio; y el controlador que es el responsable de recibir los eventos de entrada desde la vista.

MVC fue introducido inicialmente en la comunidad de desarrolladores de Smalltalk-80 [32].

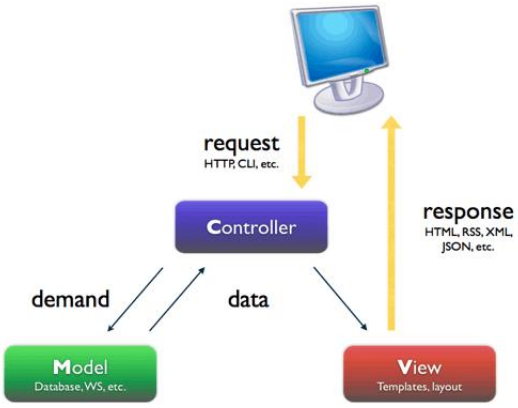


Figura 4.2: Separación de Modelo-Vista-Controlador. Fuente: [50]

4.3.1 Componentes del MVC

La finalidad del modelo es mejorar la reusabilidad por medio del desacople entre la vista y el modelo (Fig. 4.2). Los elementos de esta arquitectura son [33]:

1. **El Modelo**: Encapsula los datos y las funcionalidades. Es responsable de:

   - Acceder a la capa de almacenamiento de datos.
   - Definir las reglas de negocio, es decir, la funcionalidad del sistema.
   - Llevar un registro, tanto de las vistas como de los controladores del sistema.

   El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada

2. **La Vista**: Muestra la información al usuario y pueden existir múltiples vistas del modelo. Es responsable de:

   - Recibir los datos del modelo y mostrarlos al usuario.
   - Tener un registro de su controlador asociado.
   - Suministrar un servicio de "Actualización()", para que sea invocado por el controlador o por el modelo.

   Cada vista tiene asociado un componente controlador.

3. **El Controlador**: Recibe las entradas, usualmente como eventos codificados. Es responsable de:

   - Recibir los eventos de entrada, que por lo general son un click, un cambio dentro de un campo de texto, o el movimiento del ratón (entre otros).
   - Contener reglas de gestión de eventos del tipo "IF", por ejemplo: "IF evento X, THEN acción Y"

   Los eventos son traducidos a solicitudes de servicio ("service requests") para el modelo o la vista.

4.3.2 Flujo del MVC

El flujo que sigue el control generalmente es el siguiente [33], el cual puede cambiar dependiendo de la implementación del:

1. El **usuario** interactúa con la interfaz de usuario, es decir la **vista**, a través de un evento: por ejemplo, el usuario hace click en un enlace).

2. El **controlador** recibe a través de la **vista** la notificación de la acción solicitada por el **usuario**.

3. El **controlador** gestiona el evento que llega a través de un gestor de eventos o callback (generalmente).

4. El **controlador** accede al **modelo**, y lo actualiza de forma adecuada a la acción solicitada por el **usuario**. Por ejemplo, el **controlador** actualiza el carrito de compras del **usuario**).

5. El **controlador** delega a la **vista** la tarea de desplegar la interfaz usuaria. La **vista** obtiene sus datos del **modelo** para generar la interfaz apropiada para el **usuario** donde se refleja los cambios en el **modelo**. Por ejemplo, producir un listado del contenido del carrito de compras.

6. La interfaz usuaria (**vista**) espera nuevas interacciones del usuario, comenzando el ciclo nuevamente

4.4 Arquitectura Cliente-Servidor

IBM define al modelo Cliente-Servidor como[34][35][36]:

"Es la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo y/o, a través de la organización, en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o "clientes", resultan en un trabajo realizado por otros computadores llamados servidores".

Es una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información de forma transparente aún en entornos multiplataforma. Se trata de la arquitectura más extendida en la implementación de Sistemas Distribuidos[37].

Es un modelo para el desarrollo de sistemas de información en el que las operaciones se dividen en procesos independientes y cooperativos entre sí para intercambiar ya sea información, servicios o recursos. Se denomina cliente al proceso que inicia la "conversación", solicita los recursos; y servidor al proceso que responde a dichas solicitudes.

"En este modelo las aplicaciones se dividen de forma que el servidor contiene la parte que debe ser compartida por varios usuarios, y en el cliente permanece sólo lo particular de cada usuario"[36].

