Universidad de las Ciencias Informáticas

“Facultad 10”

Título: “Módulo Recursos para el Sistema de Control de Documentos del Ministerio del Poder Popular para la Energía y Petróleo”

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

Autor(es): Yosvani Pérez Pérez

Harold Nieves Medina

Tutor: Ing. Pedro Rodríguez Samón

Ciudad de La Habana, Junio de 2009

Hay algo más importante que la lógica: la imaginación.

Alfred Hitchcock.

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores del trabajo titulado:

Módulo Recursos para el Sistema de Control de Documentos del Ministerio del Poder Popular para la Energía y Petróleo

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 ____ días del mes de _____ del año _____.

_____ _____

Yosvani Pérez Pérez Harold Nieves Medina

_____

Ing. Pedro Rodríguez Samón

AGRADECIMIENTOS

A mi madre, padre, tutor y demás personas que me tendieron la mano, cuando lo necesité.

H@rd.

A mi padre, novia, tutor y demás personas que me ayudaron para hacer realidad este sueño.

Yosvani.

DEDICATORIA

A mis seres queridos y en especial a mi hermano para que siga el ejemplo.

H@rd.

A mis padres, novia y demás seres queridos.

Yosvani.

RESUMEN

El trabajo en cuestión lleva por título, Módulo Recursos para el Sistema de Control de Documentos del Ministerio del Poder Popular para la Energía y Petróleo. El mismo se divide en tres capítulos y ofrece una propuesta de solución para la Gestión de Documentos del Ministerio mencionado anteriormente y más específico para la Gestión de sus Recursos. A lo largo del desarrollo de la aplicación se hacen uso de herramientas y tecnologías libres, como es el caso del Netbeans 6.5 para PHP, como Entorno de Desarrollo, Alfresco como Gestor de Contenidos y el Codelgniter como framework PHP. Se implementa el patrón de arquitectura Modelo-Vista-Controlador mediante dicho framework.

Entre los principales objetivos se encuentran, desarrollar un sistema totalmente libre y liviano. Dando cumplimiento a estos, se obtuvo una aplicación capaz de sintetizar todo el proceso de Gestión de Recursos de una manera fiable y rápida. Cabe destacar, que la interfaz final, se muestra muy amigable al usuario.

PALABRAS CLAVES

Gestión Documental.

Gestión de Recursos.

TABLA DE CONTENIDOS

INTRODUCCION

La informática en los últimos tiempos se ha convertido en parte del sustrato tecnológico del proceso de globalización en el cual está inmerso todo el mundo. En la actualidad el desarrollo de esta ciencia ha ido jugando un papel primordial en el incremento de la información a partir de sus procesos de automatización. La informática como ciencia ha traído consigo valiosos aportes científicos, económicos, permitiendo a su vez el desarrollo de las importantes tecnologías que tributan, sin dudas, al progreso actual de la humanidad.

El crecimiento de las nuevas tecnologías y de la Web en su conjunto, están teniendo un impacto significativo en los sistemas de Gestión y Control de Información en sectores comerciales, industriales, bancarios, financieros, educacionales, en fin, en la vida cotidiana. Son innumerables los beneficios que brinda la computación; rapidez en la obtención de resultados, almacenamiento de grandes volúmenes de información, facilidades para encontrar la adecuada y/o actualizada por parte de científicos, investigadores y profesionales, con los cuales se trabaja intensamente para que la calidad del software sea reconocida a nivel internacional.

La Universidad de las Ciencias Informáticas (UCI), pretende ser la vanguardia del desarrollo de las empresas de software en Cuba y llevar la informatización a todos los sectores de la sociedad. Una de las características fundamentales que presenta la gran casa de estudios UCI es la estrecha relación que existe entre el estudio y la producción, siendo este último un ente para el desarrollo de múltiples proyectos, relacionados en muchos casos con empresas extranjeras como es el caso del Ministerio del Poder Popular para la Energía y Petróleo (en lo adelante, MENPET), quien ha solicitado a la Universidad, el desarrollo de un sistema informático para gestionar la documentación generada en dicho Ministerio .

Entre las diferentes dependencias y oficinas de la sede del MENPET existe un constante intercambio de documentos generados por ellas o por entidades externas, del cual actualmente no se lleva un estricto seguimiento y control.

El MENPET ha estado utilizando desde hace alrededor de 15 años un sistema nombrado Sistema de Operaciones Secretariales (S.O.S.) para llevar el control de toda la correspondencia, tanto interna como externa. Dicho sistema, fue desarrollado sobre una plataforma privativa, con interfaces de escritorio, utilizando Borland Delphi 6.0 y Microsoft Access 2000, la cual no satisface todas las necesidades de los

clientes. Actualmente, las búsquedas y realización de algunos documentos como minutas y notas de remisión se realizan manualmente. El Ministerio cuenta además con un Sistema Interno Central (SIC) que permite llevar el control y seguimiento de los procesos de reclutamiento del personal, así como mantener un registro de la trayectoria de sus trabajadores y registro de estructura de cargos propuestos y aprobados. El actual S.O.S. no está integrado al SIC, por lo que se dificulta el acceso a la información contenida en el S.O.S.

Se necesita, por tanto, desarrollar un sistema que permita llevar el control de los documentos desde su generación y durante su trayectoria a través del Ministerio. Este debe incorporar funcionalidades no implementadas por los sistemas usados actualmente. Parte importante de este trabajo es el Control de los Recursos del Sistema de Control de Documentos, que actualmente no se realiza de la forma más óptima, por lo que surge el siguiente **problema científico**: ¿Cómo gestionar de manera factible los Recursos del Sistema de Control de Documentos del Ministerio del Poder Popular para la Energía y Petróleo?

Para ello se ha definido como **objeto de estudio**: los procesos de Gestión Empresarial; el **campo de acción** por su parte, serían los procesos de Gestión de Recursos en el MENPET.

El **objetivo general** que se pretende alcanzar en este trabajo es desarrollar el módulo Recursos del Sistema de Control de Documentos del MENPET.

Para darle cumplimiento al objetivo propuesto se plantean los siguientes **objetivos específicos**:

- ✓ Definir las herramientas adecuadas para el desarrollo del módulo Recursos.
- ✓ Redefinir el diseño para el módulo.
- ✓ Implementar las funcionalidades del sistema.

Se plantea como **idea a defender** que: con el desarrollo del módulo Recursos para el Sistema de Control de Documentos (en lo adelante, SCD) del MENPET utilizando herramientas libres se facilitará la Gestión de Recursos al personal del SCD.

Aportes prácticos esperados del trabajo.

El sistema será una aplicación Web que permitirá la Gestión de Recursos del Sistema de Control de Documentos, dígase, Tipo de Información, Tipo de Acción, Tipo de Oficina, Tipo de Solicitud, Organismos, Usuarios y Trabajadores Externos. El proceso incluirá la adición, modificación, eliminación, así como el listado de dichos Recursos. Se realizarán búsquedas avanzadas y se controlará la Gestión de Usuarios, además de los roles que estos tienen dentro del sistema.

Para el desarrollo de la investigación se utilizaron métodos teóricos y empíricos. Seleccionando dentro de los teóricos, Análisis y Síntesis y dentro de los empíricos, la Entrevista, a continuación la fundamentación de su uso.

Método teórico

- ✓ Análisis histórico-lógico: A través de este método se estudia la trayectoria real de los elementos que se utilizan en la implementación de sistemas Web, dígase origen y evolución de los diferentes lenguajes de programación, frameworks, Entornos de Desarrollo Integrado y otras herramientas muy importantes para el desarrollo de la solución informática aquí propuesta.

- ✓ Análisis y síntesis: Se analizaron varios documentos relacionados con los Sistemas de Gestión Documental (SGD), resumiendo las características y aspectos más importantes que estos poseen, seleccionando de ellos las herramientas más útiles para el desarrollo del módulo Recursos. Se realizó además una investigación previa de los procesos que intervienen en el desarrollo de software y los principales elementos que integran las metodologías como son las fases, disciplinas, actividades, roles, artefactos y otros. Los resultados alcanzados en conjunto con el conocimiento empírico, permitieron conseguir un dominio abarcador del tema al que se propone solución.

Método empírico

- ✓ Entrevista: Se realizaron múltiples entrevistas a los analistas, jefes y demás implicados del proyecto en aras de obtener información referente al funcionamiento y dinámica del negocio en el que se enmarca el problema, que ya analizaron y cuya implementación se propone con este trabajo.

Estructuración del contenido.

El presente trabajo está conformado por 3 capítulos.

En el **capítulo 1** se realiza una panorámica sobre La Gestión Documental y las aplicaciones Web, haciendo alusión a sus tendencias actuales. De igual manera se detalla el surgimiento y actual auge de un nuevo paradigma de programación, La Programación Orientada a Servicios. Luego se exponen las tecnologías de software y herramientas usadas a lo largo de la implementación, haciendo énfasis en sus ventajas.

En el **capítulo 2** se realiza una valoración crítica del diseño propuesto por los analistas del proyecto Gestión Documental y Archivística, al que se le da continuidad en el presente trabajo, con la implementación. También se describen las nuevas clases y operaciones necesarias para darle solución al problema.

En el **capítulo 3** se hace referencia al diseño de las pruebas de unidad que permitan validar la solución propuesta, detallando de las mismas su objetivo, alcance y tipo, al igual que los valores utilizados para la ejecución de dichas pruebas.

CAPITULO 1. FUNDAMENTACION TEORICA

En el presente capítulo se expone el marco teórico referencial del trabajo, realizando una breve disertación sobre la Gestión Documental y las Aplicaciones Web.
Luego de haber realizado un estudio detallado, se enumeran las tecnologías de software y herramientas a utilizar, apostando por supuesto por el Software Libre. Estas fueron previamente consultadas y avaladas por la máxima dirección del proyecto en cuestión (Gestión Documental y Archivística); de conjunto con el cliente, exceptuando al paradigma de programación, framework PHP y al entorno de desarrollo utilizado, sobre los cuales se ofrece un breve estado del arte, para luego definir el que se usará.

1.1 Gestión Documental (1)

La Gestión Documental no es algo nuevo. El conjunto de principios y técnicas para la organización de los archivos en papel se ha ido desarrollando desde hace más de 1000 años de antigüedad. La disciplina de la documentación, además, ha debido evolucionar en el último siglo ante un crecimiento exponencial de la información, desarrollando nuevas técnicas para permitir el acceso a grandes volúmenes de esta.
En la actualidad, coexisten en el mundo los más diversos sistemas de Gestión Documental: desde el simple registro manual de la correspondencia que entra y sale, hasta los más sofisticados sistemas informáticos que manejan no sólo la documentación administrativa, sino que además controlan los flujos de trabajo del proceso de tramitación de los expedientes, capturan información desde bases de datos de producción, contabilidad y otros, enlazan con el contenido de archivos, bibliotecas, centros de documentación y permiten realizar búsquedas sofisticadas y recuperar información de cualquier lugar.
La Gestión Documental es definida como: el conjunto de normas, técnicas y prácticas usadas para administrar el flujo de documentos de todo tipo en una organización, permitir la recuperación de información desde ellos, determinar el tiempo que los documentos deben guardarse, eliminar los que ya no sirven y asegurar la conservación indefinida de los documentos más valiosos, aplicando principios de racionalización y economía.
Con el transcurso de los años se hizo necesario automatizar los procesos que la Gestión Documental contiene y surgen los Sistemas de Gestión Documental (SGD) con el propósito de almacenar y recuperar documentos, que debe estar diseñado para coordinar y controlar todas aquellas funciones y actividades específicas que afectan a la creación, recepción, almacenamiento, acceso y preservación de los

documentos, salvaguardando sus características estructurales y contextuales, garantizando así su autenticidad y veracidad.

1.1.1 Características de los Sistemas de Gestión Documental (2)

Son varios los autores que hacen alusión a las características principales que debe poseer un Sistema de Gestión Documental, la mayoría coinciden en las presentadas a continuación:

- ✓ Manejo de grandes volúmenes de documentación.
- ✓ Garantía de acceso a la información más actual.
- ✓ Mantenimiento coherente de la información procedente de diferentes compañías y organizaciones.
- ✓ Definición de flujos de trabajo en el sistema para la gestión de procesos operativos entre departamentos y empresas externas.
- ✓ Gestión de la información en formato nativo.
- ✓ Control de acceso a la información.
- ✓ Seguridad ante la posible pérdida de documentación.

1.1.2 Componentes de la Gestión Documental (3)

Los sistemas informáticos de Gestión Documental suelen integrar una serie de elementos comunes a todos ellos.

Metadatos

Al margen de la información contenida en el documento, conviene almacenar información del propio documento: fecha de creación, autor, tamaño, propiedades, compañía, materia que trata. En la actualidad prácticamente cualquier aplicación informática que genere documentos es capaz de almacenar todos estos metadatos.

Integración

Integración de todo tipo de documentos en la misma aplicación de gestión de forma que un usuario pueda crearlo, abrirlo, editarlo, guardar una nueva versión, todo sin salir de la aplicación.

Captura

Posibilidad de obtener y digitalizar documentos como imágenes, o textos que se tienen en papel u otros formatos, escaneándolos. Esto incluye reconocimiento óptico de caracteres, abreviado habitualmente como OCR (Optical character recognition), que al escanear texto lo traslada a documento de texto, en vez de documento de imagen. Una vez digitalizados es posible integrarlos en el Sistema de Gestión Informático.

Indexación

Consiste en poner identificadores únicos a los documentos de forma que sean más sencillos de localizar. Además de esto, cuando el volumen de documentos es grande conviene utilizar otras técnicas como categorización y agrupaciones de índices para que se puedan hacer búsquedas en árbol y no sea necesario recorrer todos los elementos en una búsqueda.

Almacenamiento

Almacenamiento de los documentos electrónicos, en una base de datos. Contemplará funcionalidad para definir cuánto tiempo se mantienen, cómo hacer migraciones a otro sistema de almacenamiento y sistema de copias de seguridad, para recuperar en caso de fallo del primer sistema.

Recuperación

Herramientas para recuperar un determinado archivo almacenado. No siempre se conocerá el identificador único del archivo, por lo que la aplicación debería proporcionar la posibilidad de encontrarlo por título, u otros metadatos como tamaño, autor, etc., pudiendo hacer búsquedas más eficientes.

Distribución

La aplicación deberá proporcionar un canal de distribución de los archivos, en caso de que estos deban ser distribuidos.

Seguridad

La aplicación debe garantizar que los documentos tengan el nivel de seguridad adecuado. Habrá documentos con información sensible que no deberían poder ser accedidos más que por el personal indicado. Además, debe aportar seguridad frente a posibles intrusiones externas a la organización.

Flujo de trabajo

El flujo de transmisión de los documentos se define por parte de la organización.

Colaboración

En muchos casos, los documentos no serán tarea de una sola persona, si no que habrá varias trabajando en el mismo. En este caso, la aplicación debe proporcionar un soporte para que las actualizaciones que se hacen sobre un documento no se alteren mutuamente. Esto se suele conseguir con el bloqueo de documentos, o sea que cuando una persona está trabajando sobre este nadie más puede utilizarlo.

Control de versiones

Como muchas personas pueden trabajar sobre el mismo documento, es necesario llevar un control de versiones del mismo. Las aplicaciones de Gestión Documental suelen integrar esto, de forma que cada vez que un documento se modifica, se guarda un histórico con el autor y los cambios realizados. Este histórico, además del control, permite volver a versiones anteriores en caso de que una nueva modificación no sea aceptada.

