Universidad de las Ciencias Informáticas

Facultad 1

Título: Diseño e implementación del componente de configuración de las planillas evaluativas del proceso de evaluaciones**.**

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

Autor: Cadete Leordan Valdes Gutierrez

Tutor: Tte. Ing. Yadir Martínez Vergara

Ciudad de La Habana, 2010.

Año del 52 de la Revolución.

Declaración de autoría

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

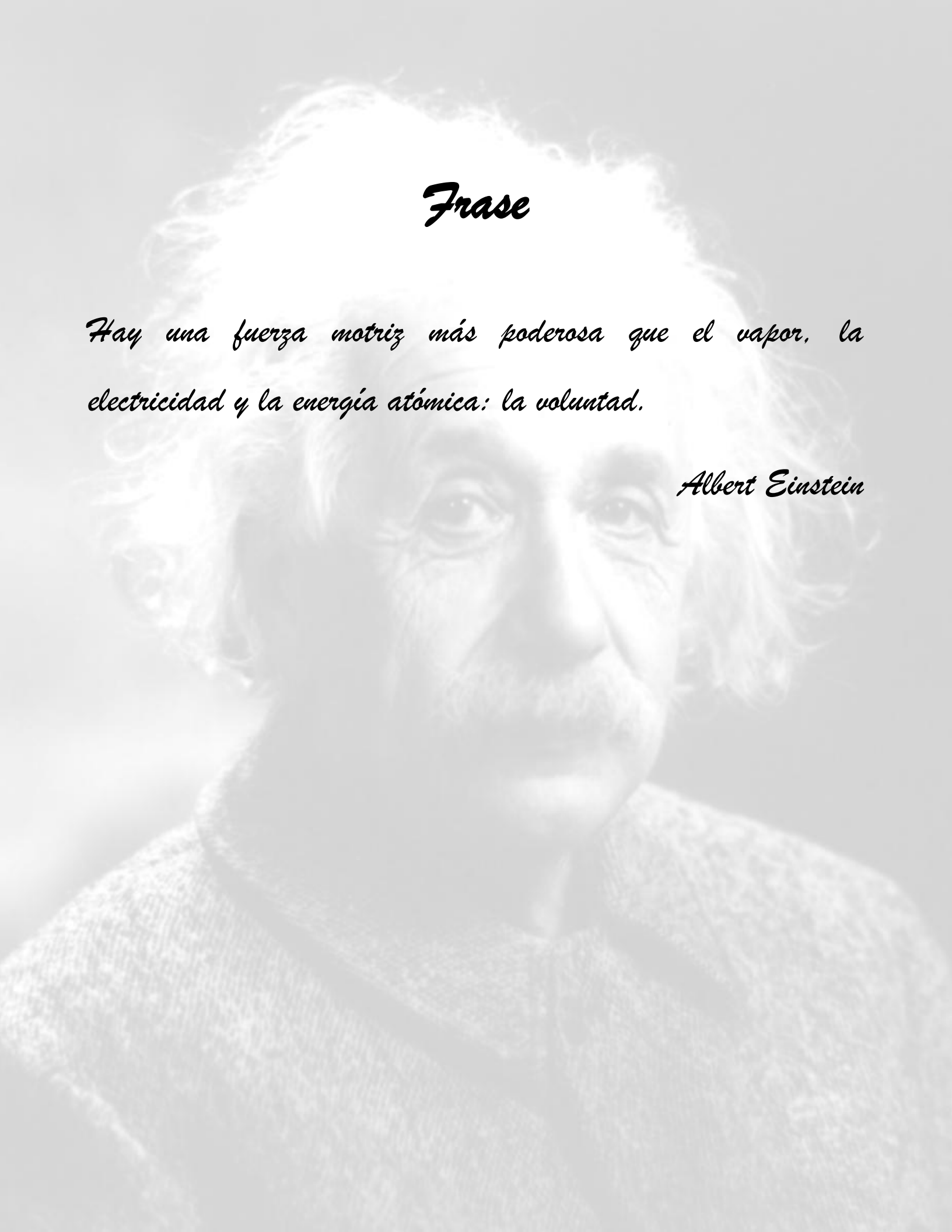
Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____	_____
Cadete Leordan Valdes Gutierrez	Tte. Ing. Yadir Martínez Vergara
Autor.	Tutor.

Frase

Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad.

Albert Einstein



Datos de contacto

Tutor: Tte. Ing. Yadir Martínez Vergara

Categoría Científica: Ingeniero.

Correo electrónico: yvergara@uci.cu

Síntesis del Tutor:

Ingeniero en Ciencias Informáticas. Graduado en julio del 2008. Arquitecto de Base de Datos en la Línea de Capital Humano de la Unidad de Compatibilización e Integración y Desarrollo de Software para la Defensa (UCID) - Universidad de la Ciencias Informáticas.

Agradecimientos

A mis padres por la confianza, el apoyo y el amor, por sus consejos, por ser mis principales guías y ayudarme a llegar hasta aquí.

A mi novia por estar a mi lado todos estos años, dedicándome su tiempo, confianza y apoyo.

A mi tutor por la ayuda incondicional que me ha brindado en estos últimos meses.

A toda mi familia, ya que de una forma u otra han contribuido con mi superación.

A todas las personas que de una forma u otra han contribuido en mi desarrollo profesional, que cada día han estado cerca para enseñarme cosas nuevas de la vida y llegar a la meta que me he propuesto, esas personas que han estado en los buenos y difíciles momentos de mi vida. Gracias por la paciencia de todos aquellos que han estado a mi lado cada segundo de preocupación y dedicación, muchas gracias. A mis amigos por los consejos, por la seguridad que siempre han tenido en mí.

A Fidel, a Raúl y a la Revolución.

A todos, muchas gracias.

Dedicatoria

Quiero dedicar este Trabajo de Diploma a todos los profesores que han contribuido con mi educación, a todos mis amigos(as), a mis primos(as), a mis tíos(as), a mi hermanastro Alexander, a mi padrastro Leon, a mis hermanas Yeny, Yanay y Meliza, a mi novia Yanet y en especial a mis padres Gladys y Justo Luis por haberme traído al mundo y ayudarme a ser lo soy, un ingeniero informático.

Resumen

El presente trabajo pretende mejorar lo referente a la creación de planillas evaluativas en los diferentes procesos evaluativos que se llevan a cabo en el país. Teniendo en cuenta que no existe ningún software que realice esta función en específico se propone como solución la implementación del componente de configuración de las planillas evaluativas del proceso de evaluaciones el cual junto con otros componentes permitirán la preparación, definición y resumen de cualquier proceso de evaluación.

Para su desarrollo se realizó un amplio estudio de los procesos de evaluación que se llevan a cabo en el país y sus principales características con el objetivo de conocer el funcionamiento de estos para garantizar un software acorde a las necesidades. Se hizo un estudio de las herramientas, lenguajes y tecnologías que fueron escogidos por el equipo de proyecto para proceder al diseño e implementación del sistema. Se generó toda la documentación referente a los flujos de trabajo que propone el proceso de desarrollo de software propuesto por el centro UCID. Finalmente, se hicieron las pruebas para verificar la calidad con el objetivo de garantizar la aceptación de los clientes y usuarios finales.

Palabras claves: Proceso evaluativo, planillas, implementación, aceptación.

Índice

Índice de figuras

Introducción

En la actualidad las Tecnologías de la Información y las Comunicaciones se han convertido en una necesidad más que un lujo, la competencia en la red se hace cada vez mayor. Por esta razón las empresas están informatizando poco a poco todos sus procesos para poder estar en este mercado futurista, necesario y de mayor demanda. Con la ayuda de las TIC[1] en los últimos años se han agilizado procedimientos que anteriormente se hacían con lentitud, además de aumentar la eficacia, como por ejemplo: la compra de un artículo, la inscripción de nacimiento de una persona, el control de servicios postales, etcétera.

Cuba no le ha dado la espalda a este movimiento revolucionario e innovador, al contrario, se ha trazado estrategias de automatización e informatización de la sociedad para así lograr una mayor producción y desempeño en las diferentes esferas. Últimamente se ha hecho énfasis en los procesos relacionados con el capital humano como por ejemplo: selección integración, organización del trabajo, evaluación de desempeño, entre otros.

El proceso de evaluaciones se desarrolla en todo el territorio nacional, este se realiza manualmente el cual es demasiado engorroso debido a todo el papeleo que este requiere. En el transcurso de dicho proceso se utiliza un documento muy importante, se trata de la planilla evaluativa la cual contiene aspectos a tener en cuenta durante el mismo, esta cambia según el fin con que esté elaborada.

Ante la situación descrita anteriormente el **problema a resolver** queda expresado mediante la siguiente incógnita: ¿Cómo facilitar la configuración de las planillas evaluativas del proceso de evaluaciones en una entidad?

Teniendo en cuenta el problema planteado se define como **objeto de estudio**: El proceso de gestión del capital humano.

Por lo que se especifica el siguiente **campo de acción:** El proceso de evaluación de desempeño.

[1] TIC: Tecnología de la Información y las Comunicaciones

El trabajo se basa en la siguiente **idea a defender:** Si se implementa un componente para la configuración de las planillas evaluativas entonces se facilitará el proceso de evaluaciones en las entidades.

Para dar respuesta al problema planteado se trazó el siguiente **objetivo general:** Diseñar e implementar un componente que facilite la configuración de las planillas evaluativas en una entidad.

Derivándose los siguientes **objetivos específicos**:

1. Realizar el diseño teórico de la investigación.
2. Realizar un estudio del estado del arte de los principales procesos de evaluación que se llevan a cabo en el país.
3. Diseñar el componente de configuración de las planillas evaluativas.
4. Implementar el componente de configuración de las planillas evaluativas.
5. Validar el componente de configuración de las planillas evaluativas.

El documento está estructurado en tres capítulos:

Capítulo I: Fundamentación teórica

Se realiza un resumen y análisis crítico de la tesis precedente donde se exponen los principales procesos de evaluación existentes en el país explicando dos de ellos, aplicaciones que pudieran de una forma u otra contribuir con el desarrollo de este componente y cambios realizados. Se hace además una descripción detallada del proceso de desarrollo y las herramientas utilizadas para alcanzar los objetivos propuestos.

Capítulo II: Diseño del sistema

Se realizan los diagramas de interacción de los escenarios de cada requisito, los diagramas de clases web con una descripción de cada uno de ellos y el diagrama entidad relación de la base de datos

describiendo además cada una de las tablas de la misma. Se hacen definiciones de diseño, tratamiento de errores, manejo de la seguridad y concepción de la ayuda del sistema.

Capítulo III: Implementación y prueba

Todo el diseño realizado anteriormente se utiliza para realizar la implementación del software. Se muestra el diagrama de componentes donde se ve plasmada la relación que existe entre cada uno de ellos. También se especifican los tipos de pruebas realizadas al sistema. Además, se presentan las diferentes interfaces de la aplicación obtenida.

Capítulo 1: Fundamentación Teórica

1.1. Introducción

En el presente capítulo se realiza un estudio crítico de la tesis precedente después del cual se verificarán cambios realizados durante el transcurso de este tiempo. Además, se tratarán las principales características del proceso de desarrollo de software a tener en cuenta durante el progreso del componente y se efectuará un estudio del estado del arte de las herramientas, lenguajes y tecnologías a utilizar.

1.2. Estudio de la tesis precedente

1.2.1. Resumen del capítulo 1

La tesis anterior comienza con la exposición de un concepto fundamental para una mayor comprensión del tema a tratar en el trabajo, se trata de evaluación, que según lo establecido en la Gaceta Oficial de Cuba en la Resolución No.21/2007 Apartado Segundo, *es la medición sistemática del grado de eficacia y eficiencia con el que los trabajadores realizan sus actividades laborales durante un período de tiempo determinado y de su potencial de desarrollo, y constituye la base para elaborar y ejecutar el plan individual de capacitación y desarrollo.* [1]

Posteriormente se exponen los principales procesos de evaluación existentes en el país como son:

- Proceso de Evaluación del PCC[2]
- Proceso de Evaluación de la UJC[3]
- Proceso de Evaluación de la CTC[4]
- Proceso de Evaluación del MININT[5]

[2] PCC: Partido Comunista de Cuba

[3] UJC: Unión de Jóvenes Comunistas

[4] CTC: Central de Trabajadores de Cuba

- Proceso de Evaluación de los Cuadros de las FAR[6]
- Proceso de Evaluación del Ministerio de Trabajo y Seguridad Social

De los procesos mencionados anteriormente se hace un resumen de los últimos dos donde se exponen las principales características de cada uno.