4.4.1 Cliente

Es el que inicia un requerimiento de servicio, formulando los requerimientos y pasandolos al servidor; se lo conoce con el término front-end. [34]. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente.

El cliente maneja, por lo general, todas las funciones que tienen que ver con la manipulación y visualización de datos, por ende están desarrolladas sobre plataformas que permiten construir interfaces gráficas de usuario (GUI)[7]

Dentro de las tareas realizadas por el cliente, destacan[37]:

- Administrar la GUI.
- Interactuar con el usuario.
- Procesar la lógica de la aplicación y, en algunos casos, hacer validaciones locales.
- Generar requerimientos de bases de datos.
- Recibir resultados del servidor y mostrarlos al cliente.

4.4.2 Servidor

Es una aplicación que ofrece un servicio a usuarios de Internet, el servidor es un programa que recibe una solicitud, realiza el servicio requerido y devuelve los resultados en forma de una respuesta. Generalmente un servidor puede tratar múltiples peticiones (múltiples clientes) al mismo tiempo.

[7]Graphical User Interface, por sus siglas en inglés.

4.4.3 Principios Cliente-Servidor

Algunas características de este tipo de arquitectura son[36][37]:

- El servidor proporciona los servicios y el cliente los utiliza.
- La única relación entre clientes y servidores es la que se establece a través del intercambio de mensajes. El mensaje es el mecanismo para la petición y entrega de solicitudes de servicio.
- El uso de recursos compartidos. Muchos clientes pueden usar los mismos servidores y, a través de ellos, comparten tanto recursos lógicos como físicos. Estos recursos son regulados por el servidor.
- El cliente inicia las comunicaciones, el servidor espera de forma pasiva.
- Los detalles de la implementación son invisibles para el cliente, es decir que existe encapsulamiento de servicios.
- Transparencia de física y de localización, es decir que el cliente no sabe si el servidor está distribuido y/o su localización física.
- Los programas y datos centralizados en servidores, facilitan tanto la integridad, como su mantenimiento.
- Ambiente heterogéneo, tanto la plataforma de hardware, como el sistema operativo del cliente y del servidor no son necesariamente la misma. Esto representa una de las principales ventajas de esta arquitectura, permitiendo conectar clientes y servidores independientemente de sus plataformas.

4.4.4 Flujo de cliente servidor

Si bien pueden haber ciertas diferencias dependiendo de la implementación, el flujo de funcionamiento de un sistema Cliente/Servidor es [37]:

1. El **cliente** solicita información al **servidor**.

2. El **servidor** recibe la petición del **cliente**.

3. El **servidor** procesa dicha solicitud.

4. El **servidor** envía el resultado obtenido al **cliente**.

5. El **cliente** recibe el resultado y lo procesa.

4.5 Arquitectura "3-Capas"

Una arquitectura bastante popular es la de 3 capas, cuyo objetivo primordial es la separación de la aplicación en una capa de presentación, de negocios y de datos.

Es importante recalcar que **las capas inferiores otorgan sus servicios a las capas superiores**, sin importar su **nivel**. Por ejemplo, si se piensa que la capa de presentación interactua con la capa de negocio, significa que ésta presenta una "interfaz" para brindar servicios a la capa de presentación.

Antes de continuar, es necesario aclarar la diferencia entre los términos **capa** y **nivel**. El primero se utiliza para hacer referencia a las distintas partes en que una aplicación se dividide desde un punto de vista lógico; mientras que el segundo hace referencia a la forma física en que se organiza una aplicación[8].

Una capa solamente debiese utilizar lo que la "interfaz" de la capa inferior le brinda, de este modo es posible intercambiar las capas respetando dicha "interfaz", sin mayores cambios en el código.

[8]En [51] puede encontrar un ejemplo detallado sobre esta diferencia.