1.1.3 Funcionalidades de los Sistemas de Gestión Documental (2)

Funcionalidades básicas:

- ✓ Sistemas y procedimientos para la incorporación masiva de documentos en diversos formatos electrónicos y procedentes de sistemas informáticos heterogéneos.
- ✓ Sistemas de escaneo para la incorporación masiva o discreta de documentos que están en formatos no electrónicos.
- ✓ Gestión de documentos y sus metadatos.
- ✓ Creación de relaciones entre documentos, así como, entre estos y el resto de componentes del sistema documental (carpetas, metadatos, entre otros).

- ✓ Mantenimiento de histórico de los documentos.
- ✓ Localización de documentos mediante técnicas de búsqueda.
- ✓ Edición de archivos utilizando sus herramientas nativas de creación.
- ✓ Disposición de un software para la visualización de archivos en los formatos más comunes.
- ✓ Control del acceso a documentos según perfiles de usuario.
- ✓ Anotación en documentos sin modificar el contenido de los mismos.
- ✓ Importación de información contenida en otras aplicaciones.
- ✓ Herramientas de flujo de trabajo que incluyan un control del mismo.
- ✓ Sistemas de impresión de documentos.

Funcionalidades avanzadas:

- ✓ La gestión del ciclo de vida completo de un documento, manteniendo la traza de revisiones y el histórico de cambios.
- ✓ La relación de los documentos con los elementos o activos.

1.1.4 Ventajas de los Sistemas de Gestión Documental:

- ✓ Reducción de costes de los procesos empresariales, mediante el rediseño de procesos, sustitución del trabajo administrativo no productivo y reducción del espacio físico de almacenamiento.
- ✓ Reducción de los ciclos de trabajo, aumentando la concurrencia de las distintas actividades necesarias.
- ✓ Unificación de los procesos empresariales en los distintos ámbitos departamentales y geográficos, potenciando los canales formales y los procedimientos de trabajo lo que facilitará el cumplimiento de los requerimientos que los sistemas de calidad imponen.
- ✓ Aumento de las capacidades de comunicación en toda la organización, mejorando la integridad y seguridad de la información.(3)

1.2 Aplicaciones Web. Tendencias y Actualidad (4)

Con la introducción de Internet y de la Web en concreto, se han abierto infinidad de posibilidades en cuanto al acceso y uso de información desde cualquier parte del mundo. Con los avances en la tecnología cada vez se demandan aplicaciones más rápidas, ligeras y robustas que permitan ser usadas sin importar

el lugar u horario.

Las aplicaciones Web ofrecen grandes ventajas que pueden ser aprovechadas por muchas organizaciones, sobre todo ahora que la globalización es una realidad. Entre las ventajas que se pueden mencionar están:

- ✓ No requieren instalación. Pues usan tecnología Web, lo cual permite el aprovechamiento de todas las características de Internet.
- ✓ Son fáciles de usar. Pues no requieren conocimientos avanzados de computación.
- ✓ Alta disponibilidad. Ya que puede realizar consultas en cualquier parte del mundo donde tenga acceso a Internet y a cualquier hora.
- ✓ Compatibilidad multiplataforma. Las aplicaciones Web tienen un camino mucho más sencillo para la compatibilidad multiplataforma que las aplicaciones de software descargables
- ✓ Actualización. Las aplicaciones basadas en Web están siempre actualizadas con el último lanzamiento sin requerir que el usuario tome acciones pro-activas.
- ✓ Menos Bugs. Con aplicaciones Web, todos utilizan la misma versión, y todos los bugs pueden ser corregidos tan pronto como son descubiertos.
- ✓ Precio. Las aplicaciones basadas en Web no requieren la infraestructura de distribución, soporte técnico y marketing requerido por el software descargable tradicional.

1.3 Arquitectura Cliente-Servidor (5)

Esta arquitectura consiste básicamente en que un programa -el cliente- realiza peticiones a otro programa -el servidor- que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la Gestión de la Información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores Web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Al estar este sistema, basado en la Web, responde a dicho tipo de arquitectura.

1.4 Paradigmas de Programación

Los paradigmas son “formas” de resolver un problema mediante la programación. Representan las técnicas y procedimientos que se llevan a cabo en la estructuración de un programa. En palabras simples, son modelos de programación; los cuales fueron diseñados para solucionar ciertos problemas.

No hay mejor o peor paradigma, solo existe la mejor o peor aplicación de modelo de programación para solucionar problemas. Por ser éstos; “métodos” de programación, es posible mezclar diversos tipos de paradigmas. (6)

1.4.1 Paradigma Imperativo o por Procedimientos (7)

La programación imperativa, en contraposición a la programación declarativa es un paradigma de programación que describe la programación en términos del estado del programa y sentencias que cambian dicho estado. Los programas imperativos son un conjunto de instrucciones que le indican a la computadora cómo realizar una tarea.

La implementación de hardware es imperativa; prácticamente todo el hardware de las computadoras está diseñado para ejecutar código de máquina, que es nativo a la misma, escrito en una forma imperativa. Esto se debe a que el hardware de estas implementa el paradigma de las Máquinas de Turing. Desde esta perspectiva de bajo nivel, el estilo del programa está definido por los contenidos de la memoria, y las sentencias son instrucciones en el lenguaje de máquina nativo a la computadora.

Los lenguajes imperativos de alto nivel usan variables y sentencias más complejas, pero aún siguen el mismo paradigma. Las recetas y las listas de revisión de procesos, a pesar de no ser programas de computadora, son también conceptos familiares similares en estilo a la programación imperativa; cada paso es una instrucción, y el mundo físico guarda el estado.

1.4.2 Paradigma Funcional (8)

La Programación Funcional es un paradigma de programación declarativa basado en la utilización de funciones matemáticas.

El objetivo es conseguir lenguajes expresivos y matemáticamente elegantes, en los que no sea necesario bajar al nivel de la máquina para describir el proceso llevado a cabo por el programa, y evitando el

concepto de estado del cómputo.

Los programas escritos en un lenguaje funcional están constituidos únicamente por definiciones de funciones, entendiendo éstas no como subprogramas clásicos de un lenguaje imperativo, sino como funciones puramente matemáticas.

Otras características propias de estos lenguajes son la no existencia de asignaciones de variables y la falta de construcciones estructuradas como la secuencia o la iteración.

Existen dos grandes categorías de lenguajes funcionales: los funcionales puros y los híbridos. La diferencia entre ambos estriba en que los lenguajes funcionales híbridos son menos dogmáticos que los puros, al admitir conceptos tomados de los lenguajes imperativos, como las secuencias de instrucciones o la asignación de variables. En contraste, los lenguajes funcionales puros tienen una mayor potencia expresiva, conservando a la vez su transparencia referencial, algo que no se cumple siempre con un lenguaje funcional híbrido.

1.4.3 Paradigma Lógico (9)

La programación lógica consiste en la aplicación del corpus de conocimiento sobre lógica para el diseño de lenguajes de programación; no debe confundirse con la disciplina de la lógica computacional.

La programación lógica comprende dos paradigmas de programación: la programación declarativa y la programación funcional. La programación declarativa gira en torno al concepto de predicado, o relación entre elementos. La programación funcional se basa en el concepto de función, de corte más matemático.

Históricamente, los ordenadores se han programado utilizando lenguajes muy cercanos a las peculiaridades de la propia máquina: operaciones aritméticas simples e instrucciones de acceso a memoria. Un programa escrito de esta manera puede ocultar totalmente su propósito a la comprensión de un ser humano, incluso uno entrenado. Hoy día, estos lenguajes pertenecientes al paradigma de la programación imperativa han evolucionado de manera que ya no son tan crípticos.

En cambio, la lógica matemática es la manera más sencilla, para el intelecto humano, de expresar formalmente problemas complejos y de resolverlos mediante la aplicación de reglas, hipótesis y teoremas. De ahí que el concepto de "programación lógica" resulte atractivo en diversos campos donde la programación tradicional es un fracaso.

1.4.4 Paradigma Orientado a Objetos (10)

La Programación Orientada a Objetos (POO u OOP según sus siglas en inglés, Object-oriented Programming) es un paradigma de programación que usa objetos y sus interacciones para diseñar aplicaciones y programas de computadora. Está basado en varias técnicas, incluyendo herencia, modularidad, polimorfismo y encapsulamiento. Su uso se popularizó a principios de la década de 1990. Actualmente son muchos los lenguajes de programación que soportan la orientación a objetos.

Los objetos son entidades que combinan estado, comportamiento e identidad:

- ✓ El estado está compuesto de datos, será uno o varios atributos a los que se habrán asignado unos valores concretos (datos).
- ✓ El comportamiento está definido por los procedimientos o métodos con que puede operar dicho objeto, es decir, qué operaciones se pueden realizar con él.
- ✓ La identidad es una propiedad de un objeto que lo diferencia del resto, dicho con otras palabras, es su identificador (concepto análogo al de identificador de una variable o una constante).

La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.

De esta forma, un objeto contiene toda la información que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases e incluso frente a objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos. A su vez, los objetos disponen de mecanismos de interacción llamados métodos que favorecen la comunicación entre ellos. Esta comunicación favorece a su vez el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separan ni deben separarse el estado y el comportamiento.

Los métodos (comportamiento) y atributos (estado) están estrechamente relacionados por la propiedad de conjunto. Esta propiedad destaca que una clase requiere de métodos para poder tratar los atributos con los que cuenta. El programador debe pensar indistintamente en ambos conceptos, sin separar ni darle mayor importancia a ninguno de ellos, hacerlo podría producir el hábito erróneo de crear clases contenedoras de información por un lado y clases con métodos que manejen a las primeras por el otro. De esta manera se estaría realizando una programación estructurada camuflada en un lenguaje de programación orientado a objetos.

Esto difiere de la programación estructurada tradicional, en la que los datos y los procedimientos están

separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida. La programación estructurada anima al programador a pensar sobre todo en términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan. En la programación estructurada sólo se escriben funciones que procesan datos. Los programadores que emplean éste nuevo paradigma, en cambio, primero definen objetos para luego enviarles mensajes solicitándoles que realicen sus métodos por sí mismos.

1.4.5 Paradigma Orientado a Servicios (11)

En la última década, los arquitectos de software descubrieron que debían dejar de pensar en su aplicación como una entidad aislada de la infinidad de aplicaciones existentes desarrolladas en múltiples y diferentes plataformas y lenguajes, y que diseñar aplicaciones ubicuas e interoperables entre sistemas heterogéneos daba a sus productos una ventaja muy grande sobre el resto de sus competidores.

La industria del software comenzó a desarrollar aplicaciones con el paradigma de la programación en capas, muy relacionado con la **Programación Orientada a Objetos** (OOP, Object-oriented Programming), que permitía entre otras bondades el acceso ubicuo a las reglas de negocio de una aplicación. Pero la gran mayoría de estas implementaciones de aplicaciones distribuidas tenían una debilidad muy importante: no permitían su utilización desde plataformas diferentes a la cuales el sistema fue construido y en el caso de que lo hicieran, se requería una gran cantidad de **gadgets** (artefactos ingeniosos y complejos) de software para su utilización, que por lo general conllevaban una penalidad de performance inaceptable. Fue así que se comenzó a desarrollar el concepto de Web Service (Web Services) que en resumidas cuentas ofrecían la posibilidad de llamar a un componente de negocios desarrollado en una plataforma X desde una aplicación corriendo en cualquier plataforma, en cualquier parte del mundo, utilizando para ello protocolos estándares como **SOAP, XML y HTTP**.

La Programación Orientada a Servicios es un complemento de la Programación Orientada a Objetos e implementa mejoras a esta última, producto de la experiencia acumulada en la última década, sobre todo en las áreas de la computación distribuida, instalación de una solución e interoperabilidad entre sistemas heterogéneos.

Los sistemas orientados a servicios, son diseñados con un bajo nivel de acoplamiento que facilita la implementación de cambios y estos servicios pueden ser desarrollados en cualquier lenguaje corriendo en diferentes plataformas.

Los servicios son utilizados por una aplicación cliente que les envía mensajes respetando un determinado contrato, pero internamente esos servicios utilizan los conceptos de OOP y programación en capas clásico.

Luego de analizar los distintos paradigmas de programación, paso fundamental para la implementación del sistema, se analizan y proponen las tecnologías de software que igualmente se usarán a lo largo del desarrollo.

1.5 Análisis de las diferentes tecnologías de software

1.5.1 Metodología de Desarrollo de Software. RUP

Teniendo en cuenta que el diseño que se realizó para la solución informática en cuestión utiliza RUP como metodología de desarrollo, se decidió seguir utilizando esta, que además es la metodología más utilizada en el mundo para soluciones de este tipo; a continuación se ofrece una breve panorámica con el objetivo de esclarecer otros motivos de por qué usar RUP.

Constituye la metodología tradicional estándar más utilizada para el análisis, implementación y documentación de Sistemas Orientados a Objetos.

Se caracteriza por ser iterativa e incremental, centrada en la arquitectura y guiada por los casos de uso. Divide el proceso de desarrollo en ciclos de vida, obteniendo un producto al final de cada ciclo, los cuales se dividen en fases que deben de terminar con un hito, dentro de estas fases se encuentran: Inicio, Elaboración, Construcción y Transición. Es una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software. Unifica los mejores elementos de metodologías anteriores, está preparada para desarrollar grandes y complejos proyectos y a la vez se ajusta a proyectos pequeños o de mediano tamaño. Utiliza el UML como lenguaje de representación visual para preparar todos los esquemas de un sistema software.

De forma general proporciona una guía para ordenar las actividades del equipo de trabajo, dirige las áreas de cada desarrollador por separado y del equipo como un todo, especifica los artefactos que deben desarrollarse y ofrece criterios para el control y la medición de los productos y actividades de proyectos.

Independientemente de ser RUP la metodología usada para el Análisis y Diseño que antecede al trabajo en cuestión, se reconocen sus facilidades para controlar y guiar el trabajo lo mismo por separado que por equipo y su fácil adaptación a proyectos de mediano y pequeño tamaño. (12)

1.5.2 Lenguaje de programación a utilizar. PHP (13)

PHP (acrónimo de PHP: Hypertext Preprocessor), es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor. Entre sus principales características cabe destacar su potencia, su alto rendimiento, su facilidad de aprendizaje y su escasez de consumo de recursos. Este es uno de los lenguajes de lado del servidor más extendidos en la Web, se trata de un lenguaje de creación relativamente creciente que ha tenido una gran aceptación en la comunidad de Webmasters debido sobre todo a la potencia y simplicidad que lo caracterizan. PHP permite insertar sus pequeños fragmentos de código dentro de la página HTML y realizar determinadas acciones de una forma fácil y eficaz, sin tener que generar programas, programados íntegramente en un lenguaje distinto al HTML.
En este momento se encuentra en una fase de consolidación tras unos cuantos años de éxito, y la fase expansiva ha sido más bien dejada atrás para madurar en aspectos más relacionados con la integración de sus partes entre sí.

Ventajas de uso.

- ✓ Es un lenguaje multiplataforma.
- ✓ Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- ✓ Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- ✓ Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- ✓ Permite las técnicas de Programación Orientada a Objetos.
- ✓ Biblioteca nativa de funciones sumamente amplia e incluida.
- ✓ No requiere definición de tipos de variables.
- ✓ Tiene manejo de excepciones (desde PHP5).

1.5.3 Patrón de Arquitectura.MVC (14)

El patrón fue descrito por primera vez en 1979 por Trygve Reenskaug, entonces trabajando en Smalltalk en laboratorios de investigación de Xerox.
Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve

frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

Describiendo el patrón.

