Universidad de las Ciencias Informáticas
Facultad 3

Título: Propuesta de una arquitectura en PHP para el sistema ERP cubano

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Orestes de la Fé Concepción

Javier Vélez Pérez

Tutor: Ing. Amado Alburquerque Arias

Junio 2008

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos 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 _____.

Orestes de la Fé Concepción **Javier Vélez Pérez**

_____ _____

Firma del Autor Firma del Autor

Ing. Amado Alburquerque

_____

Firma del Tutor

AGRADECIMIENTOS

Agradecemos a todo aquel que de una forma u otra ayudó en la investigación.

DEDICATORIA

Este trabajo esta dedicado a todas aquellas personas que confiaron en nosotros y nos apoyaron para alcanzar las metas propuestas.

RESUMEN

Este trabajo es una propuesta arquitectónica para el sistema ERP cubano. Dicho software va ser desarrollado en la UCI por el centro FAR en conjunto a otras facultades de la Universidad. Para el desarrollo del mismo se usarán las herramientas que tributen a la soberanía tecnológica, dicho en otras palabras software libre ya que el mismo tiene la posibilidad de ser modificado por el usuario a su conveniencia y sus necesidades.

El trabajo está conformado por tres capítulos. En el capítulo uno se hace un estudio de los conceptos fundamentales sobre la arquitectura, los estilos, patrones, framework y otros conceptos claves para el desarrollo de la investigación. El capítulo dos recoge la propuesta arquitectónica en la plataforma PHP, así como la descripción de los elementos fundamentales, las herramientas y tecnología a utilizar. En el último capítulo se hace una evaluación de la arquitectura propuesta por el método de ATAM explicando algunos de los escenarios involucrados en el sistema ERP cubano.

PALABRAS CLAVE

Arquitectura de software, planificación de recursos empresariales, sistema de gestión integral.

INDICE

INTRODUCCIÓN

En la actualidad las empresas están en constante evolución y desarrollo. Para poder lograr triunfar en el mercado necesitan gestionar con eficiencia los procesos que se llevan a cabo en la misma, entre los que encontramos gestionar la cadena de aprovisionamiento y fabricación en toda la empresa, proyectos, finanzas, gestión de recursos humanos, marketing, ventas y procesos de servicio. En este contexto entran a jugar un papel estratégico los sistemas de información, ya que ellos afectan directamente en las decisiones que pueden tomar los directivos de la empresa.

Un sistema de información se define como un *"conjunto de componentes interrelacionados que permiten capturar, procesar, almacenar y distribuir la información para apoyar la toma de decisiones, la coordinación, el análisis y el control en una organización "* [21].

Hoy en día las empresas pueden escoger diferentes caminos para implantar un sistema de información, pero la solución más acertada que se presenta en el panorama mundial, es la utilización de sistemas de Planificación de Recursos Empresariales, conocido por sus siglas en ingles como ERP (*Enterprise Resource Planning*).

ERP es un término industrial para abarcar un conjunto extenso de actividades soportados por una aplicación multi-módulos que ayuda a un manufacturero o a las partes más importantes de la administración empresarial como: compras, mantenimiento, inventarios y proveedores, entre otros. Un ERP también puede incluir módulos para las finanzas y aspectos de recursos humanos de una compañía. Normalmente un ERP se integra con sistema de base de datos relacional. El implementar un ERP puede abarcar considerables análisis de procesos empresariales, entrenamiento, y nuevos procesos de trabajo [21].

En el ámbito mundial podemos encontrar diferentes ERP comerciales, entre los que se destacan: SAP, Oracle y como alternativa de software libre encontramos OpenBravo.

SAP AG (sistemas, aplicaciones y productos en procesamiento de datos) Esta fue creada en Alemania por ingenieros formados en IBM. El primero producto de este fue llamado R/2, basada en una base de dato centralizada con una gran computadora. A principios de los año 90, es rediseñado este sistema con la arquitectura cliente/servido por lo cual se definió con el nombre de R/3, el mismo fue un gran

adelanto para la empresa, ya que llego a ser el tercer proveedor más grande de software en el mundo.

SAP AG en 1999 agregó otras funcionalidades, como: CRM, SCM, automatización de la fuerza de ventas y data warehousing. Hace grandes investigaciones y desarrolla versiones de R/3 (3.1, 4.0 y 4.6), haciendo otras mejoras al sistema, lo cual provoca el lanzamiento de un producto llamado MySAP.COM.

Por otra parte los primeros pasos de OpenBravo fueron a principios de los años 90, gracias a dos profesores en la Escuela de Ingeniería Tencun de la Universidad de Navarra llamados Nicolás Serrano e Ismael Ciordia dieron al desarrollo del sistema de gestión de esa universidad. En el 2001 con el apoyo de Moncho Aguinaga hicieron el Tecnicia con el objetivo de aplicar esta arquitectura a los sistemas de gestión, por lo que posteriormente se le llama OpenBravo.

OpenBravo se ha diseñado sobre la base de una arquitectura revolucionaria que resulta en una manera más eficiente de desarrollar aplicaciones. Por una parte, el modelo MVC (Model, View, Control) facilita el desacoplamiento de las áreas de desarrollo, permitiendo el crecimiento sostenible de la aplicación y una mayor facilidad en el mantenimiento del código.

Además, MDD (Model Driven Development) proporciona una mejor calidad del código al reducir drásticamente la codificación manual, al tiempo que mejora la productividad y eficiencia del desarrollo. Toda la aplicación ha sido construida siguiendo estándares abiertos: J2EE, SQL, JDBC, HTML, CSS, MDD, XML Engine y SQLC para el desarrollo y XML, FOP, PDF, RTF para el intercambio y presentación de datos. El lenguaje de desarrollo es Java y la base de datos tanto Oracle como PostgreSQL [22].

Como parte del desarrollo empresarial que ha experimentado nuestro país, así como el auge de Internet y las comunicaciones, Cuba no se ha quedado atrás en la adquisición de ERP en el exterior, así como la utilización de soluciones empresariales que se asemejan a un ERP.

El polo productivo gestión de recursos de la facultad 3 en conjunto con el centro FAR y otras facultades de la universidad se proponen desarrollar un sistema ERP desarrollado con herramientas de software libre que esté acorde a las necesidades del país y con las cuales aún no cuentan. La

inexistencia de una arquitectura para el proyecto ERP cubano afecta el comienzo del mismo. Por lo que los autores se proponen desarrollar el diseño de la arquitectura del ERP cubano.

Como consecuencia de la situación problémica antes expuesta se deriva el siguiente **problema científico de investigación**: ¿Cómo contribuir en la arquitectura de software del sistema ERP cubano de forma que posibilite la portabilidad, mantenibilidad y seguridad del mismo?

Objeto de estudio:
Arquitectura de software.

Dentro del objeto de estudio el **campo de acción** es la arquitectura de software en PHP.

Objetivo general:
Diseño de la arquitectura de software en la plataforma PHP para desarrollar el sistema ERP cubano.

Objetivos específicos:

- Realizar un estudio del estado del arte de las arquitecturas de software más utilizadas para el desarrollo de sistemas empresariales.
- Identificar patrones y estilos arquitectónicos a utilizar, así como su aplicación para conocer más a fondo sus características y darles una correcta utilización.
- Identificar frameworks.
- Evaluar la arquitectura.

Hipótesis: Sí se diseña la arquitectura de software en la plataforma PHP para desarrollar el sistema ERP cubano, entonces se contribuye a la arquitectura de software.

La investigación se desarrollará a través de las siguientes **tareas de investigación**:

- Sistematización del estado del arte.
- Identificación de patrones y estilos arquitectónicos a utilizar, así como su aplicación para conocer más a fondo sus características y darles una correcta utilización.
- Identificación de frameworks.
- Diseño de la arquitectura en PHP para el sistema ERP.
- Evaluación de la arquitectura usando algún método propuesto por el Instituto de Ingeniería de Software de la Universidad Carniege Mellon.

Toda investigación conlleva a adaptar o mantener una estrategia de investigación, la cual facilita la resolución del problema dado en las investigaciones. Para desarrollar una investigación primeramente se busca información, teorías propuestas por otras personas, datos concretos, etc. que permitirán desarrollar esta investigación y así dar nuestro criterio. Los métodos científicos usados durante la investigación:

Métodos Teóricos

-Histórico

-Lógicos

Métodos Particulares

- Entrevista

El **resultado esperado** de este trabajo será una propuesta de arquitectura en la plataforma PHP para el ERP cubano.

Este trabajo estará estructurado en tres capítulos.

El primer capítulo contemplará la fundamentación teórica del tema, donde se define el marco teórico y del modelo teórico de la investigación, estudio del estado del arte de los principales conceptos abordados durante la investigación. El capitulo dos contendrá la propuesta arquitectónica, con la descripción de los aspectos fundamentales para la solución, y el tercer capítulo la evaluación de la arquitectura propuesta.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capitulo se definirá que es la arquitectura de software, las distintas configuraciones arquitectónicas en PHP, se analizaran los distintos tipos de estilos, patrones arquitectónicos y frameworks, así como los propuestos para el desarrollo de la arquitectura del trabajo. Se explica que es un sistema ERP y los distintos tipos de sistemas que hay en el mundo.

1.2 Introducción a la arquitectura

Una arquitectura de software, también denominada *arquitectura lógica*, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información.

La arquitectura de software establece los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades.

Una arquitectura de software se selecciona y diseña con base en objetivos y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, flexibilidad e interacción con otros sistemas de información.

Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. Algunas arquitecturas son más recomendables de implementar con ciertas tecnologías mientras que otras tecnologías no son aptas para determinadas arquitecturas. Por ejemplo, no es viable emplear una arquitectura de software de tres capas para implementar sistemas en tiempo real.

La arquitectura de software define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación entre ellos. Toda arquitectura debe ser

implementada en una *arquitectura física*, que consiste simplemente en determinar qué computadora tendrá asignada cada tarea.

La arquitectura dirige el desarrollo de los sistemas software y contribuye a que estos se lleven a cabo en los límites establecidos de costos y tiempo, y fundamentalmente debe garantizar que se cumpla con los requisitos funcionales y no funcionales de los usuarios. La arquitectura de software es el resultado del trabajo durante todo el ciclo de vida del proyecto, y principalmente en las primeras iteraciones, de un grupo de trabajo encabezado por el arquitecto (o grupo de arquitectura, en dependencia de las dimensiones del proyecto) [1].

Primeramente hay que partir de las grandes definiciones que existen de la arquitectura:

Clements [1] plantea: "*La arquitectura de software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones"*.

Esta definición es de unos de los grandes investigadores del tema, Clements plantea que la arquitectura es la estructura mas importante de un sistema, puesto que esta encargada de la estructura y relaciones de los componentes de un sistema, la arquitectura representa una vista de abstracción del sistema como un todo. A la hora de realizar una arquitectura hay que tener en cuenta el funcionamiento y la interacción de los componentes, por lo que la estructura debe ser diseñada de acuerdo a los requisitos funcionales y no funcionales, para poder proporcionar una buena confiabilidad, seguridad, integridad, flexibilidad, mantenibilidad y escalabilidad del sistema.

La definición oficial de arquitectura del software que propone la IEEE [3] es:

"*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*."

La arquitectura del software tiene importantes decisiones en cuanto a [1]:

- La organización del sistema software.
- Elementos estructurales del sistema
- La composición de los elementos estructurales.
- El estilo de arquitectura que guía esta organización.

1.3 Niveles de abstracción arquitectónicos

La arquitectura de software se encarga de la descripción y estudio de las propiedades estructurales de los sistemas de software. Sin embargo, el estudio de dichas propiedades puede llevarse a cabo desde diferentes niveles de abstracción.

1.3.1 Estilos arquitectónicos

Para alcanzar resultados satisfactorios en el trabajo arquitectónico se debe hacer una correcta selección del estilo a usar.

Según Shaw y Garlan [4] definen estilo arquitectónico como*:*

"Una familia de sistemas de software en términos de un patrón de organización estructural, que define un vocabulario de componentes y tipos de conectores y un conjunto de restricciones de cómo pueden ser combinadas. Para muchos estilos puede existir uno o más modelos semánticos que especifiquen cómo determinar las propiedades generales del sistema partiendo de las propiedades de sus partes."

Buschmann [5] definen estilo arquitectónico como:

"Una familia de sistemas de software en términos de su organización estructural. Expresa componentes y las relaciones entre estos, con las restricciones de su aplicación y la composición asociada, así como también las reglas para su construcción. Así mismo, se considera como un tipo particular de estructura fundamental para un sistema de software, conjuntamente con un método asociado que especifica cómo construirlo. Éste incluye información acerca de cuándo usar la arquitectura que describe, sus invariantes y especializaciones, así como las consecuencias de su aplicación."

Los estilos generalmente proveen una guía y análisis para crear una clase amplia de arquitecturas en un dominio específico donde los patrones se enfocan en solucionar los problemas más pequeños, más específicos dentro de un estilo dado.

Algunos de los estilos arquitectónicos más difundidos son los que se muestran a continuación:

1. **Tubería y filtros**: Una tubería es una popular arquitectura que conecta componentes computacionales a través de conectores, de modo que el procesamiento de datos se ejecuta como un flujo. Los datos se transportan a través de las tuberías entre los filtros, transformando gradualmente las entradas en salidas. Este estilo se percibe como una serie de transformaciones sobre sucesivas piezas de los datos de entrada.

2. **Estilos centrados en datos**: Esta familia de estilos enfatiza la integración de los datos. Se estima apropiada para sistemas que se fundan en acceso y actualización de datos en estructuras de almacenamiento.

3. **Estilos de llamada y retorno**: El programa se constituye de un programa principal que tiene el control de sistema y varios subprogramas que se comunica mediante este, por el uso de llamadas.

    - **Arquitectura en capas**: En este estilo arquitectónico cada capa proporciona servicios a la capa superior y se sirve de las prestaciones que le brinda la inferior, al dividir un sistema en capas, cada capa puede tratarse de forma independiente, sin tener que conocer los detalles de las demás. La división de un sistema en capas facilita el diseño modular, en la que cada capa encapsula un aspecto concreto del sistema y permite además la construcción de sistemas débilmente acoplados, lo que significa que si se minimiza las dependencias entre capas, resulta más fácil sustituir la implementación de una capa sin afectar al resto del sistema.

    - **Arquitectura orientada a objetos**: Los componentes de este estilo son los objetos, o más bien instancias de los tipos de dato abstractos. En la caracterización clásica de David Garlan y Mary Shaw, los objetos representan una clase de componentes que ellos llaman managers, debido a que son responsables de preservar la integridad de su

propia representación. Un rasgo importante de este aspecto es que la representación interna de un objeto no es accesible desde otros objetos.

- **Arquitectura basada en componentes**: Los sistemas de software basados en componentes se basan en principios definidos por una ingeniería de software específica. Los componentes son las unidades de modelado, diseño e implementación, las interfaces están separadas de las implementaciones, y conjuntamente con sus interacciones son el centro de incumbencias en el diseño arquitectónico. Los componentes soportan algún régimen de introspección, de modo que su funcionalidad y propiedades puedan ser descubiertas y utilizadas en tiempo de ejecución.

4. **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**: Esta arquitectura se conoce como intérpretes basados en tablas ó sistemas basados en reglas. Estos sistemas se representan mediante un seudo-programa a interpretar y una máquina de interpretación. Estas variedades incluyen un extenso espectro que está comprendido desde los llamados lenguajes de alto nivel hasta los paradigmas declarativos no secuenciales de programación.

5. **Estilos peer-to-peer**: Esta familia se conoce también como 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.

    - **Arquitecturas basadas en eventos**: Estas se han llamado también arquitectura de invocación implícita, estas se vinculan con sistemas basados publicación-suscripción. La idea dominante en la invocación implícita es que, en lugar de invocar un procedimiento en forma directa un componente puede anunciar mediante difusión uno o más eventos.

- **Arquitecturas orientadas a servicios (SOA en inglés)**: Esta construye toda la topología de la aplicación como una topología de interfaces, implementaciones y llamados a interfaces; es una relación entre servicios y consumidores de servicios, ambos lo suficientemente amplios como para representar una función de negocio completa.

- **Arquitecturas basadas en recursos**: Esta define recursos identificables y métodos para acceder y manipular el estado de esos recursos. El caso de referencia es nada menos que la World Wide Web, donde los URIs identifican los recursos y HTTP es el protocolo de acceso. El argumento central es que HTTP mismo, con su conjunto mínimo de métodos y su semántica simplísima, es suficientemente general para modelar cualquier dominio de aplicación.

Después de haber visto los estilos mas usados en el mundo de la arquitectura de software es tiempo de pasar a analizar un estrato inferior, en el cual entra a jugar un papel fundamental los patrones arquitectónicos

1.4 Patrones de arquitectura

Buschmann [5] define patrón como una regla que consta de tres partes, la cual expresa una relación entre un contexto, un problema y una solución.

En líneas generales, un patrón sigue el siguiente esquema:

1. Contexto: Es una situación de diseño en la que aparece un problema de diseño.
2. Problema: Es un conjunto de fuerzas que aparecen repetidamente en el contexto.
3. Solución: Es una configuración que equilibra estas fuerzas. Esta abarca:
    - Estructura con componentes y relaciones
    - Comportamiento a tiempo de ejecución: aspectos dinámicos de la solución, como la colaboración entre componentes, la comunicación entre ellos, etc.

Los patrones pueden clasificarse en:

Patrones de arquitectura: Relacionados a la interacción de objetos dentro o entre niveles arquitectónicos Problemas arquitectónicos, adaptabilidad a requerimientos cambiantes, performance, modularidad, acoplamiento.

Patrones de diseño: Que fueron construidos dado los problemas con la claridad de diseño, multiplicación de clases, adaptabilidad a requerimientos cambiantes, proponiendo como solución: comportamiento de factoría, Clase-Responsabilidad-Contrato (CRC).

Patrones de análisis: Usualmente específicos de aplicación.

Patrones de proceso o de organización: Que tratan todo lo concerniente con el desarrollo o procesos de administración de proyectos, o técnicas, o estructuras de organización.

Patrones de idioma: Que reglan la nomenclatura en la cual se escriben, se diseñan y desarrollan nuestros sistemas.

Algunos de los patrones arquitectónicos más usados son [5]:

Layers: Ayuda a estructurar aplicaciones que puede estar descompuestas en grupos de subtareas en las cuales cada grupo esta en un nivel particular de abstracción.

Pipes and filtres: Modelan una estructura para los sistemas que procesan una corriente de datos. Cada fase de elaboración es encapsulada en un filtro. Los datos por tuberías entre filtros adyacentes. Combinando los filtros se pueden crear familias de sistemas relacionados.

Blackboard: Es útil para problemas e los cuales ninguna de las estrategias deterministas de solución son conocidas. En blackboard varios subsistemas especializados instrumentan su conocimiento para fortalecer una solución posiblemente parcial o aproximada.

Broker: Puede usarse para estructurar sistemas distribuidos del software con componentes desacoplados que interactúan por invocaciones remotas de servicio. Un componente del agente es responsable de coordinar comunicación, algo semejante como reenviar demandas múltiples, como para excepciones y resultados transmisores.

Modelo Vista Controlador (MVC): Divide una aplicación interactiva en tres componentes. El modelo contiene los datos y funcionalidades de fondo. La vista exhibe información para el usuario. Los controladores manipulan las entradas del usuario. La vista y el controlador conjuntamente comprenden la interfaz de usuario. Un mecanismo de propagación de cambio asegura consistencia entre la interfaz de usuario y el modelo.

Presentation-Abstraction-Control (PAC): Define una estructura para los sistemas interactivos del software en forma de una jerarquía de agentes cooperadores. Cada agente es responsable de un aspecto específico de funcionalidad de la aplicación y consta de tres componentes: La presentación, la abstracción, y el control. Esta subdivisión separa los aspectos de interfaz persona-ordenador del agente de su corazón funcional y su comunicación con otros agentes.

Microkernel: Se aplica a los sistemas del software que deben poder adaptarse a requisitos de sistema cambiantes. Separa un corazón funcional mínimo de funcionalidad extendida y partes creadas por el usuario. El microkernel también sirve de un conector para enchufar estas extensiones y coordinar su colaboración.

Reflection: Provee un mecanismo para el comportamiento y estructura cambiante de sistemas del software dinámicamente. Soporta la modificación de aspectos fundamentales, como las estructuras de tipo y los mecanismos de llamada de función. En este patrón, una aplicación es dividida en dos partes. Un meta nivel provee información acerca de propiedades seleccionadas de sistema y hace el software consciente de sí mismo. Un nivel de base incluye la lógica aplicativa. Su implementación se fundamenta en la meta nivel.

1.5 Patrones de diseño

Los patrones son unas herramientas potentes ya que unos de su objetivo fundamental es que sean reutilizados en el contexto donde el mismo se preste, puesto que cada patrón tiene un objetivo especifico a resolver. La reutilización nos permite: Reducción de tiempos, disminución del esfuerzo de mantenimiento, eficiencia, consistencia, fiabilidad y protección de la inversión en desarrollos.

Christopher Alexander [6] hace una breve definición desde su punto de visita que es un patrón:

“Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo ni siquiera dos veces de la misma forma.”

En la actualidad y a medida que pasan los años, la humanidad requiere de software mas sofisticados, complejos y mucho más grande. El desarrollo de software es una tarea bastante complicada, depende de la experiencia o capacidad de las personas que estén inmersos en él.

Los patrones de diseño hacen más fácil reutilizar con éxito los diseños y arquitecturas, ayudan a los diseñadores a reutilizar con éxito diseños para obtener nuevos diseños.

Estos patrones tiene una series elementos que los caracterizan:

- El nombre del patrón, describe el problema de diseño, su solución, y consecuencias en una o dos palabras. Tener un vocabulario de patrones nos permite hablar sobre ellos.

- El problema describe cuando aplicar el patrón. Se explica el problema y su contexto. Puede describir estructuras de clases u objetos que son sintomáticas de un diseño inflexible. Se incluye una lista de condiciones.

- La solución describe los elementos que forma el diseño, sus relaciones, responsabilidades y colaboraciones. No se describe un diseño particular. Un patrón es una plantilla.

- Las consecuencias son los resultados de aplicar el patrón.

Cada patrón nos permite el encapsulamiento y abstracción**,** encapsula un problema bien definido y su solución en un dominio particular. Extensión y variabilidad, debería ser abierto por extensión o parametrización por otros patrones, de tal forma que pueden aplicarse juntos para solución un gran problema. Generatividad y composición, una vez aplicado genera un contexto resultante, el que concuerda con el contexto inicial de uno o más de uno de los patrones del catálogo. Equilibrio, debe realizar algún tipo de balance entre sus efectos y restricciones.

Los patrones de diseños se clasifican en 3 estilos diferentes según su utilización: creación, estructurales, comportamiento.

1. **Creación:** es la encargada de crear objetos cuando sus creaciones requieren tomar decisiones.

2. **Estructurales:** Describen la forma en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros.

3. **Comportamiento:** organiza, maneja y combina comportamientos.

Cada patrón de estos presenta una gama de patrones, que los encapsula, cumpliendo con las características de estos patrones.

Creación	**Estructurales**	**Comportamiento**
▪ Abstract Factory ▪ Builder ▪ Factory Method ▪ Prototype ▪ Singleton	▪ Adapter ▪ Bridge ▪ Composite ▪ Decorator ▪ Facade ▪ Flyweight ▪ Proxy	▪ Chain of Responsability ▪ Command ▪ Interpreter ▪ Iterator ▪ Mediator ▪ Memento ▪ Observer ▪ State ▪ Strategy ▪ Template Method ▪ Visitor

Los patrones son una gran familia que permiten el ahorro del tiempo, así como ser exactos a la hora de darle respuesta a un problema determinado, si cada patrón se emplea en el contexto que él requiere o sea para lo cual fue destinado, la eficiencia y el resultado a obtener será elevado. En la actualidad, cada vez se hacen más necesario el uso de estos patrones, por la complejidad que va obteniendo el software y las necesidades del hombre, que cada vez necesita que el software sea más eficiente.