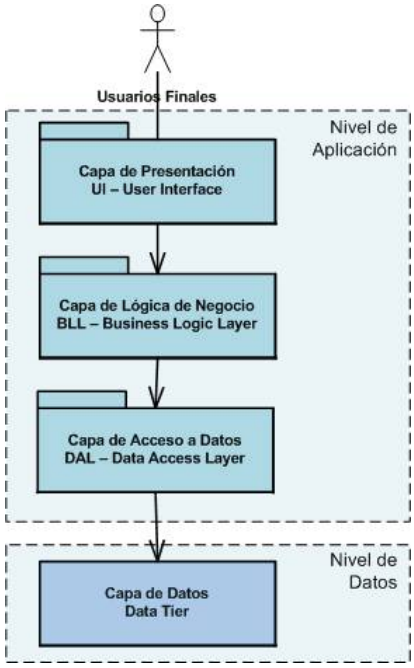


Figura 4.3: Arquitectura 3-Capas 2-Niveles; cada capa accede a la "interfaz" otorgada por la capa inferior. Fuente: [51]

4.5.1 Componentes de la arquitectura

Comenzando desde las capas inferiores, los componentes de esta arquitectura son[23][51]:

1. Una Capa de Datos: Es la encargada acceder a los datos y está formada por uno, o más, gestores de base de datos. Dicho gestor recibe las peticiones de almacenamiento, recuperación, actualización y/o eliminación de datos; desde la Capa de Acceso a Datos (Fig 4.3). Esta capa por lo general, se encuentra en un nivel independiente.

2. Una Capa de Acceso a Datos: Denominada a veces como "Capa de Persistencia", no es lo mismo que Capa de Datos. Como su nombre lo indica, realiza el acceso a los datos y la ventaja de su uso radica en que si se requiere cambiar el motor de base de datos, se debe corregir sólo esta capa.

3. Una Capa Lógica/de Negocios: Es la encargada de procesar y responder las peticiones que recibe, tanto del usuario a través de la capa de presentación, como

las de la Base de Datos mediante la Capa de Acceso a Datos (Fig 4.3). En otras palabras es una capa intermedia.

4. Una Capa de Presentación: Corresponde a la capa que ve el usuario, es decir, le presenta el sistema. Se responsabiliza de que se le comunique información al usuario por parte del sistema y viceversa. La tarea de capturar información sobre éste se realiza, por lo general, a través de formularios. Esta capa se comunica exclusiamente con la Capa Lógica.

Basicamente el objetivo principal es separar las distintas lógicas de la aplicación en niveles y que éstos posean estructuras bien planteadas.

4.6 Elección

Considerando el hecho de que uno de los objetivos de esta memoria es, realizar pruebas y obtener resultados concretos acerca de diversas herramientas de desarrollo web, es necesario enmarcar dichas pruebas bajo un estándar que asegure una base pareja para todas ellas.

Es por dicha razón que se utilizará una arquitectura de tipo **Estructural**: considerando el hecho de que hay que comparar el mismo sistema en diversas herramientas. y de tipo **Framework**: considerando que se busca un desarrollo rápido sin "reinventar la rueda".

Nota al profesor: Las pruebas realizadas hasta ahora han sido bajo la arquitectura Cliente-Servidor, las cuales constan de llenar de carga al Servidor, mediante consultas en paralelo (Sin tener en cuenta algún motor de BD). No obstante, al comienzo se pensó en una arquitectura 3-Capas, dejando la Capa de Datos constante (MySQL). Mi duda es, ¿Sigo ocupando esta arquitectura? ¿En caso de usar 3-Capas, documento los datos

obtenidos hasta ahora?

Capítulo 5

Criterio de Selección

El criterio de selección de tecnologías a utilizar, es un punto de importancia no menor en el desarrollo de esta memoria, pués consta de la combinación de una serie de herramientas que, dependiendo de cuáles y cómo se combinen, puede repercutir de manera positiva o negativa, el rendimiento del sistema y por lo tanto los resultados de las pruebas a realizar.

Para simplificar la tarea, se ha divido la selección en 2 items:

1. Tecnologías tradicionales.
2. Tecnologías emergentes.

Dentro de cada item se utiliza la siguiente subdivisión:

- Gestor de Base de Datos **(En caso de usar Capa de Datos)**.
- Servidor Web.
- Lenguaje de programación Web y uso de Framework.

5.1 Tecnologías Tradicionales

Para la combinación de tecnologías tradicionales se utilizará:

- Gestor de Base de Datos MySQL 5.5 **(En caso de usar Capa de Datos)**
- Servidor Web Apache
- Lenguaje de Programación PHP y Framework Symfony2[1] o CakePHP2[2]

Esta elección está basaba en la gran popularidad que duró más de una decada, conviertiendose en practicamente una combinación estándar durante la web 2.0.