- ✓ Modelo: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.
- ✓ Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- ✓ Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

1.5.4 Web Service (15)

El concepto de Web Service (Servicio Web) suele confundirse con el de “servicio interactivo”, es decir, con “servicios que se prestan a través de una página Web”. Sin embargo, los Web Services son un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web, intercambiando datos entre sí con el objetivo de ofrecer algunos servicios. Estos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones que interactúan entre sí para presentar información dinámica al usuario.

Los Web Services son multiplataforma, es decir, son independientes tanto de la arquitectura (pueden comunicar ordenadores, PDAs, teléfonos móviles, estaciones de trabajo, etc.) como del Sistema Operativo y el lenguaje de programación que se use, ya que se basan en XML o cualquier otro estándar para el intercambio de datos.

Los Web Services ofrecen un medio de intercambio de datos entre distintos dispositivos, lo cual los hace interesantes para dar soporte a multitud de aplicaciones distribuidas, como son las relacionadas con el desarrollo de Gestores de Contenido. Además, los Web Service pueden combinarse con el uso de agentes, de forma que se puede dotar de cierta inteligencia a las aplicaciones para permitir la monitorización inteligente de un comportamiento rutinario dentro del tiempo de vida de una aplicación distribuida, la toma automática de decisiones en situaciones de alarma, entre otras muchas ventajas que presta el desarrollo de aplicaciones haciendo uso de los mismos.

1.5.5 Arquitectura Orientada a Servicios (16)

La Arquitectura Orientada a Servicios (en inglés Service-Oriented Architecture o SOA), es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requerimientos de software del usuario. SOA proporciona una metodología y un marco de trabajo para documentar las capacidades de negocio y puede dar soporte a las actividades de integración y consolidación. Lo que es relativamente nuevo es la implementación de SOAs basadas en Web Services. Cuando la mayoría de las personas hablan de una arquitectura orientada a servicios están haciendo referencia a un juego de servicios residentes en Internet o en una intranet, usando Web Services. Hay un juego de estándares de los que se habla ligados a los Web Services: XML, HTTP, SOAP, WSDL, UDDI.

Características de la Arquitectura Orientada a Servicios:

- ✓ Los nodos de la red hacen disponibles sus recursos a otros participantes en la red como servicios independientes a los que tienen acceso de un modo estandarizado.
- ✓ La mayoría de las definiciones de SOA identifican la utilización de Web Services en su implementación, no obstante se puede implementar una SOA utilizando cualquier tecnología basada en servicios.
- ✓ Las SOAs están formadas por servicios de aplicación débilmente acoplados y altamente interoperables.
- ✓ Para comunicarse entre sí, estos servicios se basan en una definición formal independiente de la plataforma subyacente y del lenguaje de programación (WSDL).
- ✓ La definición de la interfaz encapsula (oculta) las particularidades de una implementación, lo que la hace independiente del fabricante, del lenguaje de programación o de la tecnología de desarrollo (como Java o .NET).
- ✓ Con esta arquitectura, se pretende que los componentes software desarrollados sean muy reusables, ya que la interfaz se define siguiendo un estándar; así, un servicio en ASP.NET podría ser usado por una aplicación Java.

¿Qué es HTTP? (17)

El protocolo de transferencia de hipertexto (HTTP, HyperText Transfer Protocol) es el protocolo usado en cada transacción de la Web (WWW). HTTP fue desarrollado por el consorcio W3C y la IETF, que especifica la versión 1.1. HTTP 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. Al cliente que efectúa la petición (un navegador o un spider) se le conoce como "usar agente" (agente del usuario). A la información transmitida se le llama recurso y se le identifica mediante una URL. Los recursos pueden ser archivos, el resultado de la ejecución de un programa, una consulta a una base de datos, la traducción automática de un documento, etc. HTTP es un protocolo sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores.

¿Qué es SOAP? (17)

SOAP (Simple Object Access Protocol) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo deriva de un protocolo creado por David Winer en 1998, llamado XML-RPC. SOAP fue creado por Microsoft e IBM, está actualmente bajo el auspicio de la W3C. Es uno de los protocolos utilizados en los Web Services.

¿Qué es UDDI? (17)

Es un modelo de directorios para Web Services. Es una especificación para mantener directorios estandarizados de información acerca de los Web Services, sus capacidades, ubicación, y requerimientos en un formato reconocido universalmente. UDDI utiliza WSDL para describir las interfaces de los Web Services.

Es un lugar en el cual se pueden buscar cuáles son los Web Services disponibles, una especie de directorio en el cual se encuentran los Web Services publicados y publicar los Web Services que se desarrollarán.

¿Qué es XML? (18)

XML, sigla en inglés de eXtensible Markup Language (((lenguaje de marcas extensible))), es un

metalenguaje extensible de etiquetas, permite definir la gramática de lenguajes. XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable. XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

¿Qué es WSDL?(17)

Es un protocolo basado en XML que describe los accesos al Web Service. Se podría decir que es el manual de operación del Web Service, porque indica cuáles son las interfaces que provee el Web Service y los tipos de datos necesarios para la utilización del mismo.

1.6 Herramientas

1.6.1 Gestor de Contenidos. Alfresco

Como parte del desarrollo del trabajo en cuestión, se decidió hacer uso del sistema de administración de contenidos de código libre / abierto, basado en estándares abiertos y de escala empresarial para Windows y sistemas operativos similares a Unix, Alfresco.

Debido a que la conformidad, el servicio de cliente, la continuidad del negocio o la colaboración de la gestión eficaz del documento sean requisitos para cualquier institución, los usuarios de hoy quieren simplificar la solución departamental, con simples usos de configuración de aplicaciones. Las corporaciones quieren un control consistente, un sistema robusto de gestión de contenido.

Alfresco ofrece el principal sistema de gestión de documentos de código abierto que permite búsquedas y colaboración de documentos con servicios completos de biblioteca y gestión de ciclo de vida. Posibilita además una gestión de documentos usando una interfaz familiar para conseguir una rápida adopción construida en un repositorio que ofrece transparencia.

Características principales de Alfresco (19)

Gestor Documental de Alfresco

- ✓ Sistema de archivos virtuales. Sustituir los discos compartidos y ofrecer la misma interface.

- ✓ Motor de Reglas. Configuración de reglas de plug–in rules para automatizar procesamientos manuales y ofrecer conformidad.
- ✓ Búsqueda como Google. Busca directamente a través de FireFox o IE7.
- ✓ Navegación como Yahoo!. Extracción automática de meta datos y clasificación.
- ✓ Los mejores espacios de colaboración para practicar.
- ✓ Gestión del ciclo de vida.

Contribución al Contenido

- ✓ Sistema virtual de archivos haciendo que ECM sea como un simple disco compartido.

Clasificación del Contenido

- ✓ Automática extracción de metadatos y clasificación para todas las interfaces.

Búsquedas Avanzadas

- ✓ Nativo OpenSearch de FireFox o IE7 a través de los múltiples repositorios de Alfresco y a través de otros repositorios de Open Search.
- ✓ Gestión de Datos y Motor de Transformación.
- ✓ Servicios de transformación. De Office a ODF/PDF, de PowerPoint a Flash.

Colaboración de Equipo

- ✓ Asistente de espacios. La mejor práctica de estructura de carpetas, contenido, plantillas, reglas y procesos.
- ✓ Soporte de foros. Hilos de discusiones.
- ✓ Notificaciones de cambios por RSS y E-mail.

Flujos de trabajo Integrados

- ✓ Soporte para flujos de trabajo complejos.
- ✓ Gestión de Tareas.
- ✓ Conformidad a la Normativa.

Seguridad

- ✓ Seguridad y gestión de usuarios, grupos y roles.
- ✓ Seguridad de documentos.
- ✓ Acceso a través de NTLM o LDAP.

Beneficios

- ✓ Incremento de adopción por los usuarios.
- ✓ Reducción de costo inicial y TCO.
- ✓ Riesgo reducido.
- ✓ Despliegue rápido.

1.6.2 Framework PHP

Un framework, en el desarrollo de software, es una estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, 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.

Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio. (20)

Las principales ventajas de la utilización de un framework son:

- ✓ El desarrollo rápido de aplicaciones. Los componentes incluidos en un framework constituyen una capa que libera al programador de la escritura de código de bajo nivel.
- ✓ La reutilización de componentes software al por mayor.
- ✓ El uso y la programación de componentes que siguen una política de diseño uniforme.
- ✓ Un framework orientado a objetos logra que los componentes sean clases que pertenezcan a una gran jerarquía de clases, lo que resulta en bibliotecas más fáciles de aprender a usar.

Por las ventajas antes descritas, la utilización de un framework en la implementación sería de gran ayuda, a continuación se proponen varios, seleccionando, el más conveniente.

Symfony (21)

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

Como desventajas se puede citar que, para su utilización se requiere de la generación de grandes cantidades de código y una configuración exhaustiva de antemano. También que precisa de una estructuración estricta de sus directorios que se establece por consola a la hora de crear módulos.

Zend Framework (22)

Zend Framework (ZF), es un framework de código abierto para desarrollar aplicaciones Web y servicios web con PHP5. Es una implementación que usa código 100% orientado a objetos. La estructura de los componentes de ZF es algo único; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado.

Aunque se pueden utilizar de forma individual, los componentes de la biblioteca estándar de Zend Framework conforman un potente y extensible framework de aplicaciones Web al combinarse. ZF ofrece un gran rendimiento y una robusta implementación MVC, una abstracción de base de datos fácil de usar, y un componente de formularios que implementa la prestación de formularios HTML, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos.

CodeIgniter (23)

CodeIgniter es un conjunto de herramientas para personas que construyen su aplicación Web usando PHP. Su objetivo es permitirle desarrollar proyectos mucho más rápido de lo que podría si lo escribiese desde cero, proveyéndole un rico juego de librerías para tareas comúnmente necesarias, así como una interfaz simple y estructura lógica para acceder a esas librerías. CodeIgniter le permite creativamente enfocarse en su proyecto minimizando la cantidad de código necesaria para una tarea dada.

Características:

- ✓ Sistema basado en Modelo-Vista-Controlador.
- ✓ Compatible con PHP 4.
- ✓ Extremadamente liviano.
- ✓ Clases de base de datos llenas de características con soporte para varias plataformas.
- ✓ Soporte de Active Record para Base de Datos
- ✓ Formulario y validación de datos.
- ✓ Seguridad y filtro XSS
- ✓ Manejo de sesión
- ✓ Clase de carga (upload) de archivo
- ✓ Clase de FTP
- ✓ Localización
- ✓ Paginación
- ✓ Encriptación de datos
- ✓ Historial de errores
- ✓ Perfilando la aplicación
- ✓ Clase de calendario
- ✓ Clase de agente del usuario
- ✓ Clase de codificación Zip
- ✓ Clase de Motor de Plantillas
- ✓ Librería XML-RPC
- ✓ URLs amigables a motores de búsqueda

- ✓ Soporte para ganchos, extensiones de clase y plugins
- ✓ Larga librería de funciones "asistentes"

Ventajas de utilizar Codelgniter:

- ✓ Es un framework verdaderamente liviano.
- ✓ Es un framework que requiere apenas de configuraciones.
- ✓ Es un framework para cuyo uso no se requiere seguir estrictas reglas de codificación.
- ✓ Se utiliza cuando no se necesita de librerías de gran escala como el Repositorio de Extensiones y aplicaciones de PHP (PEAR).
- ✓ No se precisa ser forzado a aprender un lenguaje de plantilla para su utilización (aunque dispone de uno si se quiere utilizar).
- ✓ Se evita la complejidad, arribando a soluciones más simples.
- ✓ Se dispone de documentación a lo largo de todo el proceso.

Una vez expuestos los frameworks, Symfony, Zend Framework y Codelgniter, se decide hacer uso de este último pues es el que mejor se ajusta a la implementación, teniendo en cuenta las necesidades que imperan en dicha solución.

1.6.3 PHP SDK

¿Qué es SDK? (24)

SDK, kit de desarrollo de software, es generalmente un conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones para un sistema concreto.

Es algo tan sencillo como una interfaz de programación de aplicaciones o API (del inglés Application Programing Interface) creada para permitir el uso de cierto lenguaje de programación, o puede, también, incluir hardware sofisticado para comunicarse con un determinado sistema embebido. Las herramientas más comunes incluyen soporte para la detección de errores de programación como un entorno de desarrollo integrado o IDE (del inglés Integrated Development Environment) y otras utilidades. Los SDKs frecuentemente incluyen, también, códigos de ejemplo y notas técnicas de soporte u otra documentación de soporte para ayudar a clarificar ciertos puntos del material de referencia primario.

PHP SDK proporciona las siguientes capacidades:

- ✓ Una librería PHP que permite el acceso remoto al repositorio de Alfresco, Alfresco PHP a través de la API
- ✓ Integración de populares aplicaciones PHP. Por el momento sólo está disponible MediaWiki.
- ✓ Plantillas de PHP y capacidades de scripting en el repositorio utilizando la API de Alfresco PHP
- ✓ Soporte para ejecutar aplicaciones PHP en el repositorio de JVM, con acceso a la API de Alfresco PHP.

1.6.4 Entorno de Desarrollo Integrado

Un entorno de desarrollo integrado o, en inglés, Integrated Development Environment ('IDE'), es un programa compuesto por un conjunto de herramientas para un programador.
Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.
Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

Netbeans 6.5 para PHP (25)

NetBeans es una aplicación de código abierto ("open source") para desarrolladores. Posee todas las herramientas necesarias para crear aplicaciones Web, de escritorio y para teléfonos móviles con los lenguajes de programación Java, C, C++ e incluso lenguajes dinámicos como: PHP, JavaScript, Groovy, y Ruby. Es fácil de usar, instalar y puede ser ejecutado en múltiples plataformas como Windows, Linux, Mac OS X y Solaris. Entre sus principales características se pueden mencionar:

- ✓ Brinda completamiento automático de código PHP, así como coloreado de código sintáctico y semántico.
- ✓ Permite depurar el código usando Xdebug.

Zend Studio (22)

Zend Studio un completo entorno integrado de desarrollo para el lenguaje de programación PHP. Está escrito en Java, y está disponible para las plataformas Microsoft Windows, Mac OS X y GNU/Linux.

Junto con su contraparte Zend Platform, son la propuesta de Zend Technologies para el desarrollo de aplicaciones Web utilizando PHP, actuando Zend Studio como la parte cliente y Zend Platform como la parte servidora. Se trata en ambos casos de software comercial, lo cual contrasta con el hecho de que PHP es software libre.
Está disponible también una versión de Zend Studio para Eclipse.

Características:

- ✓ No requiere la instalación previa de PHP ni del entorno de ejecución de Java.
- ✓ Resaltado de sintaxis, autocompletado de código, ayuda de código y lista de parámetros de funciones y métodos de clase.
- ✓ Plegado de código (comentarios, bloques de phpDoc, cuerpo de funciones y métodos e implementación de clases).
- ✓ Inserción automática de paréntesis y corchetes de cierre.
- ✓ Sangrado automático y otras ayudas de formato de código.
- ✓ Detección de errores de sintaxis en tiempo real.
- ✓ Instalación de barras de herramientas para Internet Explorer y Mozilla Firefox (opcional).
- ✓ Soporte para gestión de grandes proyectos de desarrollo.
- ✓ Manual de PHP integrado.
- ✓ Soporte para control de versiones usando CVS o Subversion (a elección del desarrollador).
- ✓ Cliente FTP integrado.

Dadas las características, ventajas y soporte que brinda este potente entorno, se eligió como IDE a utilizar para el desarrollo del sistema.

1.6.5 Herramienta Case. Visual Paradigm (26)

Visual Paradigm es una herramienta CASE que utiliza UML como lenguaje de modelado. Está diseñada para una amplia gama de usuarios interesados en construir sistemas de software fiables con el uso del paradigma orientado a objetos, incluyendo actividades como ingeniería de software, análisis de sistemas y análisis de negocios.

