UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 7

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

Título: Propuesta de Capa de Presentación Asincrónica para el Sistema Integral de Salud para la Atención Primaria

Autores: Lisandra Pérez Albear

Alberto Varona Carmenate

Tutores: Ing. Johander León Garcés

Ing. Yosvanys Sánchez Corales

Ciudad de La Habana, Junio 2009

"Año del 50 Aniversario de Triunfo de la Revolución"

Declaración de Autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los 18 días del mes de junio del año 2009.

_____
Lisandra Pérez Albear
Firma del Autor

_____
Alberto Varona Carmenate
Firma del Autor

_____
Ing. Johander León Garcés
Firma del Tutor

_____
Ing. Yosvanys Sánchez Corales
Firma del Tutor

Datos de Contacto

Johander León Garcés (jleong@uci.cu): Graduado de Ingeniero en Ciencias Informáticas en el año 2007 con Título de Oro. Actualmente labora en la Universidad de Ciencias Informáticas (UCI), desempeñándose como profesor de la Facultad No. 7 vinculado a la producción y como miembro de la reserva del Comandante en Jefe. Ha participado en diferentes eventos científicos del centro obteniendo resultados relevantes. Actualmente es parte del proyecto Sistema de Información para la Salud (SISalud) donde se desarrolla como jefe de la línea productiva y además se encuentra al frente de un equipo desarrollo. Ha cursado varios cursos de postgrado como parte de su superación profesional, así como un diplomado de Líder de Proyecto.

Yosvanys Sánchez Corales (yscorales@uci.cu): Ingeniero en Ciencias Informáticas graduado en la CUJAE. Ha impartido las asignaturas de Programación III e Inteligencia Artificial. Actualmente pertenece al proyecto de Atención Primaria para la Salud. Ha participado en diferentes eventos científicos del centro obteniendo resultados relevantes. Además posee varias publicaciones a nivel nacional. Ha cursado varios cursos de postgrado como parte de su superación profesional.

Avance con confianza en la dirección de sus sueños. Esfuércese por vivir la vida que siempre ha imaginado y se encontrará con un éxito inesperado en una cuantas horas.

Henry David Thoreau

Agradecimientos

Nuestros agradecimientos van dirigidos a todas las personas que apoyaron la realización de este trabajo de diploma, especialmente nuestros tutores por su disposición, dedicación y ayuda.

A Leo por su incondicionalidad y por haberse convertido en un tutor más para nosotros.

A nuestra oponente Mairenis por su apoyo, recomendaciones e ideas, los cuales fueron imprescindibles para el buen acabado de nuestra investigación.

A la profesora Pura por haber aportado su importante granito de arena en la realización de nuestro trabajo.

A nuestros compañeros de trabajo, en especial Ariel y Yoelvis, gracias por siempre tener un sí como respuesta ante nuestro pedido de ayuda.

A nuestros familiares por la formación y apoyo, gracias a ellos nos hemos convertido en lo que somos hoy.

A nuestras amistades, por habernos dado la oportunidad de poder contar con ellas en estos 5 años de la carrera.

Dedicatoria

Dedico este trabajo a mi mamá y mi papá, que hoy y siempre serán lo más grande que tengo en mi vida.

A mi hermanita, que es mi Sol….la luz que me acompaña siempre.

A Marcos, por portarse como un hermano más para mí y por esforzarse a la par de mis familiares para que yo pudiese concluir mi carrera.

A la niña de mis ojos, el tesoro más lindo que tengo, cuyo nombre es Gretchel, por ser la alegría de mis días.

A mi tía Elena y mi prima Yady, por su apoyo y amor.

A la familia Obregón, por poder contar con ella en lo que siempre me ha hecho falta.

A mis antiguas amigas Mary, Miry y Day, por estar siempre en el lugar indicado, por estar pendientes de mi, por tener el record de soportarme por casi 20 años.

A las amigas que pusieron en mi camino cuando entré en la UCI: Vane por su incondicionalidad, su amor, su amistad desinteresada. A Eme, Lisy y Ani, Yane, Betty e Ise,por no fallar cuando más me hizo falta.

Lisandra Pérez Albear.

Dedico este trabajo a mis padres por haberme dado la vida, tanto amor y formarme como soy,

A mi hermano, el mejor regalo que mis padres me han podido dar.

A mi abuela y a mi tía, sin ellas me hubiera sido casi imposible llegar tan lejos.

A Yanisell, mi novia, quien ha sabido entenderme, esperarme y ayudarme en todo momento.

A Grisell y Elda, mi familia postiza, gracias por contribuir a que todo lo que me he propuesto me sea más fácil realizarlo con su apoyo. A Leo Soto, Leshter, Juan Pablo, Roger, Michel, Chulo, Maikel, Felix, Rubén, Yoelkis, Oscar, a todos los que durante estos 5 años me brindaron su ayuda incondicionalmente.

Alberto Varona Carmenate.

Resumen

En la actualidad en Cuba se trabaja para la informatización del Sistema Nacional de Salud, en este esfuerzo se unen un conjunto de aplicaciones cuyo propósito fundamental ha sido desarrollar el Sistema de Información para la Salud. El mismo presenta un modelo de diseño que no satisface a todas las necesidades de sus usuarios.

El presente trabajo tiene como objetivo desarrollar una interfaz de usuario asincrónica, que brinde mayor dinamismo y usabilidad al Sistema Integral de Salud para la Atención Primaria, haciendo uso de estándares abiertos y paradigmas computacionales.

Para ello se realizó un estudio de los antecedentes de ambas tecnologías y del conjunto de herramientas a utilizar para el desarrollo de la capa de presentación. Se demostró que el uso de AJAX y JSF aporta usabilidad y dinamismo a las interfaces del SIAPS.

Como resultados de la investigación se exponen la integración de los frameworks que permiten la realización de la arquitectura candidata para esta capa y se define la primera versión de la propuesta ajustada a las necesidades presentes del sistema actual. Se realiza un estudio de su factibilidad, la cual incide en el incremento continuo y sostenido de la calidad en la atención médica, lo que permitirá aumentar la calidad, oportunidad y consistencia de la información.

Palabras Claves: capa de presentación asincrónica, asincrónico(a), interfaz de usuario, Sistema Integral de Salud para la Atención Primaria, diseño, arquitectura de información.

TABLA DE CONTENIDO

TABLA DE CONTENIDO

TABLA DE FIGURAS

TABLA DE FIGURAS

Introducción

En Cuba, desde los inicios de la Revolución se han estudiado, planificado y diseñado estrategias que han permitido encaminar los esfuerzos para solucionar los seis problemas sociales que presentaba el país (el problema de la tierra, de la industrialización, de la vivienda, del desempleo, de la educación y de la salud del pueblo). Para solucionar este último se crearon organismos sanitarios de carácter nacional bajo la salvaguarda del Estado, que han proporcionado un modelo alternativo de organización sanitaria estatal.

A partir de este momento, se toman decisiones para centralizar los servicios médicos en una organización de carácter nacional. El organismo que concretó las funciones básicas que caracterizan los servicios sanitarios es el Ministerio de Salud Pública, creado el 22 de enero de 1960 (MINSAP). Este ha ido perfeccionando sus políticas para potenciar las áreas de la educación, los servicios, la investigación. Así como para elevar el nivel de salud de la población a partir de una alianza estratégica con los sectores educativos y de las comunicaciones.

Este perfeccionamiento persigue el objetivo de reorganizar el Sistema Nacional de Salud (SNS), específicamente se trabaja para aumentar la calidad de la Atención Primaria para la Salud (APS). Para contribuir con este proceso, desde el año 2003 se puso en marcha la estrategia para utilizar la informática en el mejoramiento de la calidad de atención. Se tuvo en cuenta la gran complejidad que caracteriza al procesamiento de la información manejada en las instituciones médicas. Se decidió desarrollar un sistema que permitiera integrar la información de diferentes áreas, utilizando una arquitectura basada en componentes y orientada a servicios. Es en este momento que comienza el desarrollo del Registro Informatizado de Salud (RIS).

Ante la tarea de crear un sistema informático para la gestión de la información en cada institución de salud pública. El Ministerio de Informática y las Comunicaciones (MIC) designa para su desarrollo a la empresa SOFTEL, dedicada a la ejecución de soluciones informáticas para la salud. Además de otros actores tales como: la Universidad de las Ciencias Informáticas (UCI) y un grupo de expertos funcionales del MINSAP que han trabajado en estrecha coordinación. Por estos motivos surge en su primera etapa de trabajo el Proyecto Atención Primaria para la Salud (APS) con la expectativa de desarrollar el Sistema de

Información para la Salud (SISalud). En este se unen un conjunto de aplicaciones cuyo propósito fundamental es la informatización del SNS. Cada módulo que lo integra desarrolla y pone a disposición de los otros sus procesos; lo que permite la interoperabilidad, el intercambio y acceso de información desde los diferentes niveles de dirección, ya sea nacional, provincial, municipal o de unidad de salud.

Esta solución se concibe en 3 partes lógicas caracterizadas por objetivos específicos: Capa de Presentación, Capa de Negocio y Capa de Datos. La primera tiene una gran importancia pues su objetivo es presentar el sistema al usuario mediante una interfaz, comunicar y capturar la información del mismo; solo se comunica con la de negocio. La segunda interpreta las acciones del usuario, accediendo a las operaciones de negocio de la aplicación y modificando a partir de sus resultados el estado del modelo y la navegación entre vistas. La última capa es la encargada de almacenar, recuperar y mantener la integridad de los datos.

El diseño está sustentado en el modelo tradicional de aplicaciones web, que no es todo lo dinámico que se requiere para que el sistema alcance su mayor usabilidad. Esta dificultad se pone de manifiesto al limitar visualmente al usuario; se crean interrupciones (“espacios en blanco”) que aparentemente no se observan por su poca duración. Lo que trae consigo la no obtención de la información solicitada con rapidez y que el usuario deseche la posibilidad de utilizar la aplicación.

Los procesos que dan respuesta a cada petición se ejecutan de forma simultánea por lo que puede congelar el navegador. Esta demora en el tiempo de respuesta produce que el usuario tenga que esperar a que se recargue la página con los cambios solicitados. Cuando son peticiones continuas, la aplicación se convierte en algo más incómodo que ventajoso.

El modelo de diseño web utilizado por el Sistema de Información para la Salud no es el único a nivel mundial. Existen otros modelos que presentan un sistema de comunicación con el servidor que permite que la variación en las páginas sea casi imperceptible. En las páginas, solo se actualizan las sesiones que muestran la respuesta de la solicitud hecha al servidor. Lo que favorece el dinamismo y la usabilidad del sistema, pues implica transparencia del proceso cliente- servidor.

La situación antes analizada, manifiesta deficiencias de usabilidad en el modelo de comunicación del SISalud con el usuario final. Las sobrecargas en la conexión y las ventanas del navegador que se

quedan en blanco cuando esperan respuesta, son a causa del sincronismo de su modelo. Por otra parte, la identificación de estas deficiencias es primordial para los especialistas de la salud. Quienes en ocasiones al no poseer las habilidades ni el conocimiento necesario sobre aplicaciones web, tienden a repetir varias veces la petición antes de que se cargue la página por primera vez. Todo ello contribuye a que el tiempo de respuesta demore más y a que no se muestre la información que el usuario espera.

La problemática planteada permite definir el siguiente **Problema a Resolver**: La ausencia de una capa de presentación acorde con las necesidades del usuario en el Sistema Integral de Salud para la Atención Primaria, dificulta la usabilidad y dinamismo en los subsistemas que están soportados sobre el mismo.

Para el desarrollo de la investigación se plantea como **Objeto de Estudio** la usabilidad y dinamismo de los subsistemas soportados en el Sistema Integral de Salud. El **Campo de Acción** se centra en la usabilidad y dinamismo del Sistema Integral de Salud para la Atención Primaria.

Se define como **Objetivo General** de la investigación: desarrollar una interfaz de usuario asincrónica, que brinde mayor dinamismo y usabilidad al Sistema Integral de Salud para la Atención Primaria, haciendo uso de estándares abiertos y paradigmas computacionales.

Las tareas de la Investigación que a continuación se presentan dan cumplimiento a dicho objetivo:

1. Analizar el estado del arte de las tendencias y tecnologías de la información, concernientes a las diferentes capas de presentación en los sistemas de gestión de la información para la salud a nivel nacional e internacional.
2. Realizar un análisis crítico de las tendencias, tecnologías y herramientas de desarrollo de software a utilizar.
3. Proponer una arquitectura de la información acorde con la vista de diseño web asincrónico.
4. Confeccionar el documento “Pautas de la Vista de Diseño” para los componentes del Sistema Integral de Salud para la Atención Primaria.
5. Definir los elementos de la capa de presentación y la integración de los componentes arquitectónicos y tecnologías a utilizar.
6. Implementar los componentes de diseño de la vista y su interacción con otros componentes arquitectónicos.

7. Profundizar en los aspectos principales de Factibilidad relacionada con la propuesta.
8. Implementar un caso de estudio, aplicando los conocimientos adquiridos de la propuesta definida.
9. Desarrollar una guía de referencia de la propuesta de capa de presentación.
10. Ejecutar las pruebas necesarias para garantizar el correcto funcionamiento de la vista asincrónica implementada.

En la presente tesis el contenido está estructurado por cinco capítulos que se muestran a continuación:

CAPÍTULO 1. Fundamentación Teórica: Se describen los aspectos y conceptos asociados al dominio del problema a resolver, siendo estos esenciales para entender el entorno del mismo. Se presenta el estado del arte de las tecnologías implicadas en el objeto de estudio y el conjunto de tecnologías involucradas en el desarrollo de la propuesta.

CAPÍTULO 2. Definición del Diseño de Capa de Presentación: Se definen los elementos básicos para la creación del diseño de la capa de presentación, se detalla la estructura que poseerán las interfaces para mostrar la información, así como las pautas a seguir en la realización de las mismas. Se realiza un análisis técnico de los framework utilizado para la integración de los mismos en la capa de presentación y se define la propuesta acorde a las necesidades existentes.

CAPÍTULO 3. Definición e Integración de la Arquitectura de Capa de Presentación: A partir de la propuesta de diseño se explica la integración de los componentes implicados en su creación y se muestra el despliegue de la solución. Se proponen los mecanismos para aportar seguridad e integración de la capa de presentación con las demás capas.

CAPÍTULO 4. Estudio de Factibilidad: Se muestran los aspectos fundamentales para evaluar las diferentes alternativas que la solución aporta y se explican las perspectivas de la propuesta para conseguir los resultados esperados.

CAPÍTULO 5. Estrategia de Validación de Resultados: Contiene el resultado práctico producido gracias a los conocimientos obtenidos. Se señalan algunos parámetros que permiten ponderar los factores que contribuyen a la visualización rápida, cómoda y efectividad de las páginas para el SIAPS.

Capítulo 1. Fundamentación Teórica

El presente trabajo de diploma se fundamenta en los conocimientos más actualizados para la obtención de productos de software con calidad y dinamismo adecuado. El objetivo es desarrollar en el proyecto: Sistema Integral de Salud para la Atención Primaria (SIAPS) una capa de presentación sencilla, amigable y que contribuya con la usabilidad del producto.

En este capítulo se describen diferentes aspectos a tener en cuenta en la construcción de interfaces de usuarios, así como las tendencias actuales de las diferentes tecnologías que se utilizan en el desarrollo de las mismas. Además se muestran los aspectos teóricos que definen una arquitectura y diseño candidatos en la construcción de la capa de presentación para el SIAPS; se realiza una fundamentación coherente de sus componentes. Todo lo anterior permitirá alcanzar un modelo más refinado que modifique al actual, haciendo uso de la tecnología java y del popular modelo de aplicaciones web asincrónico.

1.1 Marco Conceptual

Diseño: Proceso previo de configuración mental "pre-figuración" en la búsqueda de una solución en cualquier campo.

Diseño gráfico: Profesión cuya actividad industrial está dirigida a idear y proyectar mensajes visuales, contemplando diversas necesidades que varían según el caso: estilísticas, informativas, identificadoras, vocativas, de persuasión, de código, tecnológicas, de producción, de innovación.

Software: Equipamiento lógico o soporte lógico de un computador digital, comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema (hardware).

Frameworks: Estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado. Puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Arquitectura Software: 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.

Arquitecto de Información: Personas que organizan los patrones inherentes en los datos, que hacen claro lo complejo. Una persona que crea el mapa o la estructura de información que permite a otros encontrar su camino personal al conocimiento.

Arquitectura de Información: Disciplina que dispone y determina los contenidos de información y estructurales de un sitio web, a partir de las necesidades y preferencias de la audiencia, con el objetivo de garantizar la calidad final del producto y la plena satisfacción de los usuarios.

Usabilidad: Capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso.

Sincronismo: Convergencia de eventos relacionados.

Asincronismo: Falta de coincidencia en los hechos. Falta de simultaneidad en el tiempo.

JSP: Lenguaje que permite la incrustación de código en las páginas HTML que el servidor convierte en programas ejecutables.

Servlets: Un servlet es un programa escrito en Java que se ejecuta del lado del servidor. Los servlets corren dentro del contexto de un contenedor de servlets (ej: Apache Tomcat, IBM WebSphere, Sun Java System Application Server) y extienden su funcionalidad.

API: (Interfaz de Programación de Aplicaciones): Conjunto de especificaciones para comunicarse con una aplicación, normalmente para obtener información y utilizarla en otros servicios.

1.2 Arquitectura de Capa de Presentación Asincrónica

La capa de presentación es una parte de vital importancia para el éxito en las aplicaciones web. Esta es la que representa la interfaz entre el usuario y el resto de la aplicación para proporcionar interacción entre ambos. Si el usuario no puede interactuar con el sistema de forma que le permita realizar su trabajo con eficacia, el éxito global de este se ve muy perjudicado, perdiendo usabilidad y dinamismo. Estas básicas especificaciones implican un estudio de las tecnologías actuales, utilizadas en la creación de la nombrada capa para proporcionarle buena experiencia al usuario.

En la figura 1.1 se muestra lo que tradicionalmente sucede cuando el usuario interactúa con la interfaz de la aplicación. En esta interacción se crea un evento que es enviado hacia el servidor por medio del protocolo para procesarlo y dar respuesta al mismo. Este protocolo, que apareció a principios de los 90', tiene dos tipos de mensajes que se suceden sincrónicamente, es decir, no hay response sin un request previo y no puede haber más de un request consecutivo sin response ni viceversa.

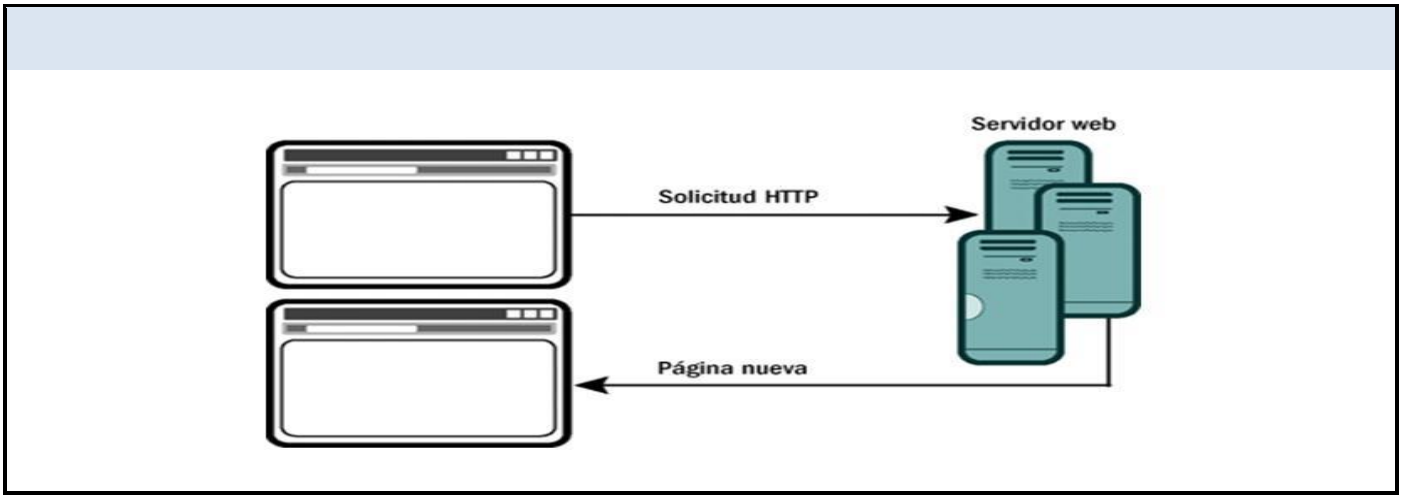

Fig1. 1 Operación tradicional de devolución de la página completa.

Estos mensajes se envían una vez establecida la conexión entre el servidor y el cliente. Este protocolo tiene una característica bastante particular: incide directamente en la arquitectura del sistema a construir y es un protocolo sin estado, o lo que es lo mismo, el servidor no recuerda si antes de un determinado mensaje se enviaron otros desde el mismo cliente ni recuerda los datos que se enviaron. Este proceso

limita la interacción de las aplicaciones pues produce la actualización completa de la página, otorgándole bajo rendimiento, poca escalabilidad y flexibilidad al sistema.

A medida que las aplicaciones web se fueron volviendo cada vez más importantes, el modelo de interfaz entre el servidor y los ejecutables empezó a ser insuficiente y se buscaron soluciones alternativas para enriquecer la arquitectura de la capa de presentación en las mismas.

Conceptualmente la arquitectura de capa de presentación es la lógica basada en patrones que proporciona una interfaz gráfica para aplicaciones web. Para proporcionarle escalabilidad y flexibilidad, se incluye en su funcionamiento los estándares abiertos que engloba el término AJAX, contribuyendo con la funcionalidad de esta capa, favoreciendo el asincronismo de su comunicación con los demás componentes con los que interactúa. A continuación se muestra una representación sencilla del uso de este término en la capa de presentación.

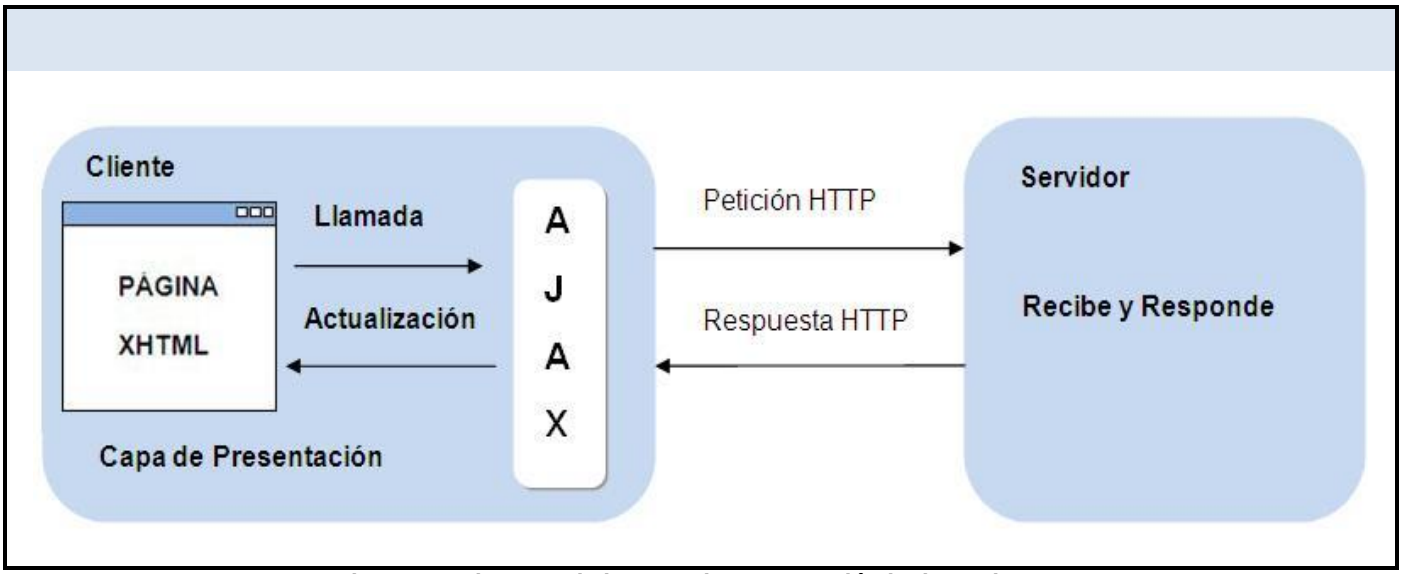

Fig1. 2 Arquitectura de la Capa de Presentación incluyendo AJAX.

Las aplicaciones que en su capa de presentación utilizan AJAX, proporcionan al sistema una mejor interacción entre el cliente y el servidor, pues este se encarga de realizar las llamadas al mismo y procesar el resultado que proviene del este. La finalidad de esta comunicación es que se actualice sólo la

sesión de la página asociada con el resultado devuelto, en vez de cargarse o dibujarse la página completamente como se realiza en muchas ocasiones, derivando un retardo mayor en el tiempo de respuesta.

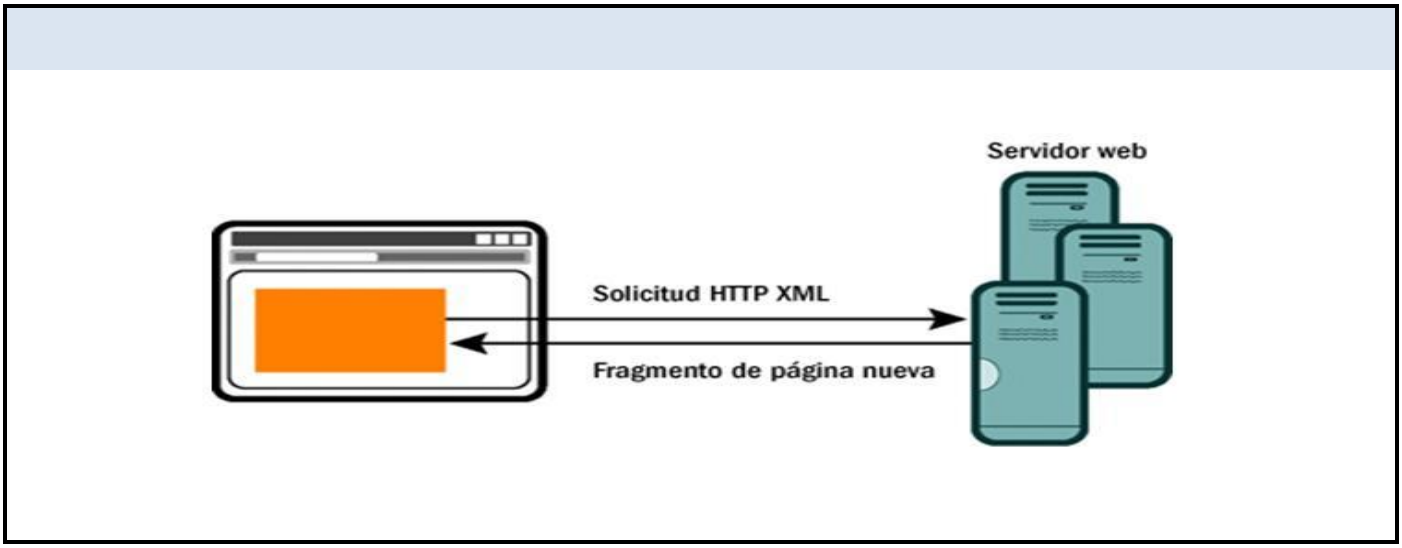

Fig1. 3 Representación parcial de AJAX XMLHttpRequest.