1.6 Arquitecturas más comunes

Generalmente, no es necesario inventar una nueva arquitectura de software para cada sistema de información. Lo habitual es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para cada caso en concreto. Así, las arquitecturas más universales son [5]:

- **Cliente-servidor**: Donde el software reparte su carga de cómputo en dos partes independientes pero sin reparto claro de funciones.

- **Arquitectura en tres capas**: Especialización de la arquitectura cliente-servidor donde la carga se divide en tres capas con un reparto claro de funciones: una capa para la presentación (interfaz de usuario), otra para el cálculo (donde se encuentra modelado el negocio) y otra para el almacenamiento (persistencia). Una capa solamente tiene relación con la siguiente.

- **Orientada a servicios**

Arquitectura cliente-servidor

La arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico.

Ventajas de la arquitectura cliente-servidor

- Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema.
- Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado.

El servidor del cliente es la arquitectura de red que separa al cliente (a menudo se utiliza una interfaz gráfica) de un servidor. Cada caso del software del cliente puede enviar peticiones a un servidor. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores terminales, y los servidores de correo. Mientras que sus propósitos varían algo, la arquitectura básica sigue siendo igual.

Arquitectura en tres capas

La ventaja principal de este estilo, es que el desarrollo se puede llevar a cabo en varios niveles y en caso de algún cambio solo se ataca al nivel requerido sin tener que revisar entre código mezclado.

La programación por capas es un estilo de programación en la que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño, un ejemplo básico de esto es separar la capa de datos de la capa de presentación al usuario. En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten).

- **Capa de presentación:** Es la que ve el usuario, presenta el sistema al mismo, le comunica la información y captura la información del usuario dando un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio.

- **Capa de negocio:** Es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) pues es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa recibe solicitudes de la capa de presentación, y presenta los resultados, solicitando a la capa de datos, mediante el gestor de base de datos para almacenar o recuperar datos de él.

- **Capa de acceso a datos:** Es donde residen las clases que se encargan de gestionar todo el acceso a la información contenida en los gestores de bases de datos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Todas estas capas pueden residir en un único ordenador (no es lo típico). Si bien lo más usual es que haya una multitud de ordenadores en donde reside la capa de presentación (son los clientes de la arquitectura cliente/servidor). Las capas de negocio y de datos pueden residir en el mismo ordenador, y si el crecimiento de las necesidades lo aconseja se pueden separar en dos o más ordenadores. Así,

si el tamaño o complejidad de la base de datos aumenta, se puede separar en varios ordenadores los cuales recibirán las peticiones del ordenador en que resida la capa de negocio.

Si por el contrario fuese la complejidad en la capa de negocio lo que obligase a la separación, esta capa de negocio podría residir en uno o más ordenadores que realizarían solicitudes a una única base de datos. En sistemas muy complejos se llega a tener una serie de ordenadores sobre los cuales corre la capa de acceso a datos, y otra serie de ordenadores sobre los cuales corre la base de datos.

Arquitectura orientada a servicios

La arquitectura orientada a servicios (en inglés *Service-Oriented Architecture* o SOA), es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requerimientos de software del usuario.

SOA proporciona una metodología y un marco de trabajo para documentar las capacidades de negocio y puede dar soporte a las actividades de integración y consolidación.

En un ambiente SOA, los nodos de la red hacen disponibles sus recursos a otros participantes en la red como servicios independientes a los que tienen acceso de un modo estandarizado. La mayoría de las definiciones de SOA identifican la utilización de servicios web (empleando SOAP y WSDL) en su implementación, no obstante se puede implementar una SOA utilizando cualquier tecnología basada en servicios.

Al contrario de las arquitecturas orientado a objetos, las SOA están formadas por servicios de aplicación débilmente acoplados y altamente interoperables. Para comunicarse entre sí, estos servicios se basan en una definición formal independiente de la plataforma subyacente y del lenguaje de programación.

1.7 Papel que juegan dentro de la ingeniería de software los estilos y patrones

Los patrones y estilos arquitectónicos, tienen su objetivo fundamental en la ingeniería de software. El mundo siempre esta en constante evolución, por lo que se hace necesario cada vez ser más eficiente y habilidoso a la hora de diseñar una arquitectura. Cuando ya tenemos los problemas identificados y

priorizados, entonces podemos definir que patrón y estilo arquitectónico emplear, pues cada uno de ellos se emplea para resolver un problema específico.

Estos reducen el costo y previene que el sistema se desvíe de su diseño original. Reutilizar soluciones confiables es una de las prácticas fundamentales de éxito en la ingeniería de software. La creación de componentes y conectores reutilizables es una forma de ahorrar el tiempo, el código sea más cómodo de percibir y de manejar, etc., ya que es usual de la reutilización arquitectónica. Un patrón y un estilo son creados para que puedan ser aplicados a una gran variedad de productos con el fin de mejorar o ayudar a asegurar ciertas cualidades deseadas en los mismos.

1.8 Sistemas ERP

Hoy en día hay una serie de definiciones sobre sistemas ERP, muchas muy extensas, otras son pequeñas, pero todas abordan lo mismo, todas llegan a la misma conclusión. Las siglas de ERP significan en inglés, Enterprise Resource Planning y en español, Planificación de Recursos Empresariales, los sistemas ERP integran las diferentes aéreas de la empresa; como recursos humanos, ventas, contabilidad y finanzas, compras, producción, entre otros. Brindando información cruzada e integrada de todos los procesos del negocio.

Los sistemas ERP integran todo lo necesario para el funcionamiento de los procesos de negocio de la empresa. No podemos hablar de un sistema ERP en el momento que tan sólo se integra uno o una pequeña parte de los procesos de negocio. La propia definición de ERP indica la necesidad de "disponibilidad de toda la información para todo el mundo todo el tiempo" [7].

Los sistemas ERP tiene los siguientes son creados para:

- Optimización de los procesos empresariales.
- Acceso a toda la información de forma confiable, precisa y oportuna (integridad de datos).
- La posibilidad de compartir información entre todos los componentes de la organización.
- Eliminación de datos y operaciones innecesarias (o redundantes).
- Reducción de tiempos y de los costes de los procesos (mediante procesos de reingeniería).

Esto conlleva que las empresas sean mas eficientes a la hora de realizar cualquier negocio, proceso, puede obtener una reducción significativa de costos, un aumento de la productividad, planificar y realizar la automatización de sus procesos, así como la integración completa del negocio e incorporar las mejores prácticas mundiales de la industria. Muchas personas dedicadas al estudio del funcionamiento y evolución de los sistemas ERP, plantean que estos sistemas presentan un ciclo de vida, Estos ciclos de vida consisten en los varios estudios por los que un sistema ERP pasa durante su vida dentro de una organización:

- decisión de adopción
- adquisición
- implantación
- uso y mantenimiento
- evolución
- abandono

Estos sistemas se distinguen de cualquier otro software ya que los mismos son sistemas integrados, Modulares y Adaptables.

- ***Integrales:*** Controla los diferentes procesos de la empresa donde todos los departamentos de la empresa se relaciona entre si.

- ***Modulares:*** Una ventaja de los ERP, tanto económica como técnicamente es que la funcionalidad se encuentra dividida en módulos, los cuales pueden instalarse de acuerdo con los requerimientos del cliente.

- ***Adaptables:*** Los sistemas ERP son adaptables, según la necesidad de la empresa, se configuran de los procesos de acuerdo con las salidas que se necesiten de cada uno.

Estos sistemas poseen otras características, como [8]:

- Bases de datos centralizadas.
- Componente de estos Sistemas interactúan entre si consolidando todas las operaciones.
- Los datos se ingresan una sola vez, los mismos deben ser completos consistentes y comunes.

- Las empresas deben modificar algunos de sus procesos para alinearlos con los sistemas ERP. Este proceso se conoce como reingeniería de Procesos, aunque no siempre es necesario.
- Incluyen un conjunto de módulos.
- Presenta un software para cada unidad funcional.
- La tendencia actual es a ofrecer aplicaciones especializadas para determinadas empresas. Es lo que se denomina versiones sectoriales o aplicaciones sectoriales especialmente indicadas o preparadas para determinados procesos de negocio de un sector (los más utilizados).

Existe una metodología para la implantación de un sistema ERP, MSSE se denomina Metodología para la Selección de un Sistema ERP, esta proveer una guía de pasos para ayudar a la selección de un sistema ERP. Para aplicar esta metodología la empresa debe a ver decidido usar un sistema ERP y no otro sistema. Por lo que se considera que la empresa ya allá analizados los procesos y sabe que áreas estarán involucradas e impactadas por el cambio. MSSE se organiza en tres fases las cuales se dividen en actividades:

- Selección del ERP
- Selección del equipo de consultaría
- Presentación y Planificación General del Proyecto

Cada una de estas fases están comprendidas por actividades, las mismas realizan una serie de pasos, que conllevan a realizar una buena implantación de un sistema ERP. Las metodologías de implantación de un sistema ERP en la empresa no siempre son todo lo simples que se desearía, dado que entran en juego múltiples facetas.

Nunca se puede asegurar que una implantación de un sistema ERP va ser exitosa; solamente con un trabajo bien realizado, una correcta aplicación de metodología y aspectos que deben cuidarse antes y durante el proceso de implantación. Durante y después de la implantación de un sistema ERP es conveniente efectuar lo siguiente:

- Definición de resultados a obtener con la implantación de un ERP.
- Definición del modelo de negocio.
- Definición del modelo de gestión.
- Definición de la estrategia de implantación.
- Evaluación de oportunidades para software complementario al producto ERP.

- Alineamiento de la estructura y plataformas tecnológicas.
- Análisis del cambio organizativo.
- Entrega de una visión completa de la solución a implantar.
- Implantación del sistema.
- Controles de calidad.
- Auditoría del entorno técnico y del entorno de desarrollo.

Existen una serie de sistemas ERP, cada uno se implanta en las empresas según sus necesidades, todos quieren estar en la cima del mercado mundial, los mas usados en la actualidad por sus características y resultados a nivel mundial son: SAP, Oracle, Peoplesoft y como una de versión de software libre esta el OpenBravo.

1.9 Estado del arte de PHP como tecnología

PHP es el acrónimo de Hypertext Preprocessor (Preprocesador de Hipertextos), PHP es un lenguaje de programación usado frecuentemente para la creación de contenido de sitios Web, con el se pueden programar las paginas HTML y código fuentes [9].

Permite desarrollar aplicaciones dinámicas sin tener que aprender un nuevo grupo de funciones, lo que agiliza y ahora tiempo de desarrollo. Debido a su diseño también permite crear aplicaciones con una interfaz gráfica para el usuario. PHP en sus etapas iniciales fue programado en Perl, posteriormente, fue diseñado en C, por el programador Danés-Canadiense Rasmus Lerdorf. El 8 de octubre del 2005 fue publicado “Personal Home Page Tools” con un interpretador dando lugar ha PHP/FI. En 1997 dos programadores israelíes de Technion, Zeev Suraski y Andi Gutmans, vuelve a diseñar el analizador gramatical (parser en inglés) donde crearon la base del PHP 3, cambiando el nombre del lenguaje a la forma actual.

En 1999 Suraski y Gutmans reescribieron el código de PHP, produciendo lo que hoy se conoce como Zend Engine o motor Zend. También conformaron a Zend Technologies en Ratmat Gan, Israel. En mayo de 2000 PHP 4 fue lanzado bajo el poder del motor Zend Engine 1.0. El 13 de julio de 2004, PHP 5 fue lanzado, utilizando el motor Zend Engine II (o Zend Engine 2). La versión más reciente de PHP es la 5.1, que incluye el novedoso PDO (Objetos de Información de PHP o PHP Data Objects) y mejoras utilizando las ventajas que provee el nuevo Zend Engine 2.

Son muchas las características de PHP, por facilidades que el mismo brinda, hoy en día hay una gran revolución con el software libre, muchos desarrolladores defensores de lo que se denomina open source, apuestan, tratan de demostrarle al mundo que el software libre es tan potente como el software propietario. PHP es un ejemplo de herramienta potente para el desarrollo de aplicaciones Web, esta posee alta rapidez ejecución, es un lenguaje especialmente diseñado para la realización de aplicaciones Web. Mientras que otros lenguajes son adaptaciones de lenguajes preexistentes, no pensados para Web.

Cuando esta herramienta esta ejecutándose no consume mucha memoria, para ejecutar aplicaciones solamente necesita de software libre, se puede relacionar con muchas bases datos, puede ser instalado en servidores tal como: Windows, con emuladores (apache + php + (MySql / PostgreSql)).Posee muchas funciones predefinidas por lo que ahora el trabajo se hace mas eficiente y cómodo. Puede interactuar con otras tecnologías como Perl, Javascript, Python, etc. Permite mezclar su código dentro de páginas HTML.

1.10 Configuraciones arquitectónicas que existen en PHP

1.10.1 Frameworks

Un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Puede incluir soporte de programas, librerías y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Proporciona una estructura al código y hace que los desarrolladores escriban código mejor, más entendible y mantenible, hace la programación más fácil, convirtiendo complejas funciones en sencillas instrucciones [10].

Un framework permite separar en capas la aplicación:

- La lógica de presentación que administra las interacciones entre el usuario y el software.
- La lógica de dominio o de negocio, que manipula los modelos de datos de acuerdo a los comandos recibidos desde la presentación.
- La lógica de datos que permite el acceso a un agente de almacenamiento persistente u otros.

Un framework representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

Los frameworks son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional. Algunos tipos frameworks desarrollados en PHP:

1.10.1.1 Symfony

Symfony es un completo framework diseñado para optimizar el desarrollo de las aplicaciones Web mediante algunas de sus principales características. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación Web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación Web compleja. Además, automatiza las tareas más comunes, permitiendo el desarrollador por completo para dedicarse a los aspectos específicos de cada aplicación.

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios Web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows. Requiere de una instalación, configuración y líneas de comando, incorpora el patrón MVC, soporta AJAX, plantillas y un gran número de bases de datos [11].

1.10.1.2 CakePHP

CakePHP es un framework de desarrollo de aplicaciones Web escrito en PHP, creado sobre los conceptos de Ruby on Rails. Al igual que Ruby on Rails, CakePHP facilita al usuario la interacción con la base de datos mediante el uso de ActiveRecord. Además hace uso del patrón Modelo Vista Controlador [12].

- Compatible con PHP4 y PHP5.
- CRUD de la base de datos integrado.

- URLs amigables.
- Sistema de plantillas rápido y flexible.
- Helpers para AJAX, Javascript, HTML, forms y más.
- Trabaja en cualquier subdirectorio del sitio.
- Validación integrada.
- Scaffolding de las aplicaciones.
- Listas de Control de Acceso (ACL).
- Saneamiento de Datos.
- Componentes de seguridad y sesión.

1.10.1.3 Kumbia

Plantean que Kumbia surge como una necesidad de disminuir esfuerzo, primeramente se creo para acortar el trabajo que se repetían en un proyecto de un sistema de información, la cual el creador del mismo estaba desarrollando, al ver que el desarrollo de Kumbia fue exitoso, posteriormente le fueron agregando más, funcionalidades. Por lo que su función es reducir el tiempo de desarrollo de una aplicación Web sin producir efectos sobre los programadores [13].

Presenta diversas características:

- Compatible con muchas plataformas.
- Fácil de instalar y configurar.
- Fácil de aprender.
- Listo para aplicaciones comerciales.
- Convención sobre Configuración.
- Simple en la mayor parte de casos pero flexible para adaptarse a casos más complejos.
  - Soportar muchas características de aplicaciones Web actuales.
  - Soportar las prácticas y patrones de programación más productivos y eficientes.
  - Producir aplicaciones fáciles de mantener.
  - Basado en Software Libre.
  - Es compatible con las bases de datos disponibles actuales mas usadas (MySQL, PostgreSQL, Oracle).

El modelo de objetos de Kumbia es utilizado en tres diferentes capas:

- Abstracción de la base de datos.
- Mapeo Objeto-Relacional.
- MVC

1.10.1.4 Fusebox

Fusebox es un framework estándar para el desarrollo de aplicaciones Web. Lo que hace diferente a Fusebox es que divide una aplicación en secciones o sea plantea que una aplicación Web esta comprendida por secciones, mas bien le llama (circuits o circuitos en el argot de Fusebox), por lo cual cada una de ellas tiene una función específica. Por ejemplo, a la hora de acceder a una aplicación, el circuito que tiene como función la seguridad del sistema, es el encargado de asignarle o mostrarle al usuario las parte de la aplicación que el mismo puede ver.

Cuando se realiza una petición en la aplicación, la maquinaria de Fusebox, enruta dicha petición al circuito correspondiente, donde es procesada de forma adecuada. La idea de encapsular las responsabilidades hace más fácil que los circuitos puedan ser reutilizados.

Dentro de cada circuito responsable de ejecutar las acciones (fuseactions) solicitadas, el analista especifica los ficheros individuales (fuses o fusibles) necesarios para realizar la acción requerida. De esta manera, Fusebox actúa como un buen gestor, “delegando” tareas a los departamentos apropiados, donde son descompuestas en tareas individuales, cada una de las cuales puede ser asignada a tareas más básicas independientes.

Fusebox es mas bien una metodología orientada a la encapsulación, modularización y reutilización del código de sus aplicaciones, las cuales se componen, fundamentalmente de circuits, fuseactions y fuses, además de los códigos núcleo de la propia arquitectura Fusebox [14].

1.11 Visión sobre las configuraciones arquitectónicas

PHP es una tecnología que con el pasar de los años ha ido ganando fuerza, esto se debe a muchas personas que están dispuestas a que el software libre sea visto igual que el software propietario, por lo

que se han dedicado a desarrollar específicamente para PHP una seria de frameworks para que el mismo cada vez sea más sencillo, fácil de usar y más potente.

Estos frameworks son capaces de hacer el desarrollo de una aplicación mucho más ágil y con mayor precisión, son una buena base para el desarrollo de cualquier aplicación ya que ahorran gran cantidad de tiempo y muchas funciones que demorarían a una personas hacer en semanas, quizás un mes, estos lo traen consigo implementado, nada más es referenciarlas.

Muchos de los frameworks de PHP reconocidos y tienen gran prestigio en el mundo, por las facilidades y la facilidades a la hora de emplearlo. Symfony, Kumbia, Fusebox, etc. son ejemplos de frameworks usados en proyectos de prestigio mundial como Yahoo.

1.12 Entorno de desarrollo integrado (IDE)

Integrated Development Environment (entorno de desarrollo integrado), es un editor de código que además puede servirnos para depurar y facilitarnos las diferentes tareas necesarias en el desarrollo de cualquier tipo de aplicación. Existen IDE que pueden funcionar con diferentes lenguajes de programación, un buen ejemplo de ello es Eclipse.

Existen muchos IDE para PHP, hay una serie de ellos que siempre se han destacado en el mercado mundial, los mismos son [15]:

Zend Studio

Son muchos los desarrolladores que trabajan con Zend Studio, es posiblemente uno de los mejores IDE del momento. Se trata de un programa de la casa Zend, uno de los mayores impulsores de PHP, orientada a desarrollar aplicaciones Web, como no, en PHP. Zend Studio es un editor de texto para páginas PHP que proporciona un buen número de ayudas desde la creación y gestión de proyectos hasta la depuración del código.

La nueva versión de Zend es Zend Studio Neon, esta basado en Eclipse, Eclipse es una plataforma de software de código abierto independiente de una plataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido".

Open Komodo Project

Es un proyecto bastante singular porque se basa en la creación de aplicaciones Web que corran bajo Firefox, además de estar construidas con el Komodo IDE. Así, la compañía creadora del proyecto se asegura estar presente en la lucha contra la Web cerrada, al igual que otros, como es el caso del conocido Mozilla. El Open Komodo Project responde a las exigencias de las herramientas para el desarrollo de la tecnología de Web libre, promoviendo la innovación en Web y la libertad de elección para los desarrolladores y usuarios finales.

Con la creación del Open Komodo Project, ActiveState se une a otros defensores como Mozilla en la promoción de la Web abierta. La filosofía de Web abierta pretende mantener la integridad de la Web como un espacio público neutral en cuanto a plataforma, idioma y navegador.

Eclipse

Eclipse es una plataforma de software de código abierto independiente y que es y ha sido muy utilizada para desarrollar entornos de desarrollo (IDE), pero del mismo modo se puede usar para otros tipos de aplicaciones cliente (Ej. BitTorrent, Azureus). Lo mejor de Eclipse es que tiene una gran comunidad de usuarios extendiendo constantemente las aplicaciones.

Algunos proyectos de IDE's con Eclipse son:

- AspectJ: es una extensión del lenguaje Java orientado a aspectos.
- Proyecto de herramientas de desarrollo en C/C++ (CDT) trabaja para proveer un Ambiente integrado de desarrollo completamente funcional para C y C++ para la plataforma Eclipse.
- Subproyecto IDE de COBOL para Eclipse (COBOL) construye un Ambiente Integrado de Desarrollo (IDE) completamente funcional para COBOL en la plataforma Eclipse.
- Herramientas de Desarrollo de Java (JDT) provee las herramientas que implementan un IDE de Java, soportando el desarrollo de cualquier aplicación Java, incluyendo los plug-ins de Eclipse.
- Photran (photran) es un IDE completamente funcional para Fortran con soporte para refactorización.
- PHP Development Tools trabaja para proveer un IDE completamente funcional para PHP

para la plataforma Eclipse.

- Wolfram Workbench es un IDE basado en Eclipse (también disponible como plugin para Eclipse) para el lenguaje Mathematica.
- PyDev un IDE completamente funcional para python con soporte para refactorización y depurador gráfico.

PDT (PHP Development Tools, Eclipse)

Este proyecto ha tenido una gran respuesta entre los desarrolladores de PHP y que ha sido descargado más de 300.000 veces.
Entre las características en la versión actual (1.0) se encuentran:

- Editor sensible al contexto, el cual provee resaltamiento de código, asistente de código y autocompletado de código.
- Integración con el modelo del proyecto Eclipse, que permite para inspeccionar el uso de las vistas del contorno del fichero y del proyecto, así como la nueva vista PHP Explorer.
- Soporte para el debug incremental del código de PHP
- Extensos frameworks y APIs que permiten a los desarrolladores e ISVs (vendedores de software independientes) fácilmente extender PDT para crear nuevas e interesantes herramientas orientadas al desarrollo de PHP.

PHP Designer

PHP Designer, es un completo entorno de desarrollo y programación especialmente diseñado para desarrolladores de PHP, aunque también permite trabajar con comodidad con otras tecnologías como HTML, XHTML, CSS y SQL.

Ofrece toda una serie de asistentes y diálogos integrados que facilitan en todo momento tu tarea, además de acceso directo a librerías de código o scripts de uso habitual, utilidades diversas y toda suerte de herramientas, todo ello en una interfaz de diseño sencillo y elegante que puedes personalizar con nada menos que dieciocho temas distintos. Cuenta con cliente de FTP y navegador de ficheros

integrado, utilidades de corrección y autocompletado, búsqueda integrada en Google y soporte para proyectos.

PHPEdit

PHPEdit ofrece un entorno de trabajo para todos aquellos programadores especializados en PHP, pero que también permite trabajar con HTML, XML y TXT; un entorno integrado para Windows y con una variada colección de herramientas.
Entre dichas herramientas se incluyen plantillas de teclado, marcadores de texto, un "debugger" integrado para detectar errores en el código y un código de color para comandos de sintaxis que hace más sencilla e intuitiva la lectura.

1.13 ¿Qué es un servidor Web?