La razón de utilizar una combinación de carácter relativamente[3] más tradicional, es la de tener una base de comparación respecto a las tecnologías relativamente recientes, y de acuerdo a los posibles resultados de las pruebas a realizar, determinar si las diferencias en cuanto a rendimiento son lo suficientemente altas, para justificar tener que comenzar a considerar el uso de estas nuevas tecnologías, aunque toda desición final dependerá del contexto específico.

La desición de utilizar PHP, se basa en el hecho de que ha sido el rey, prácticamente indiscutido, de los lenguajes tradicionales. Por otro lado soporta gran cantidad de frameworks.

5.1.1 ¿Symfony2 o CakePHP2?

La cantidad de frameworks que existen para PHP es bastante grande, es por ello que su elección no es un tema sencillo de resolver.

[1] http://symfony.com/
[2] http://cakephp.org/
[3] Si bien el servidor, y el lenguaje de programación son tradicionales, se utilizan versiones actualizadas, tanto de motor de base de datos y de frameworks.

Si bien en un princio, la elección tomó en cuenta más de dos variables, la desición se acotó a los más populares dentro de la web. Sin embargo, no hay consenso en los foros y ambos reciben tanto críticas como puntos a favor, por parte de los usuarios.

Es por ello que se realizará una instalación de ambos y se correran ciertas pruebas preliminares, para eliminar todo rastro de dudas[4].

- Symfony: Actualmente rediseñado casi por completo en su versión 2, es un framework diseñado para optimizar el desarrollo de las aplicaciones web, separando la lógica de negocio, la lógica de servidor y la presentación. Proporciona herramientas cuyo fin es reducir el tiempo de desarrollo. Además, automatiza las tareas más comunes (incluyendo algunas de seguridad), permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado: no se debe reinventar la rueda cada vez que se crea una nueva aplicación web. Para más información sobre la versión utilizada en este trabajo, puede visitar [54].

  Hasta acá llega la revision al 28-08-2013

- CakePHP:

En [42] hay una buena comparativa (del 2007), en la que a grandes rasgos se concluye que: cake es mas facil de instalar, pero symfony puede que tenga mejor performance

En [43] hay una cmparativa entre cake 2.1.3, symfony 2.1.2 y Yii, en el que cake supera a symfony (usando siege)

En [44] hay info entre ambos framworks (vschart), una especie de wiki.

[4]Para saber el resultado de estas pruebas, revise el capitulo x.x ***llenar de acuerdo al orden final***

En http://bakery.cakephp.org/articles/view/4cb4609b-1154-4fc8-8275-49e1d13e7814/lang:spa (hora de consulta 1604 13-08-2013) se habla de una prueba en la que ambos son relativamente similares en rendimiento.

En mis pruebas realizadas con el "hola mundo" en ambos frameworks, muestran diferencias críticas, dando amplia ventaja a Symfony2 sobre CakePHP2. No obstante la instalacion de cake resulto mucho más simple y rápida.

5.2 Tecnologías Emergentes

Para la combinación de tecnologías emergentes se utilizará:

- Gestor de Base de Datos SQL
- Servidor Web Nodejs
- Lenguaje de Programación Javascript y Framework Express [5]

La razón para elegir esta combinación, radica principalmente en el hecho de poder utilizar un lenguaje único, tanto para el servidor web como para desarrollar la aplicación web en sí.

5.2.1 ¿Por qué Nodejs?

Nodejs consiste en un framework de Javascript basado en el motor V8.

[5] http://expressjs.com/

La meta número uno declarada de Node es *proporcionar una manera fácil para construir programas de red escalables*. ¿Cuál es el problema con los programas de servidor actuales? En lenguajes como Java y PHP, cada conexión genera un nuevo *thread* que potencialmente viene acompañado de 2 MB de memoria. En un sistema que tiene 8 GB de RAM, esto da un número máximo teórico de conexiones concurrentes de cerca de 4.000 usuarios.