Como se observa en la figura 1.3, la principal diferencia que marca un hito entre la arquitectura sincrónica y la asincrónica es que en el modelo sincrónico por cada petición hecha por el usuario, se obtiene como respuesta la construcción de la página completamente sin necesidad. Como se observa en la figura, utilizando el asincronismo sólo se construye el área o segmento de la página involucrado en la obtención del resultado, manteniendo el contenido restante que no necesita ser actualizado. Ejemplos de dicha actualización en las áreas específicas son: las actualizaciones de las tablas, encabezados y menús.

El término asíncrono es muy utilizado debido a las ventajas que a continuación se presentan:

- **Tráfico mínimo**: Posibilita a las aplicaciones que usan AJAX que envíen y reciban una pequeña cantidad de información desde y para el servidor, por lo que se minimiza el tráfico entre el cliente y el servidor, asegurándose que no reciba o envíe información innecesaria.
- **Interactividad**: Contribuye a la recuperación asíncrona de datos, contribuyendo a que el usuario espere la respuesta de una petición en un mínimo período de tiempo.
- **Portabilidad**: Contribuye a que la aplicación sea soportada por la mayoría de los navegadores modernos.

- **La prioridad es el usuario**: Posibilita que el diseño de las aplicaciones se conciba pensando en las necesidades del usuario.
- **Tiempo de respuesta:** Contribuye a que la respuesta de la petición realizada por el usuario se haga de forma instantánea, ya que sólo se realizan actualizaciones granulares de la página (por áreas), reduciendo su tiempo de respuesta.

1.3 Estado del Arte de la Capa de Presentación en los Sistemas de Gestión Información para la Salud

Con el creciente perfeccionamiento de los sistemas informáticos de gestión de información, surgen con el transcurso de los años aplicaciones mejor diseñadas. Para obtener un diseño de éxito es necesario tener en cuenta las causas que influyen en el comportamiento humano. Una de ellas está estrechamente relacionada con el entorno del hombre y la interfaz de los objetos que maneja. Dentro de estos objetos se encuentran los ordenadores, por tal motivo se hace muy importante diseñar interfaces que faciliten la relación hombre - máquina.

Se diseña teniendo en cuenta tres pilares fundamentales para facilitar la comunicación. El primero es la forma, es en este caso donde se tiene en cuenta la comodidad de uso y el grado de atracción que poseerá la aplicación. El segundo es la adecuación al tipo de usuario que lo utilizará, valorando el hecho de que el diseño se enfocará hacia sus necesidades; para ello se debe tener en cuenta que gran parte de la población mundial es débil visual o poseen otro tipo de discapacidad. El último es la funcionalidad que le hará ser útil para el objetivo con el que se creó.

Si se logra la total conjugación de lo antes mencionado, el diseñador logrará crear una aplicación interactiva que se comporte exactamente como el usuario piensa que se haría. De esta manera se obtienen productos fáciles de usar, reduciendo el costo de desarrollo, soporte, tiempo de creación y se hace posible satisfacer las necesidades de los beneficiados.

En pocos años las aplicaciones web se han convertido en complejos sistemas que brindan servicios a procesos de negocio de considerable importancia, siendo uno de los campos de utilidad crítica el sector

de la salud. En los productos informáticos se establecen requisitos estrictos de accesibilidad y respuesta, por este motivo se exige reflexionar sobre la arquitectura y las técnicas de diseño más adecuadas en su construcción.

Existen dos necesidades básicas cuando el diseñador se enfrenta a la creación de un sistema informático; a continuación se presentan:

1. La información sea accesible desde cualquier lugar dentro de la organización o incluso desde el exterior.
2. Esta información sea compartida entre todas las partes interesadas de manera que todas tengan acceso a la información completa (o a aquella parte que le corresponda según su función) en cada momento. [1]

Estas necesidades suponen una interacción eficaz con el usuario, e inevitablemente arrastran consigo la necesidad de buscar nuevas opciones de diseño y una arquitectura candidata que facilite la construcción de los mismos.

Con el transcurso de los años las aplicaciones web han evolucionado paralelamente a la arquitectura que se utiliza en su construcción. En los inicios el modelo de las aplicaciones web no le otorgaba a las interfaces dinamismo, simplemente constituían una colección de páginas estáticas por las cuales los usuarios podían descargar y consultar información. Paulatinamente se incluyó un método para elaborarlas de forma tal que el contenido se mostrara de forma dinámica (generar datos a partir de una petición hecha por el usuario).

Este método fue conocido como CGI ("Common Gateway Interface-Interfaz de Entrada Común"), el mismo define un mecanismo mediante el que se puede pasar información entre el servidor y determinados programas externos. Estos CGIs se siguen usando debido a su sencillez y porque dan libertad para elegir el lenguaje de programación que se desee utilizar.

Sin embargo su funcionamiento tenía un punto débil pues por cada petición recibida, el servidor debía lanzar un proceso para ejecutar el programa CGI. Como la mayoría de CGIs estaban escritos en lenguajes interpretados, como Perl o Python, o en lenguajes que requerían "run-time environment", como Java o

Visual Basic, el servidor se veía sometido a una gran carga. La concurrencia de múltiples accesos al CGI podía desencadenar problemas graves.

Debido a esta deficiencia, se desarrollan alternativas a los CGIs que solucionaran el problema del rendimiento. Las vías que se tuvieron en cuenta fueron:

- Diseñar sistemas de ejecución de módulos mejor integrados con el servidor, que evitan la instanciación y ejecución de varios programas.
- Se dota a los servidores un intérprete de algún lenguaje de programación que permita incluir el código en las páginas de forma que lo ejecute el servidor, reduciendo el intervalo de respuesta.

A partir de este momento aumenta el número de arquitecturas y lenguajes que favorecen el dinamismo y la usabilidad en las aplicaciones web. Una de las más potentes es la seguida por Sun Microsystems con Java, integrado por 2 componentes; un lenguaje que permite la incrustación de código en las páginas HTML que el servidor convierte en programas ejecutables, JSP ("Java Server Pages" o "Páginas de Servidor de Java"), y un método de programación muy ligado al servidor, con un rendimiento superior a los CGIs, denominado "Java Servlet". Otra tecnología de éxito y una de las más utilizadas es el lenguaje PHP. Se trata de un lenguaje interpretado que permite la incrustación de HTML en los programas, con una sintaxis derivada de C y Perl. El hecho de ser sencillo y potente ha contribuido a hacer de PHP una herramienta muy apropiada para determinados desarrollos. [2]

Todo sistema informático debe de estar implementado bajo una arquitectura sólida y robusta que lo haga escalable, flexible ante cambios, que lo caracterice por poder ampliarse con facilidad en dependencia de las necesidades de los usuarios. En el diseño de tales sistemas, es muy popular el uso de la arquitectura multinivel o programación por capas. En la misma a cada nivel se le confía una misión, lo que permite el diseño de arquitecturas escalables. El diseño que más auge tiene es el de tres capas.

La lógica que supone la interacción de un usuario con el sistema es un elemento vital en la construcción de aplicaciones. En la medida que este suceso de eventos se realice con la mayor rapidez posible, mejor será la experiencia hombre-máquina, contribuyendo a la usabilidad de la aplicación.

El usuario interacciona con la aplicación web mediante un navegador. En consecuencia se envían peticiones al servidor, lugar donde se aloja la aplicación y normalmente hace uso de la base de datos que almacena la información relacionada con la misma. En el servidor se procesa la petición y devuelve la respuesta al navegador que la presenta al usuario. La distribución antes mencionada se basa en tres componentes: el navegador, que presenta la interfaz al usuario; la aplicación, encargada de realizar las operaciones necesarias según las acciones llevadas a cabo por este; la base de datos, lugar donde la información relacionada con la aplicación se hace persistente. Dicha distribución se conoce como modelo o arquitectura de tres capas e internacionalmente es la tendencia más utilizada.

A continuación se explica lo que se conoce como separación de incumbencias en los sistemas informáticos denominados multicapa o n-capas, específicamente el de 3 capas:

1. **Presentación:** Denominada capa de presentación o interfaz gráfica, debe de caracterizarse por ser entendible y fácil de usar. Es por la cual el sistema recibe las solicitudes y presenta los resultados. Limitada a la navegación y a gestionar los aspectos relacionados con la lógica de presentación de la aplicación, dentro de los que se encuentra la comprobación de datos de entrada, formatos de salida etc. Esta se comunica únicamente con la capa de negocio.
2. **Negocio o dominio:** Lugar donde residen los programas que se ejecutan, donde se establecen las reglas que deben cumplirse y donde se envían las peticiones del usuario. Esta capa es la intermedia entre la capa de presentación y la de datos.
3. **Acceso a datos:** Encargada de la persistencia de entidades que se manejan en el negocio, acceso a datos almacenados, actualización y recuperación de la información.

Para contribuir en la robustez de la arquitectura del software se han definido patrones arquitectónicos; estos describen componentes y sus relaciones. La mayoría de los patrones utilizados son esquemas lógicos aplicables a estas capas y se utilizan para plantear una solución de acuerdo a las particularidades de cada proceso de negocio.

Uno de los más aplicados en el diseño de aplicaciones web en sistemas empresariales distribuidos es el Modelo-Vista-Controlador (MVC). El mismo propone la separación en distintos componentes de la interfaz de usuario (vistas) para ahorrar las líneas de código con la finalidad de permitir la recurrencia de funciones. Por otro lado asegura que el cambio que se haga en algún componente de la aplicación no

afecte a la estructura general del sistema. Es el patrón de diseño arquitectural que se recomienda utilizar en la capa de presentación de aplicaciones interactivas Java.

En estos sistemas, el modelo suele ser más estable a lo largo del tiempo y menos sujeto a variaciones; sin embargo las vistas pueden cambiar con frecuencia, ya sea por necesidades de usabilidad de la interfaz o simple renovación de la estética de la aplicación. Con esta separación las vistas pueden cambiar sin afectar al modelo y viceversa.

Además de los patrones arquitectónicos existen los de diseño. Para cada capa se ha definido varios, específicamente los más utilizados para enriquecer la capa de presentación de las aplicaciones web están: Patrón Fachada, Decorating Filter / Intercepting Filter, Front Controller/ Front Component, View Helper, Composite view, Service To Worker, Dispatcher View. Con el transcurso de los años el modelo para diseñar y proveer una buena arquitectura al futuro software, es el que supone asincronismo de eventos. Esto ha sido posible gracias al uso del término AJAX.

Actualmente existen varios framework de desarrollo de aplicaciones web como Struts, Spring, WebWork o Maverick, entre otros. El de mayor difusión es Struts. Aunque este tuvo bastante éxito y es muy aceptado en las empresas por hacer desarrollos rápidos y ordenados tiene limitaciones. Este hecho ha dado lugar a que surjan otros frameworks y especificaciones mejor elaboradas. Este es el caso de Java Server Faces (JSF). Existen muchas implementaciones y librerías de componentes de JSF basadas en AJAX. Se han identificado 27, entre los más conocidos se encuentran Ajax4JSF, TomaHawk, RichFaces, IceFaces, , WebgalileoFaces, ZK, RCFaces, entre otros.

Con la evolución de las aplicaciones Web, quedó atrás la primera Web, caracterizada por fondos grises y poca o nula interacción con el usuario. La Web 2.0 significa la segunda generación histórica de la web basada en comunidades de usuarios y una gama especial de servicios, significa una nueva filosofía de hacer las mismas cosas con los estándares sobre los que se apoyan las aplicaciones y los servicios web (Web-Services).

La inauguración de la Web 2.0 fue hecha por Google el 2 de abril de 2004, con el lanzamiento de Gmail, su servicio de correo gratuito, con una capacidad insospechada para su tiempo. Desde entonces son miles los productos que los emprendedores tratan de capitalizar.

Algunos ejemplos de aplicaciones reconocidas internacionalmente que están basadas en la Web 2.0 son las que se muestran a continuación:

▪ **Gestores de correo electrónico:** Yahoo Mail [http://mail.yahoo.com], Windows Live Mail [http://www.hotmail.com].

▪ **Sistemas de cartografía:** Google Maps [http://maps.google.com], Yahoo Maps

[http://maps.yahoo.com], Windows Live Local [http://maps.live.com].

▪ **Aplicaciones web y meta páginas:** Netvibes [http://www.netvibes.com], Google Docs

[http://docs.google.com], Google Personalized Home [http://www.google.com/ig].

▪ **Otras:** Digg [noticias, http://www.digg.com], Meebo [mensajería, http://www.meebo.-com], 30 Boxes [calendario, http://www.30boxes.com], Flickr [fotografía, http://www.-flickr.com]. [3]

La tendencia que existe en Cuba dentro de este marco productivo es la de desarrollar aplicaciones web utilizando los lenguajes de programación comúnmente conocidos como PHP o Java, y en la capa de presentación XML, XHTML, XSL, entre otros. Sin embargo el estudio de las nuevas tecnologías y su aplicación no está proliferado en todas las empresas que se dedican a producir software, debido al desconocimiento de las mismas.

Sin embargo, la producción de software en Cuba se perfecciona con el transcurso del tiempo. Tales tecnologías se han podido adoptar gracias a las inversiones que el país ha hecho; lo que ha posibilitado la informatización de los sectores sociales. En consecuencia, esta experiencia que se obtiene gradualmente por los especialistas informáticos, ha favorecido a la producción de sistemas informáticos mejor elaborados y dirigidos a la satisfacción del usuario final.

La informática desde hace varias décadas ha encontrado en la medicina una de sus aplicaciones más comunes e importantes que permite al sector de la salud, no sólo contar con métodos novedosos, sencillos y eficaces de gestión administrativa en hospitales, instituciones de la APS y centros de investigación biomédica, sino también disponer de recursos informáticos de gran valor, en las

exploraciones con tratamiento de imágenes, señales bioeléctricas y otras que reducen la posibilidad de error en el diagnóstico de las enfermedades, aumentando la confiabilidad de la misma .También ofrece una gran ayuda en el campo de la investigación epidemiológica, farmacéutica, biológica, química, etcétera, todos estos aspectos se encuentran relacionados para conseguir un mejor nivel de salud en las personas y la excelencia en los servicios.

Durante los últimos 20 años un grupo de instituciones cubanas han desarrollado sistemas encaminados a lograr determinados niveles de informatización de la salud. Estas soluciones carecían del diseño adecuado para proporcionar el mayor grado de usabilidad, dinamismo, estética, flexibilidad y escalabilidad a la capa de presentación de la aplicación, así como de una definición generalizable y adaptable a cualquier entorno. Además de la no existencia de componentes asincrónicos y por ende de las actualizaciones granulares.

A partir de 1997, se concibe, una primera estrategia de informatización, como respuesta del sector de la salud a los lineamientos estratégicos para la informatización de la sociedad cubana, con la finalidad de coordinar esfuerzos para el desarrollo de este proceso en el SNS.

Para contribuir con la informatización y el desarrollo económico cubano, surge la Universidad de las Ciencias Informáticas (UCI), centro de estudios nacido como un proyecto de la Revolución Cubana, denominado al principio "Proyecto Futuro". Es la primera universidad cubana creada bajo los propósitos de la Batalla de Ideas. Cuenta con tecnología informática avanzada y constituye uno de los principales centros telemáticos de Cuba y Latinoamérica. Posee 10 facultades que tienen la misión de crear entornos de desarrollo según la temática asignada o especialidad, posibilitada por el arduo trabajo que llevan a cabo estudiantes y profesores en cada proyecto productivo.

Aunque los conocimientos en las tecnologías de nueva generación a nivel internacional en productos informáticos y arquitecturas de capa de presentación son escasos, existen proyectos que se encuentran en proceso de migración hacia ellas. Específicamente en la Facultad 7, el proyecto Atención Primaria para la Salud, tiene la misión de desarrollar software de alta calidad para la medicina cubana.

1.4 Problema a Resolver y Situación Problémica

El SISalud fue concebido con los lenguajes PHP, XML, XSL, JavaScript y fue implementado sobre el framework PlaSer, derivado de Plataforma de Servicios, que encapsula, agiliza y estandariza el trabajo con XML-Web Services.

El diseño de la capa de presentación de la aplicación está sustentado en el modelo tradicional de aplicaciones web, el que posee conexiones síncronas. Esta característica produce interrupciones en la interacción del usuario y el sistema cada vez que se hace una petición al servidor. Esto condiciona que su usabilidad se afecte considerablemente, por tal razón es necesario mejorar la versión actual, y enriquecer la capa de presentación con la arquitectura e interfaz propicia.

Una vez que el usuario (en este caso un técnico de la salud) interacciona sobre la interfaz de la aplicación mediante formularios, realiza peticiones HTTP que se envían al servidor. Luego de ser captada la petición por el servidor, el mismo ejecuta un conjunto de actividades, recopilando la información, realizando cálculos numéricos y activando procedimientos para dar la respuesta solicitada. Cuando este proceso llega a su fin, se devuelve una página XHTML con los resultados pedidos. Esto conduce a un desperdicio enorme de ancho de banda pues gran parte del HTML enviado en la página web producida como respuesta, ya estaba presente en la inicial.

Este proceso necesita un tiempo de respuesta para poder ejecutarse y, cada vez que se hace una petición se hace mayor; condicionando sobrecargas en la conexión, que la ventana del navegador se quede en blanco, pues en vez de actualizarse solo el campo por donde debe de mostrarse el resultado, se carga la página completa.

Fig1. 4 Esquema Clásico del Diseño Web.

Técnicamente este modelo no es errado, sin embargo mientras el proceso se lleva a cabo, ¿qué está haciendo el técnico de la salud? La respuesta es simple: esperar. En la medida que la tarea lleve una secuencia de pasos, la demora se incrementa, logrando impaciencia e incomodidad a la persona que interacciona con el sistema.

La problemática planteada demuestra que el SISalud en la actualidad no posee un diseño de interfaces web acorde con las necesidades del usuario; trayendo como consecuencia dificultad en la usabilidad y dinamismo en los subsistemas que están soportados sobre el mismo.

1.5 Personalización de Estilos para la Capa de Presentación

La idea fundamental de la interfaz es el de mediación, entre hombre y máquina. La interfaz es lo que "media", lo que facilita la comunicación, la interacción, entre dos sistemas de diferente naturaleza, típicamente el ser humano y una máquina como el computador. Esto implica, además, que se trata de un sistema de traducción, ya que los dos "hablan" lenguajes diferentes: verbo-icónico en el caso del hombre y binario en el caso del procesador electrónico.

De una manera más técnica se define a Interfaz de usuario, como conjunto de componentes empleados por los usuarios para comunicarse con las computadoras. El usuario dirige el funcionamiento de la máquina mediante instrucciones, denominadas genéricamente entradas. Las entradas se introducen mediante diversos dispositivos y se convierten en señales electrónicas que pueden ser procesadas por la computadora.

Al diseñar interfaces de usuario deben tenerse en cuenta las habilidades cognitivas y de percepción de las personas. Adaptar el programa a dichos individuos muchas veces puede ser un elemento importante para transmitir comodidad, bienestar, satisfacción, así como para brindar un entorno agradable y emotivo al mismo.

Los sistemas hoy en día están diseñados con hojas de estilo - CSS (Cascading Style Sheets), lo que permite cambiar fácilmente la apariencia de los mismos, diversas aplicaciones web, utilizan esta técnica para cambiar la apariencia de su interfaz y así mantener motivado al usuario. La mayoría de los sistemas de gestión importantes han puesto en esta técnica más que una atracción elegante, un punto estratégico para atraer al usuario final.

Dichos software comúnmente realizan el cambio de apariencia en la página principal del sitio, permitiendo a través de esta opción modificar los estilos predefinidos en la aplicación o a los definidos por los usuarios. Allí podrá definir los tipos de letra, fondos de menús, colores del sitio, aumentar o disminuir formas u objetos, etc.

1.6 Tendencias y Tecnologías

Habitualmente se escuchan, se usan los términos tendencia y tecnología y no se sabe su real significado. Las tecnologías son habilidades, instrumentos, recursos técnicos o procedimientos empleados en la creación del producto software, mientras que las tendencias son los elementos informáticos que dirigen la evolución de tales productos. A continuación se exponen los términos que ayudarán a comprender la posible solución técnica del problema anteriormente planteado.

1.6.1 Estilos Arquitectónicos

Patrón Modelo Vista Controlador

Este patrón se basa en realizar un diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos.

Aunque la aproximación de ligar la lógica de negocio con la interfaz puede ser válida para implementar sistemas, presenta varios inconvenientes: no permite reutilizar código en cada uno de los manejadores y se complica cuando se realizan operaciones que implican llamar a varios servicios de la lógica de negocio, sobre el navegador recae la responsabilidad de controlar la navegación, por lo que se crea dependencia entre los formularios, lo que hace que sea muy difícil modificar el flujo de navegación, añadir nuevos formularios o cambiar el comportamiento de alguno y, obliga a los formularios a acceder de forma separada a la lógica de negocio, por lo que se tendrá una copia distinta de los datos.

Para solucionar estos inconvenientes se puede utilizar un patrón de diseño llamado Modelo Vista Controlador (MVC). Este propone dividir la estructura de la aplicación cliente en tres clases distintas:

Modelo: gestiona el comportamiento y los datos de la aplicación, responde a las peticiones que realizan las vistas sobre su estado y permite su actualización normalmente desde el controlador.

Controlador: interpreta las acciones del usuario, accediendo a las operaciones de negocio de la aplicación y modificando a partir de sus resultados el estado del modelo y la navegación entre vistas.

Vista: muestra el estado al usuario de la aplicación, redirigiendo las acciones que realiza sobre la interfaz al controlador. Puede ser desarrollada por un equipo de diseñadores independiente de los programadores.

Vista Compuesta

Este patrón propone utilizar vistas compuestas que se componen de varias subvistas atómicas. Cada componente de la plantilla se podrá incluir dinámicamente dentro del total y la distribución de la página se maneja independientemente del contenido.

Contexto: Las aplicaciones web sofisticadas presentan contenido de numerosas fuentes de datos, usando múltiples subvistas, que conforman una única página a visualizar.

Solución: Usar vistas compuestas que están formadas por múltiples subvistas atómicas, cada componente de la plantilla puede ser incluido dinámicamente en el total, y el esquema de la página puede ser gestionado de forma independiente al contenido.

Un requisito típico de las aplicaciones web es el que su visualización tenga una estructura similar a lo largo de todo el sistema. Una plantilla es un componente de presentación que compone vistas separadas en una única página con un diseño específico.

Consecuencias: Mejora de la modularidad y la reutilización, mejora la flexibilidad, el mantenimiento, y la manejabilidad, tiene un impacto negativo (aunque muy pequeño) en el rendimiento.

ViewHandler o Patrón Decorador

El patrón de diseño ViewHandler ayuda a manejar todas las vistas que provee un sistema de software. Un componente VH permite a los clientes abrir, manipular y disponer de vistas. También coordina las dependencias entre vistas y organiza su actualización.

Todas las implementaciones de Java Server Faces deben tener un ViewHandler por defecto. La producción se realiza mediante la subclase de ViewHandler registrada en JSF. Forma parte de Facelets que funciona en la capa de presentación en el diseño de páginas. Se conoce también como patrón decorador.

Posee un componente ViewHandler que maneja todas las vistas que el sistema de software tiene. Le ofrece la funcionalidad necesaria para abrir, coordinar, cerrar vistas específicas y también para manejarlas. Las vistas específicas, junto a la funcionalidad para su presentación y control, están encapsuladas en componentes de vista uno para cada tipo. Provee varios beneficios, dentro de los que se encuentran el manejo uniforme de las páginas y extensibilidad, así como su coordinación.

Este es el componente central de este patrón. Es responsable de abrir vistas nuevas y los clientes pueden especificar la vista que desean. Inicia instantáneamente los correspondientes componentes de vista, cuida su correcta inicialización y pide a la nueva vista que se despliegue. Dentro de sus beneficios están el manejo uniforme de las vistas, extensibilidad de las vistas, así como la coordinación de vistas.

1.6.2 Lenguajes

XHTML (Lenguaje Extensible de Marcas de Hipertexto): Es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas web. XHTML es la versión XML de HTML, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones, más estrictas, de XML. Su objetivo es avanzar en el proyecto del World Wide Web Consortium de lograr una web semántica, donde la información, y la forma de presentarla estén claramente separadas. En este sentido, XHTML serviría únicamente para transmitir la información que contiene un documento, dejando para hojas de estilo y JavaScript su aspecto y diseño en distintos medios (ordenadores, PDAs, teléfonos móviles, impresoras).

JavaScript: Lenguaje de programación interpretado, con capacidades elementales orientadas a objeto. El código JavaScript es embebido directamente en el código HTML o XHTML, haciendo fácil la creación de páginas web con contenido dinámico. Está diseñado para controlar la apariencia y manipular los eventos dentro de la ventana del navegador web y es soportado por la gran mayoría de los navegadores. Este lenguaje ligado con el HTML o XHTML permite hacer aplicaciones web más potentes.

Java: Lenguaje orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. Tiene un modelo de objetos simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores,

como la manipulación directa de punteros o memoria. Está pensado expresamente para una arquitectura cliente/servidor en la que sólo es necesario intercambiar pequeñas porciones de código (llamadas Applets) que son ejecutadas por el cliente.