Un servidor Web sirve contenido estático a un navegador, carga un archivo y lo sirve a través de la red al navegador de un usuario. Este intercambio es mediado por el navegador y el servidor que hablan el uno con el otro mediante HTTP. Se pueden utilizar varias tecnologías en el servidor para aumentar su potencia más allá de su capacidad de entregar páginas HTML; éstas incluyen scripts CGI, seguridad SSL y páginas activas del servidor (ASP) [16].

Ejemplo:

Nombre del Servidor	**Sistema operativo**
AOLserver	Unix, Windows 2000, Windows 95/98,Windows ME, Windows NT 4.0, Windows Server 2003, Windows XP
Apache	NetWare, OS/2, Unix, Windows 95/98,Windows NT 4.0
BadBlue	Windows 2000, Windows 95/98,Windows ME, Windows NT 4.0, Windows XP
Baikonur Web App Server	Windows 95/98, Windows NT 4.0
Covalent Enterprise Ready Server	AIX, HP-UX, Linux, Red Hat Linux, SUSE Linux, Solaris, Windows 2000, Windows Server 2003

ESAWEB	VM/CMS
Enterprise WebServer for NetWare	NetWare
GoAhead WebServer	Linux, NetWare, Solaris, Windows 2000, Windows 95/98
Hawkeye	Linux
Java Server	HP-UX, IRIX, Linux, OS/2, Solaris, Windows 95/98 Windows NT 4.0
Jigsaw	Java_VM, Solaris,Windows 95/98, Windows NT 4.0
Microsoft Internet Information Services	Windows Server 2003
RapidSite	BSD, IRIX, Windows 2000, Windows Server 2003
RomPager Embedded Web Server	Embedded
Roxen WebServer	Linux, MacOS X Server, Solaris, Windows 2000, Windows Server 2003, Windows XP
Sambar Server	Red Hat Linux, Windows 2000, Windows 95/98,Windows ME, Windows NT 4.0, Windows Server 2003, Windows XP
Savant	Windows 2000, Windows 95/98,Windows ME, Windows NT 4.0
Servertec Internet Server	AIX, HP-UX, Linux, Solaris, Windows 2000, Windows 95/98, Windows ME, Windows NT 4.0,Windows Server 2003 Windows XP
Shadow Web Server	MVS
SimpleServer	Windows 2000, Windows 95/98,Windows ME, Windows NT 4.0, Windows XP
Sun Java System Web Server	Red Hat Linux, Solaris, Windows 2000, Windows XP
Tcl Web Server	Linux, MacOS, Unix, Windows NT 4.0
Viking	Windows 2000, Windows 95/98 , Windows NT 4.0, Windows

	XP
vqServer	AIX, BSD, Be OS, Digital UNIX, HP-UX, IRIX, Java_VM, Linux, MacOS, OS/2, SCO OpenServer, Solaris, Windows 95/98, Windows NT 4.0
WN	AIX, BSD, Digital UNIX, HP-UX, IRIX, Linux, SCO OpenServer, Solaris
WebBase	Windows 2000, Windows 95/98, Windows NT 4.0

Son innumerables los distintos tipos de servidores Web que existen así como en los sistemas operativos que se pueden usar. Estas son una herramienta muy útil para la publicación de los sitios Web, por lo que en el mercado internacional, siempre ha existido la competencia entre los mismo, con la rapidez que estos gestionan la paginas y se comunican a través del protocolo HTTP.

1.14 ¿Qué es un gestor de base de datos?

Los sistemas de bases de datos están diseñados para gestionar grandes volúmenes de información. Generalmente, las bases de datos requieren gran cantidad de espacio de almacenamiento, por lo que las bases de datos de las organizaciones se miden en términos de *gigabytes* o *terabytes* de datos. Un gigabyte equivale a 1000 megabytes (un billón de bytes), y un terabyte equivale a un millón de megabytes (un trillón de bytes). Un sistema de bases de datos tiene como objetivo simplificar y facilitar el acceso a los datos y hacer que los tiempos de respuesta a las solicitudes de los usuarios sean muy reducidos.

De forma sencilla, un sistema de gestión de bases de datos se puede definir como una colección de datos interrelacionados y un conjunto de programas para acceder a esos datos. Adoración de Miguel lo define como [17]: "*conjunto coordinado de programas, procedimientos, lenguajes, etc. que suministra, tanto a los usuarios no informáticos como a los analistas, programadores o al administrador, los medios necesarios para describir, recuperar y manipular los datos almacenados en la base, manteniendo su integridad, confidencialidad y seguridad.*"

Existen dos grandes modelos de sistemas de gestión de bases de datos:

Sistemas de Gestión de Bases de Datos Relacionales (SGBDR)

Las bases de datos que generan se construyen con información muy estructurada (datos) acerca de una organización o empresa determinada. Cuando un usuario realiza una consulta en una base de datos relacional, el sistema presenta como resultado la respuesta exacta a lo que se busca. A este tipo de bases de datos se les denomina bases de datos relacionales, y a los sistemas que las gestionan, Sistemas de Gestión de Bases de Datos Relacionales (SGBDR).

Sistemas de Gestión de Bases de Datos Documentales (SGBDD) o Sistemas de Recuperación de Información (SRI)

Las bases de datos que generan se construyen con información no estructurada tipo texto (documentos) sobre uno o varios temas. Cuando un usuario realiza una consulta en una base de datos documental, el sistema presenta como resultado, no una respuesta exacta, sino documentos útiles para satisfacer la pregunta del usuario. A este tipo de bases de datos se les denomina bases de datos documentales, y a los sistemas que las gestionan, Sistemas de Gestión de Bases de Datos Documentales (SGBDD) o Sistemas de Recuperación de Información (SRI).

Pese a que las bases de datos relacionales y los sistemas que las gestionan (SGBDR) son los más utilizados y por tanto los más populares, las bases de datos documentales han experimentado un fuerte auge durante estos dos últimos años, impulsado sobre todo por la popularización de Internet y la consiguiente saturación de información textual que ha traído el Internet, así como por el reciente interés de las grandes empresas por gestionar el conocimiento almacenado en documentos.

En la actualidad existen nuevos sistemas que combinan funcionalidades del modelo relacional y el modelo documental, son los Sistemas de Gestión de Bases de Datos Relacionales/Documentales o Sistemas Híbridos. Las bases de datos que generan se construyen con información muy estructurada (datos) y con información no estructurada (documentos). Su comportamiento en la recuperación de la información varía de unos sistemas a otros.

Atendiendo al lugar en el que almacenan los datos y/o la información, los sistemas de bases de datos pueden ser:

- Sistemas de bases de datos centralizados: los datos y/o la información residen en un solo ordenador.
- Sistemas de bases de datos distribuidos: los datos y/o la información se almacenan en varios ordenadores. Los ordenadores de un sistema distribuido se comunican entre sí a través de distintos medios de telecomunicaciones. A los ordenadores que forman un sistema distribuido se les llama nodos. Por tanto, un sistema distribuido de bases de datos consiste en un conjunto de nodos, y cada uno de ellos puede participar en la ejecución de transacciones que accedan a datos de uno o varios nodos.

1.15 Conclusiones

En este capítulo se han analizado algunos aspectos teóricos que nos serán de gran ayuda para proponer una solución en el capítulo 2. Se plasman conceptos generales referentes al tema central de la investigación, lo cual ayudarán al desarrollo del trabajo y una mejor compresión de las características que tiene el sistema. Por lo que se analizó las principales definiciones y caracterizaciones de los patrones arquitectónicos y estilos arquitectónicos, arquitectura de software, etc. Se define que es un sistema ERP, y PHP como una tecnología de grandes ventajas.

CAPÍTULO 2: PROPUESTA ARQUTECTÓNICA

2.1 Introducción

En este capítulo se hace una propuesta de arquitectura del sistema ERP cubano, con el objetivo de darle respuesta al problema que se plantea en el diseño teórico. El alcance de esta investigación es proponer un diseño de la vista lógica de la arquitectura, artefacto clave para realizar una arquitectura de software. Mediante este artefacto, se plantea una arquitectura vertical, la misma será capaz de mostrar los requisitos propuestos, las relaciones y funciones del mismo dentro de esta arquitectura.

2.2 Arquitectura vertical

2.2.1 Introducción

El objetivo del diseño de la arquitectura vertical es lograr una abstracción en el diseño arquitectónico. La arquitectura vertical contiene los elementos imprescindibles para lograr la mayor abstracción en el diseño arquitectónico de la aplicación. Se enumeran los patrones, las restricciones de software y hardware, las tecnologías y herramientas utilizadas en el diseño de la arquitectura [1].

El propósito de la arquitectura vertical es proporcionar la información necesaria para estructurar el sistema desde el mayor nivel de abstracción. Los usuarios son:
El equipo de arquitectos: La utiliza como guía para la toma de decisiones arquitectónicas, y son los encargados de su refinamiento.
El equipo de desarrolladores: La utiliza como guía para la implementación.
Clientes: Pueden encontrar en ella la garantía de la calidad y el conocimiento de la tecnología y herramientas seleccionadas.

Alcance

La arquitectura vertical describe la estructura del sistema en su más alto nivel de abstracción. Describe detalladamente el organigrama de la arquitectura basada en los estilos arquitectónicos que se utilizarán; los principales frameworks de desarrollo y como se adaptarán a la solución; se realiza una propuesta de la utilización de los patrones en el desarrollo del sistema, ya que los mismo resuelven problemas que se podrían presentar a lo largo del desarrollo del mismo. También se hace

una selección de las tecnologías y herramientas que soportan los estilos y patrones descritos y cumplen con las restricciones del sistema.

2.2.2 Descripción estilística de la arquitectura

El patrón Modelo Vista Controlador es propuesto para el diseño de la arquitectura del sistema ERP cubano. Para ello se decidió la utilización como framework principal Symfony, que utiliza este patrón, ya que el principio más importante de la arquitectura MVC es la separación del código del programa en tres estructuras bien definidas, dependiendo de su naturaleza.[11].

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones.

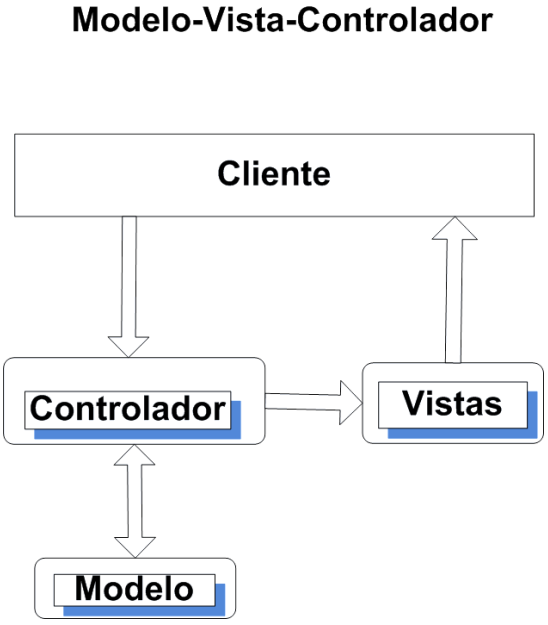


Figura 1 MVC

El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación.

La capa del modelo se puede dividir en la capa de acceso a los datos y en la capa de abstracción de la base de datos. De esta forma, las funciones que acceden a los datos no utilizan sentencias ni consultas que dependen de una base de datos, sino que utilizan otras funciones para realizar las consultas. Así, si se cambia de sistema gestor de bases de datos, solamente es necesario actualizar la capa de abstracción de la base de datos.

Todos los controladores de la aplicación entre las tareas comunes se encuentran el manejo de las peticiones del usuario, el manejo de la seguridad, cargar la configuración de la aplicación y otras tareas similares. Por este motivo, el controlador normalmente se divide en un controlador frontal, que es único para cada aplicación, y las acciones, que incluyen el código específico del controlador de cada página. Una de las principales ventajas de utilizar un controlador frontal es que ofrece un punto de entrada único para toda la aplicación.

La capa de la vista también puede aprovechar la separación de código. Las páginas Web suelen contener elementos que se muestran de forma idéntica a lo largo de toda la aplicación: cabeceras de la página, el layout genérico, el pie de página y la navegación global. Normalmente sólo cambia el interior de la página. Por este motivo, la vista se separa en un layout y en una plantilla. Normalmente, el layout es global en toda la aplicación o al menos en un grupo de páginas. El template sólo se encarga de visualizar las variables definidas en el controlador.

2.2.3 Gráfico estructural del diseño estilístico del sistema

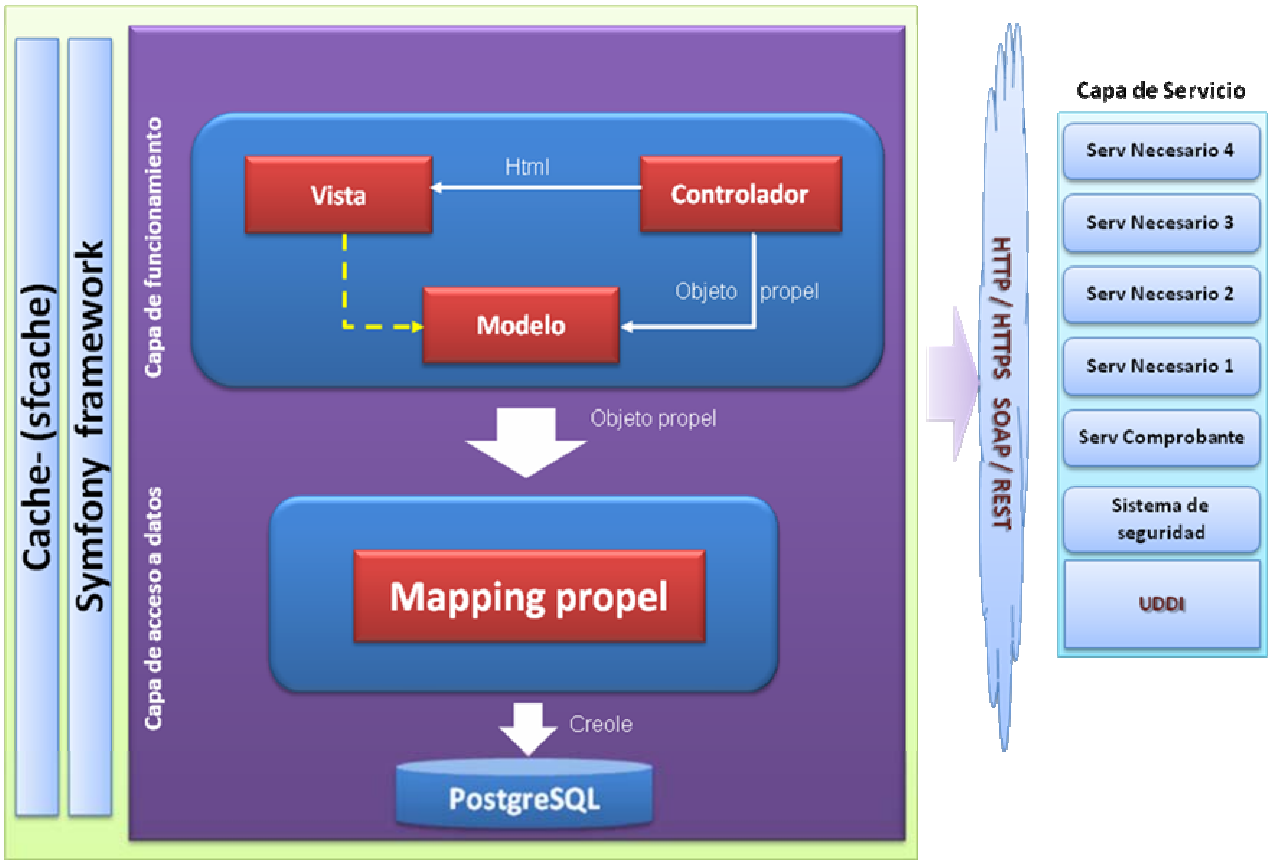


Figura 2 Organigrama de arquitectura

2.2.4 Estructura física de los directorios del framework

Symfony considera un proyecto como “un conjunto de servicios y operaciones disponibles bajo un determinado nombre de dominio y que comparten el mismo modelo de objetos”. Dentro de un proyecto, las operaciones se agrupan de forma lógica en aplicaciones. Normalmente, una aplicación se ejecuta de forma independiente respecto de otras aplicaciones del mismo proyecto [11].

También es posible definir proyectos que estén formados por varios sitios web pequeños, cada uno de ellos considerado como una aplicación. En este caso, es importante tener en cuenta que los enlaces entre aplicaciones se deben indicar de forma absoluta. Cada aplicación está formada por uno o más módulos. Un módulo normalmente representa a una página web o a un grupo de páginas con un

propósito relacionado. Por ejemplo, una aplicación podría tener módulos como home, artículos, ayuda, carritoCompra, cuenta, etc.

Framework	PHP 4	PHP 5	MVC	Múltiples DBMS	ORM	Plantillas	Cache	Validación	Ajax	Plugins
Zend Framework	-	+	+	+	-	-	+	+	-	+
CakePHP	+	+	+	+	+	-	+	+	+	
symfony	-	+	+	+	+	-	+	+	+	+
Seagull Framework	+	+	+	+	+	+	+	+	-	+
WACT	+	+	+	+	-	+	-	+	-	-
Prado	-	+	-	+	-	+	+	+	+	+
PHP on TRAX	-	+	+	+	+	-	-	+	+	-
ZooP Framework	+	+	+	+	-	+	+	+	+	+
eZ Components	-	+	-	+	-	+	+	+	-	+
CodeIgniter	+	+	+	+	-	+	+	+	-	+

Figura 3 Tabla de comparación entre los frameworks más usados de PHP [24]

2.2.4.1 Estructuración de la raíz del proyecto

```
apps/
  frontend/
  backend/
batch/
cache/
config/
data/
  sql/
doc/
```

```
lib/
  model/
log/
plugins/
test/
  unit/
  functional/
web/
  css/
  images/
  js/
  uploads/
```

Ver la tabla 1 del anexo 1, describe los contenidos de estos directorios.

2.2.4.2 Estructura de cada aplicación

Todas las aplicaciones de Symfony tienen la misma estructura de archivos y directorios [11]:

```
apps/
  [nombre aplicación]/
    config/
    i18n/
    lib/
    modules/
    templates/
      layout.php
      error.php
      error.txt
```

Ver la tabla 2 del anexo 1, describe los contenidos de estos directorios.

Las clases de una aplicación no pueden acceder a los métodos o atributos de otras aplicaciones del mismo proyecto. Además, los enlaces entre 2 aplicaciones de un mismo proyecto se deben indicar de forma absoluta. Esta última restricción es importante durante la inicialización del proyecto, que es cuando debes elegir como dividir el proyecto en aplicaciones.

2.2.4.3 Estructura de cada módulo

```
apps/
  [nombre aplicación]/
    modules/
      [nombre modulo]/
        actions/
          actions.class.php
        config/
        lib/
```

```
templates/
  indexSuccess.php
validate/
```

Ver la tabla 3 del anexo 1, describe los contenidos de estos directorios.

2.2.4.4 Estructura del directorio web

Existen pocas restricciones sobre la estructura del directorio *web*, que es el directorio que contiene los archivos que se pueden acceder de forma pública. Si se utilizan algunas convenciones básicas en los nombres de los subdirectorios, se pueden simplificar las plantillas. La siguiente es una estructura típica del directorio web [11]:

```
web/
  css/
  images/
  js/
  uploads/
```

Ver la tabla 4 del anexo 1, describe los contenidos de estos directorios.

2.2.5 La vista

La vista contiene todas las clases de interfaz de usuarios que representan la interacción del usuario con el sistema, a su vez interactúa directamente con el controlador y también depende del modelo.

La vista esta compuesta por layout y plantillas, ambas son encargadas de producir las páginas que se muestran como resultado de las acciones. La presentación en Symfony se desglosa en varias partes, estando cada una de ellas especialmente preparada para que pueda ser fácilmente modificable por la persona que normalmente trabaja con cada aspecto del diseño de la interfaz [11].

Los diseñadores web normalmente trabajan con las plantillas (que son la presentación de los datos de la acción que se está ejecutando) y con el layout (que contiene el código HTML común a todas las páginas). Estas partes están formadas por código HTML que contiene pequeños trozos de código PHP, que normalmente son llamadas a los diversos helpers disponibles.

Para mejorar la reutilización de código, los programadores suelen extraer trozos de las plantillas y los transforman en componentes y elementos parciales. De esta forma, el layout se modifica para definir zonas en las que se insertan componentes externos. Los diseñadores web también pueden trabajar fácilmente con estos trozos de plantillas.

Los programadores normalmente centran su trabajo relativo a la presentación en los archivos de configuración YAML (que permiten establecer opciones para las propiedades de la respuesta y para otros elementos de la interfaz) y en el objeto respuesta. Cuando se trabaja con variables en las plantillas, deben considerarse los posibles riesgos de seguridad de XSS (cross-site scripting) por lo que es necesario conocer las técnicas de escape de los caracteres introducidos por los usuarios.

La vista está formada por 3 paquetes fundamentales

- La presentación HTML del resultado de la acción (que se guarda en la plantilla, en el layout)
- El resto, que incluye entre otros los siguientes elementos:
    1. Declaraciones <meta>: palabras clave, descripción, duración de la caché, etc.
    2. El título de la página: no solo es útil para los usuarios que tienen abiertas varias ventanas del navegador, sino que también es muy importante para que los buscadores indexen bien la página.
    3. Inclusión de archivos: de JavaScript y de hojas de estilos.
    4. Layout: algunas acciones necesitan un layout personalizado (ventanas emergentes, anuncios, etc.) o puede que no necesiten cargar ningún layout (por ejemplo en las acciones relacionadas con Ajax).

Archivo de configuración view.yml

Todo aquello que no es código HTML se considera como configuración de la propia vista. Symfony permite 2 formas de manipular esa configuración. La forma usual es mediante el archivo de configuración *view.yml*. Se utiliza cuando los valores de configuración no dependen del contexto o de alguna consulta a la base de datos. Cuando se trabaja con valores dinámicos que cambian con cada acción, se recurre al segundo método para establecer la configuración de la vista: añadir los atributos directamente en el objeto sfResponse durante la acción.

Cuando se genera un módulo, la misma crea un archivo *view.yml* que define las opciones de su propia vista, lo que permite establecer un único archivo de configuración para un módulo entero, un ejemplo de este archivo sigue a continuación:

```
editSuccess:
metas:
title: Edita tu perfil
editError:
metas:
title: Error en la edición del perfil
all:
stylesheets: [mi_estilo]
metas:
title: Mi sitio web
```

y la configuración de la vista de la aplicación se encuentra aquí *apps/miaplicacion/config/view.yml*.

Ejemplo:

```
default:
http_metas:
content-type: text/html
metas:
title: symfony project
robots: index, follow
description: symfony project
keywords: symfony, project
language: en
stylesheets: [main]
javascripts: [ ]
has_layout: on
layout: layout
```

Como se dijo anteriormente esta configuración también se puede realizar mediante el objeto sfResponse pasándole todos sus propiedades, en la acciones.

```
class mimoduloActions extends sfActions
{
   public function executeIndex()
     {
        $respuesta = $this->getResponse();
         // Cabeceras HTTP
        $respuesta->setContentType('text/xml');
        $respuesta->setHttpHeader('Content-Language', 'en');
        $respuesta->setStatusCode(403);
        $respuesta->addVaryHttpHeader('Accept-Language');
        $respuesta->addCacheControlHttpHeader('no-cache');
         // Cookies
        $respuesta->setCookie($nombre, $contenido, $expiracion, $ruta, $dominio);
        // Atributos Meta y cabecera de la página
```

```
        $respuesta->addMeta('robots', 'NONE');
        $respuesta->addMeta('keywords', 'palabra1 palabra2');
        $respuesta->setTitle('Mi Página de Ejemplo');
        $respuesta->addStyleSheet('mi_archivo_css');
        $respuesta->addJavaScript('mi_archivo_javascript');
    }
}
```

2.2.5.1 Layout

Los layout son plantillas pero son más globales y pueden diseñarse para la aplicación en general y para una página específica. Este es un archivo llamado *layout.php* situada en *miproyecto/app/miaplicacion/template*. Almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El contenido del template se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla. Este comportamiento es una implementación del patrón de diseño llamado “decorator”.