Visual Paradigm es una herramienta que emplea las últimas notaciones del Lenguaje Unificado de Modelado, ingeniería inversa, generación del código, importación de Rational Rose, exportación/importación XML, generador de impresos, integración con el Visio. Además de lo anterior, soporta aplicaciones Web, genera código para el lenguaje Java y exporta en formato HTML, es una tecnología libre y está disponible en varios idiomas, en conjunto con esto, es fácil de instalar y fácil de actualizar. Por último, Visual Paradigm admite compatibilidad con las demás versiones. De manera general, esta herramienta de modelado ofrece:

- ✓ Entorno de creación de diagramas para UML 2.0
- ✓ Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- ✓ Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- ✓ Disponibilidad de múltiples versiones, para cada necesidad.
- ✓ Disponibilidad en múltiples plataformas.

1.6.6 Contenedor Web. Apache Tomcat (27)

Apache Tomcat es un contenedor de Servlet usado en la implementación de referencia oficial para las tecnologías Java Servlet y JavaServer Pages. Es desarrollado en un ambiente participativo y abierto. La versión 5.x es implementada a partir de las especificaciones Servlet 2.4 y JSP 2.0 y cuenta con un mecanismo de recolección de basura perfeccionado, basado en su reducción. Posee una capa envolvente nativa para Windows y Unix para la integración de las plataformas. Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

CAPITULO 2. DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

En este capítulo se presenta la concepción de la arquitectura base del sistema. Luego se le hacen críticas al diseño propuesto por el analista teniendo en cuenta algunos cambios de herramientas y tecnologías abordadas ya en el Capítulo 1. Además se explican los estándares de codificación usados en la implementación. Seguidamente se tratan más a fondo componentes e implementaciones que se usaron en el desarrollo de la aplicación. Y además se detallan las nuevas clases, necesarias para la implementación de la solución informática.

2.1 Arquitectura Base

En el campo del software, la arquitectura identifica los elementos más importantes de un sistema así como sus relaciones. Da una visión global del sistema.

¿Por qué es esto importante?

Porque se necesita una arquitectura para entender el sistema, organizar su desarrollo, plantear la reutilización del software y hacerlo evolucionar.

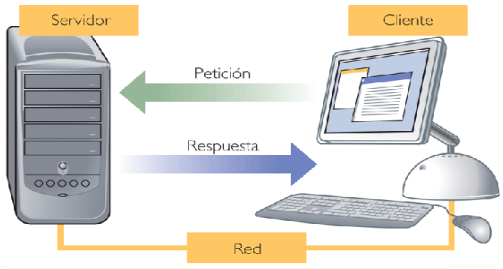
Como determinar los elementos que definen una arquitectura es difícil y muy importante. Generalmente las metodologías de desarrollo indican principios para identificar y diseñar una arquitectura, aunque por ahora la ayuda real que ofrecen es muy limitada al basarse en principios muy genéricos.

A continuación se ilustra la arquitectura base definida para la implementación del Módulo Recursos del SCD.

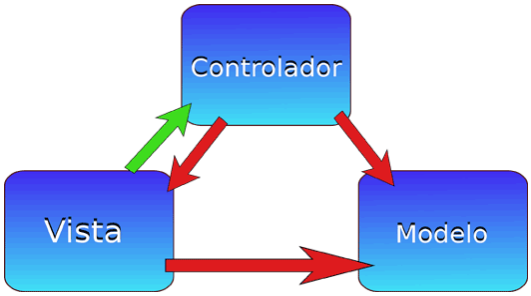
2.1.1 Patrones arquitectónicos utilizados

Patrones, sobre aspectos fundamentales de la estructura de un sistema software. Estos especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes.

Como se ha abordado en el Capítulo 1, se hizo uso de los patrones arquitectónicos Cliente-Servidor y MVC.



Cliente-Servidor



MVC

2.2 Valoración crítica del diseño propuesto por el analista

En el diseño propuesto por los analistas se definieron las clases fundamentales para que el sistema funcione satisfactoriamente, así como los atributos y métodos que deben tener las mismas, dando una idea clara de lo que se debe implementar. No obstante se determinó redefinir algunos diagramas de paquetes de clases de estereotipos Web, atendiendo a cambios surgidos, con el objetivo de lograr una implementación satisfactoria, que cuente con las herramientas y componentes más actualizados. Para lograr este objetivo se usaron en la implementación por citar los ejemplos claves, Alfresco y CodeIgniter.

Entre los cambios más notables se encuentra la sustitución del paquete Acceso a Datos, que tenía como principal objetivo el uso de funciones almacenadas para la inserción, modificación y eliminación de los registros en la base de datos.

Una vez determinado Alfresco como manejador del contenido, se desecha el paquete Acceso a Datos y en su lugar surge uno llamado Servicios_Alfresco, con similares objetivos, pero ya brindando un conjunto de servicios que permiten consultar, crear, modificar y eliminar el contenido en su propio repositorio. La lógica sigue siendo la misma, con el considerable ahorro de esfuerzo y tiempo al no tener que crear ni validar las consultas a la Base de Datos, sino utilizar el motor de búsquedas Lucene que trae consigo Alfresco para el Acceso a Datos dentro de su repositorio.

Otro cambio muy notable es que se implementa el patrón MVC, o sea que las páginas clientes que antes se generaban por una página servidora que tenía cada una de estas, ahora se generan todas a partir de una sola, que a su vez hace fluir la información desde y hacia la clase controladora, además las clases quedan mejor organizadas, o sea, se separan, las clases Modelos, las Vistas y las Controladoras.

Además se elimina el paquete Smarty que definía un conjunto de clases de interfaz en la capa de

presentación, con el objeto de lograr que a la hora de implementar el sistema fuese más fácil el trabajo con el diseño. Este no es necesario teniendo en cuenta que el framework CodeIgniter maneja las clases interfaces que son usadas en el actual sistema a partir de las clases controladoras.

A continuación se muestran diagramas de paquetes de clases propuestos por los analistas y le suceden los criticados como parte del proceso de implementación. Se seleccionaron los diagramas más generales para mostrar las críticas realizadas, aunque todos los demás fueron de igual manera criticados y actualizados previamente.

Diagrama de clases propuesto por el analista. Gestionar Tipo de Solicitud

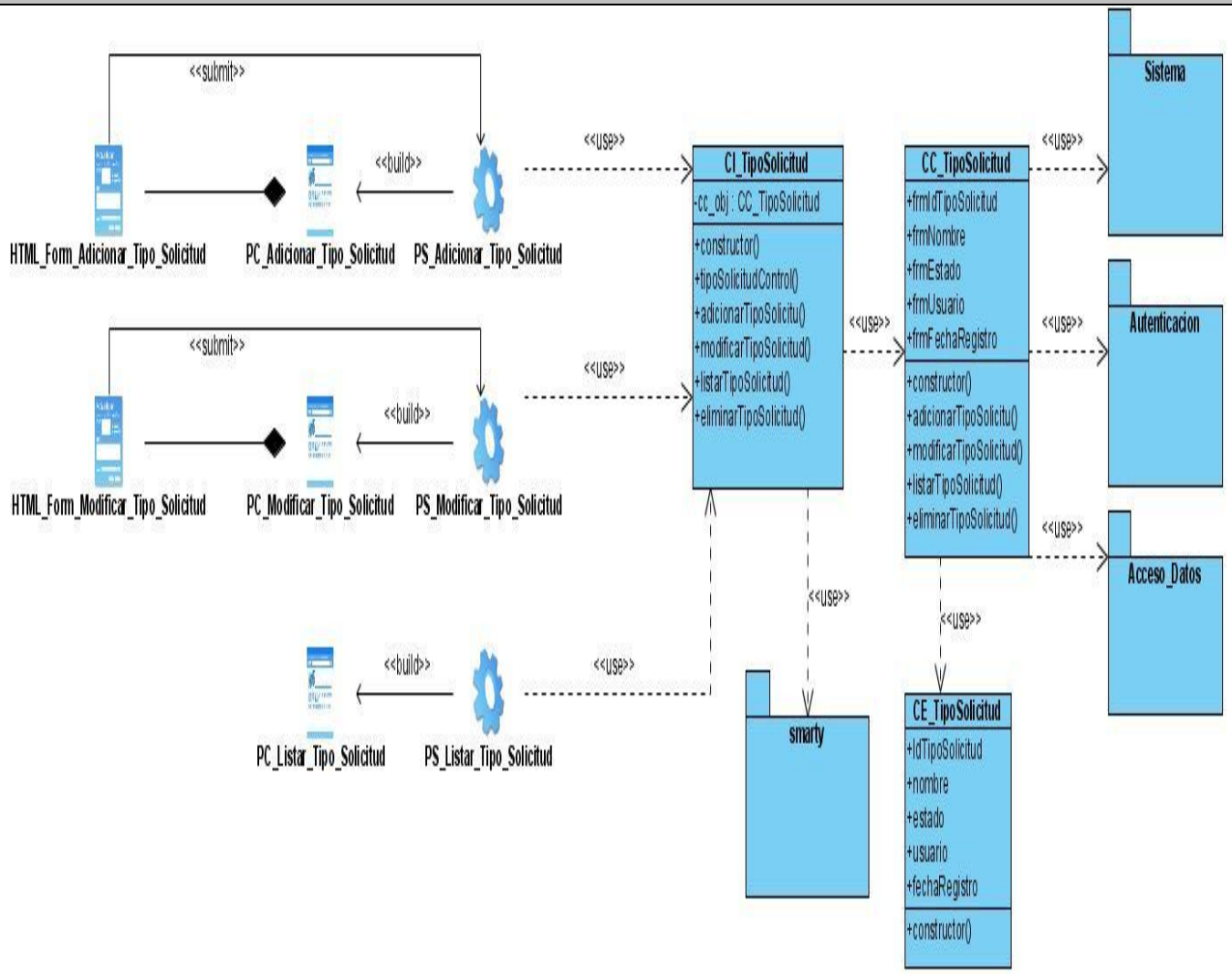


Diagrama de clases propuesto por el analista. Gestionar Usuario

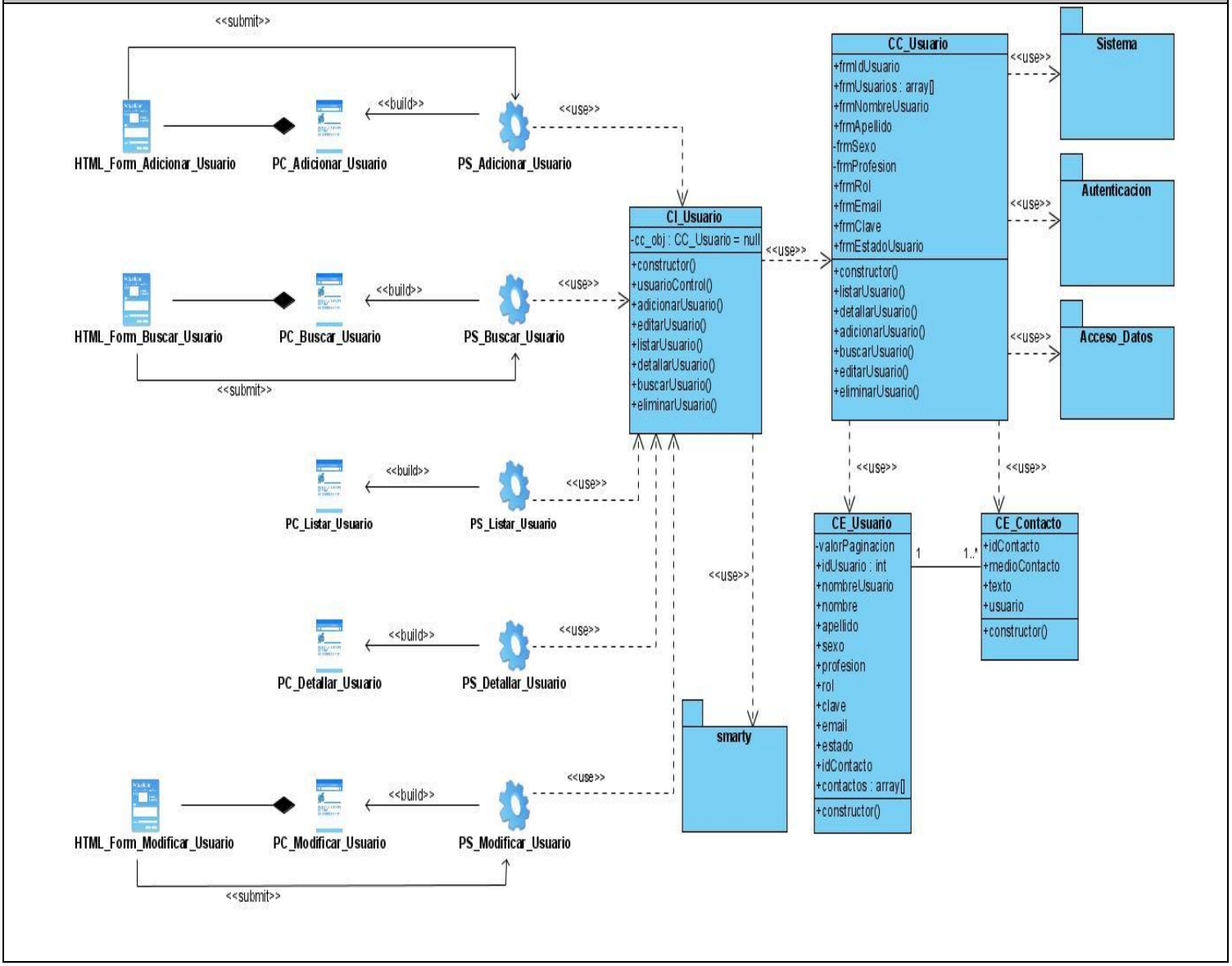


Diagrama de clases. Gestionar Tipo de Solicitud(versión final)