Una de las características que favoreció su crecimiento y difusión es la capacidad de que el código funcione sobre cualquier plataforma de software y hardware. Este programa escrito para Linux puede ser ejecutado en Windows sin ningún problema. Además es un lenguaje orientado a objetos que resuelve los problemas en la complejidad de los sistemas.

1.6.3 Servidor de Aplicación

JBoss 4.2: servidor de aplicaciones J2EE de código abierto implementado en Java puro. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo que lo soporte. Es el primer servidor de aplicaciones de código abierto, ofrece una plataforma de alto rendimiento para aplicaciones de e-business. Al combinar una arquitectura orientada a servicios con una licencia de código abierto puede ser utilizado, incrustado y distribuido sin restricciones por la licencia.

Las características destacadas de JBoss incluyen:

- Producto de licencia de código abierto sin coste adicional.
- Cumple los estándares.
- Confiable a nivel de empresa.
- Incrustable, orientado a arquitectura de servicios.
- Flexibilidad consistente.

1.6.4 Estándares

Java Server Faces (JSF): especificación de JCP que define una tecnología, especifica un estándar para hacer aplicaciones web ordenadamente. Introducida a través de la JSR 127 por Sun MicroSystem en

mayo del año 2001, dando a conocer la versión 1.1 de JSF. En mayo del año 2006 publicó la JSR 252 con la versión 1.2. Actualmente JCP trabaja en la versión 2.0 de JSF. Es un framework de interfaces de usuario del lado de servidor para aplicaciones Web sobre tecnología Java, basada en el patrón MVC.

Es un estándar sencillo que aporta los componentes básicos de las páginas Web además de permitir crear componentes más complejos (menús, pestañas, árboles, etc.). Hay disponibles diferentes implementaciones de la especificación, tanto comerciales como de código abierto, así como librerías de componentes adicionales que amplían la funcionalidad de esos componentes iníciales.

Está orientado a componentes, los maneja como una entidad básica que existe en ambos lados de la petición. Es un marco de trabajo de componentes de interfaces web basadas en Java. Encaja en la arquitectura basada en MVC ofreciendo una clara separación entre el comportamiento y la presentación del sistema. No se limita a una tecnología de script o lenguaje de marcas particular. JSF es más sencillo, flexible y permite crear componentes. Puede tener distintas implementaciones, una de ellas es la de elaborar código fuente necesario a partir de los estándares definidos, de modo que se puede utilizar como framework.

El foco de desarrollo de JSF se centra en:

1. Definir un conjunto simple de clases base de **Java** para componentes de la interfaz de usuario, estado de los componentes y eventos de entrada. Estas clases tratarán los aspectos del ciclo de vida de la interfaz de usuario, controlando el estado de un componente durante el ciclo de vida de su página.
2. Proporcionar un conjunto de componentes para la interfaz de usuario, incluyendo los elementos estándares de HTML para representar un formulario. Estos componentes se obtendrán de un conjunto básico de clases base que se pueden utilizar para definir componentes nuevos.
3. Proporcionar un modelo de **JavaBeans** para enviar eventos desde los controles de la interfaz de usuario del lado del cliente a la aplicación del servidor.
4. Definir **APIs** para la validación de entrada, incluyendo soporte para la validación en el lado del cliente.
5. Automatizar la generación de salidas apropiadas para el objetivo del cliente, teniendo en cuenta todos los datos de configuración disponibles del cliente, como versión del navegador.[4]

Puede compararse con otras tecnologías basadas en MVC como (Spring MVC, Struts, WebWork, Tapestry), es más orientada a componentes y eventos. Aunque no hay gran experiencia con su uso, es mucho más avanzado, tiene mejores expectativas de desarrollo y permite hacer el software más rápido y fácil.

Es importante destacar de la tecnología JSF que sus componentes de interfaz de usuario están representados en el servidor como objetos con estado**.** Esta característica permite a la aplicación manipular el estado del componente y conectar los eventos generados por el cliente del lado del servidor.

Permite convertir y validar entradas sobre componentes individuales y reportar cualquier error antes de que se actualicen los datos en el lado del servidor. La librería de componentes elimina la necesidad de codificar componentes de interfaz de usuario en HTML u otro lenguaje de marcas, hecho que lo hace totalmente reutilizables. Su librería principal facilita registrar eventos, validadores y otras acciones de los componentes.

Va más allá de un servlet, pues este último cubre la infraestructura básica necesaria para construir una aplicación web, manejando las propiedades del protocolo HTTP. JSF manipula el API Servlet para el funcionamiento interno. Este suministra una gran variedad de funcionalidades sobre la seguridad, la administración de usuarios, el ciclo de vida de los eventos, los filtros y otros elementos que constituyen la base fundamental de JSF.

Su arquitectura principal está diseñada para ser independiente de un protocolo específico. Su objetivo es llegar a una solución para los problemas más comunes encontrados en el desarrollo de aplicaciones web para clientes HTML. Se comunica a través del protocolo HTTP con servidores de aplicaciones Java.

1.6.5 AJAX

AJAX: unión de varias tecnologías que se desarrollan de forma autónoma y que se unen de formas nuevas y sorprendentes.

AJAX se da a conocer por primera vez en el artículo Ajax: “A New Aproach to Web Applications”, publicado por Jesse James Garret el 18 de febrero de 2005 y a partir de este momento se normaliza un nuevo tipo de modelo. Ajax significa JavaScript asíncrono+ XML y en el artículo se define como a continuación se presenta:

“Ajax no es una tecnología en sí mismo. En realidad, se trata de la unión de varias tecnologías que se desarrollan de forma autónoma y que se unen de formas nuevas y sorprendentes.”[5]

El conjunto de tecnologías a las cuales este acrónimo hace referencia son:

1. **Objeto XMLHttpRequest**: es quien permite la comunicación asincrónica entre el navegador y el servidor Web, sobre todo con el empleo de XML.
2. **DOM (Document Object Model- modelo de documento de objeto):** forma de presentar por medio de objetos los documentos XML y HTML. Junto con javascript permite modificar dinámicamente los componentes de la interfaz de usuario.
3. **CSS (Cascading Style Sheets):** plantillas de estilos en cascada, lenguaje que sirve para describir la forma de generar las páginas Web.
4. **JavaScript:** lenguaje de script de programación orientado a los objetos que constituye todos los elementos de AJAX que representa la lógica de negocios de la aplicación y que permite el comportamiento dinámico del navegador.

Dentro de sus características se encuentran:

- **Basada en estándares abiertos:** las tecnologías que lo conforman son estándares abiertos, excepto el objeto XMLHttpRequest.
- **Soportada por los navegadores más utilizados en Internet**: Internet Explorer, Mozilla y Opera.
- **Gran usabilidad:** permite a las páginas hacer una petición muy pequeña al servidor y recibirla sin necesidad de recargar la página completa.
- **Independiente del tipo de tecnología de servidor que se utilice:** AJAX funciona con cualquier navegador, es compatible con cualquier tipo de servidor estándar y lenguaje de programación web, ejemplo: PHP, ASP, ASP.Net, Perl, etc.

- **Perfecciona la estética de la Web:** es posible combinar la creatividad del desarrollador con la usabilidad de la aplicación de manera tal que si una aplicación no estuviera dentro de un navegador, pudiera pasar por una aplicación de escritorio.

El empleo simultáneo de tales tecnologías ocasionan una parcial recarga de la página como resultado de las acciones del usuario. Los resultados no relacionados con la interacción quedarán visibles y no se les hará modificaciones durante el intercambio de información. Este hecho posibilita el continuo empleo del sistema, evitando recargas completas del contenido de la página, contribuyendo al ahorro de tiempo de respuesta de la petición. El siguiente esquema muestra la diferencia más importante entre una aplicación web tradicional y una aplicación web creada con AJAX.

Como se observa en la figura 1.5, ante los eventos del usuario, en la aplicación que utiliza AJAX crea y configura un objeto XMLHttpRequest; este es el encargado de realizar las llamadas al servidor. Luego, el servidor retorna un documento XML que contienen el resultado y el objeto XMLHttpRequest lo procesa. Finalmente se actualiza el DOM de la página asociado con el resultado devuelto.

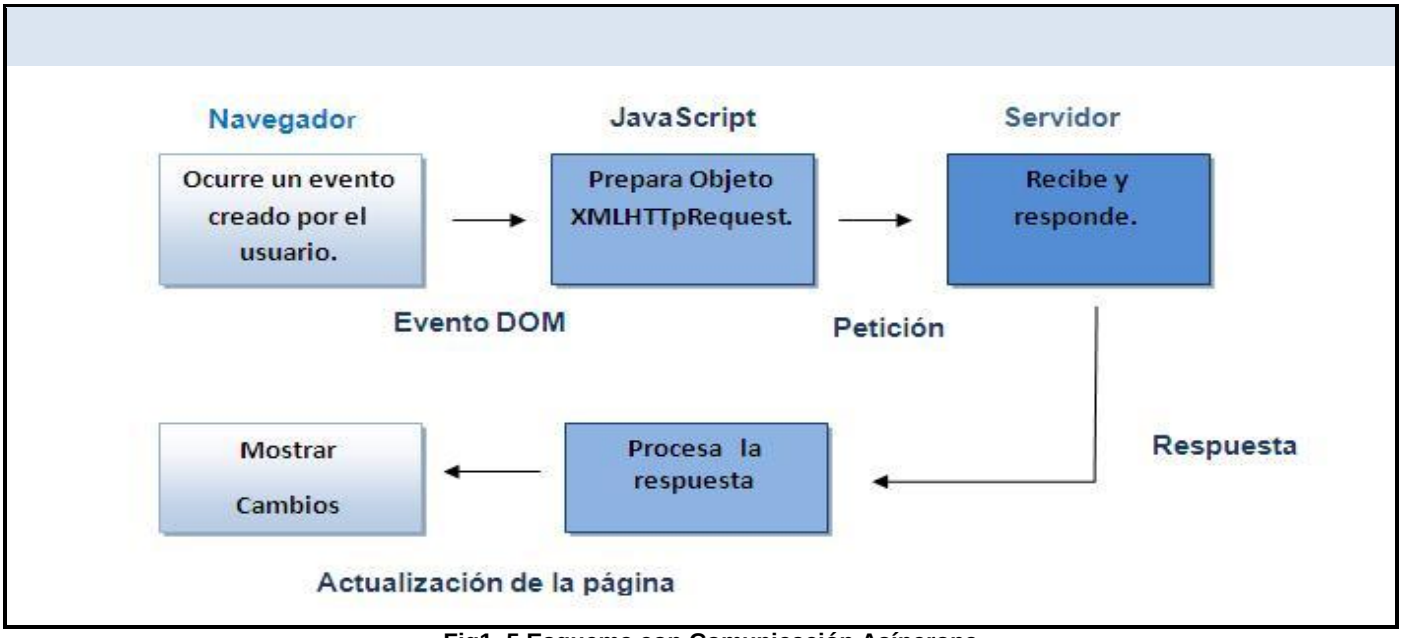

Fig1. 5 Esquema con Comunicación Asíncrona.

Los desarrolladores que usan AJAX con el afán de perfeccionar la capa de presentación de sus aplicaciones web, introducen entre el usuario y el servidor un motor denominado “motor AJAX”. Este

posibilita que al iniciarse la sesión no se cargue una página web, sino que se carga código JavaScript (en esencia es el nombrado motor AJAX). El mismo tiene la responsabilidad de mostrarle al usuario la interfaz y comunicarlo con el servidor; este intercambio es denominado asíncrono. El producto de tal comunicación es la eliminación de todo indicio de espera por una respuesta del servidor, como son las ventanas en blanco en el navegador o íconos de reloj de arena.

Con este uso, los requerimientos HTTP que generan las acciones del usuario toman la forma de llamadas Javascript al motor AJAX. Si las respuestas a estas acciones no requieren de un viaje de vuelta al servidor (en este caso particular se encuentra la validación de datos o edición de estos en memoria), estas las maneja el motor AJAX. En caso de que se necesite del servidor para responder (cargar código adicional, envío o recuperación de datos) se hacen estos pedido asíncronamente, usualmente a través de código XML; posibilitando que no se frene la interacción del usuario con la aplicación.

Al recibir la respuesta el motor AJAX se realizan los cambios en la interfaz que ve el usuario, pero solo basado en la información que se obtiene. La actualización de la página se hace de forma rápida pues se disminuye la cantidad de información transmitida.

Dentro de los usos más comunes se encuentran:

- Refrescador de datos.
- Notificaciones del servidor.
- Control en árbol, menús, barras de progreso.
- Actualización o eliminación de registros.
- GUI avanzadas.
- Validación de datos de formularios en tiempo real.
- Identificación de usuario, números de serie, códigos postales u otro código especial que necesite validación en el lado del servidor antes de ser enviado el formulario.
- Expansión de formularios web.
- Devolución de peticiones simples de búsqueda.

1.6.6 Framework

Seam: framework de código abierto (open sourse) desarrollado por la empresa JBoss, con el fin de unir diferentes tecnologías y estándares en Java en un solo framework. Esto aporta la ventaja de que une todas las funcionalidades necesarias para desarrollar software de alta calidad. Su arquitectura está basada en el MVC. Utiliza los siguientes frameworks:

Modelo: EJB3 o SeamBeans.

Vista: JSF y las librerías RichFaces/ Ajax4jsf

Controlador: Utiliza tanto Stateless/Statefull Session beans como Seam Beans.

Dado que Seam está enfocado principalmente en desarrollos web, toma como base el estándar JSF, el que puede o no estar apoyado en Facelets para el desarrollo de vistas. Con esto incluye una serie de librerías (RichFaces/ Ajax4jsf), dado que también son propiedades de JBoss.

Seam se sienta encima de Java EE 5.0 para proporcionar un consistente y comprensible modelo de programación para todos los componentes en una aplicación Web empresarial. El diseño de Seam ha sido concebido con Ajax en mente. Es capaz de manejar peticiones simultáneas de distintos usuarios preservando las condiciones de aislamiento e integridad de los datos. El soporte en la parte del cliente está a cargo de la implementación de JSF que se elija. En cualquier caso, mediante la librería Ajax4jsf de JBoss se pueden añadir funcionalidades Ajax a los componentes ya existentes.

RichFaces: biblioteca de componentes para JSF y un avanzado framework de código abierto para la integración de AJAX, con facilidad en la capacidad de desarrollo de aplicaciones de negocio. Los desarrolladores que lo utilizan aprovechan las características de estos componentes para crear aplicaciones Web que proporcionan mejoras en gran medida la experiencia del usuario. RichFaces permite**:**

- Intensificar el conjunto de beneficios JSF al trabajar con Ajax. RichFaces está completamente integrado en el ciclo de vida de JSF.

- Añadir capacidad Ajax a aplicaciones JSF. El framework proporciona dos librerías de componentes (Core Ajax y la interfaz de usuario). La librería Core permite agregar la funcionalidad Ajax en las páginas sin necesidad de escribir nada de código JavaScript. Permite definir eventos en la propia página. Un evento invoca a una petición Ajax, sincronizándose así zonas de la página y componentes JSF después de recibir la respuesta del servidor por Ajax.
- Crear rápidamente vistas complejas basándose en la caja de componentes. La librería UI (Interfaz de usuario) que contiene componentes para agregar características de interfaz de usuario a aplicaciones JSF. Se amplía el framework de RichFaces incluyendo un gran conjunto de componentes "habilitación de Ajax" que extiende el soporte de la página. Además, los componentes de RichFaces están diseñados para ser usados sin problemas con otras librerías de componentes en la misma página, de modo que existen más opciones para el desarrollo de aplicaciones
- Proporciona un paquete de recursos con clases de aplicación Java. Además de su núcleo, la funcionalidad de Rich Faces para Ajax proporciona un avanzado soporte a la gestión de diferentes recursos: imágenes, código JavaScript y hojas de estilo CSS. El framework de recursos hace posible empaquetar fácilmente estos recursos en archivos jar junto con el código de los componentes personalizados.
- Generar fácilmente recursos binarios sobre la marcha. Los recursos del framework pueden generar imágenes, sonidos, hojas de cálculo de Excel, etc.

Los componentes de la interfaz de usuario de RichFaces vienen preparados para su uso fuera del paquete, así los desarrolladores ahorrarán tiempo y podrán disponer de las ventajas mencionadas para la creación de aplicaciones Web. Como resultado, la experiencia puede ser más rápida y fácil de obtener.

Aprovecha al máximo los beneficios de JSF framework incluyendo, la validación y conversión de instalaciones, junto con la gestión de estática y dinámica los recursos. [6]

Facelets: framework para plantillas centrado en la tecnología JSF. Dentro de las características más importantes se encuentran:

- Facilidad en la creación del templating para los componentes y páginas.
- Un buen sistema de reporte de errores.

- Soporte completo a EL (Expression Language).
- Validación de EL en tiempo de construcción.
- No es necesaria configuración XML.
- Fácil composición de componentes.
- Creación de etiquetas lógicas a la medida.
- Funciones para expresiones.
- Desarrollo amigable para el diseñador gráfico.
- Creación de librerías de componentes.

Permite definir vistas JSF utilizando plantillas del tipo HTML, reduciendo el código innecesario para agregar componentes en la vista y que no necesariamente sea un contenedor web. Es un framework simplificado de presentación, en donde es posible diseñar de forma libre una página web y luego asociarle los componentes JSF específicos. Aporta mayor libertad al diseñador y mejora los informes de errores que tiene JSF. Es un framework de diseño de páginas muy fácil de usar y configurar.

Ajax4jsf: librería de código abierto que se integra totalmente en la arquitectura de JSF, se extiende la funcionalidad de sus etiquetas dotándolas con tecnología Ajax de forma limpia y sin añadir código Javascript. Con su utlización se puede variar el ciclo de vida de una petición JSF, recargar determinados componentes de la página sin necesidad de recargarla por completo, realizar peticiones al servidor automáticas y control de cualquier evento de usuario. Permite dotar a la aplicación JSF de contenido mucho más profesional con muy poco esfuerzo.

1.6.7 Herramientas

Aptana: Aptana Studio es un IDE de código abierto basado en plataforma de herramientas de Eclipse.

Con su uso se facilita el desarrollo integrado de Ajax con las tecnologías emergentes. Basado en el entorno de desarrollo Eclipse (IDE = Integrated Development Environment), también Open Source. Es una distribución focalizada en el desarrollo web, con soporte a HTML, CSS y Javascript, así como

opcionalmente a otras tecnologías mencionadas como PHP, Adobe Air o Ruby on Rails. Está disponible como una aplicación independiente o como plug-in para Eclipse.

Eclipse 3.3.x: poderosa herramienta que permite integrar diferentes aplicaciones para construir un entorno integrado de desarrollo (IDE). Es un proyecto de desarrollo de software open-source, que está dividido en tres partes: el proyecto **Eclipse** Proyect, **Eclipse** Tools, y **Eclipse** Technology Proyect.

Diseñada para afrontar las siguientes necesidades:

- Soportar la construcción de gran variedad de herramientas de desarrollo.
- Soportar herramientas que permitan manipular diferentes contenidos (HTML, Java, C, JSP, EJB, XML, y GIF).
- Facilitar una integración transparente entre todas las herramientas y tipos de contenidos sin tener en cuenta al proveedor.
- Proporcionar entornos de desarrollo gráfico (GUI) o no gráficos.
- Ejecutarse en una gran variedad de sistemas operativos, incluyendo Windows® y Linux™.
- Su objetivo es proporcionar mecanismos, reglas que puedan ser seguidas por los fabricantes para integrar de manera transparente sus herramientas mediante APIs interfaces, clases y métodos, se exponen estos mecanismos.
- Posibilita la construcción nuevas herramientas que extenderán la funcionalidad de la misma.

Photoshop: Adobe Photoshorp® (Ps) es una aplicación en forma de taller de pintura y fotografía que trabaja sobre un ″lienzo″ y que está destinado para la edición, retoque fotográfico y pintura a base de imágenes de mapa de bits o conocidos en Photoshop como gráficos rasterizados, elaborado por la compañía de software Adobe Systems inicialmente para computadores Apple pero posteriormente también para plataformas PC con sistema operativo Windows.

En este capítulo se valoraron las tendencias actuales para de los sistemas de gestión de información, resultando significativo en la comprensión de las necesidades de proporcionar una nueva propuesta de diseño para el SIAPS. Se realizó un análisis de las tecnologías y herramientas existentes vinculado con el objeto de estudio y se exponen los conceptos fundamentales asociados al dominio de estos sistemas.

Capítulo 2. Definición del Diseño de la Capa de Presentación

La manera de representar la información a los usuarios en un sistema informático forma parte de su éxito. Este es el resultado de la implementación de una serie de tecnologías que facilitan la transmisión de contenidos desde un servidor a la comunidad de consumidores; por esta razón es muy importante que la información ofrecida cumpla con ciertas reglas y/o estándares.

En este capítulo se definen los elementos básicos para la creación del diseño de la capa de presentación del SIAPS. Se precisa la arquitectura de información que poseerán las interfaces, la organización de la misma, así como el sistema de navegación y etiquetado a utilizar. Se muestra el diseño conceptual de las páginas y se delimitan las pautas a seguir en el diseño de estas. Se incluye además la propuesta de plantillas para las interfaces de usuario.

2.1 Ergonomía. La Ley de Fitt

La Ergonomía es una disciplina científica que estudia la relación de las interacciones humanas y elementos de un sistema, aplicando principios, datos y métodos a fin de optimizar el bienestar humano y el rendimiento global de la aplicación. Ella se fundamenta en que el diseño de un producto debe enfocarse a partir del conocimiento de cuáles son las capacidades, habilidades y limitaciones de los usuarios que interactuarán con el sistema. Su objetivo es adaptar los productos, las herramientas, los espacios y el entorno en general a las necesidades de las personas. El planteamiento ergonómico consiste en diseñar productos adaptables a las personas y no al contrario.

Dentro de las áreas de su especialización o profundización se encuentra la Ergonomía Cognitiva, interesada en los procesos mentales, tales como percepción, memoria, razonamiento y respuesta motora, en la medida que estas afectan las interacciones entre los seres humanos y los otros elementos componentes de un sistema. Una de las cuestiones relevantes de esta rama se encuentra la interacción humano-computadora (por ejemplo, la ley de Fitt).

2.1.2 La Ley de Fitt

Esta es la ley más básica y más conocida de entre las leyes del diseño de interfaces de usuario. Plantea que cuanto más grande y más cercano al puntero del ratón es un objeto, más sencillo es el hacer click sobre él. Esto es de sentido común, pero muchas veces es ignorado completamente en el diseño de interfaces.

Esta es prácticamente la única regla no subjetiva que está establecida dentro del campo del diseño de interfaces. Hace referencia a las características que tienen que tener los objetos para que sea más fácil o difícil pulsarlos: posición en la que se encuentren, su tamaño y la expresividad de dichos elementos. El último concepto da a entender que un objeto será más visible si este expresa su existencia como objeto de interfaz y no como simple información (ya sea dato o imagen).

Un ejemplo es el cambio que sufren ciertos enlaces web al pasar por encima de ellos con el puntero del ratón (por ejemplo cambiando de color). Cualquiera puede pensar que todo lo planteado por la Ley de Fitts es absolutamente evidente, y de hecho generalmente lo es. Más sorprendente es la gran cantidad de veces que no se aplica, logrando por tanto que la usabilidad de la interfaz se vea claramente comprometida.

2.2 Arquitectura de Información

La Arquitectura de Información organiza conjuntos de información, permitiendo que cualquier persona los entienda y los integre a su propio conocimiento, de manera simple. Cuando este concepto se utiliza en la construcción de aplicaciones web, es posible agregar que la Arquitectura de Información es el conjunto de prácticas que organiza el contenido en subconjuntos de nombres, facilitando el entendimiento del objetivo de tales sistemas.

Es una disciplina relativamente nueva que nace tras la definición que hace Richard Saul Wurman en 1975, en su libro titulado “Information Architects”. Allí plantea que el arquitecto de Información es “la persona que organiza los patrones inherentes a la información, haciendo entendible lo complejo” o que también crea un

mapa de información que permite a otros encontrar su vía personal hacia el conocimiento". Es una mezcla de tres campos: tecnología, diseño gráfico y periodismo/redacción.

Años más tarde se publican nuevas definiciones acerca de este nuevo campo, en el libro "Information Architecture for the World Wide Web - Designing Large-scale Web Sites", escrito por Louis Rosenfeld y Peter Morville. Su éxito hizo que se transformara en la biblia de quienes realizan este tipo de tarea. En varias de estas definiciones no estaban de acuerdo con Wurman, debido a que sus explicaciones no se adaptaban del todo bien a los sitios web.

Estos autores plantean que la Arquitectura de Información es:

- La combinación de esquemas de organización, etiquetado y navegación, dentro de un sistema de información.
- El diseño estructural de un espacio de información para facilitar la terminación de tareas y el acceso intuitivo al contenido.
- El arte y la ciencia de estructurar y clasificar sitios web e intranets, comunidades en línea y software, para promover la usabilidad y facilidad de encontrar información.

Con la Arquitectura de Información aparecen otros elementos como resultado del buen uso de este concepto. Estos son los campos del Diseño de Experiencia, referido a que el proceso de acceder a la información (incluye forma y contenido) sea planificado para ofertar una mejor estructura de la aplicación al usuario final; y la Usabilidad, el que hace alusión a la facilidad de uso que tendrán las interfaces ofrecidas, permitiendo una buena navegación y un buen entendimiento de la información ofertada de manera simple e intuitiva.

La UCI cuenta con el documento Línea Base de Arquitectura de Información para Sistemas de Gestión, el cual se utilizó como punto de partida para la organización de la información en las pantallas del SIAPS. Este documento define los siguientes elementos:

- Definición de las posibles pantallas del sistema.
- Definición de áreas en pantalla: Establece las áreas que han sido identificadas como estándar en los análisis realizados en los sistemas de gestión nacional e internacional y su descripción.

- Definición de los contenidos genéricos por áreas: Describe las características funcionales de los contenidos identificados como genéricos para los sistemas de gestión.
- Descripción de los elementos de la estructura: Descripción textual de cada uno de los elementos de la estructura, características, comportamiento.
- Taxonomía de la línea base: Representa el etiquetado y jerarquía de los contenidos dentro del sistema.
- Diseño de la estructura de las pantallas tipo: Representación lineal de cada uno de los elementos que componen la pantalla tipo, con el objetivo de verificar la ubicación de cada uno de ellos.