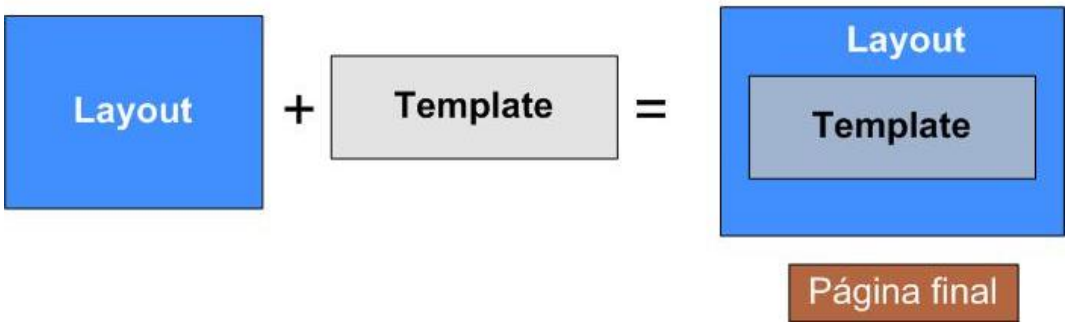


Figura 4 Vista

2.2.5.2 Templates

Su contenido está formado por código HTML y algo de código PHP sencillo, normalmente llamadas a las variables definidas en la acción (mediante la instrucción $this->nombre_variable = ‘valor’) y algunos helpers, las plantillas van embebidas dentro de las layout. No deben de tener mucho código PHP en su interior, para que la ejecución de las mismas sea mucho más rápida.

Se utilizan helpers que no son más que funciones PHP que devuelven código HTML, la utilización de los mismos ahorran gran cantidad de tiempo, ya que una función de helpers es mucho más sencilla que una de HTML, por lo que los helpers para realizar una instrucción, nada más utilizan una sola línea de código, ya que en HTML sería más complicado.

Objeto $sfdata

Anterior mente se definía como estaba compuesta la vista, en layout y templates. Ambos se comunican mediante el objeto $sfdata, este es una objeto contenedor de todas las variables que se declaran en todos los templates de la aplicación.

Se puede acceder a este objeto cuando el mecanismo de escape de datos es activado. Siempre es recomendable mantener este mecanismo activado porque cualquier usuario mal intencionado podría hacer un ataque del tipo XSS, y traería malas consecuencias, en la base de datos y en la aplicación en general.

2.2.6 El controlador

El controlador va estar compuesto por el controlador frontal y la clase action**,** las mismas son el corazón de la aplicación, puesto que contienen toda la lógica de la aplicación. Las acciones utilizan el modelo y definen variables para la vista. Cuando se realiza una petición web en una aplicación Symfony, la URL define una acción y los parámetros de la petición [11].

2.2.6.1 Controlador frontal

Todas las peticiones web son manejadas por un solo controlador frontal, que es el punto de entrada único de toda la aplicación en un entorno determinado. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL escrita (o pinchada) por el usuario. Por ejemplo, la siguiente URL llama al script index.php (que es el controlador frontal) y será entendido como llamada a la acción miAccion del módulo mi módulo: *http://localhost/index.php/mimodulo/miAccion*

El controlador frontal se encarga de despachar las peticiones, lo que implica algo más que detectar la acción que se ejecuta. De hecho, ejecuta el código común a todas las acciones, incluyendo los siguientes pasos:

1. Define las constantes del núcleo.
2. Localiza la librería de Symfony
3. Carga e inicializa las clases del núcleo del framework.

4. Carga la configuración.
5. Decodifica la URL de la petición para determinar la acción a ejecutar y los parámetros de la petición.
6. Si la acción no existe, redireccionará a la acción del error 404.
7. Activa los filtros (por ejemplo, si la petición necesita autenticación).
8. Ejecuta los filtros, primera pasada.
9. Ejecuta la acción y produce la vista.
10. Ejecuta los filtros, segunda pasada.
11. Muestra la respuesta.

El controlador frontal por defecto, llamado index.php y ubicado en el directorio web/ del proyecto, es un simple script que se presenta a continuación.

```
<?php
define('SF_ROOT_DIR', realpath(dirname(__FILE__).'/..'));
define('SF_APP', 'miaplicacion');
define('SF_ENVIRONMENT', 'prod');
define('SF_DEBUG', false);
require_once(SF_ROOT_DIR.DIRECTORY_SEPARATOR.'apps'.DIRECTORY_SEPARATOR.SF_APP.DIRECTORY_SE
PARATOR.')
sfContext::getInstance()->getController()->dispatch();
```

El controlador frontal incluye el config.php de la aplicación, que se ocupa de los pasos 2 a 4. La llamada al método dispatch() del objeto sfController (que es el objeto correspondiente al controlador del núcleo de la arquitectura MVC de Symfony) envía la petición, ocupándose de los pasos 5 a 7.

2.2.6.2 Acciones

Cada módulo esta compuesto por una clase action que hereda de la clase sfActions. Cuando la clase action es creada, la misma toma el nombre nombreModulo+Actions, en ella se crean métodos con el nombre executeNombreAction. La clase que representa las acciones de un módulo se encuentra en el archivo *actions.class.php*, en el directorio *actions/* del módulo. Es recomendable que estas clases no lleven mucho código, para no sobrecargar su tarea, fundamentalmente es para realizar operaciones sencillas y repartir las tareas en la aplicación. El valor retornado por el método de una acción determina como será producida la vista.

Ejemplo de la clase de la acción.

```
class mimoduloActions extends sfActions
 {
      public function executeIndex()
        {
        }
   }
```

Para especificar el template que se utiliza al mostrar el resultado de la acción, se emplean las constantes de la clase sfView. Cada método action va tener correspondido un template, ya que al ejecutarse el mismo va a llamar a la clase sfView la cual retorna un SUCCESS, ejemplo:

```
return sfView::SUCCESS;
```

Symfony buscará entonces un template llamado *nombreAccionSuccess.php*. Este comportamiento se ha definido como el comportamiento por defecto, por lo que si omites la sentencia return en el método de la acción, las acciones vacías también siguen este comportamiento.

2.2.6.2.1 Validación y manejo de errores

La validación de los datos de la acción –normalmente los parámetros de la petición– es una tarea repetitiva y tediosa. Symfony incluye un sistema de validación, utilizando métodos de la clase acción. Se ve en primer lugar un ejemplo. Cuando un usuario hace una petición a miAccion, Symfony siempre busca primero un método llamado validateMiAccion(). Si lo encuentra, Symfony ejecuta ese método. El valor de retorno de esta validación determina el siguiente método que se ejecuta: si devuelve true, entonces se ejecuta el método executeMiAccion(); en otro caso, se ejecuta handleErrorMiAccion(). En el caso de que handleErrorMiAccion() no exista, Symfony busca un método genérico llamado handleError().Si tampoco existe, simplemente devuelve el valor sfView::ERROR para producir la plantilla *miAccionError.php*. En el Anexo 3 se muestra el diagrama que describe este proceso.

```
class mimoduloActions extends sfActions
{
  public function validateMiAccion()
   {
    return ($this->getRequestParameter('id') > 0);
   }
  public function handleErrorMiAccion()
   {
    $this->message = "Parámetros no válidos";
     return sfView::SUCCESS;
```

```
    }
   public function executeMiAccion()
    {
     $this->message = "Los parámetros son válidos";
    }
}
```

2.2.7 Clases mapeadas

El esquema se utiliza para construir las clases del modelo que necesita la capa del ORM. Para reducir el tiempo de ejecución de la aplicación, estas clases se generan mediante una tarea de línea de comandos llamada propel-build-model [11].

Al ejecutar ese comando, se analiza el esquema y se generan las clases base del modelo, que se almacenan en el directorio *lib/model/om/* del proyecto:

- BaseArticle.php
- BaseArticlePeer.php
- BaseComment.php
- BaseCommentPeer.php

Además, se crean las verdaderas clases del modelo de datos en el directorio *lib/model/*:

- Article.php
- ArticlePeer.php
- Comment.php
- CommentPeer.php

2.2.8 El modelo

Las clases objeto representan un registro de la base de datos. Permiten acceder a las columnas de un registro y a los registros relacionados. De estas clases se crean instancias en el controlador quedando de la siguiente forma [11]:

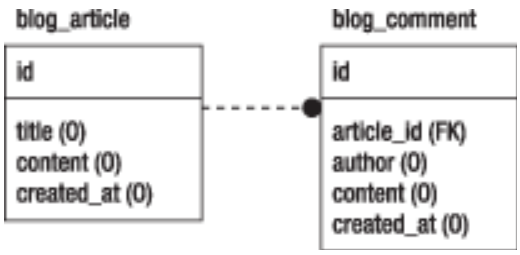
```
$articulo = new Article();
$titulo = $articulo->getTitle();
```

También están las clases inherente a la lógica de negocio, las mismas se van a ubicar en *miaplicacion/module/lib.*

2.2.9 Capa acceso a datos

Las aplicaciones Web con el pasar de los años han ido cogiendo nivel por lo que casi siempre la lógica del negocio de las aplicaciones Web depende de un modelo de datos. El componente que se encarga de gestionar los datos es una capa de tipo ORM (object-relational-mapping), la cual existe un proyecto que se ha dedicado al desarrollo de este componente. Para el acceso y modificación de los datos en la base de datos se realiza mediante objetos, logrando un alto nivel de abstracción [11].

La principal ventaja que aporta el ORM es la reutilización, esto permite llamar a los métodos de un objeto de datos desde varias partes de la aplicación e incluso desde diferentes aplicaciones. Para crear el modelo de objetos de datos que utiliza Symfony, se debe traducir el modelo relacional de la base de datos a un modelo de objetos de datos. Para realizar ese mapeo, el ORM necesita una descripción del modelo relacional, que se llama "esquema" (*schema*). En el esquema se definen las tablas, sus relaciones y las características de sus columnas. La sintaxis que utiliza Symfony para definir los esquemas hace uso del formato YAML. En la siguiente figura se muestran unas tablas de ejemplo:



En el archivo *schema.yml* quedara traducido de la siguiente manera:

```
propel:
blog_article:
_attributes: { phpName: Article }
id:
title: varchar(255)
content: longvarchar
created_at:
blog_comment:
_attributes: { phpName: Comment }
id:
article_id:
```

```
author: varchar(255)
content: longvarchar
created_at:
```

Estos ficheros se pueden encontrar en *miproyecto/config/* .Symfony utiliza este tipo de fichero ya que son fácil de leer y de comprender la estructura del mismo. Estos esquemas se emplean para obtener las clases del modelo que necesita ORM.

2.2.9.1 Clases Peer

Las clases Peer presenta métodos estáticos, por lo que no se pueden crear instancias de la misma, sus métodos devuelven normalmente un objeto o una colección de objetos de la clase objeto relacionada. Para acceder a los métodos de las misma no se realiza de la forma clásica (objeto->método ()) sino mediante (Nombre_clase**::**método()). Los métodos de esta clase son declarados en el controlador de la siguiente forma:

```
$articulos = ArticlePeer::retrieveByPks(array(123, 124, 125));
```

El modelo relacional y el modelo de objetos utilizan conceptos similares, cada uno tiene su propia nomenclatura:

Relacional	Orientado a objetos
Tabla	Clase
Fila, Registro	Objeto
Campo, Columna	Propiedad

2.2.9.2 Conexión a la base de datos

Para realizar la conexión Symfony utiliza un archivo *databases.yml* ubicado en *miproyecto/config/databases.yml***,** este archivo contiene la información mínima para conectar una aplicación a una base da datos, ejemplo del mismo:

```
prod:
 propel:
  param:
```

```
  hostspec: miservidordatos
  username: minombreusuario
  password: xxxxxxxxxx
all:
 propel:
  class: sfPropelDatabase
  param:
  phptype: mysql # fabricante de la base de datos
  hostspec: localhost
  database: blog
  username: login
  password: passwd
  port: 80
  encoding: utf8 # Codificación utilizada para crear la tabla
  persistent: true # Utilizar conexiones persistentes
```

La conexión es muy sencilla, solamente hay que configurar unas líneas. Teniendo los *schema.yml* se puede generar la estructura de la base de datos, por lo que también se puede mediante el modelo de datos generar los *schema.yml*, este proceso se realiza a preferencia del desarrollador.

La configuración de *propel.ini*

Los comandos propel-build-sql y propel-build-schema no emplean las opciones de conexión definidas en el archivo *databases.yml*. En su lugar, estos comandos utilizan las opciones de conexión de otro archivo llamado *propel.ini* que se encuentra en el directorio *config/* del proyecto:

```
propel.database.createUrl = mysql://login:passwd@localhost
propel.database.url = mysql://login:passwd@localhost/blog
```

Este archivo también contiene otras opciones que se utilizan para configurar el generador de Propel de forma que las clases del modelo generadas sean compatibles con Symfony. La mayoría de opciones son de uso interno y por tanto no interesan al usuario, salvo algunas de ellas:

```
// Base classes are autoloaded in symfony
// Set this to true to use include_once statements instead
// (Small negative impact on performance)
propel.builder.addIncludes = false

// Generated classes are not commented by default
// Set this to true to add comments to Base classes
// (Small negative impact on performance)
```

```
propel.builder.addComments = false

// Behaviors are not handled by default
// Set this to true to be able to handle them
propel.builder.AddBehaviors = false
```

Después de modificar las opciones del archivo *propel.ini*, se debe reconstruir el modelo para que los cambios surjan efecto.

Creole

Este componente es usado por Propel para la abstracción del acceso a datos. La principal ventaja de la capa de abstracción es la portabilidad, porque hace posible el cambiar la aplicación a otra base de datos, incluso en mitad del desarrollo de un proyecto. Si se debe desarrollar rápidamente un prototipo de una aplicación y el cliente no ha decidido todavía la base de datos que mejor se ajusta a sus necesidades, se puede construir la aplicación utilizando SQLite y cuando el cliente haya tomado la decisión, cambiar fácilmente a MySQL, PostgreSQL o Oracle. Solamente es necesario cambiar una línea en el archivo de configuración *databases.yml* y todo funciona correctamente.

2.2.10 Mecanismos de seguridad usados por Symfony

2.2.10.1 Sesiones

El manejo de sesiones de Symfony se encarga de gestionar automáticamente el almacenamiento de los IDs de sesión tanto en el cliente como en el servidor. Sin embargo, si se necesita modificar este comportamiento por defecto, es posible hacerlo. Se trata de algo que solamente lo necesitan los usuarios más avanzados [11].

En el lado del cliente, las sesiones son manejadas por cookies. La cookie de Symfony se llama Symfony, pero se puede cambiar su nombre editando el archivo de configuración *factories.yml*, que se encuentra en el directorio *apps/miaplicacion/config/factories.yml*
Cambiando el nombre de la cookie de sesión:

```
all:
storage:
```

```
class: sfSessionStorage
param:
session_name: mi_nombre_cookie
```

En el lado del servidor, Symfony guarda por defecto las sesiones de usuario en archivos. Se pueden almacenar en una base de datos cambiando el valor del parámetro class en *factories.yml*,
Cambiando el almacenamiento de las sesiones en el servidor:

```
all:
storage:
class: sfMySQLSessionStorage
param:
db_table: SESSION_TABLE_NAME # Nombre de la tabla que guarda las sesiones
database: DATABASE_CONNECTION # Nombre de la conexión a la base de datos que se utiliza
```

Las clases de almacenamiento de sesiones disponibles son sfMySQLSessionStorage, sfPostgreSQLSessionStorage y sfPDOSessionStorage; la última es la preferida. La opción database define la conexión a utilizar; Symfony luego utiliza *databases.yml* para determinar los parámetros de la conexión (host, nombre de la base de datos, usuario, y password) para realizar la conexión.

La expiración de la sesión se produce automáticamente después de sf_timeout segundos. El valor de esta constante es 30 minutos por defecto y puede ser modificado para cada entorno en el archivo de configuración *settings.yml*, que se ubica en el directorio *apps/miaplicacion/config/*.
Cambiando el tiempo de vida de la sesión:

```
default:
.settings:
timeout: 1800 # Tiempo de vida de la sesión en segundos
```

2.2.10.2 Acción

En Symfony la acción puede ser restringida a usuarios con ciertos privilegios o sea los usuarios pueden ejecutar todas las acciones menos las que se les prohíbe, según el usuario, por lo que los usuarios necesitan estar autenticados para poder acceder alguna característica o parte de la aplicación. Al aplicar este sistema de seguridad a una aplicación requiere de dos pasos:

1. Declarar los requerimientos de seguridad para cada acción.
2. Autenticar a los usuarios con privilegios para que puedan acceder a estas acciones restringidas.

Cuando se ejecuta una acción la misma pasa por un filtro y verifica si el usuario tiene privilegios para acceder a la acción requerida. En Symfony los privilegios están compuestos por dos partes:

1. Las acciones seguras requieren que los usuarios estén autenticados.
2. Las credenciales son privilegios de seguridad agrupados bajo un nombre y que permiten organizar la seguridad en grupos.

En cada módulo que se genera se crea una carpeta llamada *config*, en la misma hay un archivo llamado *security.yml*, en este archivo es donde se configuran los privilegios de los usuarios, se pueden especificar los requerimientos de seguridad que los usuarios deberán satisfacer para cada acción o para todas (all) las acciones.Ejemplo del archivo:

```
ver:
is_secure: off # Todos los usuarios pueden ejecutar la acción "ver"

modificar:
is_secure: on # La acción "modificar" es sólo para usuarios autenticados

borrar:
is_secure: on # Sólo para usuarios autenticados
credentials: admin # Con credencial "admin"

all:
is_secure: off # off es el valor por defecto
```

Cuando un usuario trata de acceder una acción restringida depende de sus credenciales y sucede lo siguiente:

- Si el usuario está autenticado y tiene las credenciales apropiadas, entonces la acción se ejecuta.
- Si el usuario no está autenticado, es redireccionado a la acción de login.
- Si el usuario está autenticado, pero no posee las credenciales apropiadas, será redirigido a la acción segura por defecto, como muestra la siguiente figura.

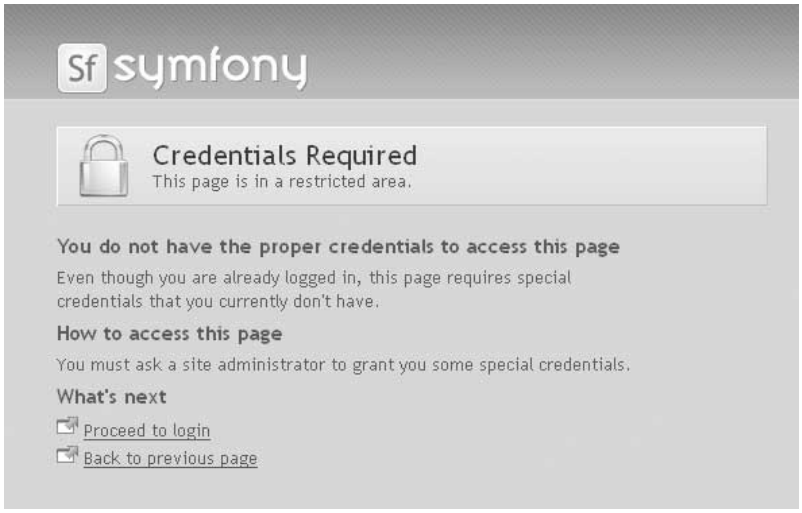


Figura 5 La página por defecto de la acción "secure"

Manejando las credenciales del usuario en la acción

```
class miCuentaActions extends sfActions
{
  public function executeEjemploDeCredenciales()
  {
    $usuario = $this->getUser();

    // Agrega una o más credenciales
    $usuario->addCredential('parametro');
    $usuario->addCredentials('parametro', 'valor');

    // Verifica si el usuario tiene una credencial
    echo $usuario->hasCredential('parametro');                    => true

    // Verifica si un usuario tiene una de las credenciales
    echo $usuario->hasCredential(array('parametro', 'valor'));    => true

    // Verifica si el usuario tiene ambas credenciales
    echo $usuario->hasCredential(array('parametro', 'valor'), true);   => true

    // Quitar una credencial
    $usuario->removeCredential('parametro');
    echo $usuario->hasCredential('parametro');                    => false

    // Elimina todas las credenciales (útil en el proceso de logout)
    $usuario->clearCredentials();
    echo $usuario->hasCredential('valor');                        => false
  }
}
```

2.2.10.3 Filtros

Un filtro es el componente por el que debe pasar cada petición antes de ejecutar la acción. Se procesa cada petición como una cadena de filtros ejecutados de forma continua. Cuando se recibe una petición, se ejecuta el primer filtro (sfRenderingFilter). Posteriormente, llama al siguiente filtro en la cadena, luego el siguiente, y así repetidamente. Cuando se ejecuta el último filtro (sfExecutionFilter), los filtros anteriores pueden finalizar, y así hasta el filtro de sfRenderingFilter. En el Anexo 3 se representa una aproximación al flujo de la cadena de filtros.

Este proceso es la razón de la estructura de la clases de tipo filtro. Todas estas clases extienden la clase sfFilter y contienen un método execute() que espera un objeto de tipo $filterChain como parámetro. En algún punto de este método, el filtro pasa al siguiente filtro en la cadena, llamando a `$filterChain->execute()`. Por lo tanto, los filtros se dividen en dos partes:

- El código que se encuentra antes de la llamada a `$filterChain->execute()` se ejecuta antes de que se ejecute la acción.
- El código que se encuentra después de la llamada a `$filterChain->execute()` se ejecuta después de la acción y antes de producir la vista.

Estructura de la clase filtro

```
class miFiltro extends sfFilter
{
    public function execute ($filterChain)
    {
        // Código que se ejecuta antes de la ejecución de la acción
        ...
        // Ejecutar el siguiente filtro de la cadena
        $filterChain->execute();
        // Código que se ejecuta después de la ejecuciñon de la acción y antes de que se genere la vista
        ...
    }
}
```

La cadena de filtros por defecto se define en el archivo de configuración de la aplicación *filters.yml*. Este archivo lista los filtros que se ejecutan para cada petición. La cadena de filtros por defecto, se encuentra en *miaplicacion/config/filters.yml*

```
rendering:      ~
web_debug:      ~
security:       ~
# Generalmente, se insertan los filtros propios aquí
cache:          ~
common:         ~
flash:          ~
execution:      ~
```

Estas declaraciones no tienen parámetros (el caracter tilde, ~, significa null en YAML), porque heredan los parámetros definidos en el núcleo de Symfony. En su núcleo, Symfony define las opciones class y param para cada uno de estos filtros. Por ejemplo, los parámetros por defecto para el filtro rendering son .

```
rendering:
  class: sfRenderingFilter      # Clase del filtro
  param:                  # Parámetros del filtro
    type: rendering
```

Si se deja el valor vacío (~) en el archivo *filters.yml* de la aplicación, Symfony aplica el filtro con las opciones por defecto definidas en su núcleo. Se pueden personalizar las cadenas de filtros en varias formas; es activando algún filtro de la cadena, agregando un parámetro enabled: off. Por ejemplo, para desactivar el filtro de depuración web (web_debug), se añade:

```
web_debug:
   enabled: off
```

No se deben borrar las entradas del archivo *filters.yml* para desactivar un filtro ya que Symfony lanzará una excepción. Se pueden añadir declaraciones propias en cualquier lugar de la cadena (normalmente después del filtro security) para agregar un filtro propio. En cualquier caso, el filtro rendering debe ser

siempre la primera entrada, y el filtro execution debe ser siempre la ultima entrada en la cadena de filtros.