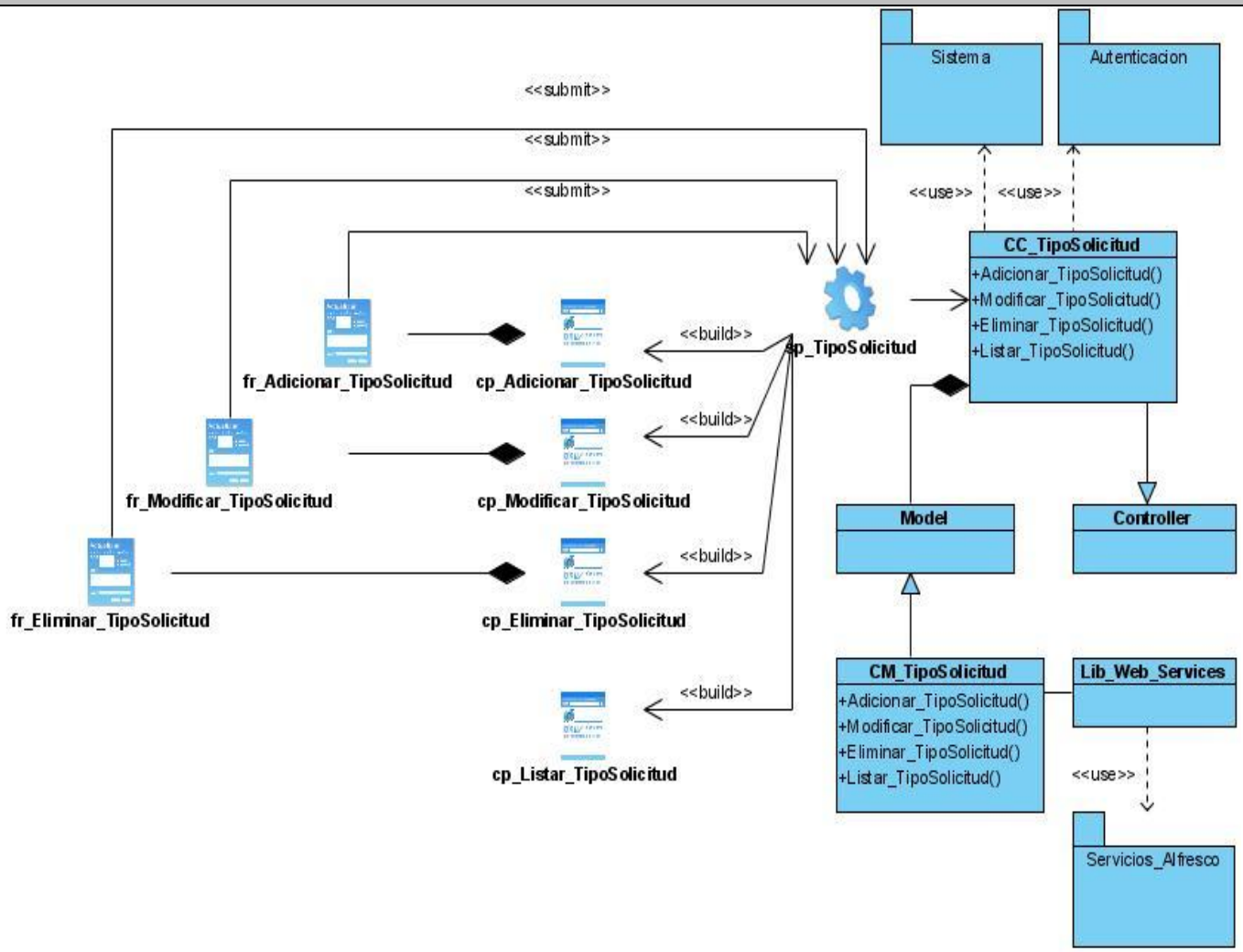
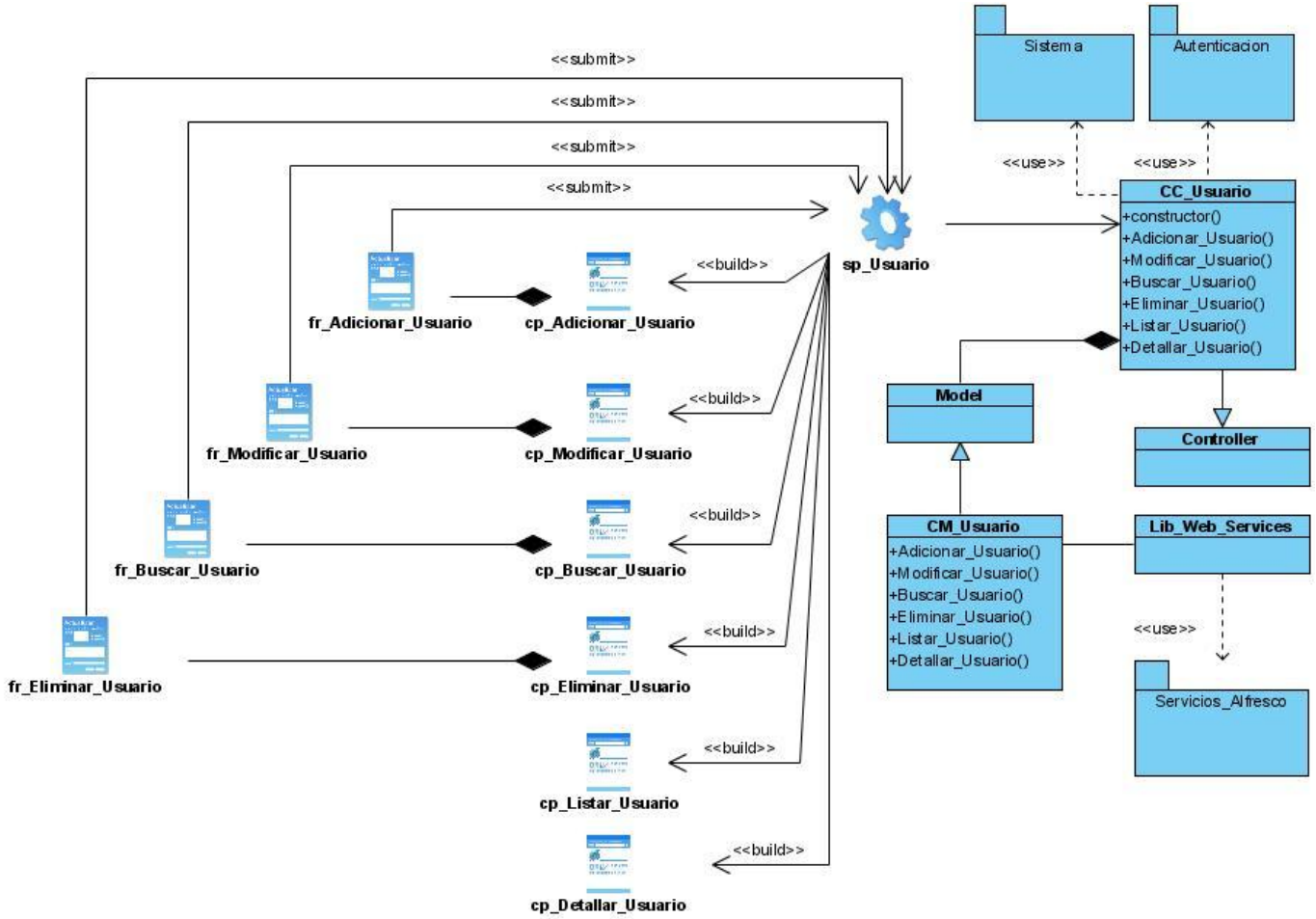


Diagrama de clases. Gestionar Usuario(versión final)



2.3 Estándares de codificación utilizados

Se definen a continuación los estándares de codificación usados para el desarrollo de la solución. Gran parte de estas reglas están basadas en las guías de estilo de grandes proyectos libres, como phpBB[1]. El uso de estándares tiene innumerables ventajas, entre ellas:

- ✓ Asegurar la legibilidad del código entre distintos programadores, facilitando la depuración del mismo.
- ✓ Proveer una guía para el encargado de mantenimiento/actualización del sistema, con código claro y bien documentado.
- ✓ Facilitar la portabilidad entre plataformas y aplicaciones.
- ✓ Permite una mejor organización y productividad en la programación de proyectos en equipos.

Variables locales

- ✓ Los nombres de algunas variables locales, como los iteradores o los contadores, pueden especificarse en minúscula y de forma abreviada, siempre que su contexto sea específicamente local y su lectura sea intuitiva. Ejemplos: $cont, $i, $j.
- ✓ Al hacer asignaciones, debe existir un espacio a ambos lados del signo igual (=), esto funciona tanto para asignar un valor fijo, de otra variable o del resultado de una función.
- ✓ En el caso de un bloque de asignaciones relacionadas entre sí (por ejemplo, al inicializar un script), se pueden alinear los signos (=) agregando espacios, para mejorar la legibilidad.

Estructuras de control

- ✓ Incluye if, for, while, switch, etc. Deben tener un espacio entre la palabra clave y el paréntesis de apertura, para diferenciarlos de las llamadas a funciones. Se recomienda, aunque no sea necesario, la utilización de llaves. Esto mejora la legibilidad y disminuye la posibilidad de errores lógicos al agregar nuevas líneas de código.

[1] Sistema de foros gratuito basado en un conjunto de paquetes de código programados en el popular lenguaje de programación web PHP y lanzado bajo la Licencia pública general de GNU, cuya intención es la de proporcionar fácilmente, y con amplia posibilidad de personalización, una herramienta para crear comunidades. Su nombre es por la abreviación de PHP Bulletin Board.

Llamadas a funciones

- ✓ Las funciones deben ser llamadas sin espacio entre el nombre de la función, el paréntesis de apertura y el primer parámetro. En caso de varios parámetros, separar con espacios entre la coma y cada parámetro, y sin espacios entre el último parámetro, el paréntesis de cierre y el punto y coma.

Definición de funciones

- ✓ Todas las funciones deben tener un comentario, antes de su declaración, explicando el objetivo de su implementación.
- ✓ Las definiciones de funciones utilizan el estilo BSD/Allman. Las características más importantes se resaltan a continuación:
    - El nombre debe ser lo más descriptivo posible.
    - Se debe evitar el uso de abreviaturas.
    - Se debe utilizar la convención lowerCamelCase.
- ✓ Colocar los argumentos con valores por defecto, al final de la lista.
- ✓ Siempre intentar retornar un valor significativo.
- ✓ La llave de inicio de la función se coloca en la misma línea de declaración, luego del paréntesis de cierre de los parámetros. (28)

Definición de clases

- ✓ El nombre debe ser descriptivo, evitando abreviaturas, usando la convención UpperCamelCase.
- ✓ La llave de inicio de la clase se coloca en la línea siguiente, indentada correctamente.
- ✓ Todos los miembros de la clase deben ser privados, es decir, únicamente accesibles a través de métodos de la misma. (28)

Comentarios

- ✓ Se aconseja el uso de comentarios en línea para facilitar la comprensión del código, sobre todo en procedimientos complejos. Los comentarios pueden ser con fin documental o bien como 'ayuda-memoria'.
- ✓ Se recomienda utilizar los estilos de C (/* */) y C++ (//), no tanto así el signo numeral o sharp (#).

Etiquetas de bloque PHP

- ✓ Siempre utilizar <?php y ?> para iniciar y terminar un bloque de código PHP, no las variantes <? y ?> o <% y %>. Esto asegura compatibilidad entre diversas configuraciones de equipos.

Formato para guardar archivos con extensión ".php"

- ✓ Los archivos con código PHP, deben de ser guardados en formato ASCII utilizando la codificación ISO-8859-1. El formato ASCII con codificación ISO-8859-1 es el formato en que se guardan los archivos de texto plano (".txt"). La razón de este estándar es que determinados editores HTML agregan códigos de carácter extraño de salto de línea (como si se tratara de un archivo binario) y esto puede ocasionar que el intérprete de PHP encuentre problemas a la hora de leer el script.

2.4 Análisis de posibles implementaciones, componentes o módulos ya existentes y que puedan ser rehusados

2.4.1 Framework PHP. CodeIgniter.

Anteriormente se dijo que por sus características y ventajas, se decidió emplear, para la implementación del sistema, el framework de PHP CodeIgniter. Entre las características mencionadas está, que implementa el patrón arquitectónico Modelo-Vista-Controlador (MVC) esto permite una buena separación entre lógica y presentación. CodeIgniter tiene un enfoque bastante flexible del MVC. CodeIgniter también permite incorporar códigos existentes, o incluso desarrollar librerías de núcleo para el sistema.

En el siguiente diagrama se muestra el flujo de datos a través del sistema.

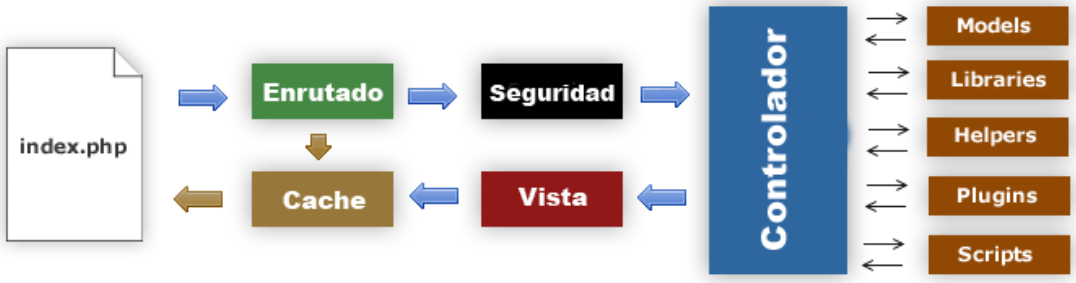


Diagrama de Flujo del framework Codelgniter implementando MVC

1. El index.php, inicializa el núcleo de la aplicación.
2. El enrutador, examina la petición HTTP y, ayuda a determinar lo que se debe hacer.
3. Si existe, la cache, devuelve el archivo HTML sin necesidad de pasar por el sistema, ahorrándose la carga que esto conlleva.
4. Seguridad. Antes que el controlador sea cargado, la petición HTTP y cualquier dato suministrado por el usuario es filtrado por seguridad.
5. El controlador, carga el modelo, librerías, helpers, plugins y todos los demás recursos necesarios para satisfacer la petición.
6. Una vez renderizada la Vista, esta es enviada al navegador, entonces si la cache se encuentra habilitada, se almacena el resultado para la próxima ocasión que la URL sea servida.

2.4.2 Web Services.

En el capítulo anterior se hizo alusión a los Web Service, los cuales se usaron para la implementación del sistema, por lo mencionado anteriormente y por estar bien definida la implementación de muchos de estos, o al menos los que por su uso cotidiano se han vuelto estándares casi para la totalidad de las aplicaciones Web que hacen uso de Alfresco de conjunto con los Web Services.

Es importante comentar la amplia y actualizada documentación sobre Web Service en la propia Wikipedia

de Alfresco, permitiendo esto, su mayor comprensión, y facilitando a la vez, su implementación.

En la tabla a continuación se muestran varios Web Services ofrecidos por Alfresco actualmente. Los cuales se han usado en el desarrollo de la aplicación.

Web Service	Descripción	Estado
Autenticación	acceso y salida	Disponible
Repositorio	modelo de consulta y manipulación	Disponible
Contenido	manipulación de contenidos	Disponible
Autoría	colaborativo de creación de contenidos	Disponible
Clasificación	aplicar las clasificaciones y categorías	Disponible
Control de Acceso	roles, los permisos y la propiedad	Disponible
Acción	gestión de acciones y normas	Disponible
Administración	administración de usuarios	Disponible

2.5 Descripción de los algoritmos no triviales a implementar. Análisis de complejidad de los mismos

En este epígrafe se describen algoritmos que tienen una complejidad superior a la media, y a percepción de los implementadores, su explicación se hace necesaria para la mejor comprensión del código y constituye un aporte a futuras implementaciones de sistemas similares.

2.5.1 Modificar Organismo.

Este se encarga de modificar el o los atributos de un determinado Organismo.

En un primer paso se obtiene el identificador del organismo a modificar. Luego dado el identificador, se obtiene el elemento a modificar. Seguidamente se realiza una verificación, para saber si realmente el

usuario desea modificar los datos de un Organismo y a su vez se comprueba que todos los campos que se desean modificar contengan información, entiéndase; Nombre, Abreviatura del Organismo, Ubicación Física, Estado (Activo o Inactivo).

Transcurridos los pasos anteriores se obtienen en variables el o los datos del Organismo a modificar, mediante el envío de la información contenida en los formularios y luego se le asignan a las propiedades del elemento que se está modificando los valores contenidos en estas variables, por último se guardan dichas propiedades con el valor final enviado.

2.5.2 Búsqueda Avanzada para Tipo de Información

Consiste en la realización de una búsqueda de todos los elementos (Tipo de Información) que en su nombre o descripción contengan la palabra especificada.

Para realizar dicha búsqueda, se verifica que se haya presionado el botón ¨Buscar¨ en la aplicación. Luego se obtienen los valores del formulario. Seguidamente se realiza una consulta base al espacio de trabajo, Tipo de Información, para concatenar los valores consultados con los valores entrados por el usuario, que fueron previamente validados.