Bajo el supuesto de que el impacto de una empresa ficticia es tal que crece la base de clientes, se desea que la aplicación soporte más y más usuarios, por lo tanto es necesario agregar más y más servidores. Lo que representa una suma en cuanto a costos de servidor, tráfico, y por lo tanto potenciales problemas técnicos: un usuario puede estar usando diferentes servidores para cada solicitud, así que cualquier recurso compartido debe almacenarse en todos los servidores. Por todas estas razones, el cuello de botella en toda la arquitectura de aplicación Web (incluyendo el rendimiento del tráfico, la velocidad de procesador y la velocidad de memoria) corresponde al número máximo de conexiones concurrentes que puede manejar un servidor.

Node resuelve este problema cambiando la forma en que se realiza una conexión con el servidor. En lugar de generar un nuevo hilo de SO[6] para cada conexión y de asignarle la memoria necesaria, cada conexión dispara una ejecución de evento dentro del proceso del motor de Node.

Corre en la máquina virtual de javascript de google conocida como v8 y se basa en el paradigma de programación dirigido a eventos de entrada y salida asincrónica.

Una importante característica es que puede levantar servidores web en casi cualquier puerto que se le indique, además de posibilitar la comunicación única bidirideccional entre el cliente y el servidor[4], mediante el uso de sockets o bien llamadas de tipo

[6]Sistema Operativo

comet [19] [7].

De la misma forma que ocurre al trabajar con Apache, es posible expandir la funcionalidad de Node instalando módulos. Los módulos que se pueden utilizar con Node mejoran en gran medida el producto. Así de importantes se han tornado los módulos, hasta el punto de convertirse en parte esencial del producto completo [10]. Los módulos se instalan usando Node Package Manager.

En lo que a plataformas se refiere y de acuerdo a la página oficial [17], están habilitadas las descargas para:

- Windows
- Mac OS X
- Linux
- SunOS

5.2.2 La lucha entre Django y Rails

Similar a lo que ocurre con los frameworks para PHP, es lo que acontece con Django y Rails, a pesar de que ambos no comparten un lenguaje en común (Python y Ruby respectivamente).

Esta pregunta es, relamente, Python vs Ruby y

this is really a Python vs Ruby question, not Django vs Rails. And, as can be gathered from the other answers to this question, there is no clear winner. Both are good, modern, well-designed, flexible, good for your productivity (unlike PHP which fails on

[7] Una llamada de tipo comet permite a un servidor web enviar información al navegador web si que este se lo pida explicitamente.

all these counts). So choosing between the two really boils down to a fit between your personality/preferences and the language strengths.

5.2.3 ¿Por qué SQL en lugar de NoSQL?

No se utiliza un gestor de Base de Datos NoSQL, ya que el objetivo es medir el comportamiento de la combinación de servidor web y lenguaje de programación, medir cómo manejan los accesos a la base de datos en igualdad de condiciones.

Capítulo 6

Evaluación

En esta sección se analizará en profundidad, como se preparará el ambiente de desarrollo de cada herramienta elegida, ya sea tradicional o emergente.

6.1 Tecnologías Tradicionales

6.1.1 Symfony

Consiste en un framework de PHP, orientado a desarrollo web. Los pasos para instalar y dejar funcional esta herramienta son:

1. Descargar el archivo desde *http://symfony.com/download*. Se recomienda bajar la última versión estable[1].
2. Descomprimir el archivo una vez descargado.
3. Abrir una consola y posicionarse en la carpeta descomprimida.
4. Seguir las instrucciones del archivo README.md:

[1]Hasta la fecha es la versión 2.2.

(a) Ejecutar php composer.phar install. En caso de error ejecutar como root.

(b) Ejecutar el script de chequeo para ver si está todo en orden php app/check.php

(c) En caso de haber errores, consultar en la sección de anexos.

(d) Actualizar el timezone en php.ini ubicado en /etc.

Para crear una aplicación de prueba:

1. Seguir las instrucciones de la documentación en [40].

2. Es posible que se deba mover la carpeta que contenga a la aplicaciónde prueba a la ruta /var/html/www

3. En caso de error de estilo:

```
Change the permissions of the "app/cache/" directory
 so that the web server can write into it.
Change the permissions of the "app/logs/" directory
 so that the web server can write into it.
```

Se puede solucionar mediante [41]:

```
rm -rf app/cache/*
rm -rf app/logs/*
chmod 777 app/cache
chmod 777 app/logs
```

Si bien hay soluciones que cuentan con un mayor nivel de seguridad, se escapan del alcance de esta memoria.