Redefinir la clase y los parámetros por defecto del filtro por defecto (normalmente para modificar el sistema de seguridad y utilizar un filtro de seguridad propio).

En Symfony se pueden crear sus propios filtros, que no son más que clases similares a las expuestas anteriormente, van ubicadas en el directorio *lib/* del proyecto general para la carga automática de las misma.

```
class rememberFilter extends sfFilter
{
  public function execute($filterChain)
   {
    // Ejecutar este filtro solo una vez
     if ($this->isFirstCall())
       {
        // Los filtros no tienen acceso directo a los objetos user y request.
        // Se necesita el contexto para obtenerlos
         $peticion = $this->getContext()->getRequest();
         $usuario = $this->getContext()->getUser();
         if ($peticion->getCookie('MyWebSite'))
           {
            // logueado
             $usuario->setAuthenticated(true);
           }
       }
    // Ejecutar el proximo filtro
    $filterChain->execute();
  }
}
```

2.2.10.4 Mecanismos de escape de los datos

Al desarrollar una aplicación dinámica en la cual se van insertar y sacar datos de una base de datos hay que lograr la integridad de los datos. En Symfony se emplea un mecanismo de escape denominado como *mecanismo de escape de los datos.*

Este mecanismo de escape es configurado en el archivo *settings.yml*, se controla con 2 parámetros:

1. La estrategia (escaping_strategy) define la forma en la que las variables están disponibles en la vista.

2. El método (escaping_method) indica la función que se aplica a los datos.

Para activar el mecanismo de escape para la opción escaping_strategy solamente hay que sustituir la palabra **bc** por **both**, ejemplo del mismo. (*miaplicacion/config/settings.yml*)

```
all:
.settings:
escaping_strategy: both
escaping_method: ESC_ENTITIES
```

Si se activa el mecanismo de escape, desde cualquier template se puede acceder a una nueva variable llamada `$sf_data`. Se trata de un objeto contenedor que hace referencia a todas las variables que se han modificado mediante el sistema de escape.

Mediante este objeto también se puede acceder a los datos originales o datos en crudo de la variable. Se trata de una opción muy útil por ejemplo cuando la variable contiene código HTML que se quiere incluir directamente en el navegador para que sea interpretado en vez de mostrado (solo se debería utilizar esta opción si se confía plenamente en el contenido de esa variable). Para acceder a los datos originales se puede utilizar el método getRaw ().

La opción escaping_strategy determina la forma en la que se muestra el contenido de las variables en las plantillas. Sus posibles valores son los siguientes:

- *bc*: (*backward compatible mode* o modo retrocompatible): El contenido de las variables no se modifica, pero el contenedor `$sf_data` almacena una versión modificada de cada variable. De esta forma, los datos de las variables se obtienen *en crudo*, a menos que se obtenga la versión modificada del objeto $sf_data. Se trata del valor por defecto de la opción, aunque se trata del modo que permite los ataques de tipo XSS.
- *both*: A todas las variables se les aplica el mecanismo de escape. Los valores también están disponibles en el contenedor `$sf_data`. Se trata del valor recomendado, ya que solamente se está expuesto al riesgo si se utilizan de forma explícita los datos originales. En ocasiones, se deben utilizar los valores originales, por ejemplo para incluir código HTML de forma que se interprete en el navegador y no se muestre el código HTML. Si una aplicación se encuentra medio desarrollada y se cambia la estrategia del mecanismo de escape a este valor, algunas funcionalidades pueden dejar de funcionar como se espera. Lo mejor es seleccionar esta opción desde el principio.

- *on*: los valores solamente están disponibles en el contenedor `$sf_data`. Se trata del método más seguro y más rápido de manejar el mecanismo de escape, ya que cada vez que se quiere mostrar el contenido de una variable, se debe elegir el método get() para los datos modificados o el método getRaw() para el contenido original. De esta forma, siempre se recuerda la posibilidad de que los datos de la variable sean corruptos.
- *off:* Deshabilita el mecanismo de escape. Las plantillas no pueden hacer uso del contenedor `$sf_data`. Si nunca se va a necesitar el sistema de escape, es mejor utilizar esta opción y no la opción por defecto bc, ya que la aplicación se ejecuta más rápidamente.

En el mecanismo de escape se utilizan los *helpers*, que son funciones que devuelven el valor modificado correspondiente al valor que se les pasa. Se pueden utilizar como valor de la opción escaping_method en el archivo *settings.yml* o para especificar un método concreto de escape para los datos de una vista. Los *helpers* disponibles son los siguientes:

- ESC_RAW: no modifica el valor original.
- ESC_ENTITIES: aplica la función htmlentities() de PHP al valor que se le pasa y utiliza la opción ENT_QUOTES para el estilo de las comillas.
- ESC_JS: modifica un valor que corresponde a una cadena de JavaScript que va a ser utilizada como HTML. Se trata de una opción muy útil para escapar valores cuando se emplea JavaScript para modificar de forma dinámica el contenido HTML de la página.
- ESC_JS_NO_ENTITIES: modifica un valor que va a ser utilizado en una cadena de JavaScript pero no le añade las entidades HTML correspondientes. Se trata de una opción muy útil para los valores que se van a mostrar en los cuadros de diálogo (por ejemplo para una variable llamada miCadena en la instrucción javascript: alert(miCadena);).

El mecanismo de seguridad que utiliza Symfony es bastante estable y confiable, ya que al mantener seguridad a nivel de acciones, establecer un mecanismo de escape para los datos introducido y filtros por los que van a pasar cada acción, es un sistema que está en constante vigilancia para cualquier usuario mal intencionado o usuario perdido en la navegación de la aplicación.

2.2.11 Cache en Symfony

Con el desarrollo de las aplicaciones ha surgido la necesidad de que las mismas tengan cada vez más un alto rendimiento, por lo que hacen uso de cache, la misma se puede definir en el lado del servidor , que en el del cliente, consiste en almacenar trozos de código HTML o incluso páginas enteras para poder servirlas en futuras peticiones. Symfony incluye un sistema de cache en el servidor muy flexible. Con este sistema es muy sencillo guardar en un archivo una página entera, el resultado de una acción, un elemento parcial o un trozo de plantilla. La configuración del sistema de cache se realiza en un archivo YAML [11].

El principio básico de las caches de HTML es muy sencillo: parte o todo el código HTML que se envía al usuario como respuesta a su petición se puede reutilizar en peticiones similares. El código HTML se almacena en un directorio especial (el directorio cache/) donde el controlador frontal lo busca antes de ejecutar la acción. Si se encuentra el código en la cache, se envía sin ejecutar la acción, por lo que se consigue un gran ahorro de tiempo de ejecución. Si no se encuentra el código, se ejecuta la acción y su respuesta (la vista) se guarda en el directorio de la cache para las futuras peticiones.

Como todas las páginas pueden contener información dinámica, la cache HTML está deshabilitada por defecto. El administrador del sitio web debe activarla para mejorar el rendimiento de la aplicación.

Symfony permite gestionar 3 tipos diferentes de cache HTML:

- Cache de una acción (con o sin layout)
- Cache de un elemento parcial, de un componente o de un slot de componentes
- Cache de un trozo de plantilla

Los dos primeros tipos de cache se controlan mediante archivos YAML de configuración. La cache de trozos de plantillas se controla mediante llamadas a *helpers* dentro de las propias plantillas.

2.2.11.1 Cache en la acción

Las acciones que muestran información estática y las acciones que leen información de una base de datos pero no la modifican son el tipo de acción ideal para almacenar su resultado en la cache. La

activación de la cache y las opciones para cada acción se definen en el archivo *cache.yml* del directorio *config/* del módulo.

Activando la cache de una acción:

```
listado:
enabled: on
with_layout: false   # Valor por defecto
lifetime: 86400      # Valor por defecto
```

La anterior configuración activa la cache para la acción listado y el layout no se guarda junto con el resultado de la acción. El sistema de cache también funciona para las páginas que utilizan parámetros. Se puede organizar de forma más clara el archivo *cache.yml* reagrupando las opciones comunes a todas las acciones del módulo bajo la clave all:

Ejemplo de *cache.yml* completo, en *miapp/modules/usuario/config/cache.yml*

```
listado:
enabled: on
ver:
enabled: on
all:
with_layout: false # Valor por defecto
lifetime: 86400 # Valor por defecto
```

Si el layout no depende por ejemplo de datos de sesión, es conveniente optar por la opción que guarda el layout en la cache. Desgraciadamente, el layout normalmente contiene elementos dinámicos, por lo que la opción habitual es la de no almacenar el layout en la cache.

2.2.11.2 Elemento parcial, componente o slot de componentes en la cache

Guardar un elemento parcial en la cache es tan sencillo como hacerlo en una acción y se activa de la misma forma. La cache para estos elementos se activa en el directorio *miapp/modules/usuario/config/cache.yml*, ejemplo del mismo:

```
_mi_parcial:
enabled: on

listado:
enabled: on
...
```

Todas las plantillas que incluyen este elemento parcial no ejecutan su código PHP, sino que utilizan la versión almacenada en la cache.

```
<?php include_partial('usuario/mi_parcial') ?>
```

La información que se guarda en la cache depende de los parámetros que se pasan al elemento parcial. El sistema de cache almacena tantas versiones diferentes como valores diferentes de parámetros se pasen al elemento parcial.

```
<?php include_partial('usuario/mi_otro_parcial', array('parametro' => 'valor')) ?>
```

2.2.11.1 Configuración dinámica de la cache

Como se ha dicho anteriormente el archivo cache.yml es el encargado de la cache, pero tiene un inconveniente ya que no puede ser configurado dinámicamente, pero se puede utilizar código PHP para hacer estos cambios dinámicamente, en vez de los archivos yml.

Donde se define las opciones de cache dinámica es en un filtro que es ejecutado antes del filtro sfCacheFilter. Todo el sistema de cache es un filtro de Symfony, como también lo son la barra de depuración de aplicaciones y las opciones de seguridad. Por ejemplo para habilitar la cache en la acción *articulo/ver* solo cuando el usuario no está autenticado, se crea el archivo conditionalCacheFilter en el directorio lib/ de la aplicación, este filtro se debe registrar en el archivo *filters.yml* antes de sfCacheFilter.

Un ejemplo de cómo configurar la cache mediante PHP

```
class conditionalCacheFilter extends sfFilter
{
   public function execute($filterChain)
    {
      $contexto = $this->getContext();
       if (!$contexto->getUser()->isAuthenticated())
         {
           foreach ($this->getParameter('pages') as $page)
             {
               $contexto->getViewCacheManager()->addCache($page['module'],
               $page['action'],array('lifeTime' => 86400));
             }
```

```
        }
    //Ejecutar el siguiente filtro
    $filterChain->execute();
   }
}
```

Para que la cache condicional pueda utilizarse, solo es necesario borrar la cache de Symfony para que se auto cargue la clase del nuevo filtro.

2.2.11.4 Cache súper rápida

Si se está completamente seguro de que una página no va a cambiar durante un periodo de tiempo, se puede saltar completamente Symfony, si se guarda en la carpeta web/ el código HTML completo de la página. Este funcionamiento es posible gracias a las opciones del módulo mod_rewrite de Apache, siempre que la regla de enrutamiento defina un patrón que no termine en ningún sufijo o en .html.

La cache de HTML, Symfony dispone de otros dos mecanismos de cache, que son completamente automáticos y transparentes para el programador. En el entorno de producción, la configuración y las traducciones de las plantillas se guardan automáticamente en la cache en los directorios *miproyecto/cache/config/* y *miproyecto/cache/i18n/*.

Los aceleradores PHP (eAccelerator, APC, XCache, etc.), también llamados módulos que guardan los *opcodes* en la cache, mejoran el rendimiento de los scripts PHP al guardar en una cache la versión compilada de los scripts, por lo que se elimina el procesamiento y compilación de los scripts cada vez que se ejecutan. Las clases de Propel contienen muchísimo código PHP, por lo que son las que más se benefician de esta técnica. Todos estos aceleradores son compatibles con Symfony y pueden fácilmente triplicar el rendimiento de cualquier aplicación. Se recomienda su uso en los servidores de producción de las aplicaciones utilizadas por muchos usuarios.

Existen comandos para el borrado de la cache, como es el caso de *clear-cache*, Si se modifican los datos de la aplicación, la información de la cache estará desfasada. Para evitar incoherencias y posibles errores, se pueden eliminar partes de la cache de varias formas en función de las necesidades de cada caso. Exciten diferentes comandos para borrar partes que uno desee de la cache como:

```
// Borrar toda la cache
> symfony clear-cache
// Atajo para borrar toda la cache
> symfony cc
// Borrar solo la cache de la aplicación miapp
> symfony clear-cache miapp
// Borrar solo la cache HTML de la aplicación miapp
> symfony clear-cache miapp template
// Borrar solo la cache de configuración de la aplicación miapp
> symfony clear-cache miapp config
```

También es necesario borrar la cache de las acciones cuando los datos de la base datos han cambiado e influyen con estas acciones, por lo que se puede crear un método que realice esta operación en la clase action, como se muestra a continuación.

```
public function executeModificar()
{
  // Modificar un usuario
  $id_usuario = $this->getRequestParameter('id');
  $usuario = UsuarioPeer::retrieveByPk($id_usuario);
  $this->foward404Unless($usuario);
  $usuario->setNombre($this->getRequestParameter('nombre'));
   ...
  $usuario->save();
  // Borrar la cache de las acciones relacionadas con este usuario
  $cacheManager = $this->getContext()->getViewCacheManager();
  $cacheManager->remove('usuario/listado');
  $cacheManager->remove('usuario/ver?id='.$id_usuario);
  ...
}
```

Si se utiliza la cache HTML, es necesario disponer de una visión clara de las dependencias y relaciones entre el modelo y las acciones, de forma que no se produzcan errores por no comprender completamente esas relaciones. Debe tenerse en cuenta que todas las acciones que modifican el modelo seguramente deben incluir una serie de llamadas al método remove() si se utiliza la cache HTML. Cuando la situación sea realmente complicada, siempre se puede borrar la cache entera cada vez que se actualiza la base de datos.

2.2.11.5 Estructura del directorio de la cache

El directorio *cache/* de cada aplicación tiene la siguiente estructura:

cache/ # sf_root_cache_dir
[nombre_aplicacion]/ # sf_base_cache_dir
[nombre_entorno]/ # sf_cache_dir
config/ # sf_config_cache_dir
i18n/ # sf_i18n_cache_dir
modules/ # sf_module_cache_dir
template/ # sf_template_cache_dir
[nombre_servidor]/
all/

2.2.12 Pruebas unitarias

Las pruebas unitarias aseguran que un único componente de la aplicación produce una salida correcta para una determinada entrada. Este tipo de pruebas validan la forma en la que las funciones y métodos trabajan en cada caso particular. Las pruebas unitarias se encargan de un único caso cada vez, lo que significa que un único método puede necesitar varias pruebas unitarias si su funcionamiento varía en función del contexto [11].

Symfony incluye su propio framework llamado Lime. Se basa en la librería Test::More de Perl y es compatible con TAP, lo que significa que los resultados de las pruebas se muestran con el formato definido en el "Test Anything Protocol", creado para facilitar la lectura de los resultados de las pruebas.

Las pruebas unitarias de Symfony son archivos PHP normales cuyo nombre termina en Test.php y que se encuentran en el directorio *test/unit/* de la aplicación. Su sintaxis es sencilla y fácil de leer.

En primer lugar, se instancia el objeto lime_test Cada prueba unitaria consiste en una llamada a un método de la instancia de lime_test. El último parámetro de estos métodos siempre es una cadena de texto opcional que se utiliza como resultado del método.

2.2.13 Otras características de Symfony

Actualmente ha ganado audiencia en el desarrollo de las aplicaciones Web, muchas empresas hoy en día utilizan este framework y sus opiniones son positivas acerca del mismo. Tiene un foro debate solamente para él, donde muchas personas plantean sus dudas y sugerencia acerca del empleo en

una aplicación. También tiene una amplia documentación y bien detallada, presenta un sitio en Internet donde se expone todo lo referente al mismo.

Unos de los aspectos más importantes que presenta este framework es que es libre, eso lo hace una herramienta más potente, ya que cualquiera puede configurarlo a su gusto y dar opiniones para mejorar su código. A continuación mencionamos alguna de las principales características que hacen deseable el uso del mismo para el desarrollo del ERP [11]:

- La capa de internacionalización que incluye Symfony permite la traducción de los datos y de la interfaz, así como la adaptación local de los contenidos.
- La capa de presentación utiliza plantillas y layouts que pueden ser creados por diseñadores HTML sin ningún tipo de conocimiento del framework. Los helpers incluidos permiten minimizar el código utilizado en la presentación, ya que encapsulan grandes bloques de código en llamadas simples a funciones.
- Los formularios incluyen validación automatizada y relleno automático de datos (“repopulation”), lo que asegura la obtención de datos correctos y mejora la experiencia de usuario.
- Los datos incluyen mecanismos de escape que permiten una mejor protección contra los ataques producidos por datos corruptos.
- La gestión de la caché reduce el ancho de banda utilizado y la carga del servidor.
- La autenticación y la gestión de credenciales simplifican la creación de secciones restringidas y la gestión de la seguridad de usuario.
- El sistema de enrutamiento y las URL limpias permiten considerar a las direcciones de las páginas como parte de la interfaz, además de estar optimizadas para los buscadores.
- El soporte de e-mail incluido y la gestión de APIs permiten a las aplicaciones Web interactuar más allá de los navegadores.
- Los listados son más fáciles de utilizar debido a la paginación automatizada, el filtrado y la ordenación de datos.
- Los plugins, las factorías (patrón de diseño “Factory”) y los “mixin” permiten realizar extensiones a medida de Symfony.
- Las interacciones con Ajax son muy fáciles de implementar mediante los helpers que permiten encapsular los efectos JavaScript compatibles con todos los navegadores en una única línea de código.

2.2.14 Vista seguridad

El despliegue de cualquier aplicación Web representa un reto en cuanto a temas de seguridad, no solo a nivel de aplicación, sino también que incluye los canales de comunicación, los protocolos de transporte, el hardware, etc. El ERP cubano no es la excepción, pero además debemos tener presente la carencia de recursos de la empresa cubana en sentido general exceptuando las empresas poderosas como ETECSA o CIMEX que no puede adquirir nodos de cómputos de última tecnología o en ocasiones tampoco cuenta con el presupuesto necesario para mejorar los recursos que posee.

Por las características de las operaciones y de la información que manipula toda empresa, es necesario en todo momento garantizar la seguridad de los mismos, así como toda operación comercial y financiera.

Por ello es importante que la aplicación sea desplegada en una infraestructura de clave pública (PKI, Public Key Infrastructure) que es una combinación de hardware y software, políticas y procedimientos de seguridad que permiten la ejecución con garantías de operaciones criptográficas como el cifrado, la firma digital o el no repudio de transacciones electrónicas.

Principales características de la infraestructura de clave pública

- Autenticación de usuarios y sistemas (*login*)
- Identificación del interlocutor
- Cifrado de datos digitales
- Firmado digital de datos (documentos, software, etc.)
- Asegurar las comunicaciones
- Garantía de no repudio (negar que cierta transacción tuvo lugar)

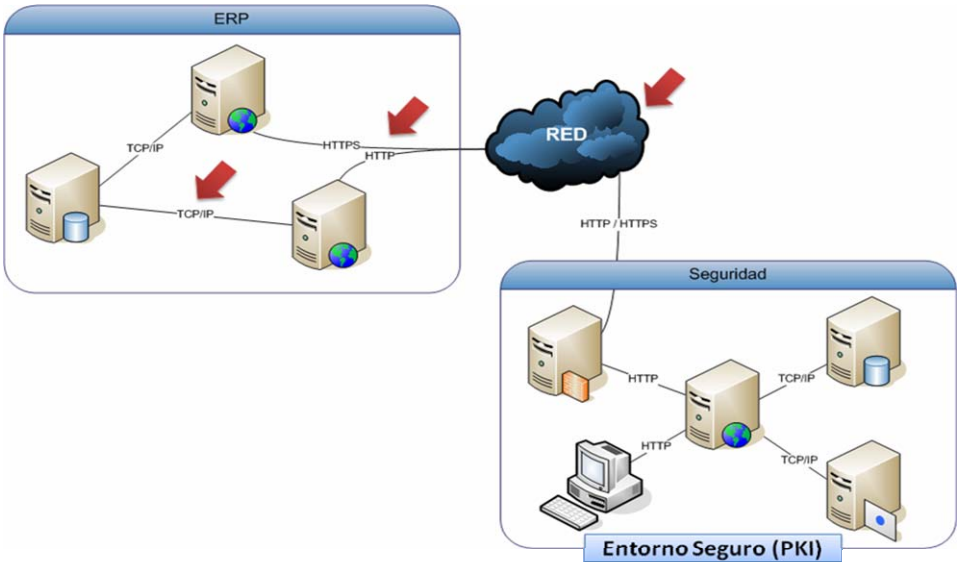


Figura 6 Entorno seguro

2.2.14.1 Sistema de seguridad

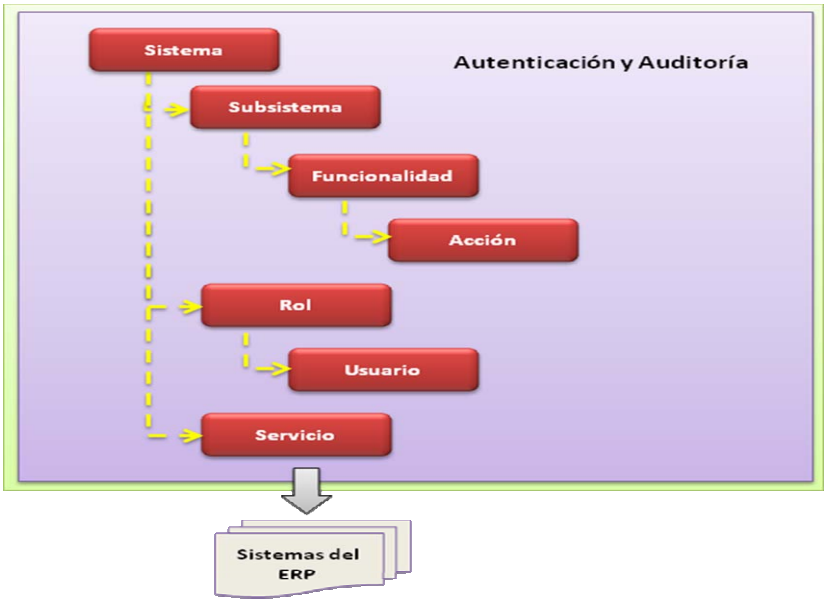


Figura 7 Sistema de seguridad

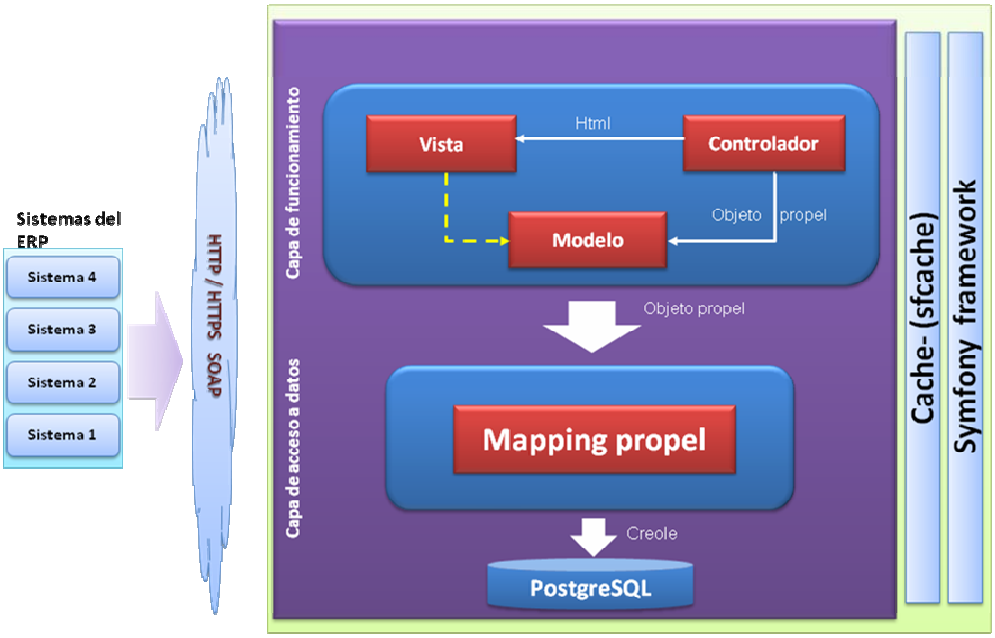


Figura 8 Arquitectura del sistema de seguridad

2.2.14.2 Otros mecanismos de seguridad

Todo lo referente al tema de la seguridad es poco lo que se pueda hacer, por eso se tuvieron en cuenta algunos elementos adicionales como:

- Seguridad en el cliente
- Seguridad de Apache
- Seguridad de PostgreSQL

Seguridad en el cliente

XSS

- Desactivar el uso de scripting en el navegador.
- Usar el mehotd POST y no GET (filtrar en el lado del servidor).
- Filtrar el contenido de las variables.