Una vez concatenados los valores, se procede a realizar la búsqueda de estos, mediante una consulta directa al repositorio de Alfresco. Esta brinda como resultado los elementos coincidentes.

Esta búsqueda se realiza haciendo uso de Lucene[2].

Estos algoritmos representan en los casos explicados un método específico, pero la modificación y búsqueda avanzada de Recursos se realizan en todos los casos de manera muy similar.

Los métodos anteriormente descritos forman parte de clases que se hicieron necesarias para la implementación, a continuación la descripción de estas.

[2] API para la recuperación de información de código abierto, esta es útil para cualquier aplicación que requiera indexado y búsqueda a texto completo.

2.6 Descripción de las nuevas clases u operaciones necesarias

2.6.1 Clases Controladoras

Nombre: Auth	
Tipo de clase: Controladora	
Atributos:	**Tipo:**
-	-
Funciones:	
Nombre:	index()
Descripción:	Verifica si el usuario está autenticado, en caso de estarlo lo redirecciona a Content, en caso contrario a la vista de autenticación.
Nombre:	login
Descripción:	Permite acceder al sistema.
Nombre:	logout
Descripción:	Permite salir de manera segura del sistema.

Nombre: Repository Access	
Tipo de clase: Controladora	
Clase paquetes de librerías	
Atributos:	**Tipo:**

_repository _ci	repository content
Nombre:	connect($usser,$paswrd)
Descripción:	Para conectar un usuario al sistema.
Nombre:	Disconnect($ticket)
Descripción:	Para desconectar un usuario del sistema.
Nombre:	getRepository()
Descripción:	Obtener un objeto de tipo Repository.

Nombre: Access	
Tipo de clase: Controladora	
Clase paquetes de librerías	
Atributos: _repository _ticket _access_control _ci	**Tipo:** repository string access content
Nombre:	getPermissions($node)
Descripción:	Obtener todos los permisos de un nodo.

Nombre:	hasPermissions($node,$permissions)
Descripción:	Permite darle un permiso determinado a un nodo.
Nombre:	getAlfPermissions($node)
Descripción:	Método que obtiene todos los permisos de un nodo de Alfresco.
Nombre:	hasAlfPermissions($node,$permissions)
Descripción:	Método que provee los permisos determinados a un nodo de Alfresco.
Nombre:	getOwners($node)
Descripción:	Obtener las propiedades de un nodo.
Nombre:	setOwners($node,$owners)
Descripción:	Cambiar las propiedades de un nodo.

Nombre: Descripcion	
Tipo de clase: Controladora	
Atributos:	**Tipo:**
-	-
Funciones:	
Nombre:	Index()
Descripción:	Se encarga de la navegación dentro del repositorio de contenidos Alfresco.
Nombre:	descripcionModuloPanelControl

Descripción:	Muestra el menú de acceso a cada uno de los módulos del sistema.
Nombre:	descripcionModuloRecursos
Descripción:	Muestra el menú de acceso a cada uno de los Recursos del sistema.
Nombre:	trabajadores
Descripción:	Muestra el menú de acceso a Usuarios y Trabajadores Externos.

Nombre: TipoAccion	
Tipo de clase: Controladora	
Atributos:	**Tipo:**
-	-
Funciones:	
Nombre:	Index()
Descripción:	Se encarga de la navegación dentro del repositorio de contenidos Alfresco.
Nombre:	tipoaccion
Descripción:	Constructor de la clase.
Nombre:	nuevoTipoAccion
Descripción:	Permite adicionar un nuevo tipo de acción.
Nombre:	modificarTipoAccion
Descripción:	Permite modificar un tipo de acción existente.

Nombre:	buscarTipoAccion
Descripción:	Permite realiza una búsqueda avanzada sobre los tipos de acción.
Nombre:	listarTipoAccion
Descripción:	Permite mostrar un listado de tipos de acción.
Nombre:	eliminarTipoAccion
Descripción:	Permite eliminar un tipo de acción existente.

Nombre: TipoInformacion	
Tipo de clase: Controladora	
Atributos:	**Tipo:**
-	-
Funciones:	
Nombre:	Index()
Descripción:	Se encarga de la navegación dentro del repositorio de contenidos Alfresco.
Nombre:	nuevoTipoInformacion
Descripción:	Permite adicionar un nuevo tipo de información.
Nombre:	modificarTipoInformacion
Descripción:	Permite modificar un tipo de información existente.
Nombre:	buscarTipoInformacion

Descripción:	Permite realizar una búsqueda avanzada sobre los tipos de información.
Nombre:	listarTipoInformacion
Descripción:	Permite mostrar un listado de tipos de información.
Nombre:	eliminarTipoInformacion
Descripción:	Permite eliminar un tipo de información existente.

Nombre: TipoOficina	
Tipo de clase: Controladora	
Atributos:	**Tipo:**
-	-
Funciones:	
Nombre:	Index()
Descripción:	Se encarga de la navegación dentro del repositorio de contenidos Alfresco.
Nombre:	nuevoTipoOficina
Descripción:	Permite adicionar un nuevo tipo de oficina.
Nombre:	modificarTipoOficina
Descripción:	Permite modificar un tipo de oficina existente.
Nombre:	buscarTipoOficina
Descripción:	Permite realizar una búsqueda avanzada sobre los tipos de oficina.

Nombre:	listarTipoOficina
Descripción:	Permite mostrar un listado de tipos de oficina.
Nombre:	eliminarTipoOficina
Descripción:	Permite eliminar un tipo de oficina existente.

Nombre: TipoSolicitud	
Tipo de clase: Controladora	
Atributos:	**Tipo:**
-	-
Funciones:	
Nombre:	Index()
Descripción:	Se encarga de la navegación dentro del repositorio de contenidos Alfresco.
Nombre:	nuevoTipoSolicitud
Descripción:	Permite adicionar un nuevo tipo de solicitud.
Nombre:	modificarTipoSolicitud
Descripción:	Permite modificar un tipo de solicitud existente.
Nombre:	buscarTipoSolicitud
Descripción:	Permite realizar una búsqueda avanzada sobre los tipos de solicitud.

Nombre:	listarTipoSolicitud
Descripción:	Permite mostrar un listado de tipos de solicitud.
Nombre:	eliminarTipoSolicitud
Descripción:	Permite eliminar un tipo de solicitud existente.

Nombre: Organismo	
Tipo de clase: Controladora	
Atributos:	**Tipo:**
-	-
Funciones:	
Nombre:	Index()
Descripción:	Se encarga de la navegación dentro del repositorio de contenidos Alfresco.
Nombre:	nuevoOrganismo
Descripción:	Permite adicionar un nuevo organismo.
Nombre:	modificarOrganismo
Descripción:	Permite modificar un organismo existente.
Nombre:	buscarOrganismo
Descripción:	Permite realizar una búsqueda avanzada sobre los organismos existentes.
Nombre:	listarOrganismo

Descripción:	Permite mostrar un listado de organismos.
Nombre:	eliminarOrganismo
Descripción:	Permite eliminar un organismo existente.

Nombre: Usuario	
Tipo de clase: Controladora	
Atributos:	**Tipo:**
-	-
Funciones:	
Nombre:	Index()
Descripción:	Se encarga de la navegación dentro del repositorio de contenidos Alfresco.
Nombre:	nuevoUsuario
Descripción:	Permite adicionar un nuevo usuario.
Nombre:	modificarUsuario
Descripción:	Permite modificar un usuario existente.
Nombre:	buscarUsuario
Descripción:	Permite realizar una búsqueda avanzada sobre los usuarios existentes.
Nombre:	listarUsuario
Descripción:	Permite mostrar un listado de usuarios.

Nombre:	eliminarUsuario
Descripción:	Permite eliminar un usuario existente.

Nombre: TrabajadorExterno	
Tipo de clase: Controladora	
Atributos:	**Tipo:**
-	-
Funciones:	
Nombre:	Index()
Descripción:	Se encarga de la navegación dentro del repositorio de contenidos Alfresco.
Nombre:	nuevoTrabajadorExterno
Descripción:	Permite adicionar un nuevo trabajador externo.
Nombre:	modificarTrabajadorExterno
Descripción:	Permite modificar un trabajador externo existente.
Nombre:	buscarTrabajadorExterno
Descripción:	Permite realizar una búsqueda avanzada sobre los trabajadores externos existentes.
Nombre:	listarTrabajadorExterno
Descripción:	Permite mostrar un listado de trabajadores externos.

Nombre:	eliminarTrabajadorExterno
Descripción:	Permite eliminar un trabajador externo existente.

2.6.2 Clases Modelos

Nombre: Descripcion	
Tipo de clase: Modelo	
Atributos:	**Tipo:**
-	-
Funciones:	
Nombre:	descripcionModuloPanelControl()
Descripción:	Retorna el menú de acceso a cada uno de los módulos del sistema.
Nombre:	descripcionModuloRecursos()
Descripción:	Retorna el menú de acceso a cada uno de los Recursos del sistema.
Nombre:	trabajadores()
Descripción:	Retorna el menú de acceso a Usuarios y Trabajadores Externos.

Nombre: TipoAccion	
Tipo de clase: Modelo	
Atributos:	**Tipo:**
-	-
Funciones:	
Nombre:	nuevoTipoAccion()

Descripción:	Adiciona un nuevo tipo de acción.
Nombre:	modificarTipoAccion()
Descripción:	Modifica un tipo de acción existente.
Nombre:	buscarTipoAccion()
Descripción:	Permite realizar una búsqueda avanzada sobre los tipos de acción.
Nombre:	listarTipoAccion()
Descripción:	Retorna un listado de tipos de acción.
Nombre:	eliminarTipoAccion()
Descripción:	Elimina un tipo de acción existente.

Nombre: TipoInformacion	
Tipo de clase: Modelo	
Atributos:	**Tipo:**
-	-
Funciones:	
Nombre:	nuevoTipoInformacion()
Descripción:	Adiciona un nuevo tipo de información.
Nombre:	modificarTipoInformacion ()
Descripción:	Modifica un tipo de información existente.
Nombre:	buscarTipoInformacion ()

Descripción:	Realiza una búsqueda avanzada sobre los tipos de información.
Nombre:	listarTipoInformacion ()
Descripción:	Retorna un listado de tipos de información.
Nombre:	eliminarTipoInformacion ()
Descripción:	Elimina un tipo de información existente.

Nombre: TipoOficina	
Tipo de clase: Modelo	
Atributos:	**Tipo:**
-	-
Funciones:	
Nombre:	nuevoTipoOficina()
Descripción:	Adiciona un nuevo tipo de oficina.
Nombre:	modificarTipoOficina()
Descripción:	Modifica un tipo de oficina existente.
Nombre:	buscarTipoOficina()
Descripción:	Realiza una búsqueda avanzada sobre los tipos de oficina.
Nombre:	listarTipoOficina ()
Descripción:	Retorna un listado de tipos de oficina.
Nombre:	eliminarTipoOficina()

Descripción:	Elimina un tipo de oficina existente.

Nombre: TipoSolicitud	
Tipo de clase: Modelo	
Atributos:	**Tipo:**
-	-
Funciones:	
Nombre:	nuevoTipoSolicitud()
Descripción:	Adiciona un nuevo tipo de solicitud.
Nombre:	modificarTipoSolicitud()
Descripción:	Modifica un tipo de solicitud existente.
Nombre:	buscarTipoSolicitud()
Descripción:	Realiza una búsqueda avanzada sobre los tipos de solicitud.
Nombre:	listarTipoSolicitud()
Descripción:	Retorna un listado de tipos de solicitud.
Nombre:	eliminarTipoSolicitud()
Descripción:	Elimina un tipo de solicitud existente.

Nombre: Organismo	
Tipo de clase: Modelo	
Atributos:	**Tipo:**
-	-

Funciones:	
Nombre:	nuevoOrganismo()
Descripción:	Adiciona un nuevo organismo.
Nombre:	modificarOrganismo()
Descripción:	Modifica un organismo existente.
Nombre:	buscarOrganismo()
Descripción:	Realiza una búsqueda avanzada sobre los organismos existentes.
Nombre:	listarOrganismo()
Descripción:	Retorna un listado de organismos.
Nombre:	eliminarOrganismo()
Descripción:	Elimina un organismo existente.

Nombre: Usuario	
Tipo de clase: Modelo	
Atributos:	**Tipo:**
-	-
Funciones:	
Nombre:	nuevoUsuario()
Descripción:	Adiciona un nuevo usuario.
Nombre:	modificarUsuario()
Descripción:	Modifica un usuario existente.

Nombre:	buscarUsuario()
Descripción:	Realiza una búsqueda avanzada sobre los usuarios existentes.
Nombre:	listarUsuario()
Descripción:	Retorna un listado de usuarios.
Nombre:	eliminarUsuario()
Descripción:	Elimina un usuario existente.

Nombre: TrabajadorExterno	
Tipo de clase: Modelo	
Atributos:	**Tipo:**
-	-
Funciones:	
Nombre:	nuevoTrabajadorExterno()
Descripción:	Adiciona un nuevo trabajador externo.
Nombre:	modificarTrabajadorExterno()
Descripción:	Modifica un trabajador externo existente.
Nombre:	buscarTrabajadorExterno()
Descripción:	Realiza una búsqueda avanzada sobre los trabajadores externos existentes.
Nombre:	listarTrabajadorExterno()

Descripción:	Retorna un listado de trabajadores externos.
Nombre:	eliminarTrabajadorExterno()
Descripción:	Elimina un trabajador externo existente.

2.6.3 Clases Vistas

Nombre: Descripcion_Modulo_Recursos
Tipo de clase: Interfaz
Descripción General: Muestra la interfaz para seleccionar la entrada al Módulo Recursos mediante un menú que contiene los demás módulos.

Nombre: Trabajadores
Tipo de clase: Interfaz
Descripción General: Muestra la interfaz para seleccionar la entrada a los recursos Usuarios y Trabajadores Externos mediante un menú.

Nombre: Nuevo_TipoAccion
Tipo de clase: Interfaz
Descripción General: Muestra la interfaz para adicionar un nuevo tipo de acción.

Nombre: Nuevo_TipoInformacion
Tipo de clase: Interfaz
Descripción General: Muestra la interfaz para adicionar un nuevo tipo de información.

Nombre: Nuevo_TipoOficina
Tipo de clase: Interfaz
Descripción General: Muestra la interfaz para adicionar un nuevo tipo de oficina.

Nombre: Nuevo_TipoSolicitud
Tipo de clase: Interfaz
Descripción General: Muestra la interfaz para adicionar un nuevo tipo de solicitud.

Nombre: Nuevo_ Organismo
Tipo de clase: Interfaz
Descripción General: Muestra la interfaz para adicionar un nuevo organismo.

Nombre: Nuevo_Usuario
Tipo de clase: Interfaz
Descripción General: Muestra la interfaz para adicionar un nuevo usuario.

Nombre: Nuevo_TrabajadorExterno
Tipo de clase: Interfaz
Descripción General: Muestra la interfaz para adicionar un nuevo trabajador externo.

Nombre: Modificar_TipoAccion
Tipo de clase: Interfaz

Descripción General: Muestra la interfaz para modificar un nuevo tipo de acción.

Nombre: Modificar_TipoInformacion
Tipo de clase: Interfaz
Descripción General: Muestra la interfaz para modificar un nuevo tipo de información.

Nombre: Modificar _TipoOficina
Tipo de clase: Interfaz
Descripción General: Muestra la interfaz para modificar un nuevo tipo de oficina.

Nombre: Modificar _TipoSolicitud
Tipo de clase: Interfaz
Descripción General: Muestra la interfaz para modificar un nuevo tipo de solicitud.