6.1.2 CakePHP

Consiste en un framework de PHP, orientado a desarrollo web. Los pasos para instalar y dejar funcional esta herramienta son:

1. Completar
2. Descargar el archivo desde *http://cakephp.org/*, en "Download" Se recomienda bajar la última versión estable[2].
3. Descomprimir el archivo una vez descargado.

6.1.3 Apache Server

Consiste en el servidor web *...Completar*

6.2 Tecnologías Emergentes

6.2.1 Rails

Consiste en un framework de Ruby, orientado a desarrollo web. Los pasos para instalar y dejar funcional esta herramienta son:

1. Completar
2. Abrir una consola y posicionarse en la carpeta descomprimida.
3. despues de realizar las configuraciones necesarias (con rails new y eso) **modificar y completar este punto**
4. Acceder a app/config dentro de la instalación, y como root ejecutar rails server.

[2]Hasta la fecha es la versión 2.3.x.

6.2.2 Django

Consiste en un framework de Python, orientado a desarrollo web. Los pasos para instalar y dejar funcional esta herramienta son:

1. Descargar el archivo desde *https://www.djangoproject.com/download/*. Se recomienda bajar la última versión estable[3].
2. Descomprimir el archivo una vez descargado.
3. Abrir una consola y posicionarse en la carpeta descomprimida.
4. Ejecutar python setup.py install (como root).
5. Para sincronzar con la base de datos Mysql, utilizando syncdb [4]
   (a) Instalar mysql-python, en Fedora 17 yum install mysql-python.
   (b) Para ver la solucion a posibles errores, puede visitar la sección de anexos.

   Django cuenta con una serie de archivos que se crean de forma automatica a la hora de crear un proyecto, ellos son:

   - **manage.py**: Es una utilidad de linea de comandos que permite al desarrollador interactuar con el proyecto de diversas formas.
   - **__init__.py**: Es un archivo vacio que le dice a Django que, la carpeta en la que se encuentra, debe ser considerada como un paquete.
   - **settings.py**: Es un archivo que contine configuraciones para el proyecto en particular.
   - **url.py**: Es un archivo que posee una "taba de contenidos" de como funcionan las URL.

   Para consultar información detallada, puede acceder a la documentación de Django [39].

[3] Hasta la fecha es la versión 1.5.1.
[4] Considerando Mysql instalado.

6.2.3 Nodejs

Consiste en un framework de js basado en el motor V8 de Google. Los pasos para instalar y dejar funcional esta herramienta son:

1. Descargar el archivo en *http://nodejs.org/*, install.
2. Una vez descargado, descomprimir y seguir los pasos del archivo "Readme.md".
3. En una consola y posicionandose en la carpeta descomprimida, ejecutar:
   (a) ./configure
   (b) make
   (c) make install (pide permisos de root)
4. Es necesario tener gcc instalado, sino aparecerá un error.
5. Para ver soluciones a posibles errores, puede visitar la sección de anexos.

Para configurar la herramienta con la Base de Datos: *agregar informacion cuando se realice*

6.3 Motor de Base de Datos

Para instalar y configurar el motor de Base de Datos Mysql (común para todas las herramientas) en Fedora 17 [38], es necesario:

1. Abrir una consola y acceder como usuario root.
2. Ejecutar yum install mysql-server.
3. Una vez instalado, ejecutar systemctl start mysqld.service.

4. Ejecutar systemctl enable mysqld.service para que inicie el servicio al iniciar el SO [5].

5. Conectarse a mysql a través de **mysql -u root** y:

   (a) Identificar la información inicial de la Base de Datos:

```
    mysql> select user,host,password from mysql.user;
+------+-----------+-------------------------------------------+
| user | host      | password                                  |
+------+-----------+-------------------------------------------+
| root | localhost |                                           |
| root | tarrito   |                                           |
| root | 127.0.0.1 |                                           |
| root | ::1       |                                           |
|      | localhost |                                           |
|      | tarrito   |                                           |
+------+-----------+-------------------------------------------+
```

   (b) Establecer contraseñas:

```
mysql> set password for root@localhost=password('******');
mysql> set password for root@'tarrito'=password('******');
mysql> set password for root@'127.0.0.1'=password('******');
```

   (c) El password no necesariamente debe ser el mismo de root.