  Para evitar XSS en Symfony se usa el mecanismo de escape
- Realizar validaciones en el “cliente” y en el “servidor”.

SQL Injection

- Principio del mínimo privilegio.
- Consultas concretas.
- Filtrar errores.
- Comprobar parámetros de entrada.

Seguridad de Apache

Autenticación:

- Gestionando ficheros con listas de usuarios y contraseñas (encriptadas): **mod_auth**.
- Base de datos: **mod_auth_dbm**.

Autorización:

- Directiva <directory> en el fichero principal de configuración.
- Configuración de la carpeta a través de ficheros .htaccess.

Control de acceso:

- Directivas <directory><files>y <location>
- Fichero de configuración .htaccess para controlar una carpeta especifica.

Encriptarción:

- Mediante el módulo **mod_ssl**

Seguridad PostgreSQL

Protección de los ficheros de la base de datos:

Todos los ficheros almacenados en la base de datos deberán estar protegidos contra escritura por cualquier cuenta que no sea la del superusuario de Postgres.

Conexiones de los clientes al servidor:

- Permitidas por defecto únicamente mediante sockets Unix locales y no mediante sockets TCP/IP
- Se pueden restringir por dirección IP y/o por nombre de usuario mediante el fichero **pg_hba.conf**
- Los usuarios no deberán tener permiso de escritura a bases de datos que no hayan creado.

- El acceso a las tablas puede restringirse en base a grupos.

Otras consideraciones

- Seguridad en las contraseñas (Historial y Restricciones)
- Chequeos de integridad de los datos y del código (CheckSum)
- Seguridad dependiente de otros sistemas (SGIS)
- Mecanismo interno de salvas y salvas automáticas
- Actualización automática de la Bitácora del sistema y del servidor
- Se valorará el uso de OpenSSL y OpenSSH
- Ayudan al sistema a implementar el Secure Sockets Layer (SSL), así como otros protocolos relacionados con la seguridad, como el Transport Layer Security (TLS).
- Este paquete de software es importante para usar cierto nivel de seguridad en su máquina con un sistema operativo Libre basado en GNU/Linux.

2.2.15 Vista de despliegue

2.2.15.1 Clientes ligeros

Los clientes ligeros son aquellas empresas que por su envergadura, son consideras pequeñas dentro del mundo empresarial. Ver Anexo 4.

2.2.15.2 Centros de gestión contable

Los centros de gestión contable son aquellas empresas con un mayor peso dentro de la empresa cubana, donde el cúmulo de operaciones es mayor. Ver Anexo 4.

2.2.15.3 Empresas

Las empresas ya van a tener un despliegue mucho mayor por la cantidad de información que se manipula en la misma además de tener mayores recursos tecnológicos. Ver Anexo 4.

2.3 Herramientas horizontales

2.3.1 Sistema operativo

Unos de los objetivos principales en este trabajo es realizar el diseño de la arquitectura del sistema ERP cubano completamente en software libre, ya que esto facilita más el desarrollo de la misma, ya que podemos usar cualquier herramienta sin tener la preocupación del pago de una licencia y los costos que esto provoca.

GNU/Linux es un sistema operativo libre y con el avance de los años muchas compañías se han dedicado a perfeccionar este sistema operativo, ya que tiene un gran rival en el mercado (*Windows*), la evolución del mismo se ha basado más en la interfaz de usuario, ya que muchos de estos no saben navegar por el interior de este sistema operativo. Gracias a eso han surgido muchas distribuciones de GNU/Linux, mejorando grandemente la interfaz del sistema operativo y la interacción con los usuarios finales. Ejemplo:

1. Familia de Ubuntu.
2. Debian
3. Knoppix
4. Gentoo
5. Suse

2.3.1.1 ¿Por qué usar Ubuntu 7.10?

Son muchos las distribuciones que han surgido de GNU/Linux, una de ellas es la familia de Ubuntu. El sistema operativo propuesto para la producción es Ubuntu 7.10 pues es una de las distribuciones de GNU/Linux más usadas, goza de gran prestigio y popularidad dentro de las distribuciones de GNU/Linux, el mismo proporciona una serie de ventajas como [18]:

- Bajo costo: Al no cobrarse una licencia de uso, se garantiza que haya una reducción en los costes totales de su implantación con respecto al software privado. El resto de servicios que acompañan a la implantación, como podrían ser la formación, adaptación, soporte técnico, suscripción para actualización del sistema Linux, etc.

- Competitividad: Al no cobrarse por licencia de uso, se permite un abaratamiento de los precios, con lo cual aumenta la competitividad de la empresa que ofrece productos y servicios en torno al Software Libre. El sector informático y la región donde se promueve este tipo de software se hacen más competitivos.
- Independencia tecnológica: Independencia parcial de cualquier sector privado o empresa. Esto supone no estar atado a las condiciones del mercado impuestas por ciertas empresas en situación de monopolio, y el sometimiento a ciertas multinacionales norteamericanas. Hay que asumir que los centros educativos deberían enseñar con Software Libre, porque no se puede favorecer solo a una determinada empresa extranjera.
- Seguridad y privacidad: Al disponer siempre del código fuente, se conocerá siempre su funcionamiento interno, y se encontrarán y corregirán mucho antes los posibles errores y otros problemas de seguridad. Actualmente Linux es inmune a la inmensa mayoría de virus informáticos, que afectan casi exclusivamente a los sistemas Windows.
- Adaptabilidad y actualización: Las modificaciones y posibles correcciones se realizan de forma inmediata debido a la posibilidad vía Internet de colaboración mundial entre empresas, instituciones y usuarios expertos, estando en continua mejora y proceso de evolución.
- Calidad: El software libre, al ser de dominio público, está siendo constantemente usado y depurado por un gran numero de desarrolladores y usuarios, que añaden y demandan continuamente nuevas funcionalidades, y que están al alcance de los demás miembros de la comunidad.
- Respeto a los estándares: Facilita la interoperabilidad entre distintos sistemas, siendo este aspecto fundamental para las Administraciones Públicas del estado, dada la gran cantidad de unidades distintas de informática con sus propios procesos internos. Ha sido igualmente importante el hecho de que los protocolos que definen la arquitectura de Internet sean abiertos y que no hayan sido controlados por ninguna empresa, siendo ahora estándares importantes. También el uso de formatos de documento propietarios de una determinada empresa, obliga al resto a tener que comprar su software para poder leer y usar esos documentos, cosa que no ocurre en el Software Libre.
- Redistribución: Cualquier mejora y cambio que se introduzca en programas de software libre debe ser incluido en versiones posteriores y publicadas sus códigos fuente. Así el desarrollo tecnológico esta garantizado, es continuo y dinámico, y toda la sociedad se beneficia de él. No hay restricción legal en el número de copias que se pueden repartir.

- Continuidad: El hecho de que el código este libre para todo el mundo, garantiza el derecho de cualquier persona, empresa o institución, a continuar con su desarrollo de por vida. No ocurre lo mismo con el software privado, donde los desarrollos son propios del fabricante.
- Integración, fiabilidad y escalabilidad: Los sistemas de Software Libre como Linux funcionan sin necesidad de reinicio cada cierto tiempo, como le ocurre al software privativo, por lo tanto son funcionales todos los días del año. Además se puede integrar fácilmente con otras soluciones informáticas; gracias al acceso al código fuente para ver cómo funciona es más fácil adaptarlo a un sistema distinto. También es posible escalarlo, ya que el diseño está formado por módulos de software, que funcionan en multitud de plataformas diferentes y que permite ir aumentando su número según las necesidades.

Son innumerables las ventajas y beneficios que posee la utilización de Ubuntu.

- Configurar una red en Ubuntu es lo más sencillo que te puedas imaginar. Posee herramientas de redes para configurarla de manera fácil, rápida y sobre todo segura, ya que posee IPTABLES que es el firewall más seguro y viene con el kernel.
- Actualizaciones de Seguridad muy seguidas.
  ¿Eso quiere decir que Ubuntu es poco seguro? Todo lo contrario, Ubuntu es una distribución muy segura y enfocada al entorno del usuario, estas actualizaciones se dan según un programa cambie de versión. En pocas palabras, siempre estarás actualizado con Ubuntu.
- Gestor de Paquetes Gráfico, esta puede ser una de las razones más contundentes.
- Actualizaciones cada 6 meses. Cada seis meses Ubuntu tiene actualizaciones en su distribución.
- Soporte de todo tipo de plataforma. Ubuntu esta disponible para i386 (386/486/Pentium(II/III/IV) y procesadores Athlon/Duron/Sempron), AMD64 (Athlon64, Opteron, y el procesador Intel 64-bit), y arquitecturas Sun UltraSPARC [12].

2.3.2 Gestión de proyecto

Actualmente dotProject es una herramienta orientada a la Gestión de Proyectos. Para eso se orienta a la administración de recursos para desarrollar un producto, cuya producción requiera de un conjunto de actividades o tareas que se desarrollen entre ellas en forma paralela o independiente. Esta herramienta te facilitará enormemente la gestión de proyectos en los que estén implicadas varias

personas, pudiendo asignar permisos muy concretos a cada uno de ellos y ocultando información que no sea necesaria para otros equipos de trabajo.

Muy indicado para equipos de trabajo con sedes en distintos puntos geográficos. Además disponer de foro asignado a cada proyecto, facilita tener una base de conocimiento para nuevas incorporaciones a los proyectos.

Los responsables de los proyectos son automáticamente informados vía e-mail de los avances en las tareas según se van produciendo cambios.

Dispones además de históricos de los partes de trabajo generados para cada una de las tareas, sirviendo para analizar fácilmente los costos de la gestión de determinadas tareas o proyectos. Las zonas más importantes de la herramienta son:

Compañías: Son las entidades que agrupan proyectos, actividades y usuarios.
Departamentos: Son áreas dentro de las compañías, que permiten agrupar usuarios en dicho nivel.
Usuarios/Contactos: dotProject tiene usuarios los cuales son capaces de autenticarse a dotProject y trabajar dentro del esquema de permisos que posea el rol de dicho usuario. Los contactos son usuarios especiales que asignados a un determinado proyecto pueden recibir por ejemplo: correo, actualizaciones y noticias pero no necesariamente deben tener acceso al sistema dotProject. Los usuarios y contactos pertenecen a una compañía.
Proyectos: Es la entidad que contiene el grupo de tareas necesarias para desarrollar un determinado producto.
Actividades: son las tareas asignadas dentro de un proyecto. Son los componentes sobre los cuales se controla: la duración, dependencias, recursos asignados y progreso. Las actividades deben de pertenecer a un único proyecto.
Diagramas de Gantt: Permite ver en forma grafica las actividades ordenadas jerárquicamente, mostrando las dependencias y solapamientos de las mismas.
Tickets: para administrar todos los problemas relacionados a un proyecto.
Archivos: Permite almacenar archivos dentro de un proyecto permitiendo un versionado básico de los mismos.
Foros: Permite la creación de foros de discusión dentro de cada proyecto para distribuir información y discutir temas relativos al proyecto del foro.

Administración del sistema: Contiene la actividades relacionadas a la administración de usuarios, roles y configuración del sistema.
Recursos: Permite asignar recursos no humanos (oficinas, equipamiento, etc) a un proyecto.
Filosofía del proyecto:

- Proveer a los usuarios de funcionalidad orientada a la Gestión de Proyectos.
- Construir una herramienta con una interfaz de usuario simple, clara y consistente.
- Ser de código abierto, libre acceso y utilización. [19]

2.3.3 Servidor de aplicaciones

Un servidor de aplicaciones es un software que ayuda al servidor Web a procesar las páginas que contienen scripts o etiquetas del lado del servidor. Cuando un navegador solicita una página de este tipo, el servidor web remite la página al servidor de aplicaciones para su procesamiento antes de enviarla al navegador. El servidor de aplicaciones será PHP 5.

2.3.4 Lenguaje de programación

Unos de los lenguajes con mayor prestigios en el desarrollo de aplicaciones web es PHP, el mismo presenta un gran soporte en números gestores de bases datos como: MicrosoftSQL server, LDAP, MySQL, ODBC, Oracle, PostgreSQL, Sybase, etc. Ofrece la integración con varias bibliotecas externas, que permiten que el desarrollador haga casi cualquier cosa desde generar documentos en formato digital portable hasta analizar código XML.

Da una solución simple y universal para las paginaciones dinámicas del web de fácil programación. Su diseño elegante lo hace perceptiblemente más fácil de mantener y ponerse al día que el código comparables en otros lenguajes.

Debido a su amplia distribución PHP esta perfectamente soportado por una gran comunidad de desarrolladores. Como producto de código abierto, PHP goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y se reparan rápidamente. El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP. Es utilizado en aplicaciones Web-relacionadas por algunas de las

organizaciones más prominentes tales como Mitsubishi, Redhat, Der Spiegel, MP3-Lycos, Ericsson y NASA.

2.3.5 Sistema gestor de base de datos

El Sistema ERP cubano, como software de gestión, necesita un repositorio de información, donde almacenar los datos necesarios para realizar las distintas operaciones.

La utilización del estilo de arquitectura en capas permite que se abstraiga la lógica del negocio al almacenamiento de los datos, la capa de acceso a datos contiene toda la lógica de acceso, ya sea consultas y procedimientos almacenados, dejando a la Base de Datos como simple almacén de datos sin ninguna lógica.

El gestor de base de datos propuesto para esta arquitectura es PostgresSQL, ya que está considerado como la base de datos de código abierto más avanzada del mundo. PostgreSQL proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales tales como DB2 u Oracle [5]:

DBMS Objeto-Relacional: PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia, y arreglos.

Altamente_Extensible: Soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario.

Soporte_SQL_Comprensivo: Soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.

Integridad Referencial: Soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.

API Flexible: La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas interfacaes incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike.

Lenguajes Procedurales: Tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.

2.3.6 Servidor web

El servidor usado va a ser Apache, que por su robustez, configurabilidad y estabilidad hacen que cada vez más millones de servidores reiteren su confianza en este programa y su licencia es descendiente de la BSD, que permite la modificación del código fuente. Ha alcanzado gran popularidad en ámbitos empresariales debido, entre otras características a:

- Corre en una multitud de sistemas operativos, lo que lo hace prácticamente universal.
- Apache es una tecnología gratuita de código fuente abierta. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto.
- Apache es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar sus capacidades mediante la integración o la construcción de módulos.
- Apache te permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.

2.3.7 Entorno de desarrollo integrado (IDE)

El IDE propuesto para la creación de esta aplicación es Eclipse PDT, por cualidades antes expuestas, además que Eclipse da soporte a todo tipo de proyectos que abarcan desde el ciclo de vida del desarrollo de aplicaciones, incluyendo soporte para modelado. La arquitectura de plugins permite integrar diversos lenguajes sobre un mismo IDE e introducir otras aplicaciones accesorias. Conservan el registro de las versiones, generan y mantienen la documentación de cada etapa del proyecto.

Eclipse PDT (PHP Development Tools) es un IDE para PHP basado en Eclipse. Esta versión incluye todos los componentes necesarios (Eclipse 3.3, GEF 3.3, DTP 1.5, WTP 2.0, etc.), incluye [9]:

- Integración con e-mail
- Soporte de plantillas Smarty
- Bloques de plantillas PHPDoc
- Soporte de anotación
- Problemas PHP - adiciona filtros
- Sintaxis coloreada – opciones adicionales
- Soporte para refactoring
- Generación de clases a partir de interfaces
- Permite adicionar comentarios en forma de notas
- Integracion HTML, CSS y JavaScript
- Permite adicionar, editar listas de códigos.
- Auto activación del asistente de código para las clases
- Integración con PHPUnit

Presenta requisitos como:

- Editor sensible al contexto, el cual provee de resaltamiento de código, asistente de código y autocompletado de código.
- Integración con el modelo del proyecto Eclipse, que permite para inspeccionar el uso de las vistas del contorno del fichero y del proyecto, así como la nueva vista PHP Explorer.
- Soporte para el debug incremental del código de PHP
- Extensos frameworks y APIs que permiten a los desarrolladores e ISVs (vendedores de software independientes) fácilmente extender PDT para crear nuevas e interesantes herramientas orientadas al desarrollo de PHP.

2.4 Conclusiones

El propósito de desarrollar la arquitectura de software es, primeramente crear la estructura o el esqueleto del sistema que cohesiones las funcionalidades más criticas y relevantes, y que sirva de soporte a al resto de las funcionalidades finales. Luego realizar el acoplamiento de las distintas partes. Todo esto hace del diseño de un software particular una tarea generalmente única. La arquitectura de software constituye un documento guía, que se define a partir del rendimiento o de la calidad, donde se

especifica detalladamente los principales componentes que integran el sistema y como se relacionan entre si, facilitando el entendimiento de cada uno de los miembros del equipo de desarrollo.

CAPÍTULO 3: EVALUACIÓN DE LA ARQUITECTURA PROPUESTA

3.1 Introducción

Al final del desarrollo del software se conoce si éste cumplió o no con al menos un atributo de calidad que se especificaron en los requerimientos no funcionales, lo que implica tomar demasiados riesgos innecesarios. Para evitar estos riesgos es recomendable realizar evaluaciones a la arquitectura durante su diseño.

Realizar un buen diseño de arquitectura de software garantiza que el sistema cumpla con uno o varios atributos de calidad, este diseño puede determinar el éxito o el fracaso de un sistema de software

3.2 Evaluación de la arquitectura de software

Uno de los factores que determina el éxito o el fracaso de software es su arquitectura, es decir, la estructura del sistema. Si la arquitectura ha sido bien diseñada, entonces, garantiza que el sistema cumpla con varios atributos de calidad, como por ejemplo confiabilidad, seguridad, etc.

3.2.1 ¿Por qué es necesario evaluar una arquitectura de software?

Evaluar una arquitectura de software sirve para prevenir todos los posibles desastres de un diseño que no cumple con los requerimientos de calidad y para saber que tan adecuada es la arquitectura de software diseñada para el sistema. Una evaluación de una arquitectura no da un sí o un no, si es buena o mala, o una calificación, expresa donde está el riesgo, es decir, fortalezas y debilidades identificadas de la arquitectura.

Después de la evaluación de una arquitectura de software, se pueden tomar algunas decisiones como: si se puede seguir el proyecto con las áreas de debilidad dadas en la evaluación, si hay que reforzar la arquitectura de software o si hay que comenzar de nuevo toda la arquitectura.

3.2.2 ¿Cuándo es recomendable evaluar la arquitectura?

La evaluación clásica de la arquitectura se realiza cuando ésta se encuentra especificada totalmente y no se ha iniciado su implementación. La arquitectura puede ser evaluada en cualquier momento de desarrollo.

Existen dos variaciones útiles para realizar esta evaluación, la evaluación temprana y la evaluación tardía [14].

La evaluación temprana

Para realizar esta evaluación no es necesario que la arquitectura se encuentre completamente especificada. Esta evaluación permite efectuar decisiones sobre la arquitectura en cualquier nivel, puesto que se pueden imponer cambios arquitectónicos producto de una evaluación en función de los atributos de calidad esperados.

La evaluación tardía

Se realiza cuando la arquitectura del sistema se encuentra establecida y se ha terminado su implementación, es decir, en el momento de adquisición de un sistema ya terminado. Se considera por parte de los autores que la evaluación en este punto es muy importante y útil, puesto que puede observarse el cumplimiento de los atributos de calidad asociados al sistema y cómo será su comportamiento general.

La evaluación de la arquitectura de software debe realizarse cuando esta contiene suficientes elementos como para justificarla. Una buen momento para determinar cuando realizar la evaluación es cuando el equipo de desarrollo comienza a tomar decisiones que dependen de la arquitectura y que el costo de no tenerlas en cuenta son mayores que el costo de realizar una evaluación.

3.2.3 ¿Quiénes participan en la evaluación?

Generalmente, las evaluaciones son realizadas por miembros del equipo de desarrollo, por ejemplo el arquitecto, el diseñador y el administrador de proyecto. Aunque pueden existir excepciones, donde las pruebas son realizadas por un grupo de especialistas.

El cliente también se interesa por las evaluaciones, pues en dependencia de los resultados obtenidos durante las pruebas puede decidir si se continua o no con el proyecto.

3.2.4 Resultado de la evaluación

La evaluación de la arquitectura produce un informe, el cual varía en dependencia del método utilizado. Este informe produce respuesta a dos tipos de preguntas.

- ¿Es esta arquitectura adecuada para el sistema para el cual fue diseñada?
- ¿Cual de las arquitecturas propuestas es la más adecuada para el sistema?

Una arquitectura es adecuada cuando cumple dos criterios

- El sistema resultante cumple con los atributos de calidad.
- El sistema puede ser construido con los recursos disponibles, es decir, es construible.

La evaluación de la arquitectura no produce resultados cuantitativos, no es de interés conocer la cantidad de transacciones por segundos debido a que el sistema aún no está implementado. Lo que interesa es aprender como un atributo de calidad es afectado por una decisión de diseño arquitectónico.

3.2.5 ¿Por qué cualidades puede ser evaluada una arquitectura?

Los atributos de calidad son requerimientos adicionales del sistema que hacen referencia a características que éste debe satisfacer.

Algunos de los atributos por los cuales puede ser evaluada una arquitectura son:

- Disponibilidad (Availability). Es la medida de disponibilidad del sistema para el uso.
- Desempeño (Performance). Es el grado en el cual un sistema o componente cumple con sus funciones designadas, dentro de ciertas restricciones dadas, como velocidad, exactitud o uso de memoria. (IEEE 610.12).

- Confiabilidad (Reliability). Es la medida de la habilidad de un sistema a mantenerse operativo a lo largo del tiempo
- Seguridad (Security). Es la mediada de la habilidad del sistema para resistir a intentos de uso no autorizados y negación de servicios, mientras se sirve a usuarios legítimos.
- Portabilidad (Portability). Es la habilidad del sistema de ser ejecutado en diferentes ambientes de computación. Estos ambientes pueden ser hardware, software o una combinación de los dos.

3.3 Técnicas de evaluación

Existen varias técnicas que permiten realizar evaluaciones a la arquitectura, estas se clasifican en cualitativas y cuantitativas.

En las técnicas de evaluación cualitativas se pueden utilizar escenarios, cuestionarios o listas de verificación. Mientras que en las técnicas de evaluación cuantitativas se pueden emplear métricas, simulaciones, prototipos, experimentos o modelos matemáticos.