2.2.1 Diseño Conceptual: Descripción y Organización de la Información

Diseñar es el proceso de creación y desarrollo de un nuevo objeto o medio de comunicación (objeto, proceso, servicio, conocimiento o entorno) para uso humano. Diseño es el fruto del proceso de diseñar (dibujo, proyecto, maqueta, plano o descripción técnica), o el resultado de poner ese plan final en práctica (la imagen o el objeto producido).

Esta tarea requiere determinadas consideraciones funcionales y estéticas; necesita de numerosas fases de investigación, análisis, modelado, ajustes y adaptaciones previas a la producción definitiva del objeto.

Una de las actividades en el diseño web es el diseño conceptual, este describe la arquitectura de información y navegación, con él se desarrolla un modelo de la aplicación que esté en sintonía con el modelo mental y expectativas del usuario. Para crear este modelo se acude al desarrollo de los prototipos o plantillas.

Las plantillas o prototipos, permiten estructurar y disponer de todos los elementos seleccionados de lo que será un esbozo de las interfaces de la aplicación. Estas se componen por el conjunto de elementos que permiten al usuario realizar acciones sobre el sistema. Se considera parte de la interfaz a sus elementos de identificación, de navegación, de contenidos y de acción.

Se ha comprobado a través investigaciones la existencia, ubicación e importancia de dichos elementos. Con ellas se demuestra que las zonas que normalmente se ven en una visita inicial, están conformadas por una letra F o por un triángulo, cuya sección más revisada es la que se encuentra en la esquina superior izquierda.

Teniendo en cuenta esta evidencia, las interfaces van a poseer ciertas características, con el fin de asegurar que los usuarios finales reciban la información adecuadamente, permitiendo el mejor uso de los contenidos y funcionalidad. Respecto a los elementos de la interfaz, los aspectos más relevantes a tener en consideración son los siguientes: uso de logotipos, sistema de navegación, áreas de contenidos, experiencia de usuario.

En los siguientes prototipos, llamados también wireframe, se muestra la ubicación relativa a cada uno de los elementos que conformarán las interfaces del SIAPS:

Este prototipo alude a la representación de la interfaz de autenticación.

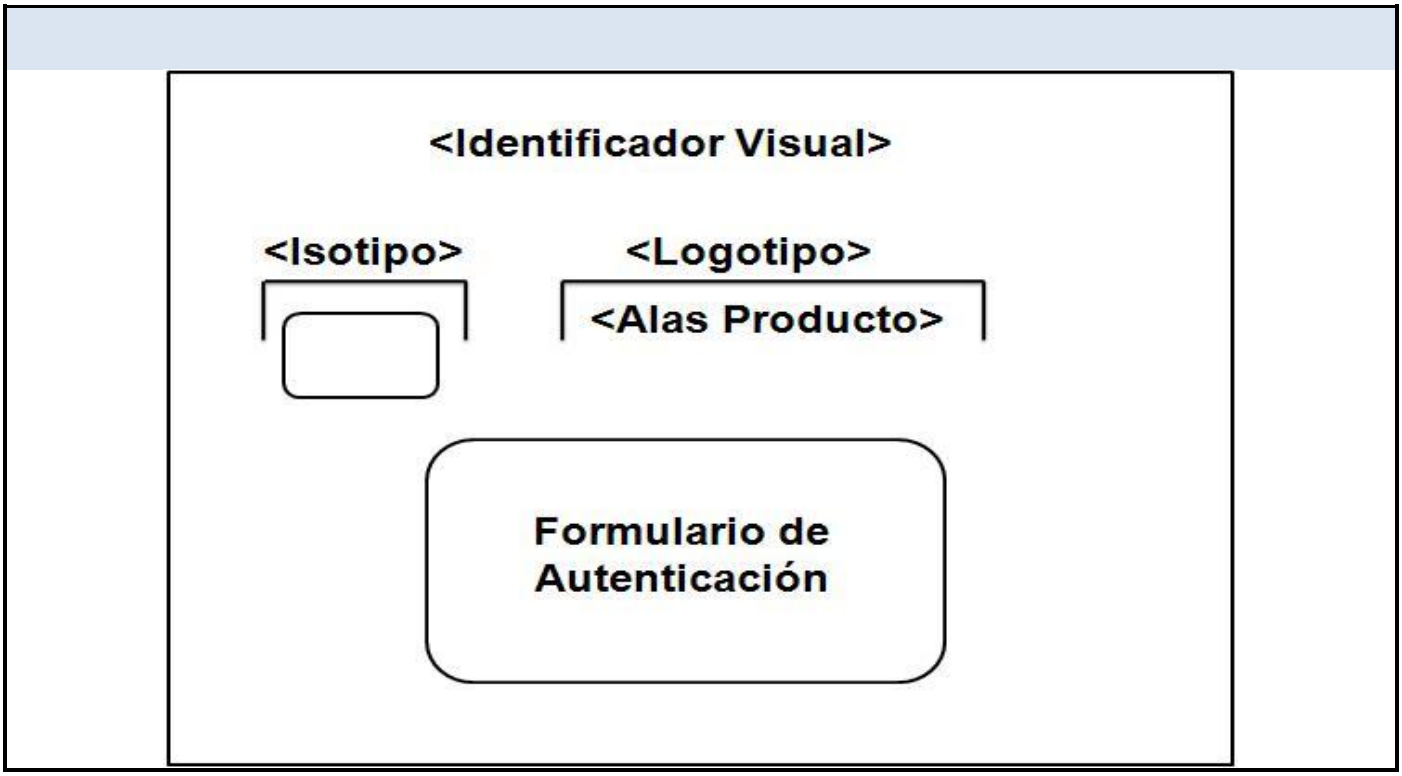

Fig2. 1 Prototipo de Autenticación.

Estructura taxonómica definida:

Pantalla de Bienvenida:

1. Identificador Visual.
2. Formulario de Autenticación.

Este segundo prototipo alude a la representación de la interfaz de acceso a los módulos luego de haberse autenticado el usuario.

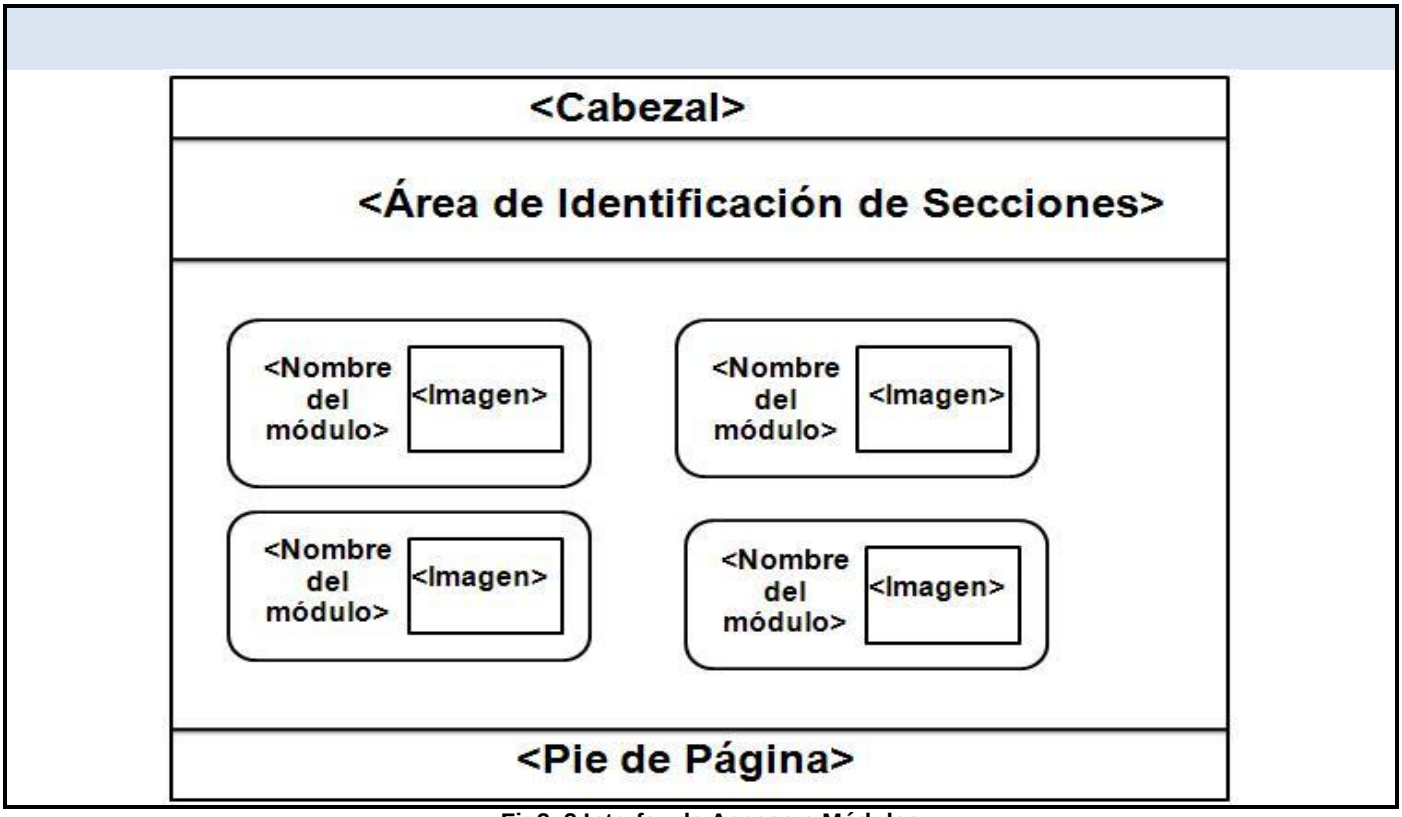

Fig2. 2 Interfaz de Acceso a Módulos.

Estructura taxonómica definida:

Pantalla de Acceso a Subsistemas y Módulos:

1. Cabezal

2. Identificador Visual.
3. Componente para acceder a los diferentes módulos (compuesto por una imagen y el nombre del módulo).
4. Pie de Página.

Este tercer prototipo alude a la representación de la interfaz genérica de los subsistemas.

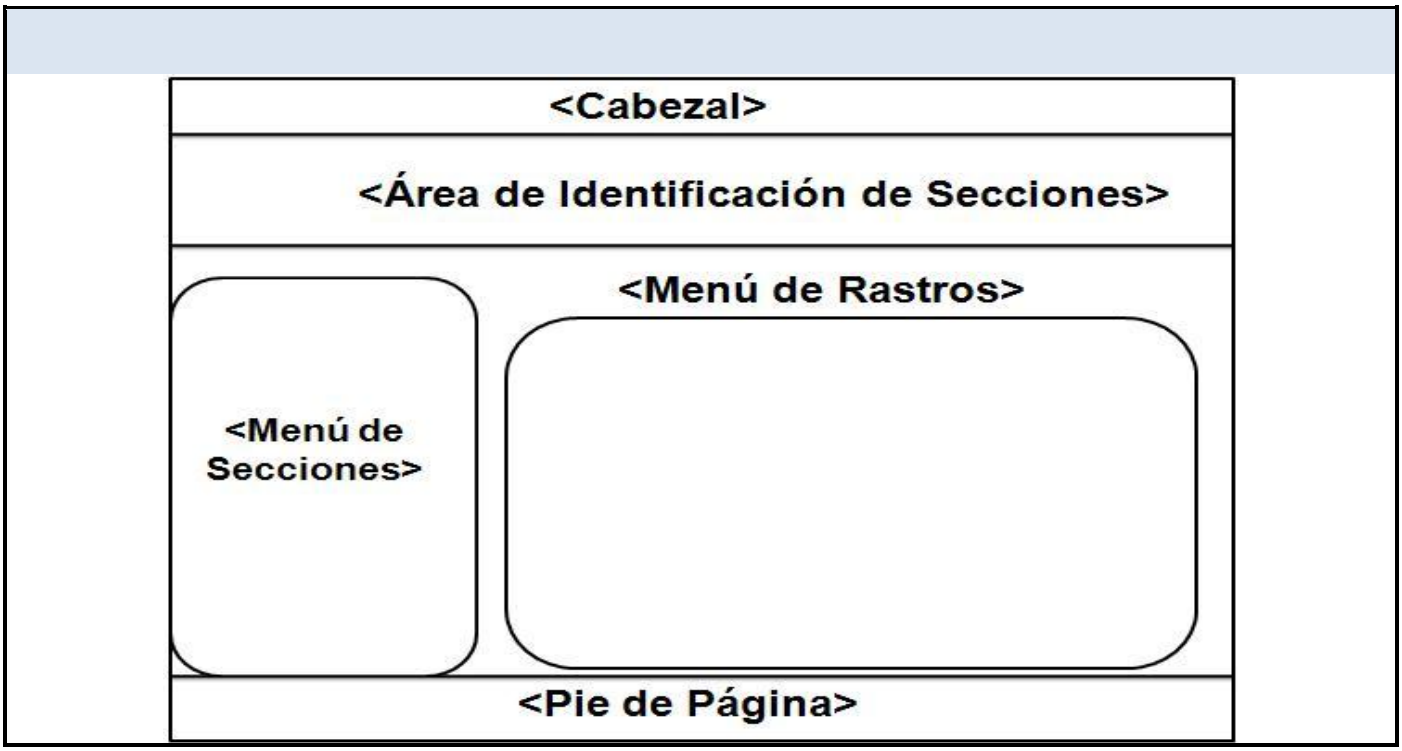

Fig2. 3 Interfaz Genérica.

Estructura taxonómica definida:

Pantalla Tipo Genérica:

1. Cabezal.
2. Área de Identificación de Secciones.
3. Usuario Autenticado.

4. Pie de Página.

2.2.2 Descripción de elementos

2.2.2.1 Sistema de Navegación

El diseño de un Sistema de Navegación (SN) es necesario para brindar un sentido del entorno y permitir flexibilidad en la navegación dentro de la aplicación.

Se denomina SN al conjunto de elementos presentes en cada una de las pantallas, que permite a un usuario moverse por las diferentes secciones de una aplicación web y retornar hasta la portada, sin sentir la sensación de haberse perdido en ese camino.

Es válido aclarar que existen varios tipos de SN, los cuales son: Sistemas de Navegación Jerárquicos (SNJ), Sistemas de Navegación Globales (SNG), Sistemas de Navegación Locales (SLN), Sistemas de Navegación Específicos (SNE). Teniendo en cuenta las características propias de un sistema de gestión de información, como lo es el SIAPS, se definió utilizar el SNG.

Este SN generalmente complementa a uno jerárquico. Le brinda al usuario la posibilidad de navegación tanto a lo profundo como a lo largo del sitio (navegación vertical y horizontal). Este se basa en una barra de navegación gráfica que permite el acceso a las principales secciones, cada sección en su interior se complementa con otras barras o menús laterales relativos a estas. Los elementos que enriquecen al diseño del SN son gráficos, barras de menús horizontales y verticales; así como dos reglas que evitan la pérdida y desorientación del usuario dentro del sistema, las cuales se exponen a continuación:

- Inclusión del nombre de la organización en todas las páginas de la aplicación.
- Representación de la estructura jerárquica de una manera clara y consistente para indicar la localización exacta dentro de esta jerarquía.

En el diseño web se contempló que el sistema de navegación cuente con los siguientes elementos para conseguir su objetivo:

Servicios: elementos que permiten realizar acciones directas relativas a la navegación y que se muestran como parte de ésta, tales como los correspondientes a: “Inicio”, “Ayuda”, “Manual de Usuario”, “Cerrar Sección”. Se encuentran en el cabezal de las páginas genéricas y en la de acceso a módulos.

- Inicio: Link que permite regresar a la Pantalla de Bienvenida
- Ayuda: Acceso a la ayuda del sistema.
- Manual de Usuario: Acceso al manual de usuario.
- Cerrar Sección: Acceso que permite cerrar la sección.

Identificador Visual: constituye la representación visual de los atributos que desea transmitir la Marca para generar la imagen del producto. Compuesto por el isotipo y el logotipo de la marca.

- Isotipo: tributa a la imagen que persigue el producto. Influye en la percepción de solidez y fortaleza de la solución. Está conformado por elementos que parecieran girar sobre sí mismos, en movimiento perpetuo, ligero y dinámico, gravitando sobre un centro en espiral, que le confiere dinamicidad, ligereza y cierta organicidad con vida propia dentro del sistema.

- Logotipo: constituye la frase hecha a imagen de aquello que se va a enunciar. En el caso del proyecto APS, dirigido al ámbito de la Informática Médica, está compuesto por la palabra Alas que tributa a la suavidad, ligereza y dinamismo de la atención médica en el SNS en Cuba. Estará acompañada de las siglas SIAPS las cuales orientan al usuario que está en presencia del Sistema Integral para la Atención Primaria de Salud. El nombre genérico de la marca hará referencia a las palabras Atención Primaria de Salud.

Menú principal: zona de la interfaz en la que se detallan las secciones o categorías en las que está dividida la información. Se ubica en la zona superior izquierda de cada interfaz. Tiene un ancho de un 20 %.

Identificación de usuario: especifica el usuario autenticado y el rol que desempeña. Ejemplo: Usuario: Leonardo González Rol: Diseñador.

Menú de rastros: indica mediante los nombres de cada sección o categoría del menú, la distancia que separa a la página actual de la portada. Por ejemplo, si el usuario está revisando la página del “Sección

A", el menú correspondiente debe indicar Portada > Secciones > "Sección A". Se encuentra debajo de la identificación de la sección o categoría y sobre el título. Se encuentra incluido en el área de contenidos.

Área de identificación de secciones: se ubicó en la zona superior de la página, de manera cercana a la al identificador visual de la aplicación. Está caracterizada por poseer una imagen alusiva a la sección, incorpora además texto para destacar el nombre de la misma. Aparece en todas las pantallas que pertenecen a esta. Para su adecuada visualización tiene un ancho de 35 %, su largo de 100 %.(lo mismo).

Área de contenidos: zona donde se muestra la información en cada interfaz pedida por el usuario. Tiene de largo un 80 % y de ancho un 70 %.

Pie de página o footer: zona inferior donde se completa la información que se ofrece en las zonas superiores de navegación. La parte izquierda hará referencia a los derechos de la entidad donde se elaboró el producto, el año de creación y el nombre completo del producto; en este caso es Copyright © 2009, UCI-APS. En la parte derecha se mantendrá el isotipo y el logotipo del producto. Tiene un tamaño de 100%.

2.2.1.1 Sistema de Etiquetado

Las etiquetas se utilizan en la representación de un conjunto de información en los sistemas informáticos. Permiten la comunicación entre la aplicación y los usuarios, mostrando la organización de la información y las posibilidades de navegación que presenta. Las etiquetas describen o designan los elementos que integran el sistema de navegación.

Tipos:

Las formas de crear etiquetas pueden ser textualmente o mediante íconos, aunque la combinación de ambas también es frecuente. Habitualmente se emplean con dos objetivos: como vínculos hacia grupos o conjuntos de información en otras páginas (casi siempre dentro del contexto de los sistemas de navegación, como índices, o como etiquetas de vínculos) y como encabezados que dividen e identifican los grupos de información. Una etiqueta simple puede cumplir las dos funciones a la vez.

Las etiquetas que integran el SN requieren una gran consistencia, que brinde sensación de seguridad al usuario además que le permita navegar coherentemente, que pueda reconocer de antemano el paso que va a seguir al estar correctamente identificado con la etiqueta que le corresponde semánticamente y que resulta, a la vez, familiar para el navegante. [7]

Los sistemas de etiquetado pueden dividirse en cuatro tipos:

- **Etiquetas del SN:** son las que interactúan en un primer momento con el usuario. Se toman como referencia para la navegación.
- **Etiquetas de sistemas de enlaces:** aquellas que aparecen en el cuerpo de los párrafos y se enlazan con otros textos en función del contexto y su significado. Debe tratarse que estas resalten lo suficiente dentro del texto y no tengan más de cuatro términos.
- **Etiquetas del sistema de cabeceras o títulos:** se utilizan para encabezar o titular los bloques de información. Hacen el papel de títulos o subtítulos, su significado está condicionado por el contexto.
- **Etiquetas del sistema de indización**: estas etiquetas son "invisibles" para el usuario, tienen una función de suma importancia en la representación del contenido de las páginas para la identificación de estas en los motores de búsqueda. Estas son los META tags.

En el SIAPS se definió la utilización de dos tipos de etiquetas: las del sistema de navegación y las del sistema de cabeceras o títulos. Para contribuir a la satisfacción en la experiencia del usuario cuando interactúa con la aplicación, se decidió usar la combinación de textos e íconos en las opciones del menú. Todas las etiquetas serán en idioma español, serán palabras similares a las operaciones que los usuarios deseen llevar a cabo, ejemplo: Inicio, Acerca de, Ayuda, Manual de Usuario, etc. Esto facilita la comprensión de los contenidos que se muestran e incrementan la usabilidad del sistema.

2.3 Patrones de Diseño

En las aplicaciones web cada página JSPs gestiona servicios de seguridad, de recuperación de contenidos, y la navegación. Esto conlleva a un modelo con alto coste de mantenimiento, con grandes

cantidades de código duplicado en numerosas JSP pues casi siempre en la solución para el desarrollo de la aplicación, se usa la conocida técnica de copiar-pegar y modificar un poco.

Es posible mejorar la calidad de estas aplicaciones centralizando y encapsulando algunos de los mecanismos que le proporcionan a las interfaces claridad y sencillez, cuando se elimina gran cantidad de scriptlets (código java embebido en las páginas JSPs). Para la consecución de estos objetivos no hay nada mejor que la experiencia condensada de muchos años de desarrollo y diseño: los patrones de diseño J2EE.

Un patrón describe una solución a un problema recurrente en el diseño de software, haciendo énfasis en el contexto y las fuerzas que rodean al problema, junto con los resultados y el impacto de la solución. Los patrones crean un lenguaje común a los desarrolladores, para que estos puedan comunicar experiencia sobre los problemas y sus soluciones.

Los patrones ofrecen tres ventajas fundamentales:

• Ya están **probados**: Son soluciones que han sido utilizadas con anterioridad de manera repetida y se ha comprobado que funcionan.

• Son **reutilizables:** Corresponden con problemas que no son específicos de un caso concreto, sino que se presentan una y otra vez en distintas aplicaciones.

• Son **expresivos**: Cuando un equipo de desarrolladores tiene un vocabulario común de patrones, se puede comunicar de manera fluida y precisa las ideas fundamentales sobre el diseño de una aplicación.

Con la aparición de J2EE surgen un grupo de patrones orientados específicamente a los problemas comunes a todas las aplicaciones desarrolladas en esta plataforma. Algunos están basados en los patrones originales mientras que otros son más específicos del tipo de problemas que surgen específicamente en J2EE, bien sea por los tipos de aplicaciones que se suelen desarrollar con la plataforma o por las características (o deficiencias, incluso) de la tecnología. Atendiendo al modelo de arquitectura en capas, distinguiendo entre capa de presentación, capa de negocio y capa de integración, hay definidos una gama de patrones agrupados en dependencia de la capa arquitectónica.

Todos los patrones no dan la solución al problema de diseño y desarrollo del sistema. Como cualquier herramienta o metodología son susceptibles de malos usos y de abusos. En el diseño de la capa de presentación del SIAPS, se define la utilización de dos de estos patrones: los patrones Vista Compuesta (Composite View) y Decorador (Decorator).

Vista Compuesta

Este patrón propone utilizar vistas compuestas que se componen de varias subvistas atómicas. Cada componente de la plantilla se podrá incluir dinámicamente dentro del total y la distribución de la página se maneja independientemente del contenido. Se propone su utilización porque el SIAPS presentará contenido de numerosas fuentes de datos, usando múltiples subvistas, que conforman una única página a visualizar.

El diseño de clases de este patrón es el siguiente:

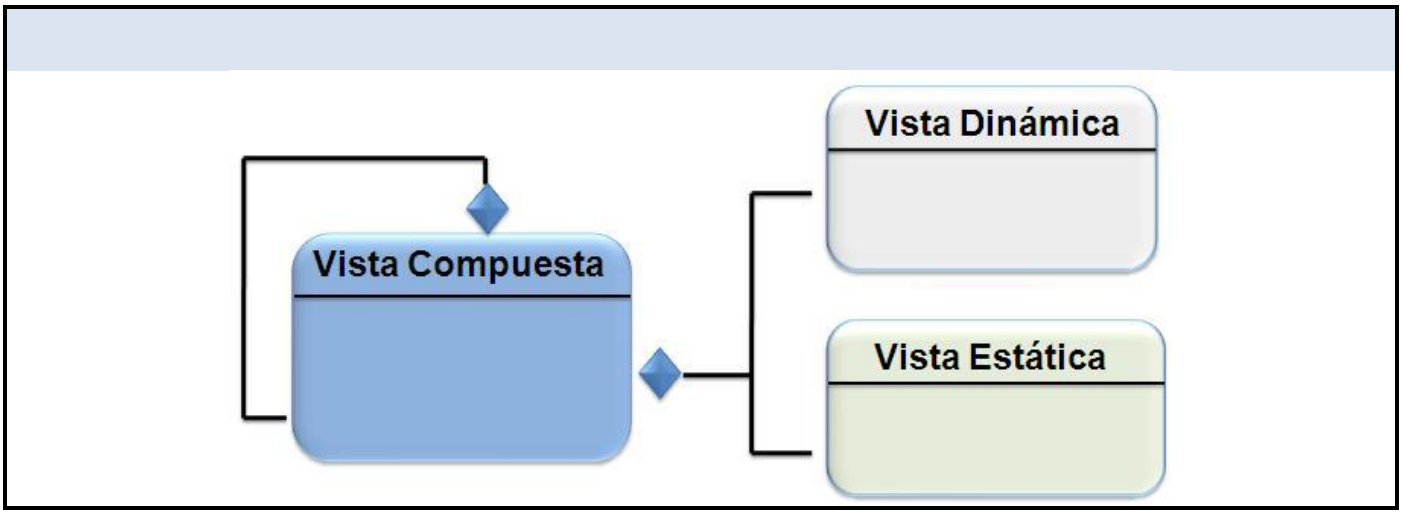

Fig2. 4 Patrón Vista Compuesta (Composite View).

El mecanismo para implementar este patrón se basa en el uso de la etiquete <jsp:include> que puede incluir tanto contenido estático como contenido dinámico en la página cuando esta está siendo servida.

ViewHandler o Patrón Decorador

Todas las implementaciones de JSF deben tener un ViewHandler por defecto. La producción se realiza mediante la subclase de ViewHandler registrada en JSF. Forma parte de Facelets que funciona en la capa de presentación en el diseño de páginas. Se conoce también como patrón decorador.

Se propone utilizarlo en el SIAPS porque brinda funcionalidades adicionales para la manipulación de todas sus vistas, coordina las dependencias entre ellas y organiza su actualización. Va a aportar la posibilidad de que el sistema tenga salida en Wireless Markup Language (WML).

Este patrón separa la presentación del núcleo funcional. No provee una estructura general al sistema por sí mismo, solo quita la responsabilidad de manejar la totalidad de las vistas, sus dependencias mutuas del modelo y los componentes de esta. Es una granularidad más fina que el patrón MVC pues ayuda a refinar las relaciones entre el modelo y sus vistas asociadas.

2.4 Análisis de Frameworks para la Capa de Presentación

Las aplicaciones que tienen como base el estándar JSF, apoyado de Facelets como soporte para la creación de plantillas; así como la utilización de RichFaces y Ajax4jsf para añadir funcionalidades Ajax, poseen mayor robustez y flexibilidad en su arquitectura. Para la comprensión de las nuevas funcionalidades que tales tecnologías aportan, es necesario el análisis del funcionamiento y relación de cada uno. El ciclo de vida de cada una de ellas comienza con una petición realizada por el usuario y concluye cuando se le da respuesta a esta.

2.4.1 JSF. Ciclo de Vida:

La ejecución de JSF comienza al recibir una petición del cliente. El proceso de tránsito de la petición en este framework consta de 6 fases o pasos que a continuación se detallan:

1. **Aplicar los valores de la petición:** en ella se van recorriendo todos los componentes del formulario al mismo tiempo que se recuperan sus valores a partir de los parámetros de la petición o request enviada al servidor.
2. **Procesamiento de las validaciones:** consiste en la validación del dato suministrado por el usuario en función de las reglas de validación definidas para el componente.
3. **Proceso de Eventos:** se llevan a cabo todos los eventos asociados a las acciones realizadas por el usuario.
4. **Actualización de los valores del modelo**: tras las validaciones y/o ejecución de los eventos tiene lugar la asignación del 'value' de los componentes a los atributos del bean.
5. **Invocar la aplicación:** aquí se usan los valores del bean para ejecutar la lógica de negocio y se llevan a cabo las operaciones correspondientes a las acciones del usuario.
6. **Presentación de la respuesta**: la implementación de JSF delega su autoridad en el contenedor JSP para enseñar la página que el usuario ve por pantalla, y donde todos los campos están informados con su valor en ese momento.

El ciclo de vida descrito con anterioridad ofrece grandes beneficios para poder tratar de forma dinámica el procesamiento de los datos y manejar eventos que se generan durante el funcionamiento de una aplicación.

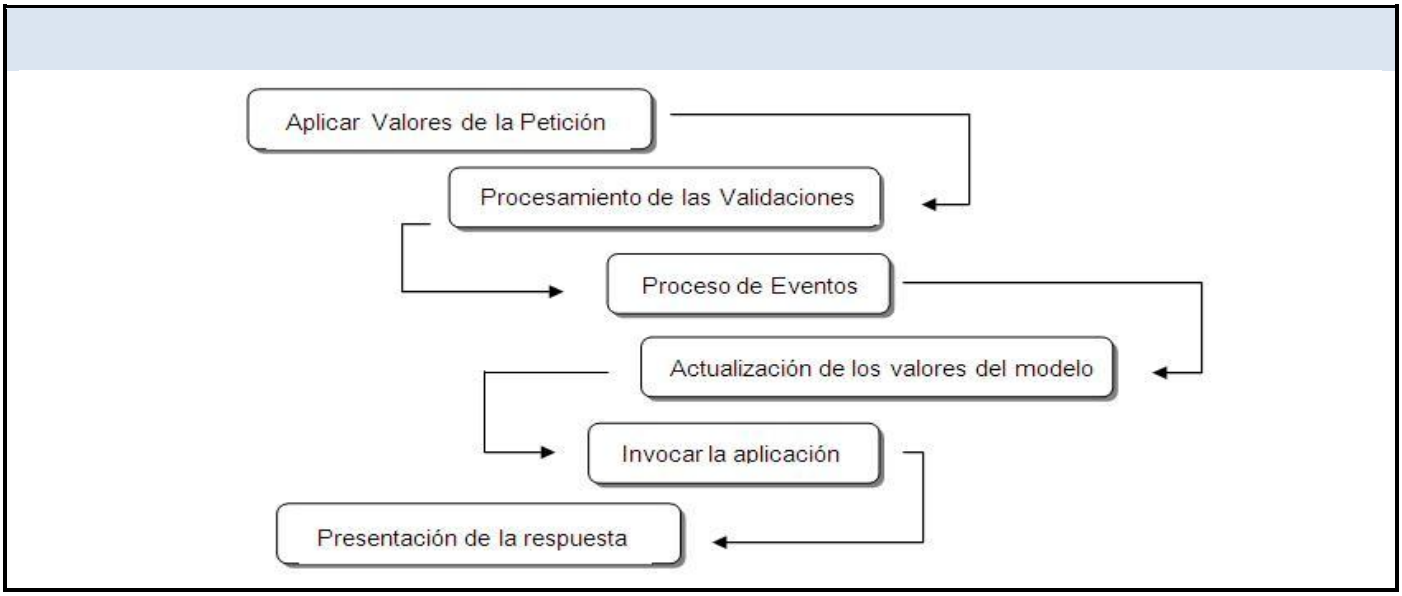

Fig2. 5 Ciclo de Vida de JSF.

2.4.2 RichFaces y Ajax4jsf .Ciclo de vida.

RichFaces y Ajax4jsf son librerías que se integran a JSF para extender las funcionalidades de este mediante Ajax. Incluyen sus propios beneficios en el ciclo de vida, facilidades de validación y el manejo de recursos dinámicos y estáticos. Esta fusión permite definir un evento en una página que invoca una petición Ajax, luego las áreas deben sincronizarse con el árbol de componentes JSF para que esta petición cambie los datos en el servidor.

Cuando el usuario interactúa con la capa de presentación, normalmente crea eventos, los cuales ocasionan la invocación del motor de AJAX del lado del cliente, implementado en Javascript. Un objeto XMLHttpRequest es creado y enviado al servidor. En el lado del servidor, el filtro de RichFaces intercepta la petición y los datos transferidos son extraídos; luego la petición es procesada normalmente por JSF. Cuando el ciclo de vida de JSF ha finalizado y la respuesta está lista, el filtro de RichFaces es invocado nuevamente. Codifica solamente la región de la petición y la envía en formato XML. Al finalizar, el motor de AJAX transforma en el cliente la información del XML en el DOM que será mostrado en la página.

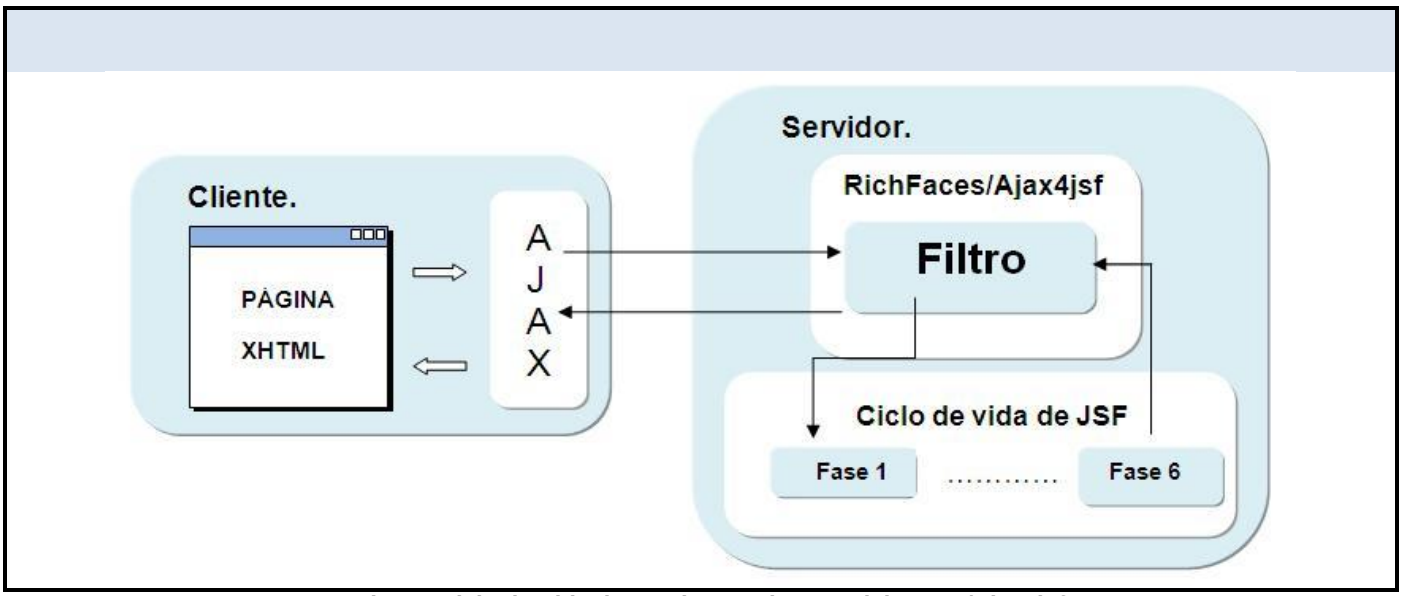

Fig2. 6 Ciclo de Vida de JSF integrado con RichFaces/Ajax4jsf.

2.4.3 Facelets como Motor de Plantillas

Este framework es el motor de plantillas que se integra a la capa de presentación para ingresar en las vistas información dinámica (listas o comportamiento condicional). Los motores de plantillas suelen tener lenguaje Javascript mínimo pues solo se necesita para posibilitar el dinamismo.

Facelets construye un árbol de componentes permitiendo una gran reutilización. Fue concebido teniendo en cuenta el ciclo de vida JSF, por este motivo al usar Facelets sólo existe el servlet de JSF. De esta manera se facilita su integración con JSF. Sustituye el ViewHandler de JSF por Facelets ViewHandler, el cual construye las vistas a partir de documentos XHTML. Esto posibilita un uso óptimo de AJAX ya que una de las tecnologías que la componen es precisamente XHTML.

El Facelets ViewHandler es invocado solo en la primera y la última fase del ciclo de vida de JSF (Aplicar Valores de la Petición y Presentación de la Respuesta*)*. En la fase de Valores de la Petición existen dos posibilidades:

1. La petición que llega al servidor es la primera proveniente de una página particular. En este caso una nueva vista es devuelta a partir del documento XHTML.
2. La página actual ya fue pedida al menos una vez. En este caso la página ya existe en el UIViewRoot, por tanto delega en el ViewHandler de JSF.

2.5 Pautas de Diseño

En este acápite se exponen alguno de los elementos pautados para el diseño de la capa de presentación. De manera general se encuentran en el documento Pautas de Diseño APS-SIAPS.

2.5.1 General

Regla Nº 1: El género de todos los actores que aparezcan en el prototipo será masculino.
Ejemplos: Enfermero, Médico, Ginecólogo.
Regla Nº 2: El formato de la fecha es dd/mm/yyyy (día/mes/año).

Ejemplos: 09/07/2008, 10/01/2007.

Regla Nº 3: El formato en el caso de la hora es hh:mm (hora/minutos) AM/PM.

Ejemplo: 12:00 PM, 02:01 AM, 12:31 AM, 03:01 PM.

Regla Nº 4: En el caso de que se requieran ambos campos se manipularán por separado, es decir, se tratarán como dos atributos diferentes. En ambos casos se utilizará para ello el componente del framework Richfaces rich: calendar (sólo un calendario para los campos).

Regla Nº 5: En el título de los formularios se pondrá el enunciado de la acción que realiza sin usar infinitivos.

Ejemplo: Nuevo capítulo, Nueva enfermedad de declaración obligatoria.

Regla Nº 6: No se podrán utilizar datos pertenecientes a miembros del grupo de desarrollo como juego de datos.

Regla Nº 7: Se utilizará la gama de colores y tonalidades definidas para la marca Alas.

Regla Nº 8: El tipo de letra que se usará en los formularios y sus componentes es Tahoma 11px, color negro (000000).

Fig2. 7 Gama de Colores y Tonalidades.

2.5.2 Cabezal

Para este elemento se debe utilizar el color # 222255. Estará Extendido a todo lo largo de la pantalla, posee 100% de largo, su ancho es de 22 píxeles. En él aparecerán los vínculos a Inicio, Acerca de, Manual Ayuda y Cerrar Sesión. La ubicación de los enlaces será en la zona superior de la pantalla, en la parte derecha del cabezal, el tipo de letra para ellos será Arial 10.

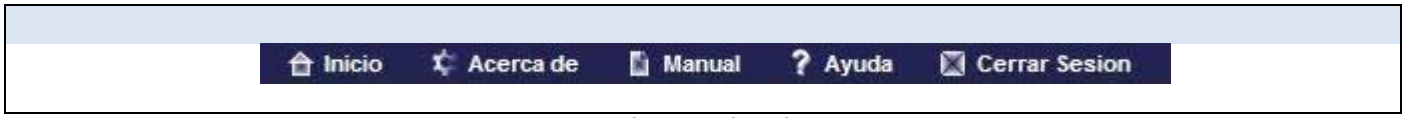

Fig2. 8 Cabezal

2.5.3 Botones

Para los botones se usar para la letra el color #FFFFFF. Su un tamaño será el estándar del IDE. En el fondo debe utilizarse la imagen bg_botton.png. Su estilo es Clase: botón. Irá debajo del formulario y centrados. El nombre de los botones será en infinitivo y describirá la acción que estos representen. Ejemplo: Insertar, Buscar, Editar, etc.

Fig2. 9 Botones

El orden de los botones será de máxima a mínima prioridad, es decir, se ubican de izquierda a derecha. En este caso las acciones positivas (Aceptar, Buscar) van siempre primero que las negativas y las nulas (Eliminar, Cancelar).

2.5.4 Encabezado de las Opciones

Los encabezados van a estar compuestos por el ícono representativo de la acción a realizar, y la acción desarrollada.

Fig2. 10 Ejemplo de un encabezado.

2.5.5 Formularios

Los formularios deben poseer el nombre: “frmsiaps”, lo que posibilitará compartir script y funcionalidades generales para dichos formularios. Sus controles deben seguir las siguientes pautas para su nomenclatura

Control	Prefijo	Ejemplo
Formulario	Frm	frmsiaps
Tabla	Tb	tbListado
Botón	Btn	btnAceptar
Etiqueta (label)	Lbl	lblNombre
Lista/Menú (select)	Mn	mnPrincipal
Campo de Texto (textfield)	Txt	txtFecha
Botón de Opción (radio)	Opt	optSexo
Casilla de Verificación (checkbox)	Chx	chxBorrar
Grid o rejilla	Grid	grUsuario

Fig2. 11 Nombre de Controles.

2.5.6 Opción Listar

Las tablas aparecerán centradas y con los colores definidos en el estilo CCS:

Estilos

```
border-top-color: #789B77;
border-right-color: #789B77;
border-bottom-color: #789B77;
border-left-color: #789B77;
```

El tamaño es variable, depende del largo de la tabla. El nombre de la tabla aparecerá en texto alineado a la izquierda y en negrita. El listado aparecerá por defecto cuando se ejecute la búsqueda. El usuario después podrá buscar información específica seleccionando criterios de búsqueda y presionando el botón buscar.

El nombre de la columna aparecerá alineado a la izquierda y en negro (R0 G0 B0 - #000000). El contenido de la tabla será alineado a la izquierda. Para el paginado de las tablas se propone que sea 1 al 10 elementos por página.

Los botones primero [ ], anterior [ ], siguiente [ ], último [ ] brindan la opción al usuario de buscar el elemento deseado. En el campo de texto el usuario puede poner el número de la página al que desea acceder [ ]. El botón Imprimir permitirá imprimir el resultado en formatos .pdf o .xls según el usuario seleccione. El botón Imprimir tendrá el tamaño estándar del IDE.

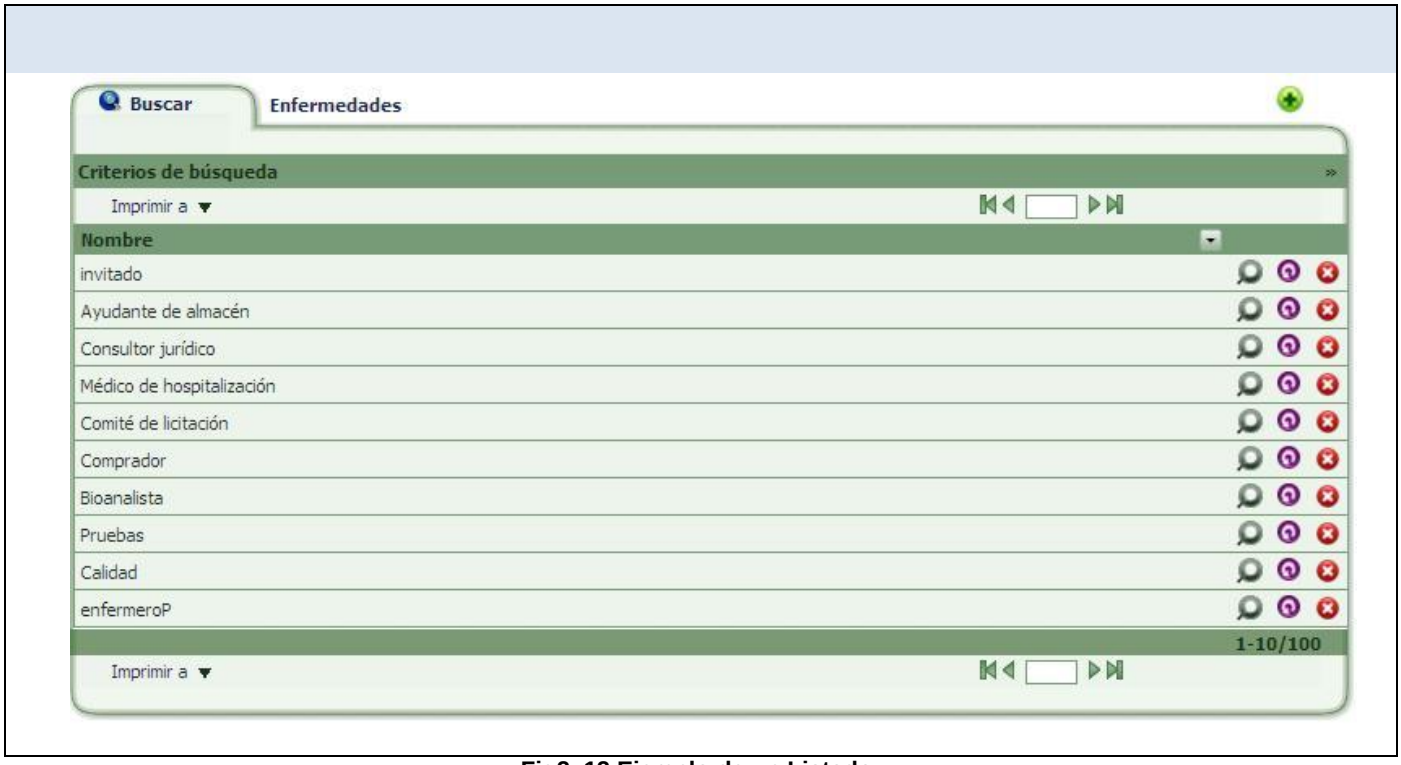

Fig2. 12 Ejemplo de un Listado.

2.6 Propuesta de Plantillas para las Vistas

Para el fondo de la pantalla de autenticación está compuesta por una imagen de líneas blancas y grises: bg_login.gif. En su centro se ubican horizontalmente dos franjas de color verde: #356734, el verde opaco #7F9F7E, azul: #222255 y opaco: #737393.

La figura 2.13 muestra la página se visualizará una vez ingresado al sistema, dará la posibilidad de que el usuario se autentique. El componente de autenticación aparecerá en el centro de la página de autenticación. El componente de autenticación estará compuesto por el identificador visual, dos campos de textos para introducir el usuario y la contraseña, además un botón que permite entrar al sistema.

Fig2. 13 Plantilla de Autenticación.

La figura 2. 14 muestra la página se visualizará una vez el usuario se haya autenticado en el sistema. Ella mostrará los módulos a los que tiene acceso dicho usuario. El nombre del componente siempre se mostrará primero y al lado una imagen que identificativa del proceso que gestiona dicho módulo. Cuando se haga clic encima de un módulo se enlazará al mismo.

Fig2. 15 Plantilla de Acceso a Módulos.

La figura 2.15 se muestra el formato que debe tener la opción de ingresar un nuevo elemento. El icono representativo de la opción agregar es [ ] y su tamaño es de 16 pixeles. El nombre de los campos irá alineado a la izquierda y en la parte derecha irá el cuadro de texto donde se introducirán los datos.

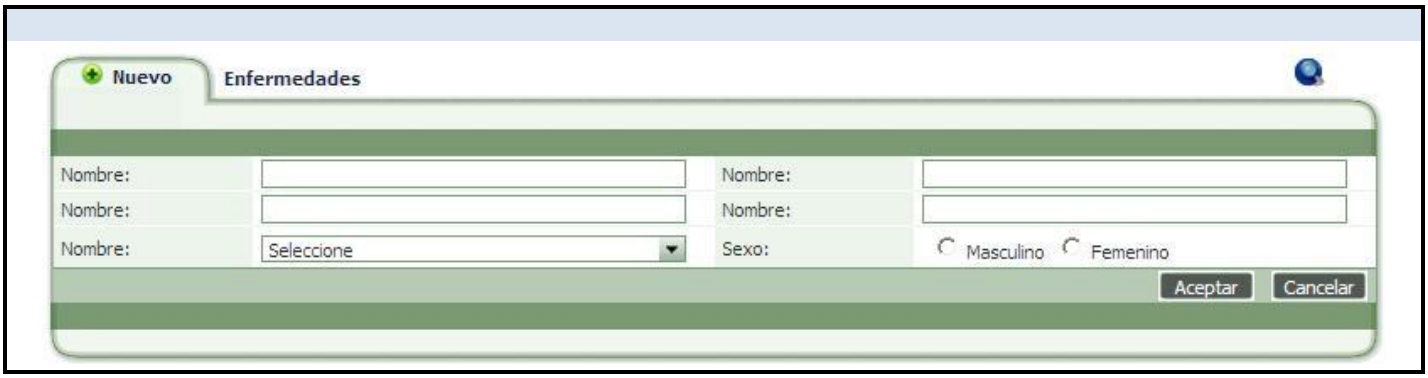

Fig2. 16 Plantilla Opción Nuevo.

Para la opción Actualizar debe usarse el icono representativo de la opción actualizar es [ ] y su tamaño es de 16 pixeles. Cumplirá con el mismo formato que el Agregar, la diferencia radica en que una vez seleccionado un objeto que se quiera actualizar, la página cargará cada uno de los valores introducidos previamente por el usuario posibilitando que este pueda editarlos nuevamente.

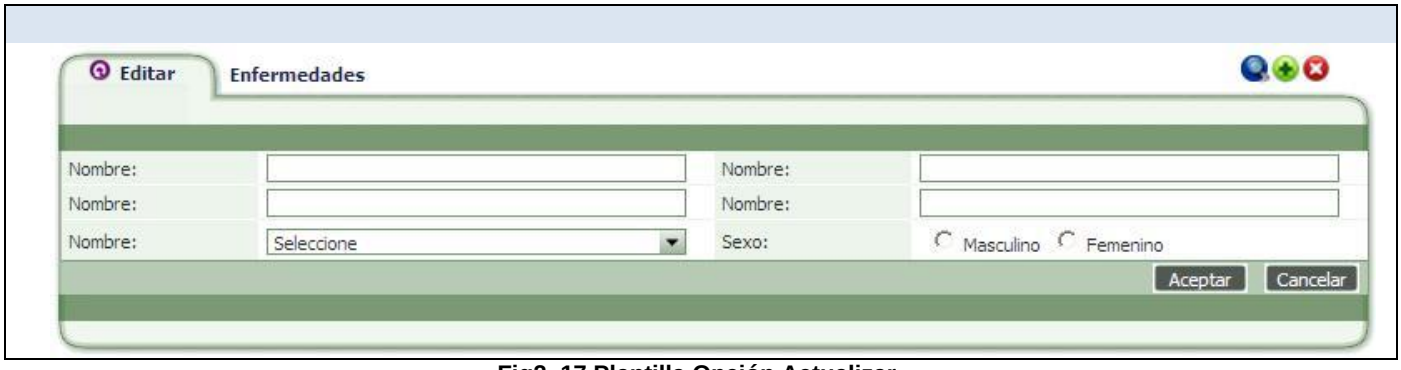

Fig2. 17 Plantilla Opción Actualizar.

La figura 2.17 muestra el formato que debe tener la plantilla para las búsquedas.