Nombre: Modificar _ Organismo
Tipo de clase: Interfaz
Descripción General: Muestra la interfaz para modificar un nuevo organismo.

Nombre: Modificar _Usuario
Tipo de clase: Interfaz
Descripción General: Muestra la interfaz para modificar un nuevo usuario.

Nombre: Modificar _TrabajadorExterno
Tipo de clase: Interfaz
Descripción General: Muestra la interfaz para modificar un nuevo trabajador externo.

Nombre: Listar_TipoAccion
Tipo de clase: Interfaz
Descripción General: Muestra un listado de tipos de acción.

Nombre: Listar _TipoInformacion
Tipo de clase: Interfaz
Descripción General: Muestra un listado de tipos de información.

Nombre: Listar _TipoOficina
Tipo de clase: Interfaz
Descripción General: Muestra un listado de tipos de oficina.

Nombre: Listar _TipoSolicitud
Tipo de clase: Interfaz
Descripción General: Muestra un listado de tipos de solicitud.

Nombre: Listar _ Organismo
Tipo de clase: Interfaz

Descripción General: Muestra un listado de organismos.

Nombre: Listar _Usuario
Tipo de clase: Interfaz
Descripción General: Muestra un listado de usuarios.

Nombre: Listar _TrabajadorExterno
Tipo de clase: Interfaz
Descripción General: Muestra un listado de trabajadores externos.

Nombre: Buscar_TipoAccion
Tipo de clase: Interfaz
Descripción General: Muestra la interfaz para realizar una búsqueda avanzada sobre el recurso tipo de acción.

Nombre: Buscar _TipoInformacion
Tipo de clase: Interfaz
Descripción General: Muestra la interfaz para realizar una búsqueda avanzada sobre el recurso tipo de información.

Nombre: Buscar _TipoOficina
Tipo de clase: Interfaz

Descripción General: Muestra la interfaz para realizar una búsqueda avanzada sobre el recurso tipo de oficina.

Nombre: Buscar _TipoSolicitud
Tipo de clase: Interfaz
Descripción General: Muestra la interfaz para realizar una búsqueda avanzada sobre el recurso tipo de solicitud.

Nombre: Buscar_ Organismo
Tipo de clase: Interfaz
Descripción General: Muestra la interfaz para realizar una búsqueda avanzada sobre el recurso organismo.

Nombre: Buscar _Usuario
Tipo de clase: Interfaz
Descripción General: Muestra la interfaz para realizar una búsqueda avanzada sobre el recurso usuario.

Nombre: Buscar_TrabajadorExterno
Tipo de clase: Interfaz
Descripción General: Muestra la interfaz para realizar una búsqueda avanzada sobre el recurso trabajador externo.

CAPITULO 3. VALIDACION DE LA SOLUCION PROPUESTA

Toda solución informática es propensa a tener errores una vez implementada e incluso durante su desarrollo. Para la detección de estos se realizan pruebas de software. Las pruebas de software son los procesos que permiten verificar y revelar la calidad de un producto software. Para determinar el nivel de calidad se deben efectuar pruebas que permitan comprobar el grado de cumplimiento respecto a las especificaciones iniciales del sistema.

Hay muchos planteamientos a la hora de abordar el proceso de pruebas de software, pero para verificar productos complejos de forma efectiva, se requiere de un proceso de investigación, más que seguir un procedimiento al pie de la letra.

En general, los informáticos distinguen entre errores de programación (o "bugs") y defectos de forma. En un defecto de forma, el programa no realiza lo que el usuario espera. Por el contrario, un error de programación puede describirse como un fallo en la semántica de un programa de ordenador. Éste podría presentarse, o no, como un defecto de forma si se llegan a dar ciertas condiciones de cálculo.

Una práctica común es que el proceso de pruebas de un programa sea realizado por un grupo independiente de "testers"[3] al finalizar su desarrollo.

Otra práctica es que el proceso de pruebas se realice desde el mismo momento en que comienza el desarrollo y continúe hasta que finaliza.

A la aplicación en cuestión se le realizarán pruebas de unidad. Estas se implementan contra los elementos más pequeños que se pueden probar (unidades) del software, e implica probar la estructura interna como: los flujos lógicos y de datos, y los comportamientos de función y observables de la unidad. Las pruebas de diseño y de implementación que se centran en la estructura interna de una unidad se basan en el conocimiento de la implementación de la unidad, enfoque de caja blanca. El diseño y la implementación de pruebas para verificar los comportamientos observables de una unidad y sus funciones no se basan en el conocimiento de la implementación y por lo tanto se conocen como enfoque de caja negra.

Ambos enfoques se utilizan para diseñar e implementar diferentes tipos de pruebas necesarios para probar las unidades satisfactoriamente y completamente.

Una vez descritas las pruebas de unidad se hará énfasis en las pruebas de caja negra, que precisamente

[3] Grupo de desarrolladores que realiza pruebas al software.

serán las aplicadas a la solución definitiva del sistema aquí tratado.

3.1 Pruebas de Caja Negra

El objetivo de una prueba de caja negra es verificar la función y el comportamiento observable especificado de la unidad sin conocer cómo implementa la función y el comportamiento. Las pruebas de caja negra se centran y se basan en la entrada y la salida de la unidad.

Derivar las pruebas de la unidad basándose en el enfoque de caja negra, utiliza los argumentos de entrada y de salida de las operaciones de la unidad, y/o el estado de salida para su evaluación. Por ejemplo, la operación puede incluir un algoritmo (que requiere dos valores como entrada y devolución de una tercera como salida), o iniciar el cambio en el estado de un objeto o de un componente, como añadir o suprimir un registro de base de datos. Deben probarse totalmente. Para probar una operación, debe derivar suficientes guiones de prueba para verificar lo siguiente:

- ✓ la operación ha devuelto un valor apropiado para cada valor válido utilizado como entrada
- ✓ la operación ha devuelto un valor apropiado para cada valor no válido utilizado como entrada
- ✓ un estado de salida apropiado ocurre para cada estado de entrada válido
- ✓ un estado de salida apropiado ocurre para cada estado de entrada no válido.

Existen varios métodos de pruebas de caja negra, a continuación se describen los utilizados con más frecuencia.

3.1.1 Métodos de prueba basados en grafos

En este método se deben entender los objetos (objetos de datos, objetos de programa tales como módulos o colecciones de sentencias del lenguaje de programación) que se modelan en el software y las relaciones que conectan a estos objetos. Una vez que se ha llevado a cabo esto, el siguiente paso es definir una serie de pruebas que verifiquen que todos los objetos tienen entre ellos las relaciones esperadas. En este método:

- ✓ Se crea un grafo de objetos y sus relaciones.
- ✓ Se diseña una serie de pruebas que cubran el grafo de manera que se ejerciten todos los objetos y sus relaciones para descubrir errores.

Beizer, en su libro Black-Box Testing, describe un número de modelados para pruebas de comportamiento que pueden hacer uso de los grafos:

- ✓ Modelado del flujo de transacción. Los nodos representan los pasos de alguna transacción, y los enlaces representan las conexiones lógicas entre los pasos.
- ✓ Modelado de estado finito. Los nodos representan diferentes estados del software observables por el usuario, y los enlaces representan las transiciones que ocurren para moverse de estado a estado.
- ✓ Modelado de flujo de datos. Los nodos objetos de datos y los enlaces son las transformaciones que ocurren para convertir un objeto de datos en otro.
- ✓ Modelado de planificación. Los nodos son objetos de programa y los enlaces son las conexiones secuenciales entre esos objetos. Los pesos de enlace se usan para especificar los tiempos de ejecución requeridos al ejecutarse el programa.
- ✓ Gráfico Causa-efecto. La gráfica Causa-efecto representa una ayuda gráfica para seleccionar, de una manera sistemática, un gran conjunto de casos de prueba. Tiene un efecto secundario beneficioso en precisar estados incompletos y ambigüedades en la especificación.

  Un gráfico de Causa-efecto es un lenguaje formal al cual se traduce una especificación. Es realmente un circuito de lógica digital (una red combinatoria de lógica), pero en vez de la notación estándar de la electrónica, se utiliza una notación algo más simple. No hay necesitad de tener conocimiento de electrónica con excepción de una comprensión de la lógica boleana (entendiendo los operadores de la lógica *y, o,* y *no*).

3.1.2 Análisis de valores límites.

Los errores tienden a darse más en los límites del campo de entrada que en el centro. Por ello, se ha desarrollado el análisis de valores límites (AVL) como técnica de prueba. El análisis de valores límites lleva a una elección de casos de prueba que ejerciten los valores límites.

El análisis de valores límites es una técnica de diseño de casos de prueba que completa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el AVL lleva a la elección de casos de prueba en los extremos de la clase. En lugar de centrarse solamente en las condiciones de entrada, el AVL obtiene casos de prueba también para el campo de salida.

Condiciones sublímite.

Las condiciones límites normales son las más obvias de descubrir. Estas son definidas en la especificación o son evidentes al momento de utilizar el software. Algunos límites, sin embargo, son internos al software, no son necesariamente aparentes al usuario final pero aún así deben ser probadas por el probador. Estas son conocidas como condiciones sublímites o condiciones límites internas.

Y por último se hace alusión al método Partición Equivalente que será el aplicado al software en cuestión.

3.1.3 Partición equivalente.

Pressman, presenta la partición equivalente como un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

Una clase de equivalencia representa un conjunto de estados válidos o no válidos para condiciones de entrada. Típicamente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica.

El objetivo de partición equivalente es reducir el posible conjunto de casos de prueba en uno más pequeño, un conjunto manejable que evalúe bien el software. Se toma un riesgo porque se escoge no probar todo. Así que se necesita tener mucho cuidado al escoger las clases.

La partición equivalente es subjetiva. Dos probadores quienes prueban un programa complejo pueden llegar a diferentes conjuntos de particiones.

En el diseño de casos de prueba para partición equivalente se procede en dos:

- ✓ Se identifican las clases de equivalencia. Las clases de equivalencia son identificadas tomando cada condición de entrada (generalmente una oración o una frase en la especificación) y repartiéndola en dos o más grupos.

  Es de notar que dos tipos de clases de equivalencia están identificados: las clases de equivalencia válidas representan entradas válidas al programa, y las clases de equivalencia inválidas que

representan el resto de los estados posibles de la condición (es decir, valores erróneos de la entrada).

- ✓ Se definen los casos de prueba. El segundo paso es el uso de las clases de equivalencia para identificar los casos de prueba. El proceso es como sigue: se asigna un número único a cada clase de equivalencia. Hasta que todas las clases de equivalencia válidas han sido cubiertas por los casos de prueba, se escribe un nuevo caso de prueba que cubra la clase de equivalencia válida. Y por último hasta que los casos de prueba hayan cubierto todas las clases de equivalencia inválidas, se escribe un caso de la prueba que cubra una, y solamente una, de las clases de equivalencia inválidas descubiertas.

3.2 Descripción de los test de caja negra.

3.2.1 Objetivo del test

Detectar el incorrecto o incompleto funcionamiento, así como los errores de interfaces, rendimiento, errores de inicialización y terminación de un caso de uso crítico:

- ✓ Gestionar Tipo de Solicitud.

Las pruebas realizadas y descritas a este caso de uso son:

- ✓ Adicionar Tipo de Solicitud.
- ✓ Modificar Tipo de Solicitud.
- ✓ Eliminar Tipo de Solicitud.
- ✓ Listar Tipo de Solicitud.

CPR1: Adicionar Tipo de Solicitud.

Descripción

El escenario se inicia cuando el Administrador necesita adicionar un Tipo de Solicitud. Selecciona la opción de “Nuevo Tipo de Solicitud” en el menú principal del submódulo Tipo de Solicitud y llena los datos del mismo,

finalizando así el escenario.

Flujo central

El flujo central no es más que la interacción Administrador y aplicación, paso a paso; el pedido y la respuesta.

- ✓ El sistema muestra la interfaz que permite la autenticación al Administrador.
- ✓ El Administrador se autentica correctamente, el sistema muestra la interfaz principal del panel de control.
- ✓ El Administrador selecciona el módulo Recursos y dentro de él, el submódulo Tipo de Solicitud.
- ✓ El Administrador selecciona la opción “Nuevo Tipo de Solicitud” del menú principal de trabajo.
- ✓ El Administrador llena los datos del Nuevo Tipo de Solicitud.
- ✓ El Administrador pulsa el botón “Aceptar”.

Pre condiciones

- ✓ Administrador autenticado satisfactoriamente en el sistema.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El Administrador autenticado tiene los permisos adecuados.	El sistema no muestra la interfaz para adicionar un Nuevo Tipo de Solicitud.	El sistema muestra la interfaz correspondiente.	Satisfactorio	
El Administrador selecciona la opción “Nuevo Tipo de Solicitud” del menú principal de trabajo.	No se muestra la interfaz del Nuevo Tipo de Solicitud.	Muestra la interfaz del Nuevo Tipo de Solicitud.	Satisfactorio	

El Administrador llena los datos del Nuevo tipo de Solicitud.		El sistema muestra el siguiente mensaje: "Recurso adicionado satisfactoriamente".	Satisfactorio	
	El Administrador no llena todos los datos del Nuevo Tipo de Solicitud.	El sistema muestra el siguiente mensaje: "El campo Nombre es obligatorio" o "El campo Estado es obligatorio", en dependencia del campo que quede vacío.	Satisfactorio	
	El Administrador no llena ninguno de los datos del Nuevo Tipo de Solicitud.	El sistema muestra el siguiente mensaje: "El campo Nombre es obligatorio" y "El campo Estado es obligatorio",	Satisfactorio	

Tabla Sección: Adicionar Nuevo Tipo de Solicitud.

CPR2: Modificar Tipo de Solicitud.

Descripción

El escenario se inicia cuando el Administrador necesita modificar un Tipo de Solicitud. Selecciona la opción Tipo de Solicitud en el menú principal del submódulo de Recursos y realiza una búsqueda para encontrar el Tipo de Solicitud a modificar. Una vez encontrado el Tipo de Solicitud a modificar, lo selecciona y modifica.

Flujo central

- ✓ El sistema muestra la interfaz que permite la autenticación al Administrador.
- ✓ Si el Administrador se autentica correctamente, el sistema muestra la interfaz principal del panel de control.

- ✓ El Administrador selecciona el módulo Recursos y dentro de él el submódulo Tipo de Solicitud.
- ✓ El sistema le muestra el panel de control correspondiente.
- ✓ El Administrador llena los campos necesarios para realizar la búsqueda.
- ✓ El sistema muestra el o los Tipos de Solicitud encontrados.
- ✓ Una vez obtenido el Tipo de Solicitud a modificar, el Administrador lo selecciona.
- ✓ El Administrador modifica los campos deseados.
- ✓ El Administrador pulsa el botón “Modificar”.

Pre condiciones

- ✓ Administrador autenticado satisfactoriamente en el sistema.
- ✓ Debe existir el Tipo de Solicitud a modificar.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El Administrador autenticado tiene los permisos adecuados.	El sistema no muestra la interfaz para modificar el Tipo de Solicitud.	El sistema muestra la interfaz correspondiente.	Satisfactorio	
El Administrador introduce datos válidos para la búsqueda.		El sistema muestra el o los elementos encontrados.	Satisfactorio	
	El Administrador no llena todos los campos de búsqueda.	El sistema muestra el o los elementos encontrados.	Satisfactorio	
El Administrador	No se muestra la	Muestra la interfaz del	Satisfactorio	

selecciona el Tipo de Solicitud a modificar.	interfaz del Tipo de Solicitud.	Tipo de Solicitud.		
El Administrador modifica los datos deseados.		El sistema muestra el siguiente mensaje: "Recurso modificado satisfactoriamente".	Satisfactorio	

Tabla Sección: Modificar Tipo de Solicitud.