1.2.1.1. Proceso de evaluación de los Cuadros de las FAR

Las evaluaciones se realizan a todas las categorías de personal (cuadros, civiles, oficiales, camilitos, cadetes, alumnos, etc.) de las FAR y a todos los niveles. En el caso específico de los oficiales existen dos modelos de evaluación, el integral que se realiza cada 3 años y el parcial que se realiza todos los años, los cuales pueden ser modificados con el transcurso del tiempo.

Los cuadros deben ser evaluados periódicamente debido a diferentes causas como la propuesta de movimiento, tiempo de estancia en el mismo cargo, cumplimiento del periodo de instrucción, entre otras.

De los resultados individuales del proceso evaluativo dependen la selección, movimientos, preparación y superación, la atención y estímulos de los cuadros, ello obliga a poner un cuidado especial en todo el proceso en particular en las valoraciones que se realicen, pues un error puede determinar desde el destino de un cuadro hasta su cese en el servicio. Del análisis de los resultados de las valoraciones a cualquier nivel se originan decisiones como la inclusión en la reserva de cuadros, en los Listados de Candidatos a Curso, la elaboración del Plan de Desarrollo Perspectivo y Proyección de los Cuadros, y otros, que convierten la evaluación en un efectivo instrumento para la aplicación de la Política de Cuadros.

---

[5] MININT: Ministerio del Interior

[6] FAR: Fuerzas Armadas Revolucionarias

1.2.1.2. Proceso de evaluación del Ministerio de Trabajo y Seguridad Social

Para acometer la evaluación del desempeño se deben tener un *"Reglamento de evaluación del desempeño de los trabajadores"*, donde cada entidad, lo elaborará, de común acuerdo con la organización sindical correspondiente a ese nivel, lo cual se inscribe en el *Convenio Colectivo de Trabajo*.

Para realizar la evaluación del desempeño de los trabajadores se establecen los indicadores fundamentales siguientes:

- Cumplimiento de las recomendaciones derivadas de la evaluación del desempeño anual anterior.
- Cumplimiento de sus objetivos, funciones y tareas individuales, y la realización del trabajo con eficiencia, calidad y productividad requerida.
- Comportamiento de la disciplina y el aprovechamiento de la jornada de trabajo.
- Cumplimiento de las normas de seguridad y salud en el trabajo.
- Uso y cuidado de los recursos materiales, fundamentalmente de los portadores energéticos, de los equipos y medios de protección personal.
- Cumplimiento del plan de capacitación y desarrollo individual.

La evaluación del desempeño se realiza anualmente, con cortes parciales mensuales y trimestrales. Para realizar la evaluación de desempeño, el trabajador deberá tener laborado, como mínimo, el 70% de su tiempo de trabajo.

La administración debe llevar un registro en el que aparecen los avances en el desempeño del trabajador y las deficiencias que aún subsisten, así como las recomendaciones que se le realizan. Las evaluaciones parciales deberán archivarse por el jefe del trabajador que realiza la evaluación y la del año se archivará en el expediente laboral del trabajador, la cual podrá servir como base de análisis de la trayectoria del trabajador a los efectos de la promoción, reconocimiento moral o envío a cursos.

La evaluación del desempeño la realiza el jefe inmediato superior del trabajador, oído el parecer de la organización sindical y de otros trabajadores, de resultar necesario y comprende a los trabajadores de todas las categorías ocupacionales, excepto aquellos dirigentes, considerados como cuadros, cuya evaluación se rige por lo establecido en la política de cuadros vigente en el país.

La decisión del director general de la empresa sobre la evaluación del desempeño de un trabajador no tiene reclamación ni por la vía administrativa ni la judicial.
El trabajador evaluado puede reclamar ante el Órgano de Justicia Laboral de Base, en el caso en que aprecie violaciones de las normas y procedimientos establecidos para la evaluación del desempeño.

1.2.1.3. Tecnologías y herramientas propuestas

Después de realizada la explicación de estos procesos se efectúa un estudio de las tecnologías actuales a utilizar para darle solución a los problemas planteados las cuales fueron tomadas de la arquitectura del proyecto Cuadros del UCID[7]. Se seleccionó el Modelo Orientado a Componentes del Proyecto ERP-Cuba como metodología de desarrollo y Visual Paradigm for UML[8] 6.1 Enterprise Edition como herramienta para el modelado usando BPMN [9] y UML, PHP 5.2.4 para la implementación del lado del servidor y HTML[10] 4.0, JavaScript y CSS[11] como lenguajes del lado del cliente, el Marco de Trabajo UCID como marco de trabajo para la integración de los lenguajes ya seleccionados y así optimizar la construcción de la propuesta solución, Zend Studio for Eclipse 6.0.0 como IDE de desarrollo, PostgreSQL 8.3 como SGBD[12], pgAdmin III v1.8.2 como herramienta de

[7] UCID: Unidad de Compatibilización Integración y Desarrollo de Software para la Defensa

[8] UML: Unified Modeling Language (Lenguaje Unificado de Modelado)

[9] BPMN: Business Process Management Notation (Notación para el Modelado de Procesos de Negocio)

[10] HTML: Hypertext Markup Language (Lenguaje de Marcas de Hipertexto)

[11] CSS: Cascading Style Sheets (Hojas de Estilo en Cascada)

[12] SGBD: Sistema Gestor de Base de Datos

administración para el SGBD y como generador de reportes dinámicos el Sistema de gestión de reportes dinámicos v1.5. Y todo en completa armonía bajo el patrón MVC[13].
De cada una de estas tecnologías y herramientas se exponen sus características principales.

1.2.2. Resumen del capítulo 2

Este capítulo comienza con un estudio de aplicaciones que pudieran satisfacer las necesidades o simplemente aportar ideas para el desarrollo de la nuestra. Las herramientas encontradas fueron Lavasoft, EVAL 360, QTraining y Sistema de Registro y Control de Cuadros (SRCC) donde se llegó a la conclusión que:

- Lavasoft: No utiliza tecnologías web ni de software libre y su carácter estático es un obstáculo en nuestro camino a crear una aplicación genérica capaz de humanizar el proceso de gestión de las evaluaciones.
- EVAL 360: Es un software propietario, por lo que no nos brinda un paso para la soberanía tecnológica.
- QTraining: Es un software propietario y no permite la integración con otros sistemas existentes para obtener información y no nos brinda un paso para la soberanía tecnológica.
- Sistema de Registro y Control de Cuadros (SRCC)

  No permite:

  - Establecer reglas de negocio para el cálculo del resultado final de la evaluación.
  - Establecer prioridades entre los indicadores.
  - Configuración básica para la gestión del proceso evaluativo.
  - Recuperaciones dinámicas de estadísticas del proceso evaluativo.
  - Integración con otros sistemas como el ERP-Cuba.

[13] MVC: Modelo Vista Controlador

Debido a los datos emitidos anteriormente las aplicaciones no cumplen con las expectativas que se tienen con este trabajo por lo cual surge la necesidad de implementar un sistema de software que permita:

- Gestionar recursos, dominios, planillas, indicadores y escalas.
- Diseñar y mostrar las planillas creadas.

Posteriormente se adentra en el proceso de evaluación de los cuadros de las FAR realizando una descripción detallada de las principales fases por las que pasa:

- Preparar Proceso Evaluativo.
- Realizar Evaluación.
- Resumir Proceso Evaluativo.

1.2.3. Vista general de cambios realizados

Después de un análisis minucioso se detectaron algunos cambios producto de la interacción con los clientes y sus nuevos intereses. A grandes rasgos las novedades son las siguientes:

- La metodología Modelo Orientado a Componentes del Proyecto ERP-Cuba establecida anteriormente es sustituida por el Proceso de Desarrollo y Gestión de Proyectos de Software establecido en el centro UCID.

- Se hace una restructuración de las relaciones entre cada una de las clases además de agregarle otras necesarias en el proceso, lo dicho se ve reflejado en la Figura 1.1.

- Para una mejor organización y control de las planillas e indicadores se crean recursos y dominios. Un dominio evaluativo está asociado a un recurso determinado, es decir, cada recurso está compuesto por varios dominios en los cuales residirán las planillas e indicadores.

- Al nomenclador de escalas se le agrega un nuevo tipo, imagen, este contiene un código y un valor que sería una imagen como su nombre lo indica.

- Al nomenclador de indicadores se le agrega un nuevo tipo, imagen, este tiene los mismos atributos que los indicadores de texto además de nombre, extensión y en el caso de la denominación se guardará, en vez de un texto, una imagen.

- A los indicadores que son asignados a las planillas se les podrá determinar un complemento textual y fijar una escala por la cual serán evaluados posteriormente excepto a los indicadores de agrupación, se les llamarán así a los indicadores que no serán calculados.

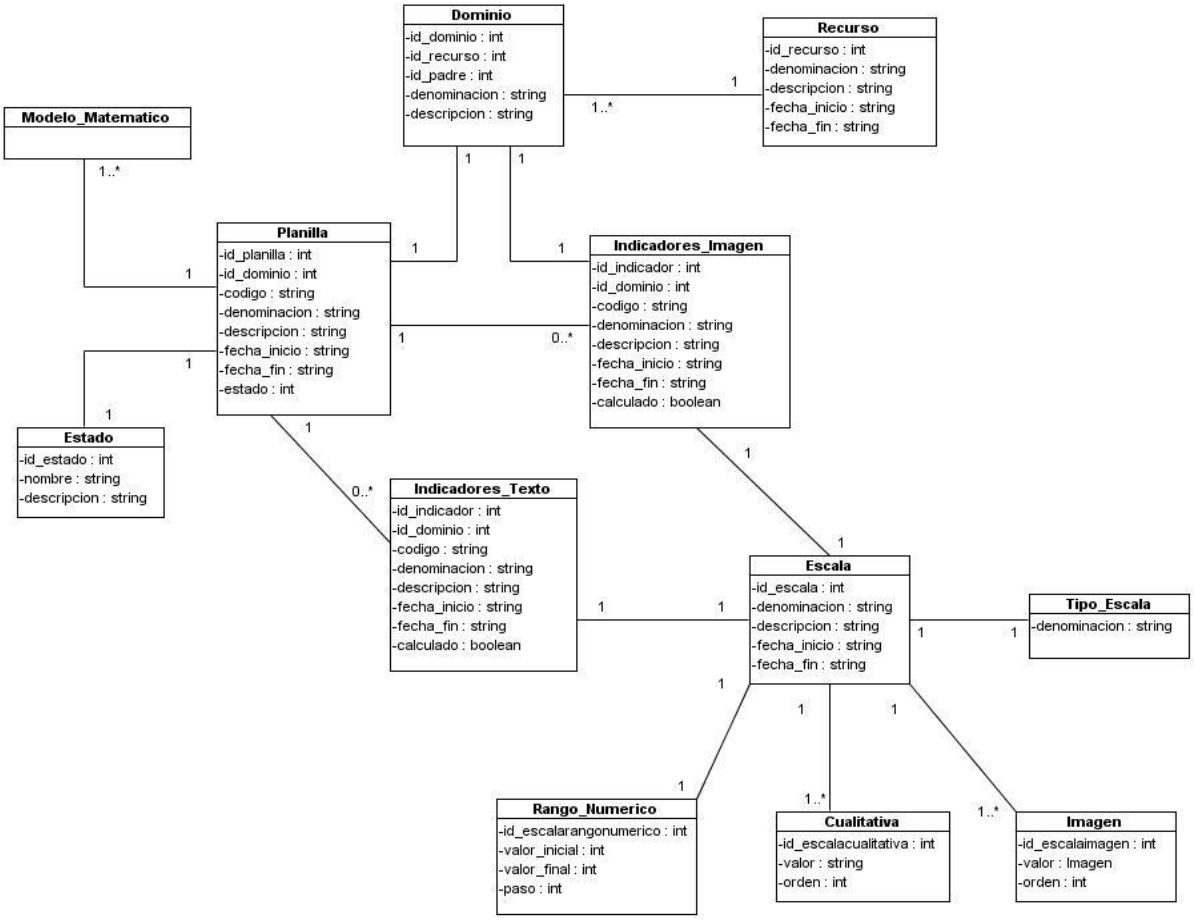


Figura 1: Modelo conceptual de planillas evaluativas

1.3. Propuesta de solución

Concluido el estudio de la tesis precedente se decide realizar una aplicación web de fácil manejo, multiplataforma, que cumpla con todos los requisitos especificados por los clientes que posibilite la gestión y configuración de las planillas evaluativas y de todos los elementos necesarios para su confección, siendo estos elementos recursos y dominios, indicadores, escalas, valores cualitativos y valores imágenes.