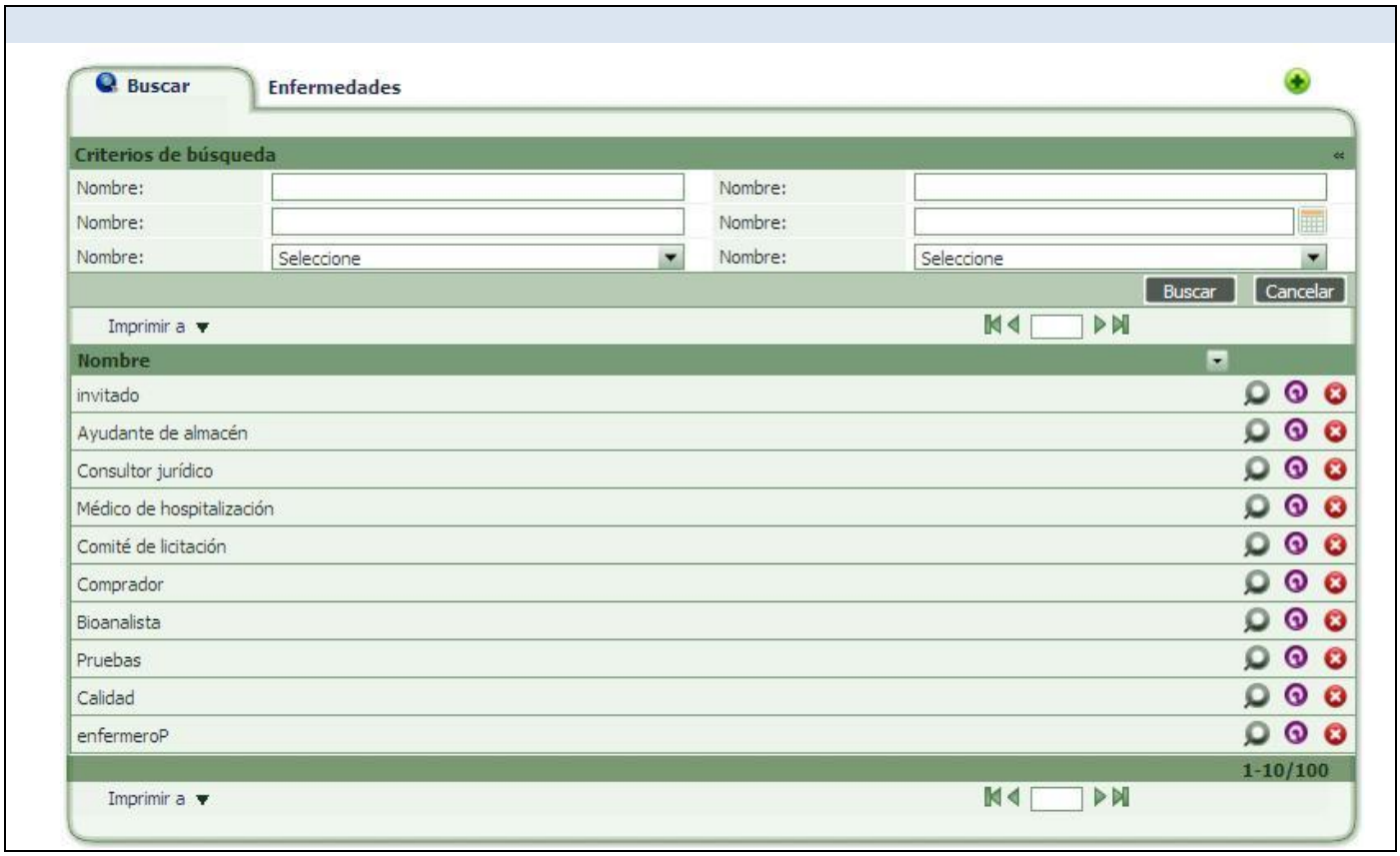

Fig2. 18 Plantilla Opción Buscar.

Esta contendrá las siguientes características:

El nombre de la tabla aparecerá en texto alineado a la izquierda y en negrita. El listado aparecerá por defecto cuando se ejecute la búsqueda. El usuario después podrá buscar información específica seleccionando criterios de búsqueda y presionando el botón buscar. El nombre de la columna aparecerá alineado a la izquierda y en negro (R0 G0 B0 - #000000). Cumpliendo las mismas reglas de capitalización que las etiquetas.

En este capítulo se realizó un análisis de los factores que inciden en el diseño web, así como las consideraciones a tener en cuenta para estimular buenas prácticas para el mismo. Se definieron y pautaron los prototipos de interfaz de usuario propuestos para la solución de las necesidades actuales del sistema.

Capitulo 3. Definición e Integración de la Arquitectura de la Capa de Presentación

Los sistemas de gestión de información se organizan en tres apartados distintos para delimitar responsabilidades entre estos. Con esta separación se decremento la duplicación de código y se centraliza el control. Para obtener este propósito se concretan las capas de Presentación, Negocio y Datos; cada una de ellas debe de poseer una arquitectura bien definida para el correcto funcionamiento de del sistema. En las aplicaciones interactivas Java, el patrón de diseño arquitectural recomen dado para la capa de presentación es el MVC.

Este capítulo tiene como objetivo explicar la integración de los componentes a utilizar en la arquitectura de la capa de presentación del SIAPS, teniendo en cuenta que JSF se basa en este patrón. La meta de su uso es abstraer lo suficiente el desarrollo del sistema, el protocolo http y sus limitaciones.

3.1 Patrón Arquitectónico

MVC es el patrón de diseño arquitectural que ayuda a los desarrolladores a enfocar sus habilidades a la creación de interfaces, ofreciendo una separación entre el comportamiento y la presentación. MVC 2 es una variación de MVC específica para aplicaciones web. Este es el patrón de diseño arquitectural para el desarrollo de la capa de presentación que encaja perfectamente en la tecnología JSF.

Cuando se pone en práctica, se le proporciona a la capa de presentación una definición clara, decrementando de esta manera la duplicación de código, centralizando el control y por tanto se crea una aplicación más extensible.

Como se explicó en el capítulo anterior, JSF es el estándar presentado por Sun para la capa de presentación Web. Forma parte de la especificación J2EE 5 y se erige como una evolución natural de los frameworks actuales hacia un sistema de componentes. Contiene un API para representar componentes de Interfaz de Usuario (UI), manejando estados y eventos de los mismos; así como la validación del lado del servidor, la conversión de datos y una clara definición de la navegación entre páginas. También

contiene dos librerías de etiquetas JSP (Java Server Pages) personalizadas para expresar componentes UI dentro de una página JSP y para conectar componentes a objetos del lado del servidor.

JSF tiene un servlet controlador que se denomina FacesServlet el cual representa el enlace entre el usuario y la aplicación. Es el encargado de recibir las peticiones y ejecutar las acciones definidas en el fichero de configuración faces-config.xml. Las páginas JSF se construyen como un árbol de componentes UI, los que se asocian con un modelo de objetos denominado backing beans. Estos objetos son los encargados de implementar la lógica de negocio, permitiendo mapear parámetros de la petición, sobre atributos de estos objetos y eventos del cliente sobre sus métodos. Las respuestas se construyen cogiendo los componentes de interfaz y transformándolos en un tipo de cliente particular (por ejemplo, un navegador HTML o un navegador VML de un teléfono móvil).

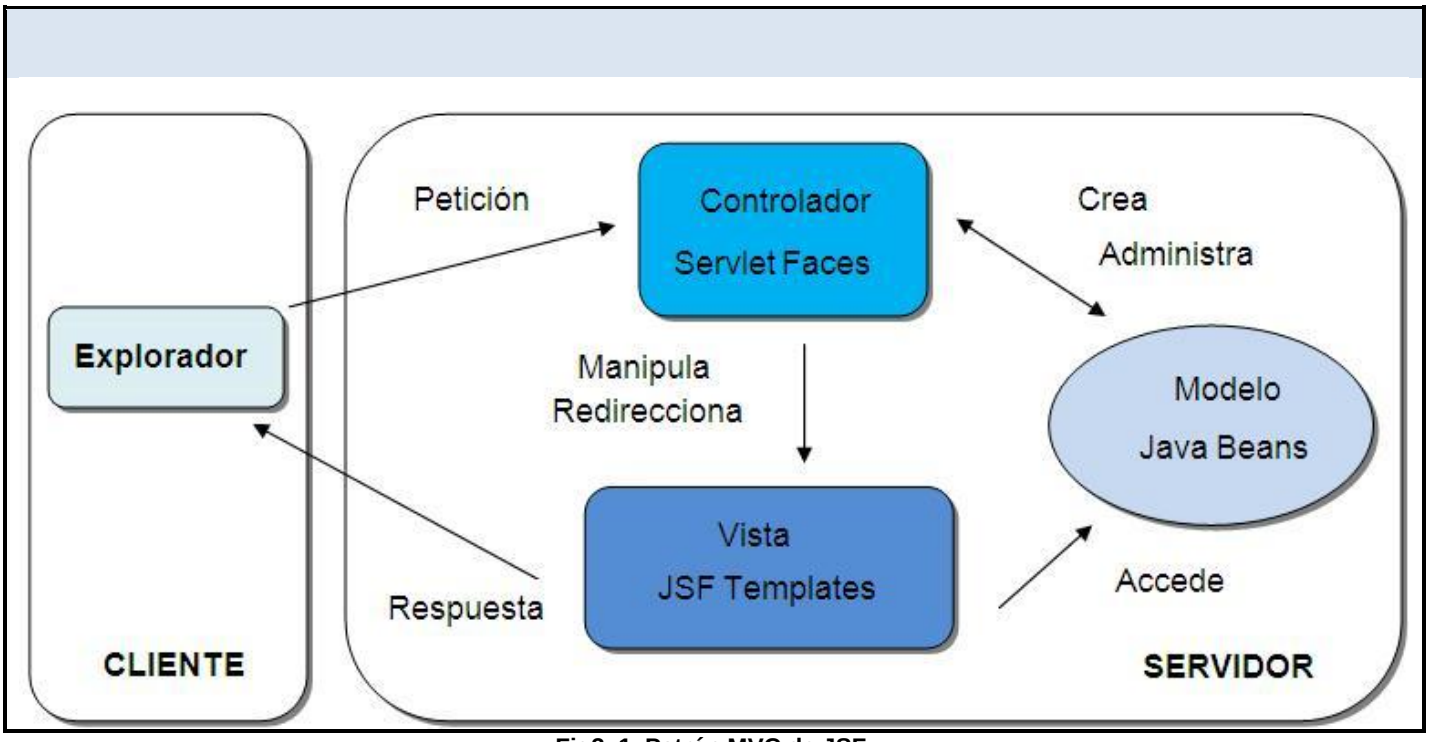

Fig3. 1 Patrón MVC de JSF.

3.2 Propuesta de Capa de Presentación para el SIAPS

Para la creación de la capa de presentación del SIAPS, se tuvieron en cuenta los beneficios que proporciona la utilización de varios componentes pertenecientes a JSF, Facelets y RichFaces/Ajax4jsf. De esta manera se propone el desarrollo de un sistema simplificado de presentación, posibilitando un diseño personalizado de plantillas y dentro de ellas un conjunto de regiones editables.

3.2.1 Integración de RichFaces/Ajax4jsf y Facelets para el Diseño de la Capa de Presentación del SIAPS

La configuración para poder ingresar las ventajas de cada uno de estos frameworks a la aplicación es simple. A continuación se explica cómo se realiza la inclusión de cada uno.

3.2.1.1 RichFaces. Configuración

Para usar RichFaces en la aplicación, hay que descomprimir el fichero "richfaces-ui-3.1.0-bin.zip" en el directorio elegido. Luego se deben incluir dentro del directorio "WEB-INF/lib" los siguientes archivos jar: richfaces-api-3.1.0.jar, richfaces-impl-3.1.0.jar, richfaces-ui-3.1.0.jar.

Para integrarlo a JSF, se debe agregar su filtro al archivo de configuración web.xml antes del Servet de JSF.

Filtro de RichFaces incluido en el fichero WEB-INF/web.xml.

```
<context-param>
<param-name>org.richfaces.SKIN</param-name>
<param-value>blueSky</param-value>
</context-param>
<filter>
<display-name>RichFacesFilter</display-name>
```

```
<filter-name>richfaces</filter-name>
<filter-class>org.ajax4jsf.Filter</filter-class>
</filter>
<filter-mapping>
<filter-name>richfaces</filter-name>
<servlet-name>FacesServlet</servlet-name>
<dispatcher>REQUEST</dispatcher>
<dispatcher>FORWARD</dispatcher>
<dispatcher>INCLUDE</dispatcher>
</filter-mapping>
```

Al utilizar páginas XHTML se debe de incluir estás líneas de código:

Páginas XHTML.

```
<xmlns:a4j="http://richfaces.org/a4j">
<xmlns:rich="http://richfaces.org/rich">
```

3.2.1.2 RichFaces/Ajax4JSF. Etiquetas y Librería de Componentes

3.2.1.2.1 RichFaces

RichFaces combina la funcionalidad de Ajax4JSF en un único paquete e introduce una cantidad importante de nuevos componentes. Mediante él se definen cuáles serán las áreas a actualizar en las páginas. Previamente, Ajax4JSF y RichFaces eran distribuidas de manera separada debido a que Ajax4JSF era open source mientras que RichFaces tenía una licencia comercial. Desde que el proyecto RichFaces se transformó en Open Source, esta distribución separada comenzó a carecer de sentido, por lo que a partir de ahora se distribuye de manera conjunta.

CAPÍTULO 3. DEFINICIÓN E INTEGRACIÓN DE LA ARQUITECTURA DE LA CAPA DE PRESENTACIÓN

RichFaces posee varios componentes que su utilización será de gran utilidad para la aplicación. Para usarse en las vistas, se preceden por el tag **rich** que identifica al framework.

Componentes:

<rich: panelMenu>: permite definir un menú en línea vertical dentro de una página. Posibilita organizar el acceso a todas las vistas de la aplicación pues anteriormente los hipervínculos de la navegación se encontraban distribuidos en cada una de las páginas. Es, por tanto, el componente que facilita la navegabilidad dentro del sitio. Este componente, para su funcionamiento, integra otros que no cumplen ningún objetivo por separado. El componente **<rich: panelMenuGroup>** es utilizado para definir un grupo expandible de opciones dentro del **<rich: panelMenu>** o del mismo. Además, el componente **<rich: panelMenuItem>** es usado para definir una opción simple dentro de una lista desplegable.

<rich: panel>: es un panel personalizable que se dibuja como un rectángulo con bordes definidos y que puede tener encabezado o no. Este componente permite enmarcar el contenido de cada una de las vistas. De esta manera se logran homogenizar todas las páginas de la aplicación haciendo sutil la transición entre ellas.

<rich: calendar>: utilizado para crear elementos de calendario en una página. Sus características principales son:

- Representación en forma de popup.
- Soporte para deshabilitarlo.
- Celdas personalizables.
- Sustituciones grandes basadas en personalización.
- Posicionamiento ágil definido por el usuario.

<rich: message> y <rich: messages>: el componente **<rich: message>** es usado para mostrar un mensaje en un componente específico. Sus principales características son:

- Altamente personalizable.
- Peticiones basadas en AJAX.
- Tooltip para mostrar los detalles de una parte del mensaje.

Su comportamiento es similar al **<h: message>** de la librería HTML de JSF, su diferencia radica en que el componente es actualizado automáticamente después de una petición AJAX, sin necesidad de usar un **<a4j: outputPanel>** y se le puede adicionar un marcador al mensaje. El componente **<rich: messages>** tiene un comportamiento muy parecido al **<rich: message>,** su diferencia radica en que es usado para mostrar todos los mensajes de los componentes de una vista.

Posee el atributo globalOnly que si toma valor verdadero posibilita captar sólo mensajes del contexto de la aplicación y obviar los mensajes de error de los componentes de la vista. La función principal de ambos componentes es que permiten hacer validaciones sencillas del lado del cliente. **<rich: message>** posibilita hacer las validaciones necesarias para cada uno de los componentes correspondientes.

<rich: modalPanel>: implementa una ventana de diálogo. Todas las operaciones de la ventana principal de la aplicación son bloqueadas mientras está activada la ventana. Las operaciones de abrir y cerrar esta ventana son utilizadas a través de código JavaScript en el cliente.

Sus características principales son:

- Altamente personalizable.
- Posibilidad de restaurar el estado previo del componente en la página anterior.
- Posibilita bloquear la ventana principal para informar al usuario sobre el estado de algunos recursos de la aplicación.

RichFaces permite integrar AJAX a los componentes nativos de JSF usando el componente **<a4j: support>.** Este componente capta los eventos lanzados por un componente JSF y envía en peticiones AJAX lo que antes JSF enviaba en peticiones tradicionales, junto con los demás datos de la página. Es el único framework que logra usar todos los componentes de la implementación de JSF en una aplicación con características AJAX. En otros frameworks se debían sustituir los componentes, en este caso se adicionan nuevos y por tanto se adicionan más funcionalidades a JSF.

3.2.1.2.2 Ajax4JSF

Ajax4JSF es una librería open source que se integra totalmente en la arquitectura de JSF. Extiende la funcionalidad de sus etiquetas dotándolas con tecnología Ajax de forma limpia y sin añadir código Javascript.

Presenta mejoras sobre los propios beneficios del framework JSF incluyendo el ciclo de vida, validaciones, facilidades de conversión y el manejo de recursos estáticos y dinámicos. Permite definir un evento en una página que invoca una petición Ajax y luego las áreas de la página deberán sincronizarse con el árbol de componentes de JSF, luego de que la petición Ajax cambie los datos en el servidor.

El funcionamiento del framework es sencillo. Mediante sus propias etiquetas se generan eventos que envían peticiones al contenedor Ajax. Estos eventos se pueden producir por pulsar un botón, un enlace, una región específica de la pantalla, un cambio de estado de un componente, cada cierto tiempo. No hay que preocuparse de crear el código Javascript y el objeto XMLHttpRequest para que envíe la petición al servidor ya que esto lo hará el framework.

Para su uso, los componentes estarán precededidos por el tag **aj4** que identifica al framework:

<aj4: support>: etiqueta que se puede añadir a cualquier otra etiqueta JSF para dotarla de funcionalidad Ajax. Permite al componente generar peticiones asíncronas mediante eventos (onclick, onblur, onchange) y actualizar campos de un formulario de forma independiente, sin recargar toda la página.

<aj4: poll>: realiza cada cierto tiempo una petición al servidor. RichFaces utiliza formularios basados en el envío de peticiones Ajax. Esto significa que cada vez que un usuario hace clic en un botón o en un a4j: poll se produce una petición asíncrona, realizándose un submit con el objeto XMLHTTPRequest. El formulario contiene datos de entrada e información auxiliar sobre el estado.

<aj4: commandButton>: botón de envío de formulario similar a **<h: commandButton>** de JSF. La principal diferencia es que se puede indicar que únicamente actualice ciertos componentes evitando la recarga de todo el formulario.

<aj4: commandLink>: comportamiento similar a **<aj4: commandButton>** pero en un link.

<aj4: htmlCommandLink>: muy parecida a la anterior etiqueta con pequeñas diferencias en la generación de links y cuando se utilizan etiquetas **<f: param>.**

<aj4: region>: determina un área a decodificar en el servidor después de la petición Ajax.

<aj4: status>: muestra el estado de la petición Ajax. Hay 2 estados posibles: procesando petición y petición terminada. Por ejemplo mientras dure el proceso de la llamada al servidor y la evaluación de la petición se puede mostrar el texto "procesando..." y cuando termine la petición y se devuelva la respuesta a la página se cambia el texto por *"*petición finalizada*".*

<aj4: form>: similar al **<h: form>** con la diferencia de que se puede enviar previamente el contenido al contenedor Ajax.

<aj4:actionparam>: etiqueta que combina la funcionalidad de la etiqueta **<f:param>** y **<f:actionListener>.**

<aj4: outputPanel>: se utiliza para agrupar componentes para aplicarles similares propiedades, por ejemplo a la hora de actualizar sus valores tras la petición Ajax.

<aj4: ajaxListener>: similar a la propiedad actionListener o valueChangeListener pero con la diferencia de que la petición se hace al contenedor Ajax.

<aj4: jsFunction>: se utiliza para pasarle un valor automáticamente a una función Javascript tras recibirlo del servidor.

<aj4: loadScript>: inserta en la página las funciones Javascript contenidas en un archivo .js

<aj4: loadStyle>: igual que la anterior etiqueta pero para una hoja de estilos .css

<aj4: log>: carga en la página una consola que muestra las trazas de los logs que devuelve el contenedor Ajax.

<aj4: include>: se utiliza para incluir en la página el contenido de otra de acuerdo a la definición que se haga en las reglas de navegación del faces-config. Es decir, la siguiente página a cargar de acuerdo a la navegación especificada se cargaría en la vista actual.

<aj4: repeat>: etiqueta para iterar sobre una colección y mostrar todos sus campos.

<aj4: keepAlive>: permite mantener un bean en un estado determinado durante peticiciones.

<aj4: mediaOutput>: componente que permite mostrar contenido multimedia como imágenes, vídeos, archivos sonoros, etc.

3.2.2 Facelets. Configuración

Luego de un estudio realizado a los frameworks, se va a utilizar como framework simplificado de presentación a Facelets, pues le brinda a los desarrolladores diseñar de manera libre páginas web y luego asociarle los componentes específicos de JSF.

Se nombra como el framework de plantillas ya que mediante él pueden definirse plantillas para construir un árbol de componentes de forma que se pueden definir los mismos como composición de otros componentes.

Una de las partes que compone una aplicación JSF es el ViewHandler, se trata del patrón decorador que permite a partir de una salida estándar al usuario, modificarla para adaptarla a otros tipos de usuarios, aportando de esta manera la posibilidad de que la aplicación no esté limitada a una salida de tipo HTML sino que pudiese ser en WML o SVG.

Definidas las vistas JSF utilizando plantillas del tipo HTML, se reduce el código innecesario para agregar componentes en la vista .Esta tecnología es independiente del contenedor JSP, lo que significa que una aplicación puede comenzar utilizando las nuevas características de JSF 1.2 sin esperar a que un contenedor tenga soporte para JSP 2.1.

Para agregar las funcionalidades a JSF, hay que tener en cuenta 3 pasos.

1. Incluir el fichero jsf-facelets.jar como librería de la aplicación en WEB-INF\lib.
2. Realizar la configuración en el fichero faces-config.xml para manejar las vistas.
3. Indicar en el fichero web.xml que las páginas son XHTML.

Paso #2. Inclusión de Facelets en el fichero faces-config.xml.

```
<faces-config>
<application>
 …..
        <view-handler>com.sun.facelets.FaceletsViewHandler</view-handel>
 ….
  </application>
<faces-config>
```

El primer parámetro le dice a Facelets qué sufijo tendrán las páginas, y el segundo va a permitir ver el debug en la consola. Agregar estas líneas al web.xml:

Paso #3. Configuración de las páginas XHTML en el fichero web.xml.

```
<context-param>
<param-name>javax.faces.DEFAULT_SUFFIX</param-name>
<param-value>.xhtml</param-value>
</context-param>
<context-param>
<param-name>facelets. DEVELOPMENT</param-name>
<param-value>true</param-value>
</context-param>
```

3.2.2.1 Facelets. Etiquetas

Proporciona una librería de tag para la construcción de las vistas, entre las cuales se encuentran:

- **<ui: insert>:** declara las partes del documento que serán sobrescritas.
- **<ui: composition>:** solo lo que este dentro podrá ser creado o editado con un editor visual.
- **<ui: define>:** especifica la parte de la página que será creada. Evita la duplicación de código.
- **<ui: include>:** incluye en la página cualquier tag composition o component.

Dentro del código XHTML de la plantilla, usando el tag **<ui: insert>** se definen las regiones lógicas, en las cuales se incluirán todos los contenidos definidos en las páginas de la aplicación. De esta forma se pueden agregar tantas regiones editables como se necesite. Esto permite que en las nuevas páginas sólo se encuentre el conjunto de componentes necesarios en ellas, sin necesidad de repetir en cada una todo el código HTML de la plantilla.

Facelets permite definir la plantilla general para la aplicación y dentro de ella un conjunto de regiones editables.

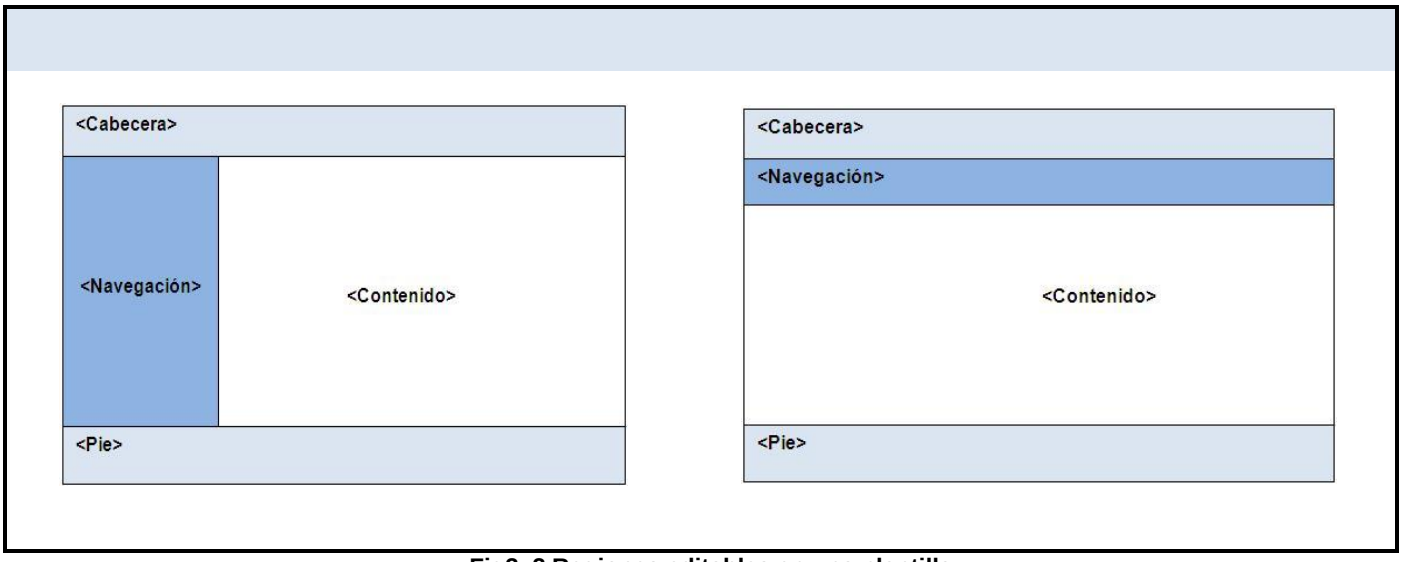

Fig3. 2 Regiones editables en una plantilla.

Para crear un layout o plantilla se tienen que seguir 3 pasos:

1. Crear la plantilla plantilla.xhtml donde se indicarán todos los divs con un identificador que corresponda al definido en la hoja de estilo.
2. Crear las páginas xhtml por cada parte de la estructura, generalmente con mayor contenido estático son el encabezado y el pie.
3. Crear las páginas xhtml con mayor contenido dinámico, las listas y las formas. Basta con indicar cuál es la plantilla a seguir y "sobrescribiendo" la parte que se definió en la plantilla con un <ui: insert name="contenido"> y un <ui: define name="contenido">.

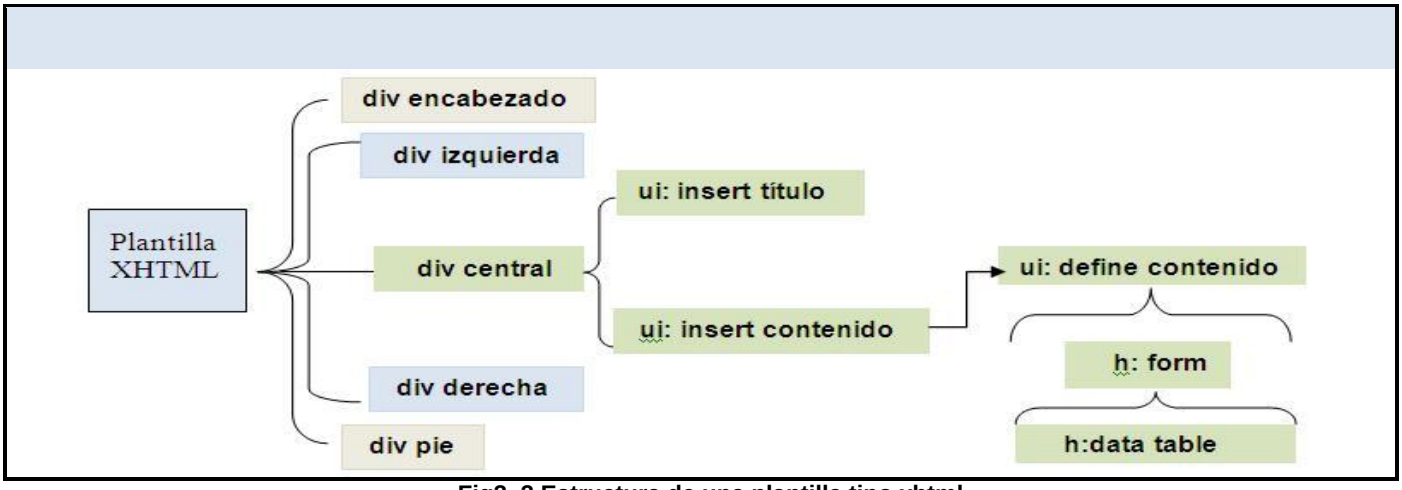

Fig3. 3 Estructura de una plantilla tipo xhtml.

Cuando estos frameworks se fusionan, es posible agregar funcionalidades Ajax al sistema, de forma tal que cuando el usuario final interactúe con la capa de presentación del SIAPS, se cree una invocación al motor de AJAX del lado del cliente. Luego será creado y enviado al servidor un objeto XMLHttpRequest, el cual será interceptado por el filtro de RichFaces.

Este filtro invoca el componente AjaxViewRoot, mediante el cual se describe el área de la página que debe decodificarse durante la solicitud AJAX, luego se le transfieren los datos al servlet de JSF donde se ejecuta su ciclo de vida, procesando de este modo la petición. Terminado este ciclo, la respuesta estará lista en formato XML, para renderizar este formato se utiliza la colección AjaxRenderKit, encargada de

llamar a los componentes UI implicados en el flujo de salida y asociados con la respuesta creada para esa solicitud.

Nuevamente se invoca el filtro de RichFaces para codificar solo la región donde se mostrará la respuesta de la petición. Al finalizar, el motor de AJAX transforma en el cliente la información del XML en el DOM que será mostrado en la página. Esta página es una plantilla creada por componentes del framework Facelets, proporcionándole de este modo regiones editables posibles de actualizar.

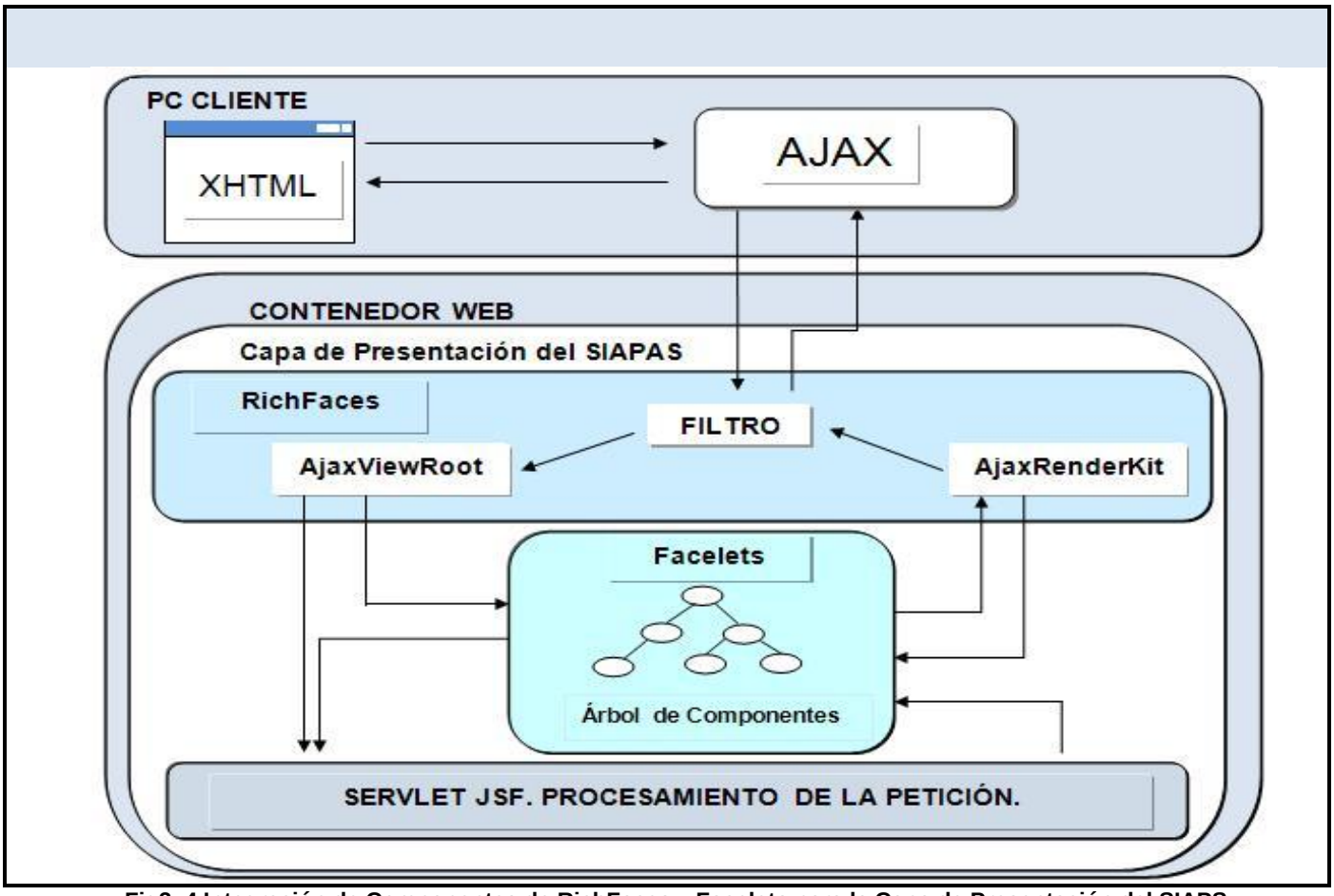

Fig3. 4 Integración de Componentes de RichFaces y Facelets para la Capa de Presentación del SIAPS.

3.3 Propuesta de Integración de la Capa de Presentación con la Capa de Negocio

Como se explicó anteriormente, la capa de presentación del SIAPS contendrá como framework base a JSF, para optimizar sus vistas se hará uso del framework de plantillas Facelets y para las actualizaciones con AJAX los framework RichFaces/Ajax4jsf. Es de vital importancia que la integración con las demás capas sea de forma exquisita para su total funcionamiento.

Para la integración de ambas capas, se propone utilizar JBoss Seam, pues integra y unifica los distintos Standards de la plataforma Java EE 5.0, pudiendo trabajar con todos ellos siguiendo el mismo modelo de programación. El núcleo principal de Seam está formado por las especificaciones Enterprise JavaBeans 3 (EJB3) y JSF. A grandes rasgos EJB3 es una arquitectura para un sistema (como bases de datos) de objetos distribuidos basado en componentes que permite construir aplicaciones portables, reusables y escalables.

Este framework sigue el modelo MVC Pull para la integración entre estas capas, permitiendo que desde una página puedan referenciarse cualquier componente que esté asociado a un contexto accesible. La estrategia que sigue este modelo consiste en que sea el cliente el encargado de conectarse con el servidor, logrando que una vez hecha la conexión la información se actualice automáticamente. Trae como ventaja la liberación de carga al servidor y permite el acceso a un mayor número de usuarios simultáneamente.

En esta variante del MVC, el modelo representa los datos y las reglas que gobiernan el acceso y actualización de los mismos, representando una aproximación de software a un proceso del mundo real. La vista muestra el contenido del modelo y especifica exactamente cómo los datos de este deben ser presentados. Si los datos del modelo cambian, la vista debe actualizar su representación. En este caso particular, donde se sigue el modelo Pull, la vista es responsable de llamar al modelo cuando necesita obtener los datos actualizados.

El controlador se encarga traducir las interacciones del usuario con la vista en acciones que el modelo puede realizar. El uso de este modelo posibilita una buena definición de la arquitectura de información, así

como la creación las interfaces del SIAPS, pues la vista se responsabiliza de llamar al modelo cuando se necesita recuperar datos y que estos sean mostrados.

3.4 Despliegue

Para el correcto despliegue de la capa de presentación, es necesario crear un empaquetamiento adecuado de todos los ficheros implicados en la aplicación y crear un descriptor de despliegue.

La capa de presentación deberá empaquetarse con un formato de archivo WAR. En este WAR deberá haber además como mínimo el siguiente contenido:

- Un descriptor, llamado web.xml, que configura los recursos necesarios en la aplicación.
- Ficheros JAR que contengan las clases esenciales y necesarias.
- Clases de la aplicación, páginas JSF y otros recursos como imágenes.
- Fichero de configuración de la aplicación (faces-config.xml).

Un descriptor de despliegue es un documento XML con una extensión .xml que describe las características de despliegue de una aplicación, un módulo o un componente. Debido a que la información del descriptor de despliegue es declarativa, ella puede ser modificada sin necesidad de cambiar el código fuente. En tiempo de ejecución el servidor de aplicación lee el descriptor de despliegue y se comporta de acuerdo con el componente, módulo o aplicación.

3.5 Estructura del Directorio de la Capa de Presentación

Teniendo en cuenta la definición e integración antes explicada, para crear el diseño de la capa de presentación se seguirá la siguiente jerarquía de carpetas en el servidor para su organización:

img: carpeta que contiene las imágenes generales de la aplicación: imágenes de las plantillas, banner, etc.
layout: carpeta donde se ubican las plantillas utilizadas para la aplicación.

stylesheet: contiene los estilos generales para los componentes del sistema. Con el objetivo de mejorar el rendimiento de la aplicación se usarán estilos de componentes específicos que trae por defecto Jboss Seam.

WEB-INF: esta carpeta contiene las configuraciones necesarias para la navegación, el uso de filtros y servlet.

modCommon: en esta carpeta se ubican los archivos generales del sistema. Un ejemplo es la página de inicio y la de autenticación de usuarios.

modConfiguración: en esta carpeta se ubican los archivos de configuración comunes para los módulos de la aplicación.

Mod_CIE: cada uno de los módulos que integran al sistema serán colocados en carpetas diferentes en el directorio WebContent como se observa en la figura. El nombre de cada carpeta cumplirá con la regla de escribir Mod + Acrónimo del Módulo.

.svn: carpeta creada para guardar la información en el subvertion.

funSelector: carpeta con el propósito de agrupar funcionalidades.

listarCapítulo: carpeta donde se almacenan las plantillas para las vistas de esa funcionalidad.

listarGrupos: carpeta donde se almacenan las plantillas para las vistas de esa funcionalidad.

Un módulo contendrá tres tipos de carpetas básicamente, una para sus imágenes, otra para los ficheros de configuración (generales) y un número mayor para agrupar funcionalidades afines.

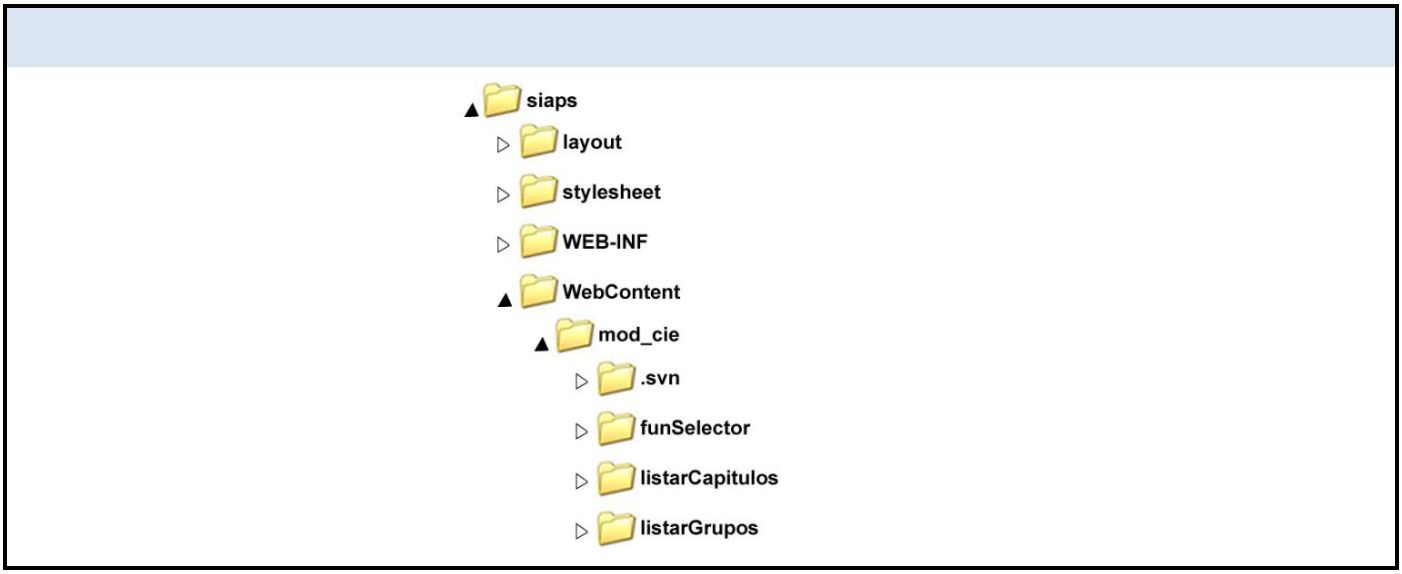

Fig3. 5 Jerarquía de Carpetas de la Capa de Presentación del SIAPS.

Como el SIAPS utiliza el framework JSF en el diseño de la capa de presentación, esta configuración es similar a todas las aplicaciones JSF:

El primer paso es definir el fichero de configuración JSF en el WEB-INF:

```
<context-param>
  <param-name>javax.faces.CONFIG_FILES</param-name>
  <param-value>/WEB-INF/faces-config.xml</param-value>
</context-param>
```

Luego se define clase servlet que se encargará de arrancar todo el framework JSF:

```
<servlet>
  <servlet-name>Faces Servlet</servlet-name>
  <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
  <load-on-startup>0</load-on-startup>
</servlet>
```

El último paso es definir el patrón URL que lanzará la aplicación:

```
<servlet-mapping>
  <servlet-name>Faces Servlet</servlet-name>
  <url-pattern>*.faces</url-pattern>
</servlet-mapping>
```

3.6 Reglas de Navegación

Cuando se produce una petición que incluye una acción se ejecuta el mecanismo de navegación de JSF. Tras la ejecución de la acción, el controlador determina cómo se debe mostrar al usuario el resultado de la petición. La navegación proporciona la habilidad de moverse de una página a la siguiente.

Las reglas de navegación de una aplicación JSF se configuran dentro del fichero de configuración de la aplicación (faces-config.xml), mediante los elementos XML navigation-rule. Es aquí donde se define cuál es la siguiente página que se mostrará después de procesar la orden del usuario.

Debido a la utilización de RichFaces y Facelets no es necesario usar la misma regla de navegación siempre para ir de una página a otra. El componente de RichFaces <a4j: include> es el que permite organizar el flujo de la página, el cual es definido en el fichero faces-navigation-rules.xml de la aplicación.

Este componente se incluye dentro del tag <ui: define> con el propósito de poder asignarle la nueva vista contenida dentro del <a4j: include> al <ui: define>, para mediante la entrega de la vista al tag <ui: insert> pueda ser incorporada en la plantilla de la página. De esta forma, el tránsito de una página a otra de la aplicación se hace siguiendo este flujo, cambiando solamente la región definida por el valor del <a4j: include>.

Con la utilización de RichFaces se hace uso del atributo reRender a todos los componentes para la optimización de las peticiones AJAX. La importancia de este atributo radica en que permite definir las áreas de la página que serán actualizadas como resultado de la interacción con la respuesta AJAX. Su valor es un id o una lista de id de los componentes del DOM.

Para encontrar los componentes en el árbol de componentes, él utiliza el algoritmo UIComponent.findComponent () definido en el API de JSF 1.2, por tanto se puede definir lo rápido que será encontrado el componente en cuestión si se le menciona con más precisión en qué parte del árbol está.

3.7 Propuesta de Seguridad en la Capa de Presentación

Una característica necesaria para cualquier aplicación web es su seguridad. Cualquier sistema que requiera algún tipo de servicio personal debe conocer la identidad del usuario. Este conocimiento, así como los derechos del mismo, son datos imprescindibles para aportarle a la aplicación este aspecto tan importante.

Para proveer un buen mecanismo de autenticación y autorización muchos sistemas utilizan el método de JAAS (Java Authentication and Authorization Service) para proporcionar tales servicios. Teniendo en cuenta las necesidades y controles de seguridad que requiere el SIAPS se propone utilizar para la capa de presentación el API de JBoss Seam basada en AAS. Este sistema se encarga del manejo de errores de autorización o autenticación permitiendo redirigir al usuario a una página determinada.

Este API sirve de ayuda en la inyección de la funcionalidad de acceso a la aplicación, contribuyendo con la capacidad de exigir la autenticación y comprobación de derechos para acceder a información particular. Es necesario para la autenticación de usuarios pues así se comprueban los derechos de acceso.

Utilizándola se puede autorizar a los usuarios en los distintos niveles de la aplicación, pudiendo especificar los límites de acceso para los distintos módulos de la misma. Estos derechos de acceso se especifican usando funciones o permisos. Al igual que los servicios de autenticación dentro de Seam, puede especificar las normas de acceso de entidades que utilizan el archivo de configuración de entradas y código de anotaciones.

En este capítulo se definieron los elementos que propician la integración de AJAX a JSF para nutrir de asincronismo a la capa de presentación y se ha explicado técnicamente este proceso. Se hace una propuesta de cómo puede ser la integración de esta capa con la de negocio, así como el mecanismo que aportará seguridad a la aplicación.

Capítulo 4. Estudio de Factibilidad

El análisis de factibilidad se completa en la fase de diseño de sistemas y los mismos consideran la factibilidad técnica, económica y operacional; por tal motivo es indispensable el conocimiento de las posibilidades de éxito que dicha propuesta proporciona.

En el presente capítulo se pretende exponer la factibilidad de fusionar AJAX y JSF en el desarrollo del diseño de interfaces web. La demostración de tal idea se justifica con las ventajas que esta unificación proporciona en la implementación de una capa de presentación dinámica y con mayor sencillez.

4.1 Factibilidad Técnica

La construcción de las interfaces de usuario suele ser la parte más costosa del esfuerzo de desarrollo, sobre todo por la dificultad de mantenimiento. La tecnología para la solución propuesta está actualmente disponible, los framework utilizados son JSF soportado con AJAX mediante el uso de RichFaces y Facelets para crear un sistema simplificado de plantillas. La ventaja más importante que ofrece JSF es la separación de conceptos, es decir, ofrece al diseño de la capa de presentación del SIAPS una clara separación entre el comportamiento y la presentación.

Otra ventaja de JSF se relaciona con los componentes de interfaz de usuario y la capa web, la cual no se limita a una tecnología de código en particular o un lenguaje de marcas. Aunque la tecnología JSF incluye una librería de etiquetas personalizadas para representar componentes en una página JSP, el API de la tecnología JSF se ha creado directamente sobre el API Servlet.

Esto permite usar otra tecnología de presentación con o sin JSP, crear componentes personalizados directamente desde las clases de componentes y generar salida para diferentes dispositivos cliente. Lo más importante de la tecnología JSF es que proporciona una rica arquitectura para manejar el estado de los componentes, procesar los datos, validar la entrada del usuario y manejar eventos.

Los componentes de la interfaz de usuario de RichFaces vienen preparados para su uso fuera del paquete, así los desarrolladores ahorrarán tiempo y podrán disponer de las ventajas mencionadas para la creación del diseño. Como resultado, la experiencia puede ser más rápida y fácil de obtener. Permite definir (por medio de etiquetas de JSF) diferentes partes de una página JSF que se desee actualizar con una solicitud Ajax, proporcionando así varias opciones para enviar peticiones al servidor.

Se consideró el uso de del framework Facelets debido a que es un sistema simplificado de plantilla para crear un árbol de componentes. Esto permite una gran reutilización, de modo que se pueden definir componentes como composición de otros. Mediante él es posible definir las regiones lógicas y editables en cada una de las vistas. De esta manera sólo tendrán el conjunto de componentes necesarios sin necesidad de repetir en cada una todo el código HTML de la plantilla. Así se obtiene un código ordenado, fácil de desarrollar y rehusar. Fue creado tiendo en cuenta el ciclo de vida de JSF. El aumento en el rendimiento de la aplicación se hace perceptible al usuario final cuando se fusionan tales tecnologías.

4.2 Factibilidad Operacional

Para determinar si la nueva propuesta es factiblemente operativa, se hizo una comparación entre las características que difieren de un diseño síncrono como el del SISalud y el asíncrono que se propone para el SIAPS.

El proceso petición-respuesta entre el cliente y el servidor que ofrece el SISalud posee la naturaleza síncrona propia de la interacción del binomio HTML-HTTP. Dicha interacción obliga al cliente a esperar en cada petición mediante la que se transmiten los datos hacia el servidor mediante este protocolo, el procesamiento de los mismos y la transferencia de la respuesta en forma de una página HTML devuelta hacia él. A continuación el navegador del cliente debe reemplazar la página web original con aquella recibida permitiendo que continúe la interacción del usuario con el sistema.

Este mecanismo implica frágiles conexiones entre quienes envían y reciben datos, volviéndose difíciles de mantener a medida que su número incrementa. Habitualmente en las peticiones sincrónicas al servidor, la página que el usuario está viendo se recarga totalmente, aportando lentitud y menos interacción al

sistema. Estos puntos débiles traen consigo problemas de escalabilidad, eficiencia y usabilidad del SISalud, pues la presentación de la información entregada a los usuarios representa una "foto" de un momento específico del sistema, persistiendo en él el inconveniente de generar vistas potencialmente obsoletas.

El modelo asíncrono es radical, es decir, al realizar peticiones de forma asincrónica el navegador Web no bloquea a la aplicación que está corriendo. Esto se debe a que el navegador no debe esperar por la finalización de la ejecución de la petición en el servidor. En la web asíncrona es posible entregar al usuario cambios espontáneos en la presentación, a medida que cambia el estado de un sistema dinámico.

Para lograr la realización de peticiones asíncronas sin que importen las limitantes del protocolo HTTP, es necesario manipular el mecanismo petición/respuesta para lograr el efecto de poder enviar respuestas al navegador. Esta es la necesidad básica que impulsó la idea de realizar dicha operación utilizando AJAX en el SIAPS.

La propuesta de incluir AJAX en la capa de presentación permite a las páginas hacer una pequeña petición de datos al servidor y recibirla sin necesidad de cargarla completamente. Con este proceso el usuario tiene la percepción de que todo funciona con mayor rapidez, lo cual se debe a que la interacción con el sistema se vuelve parecida a la forma con la que interactúa con las aplicaciones de escritorio tales como el Word, Excel, Outlook o similares.

Las ventajas que le proporciona al SIAPS esta característica son obvias, ya que va a ser posible mantener una representación exacta del sistema en el navegador del usuario. Su uso reduce significativamente los tiempos de respuesta, el ancho de banda consumido por la aplicación es menor y por ende el rendimiento y la velocidad serán incrementados; contribuyendo a un menor consumo de recursos tanto del lado del cliente como del servidor.

4.3 Factibilidad Económica

Luego de haber analizado técnico y operacionalmente la propuesta, la misma debe ser factible desde el punto de vista económico. Con tal objetivo se investigó acerca del costo del uso de estas tecnologías y si son justificados por los beneficios que ofrecerá.

El uso de estas tecnologías es económicamente factible pues todas están bajo la licencia Pública General (inglés: General Public License o GPL), por esta razón es posible su disponibilidad, accesibilidad y su gratuidad para todo desarrollador que necesite de su utilización. Esta licencia además contribuye al mejoramiento y evolución del software, lo que permite la expansión del conocimiento depositado en cada pieza del mismo.

Con la inclusión de AJAX en la capa de presentación del SIAPS es posible disminuir el coste de creación, facilidad de soporte y mantenimiento, menores tiempos a la hora de desarrollar el sistema y se consigue un mejor resultado de cara al usuario final. Con el uso de los framework se aceleró el proceso de desarrollo pues les fue posible a los diseñadores de interfaces la reutilización y personalización de componentes ya existentes, promoviendo de esta forma buenas prácticas de diseño.

En este capítulo se ha fundamentado la factibilidad del uso de AJAX en la creación del diseño web del SIAPS. Para ello se han evaluado las ventajas que suponen el uso técnico, operacional y económico de las librerías y framework que intervienen en la correcta utilización de este. También se ha tenido en cuenta que en la actualidad no existe un sistema de gestión para la atención primaria con tales características, por lo que la propuesta es viable para los estudios realizados.

Capítulo 5. Caso de Estudio

En los capítulos anteriores se definió la Arquitectura de Información y las Pautas de Diseño, por lo que todo ello se tiene en cuenta para el diseño de las pantallas del codificador CIE-10. Se tienen en cuenta las buenas prácticas que permiten ponderar los factores que contribuyen a la visualización rápida, cómoda y efectividad de las páginas. A continuación se presentan:

5. 1 Diseño de las pantallas Nuevo, Editar y Buscar de CIE-10

Para ingresar un nuevo capítulo el usuario podrá hacerlo mediante esta pantalla, la cual posee un menú de rastros en su parte inferior izquierda, así como cajas de textos ubicados en la parte derecha del nombre del elemento al cual se le desea ingresar los datos. En la parte inferior derecha se encuentran los botones Aceptar y Cancelar que le posibilitan ingresar el nuevo elemento o cancelar la acción.

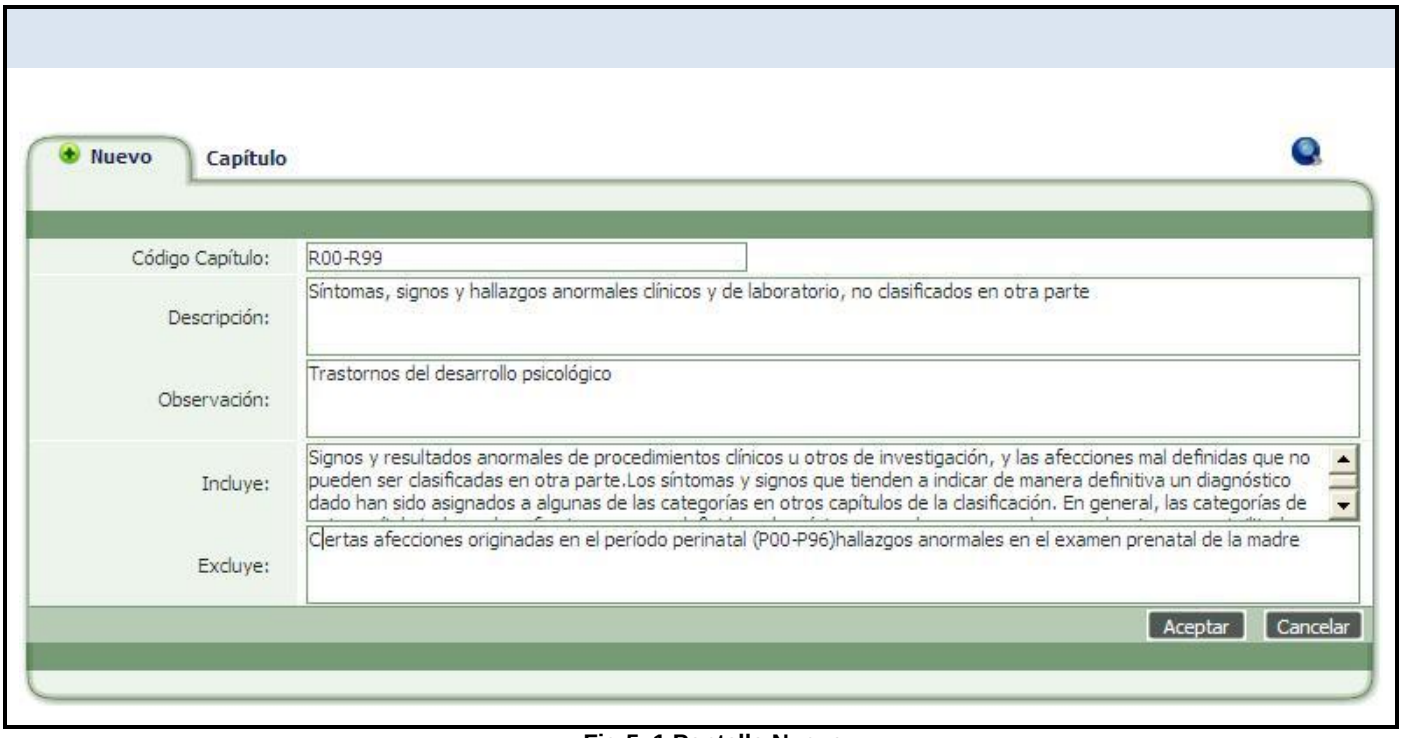

Fig.5. 1 Pantalla Nuevo

Mediante esta pantalla se le da la opción al usuario editar un capítulo, posee las particularidades definidas en las pautas de diseño. Los íconos que se encuentran en la parte superior derecha dan la posibilidad de que el usuario llegue a las páginas cuyas opciones son de búsqueda, agregar y eliminar con un solo clic,

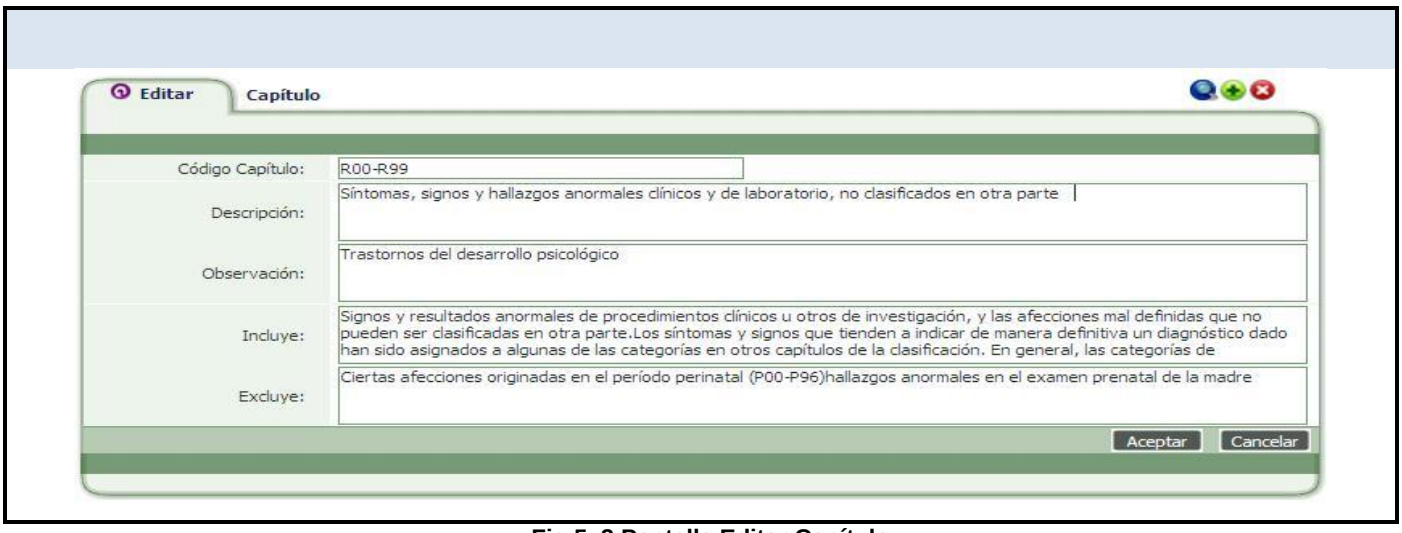

Fig.5. 2 Pantalla Editar Capítulo.

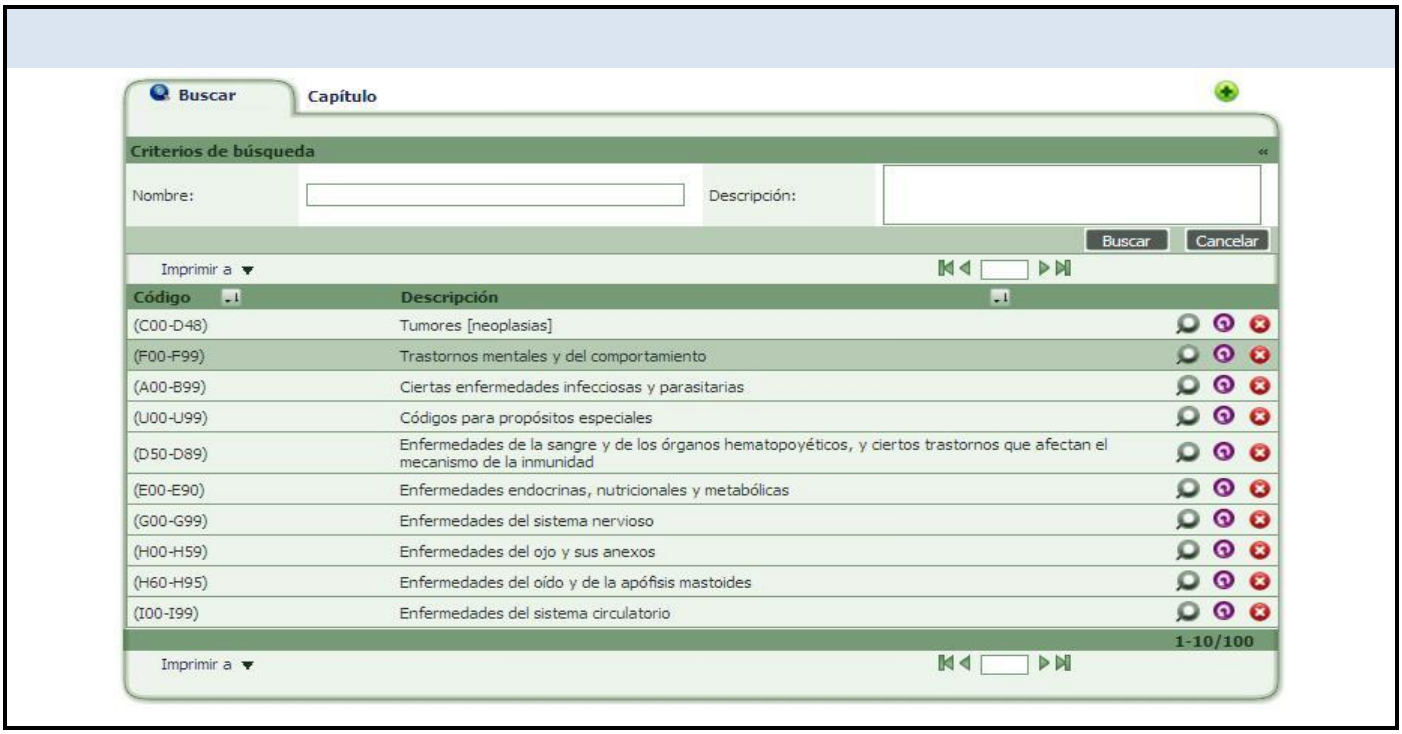

Fig.5. 3 Buscar Capítulo.

La pantalla anterior muestra un listado de capítulos, a los cuales se les puede ver sus descripciones dando clic en el ícono que representa la búsqueda, al pinchar en el ícono representativo de la opción Editar se da acceso a la pantalla que permite llevar a cabo esta acción. En caso de que el usuario desee eliminar uno de estos, dando clic en el ícono último ícono le saldrá un mensaje de confirmación de la acción como se muestra en la siguiente figura.

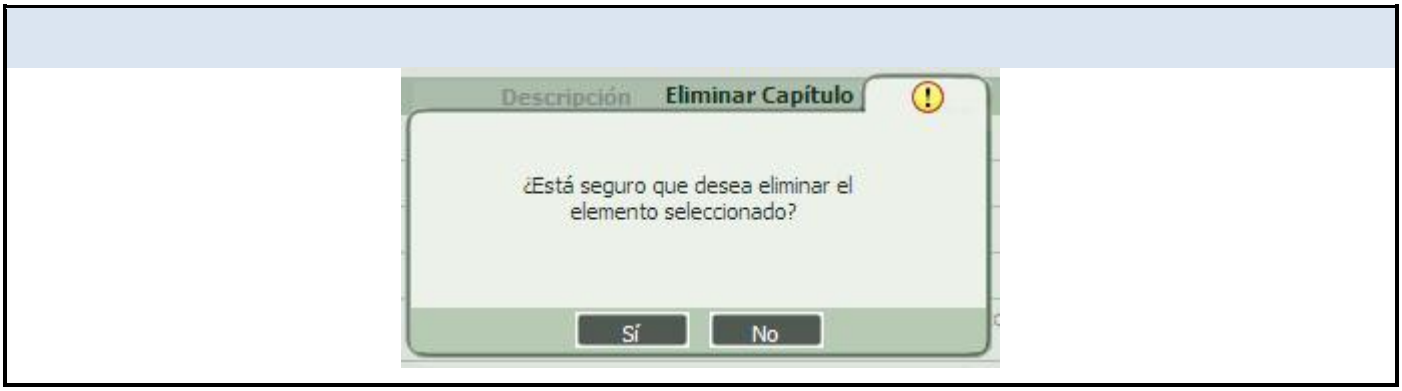

Fig.5. 4 Mensaje de Confirmación Eliminar Capítulo.

5.2 Buenas Prácticas

- Para evitar incomodidad los ficheros no pueden tener más de 2000 líneas de código.
- Se evadieron las líneas de más de 80 caracteres, ya que no son bien manejadas por muchas terminales y herramientas. Ejemplos: para uso en la documentación deben tener una longitud inferior, generalmente no más de 70 caracteres.
- Con la finalidad de ampliar el área de trabajo se utilizó el componente rich: effectt, el que permite que el menú principal y el cabezal puedan ocultarse.
- El menú principal, las tablas, los botones e íconos tienen bordes redondeados para que el diseño de la aplicación tenga correspondencia con la línea de identidad visual de alas.
- Los colores dominantes en la aplicación son el verde, el azul y el blanco para mantener los colores del identificador visual de alas.

- Se puso en práctica los principios de la Ley de Fitt para proporcionarle expresividad a los íconos, incrementando su tamaño y posicionándolos en la región superior derecha de las tablas, con lo cual se consigue que el usuario tenga acceso con un solo clic a la funcionalidad deseada.
- Se definieron clases para los diferentes tipos de input text para lograr homogeneidad en el diseño.
- Para minimizar el tamaño de los íconos e imágenes se utilizó el formato .gif y .pnp, con ello se contribuye a que no se torne lento el tiempo de carga de la aplicación.
- Los textos tienen el mismo tipo de letra (Tahoma), mismo color para fondos claros (#1A331A) y para fondos oscuros (#FFFFFF), así como el mismo tamaño (11px).
- Al posicionarse sobre una fila de un listado o del menú principal, cambia de color el fondo para realzar la visibilidad y la importancia de la información que pueda ofrecer la opción.
- En los listados las opciones de paginado e imprimir aparecen en la parte superior e inferior del listado para que sea de mejor acceso por parte del usuario.
- El paginado cuenta con un input text mediante el cual el usuario puede ir de manera directa a una página**.**
- El nombre y el rol del usuario registrado apareceré con un color que realza su visibilidad (Naranja).
- En la importación de librerías, se debe usar el prefijo “f” para hacer referencia a etiquetas del núcleo de la implementación mientras que el prefijo “h” para hacer referencia a etiquetas de componentes HTML:

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
```

- Se debe poner en minúsculas los nombres de los ficheros .xhtml.
- La estructura de directorios estándar de una aplicación JSF ha de ser de la siguiente forma: en el directorio “src” se sitúan los beans de la aplicación, en “web” todas las páginas jsp ó html, así como las imágenes y demás archivos necesarios para la aplicación, y en “WEB-INF” los ficheros de configuración como el fichero descriptor de despliegue (web.xml) y el de configuración JSF (faces-config.xml).
- Para obtener código ordenado y no duplicado se utilizó el patrón Vista Compuesta.

En este capítulo, se mostró que con el uso de JSF y los componentes de AJAX se aumentó la robustez y la flexibilidad de la capa de presentación del SIAPS, lo que proporciona que el sistema sea escalable, dinámico y que provea una generación de respuestas rápidas para con el usuario. Con las buenas prácticas de diseño se obtuvo como resultado una interfaz uniforme acorde con las necesidades actuales y por ende la usabilidad del sistema es enriquecida.

Conclusiones

Al culminar la investigación, se han cumplido el objetivo y las tareas propuestas. Se ha profundizado el conocimiento sobre las tecnologías que contribuyen a mejorar el diseño de las aplicaciones web con el propósito de apoyar la informatización de la salud en Cuba. Se arribaron a las siguientes conclusiones:

Se desarrolló una interfaz de usuario asincrónica para el Sistema Integral de Salud para la Atención Primaria que le brinda mayor dinamismo y usabilidad.

Con la puesta en práctica de las pautas de diseño se obtuvo una interfaz de usuario estandarizada.

Con la realización de este trabajo se demostró que el uso de de AJAX y JSF aporta flexibilidad a las interfaces del SIAPS.

Con la utilización de Facelets fue posible llevar a cabo buenas prácticas para obtener código ordenado y sin duplicación en el diseño de la capa de presentación.

Recomendaciones

Las recomendaciones de la investigación están dirigidas a sugerir acciones para complementar el producto obtenido. Por lo que para el buen desempeño y puesta en marcha de la aplicación se hacen las siguientes recomendaciones:

Utilizar la propuesta definida e implementada, para su aplicación y estandarización en futuras aplicaciones del proyecto.

Poner en práctica la propuesta realizada para los sistemas de gestión de información, en cualquier área que se estime necesario.

Continuar la investigación sobre el tema, teniendo en cuenta las nuevas versiones que puedan surgir, con el objetivo de aportar nuevas mejoras aplicables al diseño del SIAPS.

Referencias Bibliográficas

1. Pérez, J. E. (2008). Introducción a AJAX.
2. Pérez, J. E. (2008). Introducción a AJAX.
3. Marcos, M.-C. (1999). DISEÑO DE INTERFACES WEB. Madrid
4. Derus...el Blog. (3 de agosto de 2006). Recuperado el 12 de enero de 2009, de Derus...el Blog: http://dersu.wordpress.com/2006/08/03/coincidencias/
5. Derus...el Blog. (3 de agosto de 2006). Recuperado el 12 de enero de 2009, de Derus...el Blog: http://dersu.wordpress.com/2006/08/03/coincidencias
6. Santiago, J. C. (24 de agosto de 2008). Recuperado el 20 de enero de 2009, de http://dukechile.blogspot.com/
7. Santiago, J. C. (24 de agosto de 2008). Recuperado el 2009 de enero de 20, de http://dukechile.blogspot.com

BIBLIOGRAFIA

1. ABU, MZE, POS, RBO, AMA. EL4J Reference Documentation Version 1.3 Incremental Improvements for Spring. Suiza: ELCA Informatique SA, 2008.
2. ASP de .NET http://quickstarts.asp.net/QuickStartv20/aspnet/doc/pages/pages.aspx
3. Ajax4jsf. http://labs.jboss.com/portal/jbossajax4jsf/downloads.
4. Ajax no es un fin. http://www.di.uniovi.es/~labra/cursos/Web20/web20.html#(23).
5. A Web-based Test of Fitts' Law http:/ http://www.tele-actor.net/fitts/index.html /.
6. Arquitectura de información y usabilidad - Sistemas de etiquetado. http://www. arquitectura_de_informacion_y_usabilidad-sistemas_de_etiquetado/8135-13
7. "CSS Mastery : advanced web standards solutions". Andy Budd. Apress Company, 2006
8. Demos vivos de RichFaces http://livedemo.exadel.com/richfaces- demo/richfaces/actionparam.jsf
9. Developer's Guide. s.l.: ICEsoft Technologies, Inc., 2007.
10. Facelets http://www.javahispano.org/contenidos/es/caracteristicas_jsf_2_0_opinion_publicada_en_solo_programadores/
11. Facelets http://www.elholgazan.com/2007/08/facelets-de-ajusta-jsf-como-un-guante.html

9. Fitts'Law:ModelingMovement Time in HCI http://www.cs.umd.edu/class/fall2002/cmsc838s/tichi/fitts.html

10. Filtors y Servlets en Java. http://www.saleman.com

11. Gallardo, David, McGovern, Robert. Eclipse In Action: A Guide for Web Developers. 2003.

12. Geary, David, Horstmann, Cay. Core JavaServer™ Faces, Second Edition. s.l.: Prentice Hall, 2007. 0-13-173886-0.

13. Hightower, Richard. IBM. Facelets fits JSF like a glove. [En línea] 21 de Febrero de 2006. [Citado el: 7 de mayo de 2009.] http://www.ibm.com/developerworks/java/library/j-facelets/.

14. Hookom, Jacob. Facelets - JavaServer Faces View Definition Framework. Facelets - JavaServer Faces View Definition Framework Developer Documentation. [En línea] 2008. [Citado el: 20 de mayo de 2009.] https://facelets.dev.java.net/nonav/docs/dev/docbook.html.

15. Java EE 5 (sucesor de J2EE):el reto de volver a empezar http://www.cincosoft.com

16." Java EE 5 Tutorial": http://java.sun.com/javaee/5/docs/tutorial/doc

17. Javalobby. Ajax: a new approach to web application. [En línea] 18 de febrero de 2005. [Citado el: 29 de abril de 2009.] http://www.javalobby.org/articles/ajax/.

18. JCP. The Java Community Process(SM) Program. JSRs: Java Specification Request -Li. [En línea] Mayo de 2006. [Citado el: 20 de mayo de 2009.] http://jcp.org/en/home/index.

19. JSF AJAX Component Library Feature Matrix. [En línea] 4 de Junio de 2008. [Citado el: 4 de abril de 2009.] http://www.jsfmatrix.net/.

20. JSF http://ame.endesa.es/confluence/display/AMEBASE/JSF

21. GPLhttp://www.monografias.com/trabajos55/licencias-de-software/licencias-de-software2.shtml

22. GPL http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=migrateJSF2Facelets

23. Mann, Kito D. JavaServer Faces in Action. s.l. : Manning Publications, 2005.

24. Manuales de Seam (web beans) http://www.seamframework.org/Documentation

25. Mateu, Carles. Desarrollo de aplicaciones web. Catalunya : Eureca Media, SL, 2004. 84-9788-118-4.

26. Microsystems, Sun. Sun Microsystems. JavaServer Pages Overview. [En línea] [Citado el: 7 de mayo de 2008.] http://java.sun.com/products/jsp/overview.html

27. Leyes y principios. Usabilidad y tecnología para entender la web [Citado el: 20 de Mayo de 2008.]

http://www.Usandolo.com /.

28. Leyes y principios: Fitts' Law.http://www.asktog.com/columns/022DesignedToGiveFitts.html /.

29. Pérez, Javier Eguíluz. Introducción a AJAX. 2007.

30. Pimentel, Luis Alberto González, Pérez, Iósev Rivero. ArBaWeb: arquitectura base sobre la web. 2007.

31. RichFaces Developer Guide. s.l.: Red Hat, 2007.

32. RichFaces Madeja: http://www.juntadeandalucia.es/ MADEJA/RichFaces

33. Ruiz, Yenivet Paula, González, Yanaisy Galbán. Tecnología Ajax para el desarrollo Web. 2007.

34. Sun. Java en castellano. Introducción a la Tecnología JavaServer Faces. [En línea] [Citado el: 20 de Mayo de 2008.] http://www.programacion.com/java/tutorial/jsf_intro/.

35. Servlet JSF http://localhost:8080/login/index.faces

36. Usabilidad http://www.usandolo.com/index.php?

37. Usabilidad http://www.gaiasur.com.ar/infoteca/siggraph99/diseno-de-interfaces-y-usabilidad

Glosario de Términos

API (Interfaz de Programación de Aplicaciones): conjunto de especificaciones para comunicarse con una aplicación, normalmente para obtener información y utilizarla en otros servicios. Una API representa una interfaz de comunicación entre componentes software. Se trata del conjunto de llamadas a bibliotecas que ofrecen acceso a servicios desde los procesos y representa un método para conseguir abstracción en la programación, generalmente entre los niveles o capas inferiores y los superiores del software. Uno de los principales propósitos de una API consiste en proporcionar un conjunto de funciones de uso general, por ejemplo, para dibujar ventanas o íconos en la pantalla.

Asíncrona: comunicación que no coincide en tiempo real entre el emisor y el receptor.

Beans: simples clases de Java con sus métodos de acceso. Se usan para conservar el estado del usuario y los datos de los componentes. Usualmente implementan métodos de validaciones y manejadores de eventos que son invocados por la aplicación desde sus componentes.

Cliente-Servidor: tipo de arquitectura distribuida formada por dos tipos de componentes: los clientes que hacen peticiones de un servicio, y los servidores que proveen este servicio.

Ergonomía: disciplina que busca que los humanos y la tecnología trabajen en completa armonía, diseñando y manteniendo los productos, puestos de trabajo, tareas, equipos, etc. en acuerdo con las características, necesidades y limitaciones humanas. Analiza aquellos aspectos que abarcan al entorno artificial construido por el hombre, relacionado directamente con los actos y gestos involucrados en toda actividad de éste.

HTTP (Protocolo de Transferencia de Hipertexto): protocolo usado en las transacciones de la Web. Define la sintaxis y la semántica que utilizan los elementos software de la arquitectura web (clientes, servidores, proxies) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

GLOSARIO DE TÉRMINOS

Java: lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Renderers: elementos que permiten generar una imagen a partir de un modelo. Son los responsables de mostrar los componentes de interfaz de usuarios y trasladar los datos de entrada del usuario al los valores del componente. Pueden ser diseñados para trabajar con uno o más componentes y un componente puede ser asociado a más de un Render.

RIA (Aplicaciones Ricas en Internet): formas avanzadas de que un usuario interactúe con una aplicación o página web, ofreciéndole funciones y nuevas posibilidades útiles intentando al mismo tiempo mantener la simplicidad aparente. Los entornos RIA no producen recargas de página, ya que desde el principio se carga toda la aplicación y sólo se produce comunicación con el servidor cuando se necesitan datos externos como datos de una base de datos o de otros ficheros externos.

Síncrona: comunicación que depende de la sincronización de los datos transmitidos, enviados y de sus propios mecanismos de transmisión.

Plugins: aplicación que interactúa con otra para agregarle una funcionalidad específica y es ejecutada por la aplicación principal. En el caso particular de Eclipse es un conjunto de clases que permite hacer el IDE más extensible.

XMLHttpRequest (Lenguaje de Marcado Extendido/Protocolo de Transferencia de Hipertexto): referido como XMLHTTP. Interfaz empleada para realizar peticiones HTTP y HTTPS a servidores WEB. Para los datos transferidos se usa cualquier codificación basada en texto, incluyendo: texto plano, XML, JSON, HTML y codificaciones particulares específicas. La interfaz se presenta como una clase de la que una aplicación cliente puede generar tantas instancias como necesite para manejar el diálogo con el servidor.