   (d) Y verificando la información:

```
mysql> select user,host,password from mysql.user;
+------+-----------+-------------------------------------------+
```

[5] Sistema Operativo

```
| user | host      | password                                  |
+------+-----------+-------------------------------------------+
| root | localhost | *7C65DFE6EA91038267275D4DB8D9C16DAFCBD3F8 |
| root | tarrito   | *7C65DFE6EA91038267275D4DB8D9C16DAFCBD3F8 |
| root | 127.0.0.1 | *7C65DFE6EA91038267275D4DB8D9C16DAFCBD3F8 |
| root | ::1       |                                           |
|      | localhost |                                           |
|      | tarrito   |                                           |
+------+-----------+-------------------------------------------+
```

Capítulo 7

Pruebas

agregar parrafo introductorio...

7.1 Herramientas para realizar pruebas

Las herramientas para medir los parametros mencionados son:

- **ab:**
- **siege:**
- **httperf:**
- **jmeter:**

De ellas, jmeter tiene la ventaja de utilizarce con plugins, que permiten la obtención de gráficos.

Capítulo 8

Resultados

Capítulo 9

Conclusiones

Capítulo 10

Bibliografía

[1]http://www.slideshare.net/Mayrasaquinga/e-learning-12831321 pagina 3 (hora de consulta 14:54 23/08/2013)

[2]Propuestas metodológicas para el desarrollo de aplicaciones Web: una evaluación según la ingeniería de métodos M. Mendoza* y J. Barrios** Postgrado en Computación, Universidad de Los Andes, Facultad de Ingeniería, Escuela de Ingeniería de Sistemas, Mérida, Venezuela. *nella_1@hotmail.com, **ijudith@ing.ula.ve Revista Ciencia e Ingeniería. Vol. 25 No. 2. 2004

[3]http://www.koala-soft.com/de-web-10-a-web-30 (hora de consulta 17:00 17/10/2012)

[4] http://www.quantium.com.mx/2012/08/06/nodejs/ (hora de consulta 18:18 28/11/2012)

[5]La evolución de los servicios de referencia digitales en la Web 2.0 Catuxa Seoane García catuxa@gmail.com Documentalista. Bibliotecas Municipales de A Coruña Vanesa Barrero Robledo uveybe@gmail.com Documentalista. Yahoo España Autoras del blog Deakialli DocuMental (http://www.deakialli.com)

[6]http://www.oreillynet.com/oreilly/tim/news/2005/09/30/what-is-web-20.html (hora de consulta 18:10 17/10/2012)

[7]http://www.ecured.cu/index.php/Qooxdoo (hora de consulta 02:45 22/10/2012)

[8]http://www.desarrolloweb.com/articulos/que-es-html.html (hora de consulta 22:37 17/10/2012)

[9]http://www.monografias.com/trabajos7/html/html.shtml (hora de consulta 22:44 17/10/2012)

[10] https://npmjs.org (hora de consulta 18:10 28/11/2012)

[11]http://www.monografias.com/trabajos62/sistemas-informacion-web/sistemas- informacion-web2.shtml (hora de consulta 01:59 18/10/2012)

[12]Procesos Ágiles Para el Desarrollo de Aplicaciones Web Paloma Cáceres, Esperanza Marcos Grupo Kybele Departamento de Ciencias Experimentales e Ingeniería Universidad Rey Juan Carlos C/ Tulipán, s/n, 28933 – Móstoles, Madrid (España) p.caceres/cuca@escet.urjc.es

[13]http://www.genbetadev.com/bases-de-datos/una-introduccion-a-mongodb (hora de consulta 03:29 22/10/2012)

[14]I Congreso Internacional de Ciberperiodismo y Web 2.0. Bilbao: Noviembre 2009 ¿Web 2.0, Web 3.0 o Web Semántica?: El impacto en los sistemas de información de la Web Por Lluís Codina Universidad Pompeu Fabra www.lluiscodina.com — www.lluiscodina.com/diagramas Página 4

[15]http://www.kabytes.com/programacion/kendo-ui-framework-html5-css3-y- jquery/ (hora de consulta 03:35 22/10/2012)

[16]http://interfacemindbraincomputer.wetpaint.com/page/2.A.1.1.7.-Computacion+ Ubicua+o+Pervasiva+e+Inteligencia+ambiental (hora de consulta 21:22 24/10/2012)

[17] http://nodejs.org/download/ (hora de consulta 19:10 29/11/2012)

[18] http://slidesha.re/Yl7RJH (hora de consulta 15:39 10/04/2013)

[19] http://book.mixu.net/ch13.html (hora de consulta 16:55 05/04/2013)

[20] http://www.ecured.cu/index.php/Arquitectura_de_software (hora de consulta 16:32 07/03/2013)

[21] http://sg.com.mx/content/view/922 (hora de consulta 16:42 07/03/2013)

[22] http://bit.ly/11CxHEO (hora de consulta 16:02 04/06/2013)

[23] http://davidjguru.com/2010/02/08/la-arquitectura-de-tres-capas-introduccion/ (hora de consulta 13:46 04/06/2013)

[24] http://bit.ly/11GTGe3 (hora de consulta 15:12 17/06/2013)

[25] http://bit.ly/ZxL9OX (hora de consulta 15:24 17/06/2013)

[26] http://bit.ly/19dWxRZ (hora de consulta 15:32 17/06/2013)

[27] http://bit.ly/11sQ1VT (hora de consulta 16:35 21/06/2013)

[28] http://bit.ly/1amdWtj (hora de consulta 15:56 17/06/2013)

[29] http://bit.ly/14nHu5g (hora de consulta 17:03 21/06/2013)

[30] http://slidesha.re/1aImbQs (hora de consulta 15:52 24/06/2013)

[31] http://bit.ly/10eAXul(hora de consulta 16:05 17/06/2013)

[32] http://bit.ly/l8dMhw (hora de consulta 16:13 17/06/2013)

[33] http://bit.ly/11k1vuJ (hora de consulta 16:35 17/06/2013)

[34] http://bit.ly/16zWr6c (hora de consulta 13:32 17/06/2013)

[35] http://bit.ly/18bQ5gI (hora de consulta 17:45 24/06/2013)

[36] http://bit.ly/19mazo5 (hora de consulta 17:41 24/06/2013)

[37] http://bit.ly/11KJzG5 (hora de consulta 14:19 17/06/2013)

[38] http://bit.ly/10YgYzZ (hora de consulta 16:45 23/05/2013)

[39] https://docs.djangoproject.com/en/1.5/intro/tutorial01/ (hora de consulta 17:04 25/06/2013)

[40] http://symfony.com/doc/2.2/book/page_creation.html (hora de consulta 17:55 25/06/2013)

[41] http://stackoverflow.com/questions/11475794/permissions-issues-on-symfony2 (hora de consulta 16:42 23/05/2013)

[42] http://www.dustinweber.com/main-page/php-web-frameworks-cakephp-versus-symfony/ (hora de consulta 15:46 13/08/2013)

[43] http://jorgonor.blogspot.com/2012/09/php-frameworks-benchmark.html (hora de consulta 15:56 17/06/2013)

[44] http://vschart.com/compare/cakephp/vs/symfony (hora de consulta 14:34 14/06/2013)

[45] http://www.alegsa.com.ar/Dic/bytecode.php (hora de consulta 15:03 26/08/2013)

[46] http://www.rubyonrails.org.es/ (hora de consulta 16:08 26/08/2013)

[47] http://bit.ly/14yM1F0 (hora de consulta 17:12 26/08/2013)

[48] http://bit.ly/14WyQKr (hora de consulta 17:17 26/08/2013)

[49] http://computacionpervasiva.blogspot.com/ (hora de consulta 17:33 26/08/2013)

[50] http://bit.ly/1chgZD1 (hora de consulta 13:10 28/08/2013)

[51] http://bit.ly/14njijm (hora de consulta 13:12 21/06/2013)

[52] http://bit.ly/19ZKax7 (hora de consulta 15:12 28/08/2013)

[53] http://bit.ly/6lCJhm (hora de consulta 17:01 28/08/2013)

[54] http://librosweb.es/symfony_2_2/ (hora de consulta 19:38 28/08/2013)

Apéndice A

Apéndice

Apéndice ...