Figura 2: Propuesta de solución

1.4. Proceso de desarrollo de software

Un proceso de desarrollo de software tiene como propósito la producción eficaz y eficiente de un producto de software que reúna los requisitos del cliente. Este proceso es intensamente intelectual, afectado por la creatividad y juicio de las personas involucradas. Aunque un proyecto de desarrollo de software es equiparable en muchos aspectos a cualquier otro proyecto de ingeniería, en el desarrollo de software hay una serie de desafíos adicionales, relativos esencialmente a la naturaleza del producto obtenido.

1.4.1. Proceso de Desarrollo y Gestión de Proyectos de Software establecido en el centro UCID

Principales características del ciclo de vida del proyecto

- Las fases son secuenciales y su transferencia debe ser precedida por un proceso de revisión o liberación del Centro de Calidad y su aprobación en Consejo Técnico Formal.
- El nivel del personal es bajo al comienzo, alcanza su máximo nivel en la fase de construcción y decae rápidamente cuando el proyecto se aproxima a su conclusión.
- La participación de los interesados es alta en las etapas de Inicio y Modelación, baja en la etapa de Construcción y vuelve a subir en las etapas finales del proyecto.

El ciclo de vida del proyecto se considera como parte del ciclo de vida del producto, el cual representa un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en la adquisición, el desarrollo, la explotación y el mantenimiento de un producto

Fases principales del ciclo de vida del proyecto

- Inicio: Se logra una visión preliminar de la problemática a resolver, se identifica el alcance preliminar del proyecto, se especifican los involucrados y las líneas de desarrollo ejecutoras del proyecto constituyéndose el equipo de desarrollo y se estiman los recursos necesarios que deberán ser asignados al mismo.
- Modelación: Se definen los procesos del dominio del problema, se estiman los principales riegos que presenta el proyecto y se especifica la forma de mitigarlos, se identifican las necesidades del usuario de las que se derivan los requerimientos del producto a desarrollar, se determina la factibilidad operativa, técnica y/o económica de continuar el proyecto. Se define la arquitectura del software, se valida y establece para disponer de cimientos sólidos sobre los que se basará el grueso del esfuerzo durante la fase de Construcción. Se realiza la planificación detallada de la siguiente fase.

Los principales hitos de esta fase son:

- Análisis de las necesidades del usuario.
- Especificación de la Arquitectura.
- Planeamiento de la construcción del software.

- Construcción: En esta fase todas las características, componentes y requerimientos deben ser integrados, implementados y probados en su totalidad, obteniendo una versión estable del producto comúnmente llamada versión beta. Se realiza la planificación detallada de la siguiente fase.

  Los principales hitos de esta fase son:

  - Versión Funcional del producto (Versión Beta).
  - Manuales de usuario y de instalación.
  - Liberación de calidad del producto.
  - Planeación de la Explotación Experimental.

- Explotación experimental: Incluye las pruebas del producto como parte de su preparación para ser entregado, y la realización de ajustes en respuesta a la retroalimentación recibida de los usuarios. En este punto del ciclo de vida la retroalimentación de los usuarios debe enfocarse fundamentalmente en ajustes específicos y de corto alcance al producto junto a otros temas como configuración, instalación, y usabilidad.

  Los principales hitos de esta fase son:

  - Solución Estable.
  - Planeamiento para el despliegue.

- Aceptación de los resultados.

➢ Despliegue: Se realiza la generalización del producto en las entidades y órganos según lo aprobado en el Cronograma de implantación. Durante el proceso de implantación por lo general no es necesaria la participación de los integrantes del equipo de desarrollo.

Figura 3: Etapas del ciclo de vida del proyecto

Características esenciales

➢ Desarrollo iterativo e incremental: Es un enfoque en el que el ciclo de vida está compuesto por iteraciones, estas son pequeños procesos compuestos de varias actividades cuyo objetivo es entregar una parte del sistema parcialmente completo, probado, integrado y estable. Todo el software es integrado en cada entrega de cada iteración hasta obtener el producto de software completo en la última iteración. En cada iteración se obtiene como resultado un incremento.

- Desarrollo basado en componentes: Lleva a alcanzar un mayor nivel de reutilización de software, aún en contextos distintos de aquellos para los que fue diseñado. Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados. Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema. Dado que un componente puede ser construido y luego mejorado continuamente, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo.

- Orientado a procesos: La modelación de procesos de negocio nos permite realizar una rápida y profunda exploración del dominio del problema, con el fin de lograr comprensión por parte del equipo de desarrollo de los procesos que se realizan actualmente en la entidad y la relación que existe entre estos. De esta forma, se van determinando necesidades operacionales, así como restricciones que presenta la entidad, obteniéndose finalmente un entendimiento del negocio para dar paso a la fase inicial del sistema.

1.5. Herramientas, lenguajes y tecnologías utilizadas

Las herramientas, lenguajes y tecnologías que se describen a continuación son las utilizadas en la implementación del componente. Fue una decisión determinada por el equipo de arquitectura del Centro de Soluciones de Gestión.

1.5.1. Marco de Trabajo Sauxe

El Marco de Trabajo Sauxe se ha estructurado de manera tal que facilite la reutilización de los diferentes componentes y que sea de fácil entendimiento para todos los programadores que desarrollen sobre el mismo. Sauxe utiliza ExtJS para implementar la capa de presentación, se apoya en Zend, una extensión de Zend Framework para el desarrollo de la lógica del negocio y para la gestión de los datos utiliza Doctrine. Este utiliza como patrón arquitectónico MVC. También tiene como propósito insertar la programación orientada a aspectos así como la inversión de controles. El framework define que la conexión a la base de datos será configurada en un xml almacenado en la

carpeta de recursos comunes del proyecto. Es válido aclarar que este cuenta con un componente de transacciones mediante el cual serán salvados automáticamente los datos de modificaciones e inserciones. Es decir, ya no será necesaria la implementación por parte del programador de las consultas de inserción, éste solamente deberá programar la obtención de los campos de la presentación y el Sauxe será el responsable de que los mismos sean guardados en la base de datos.

Algunas de las ventajas más importantes del marco de trabajo:

- Los identificadores de cada tupla de las tablas serán generados automáticamente en la base de datos como una secuencia.
- El antiguo método de try y catch para el lanzamiento de excepciones desaparece, solo basta registrar las excepciones en el manager de excepciones y el framework será capaz de realizar un tratamiento óptimo de las mismas.
- No será necesario en cada método de inserción de información a la base de datos ejecutar el método correspondiente en la clase del negocio correspondiente, el framework será capaz de que una vez tomados los datos de la presentación salvarlos en la base de datos directamente.

1.5.2. ExtJS

ExtJS posee una librería JavaScript ligera y de alto rendimiento, compatible con la mayoría de los navegadores, que nos permite crear páginas e interfaces Web dinámicas usando tecnologías como AJAX, DHTML y DOM de manera semejante a aplicaciones desktop, incluye la mayoría de los controles de los formularios Web basándose en Grids para mostrar datos y elementos semejantes a la programación desktop como los formularios, paneles, barras de herramientas, menús y muchos otros. Dentro de su librería contiene componentes para el manejo de datos, lectura de XML, lectura de datos JSON e implementaciones basadas en AJAX. Permite balancear la carga entre Cliente – Servidor, la carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo; una comunicación asíncrona, en este tipo de aplicación puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se de cuenta, además de que facilita eficiencia de la

red, el tráfico de red puede disminuir al permitir que la aplicación elija que información desea transmitir al servidor y viceversa, sin embargo, la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio por el incremento del tráfico.

Actualmente ExtJS es considerado un Framework independiente ya que a principios del 2007 se creó una compañía para comercializar y dar soporte al mismo, dicha compañía proporciona los servicios de consultoría necesarios para ayudar a los clientes en el aprovechamiento máximo de sus ventajas además la comunidad que está detrás de esta herramienta es muy grande y la documentación cada vez es más extensa.

1.5.3. Zend Framework

Es un framework para desarrollo de aplicaciones y servicios Web con PHP. Brinda soluciones para construir sitios web modernos, robustos y seguros. Además, es de código abierto y trabaja con PHP 5.
Principales características:

- Trabaja en 3 capas, usando el Modelo Vista Controlador.
- Cuenta con módulos para manejar archivos en formato de documento portátil, canales de sindicación de noticias y servicios web.
- Incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple para consultar la base de datos.
- Completa documentación y pruebas de alta calidad.
- Robustas clases para autenticación y filtrado de entrada.
- Clientes para servicios web. [2]

1.5.4. Doctrine

Doctrine es un potente y completo sistema Mapeador de Objeto Relacional (*Object Relational Mapper*, ORM por sus siglas en inglés) para PHP 5.2+ que incorpora una DBL (capa de abstracción a base de datos). Entre muchas otras cosas tiene la capacidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir, convertir clases a tablas de una base de datos. [3]

Una de sus principales características es la de escribir las consultas de bases de datos en un objeto propiamente orientado al Dialecto SQL llamado DQL (*Doctrine Query Language* por sus siglas en inglés) inspirado en Hibernate SQL. Esto ofrece a los desarrolladores una poderosa alternativa frente a SQL manteniendo la flexibilidad y permitiendo ampliamente la reutilización de código.

Una fila en la tabla de la base de datos se envuelve en una clase, de manera que se asocian filas únicas de la base de datos con objetos del lenguaje de programación usado. Cuando se crea uno de estos objetos se añade una fila a la tabla de la base de datos, si se modifican sus atributos se actualiza dicha fila con los nuevos valores.

1.5.5. Arquitectura Modelo- Vista- Controlador

El Modelo Vista Controlador (*Model View Controller*, MVC por sus siglas en inglés) 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 interfaz de usuario y el código es el 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.

Modelo:

Esta es la representación específica de la información con la cual el sistema opera. Maneja los datos y controla sus transformaciones. Este no tiene conocimiento específico de los controladores y las vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el modelo y sus vistas y notificar a las vistas cuando cambia el modelo.

Vista:

Maneja la presentación visual de los datos representados por el modelo el cual es presentado en un formato adecuado para interactuar, usualmente la interfaz de usuario.

Controlador:

Responde a eventos, usualmente acciones del usuario, e invoca cambios en el modelo y probablemente en la vista. [4]

Generalmente el flujo que sigue el MVC es el siguiente:

1. El usuario interactúa con la interfaz de alguna forma (pulsa un botón, enlace).
2. El controlador recibe por parte de los objetos de la interfaz-vista la notificación de la acción solicitada por el usuario. Gestiona el evento que llega, frecuentemente a través de un gestor de eventos *handler* o *callback*.
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario.
4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario donde se reflejan los cambios en el modelo. El modelo no debe tener conocimiento directo sobre la vista.
5. La interfaz espera nuevas interacciones del usuario comenzando el ciclo nuevamente.

1.6. Conclusiones parciales

En este capítulo se hizo un estudio de la tesis precedente donde logró obtener una visión más ampliada del contenido del trabajo. Concluido este análisis, se verificaron algunos cambios realizados producto de la interacción con los clientes, además se expusieron las diferentes características del proceso de desarrollo de software a utilizar así como de las herramientas, lenguajes y tecnologías que protagonizarán el desarrollo del componente. Todo esto favoreció en la posterior realización del diseño e implementación del sistema debido que se pudo alcanzar una mayor preparación.

Capítulo 2: Diseño del sistema

2.1. Introducción

En el desarrollo de este capítulo se realiza una descripción del sistema a desarrollar donde se exponen los diferentes patrones de diseño a utilizar, se hace una representación gráfica de los diagramas de interacción, diagramas de clases del diseño con estereotipos web, diagrama entidad relación así como una descripción de cada uno de ellos.

2.2. Patrones de diseño

Son definidos como una solución reusable para un problema que comúnmente ocurre durante el diseño del software. Una solución para que sea considerada como patrón debe cumplir con determinadas características como son su efectividad resolviendo problemas, reusabilidad, lo que quiere decir que debe ser aplicable en diferentes circunstancias.

2.2.1. Patrones GRASP

Con el objetivo de alcanzar mayor calidad en el diseño se tuvieron en cuenta los siguientes patrones de diseño GRASP. Las responsabilidades están relacionadas con las obligaciones de un objeto en cuanto a su comportamiento.

2.2.1.1. Patrón Controlador

Problema: ¿Quién debería encargarse de atender un evento del sistema?

Solución: Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema, a una clase que represente una de las siguientes opciones:

- El “sistema” global (controlador de fachada).
- La empresa u organización global (controlador de fachada).

- Algo en el mundo real que es activo (por ejemplo, el papel de una persona) y que pueda participar en la tarea (controlador de tareas).

Aplicación**:** En el sistema cada requisito funcional cuenta con una clase DatEval[ ]Controller ó NomEval[ ]Controller que es la encargada de manejar los eventos y la lógica del negocio del sistema relacionados al mismo.

2.2.1.2. Patrón Experto

Problema**:** ¿Quién debiera ser el responsable de conocer la información?

Solución: La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada.

Hay que tener en cuenta que esto es aplicable mientras estemos considerando los mismos aspectos del sistema:

- Lógica de negocio.
- Persistencia a la base de datos.
- Interfaz de usuario.

Aplicación: El sistema implementa este patrón en forma de servicios; ejemplo de su aplicación: la clase plantillasController.php al necesitar datos del nomenclador estado y de las planillas evaluativas obtiene la información a través de las clases NomEvalEstado.php y DatEvalplanillasevaluativas.php, las cuales son las encargadas del acceso a datos.

2.2.1.3. Patrón Creador

Problema: ¿Quién debería ser responsable de crear una nueva instancia de alguna clase?

Solución: Este patrón como su nombre lo indica es el que crea, el guía la asignación de responsabilidades relacionadas con la creación de objetos, se asigna la responsabilidad de que una clase B cree un Objeto de la clase A solamente cuando:

- B contiene a A.
- B es una agregación (o composición) de A.
- B almacena a A.
- B tiene los datos de inicialización de A (datos que requiere su constructor).
- B usa a A.

Aplicación**:** Las clases [ ]Controller son responsables de crear instancias de las clases DatEval[ ]Model ó NomEval[ ]Model para realizar las funciones de insertar, modificar y eliminar datos en la base de datos e instancias de las clase DatEval[ ] ó NomEval[ ] en caso de necesitar cargarlos.

2.2.1.4. Patrón de Alta Cohesión

Problema: ¿Cómo mantener la complejidad dentro de límites manejables?

Solución: Asignar una responsabilidad de modo que la cohesión siga siendo alta.

Aplicación: Cada elemento del diseño realiza una labor única dentro del sistema, no desempeñada por el resto de los elementos, aunque las planillas evaluativas están relacionadas con los indicadores, ya sean de texto o de imagen, no se decide la implementación de una clase general sino la creación de clases para cada proceso de gestión (plantillasController.php e indicadoresController.php) evitando la sobrecarga, compresión y reutilización de funcionalidades.

2.2.2. Patrones de arquitectura

Ofrecen soluciones bien establecidas a problemas arquitecturales dentro de la Ingeniería del software; describen los elementos y las relaciones junto a especificaciones de cómo pueden ser usados. La Arquitectura del Software comprende los primeros pasos de las decisiones del diseño para un sistema, por lo tanto, se acude a la separación de capas para contrarrestar las consecuencias de las modificaciones futuras.

2.2.2.1. Modelo vista controlador (MVC)

El modelo vista controlador es un patrón de arquitectura de software que se ve comúnmente en aplicaciones web donde la vista es la página HTML y el código que suministra de datos dinámicos a la página, el modelo es el sistema de gestión de base de datos y el controlador representa la lógica del negocio.

El acceso a dato está implementado por clases .php denominadas como DatEval[ ] ó NomEval[ ] las cuales son encargadas de la lectura de datos y las DatEval[ ]Model ó NomEval[ ]Model de las funcionalidades de insertar, modificar y eliminar.

Las vistas son realizadas por ficheros JavaScript como plantillas.js y escalas.js los cuales son llamados desde ficheros como plantillas.phtml y escalas.phtml respectivamente. Las clases .php controladoras serán representadas como DatEval[ ]Controller ó NomEval[ ]Controller, estas son las responsables de recibir las peticiones de los usuarios y hacer llamadas a las funciones necesarias para dar cumplimiento a las mismas.

2.3. Diagramas de clases del diseño

El diagrama de clases es el diagrama principal de diseño y análisis para un sistema. En él, la estructura de clases del sistema se especifica, con relaciones entre clases y estructuras de herencia. Durante el análisis del sistema, el diagrama se desarrolla buscando una solución ideal. Durante el diseño se usa el mismo diagrama y se modifica para satisfacer los detalles de las implementaciones.

2.3.1. Diagrama de clases genérico de las vistas

El objetivo de este diagrama es representar de forma global las vistas de la aplicación haciendo uso del ExtJS donde se resaltan las clases fundamentales.

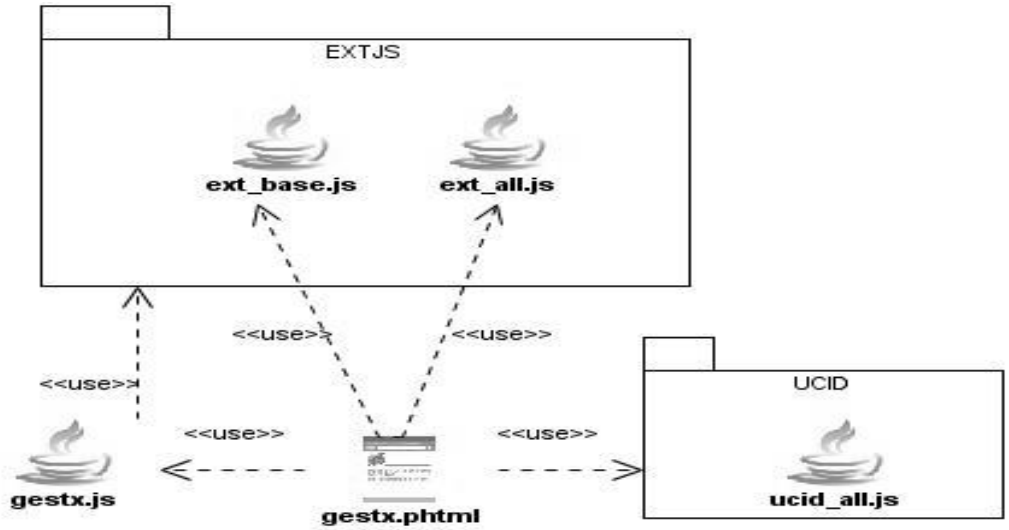


Figura 4: Diagrama de clases genérico de las vistas

Descripción de las clases

Nombre: ext-base.js	
Tipo de clase: interfaz	
Descripción:	Encargada del manejo de las solicitudes y respuestas, trabajo con Ajax y manejo de componentes de ExtJS. Esta incluida en el paquete original.

Nombre: ext-all.js	
Tipo de clase: interfaz	
Descripción:	Es la encargada de la creación de los componentes visuales de la vista. Esta incluida dentro de las clases que trae ExtJS.

Nombre: gestX.phtml

Tipo de clase: interfaz	
Descripción:	Representa la vista que se muestra al usuario.

Nombre: gestx.js	
Tipo de clase: interfaz	
Descripción:	Fichero js con las funciones Java Script asociadas a la vista. Aquí se establece la referencia a las clases de ExtJS.

2.3.2. Diagrama de clases genérico

El objetivo de este diagrama es dar un panorama general de la estructura de los diagramas de clases de diseño del sistema.

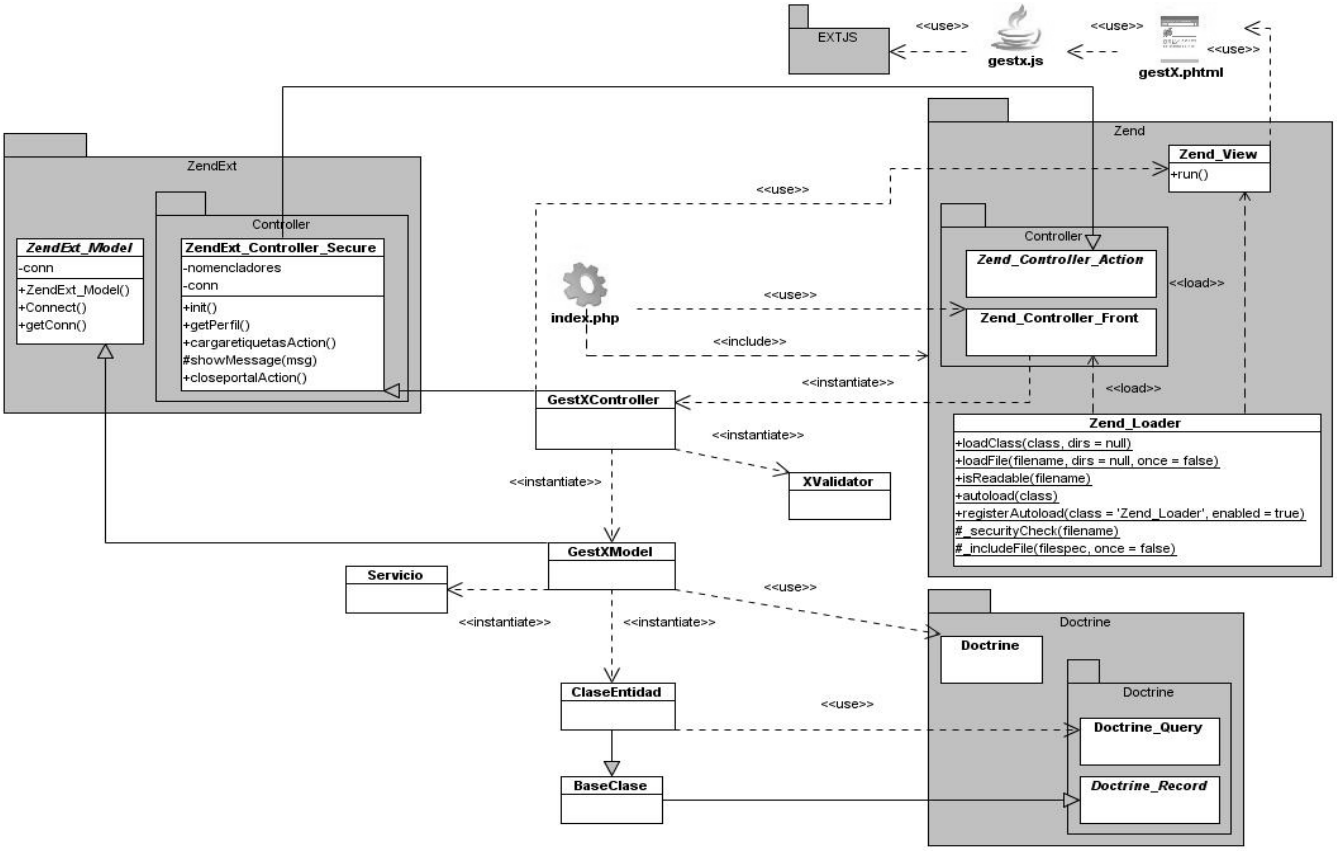


Figura 5: Diagrama de clases genérico general

Descripción de las clases

Nombre: Zend_Controller_Action.php	
Tipo de clase: controladora	
Descripción:	De esta clase deben heredar Zend_Controller_Secure, en ella se incluyen numerosas funcionalidades comunes.

Nombre: Zend_Controller_Secure.php	
Tipo de clase: controladora	
Descripción:	De esta clase deben heredar todas las controladoras. Es un controlador de acciones personalizado e integrado a seguridad.

Nombre: GestXController.php	
Tipo de clase: controladora	
Descripción:	Representa el controlador del Caso de Uso en cuestión.

Nombre: Zend_View.php	
Tipo de clase: controladora.	
Descripción:	Es la clase de Zend Framework encargada del manejo de las vistas.

Nombre: Zend_Controller_Front.php	
Tipo de clase: controladora.	
Descripción:	Representa el controlador frontal de la aplicación. Se encarga de manejar las solicitudes y respuestas. Es manejado por index.

Nombre: Index.php

Tipo de clase: servidora	
Descripción:	Constituye el único punto de acceso a la aplicación, conjuntamente con la clase Zend_Loader y Zend_Controller_Front se encarga del funcionamiento de la aplicación, atención a solicitudes y respuestas.

Nombre: ZendExt_model.php	
Tipo de clase: Modelo	
Descripción:	Se encarga de gestionar el modelo y el negocio.

Nombre: GestXModel.php	
Tipo de clase: Modelo	
Descripción:	En ella se implementan las funciones básicas para el trabajo con la base de dato como son insertar, modificar y eliminar.

Nombre: BaseClase.php	
Tipo de clase: Modelo	
Descripción:	Representa la clase modelo del Caso de Uso, es generada dinámicamente por Doctrine, por lo general lleva el mismo nombre de la tabla asociada. Hereda de doctrine_record.

Nombre: Clase.php

Tipo de clase: Modelo	
Descripción:	En ella se implementan todas las funcionalidades para obtener datos de la base de datos. Hereda de BaseClase.

2.3.3. Diagramas de clases específicos

Estos diagramas representan lo específico, es decir, omite los paquetes y clases comunes del diagrama de clases genérico.

2.3.3.1. Diagrama de clases del diseño Gestionar Planilla

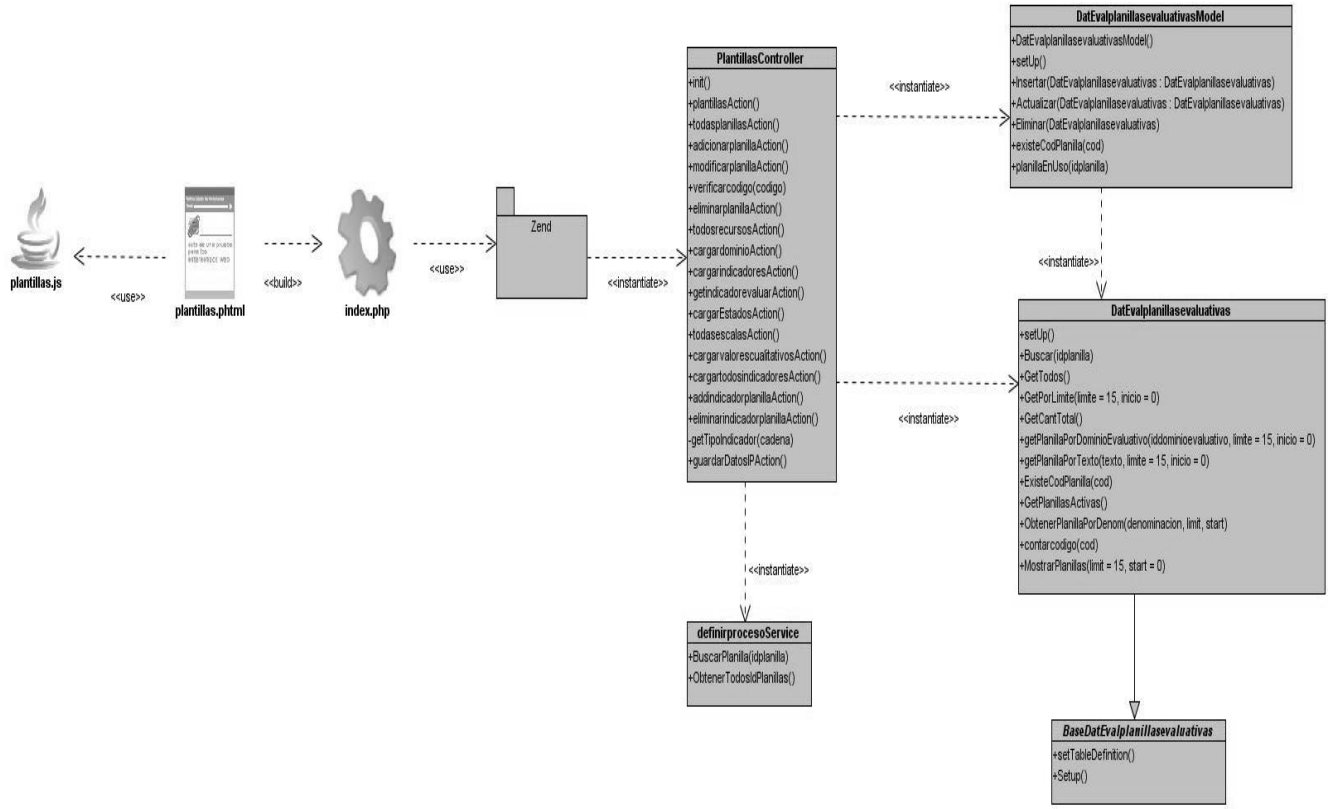


Figura 6: Diagrama de clases del diseño Gestionar Planilla

Descripción de las clases

Nombre: plantillas.js	
Tipo de clase: interfaz.	
Para cada responsabilidad:	
Nombre:	cargarInterfaz()
Descripción:	Función principal, encargada de mostrar la interfaz.
Nombre:	VentanaPlanilla(accion)
Descripción:	Muestra una ventana para adicionar o modificar según la acción que se realice.
Nombre:	AddModPlanilla(accion, boton)
Descripción:	Adiciona o modifica según la acción y el botón que se especifique.
Nombre:	VentanaGestionarIndicador()
Descripción:	Muestra una ventana para gestionar los indicadores a una planilla seleccionada.
Nombre:	CargarDatosIP(datosIP)
Descripción:	Carga los datos del indicador de una planilla seleccionado.
Nombre:	VerValoresEscala(idescala)
Descripción:	Muestra una ventana con los valores de una escala seleccionada.
Nombre:	BuscarPlanillaPorDenom(denomPlanilla)

Descripción:	Busca las planillas por la denominación pasada por parámetro.
Nombre:	MostrarVentanaIndicador(accionI, tipoI)
Descripción:	Muestra una ventana para adicionar los indicadores.
Nombre:	AddIndicadorPlanilla(accion, tipoI)
Descripción:	Adiciona un indicador a la planilla.
Nombre:	GuardarDatosIP($idindicadorplanilla)
Descripción:	Guarda los datos de los indicadores de la planilla.
Nombre:	EliminarIP()
Descripción:	Elimina el indicador de la planilla seleccionado.
Nombre:	representarImagen()
Descripción:	Para mostrar imágenes en un gridPanel.

Nombre: PlantillasController.php	
Tipo de clase: controladora	
Para cada responsabilidad:	
Nombre:	init()
Descripción:	Inicializa el controlador.
Nombre:	todasplanillasAction()

Descripción:	Devuelve un arreglo con todas las planillas existentes en la base de datos.
Nombre:	adicionarplanillaAction()
Descripción:	Adiciona una planilla a la base de datos.
Nombre:	modificarplanillaAction()
Descripción:	Modifica una planilla en la base de datos.
Nombre:	eliminarplanillaAction()
Descripción:	Elimina un indicador de imagen de la base de datos.
Nombre:	verificarcodigo($codigo)
Descripción:	Verifica si el código $codigo ya existe en la base de datos.
Nombre:	todosrecursosAction()
Descripción:	Devuelve todos los recursos existentes.
Nombre:	cargardominioAction()
Descripción:	Devuelve los dominios asociados al recurso seleccionado.
Nombre:	cargarindicadoresAction()
Descripción:	Devuelve los indicadores asociados a la planilla seleccionada.
Nombre:	cargarEstadosAction()
Descripción:	Devuelve los estados existentes.

Nombre:	todasescalasAction()
Descripción:	Devuelve todas las escalas seleccionadas.
Nombre:	getTipoEscalaAction()
Descripción:	Devuelve el tipo de escala de una escala.
Nombre:	cargarvaloresescalaAction()
Descripción:	Devuelve los valores de una escala.
Nombre:	cargartodosindicadoresAction()
Descripción:	Devuelve todos los indicadores existentes.
Nombre:	addindicadorplanillaAction()
Descripción:	Adiciona un indicador a una planilla.
Nombre:	eliminarindicadorplanillaAction()
Descripción:	Elimina un indicador de una planilla.
Nombre:	guardarDatosIPAction()
Descripción:	Guarda los datos de un indicador de una planilla.

Nombre: DatEvalplanillasevaluativasModel.php
Tipo de clase: modelo
Para cada responsabilidad:

Nombre:	Insertar(DatEvalplanillasevaluativas $DatEvalplanillasevaluativas)
Descripción:	Inserta el nomenclador contenido en $DatEvalplanillasevaluativas en la base de datos.
Nombre:	Actualizar(DatEvalplanillasevaluativas $DatEvalplanillasevaluativas)
Descripción:	Actualiza el nomenclador contenido en $DatEvalplanillasevaluativas en la base de datos.
Nombre:	Eliminar($DatEvalplanillasevaluativas)
Descripción:	Elimina el nomenclador contenido en $DatEvalplanillasevaluativas en la base de datos.
Nombre:	existeCodPlanilla($cod)
Descripción:	Verifica si ya existe el código que se va a adicionar.

Nombre: DatEvalplanillasevaluativas.php	
Tipo de clase: modelo	
Para cada responsabilidad:	
Nombre:	Buscar($idplanilla)
Descripción:	Devuelve la planilla con id = $ idplanilla.
Nombre:	GetTodos()
Descripción:	Devuelve todas las planillas existentes en la base de datos.

Nombre:	GetCantTotal()
Descripción:	Devuelve la cantidad total de planillas existentes en la base de datos.
Nombre:	GetPorLimite($limite = 15,$inicio = 0)
Descripción:	Devuelve las planillas existentes en la base de datos del $inicio al $limite.
Nombre:	getPlanillaPorDominioEvaluativo($iddominioevaluativo,$limite = 15,$inicio = 0)
Descripción:	Devuelve las planillas que pertenecen al dominio pasado por parámetro.
Nombre:	getPlanillaPorTexto($texto,$limite = 15,$inicio = 0)
Descripción:	Devuelve las planillas con denominación = $texto.
Nombre:	ExisteCodPlanilla($cod)
Descripción:	Verifica si el código a adicionar existe.
Nombre:	GetPlanillasActivas()
Descripción:	Devuelve las planillas con estado Activa.

Para ver los diagramas de clases del diseño restantes *Ver Anexos*.

2.4. Diagramas de secuencia

Un diagrama de secuencia es una forma de diagrama de interacción que muestra los objetos como líneas de vida a lo largo de la página y con sus interacciones en el tiempo representadas como mensajes dibujados como flechas desde la línea de vida origen hasta la línea de vida destino. Los diagramas de secuencia son buenos para mostrar qué objetos se comunican con qué otros objetos y qué mensajes disparan esas comunicaciones. [5]

2.4.1. Diagrama de secuencia Listar Planillas

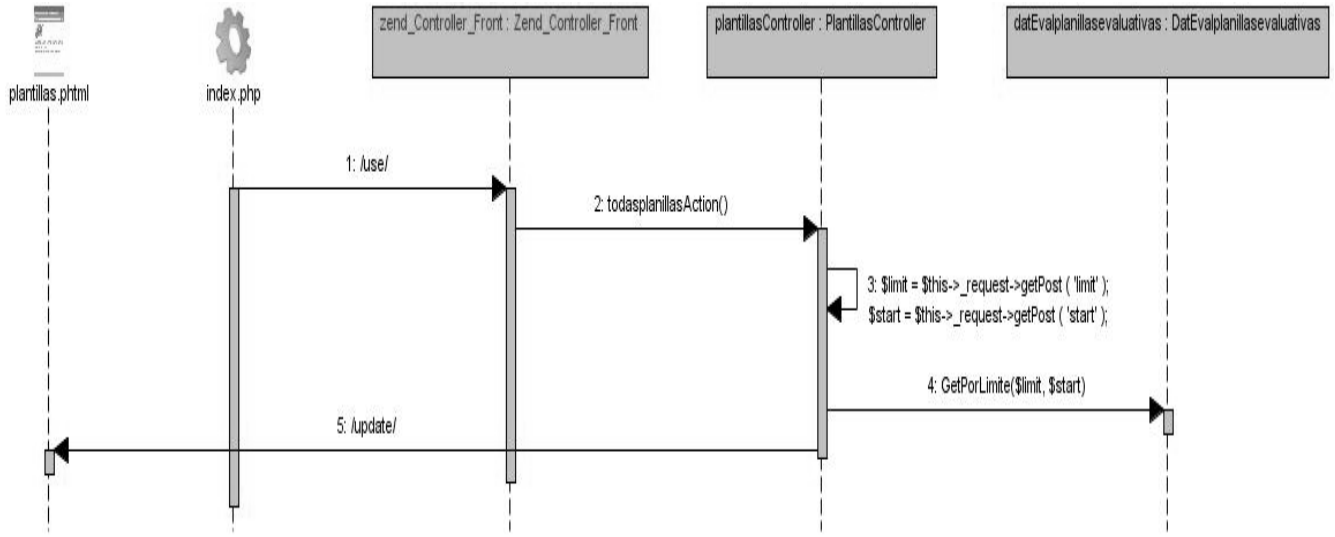


Figura 7: Diagrama de secuencia Listar Planillas

2.4.2. Diagrama de secuencia Adicionar Planilla

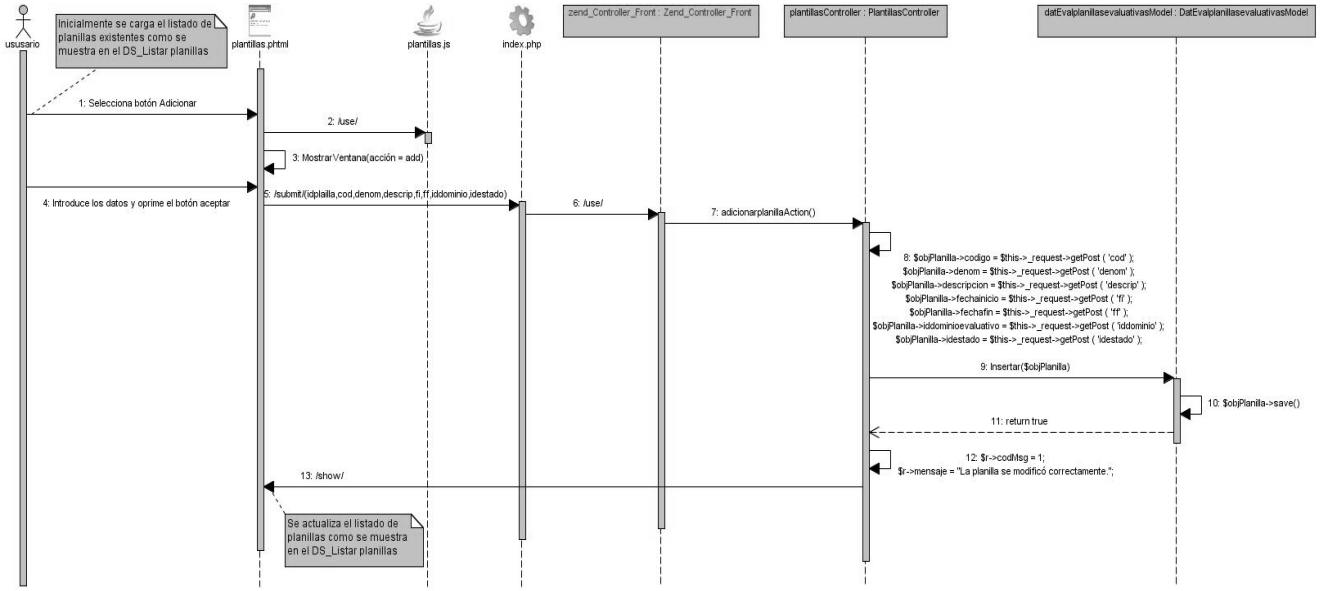


Figura 8: Diagrama de secuencia Adicionar Planilla

2.4.3. Diagrama de secuencia Modificar Planilla

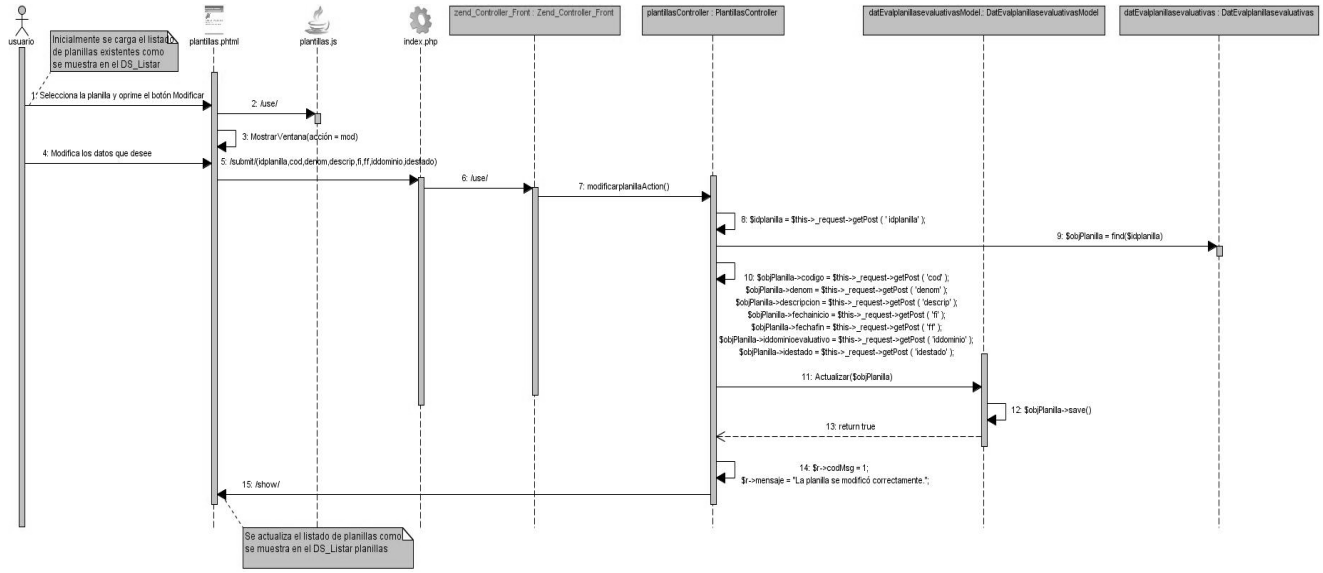


Figura 9: Diagrama de secuencia Modificar Planilla

2.4.4. Diagrama de secuencia Eliminar Planilla

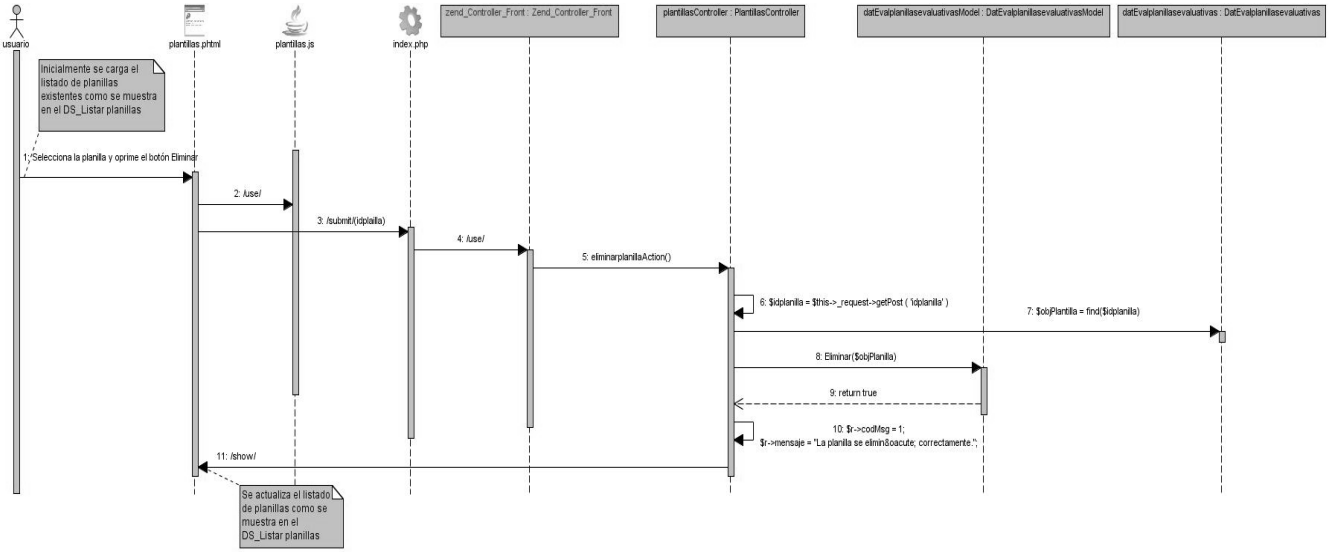


Figura 10: Diagrama de secuencia Eliminar Planilla

Para ver los diagramas de secuencia restantes *Ver Anexos*.

2.5. Diagrama Entidad Relación de la Base de Datos

En la figura 11 se muestra el diagrama entidad relación de la base de datos del módulo de evaluaciones donde se resaltan, en color gris claro, las tablas relacionadas con el componente de planillas evaluativas. Con dicho modelo se logra un eficiente guardado de los datos.

Figura 11: Diagrama Entidad Relación de la Base de Datos

Para conocer la descripción de las tablas de la base de datos *Ver Anexos* .

2.6. Interfaz

Los componentes a implementar están enmarcados dentro del subsistema Evaluaciones de la Línea Capital Humano. *(Ver Anexos)*

Para la confección de una planilla con todos los aspectos y características que esta debe tener se hace necesario crear las condiciones antes de realizar esta operación. Se debe crear el recurso al que pertenecerá la planilla a elaborar y el dominio al que hará parte este recurso. Los valores cualitativos, valores imágenes o de rango numérico que conformarán las escalas por los cuales serán evaluados los distintos indicadores, ya sean de texto o de imagen, serán los que se deberán establecer posteriormente. El sistema le dará al usuario, según sus privilegios, la opción de acceder al menú que contendrá las funcionalidades para la gestión de todos los elementos necesarios para conformar la planilla.

Recursos y dominios

El usuario accederá al sistema donde se encontrará a la izquierda los recursos y a la derecha los dominios representados en forma de árbol con el trabajo administrativo de los mismos como adicionar, modificar y eliminar. *(Ver Anexos)* Para adicionar un dominio es imprescindible seleccionar el recurso al cual se le va a agregar el nuevo dominio y del árbol de dominios el nodo padre del dominio a insertar además de especificar un grupo de datos necesarios.

Valores cualitativos, valores imágenes

El usuario accederá al sistema donde procederá a adicionar, modificar o eliminar dichos valores. Para adicionar es imprescindible insertar un grupo de datos necesarios. (*Ver Anexos*) Estos valores formarán parte después de las diferentes escalas.

Escalas

El usuario accederá al sistema donde procederá a adicionar, modificar o eliminar dichas escalas según prefiera. *(Ver Anexos)* Para adicionar es imprescindible insertar un grupo de datos necesarios entre los cuales se encuentra el tipo de escala. Si el tipo de escala es de:

Rango numérico: Se especifica el valor inicial, valor final y paso.

Cualitativa: Aparecerá una lista de valores cualitativos creados previamente donde se escogerán según hagan falta para conformar la escala.

Imagen: Aparecerá una lista de imágenes previamente insertadas en la base de datos donde se escogerán según hagan falta para conformar la escala.

Indicadores

El usuario accederá al sistema donde podrá adicionar, modificar y eliminar indicadores ya sea de texto o de imagen. *(Ver Anexos)* Estos indicadores formarán parte de las planillas evaluativas.

Planillas

Después de realizadas estas operaciones se pasará a crear la planilla *(Ver Anexos)* donde se procederá a insertar un grupo de datos necesarios como son código, denominación, dominio, estado, una breve descripción de para que está destinada esta planilla, además se puede modificar, eliminar o gestionar los indicadores asociados a ella una vez ya creada. *(Ver Anexos)* Gestionar los indicadores significa conformar la planilla con todos los indicadores que esta requiere, a los cuales se les asigna la escala por la cual van a ser evaluados posteriormente.

2.7. Tratamiento de errores

El proyecto consta de un directorio con todos los archivos XML que se van creando para su perfeccionamiento dentro de los cuales se encuentra:

- exceptions.xml: Se realiza con el objetivo de tratar el lanzamiento de excepciones. Aquí se trabaja con 25 excepciones donde cada una tiene un código, un tipo y el idioma en que se muestra el mensaje, este mensaje tiene a su vez un nombre y una descripción.
- validation.xml: Se valida que el tipo de los datos que se entraron por parámetros sea cierto, comprobando que un nombre y una edad que se introduzcan sean válidos, en el caso del nombre se comprueba que el nombre sólo contenga caracteres y la edad enteros positivos y además que tenga un rango razonable, en caso de que no cumpla con la expresión regular que se especificó en el fichero expressions.xml (enterospos para la edad y letras para los nombres) se lanza el mensaje de error.
- expressions.xml: Aquí se especifican los tipos de errores que pueden existir cuando se pasan por parámetros tipos de datos que no son correctos, por ejemplo cuando en un campo envían

números y sólo deben ser letras. Se definen expresiones regulares para los enteros positivos que sólo deben contener los dígitos del 0 hasta el 9. De igual forma se definen las expresiones regulares en datos que sólo deben contener letras, los datos que incluyen números y letras únicamente, los que deben ser direcciones de correo electrónico, las expresiones que deben contener números reales y las expresiones de fecha.

- tipos_excepciones.xml: Especifica los tipos de excepciones que pueden ocurrir y la localización del archivo que tratará dicha excepción dentro del framework ZendExt. Existen 4 tipos de excepciones etiquetados, L que se transfiere a ZendExt_Exception_Log y se refiere a las excepciones que al ocurrir se les dará un determinado mensaje al usuario que será sencillo y poco específico y en el archivo interno del sistema se guardará otro texto especificando hora, fecha, el código de la excepción, su descripción y el texto real y explícito del error que ocurrió, o sea, el texto que muestra donde fue el error. Las excepciones de tipo P son aquellas que se tratan en el directorio ZendExt_Exception_Presentation es decir en la carpeta del framework ZendExt hay una que se llama Exception y dentro están el archivo .php llamado Presentation, y son las que se muestran únicamente al usuario. La excepciones LP realizan su tratamiento en ZendExt_Exception_LogPresentation y son en las que se guarda el mensaje de igual forma tanto del lado del servidor como del lado del cliente, o sea, el mismo mensaje que se le muestra al usuario es el que se guarda en el fichero del servidor y por último las de tipo B que se transfieren al fichero ZendExt_Exception_Blind que son aquellas excepciones ciegas de las cuales no se tiene ninguna información.

2.8. Seguridad del sistema

La seguridad es un aspecto fundamental y de mucho peso en la implementación de cualquier aplicación, esta verifica que la información no pueda ser filtrada, es decir, visualizada por personal no autorizado, utilizada con fines impropios o contrarios a las políticas del cliente. En nuestros días la idea de crear un software sin ningún mecanismo de seguridad no se concibe, sería un total fracaso.

En el sistema, la seguridad se gestiona a través de un módulo ya implementado del cual se consumen servicios. A través de este se pueden crear usuarios con niveles de acceso específicos como por

ejemplo realizar determinada acción (adicionar, modificar, eliminar, listar, etc.) en una o varias funcionalidades (planillas, indicadores, escalas, recursos, etc.) de la aplicación.

2.9. Concepción de la ayuda

La ayuda del sistema está concebida para que en cada interfaz aparezca un botón Ayuda donde se dará una explicación detallada de las acciones a realizar en cada uno de los componentes. El objetivo de esta es que sirva de guía para el usuario en cuestión.

2.10. Conclusiones parciales

Con el desarrollo de este capítulo se obtuvieron los diferentes diagramas de interacción siendo estos diagramas de clases y de secuencia de cada uno de los componentes a implementar los cuales dan a conocer su grado de complejidad. Se realizan descripciones de las clases que se utilizan así como de las tablas de la base de datos contribuyendo así a un mejor entendimiento de las mismas. Se hace un resumen del tratamiento de errores y seguridad presente en la aplicación siendo estos aspectos fundamentales a tener en cuenta durante su progreso. Todo esto contribuyó a la realización de una mejor implementación del sistema ya que un diseño bien definido hace que el trabajo del desarrollador sea más rápido y eficiente.

Capítulo 3: Implementación y prueba

3.1. Introducción

En este capítulo se traduce el diseño definido antes de la implementación del sistema. Además, se realiza una representación gráfica de los componentes que intervienen así como la integración de los mismos con otros componentes. Se hace una descripción de los diseños de casos de pruebas.

3.2. Diagrama de componentes

Un *diagrama de Componentes* ilustra los fragmentos de software, controladores embebidos, etc. que conformarán un sistema. Un diagrama de componentes tiene un nivel de abstracción más elevado que un diagrama de clase - usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Estos son bloques de construcción, como así eventualmente un componente puede comprender una gran porción de un sistema. [6]

3.2.1. Diagrama de componentes general

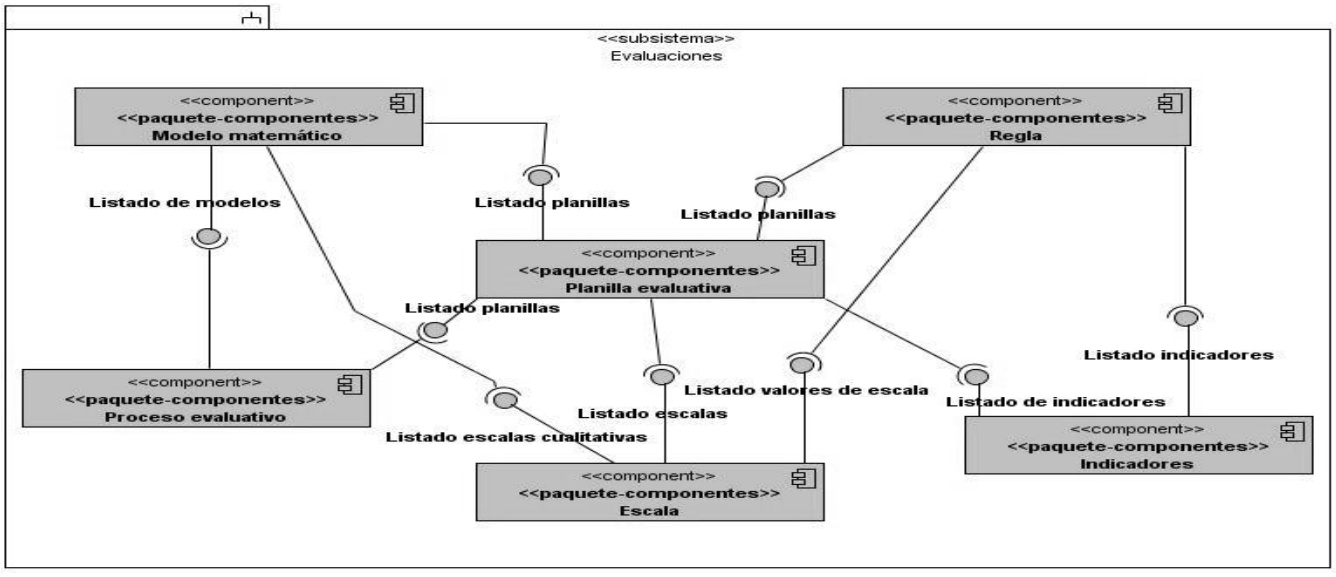


Figura 12: Diagrama de componentes

3.3. Matriz de integración de componentes interna

	Componentes Internos				
Componentes Internos	**Modelos matemáticos**	**Reglas**	**Planillas**	**Indicadores**	**Escalas**
Proceso evaluativo	Modelos matemáticos	—	Planillas	—	—
Modelos matemáticos	—	Reglas	Planillas	—	Escalas cualitativas
Reglas	—	—	Planillas	Indicadores	Valores de la escala
Planillas	—	—	—	Indicadores	Escalas

3.4. Pruebas

Uno de los mayores problemas que se enfrentan en la actualidad con el desarrollo de software es la calidad de los mismos. Debido a esto el proceso de pruebas es sin duda uno de los aspectos fundamentales para medir la eficacia de una aplicación informática. Antes de la liberación de un producto es necesario tener la certeza de que se encuentra libre de errores aunque se considera que es prácticamente imposible estar seguro de esto.

3.4.1. Pruebas de caja negra

Se realizarán pruebas de caja negra las cuales se centran principalmente en los requisitos funcionales del sistema. Con estas pruebas se intenta encontrar casos en que el componente no cumple con las especificaciones, por tal motivo se le denominan pruebas funcionales, donde el probador se limita a

suministrarle datos como entrada y estudiar si la salida es correcta, así como que la integridad de la información externa se mantiene sin tener mucho en cuenta la estructura interna del software.

3.4.1.1. Técnica de partición de equivalencia

Roger S. 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. [7]

3.4.1.2. Técnica de análisis de valores límite

El análisis de valores límite 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, llega a la elección de casos de prueba en los extremos de la clase. En lugar de centrarse solamente en las condiciones de entrada, obtiene casos de prueba también para el campo de salida. [7]

3.4.2. Descripción de los casos de prueba

3.4.2.1. Gestionar planillas evaluativas

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Gestionar planillas evaluativas.	El objetivo de este requisito es adicionar planillas evaluativas.	EP 1.1 Adicionar planilla correctamente.	- Se selecciona la opción Evaluaciones/Planillas/Planillas evaluativas en el menú principal del sistema. - Se muestra la interfaz Gestionar planillas evaluativas con todas las planillas existentes en ese momento. - Se oprime el botón **Adicionar**. - Se muestra la interfaz Adicionar planilla con una serie de campos para introducir los datos de la misma (código, denominación, dominio, estado, fecha inicio, fecha fin, descripción). - Se introduce el código, denominación, se escoge el dominio al cual va a pertenecer la planilla, el estado aparece por defecto Elaboración sin opción para cambiarlo, la fecha de inicio aparece por defecto con la actual aunque se puede cambiar, se selecciona la fecha fin (opcional) y se da una breve descripción (opcional). - Se oprime el botón **Aceptar**.

		EP 1.2: Adicionar planilla incorrectamente.	- Se selecciona la opción Evaluaciones/Planillas/Planillas evaluativas en el menú principal del sistema. - Se muestra la interfaz Gestionar planillas evaluativas con todas las planillas existentes en ese momento. - Se oprime el botón **Adicionar**. - Se muestra la interfaz Adicionar planilla con una serie de campos para introducir los datos de la misma (código, denominación, dominio, estado, fecha inicio, fecha fin, descripción). - El usuario deja campos como código, denominación y dominio en blanco o llena los campos con datos incorrectos.
		EP 1.3: Cancelar operación.	- Se selecciona la opción Evaluaciones/Planillas/Planillas evaluativas en el menú principal del sistema. - Se muestra la interfaz Gestionar planillas evaluativas con todas las planillas existentes en ese momento. - Se oprime el botón **Adicionar**. - Se muestra la interfaz Adicionar planilla con una serie de campos para introducir los datos de la misma (código, denominación, dominio, estado, fecha inicio, fecha fin, descripción). - Se introduce el código, denominación, se escoge el dominio al cual va a pertenecer

		la planilla, el estado aparece por defecto Elaboración sin opción para cambiarlo, la fecha de inicio aparece por defecto con la actual aunque se puede cambiar, se selecciona la fecha fin (opcional) y se da una breve descripción (opcional). - Se oprime el botón **Cancelar**. - El sistema cancela las operaciones.

Descripción de las variables

No	Nombre de campo	Clasificación	Puede ser nulo	Descripción
1.	Código	Textfield	No	Permite introducir un código único para la planilla
2.	Denominación	Textfield	No	Permite introducir la denominación
3.	Dominio	Treepanel	No	Permite seleccionar un dominio
4.	Estado	Combobox	No	Muestra el estado con que se va a crear la planilla
5.	Fecha inicio	Datefield	No	Permite seleccionar la fecha de inicio.
6.	Fecha fin	Datefield	Sí	Permite seleccionar la fecha fin.
7.	Descripción	Html editor	Sí	Permite introducir una descripción

Juego de datos a probar

Id del escenario	Escenario	Código	Denominación	Dominio	Estado	Fecha inicio	Fecha fin	Descripción	Respuesta del sistema	Resultado de la prueba
EP 1.1	Adicionar planilla correctamente	V(56asd343sfrr)	V(fgd5420/*-+.)	V(Test)	V(Elaboración)	V(10/05/2010)	V(20/05/2010)	V(fgd520/*-+.)	Se adiciona la planilla evaluativa y se muestra el mensaje de información	Satisfactorio
		V(56asd343sfrr)	V(fgd5420/*-+.)	V(Test)	V(Elaboración)	V(10/05/2010)	V(vacío)	V(vacío)	Se adiciona la planilla evaluativa y se muestra el mensaje de información	Satisfactorio
EP 1.2	Adicionar planilla incorrectamente	I(vacío)	I(vacío)	I(vacío)	V(Elaboración)	V(10/05/2010)	V(vacío)	V(vacío)	Se muestra el campo en color rojo y un mensaje indicando que es obligatorio llenarlo	Satisfactorio

		V(56as/*-+._)	I(vacío)	V(Test)	V(Elaboración)	V(10/05/2010)	V(vacío)	V(vacío)	Se muestra el campo en color rojo y un mensaje indicando que es obligatorio llenarlo	Satisfactorio
		V(56as/*-+._)	V(fgd5420/*-+.)	I(vacío)	V(Elaboración)	V(10/05/2010)	V(vacío)	V(vacío)	Se muestra el campo en color rojo y un mensaje indicando que es obligatorio llenarlo	Satisfactorio
		I(vacío)	V(fgd5420/*-+.)	V(Test)	V(Elaboración)	V(10/05/2010)	V(vacío)	V(vacío)	Se muestra el campo en color rojo y un mensaje indicando que es obligatorio llenarlo	Satisfactorio
EP 1.3	Cancelar operación.	NA	NA	NA	V(Elaboración)	V(10/05/2010)	NA	NA	El sistema cancela las operaciones	Satisfactorio
		V(56as/*-+._)	V(fgd5420/*-+.)	V(Test)	V(Elaboración)	V(10/05/2010)	V(20/05/2010)	V(fgd520/*-+.)	El sistema cancela las operaciones	Satisfactorio

Para ver los escenarios de prueba restantes ir a Anexos.

Se realizaron 12 diseños de casos de prueba donde se encontraron 7 no conformidades en la primera iteración las cuales fueron solucionadas en la segunda iteración obteniendo un resultado satisfactorio para cada una de las combinaciones de datos realizadas por escenario.

3.5. Conclusiones parciales

En este capítulo se desarrolló el diagrama de componentes y la matriz de integración de los mismos lo cual refleja la combinación de estos, los diferentes servicios que consumen unos de otros, es decir, como está estructurado el módulo completo de evaluaciones al cual pertenece el componente desarrollado. Últimamente se ejecutan las pruebas pertinentes. Todo lo realizado en este capítulo da lugar a una mayor comprensión del software desde el punto de vista de implementación además de la obtención de una aplicación libre de errores, lista para ser usada.

Conclusiones

La informática es una asignatura que necesita de mucho estudio y dedicación. Trabajos como estos ayudan al desarrollo de los conocimientos en esta materia donde se eleva la formación profesional de las personas involucradas.

Con la elaboración de este trabajo se logró:

- ✓ Realizar el diseño teórico de la investigación.
- ✓ Realizar un estudio del estado del arte de los principales procesos de evaluación que se llevan a cabo en el país con el cual se pudo lograr una mejor visión de los mismos.
- ✓ Dar cumplimiento de forma satisfactoria a la realización de un diseño que posteriormente sería clave para la implementación del componente donde se obtuvieron los diferentes diagramas de clases del diseño y secuencia.
- ✓ Desarrollar la aplicación con todos los requisitos especificados por los clientes.
- ✓ Validar el software a través de las pruebas pertinentes con el objetivo de librar el mismo de posibles errores los cuales estarían en contra de la conformidad de los clientes.

Recomendaciones

Luego de la presentación del estudio realizado que finaliza con la implementación del componente de configuración de las planillas evaluativas se recomienda:

- Realizar el despliegue de la aplicación para comprobar si cumple con las expectativas de los clientes.
- Hacer uso de la aplicación en los diferentes órganos para así poder agilizar los procesos de evaluaciones a realizar.
- Seguir perfeccionando la aplicación de acuerdo con las nuevas necesidades que van surgiendo en el transcurso del tiempo.

Bibliografía

1. **Morales, Alfredo.** *Resolución No. 21/2007 del Ministro de Trabajo y Seguridad Social.* La Habana : Gaceta Oficial de la República, 2007.

2. **Enterprise Architect.** Guía de Usuario de Enterprise Architect 7.0. *Guía de Usuario de Enterprise Architect 7.0.* [En línea] Enterprise Architect. [Citado el: 6 de abril de 2010.] http://www.sparxsystems.com.ar/download/ayuda/index.html?componentdiagram.htm.

3. **Garnet, Joan.** Doctrine: ORM Ooen Source para PHP 5.2+. *Doctrine: ORM Ooen Source para PHP 5.2+.* [En línea] 25 de octubre de 2007. [Citado el: 20 de febrero de 2010.] http://www.joangarnet.com/blog/?p=415..

4. **Martín, Sergio, y otros.** E Boletín N°8 Rama de Estudiantes de IEEE-UNED. [En línea] 2007. [Citado el: 10 de febrero de 2010.] http://www.ieec.uned.es/ieee/investigacion/ieee_dieec/sb/boletin_8_Octubre_2007.pdf.

5. **Weidner, Thomas.** Zend Framework. [En línea] 2007. [Citado el: 20 de febrero de 2010.] http://www.slideshare.net/thomasw/phpbootcamp-zend-framework?src=related_normal&rel=184203.

6. **UCID.** *Estandar para el diseño de interfaces v1.1.* 2008.

7. —. *Normas y estándares de codificación del ERP.* 2008.

8. **Sparx Systems.** Sparx Systems - Tutorial UML 2 - Diagrama de Secuencia. *Sparx Systems - Tutorial UML 2 - Diagrama de Secuencia.* [En línea] 2007. [Citado el: 13 de abril de 2010.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_sequencediagram.html.

9. **Rojas, Johanna y Barrios, Emilio.** Métodos de prueba de caja negra. *Métodos de prueba de caja negra.* [En línea] 2007. [Citado el: 24 de Mayo de 2010.] http://www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node28.html.

Referencias bibliográficas

1. **Morales, Alfredo.** *Resolución No. 21/2007 del Ministro de Trabajo y Seguridad Social.* La Habana : Gaceta Oficial de la República, 2007.

2. **Weidner, Thomas.** Zend Framework. [En línea] 2007. [Citado el: 20 de 2 de 2010.] http://www.slideshare.net/thomasw/phpbootcamp-zend-framework?src=related_normal&rel=184203.

3. **Garnet, Joan.** Doctrine: ORM Open Source para PHP 5.2+. [En línea] 25 de 10 de 2007. [Citado el: 20 de 2 de 2010.] http://www.joangarnet.com/blog/?p=415.

4. **Martín, Sergio, y otros.** E Boletín N°8 Rama de Estudiantes de IEEE-UNED. [En línea] 2007. [Citado el: 10 de 2 de 2010.] http://www.ieec.uned.es/ieee/investigacion/ieee_dieec/sb/boletin_8_Octubre_2007.pdf.

5. **Sparx Systems.** Sparx Systems - Tutorial UML 2 - Diagrama de Secuencia. *Sparx Systems - Tutorial UML 2 - Diagrama de Secuencia.* [En línea] 2007. [Citado el: 13 de abril de 2010.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_sequencediagram.html.

6. **Enterprise Architect.** Guía de Usuario de Enterprise Architect 7.0. *Guía de Usuario de Enterprise Architect 7.0.* [En línea] Enterprise Architect. [Citado el: 6 de abril de 2010.] http://www.sparxsystems.com.ar/download/ayuda/index.html?componentdiagram.htm.

7. **Rojas, Johanna y Barrios, Emilio.** Métodos de prueba de caja negra. *Métodos de prueba de caja negra.* [En línea] 2007. [Citado el: 24 de Mayo de 2010.] http://www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node28.html.

Glosario de términos

[A]

Abstracción: Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar cómo se implementan estas características. Los procesos, las funciones o los métodos pueden también ser abstraídos y cuando lo están, una variedad de técnicas son requeridas para ampliar una abstracción.

Algoritmo: Es un conjunto finito de instrucciones o pasos que sirven para ejecutar una tarea o resolver un problema.

Atributo: Contenedor de un tipo de datos asociados a un objeto, que hace los datos visibles desde fuera del objeto, y cuyo valor puede ser alterado por la ejecución de algún método.

[C]

Clase: Definiciones de las propiedades y comportamiento de un conjunto de objetos concretos. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ellas.

Componente: Un componente es una parte no trivial, casi independiente, y reemplazable de un subsistema que llena claramente una funcionalidad dentro de un contexto en una arquitectura bien definida. Un componente se conforma y provee la realización física por medio de un conjunto de interfaces.

[E]

Evento: Un suceso en el sistema. El sistema maneja el evento enviando el mensaje adecuado al objeto pertinente. También se puede definir como evento, a la reacción que puede desencadenar un objeto, es decir la acción que genera.

[F]

Framework: Denota la infraestructura sobre la cual se reúnen un conjunto de lenguajes, herramientas y servicios que simplifican el desarrollo de aplicaciones en entorno de ejecución distribuido.

[H]

Herencia: Las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que re-implementar su comportamiento. Esto suele hacerse habitualmente agrupando los objetos en clases y estas en árboles o enrejados que reflejan un comportamiento común. Cuando un objeto hereda de más de una clase se dice que hay herencia múltiple; esta característica no está soportada por algunos lenguajes (como Java).

[M]

Mensaje: Una comunicación dirigida a un objeto, que le ordena que ejecute uno de sus métodos con ciertos parámetros asociados al evento que lo generó.

Método: Algoritmo asociado a un objeto, cuya ejecución se desencadena tras la recepción de un "mensaje". Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, o la generación de un "evento" con un nuevo mensaje para otro objeto del sistema.

[N]

Nomenclador: Catálogo de reglas estipuladas a nivel central o de entidad.

[S]

Sistema de software: Es un programa o aplicación de software que permite a los usuarios el control o realización de varias tareas y que hacen que el trabajo sea más cómodo, rápido y eficiente.

Subsistema: Son las partes que forman un sistema. Cada sistema está compuesto de subsistemas, los cuales a su vez son parte de otros subsistemas; cada subsistema es delineado por sus límites.

[O]

Objeto: Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad. Corresponden a los objetos reales del mundo que nos rodea, o a objetos internos del sistema.

[P]

Planilla evaluativa: Documento utilizado en los procesos de evaluaciones que recoge aspectos a tener en cuenta durante el mismo.

Proceso: Conjunto de actividades que guían los esfuerzos de las personas implicadas para el cumplimiento de un objetivo.

Polimorfismo: Comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre, al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando. O dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado. Cuando esto ocurre en "tiempo de ejecución", esta última característica se llama asignación tardía o asignación dinámica. Algunos lenguajes proporcionan medios más estáticos (en "tiempo de compilación") de polimorfismo, tales como las plantillas y la sobrecarga de operadores de C++.

[U]

UML: Lenguaje Unificado de Modelado, es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software.