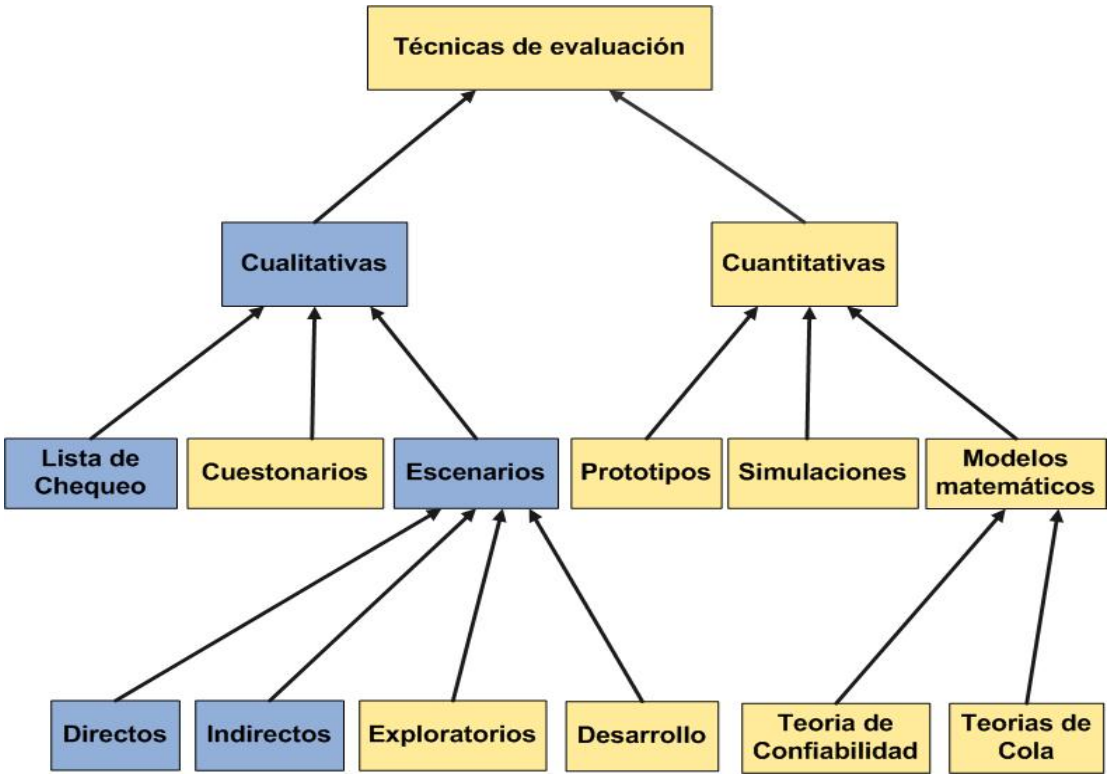


Figura 9 Técnicas de evaluación

La mayoría de los métodos de evaluación utilizan escenarios, que son secuencias específicas de pasos que involucran el uso o la modificación del sistema. Por lo regular, las técnicas de evaluación cualitativas son usadas cuando la arquitectura se encuentra en construcción, mientras que las técnicas de evaluación cuantitativas, se usan cuando la arquitectura ya ha sido implantada.

3.4 Evaluación de la arquitectura de software propuesta

La solución de arquitectura que se propone en este trabajo se encuentra especificada a alto nivel, es decir, un diseño poco detallado. Esta especificación a alto nivel impide la selección de escenarios que permitan validar si la arquitectura diseñada satisface a requisitos específicos. Como resultado de esto se realizará la evaluación atendiendo a como responden algunos elementos ante los principales atributos de calidad. Para ello se utilizó el método de evaluación ATAM (Analysis Tradeoff Architecture Method).

3.4.1 ATAM. Principales características

ATAM permite determinar el impacto de los atributos de calidad en el arquitectura, además provee un entendimiento interno, de cómo interactúan los principales objetivos de calidad. Cuando se evalúa usando ATAM, el objetivo es entender las consecuencias de las decisiones arquitecturales que respectan a los requerimientos de los atributos de calidad del sistema [].

ATAM permite determinar si los objetivos concebidos se pueden alcanzar por la arquitectura propuesta. El método de análisis de arquitectura usado por ATAM permite que el análisis se pueda repetir. Además permite asegurar que las preguntas fundamentales respecto a los temas de arquitectura puedan ser respondidas tempranamente durante los requerimientos y los escenarios de diseño, cuando ocurre algún problema puede ser solucionado con bajos costos. Además provee a los stakeholders encontrar un conflicto y la solución en la arquitectura de software.

¿Cuál es el propósito de ATAM?

Evaluar las consecuencias de las decisiones arquitecturales tomadas en consideración a los atributos de calidad requeridos. ATAM es un método para identificar riegos, esto significa que se detecta áreas

de riesgos potenciales dentro de la arquitectura de un complejo sistema de software. Esto tiene algunas implicaciones [20]:

- ATAM debe ser ejecutado tempranamente en el ciclo de desarrollo del software.
- Debe ejecutarse relativamente con un bajo costo y rápido, ya que evalúa los artefactos del diseño arquitectónico.
- ATAM producirá un análisis en proporción con el nivel de detalle de la especificación de la arquitectura. Además no siempre cuando se obtiene un análisis detallado de los atributos de calidad medibles de un sistema. (ej. latencia) puede ser el éxito. En cambio el éxito es lograr identificar las amenazas potenciales para el sistema.

Este último aspecto es crucial para entender los objetivos de ATAM, donde el objetivo no es predecir exactamente el comportamiento de un atributo de calidad. Esto será imposible en etapas tempranas en el escenario de diseño, porque todavía no se tiene suficiente información para hacer esta predicción. Es por ello que ATAM se centra en la identificación de riesgos.

Es sumamente importante en ATAM registrar cualquier riesgo, punto sensible y puntos de intercambio.

Los riesgos son decisiones arquitecturalmente importantes, que no han sido tomadas (ej. El equipo de arquitectura no ha decidido como distribuir el tiempo o no ha decidido si usará base de datos relacional o base de datos Orientada a Objetos) o decisiones que han sido tomadas pero las consecuencias no han sido entendidas a plenitud (ej. el equipo de arquitectura ha decidido incluir una capa de portabilidad en el sistema operativo, pero no están seguro que función usar para acceder dentro de la capa).

Los puntos sensibles son parámetros en la arquitectura en los cuales la respuesta medible de algunos atributos de calidad son altamente correlacionados (ej. se puede determinara que la cantidad total de datos trasmitido en un sistema es altamente correlacionar con la cantidad de datos que se transmiten por un canal de comunicación en especifico).

Un punto de intercambio (tradeoff) es descubierto en la arquitectura cuando un parámetro de construcción arquitectural es un anfitrión para más de un punto sensible donde los puntos de calidad medibles son afectados indistintamente por cambio en el parámetro, (ej. si se aumenta la velocidad en

el canal de comunicación mencionado anteriormente esto mejoraría en flujo de datos pero reduciría la confiabilidad, entonces la velocidad del canal es un punto de intercambio (tradeoff).

La idea de caracterizar un atributo de calidad es un concepto clave sobre el cual se ha fundado ATAM así como:

- Los escenarios
- El tercer concepto sobre el cual se basa ATAM es el conocimiento sobre estilos arquitectónicos (ABAS).

ATAM es un método de análisis organizado alrededor de la idea que los estilos de arquitectura son determinantes de los atributos de calidad arquitectónicos. Los pasos para aplicar el método se explican continuación:

Presentación

1. Presentar el ATAM. Se describe el método a los stakeholders(Clientes representativos, Equipo de arquitectura, etc)
2. Presentar la directriz del negocio: El jefe del proyecto describe las metas del negocio y en que lugares se debe centrar el esfuerzo, cual va ser la directriz que guiará la arquitectura (ej. alta disponibilidad y alta seguridad)
3. Presentar arquitectura: El arquitecto describe la arquitectura propuesta, centrándose como va a estar direccionada hacia el negocio.

Investigación y análisis

4. Identificar estrategias arquitectónicas. Estas son identificadas por el arquitecto pero no son analizadas.
5. Generar árbol de utilidad de atributos de calidad. Los factores de calidad que comprenden la utilidad del sistema (rendimiento, disponibilidad, seguridad modificabilidad, etc.) son capturados, especificados bajo un nivel de escenarios, comentados con estímulos, respuestas y priorizados.
6. Analizar estrategias arquitectónicas. Basada en los factores de mayores prioridad detectado en el paso 5, las estrategias arquitectónicas que guían esos factores son capturados y analizados

(ej. una estrategia arquitectónica puede apuntar al entendimiento de las metas de rendimiento y esto puede ser subordinado a un análisis de rendimiento). Durante este paso son detectados los riesgos arquitectónicos, los puntos sensibles y los puntos de intercambio.

Pruebas

7. Tormenta de ideas y priorizar los escenario. Basado en los escenarios obtenidos en el árbol de utilidad del paso 5, se agranda el número de escenarios obtenidos del grupo de stakeholders. Este grupo de escenarios es priorizado mediante un proceso de votación que involucra a todo los stakeholders.
8. Analizar estrategia arquitectónica. Este paso repite el paso 6 , pero aquí se resalta los escenarios obtenidos en el paso 7.

Reporte

9. Presentar resultados.

En esta evaluación se analizarán los siguientes atributos:

- Portabilidad
- Modificabilidad
- Seguridad

3.4.2 Resultados de la evaluación propuesta

3.4.2.1 Árbol de utilidad

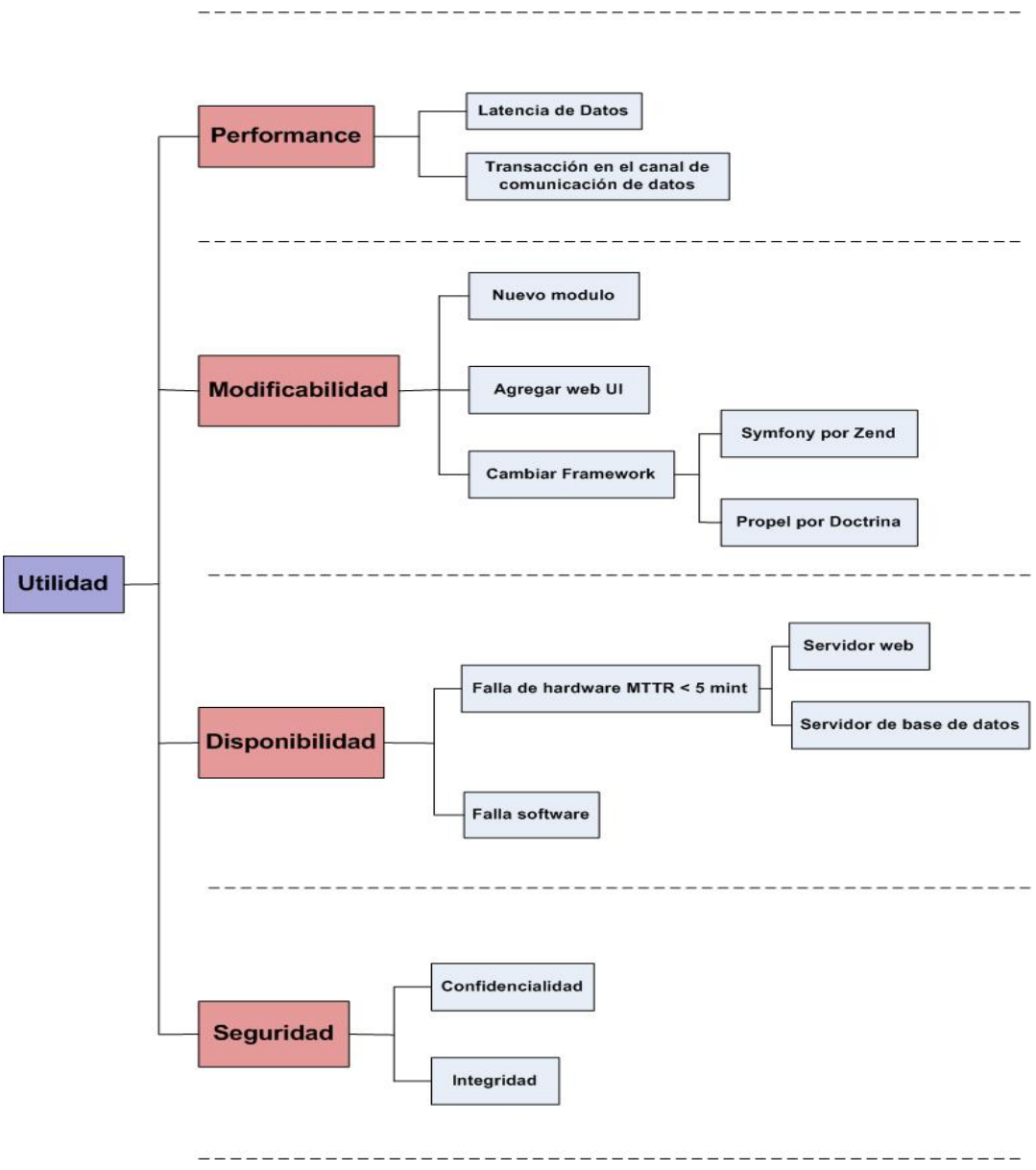


Figura 9 Árbol de utilidades

Después de haberse realizado un análisis se llegó a la conclusión que los principales atributos de calidad que van a afectar al proyecto son los descritos en el árbol de utilidad. Por eso podemos decir que el sistema ERP cubano va a estar afectado por estos atributos. Entonces la calidad del sistema ($\mathbf{C}_{erp}$) se puede decir que es una función **f,** que es afectada por la calidad que tenga el desempeño ($\mathbf{C}_{o}$), la modificabilidad ($\mathbf{C}_{m}$), la disponibilidad ($\mathbf{C}_{d}$) y la seguridad ($\mathbf{C}_{s}$).

$$\mathbf{C}_{erp} = \mathbf{f}(\mathbf{C}_{o},\mathbf{C}_{m},\mathbf{C}_{d},\ \mathbf{C}_{s})$$

A continuación se presenta algunos escenarios arquitectónicos que son de interés dentro del sistema ERP cubano.

Escenario	Cuando la capa de funcionamiento le solicita información a la capa de acceso a datos.
Objetivos del negocio	Procesar algún dato para persistirlo u obtenerlo de la base de datos.
Atributo de calidad	Mantenibilidad, modificabilidad, flexibilidad.
Estímulo	Obtener una colección de objetos
Origen del estímulo	Un gestor
Elemento	Un método
Ambiente	Se está realizando un proceso de búsqueda
Respuesta	Se retorna una la colección de objetos
Medida de la respuesta	Nivel de abstracción, repercusión
Preguntas	¿Cuáles y cómo se afectan los componentes involucrados en la petición?
Anotación	Propel inicialmente implementa el patrón entrada de datos en fila. También genera las clases para cada tabla que exhibe algunas de las propiedades de la tabla del patrón datos de entrada. [23]

Tabla 1

La capa de funcionamiento y de acceso a datos (ver tabla 1) se comunican a través de objetos de Propel. En este escenario se le presta especial atención a tener el 100% de desacoplamiento puesto que la capa de datos es bastante crítica y propensa a cambios. Esta situación es fácil de resolver porque Propel es un ORM que usa Creole para lograr la abstracción total de la base de datos que se use, cualquier cambio que se haga, como por ejemplo cambiar el gestor de base de datos, sólo afectaría un fichero de configuración (*databases.yml*) o si se cambia la estructura de la base de datos, también afectaría un solo archivo (*schema.yml*) Como se puede observar los componentes que se afectan en este nivel son mínimos y por tanto el sistema se hace altamente mantenible y flexible. Cualquier cambio y crecimiento en las implementaciones involucraría mayormente un fichero YAML de configuración y no el código fuente.

Escenario	Cuando un módulo necesita comunicarse con otro para solicitar algún tipo de información o utilizar alguna funcionalidad.
Objetivos del negocio	Llevar a cabo una operación que se salga del dominio del módulo
Atributo de calidad	Mantenibilidad, modificabilidad, flexibilidad, escalabilidad, integrabilidad.
Estímulo	Contabilizar
Origen del estímulo	Un gestor
Elemento	Un método
Ambiente	Se está llevando a cabo la contabilización de una operación contable
Respuesta	Se registra la contabilización en un comprobante contable
Medida de la respuesta	Nivel de abstracción, repercusión, capacidad de incorporación de funcionalidades
Preguntas	¿Cómo compartir los componentes de un módulo para llevar a cabo la operación? ¿Qué implica un cambio en alguno de estos componentes? ¿Cuáles componentes del módulo en que se está trabajando se ven afectados en el proceso ?

Tabla 2

La comunicación entre los módulos de la solución (ver tabla 2) se realizará a través servicios web que van a estar publicados en un servidor UDDI donde los servicios que requiera cada módulo los buscará en el UDDI así como publicará los servicios que prestará a los demás módulos. Esto posibilita una disgregación de las funcionalidades, y si en algún momento cambia la funcionalidad, solamente se debería cambiar en el módulo correspondiente sin que afecte el código de ningún módulo.

La asignación de los permisos a los módulos o algunas funciones dentro de los mismos se hará a través de un modulo de seguridad que estará independiente del ERP, y será el encargado de asignar las credenciales para poder realizar aquellas funcionalidades que requieran un nivel de acceso en correspondencia con un rol en particular. Esto permitirá tener la seguridad del sistema 100 % separada de la aplicación, pudiéndose afectar los privilegios de acceso para cualquier componente del sistema y

solo se afectaría el sistema de seguridad, lo que permitiría una matenibilidad de la aplicación en todo momento, sin tener que tocar una sola línea de código de ningún otro módulo (ver tabla 3).

Escenario	Cuando un módulo requiere un de nivel de acceso seguro.
Objetivos del negocio	Llevar a cabo una operación que necesite privilegios
Atributo de calidad	Mantenibilidad, modificabilidad, seguridad
Estímulo	Obtener privilegios de accesibilidad
Origen del estímulo	Un gestor
Elemento	Un método
Ambiente	Se está llevando a cabo una operación restringida para usuarios normales.
Respuesta	Se autentica al usuario permitiéndole interactuar con el sistema
Medida de la respuesta	Nivel de acceso, permisos de usuarios y roles
Preguntas	¿Cómo permitir que determinados usuario tengan solo acceso a determinado módulo? ¿Cómo agregar privilegios a los usuarios para que tengan un nivel de acceso diferente a otros dentro de un mismo módulo? ¿Cuáles componentes del módulo en que se está trabajando se ven afectados en el proceso?

Tabla 3

Los resultados reflejan la escalabilidad en gran medida en el sistema ERP, el cual facilita el crecimiento y adaptación a nuevos requerimientos del negocio puesto que las bases están creadas dentro de la infraestructura propuesta.

De manera general es importante destacar que el hardware según se vio en la vista de despliegue (ver Anexo 4) contribuye a la escalabilidad y disponibilidad de la arquitectura si tenemos en cuenta que puede ocurrir fallas, se asegura el respaldo de la información mediante mecanismos de réplica de datos y así mismo liberar la tensión y la sobrecarga de la red y el servidor.

Por la importancia que reviste este sistema para la toma de decisiones a nivel de país es fundamental la disponibilidad. Se considera que el sistema esta funcionando correctamente si los servidores de base de datos están trabajando así como los servidores web. El tiempo de respuesta de recuperación

de uno de estos servidores va a estar en el tiempo en que los servidores de respaldo entren en funcionamiento hasta que sea solucionado el problema.

Como se puede observar en el árbol de utilidad un estímulo clave puede se la falla de un nodo en el sistema, provocado por una falla de hardware o de software. Esta falla implicaría que dejara de trabajar el servidor web o el servidor de base de datos (BD), cualquiera de estos dos nodos afectarían el funcionamiento del sistema, en el caso del servidor de BD, que en los ERP es centralizado significaría la caída total del sistema, el servicio se restablecería cuando los servidores de respaldo o réplica entraran en acción, esto puede demorar o bien puede que no ocurra como se desea por lo tanto es importante que alguna persona se percibiera la falla y actuara de inmediato. Por esta razón una posible solución para la arquitectura seria el monitoreo veinticuatro horas al día de aquellos nodos que representan decisiones críticas en la arquitectura.

Además en los escenarios evaluados se pudo dejar claramente plasmado que la seguridad en el sistema va a estar centralizada, lo que garantizaría el cumplimiento exitoso de este atributo de calidad tan sensible dentro de cualquier arquitectura de sistemas. Lográndose a través de niveles de acceso restringidos mediante comunicaciones protegidas.

A partir de todos los elementos que se han reflejado, se puede afirmar que la solución propuesta es totalmente flexible, modificable, reparable y mantenible. Igualmente los factores que aseguran estas cualidades influyen directa o indirectamente en la integrabilidad, interoperabilidad y la escalabilidad, aunque en este caso, el hardware también juega su papel.

3.5 Conclusiones

Producto que la arquitectura del sistema no está detallada, en este capitulo se realizó una pre evaluación de la arquitectura propuesta descrita en el capítulo dos, mostrando como se puede analizar los atributos de calidad que se tienen como objetivo atendiendo a cada uno de los elementos de la arquitectura propuesta usando ATAM y basada en los distintos escenarios.

CONCLUSIONES

Con el desarrollo del presente trabajo basado en la propuesta de un diseño arquitectónico adecuado para el sistema ERP cubano se obtuvieron resultados concretos que demostraron el cumplimiento de los objetivos propuestos al inicio de la investigación.

La arquitectura de software juega un papel fundamental en el proceso de desarrollo de un software porque rige el proceso de construcción. Por tal motivo es importante prestar especial atención al estado de la documentación y las tecnologías existentes actualmente en el mundo referentes a esta materia. A partir de la investigación resultaron conceptos y tecnologías importantes:

- Los relacionados con la disciplina: el concepto de arquitectura de software.
- Los patrones y estilos de arquitectura.
- La calidad arquitectónica y atributos de calidad.
- Framework de desarrollo existentes (Symfony,Propel).

Así mismo, definir la arquitectura del sistema ERP cubano posibilitó llegar a un entendimiento entre todos los involucrados en el proceso de desarrollo, así como establecer un marco idóneo para representar todo los componentes y elementos que la conforman. En este sentido:

- Se describió la arquitectura desde el enfoque vertical.
- Se describieron los conceptos, terminologías y el entorno donde se va a desenvolver la arquitectura del sistema.
- Se describieron las políticas de integración para lograr la comunicación inter modular.
- Se describieron la vista lógica y la vista de despliegue, así como la vista del sistema de seguridad.
- Se seleccionaron las herramientas verticales y horizontales para el desarrollo del sistema.

Como toda propuesta arquitectónica, necesita ser evaluada, para obtener la validación de la misma y analizar si se cumplieron los atributos de calidad que se tuvieron como objetivo, para lograrlo y como parte de este proceso:

- Se definieron cuales son los atributos de calidad que se desean priorizar en la arquitectura del sistema (desempeño, modificabilidad, seguridad,disponibilidad).

- Se seleccionó como método para validar la arquitectura propuesta ATAM.
- Se definieron los escenarios principales para la arquitectura del sistema los cuales abarcan la comunicación entre capas, la comunicación entre módulos y la seguridad del sistema.

RECOMENDACIONES

Con el propósito de ampliar y mejorar la documentación de la arquitectura presentada en este trabajo, se plantean las siguientes recomendaciones:

- Se recomienda manejar los temas relacionados con el estudio sistemático de frameworks de PHP por la evolución constante que presentan los mismos y que pudieran servir en determinados puntos en la arquitectura.
- Se recomienda continuar profundizando en la validación de la arquitectura propuesta. En este sentido sería bueno aplicar la totalidad del método ATAM puesto que este brindaría un resultado completo y detallado de aquellos riesgos que afectan la arquitectura.

BIBLIOGRAFÍA

[1] Jacobson, I., G. Booch, et al. (2000). El Proceso Unificado de Desarrollo de Software, PEARSON EDUCACION.

[2] Clements, P. (1996). A survey of Architecture Description Languages.

[3] IEEE. (2000). "IEEE Std 1471-2000 IEEE Recommended Practice for Architectural Description of Software-Intensive Systems -Description." from http://standards.ieee.org/reading/ieee/std_public/description/se/1471-2000_desc.html

[4] Shaw, M. (1996). Introduction to Software Architectures New perspectives on an emerging discipline.

[5] Buschmann, F., R. Meunier, et al. (1996). Pattern – Oriented Software Architecture. A System of Patterns.

[6] Alexander, C., S. Ishikawa, et al. (1977). A Pattern Language: Towns, Buildings, Construction.

[7] Cez, G. d. S. "Sistemas ERP." from http://www.cez.com.pe/Sistemas/ERP.html.

[8] Maturana, S. (1999). "¿Cuánto ayudan los sistemas ERP en la planificación y programación en las actividades de una cadena de abastecimiento?"

[9] (2007). "Un poco de Historia." from http://php.uci.cu/?q=node/9

[10] (2007). "Framework." from http://wiki.prod.uci.cu/index.php/Framework.

[11] Potencier, F. and F. Zaninotto (2008). Symfony la guía definitiva.

[12] (2007). "CakePHP" from http://wiki.prod.uci.cu/index.php/CakePHP.

[13] Cortez, J. and S. Beltrán Libro de Kumbia: Porque Programar debería ser más fácil.

[14] (2007). "Fusebox." from http://wiki.prod.uci.cu/index.php/ Fusebox.

[15] (2007). "10 mejor IDE para PHP." from http://www.tufuncion.com/ide-php.

[16] "¿Qué es un servidor web (Web Servers)? - Definición de servidor web." from http://www.masadelante.com/faq-servidor.htm.

[17] MIGUEL, Adoración de y PIATTINI, Mario G. (1997) Fundamentos y modelos de bases de datos. Madrid: RA-MA

[18] (2007). "Ubuntu 7.10." from http://sushiknights.org/2007/10/ubuntu_7_10_gutsy_gibbon_beta.html.

[19] http://www.yafaonline.com/portal/desarrollo-proyectos/gestion-proyectos-online/dotproject.html

[20] Kazman, R., M. Klein, et al. (2000). ATAM: Method for Architecture Evaluation.

[21] Ramírez, P. IMPLANTACIÓN DE UN SISTEMA ERP EN UNA ORGANIZACIÓN. Rol y contribución de los sistemas ERP.

[22] “La historia de Openbravo.” from http://www.openbravo.com/es/about-us/our-company/history/.

[23] Lellelid, H. (2004). "Propel." from http://propel.phpdb.org/docs/es/user_guide/.

[24] Moreno, J. R. H. (2008). "¿Por qué escoger Symfony?" UXI REVISTA DE SOFTWARE LIBRE DE LA UCI **2**(3).

Erich Gamma, R. H., Ralph Johnson, and John Vlissides (1994). Design Patterns: Elements of Reusable Object-Oriented Software

Clements, P. (2004). Documenting Software Architecture Views and Beyond.

Reynoso, C. B. (2004). "Introducción a la Arquitectura de Software."

"METODOLOGÍA PARA SELECCIÓN DE SISTEMAS ERP."

Alexys Díaz, J. C. G., María Elena Ruiz (2005). IMPLANTACIÓN DE UN SISTEMA ERP EN UNA ORGANIZACIÓN.

Bass, L., P. Clements, et al. (1998). Software Architecture in practice.

Chiesa, F. "METODOLOGÍA PARA SELECCIÓN DE SISTEMAS ERP."

ANEXOS

ANEXO 1

Estructura del framework Symfony

Tabla 1 Directorios en la raíz de los proyectos Symfony

Directorio	Descripción
apps/	Contiene un directorio por cada aplicación del proyecto (normalmente, frontend y backend para la parte pública y la parte de gestión respectivamente)
Batch/	Contiene los scripts de PHP que se ejecutan mediante la línea de comandos o mediante la programación de tareas para realizar procesos en lotes (*batch processes*)
cache/	Contiene la versión cacheada de la configuración y (si está activada) la versión cacheada de las acciones y plantillas del proyecto. El mecanismo de cache utiliza los archivos de este directorio para acelerar la respuesta a las peticiones web. Cada aplicación contiene un subdirectorio que guarda todos los archivos PHP y HTML preprocesados
Config/	Almacena la configuración general del proyecto
data/	En este directorio se almacenan los archivos relacionados con los datos, como por ejemplo el esquema de una base de datos, el archivo que contiene las instrucciones SQL para crear las tablas e incluso un archivo de bases de datos de SQLite
doc/	Contiene la documentación del proyecto, formada por tus propios documentos y por la documentación generada por PHPdoc
lib/	Almacena las clases y librerías externas. Se suele guardar todo el código común a todas las aplicaciones del proyecto. El subdirectorio model/ guarda el modelo de objetos del proyecto.
log/	Guarda todos los archivos de log generados por Symfony. También se puede utilizar para guardar los logs del servidor web, de la base de datos o de

	cualquier otro componente del proyecto. Symfony crea un archivo de log por cada aplicación y por cada entorno.
plugins/	Almacena los plugins instalados en la aplicación
test/	Contiene las pruebas unitarias y funcionales escritas en PHP y compatibles con el framework de pruebas de Symfony. Cuando se crea un proyecto, Symfony crea algunos pruebas básicas
web/	La raíz del servidor web. Los únicos archivos accesibles desde Internet son los que se encuentran en este directorio

Tabla 2 Subdirectorios de cada aplicación Symfony

Directorio	Descripción
config/	Contiene un montón de archivos de configuración creados con YAML. Aquí se almacena la mayor parte de la configuración de la aplicación, salvo los parámetros propios del framework. También es posible redefinir en este directorio los parámetros por defecto si es necesario.
i18n/	Contiene todos los archivos utilizados para la internacionalización de la aplicación, sobre todo los archivos que traducen la interfaz (el Capítulo 13 detalla la internacionalización). La internacionalización también se puede realizar con una base de datos, en cuyo caso este directorio no se utilizaría
lib/	Contiene las clases y librerías utilizadas exclusivamente por la aplicación
modules/	Almacena los módulos que definen las características de la aplicación
templates/	Contiene las plantillas globales de la aplicación, es decir, las que utilizan todos los módulos. Por defecto contiene un archivo llamado layout.php, que es el layout principal con el que se muestran las plantillas de los módulos

Tabla 3 Subdirectorios de cada módulo

Directorio	Descripción
actions/	Normalmente contiene un único archivo llamado actions.class.php y que corresponde a la clase que almacena todas las acciones del módulo. También es posible crear un archivo diferente para cada acción del módulo
config/	Puede contener archivos de configuración adicionales con parámetros exclusivos del módulo
lib/	Almacena las clases y librerías utilizadas exclusivamente por el módulo
templates/	Contiene las plantillas correspondientes a las acciones del módulo. Cuando se crea un nuevo módulo, automáticamente se crea la plantilla llamada indexSuccess.php
validate/	Contiene archivos de configuración relacionados con la validación de formularios.

Tabla 4 Subdirectorios habituales en la carpeta web

Directorio	Descripción
css/	Contiene los archivos de hojas de estilo creados con CSS (archivos con extensión .css)
images/	Contiene las imágenes del sitio con formato .jpg, .png o .gif
js/	Contiene los archivos de JavaScript con extensión .js
uploads/	Se almacenan los archivos subidos por los usuarios. Aunque normalmente este directorio contiene imágenes, no se debe confundir con el directorio que almacena las imágenes del sitio (images/). Esta distinción permite sincronizar los servidores de desarrollo y de producción sin afectar a las imágenes subidas por los usuarios

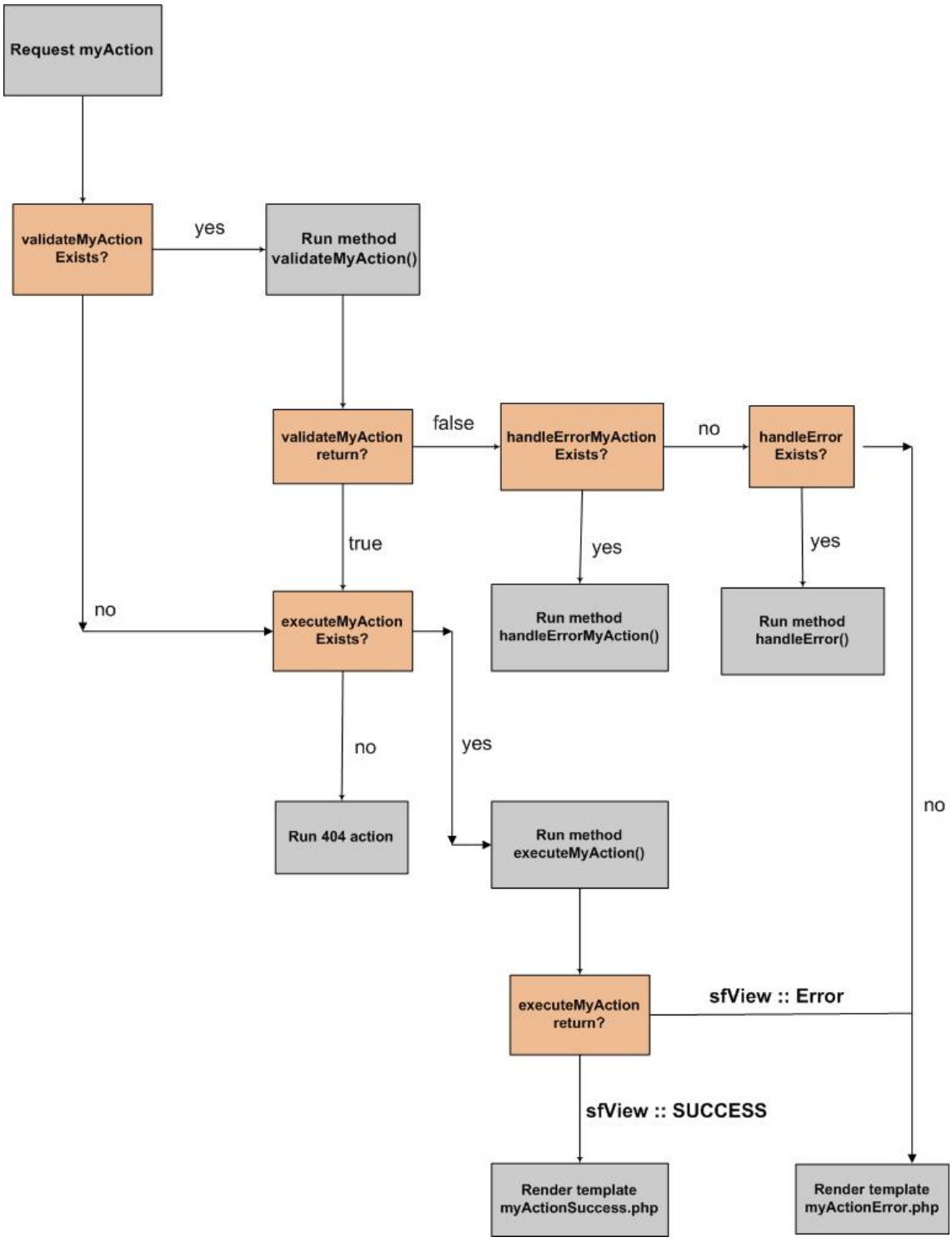
ANEXO 2

Tabla de herramientas

Nombre	Desarrollado por	Versión	Plataforma	Licencia	Modelo	Categoría
Ubuntu	Canonical Ltd. / Ubuntu Foundation	7.10	i386, AMD64, IA-64, UltraSPARC , PowerPC	GPL y GFDL	Software libre/open source	Sistema operativo
dotProject	Adam Donnison, Karen Chisholm, Gregor Erhardt, Ivan Peevski, Eamon Brosnan, Benjamin Young	2.1.1	Multiplataforma	GPL v2.0	Software libre/open source	Gestión de proyectos
PHP	PHP Group	5.2.6	Multiplataforma	PHP License	Software libre/open source	Lenguaje de Programación
Symfony	Sensio Labs	1.0.13	Multiplataforma	MIT License	Software libre/open source	Framework de aplicación web
PostgreSQL	PostgreSQL Global Development Group	8.2	Multiplataforma	BSD License	Software libre/open source	ORDBMS
Eclipse PDT	Eclipse	1.0 Released	Multiplataforma	Eclipse Public License	Software libre/open source	IDE
Subversion	Tigris	1.4.x	Multiplataforma	Apache/BSD	Software libre/open source	Control de versiones

ANEXO 3

Proceso de validación y manejo de errores



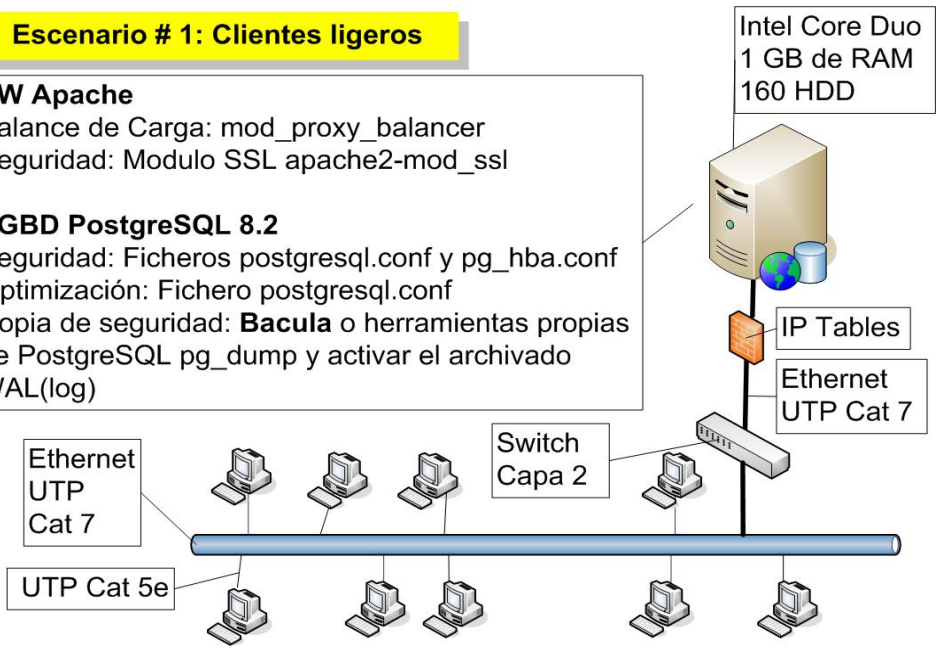
ANEXO 4

Vista de despliegue del ERP cubano

Escenario # 1: Clientes ligeros

SW Apache
Balance de Carga: mod_proxy_balancer
Seguridad: Modulo SSL apache2-mod_ssl

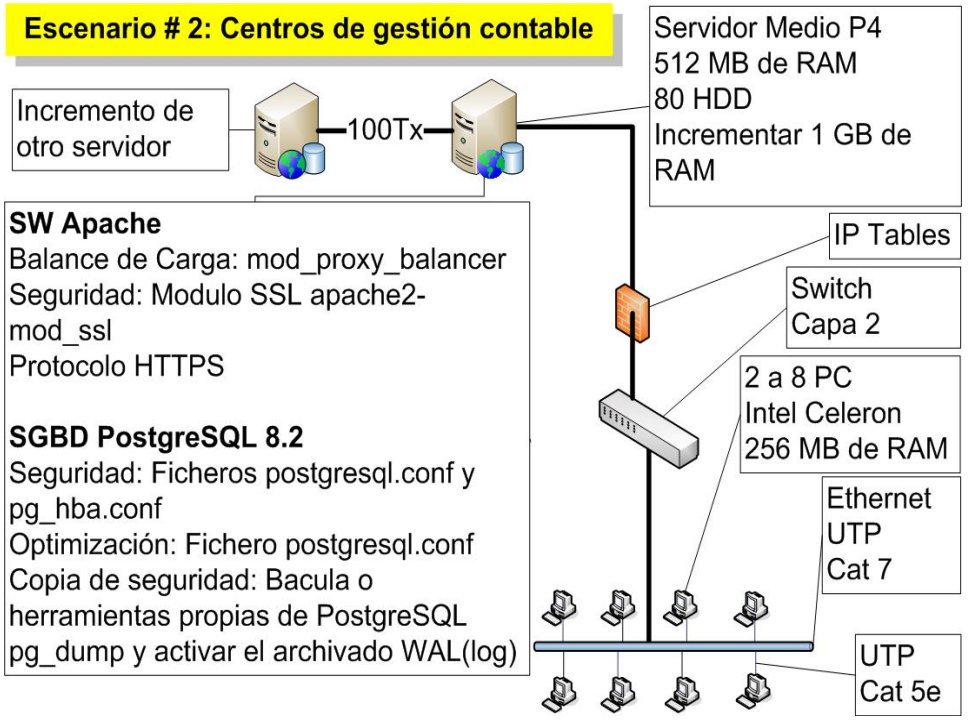
SGBD PostgreSQL 8.2
Seguridad: Ficheros postgresql.conf y pg_hba.conf
Optimización: Fichero postgresql.conf
Copia de seguridad: **Bacula** o herramientas propias de PostgreSQL pg_dump y activar el archivado WAL(log)



Escenario # 2: Centros de gestión contable

SW Apache
Balance de Carga: mod_proxy_balancer
Seguridad: Modulo SSL apache2-mod_ssl
Protocolo HTTPS

SGBD PostgreSQL 8.2
Seguridad: Ficheros postgresql.conf y pg_hba.conf
Optimización: Fichero postgresql.conf
Copia de seguridad: Bacula o herramientas propias de PostgreSQL pg_dump y activar el archivado WAL(log)

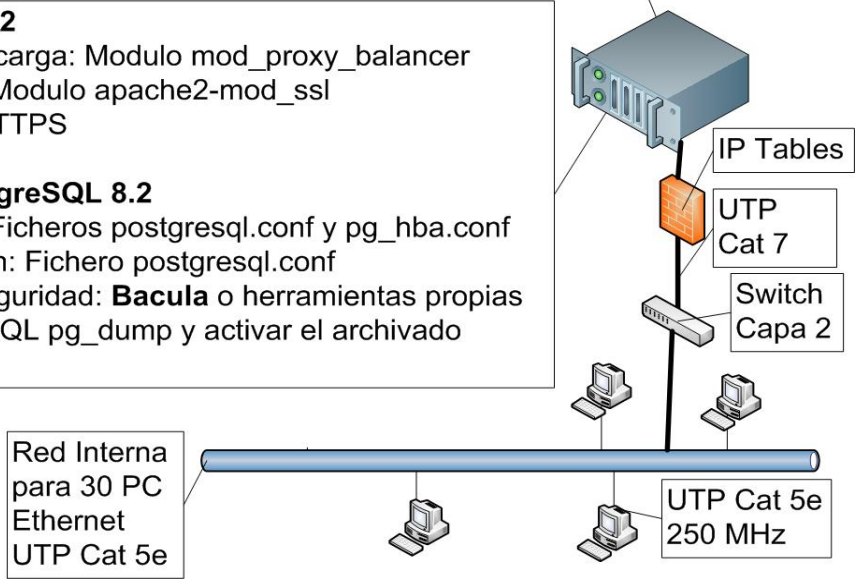


Escenario # 3: Empresas

1 Servidor Dedicado DELL
1 GB de RAM
Discos Espejos RAID 5

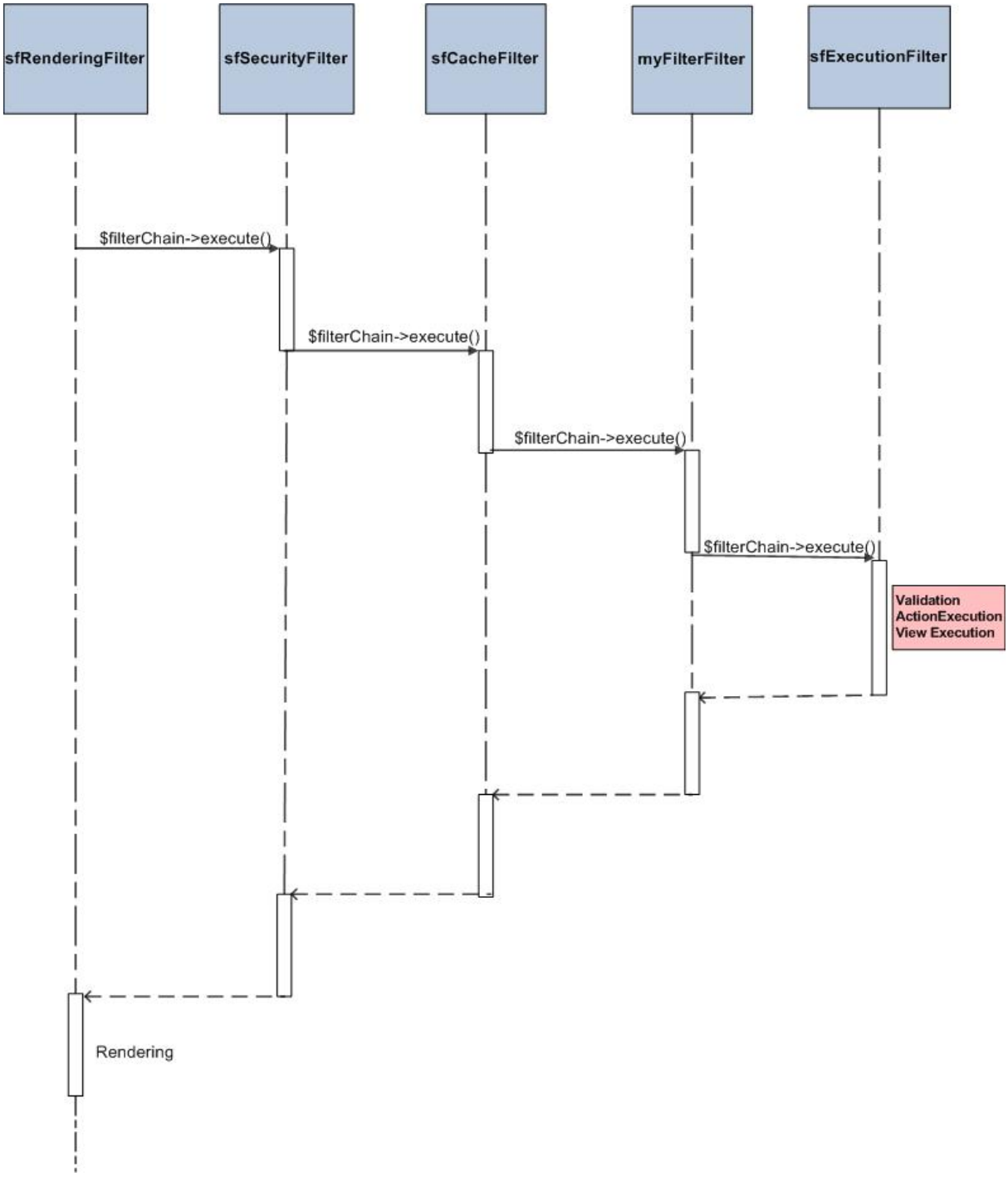
SW Apache2
Balance de carga: Modulo mod_proxy_balancer
Seguridad: Modulo apache2-mod_ssl
Protocolo HTTPS

SGBD PostgreSQL 8.2
Seguridad: Ficheros postgresql.conf y pg_hba.conf
Optimización: Fichero postgresql.conf
Copia de seguridad: **Bacula** o herramientas propias de PostgreSQL pg_dump y activar el archivado WAL(log)



ANEXO 5

Cadena de filtros



1 GLOSARIO