CPR3: Eliminar Tipo de Solicitud.

Descripción

El escenario se inicia cuando el Administrador necesita eliminar un Tipo de Solicitud. Selecciona la opción Tipo de Solicitud en el menú principal del submódulo de Recursos y realiza una búsqueda para encontrar el Tipo de Solicitud. Una vez encontrado el Tipo de Solicitud se selecciona y elimina.

Flujo central

- ✓ El sistema muestra la interfaz que permite la autenticación al Administrador.
- ✓ Si el Administrador se autentica correctamente, el sistema muestra la interfaz principal del panel de control.
- ✓ El Administrador selecciona el módulo Recursos y dentro de él el submódulo Tipo de Solicitud.
- ✓ El sistema le muestra el panel de control correspondiente.
- ✓ El Administrador llena los campos necesarios para realizar la búsqueda.
- ✓ El sistema muestra el o los Tipos de Solicitud encontrados.
- ✓ El Administrador selecciona el o los Tipos de Solicitud a eliminar.
- ✓ El Administrador pulsa el botón "Eliminar".

Pre condiciones

- ✓ Administrador autenticado satisfactoriamente en el sistema.
- ✓ Debe existir el Tipo de Solicitud a modificar.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El Administrador autenticado tiene los permisos adecuados.	El sistema no muestra la interfaz para eliminar el o los Tipos de Solicitud.	El sistema muestra la interfaz correspondiente.	Satisfactorio	
El Administrador introduce datos validos para la búsqueda.		El sistema muestra el o los elementos encontrados.	Satisfactorio	
	El Administrador no llena todos los campos de búsqueda.	El sistema muestra el o los elementos encontrados.	Satisfactorio	
El Administrador selecciona el o los Tipos de Solicitud a eliminar.	No se muestra el o los Tipos de Solicitud.	Muestra el o los Tipos de Solicitud.	Satisfactorio	
El Administrador elimina el o los Tipos de Solicitud deseados.		El sistema elimina el o los Tipos de Solicitud.	Satisfactorio	

Tabla Sección: Eliminar Tipo de Solicitud.

CPR4: Listar Tipo de Solicitud.

Descripción

El escenario se inicia cuando el Administrador necesita listar uno o varios Tipos de Solicitud. Selecciona la opción Tipo de Solicitud en el menú principal del submódulo de Recursos y realiza una búsqueda para encontrar el o los Tipos de Solicitud. Una vez realizada la búsqueda se listan los elementos encontrados.

Flujo central

- ✓ El sistema muestra la interfaz que permite la autenticación al Administrador.
- ✓ Si el Administrador se autentica correctamente, el sistema muestra la interfaz principal del panel de control.
- ✓ El Administrador selecciona el módulo Recursos y dentro de él el submódulo Tipo de Solicitud.
- ✓ El sistema le muestra el panel de control correspondiente.
- ✓ El Administrador llena los campos necesarios para realizar la búsqueda.
- ✓ El sistema lista los elementos encontrados.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El Administrador autenticado tiene los permisos adecuados.	El sistema no muestra la interfaz para realizar la búsqueda los Tipos de Solicitud.	El sistema muestra la interfaz correspondiente.	Satisfactorio	
El Administrador introduce datos válidos para realizar la búsqueda de Tipos de Solicitud a listar.	El sistema muestra el listado con los elementos encontrados.	El sistema muestra el o los elementos encontrados.	Satisfactorio	

	El Administrador no llena todos los campos de búsqueda.	El sistema muestra el o los elementos encontrados.	Satisfactorio	

Tabla Sección: Eliminar Tipo de Solicitud.

La robustez del software es un problema que a todos importa pero que pocas personas aplican en sus productos. De ahí la necesidad de realizarle pruebas al software con un determinado tiempo por delante. Una vez concluida la etapa de prueba a la aplicación en cuestión se ve claramente esta fase como una de las más importantes, si a calidad se refiere. Es aquí donde se valida un producto con una terminación bien concebida.

CONCLUSIONES

La Gestión de Recursos supone un gran esfuerzo, si se tiene en cuenta los procesos a los que pueden ser sometidos para su propia gestión, más; este es un proceso que debe soportar grandes volúmenes de información, de ahí la necesidad de la solución aquí propuesta, pues una vez que existe una herramienta como esta, se humanizan los procesos y lo que puede llegar a ser abrumador, se convierte en algo simple.

Ha sido objetivo siempre de los implementadores de la aplicación aquí propuesta, ofrecer al MENPET una aplicación Web, robusta y liviana a la vez. Para dar cumplimiento a esto se usaron herramientas muy bien definidas desde el inicio de la implementación, se pueden citar entre estas: Alfresco, Codelgniter, Zend Studio, PHP como lenguaje de desarrollo y Apache Tomcat como contenedor web. Para hacer un trabajo actualizado se tuvieron en cuenta además de las herramientas mencionadas, tecnologías como: Web Services, el concepto de arquitectura SOA y el patrón de arquitectura MVC.

Luego de analizadas las herramientas y tecnologías, se hizo un detallado estudio sobre el Diseño propuesto por el analista. El estudio se basó principalmente en los cambios tecnológicos sufridos desde que se diseñó el sistema hasta este momento, donde reuniendo lo más actual en cuanto a desarrollo software refiere, se decidió cambiar algunas herramientas y tecnologías, mencionadas con anterioridad.

Al concluir la implementación, intentando reunir los mejores elementos para esta, se procedió a realizarle pruebas al software, con el objetivo de validar un trabajo que atraiga al usuario desde el propio momento en que accede al sistema y quede satisfecho una vez que lo use, para lograr este propósito y teniendo en cuenta, que el usuario tiene prioridad, se realizaron Pruebas de Caja Negra, que son las más cercanas a la interfaz. En el desarrollo del documento aquí propuesto se citan solo algunas de estas pruebas, pero las mismas fueron aplicadas al software completo. Con la realización de dichas pruebas, se validó el producto, quedando listo para su uso.

De manera general se cumplieron los objetivos trazados, aunque en las Recomendaciones se muestran en orden de prioridad, puntos que en un futuro agregarían valor al sistema, complaciendo así a las dos partes cliente e implementadores.

RECOMENDACIONES

Con vistas a dar continuidad al desarrollo de este proyecto se recomienda:

- ✓ Integrar el módulo al Sistema de Control de Documentos.
- ✓ Poner en práctica la solución aquí propuesta en la propia institución para la cual se diseñó y así validar su uso y adaptación.
- ✓ Realizar una encuesta en la propia institución una vez puesta en práctica la solución, para futuras mejoras de esta.
- ✓ Realizar en versiones futuras una aplicación genérica.

REFERENCIAS BIBLIOGRAFICAS

1. *Gestión de Contenidos*. [Consultado el: 7 de Noviembre de 2008]. Disponible en: http://www.gestiondelconocimiento.com/documental.htm.

2. WIKIPEDIA, L. E. L. *Gestión Documental* [Consultado el: 7 de Noviembre de 2008]. Disponible en: http://es.wikipedia.org/wiki/Gesti%C3%B3n_documental.

3. *Componentes de la Gestión Documental* [Consultado el: 15 de Noviembre de 2008]. Disponible en: http://www.ekontsulta.net/ekontsulta/wiki/index.php/Gesti%C3%B3n_documental.

4. *Aplicaciones Web a la medida* México: [Consultado el: 20 de Noviembre de 2008]. Disponible en: http://www.intellia.com.mx/esp/servicios/aplicaciones_web_a_la_medida.php.

5. *Cliente-servidor* [Consultado el: 11 de Febrero de 2009]. Disponible en: http://es.wikipedia.org/wiki/Cliente-servidor.

6. TORRES, P. N. *Paradigmas de Programación* [Consultado el: 20 de febrero de 2009] Disponible en: http://asdfers.awardspace.com/wiki/doku.php?id=wiki:paradigmas.

7. *Programación imperativa*. [Consultado el: 20 de febrero de 2009]. Disponible en: http://es.wikipedia.org/wiki/Programaci%C3%B3n_imperativa.

8. *Programación funcional*. [Consultado el: 20 de febrero de 2009]. Disponible en: http://es.wikipedia.org/wiki/Programaci%C3%B3n_funcional.

9. *Programación lógica*. [Consultado el: 20 de febrero de 2009]. Disponible en: http://es.wikipedia.org/wiki/Programaci%C3%B3n_l%C3%B3gica.

10. *Programación orientada a objetos*. [Consultado el: 20 de febrero de 2009]. Disponible en: http://es.wikipedia.org/wiki/Paradigma_de_programaci%C3%B3n.

11. SCHWINDT, A. *Construcción de Sistemas Multiplataforma basados en Servicios* [Consultado el: 10 de Mayo de 2009]. Disponible en: http://msdn.microsoft.com/es-es/library/bb972254.aspx.

12. *Proceso Unificado*. [Consultado el: 05 de febrero de 2009]. Disponible en: http://es.wikipedia.org/wiki/Proceso_Unificado.

13. *PHP*. [Consultado el: 10 de febrero de 2009]. Disponible en: http://es.wikipedia.org/wiki/Php.

14. *Modelo Vista Controlador*. [Consultado el: 20 de marzo de 2009]. Disponible en: http://es.wikipedia.org/wiki/Modelo_Vista_Controlador.

15. *Web Service*. [Consultado el: 20 de marzo de 2009] Disponible en: http://wiki.alfresco.com/wiki/Alfresco_Content_Management_Web_Services#Web_Services_Reference.

16. *Arquitectura Orientada a Servicios*. [Consultado el: 26 de febrero de 2009]. Disponible en: http://www-03.ibm.com/e-business/la/ec/soa/?cm_re=masthead-_-solutions-_-soa.

17. MORA, R. C. *¿Qué ofrece Autentia?* España: [Consultado el: 25 de febrero de 2009] Disponible en: http://www.adictosaltrabajo.com/tutoriales/pdfs/soa.pdf.

18. WIKIPEDIA, L. E. L. *Extensible Markup Language* [Consultado el: 10 de Febrero de 2009]. Disponible en: http://es.wikipedia.org/wiki/XML#Objetivos_y_usos_de_XML.

19. ALFRESCO.COM. *The Open Source Alternative for Enterprise Content Management* [Consultado el: 11 de Febrero de 2009]. Disponible en: http://www.alfresco.com/es/.

20. *Framework*. [Consultado el: 21 de febrero de 2009]. Disponible en: http://es.wikipedia.org/wiki/Framework.

21. *Symfony*. [Consultado el: 21 de febrero de 2009]. Disponible en: http://es.wikipedia.org/wiki/Symfony.

22. *Zend Framework*. [Consultado el: 21 de febrero de 2009]. Disponible en: http://es.wikipedia.org/wiki/Zend_Framework.

23. *CodeIgniter*. [Consultado el: 21 de febrero de 2009]. Disponible en: http://sentidoweb.com/2007/04/24/codeigniter-framework-para-php.php.

24. *Software development kit*. [Consultado el: 21 de marzo de 2009]. Disponible en: http://es.wikipedia.org/wiki/Software_development_kit.

25. *NetBeans IDE 6.5*. [Consultado el: 21 de marzo de 2009]. Disponible en: http://www.netbeans.org/features/php/index.html.

26. *Visual Paradigm*. [Consultado el: 21 de marzo de 2009]. Disponible en: http://www.visual-paradigm.com/.

27. WIKIPEDIA, L. E. L. *Servidor HTTP Apache* [Consultado el: 9 de Enero de 2009]. Disponible en: http://es.wikipedia.org/wiki/Apache_http_server.

28. *CamelCase*. [Consultado el: 21 de marzo de 2009]. Disponible en: http://en.wikipedia.org/wiki/CamelCase.

BIBLIOGRAFIA

Apache Tomcat [En línea].
http://tomcat.apache.org/.

Aplicaciones Web a la medida México: [En línea]
http://www.intellia.com.mx/esp/servicios/aplicaciones_web_a_la_medida.php.

Beneficios De Las Aplicaciones Basadas En Web Y El Anuncio De Microsoft De La Era “En Vivo” [En línea]
http://www.masternewmedia.org/es/aplicaciones_web/temas_de_aplicaciones_web/Beneficios_De_Las_Aplicaciones_Basadas_En%20_Web_Y_El_Anuncio_De_Microsoft_De_La_Era_En_Vivo.htm.

Bonaparte, Ing. Ubaldo. *Paradigmas de Programación.* [En línea]
http://www.frt.utn.edu.ar/sistemas/paradigmas/page22.html.

CamelCase. [En línea].
http://en.wikipedia.org/wiki/CamelCase.

Cerami, Ethan. 2002. *Web Services Essentials.* s.l. : O'Reilly, 2002. 0-596-00224-6.

Codelgniter: [En línea].
http://sentidoweb.com/2007/04/24/codeigniter-framework-para-php.php.

Collin, Peter. *Publishing Dictionary of Computing, 4th Edition.*

Componentes de la Gestión Documental [En línea]
http://www.ekontsulta.net/ekontsulta/wiki/index.php/Gesti%C3%B3n_documental.

Framework: [En línea].
http://es.wikipedia.org/wiki/Framework.

Gestión de Contenidos. [En línea]
http://www.gestiondelconocimiento.com/documental.htm.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El Proceso Unificado de Desarrollo de Software.*

NetBeans IDE 6.5. [En línea].
http://www.netbeans.org/features/php/index.html.

Pressman, Roger S**.** *Ingeniería del Software, Un enfoque práctico, Parte 1.*

. *Ingeniería del Software, Un enfoque práctico, Quinta edición.*

Programación imperativa: [En línea].
http://es.wikipedia.org/wiki/Programaci%C3%B3n_imperativa.

Programación funcional: [En línea].
http://es.wikipedia.org/wiki/Programaci%C3%B3n_funcional.

Programación lógica: [En línea].
http://es.wikipedia.org/wiki/Programaci%C3%B3n_l%C3%B3gica.

Programación orientada a objetos: [En línea].
http://es.wikipedia.org/wiki/Paradigma_de_programaci%C3%B3n.

Sistemas de Gestión Documental y Bibliográfica. [En línea]
http://www.csi.map.es/csi/silice/Sisgesdoc1.html.

Software development kit: [En línea].
http://es.wikipedia.org/wiki/Software_development_kit.

Symfony: [En línea].
http://es.wikipedia.org/wiki/Symfony.

TORRES, P. N. Paradigmas de Programación: [En línea]
http://asdfers.awardspace.com/wiki/doku.php?id=wiki:paradigmas.

Visual Paradigm. [En línea].
http://www.visual-paradigm.com/.

Zend Framework: [En línea].
http://es.wikipedia.org/wiki/Zend_Framework.

GLOSARIO DE TERMINOS

Bugs: Un defecto de software, es el resultado de un fallo o deficiencia durante el proceso de creación de programas de ordenador o computadora (software). Dicho fallo puede presentarse en cualquiera de las etapas del ciclo de vida del software aunque los más evidentes se dan en la etapa de desarrollo y programación. Los errores pueden suceder en cualquier etapa de la creación de software.

Clase: Es una abstracción sobre un conjunto de objetos que tienen en común estructura y comportamiento.

Herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Ordenador): Son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

IDE: Entorno de Desarrollo Integrado

Ingeniería de software: es una disciplina que integra procesos, métodos y herramientas para el desarrollo de software de computadora.

Módulo: Encapsula un conjunto de funciones que debe realizar el sistema, las cuales son agrupadas por tener características muy similares y se definen en la etapa de diseño.

Patrón: Son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Software: Es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación.