

Universidad de las Ciencias Informáticas

Facultad 1



**Diseño e implementación del componente Configuración del
Proceso Evaluativo**

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

Autor:

Cadete Israel Mora Sánchez

Tutor:

Tte. Ing. Yadir Martínez Vergara

Ciudad de La Habana, 2010.

Año 52 de la Revolución

Declaración de Autoría


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 Israel Mora Sánchez
Autor

Tte. Ing. Yadir Martinez Vergara
Tutor



“La organización del futuro tendrá una tecnología tan avanzada que solo será necesario una persona y un perro para administrarla. La persona se encargará de alimentar al perro y éste se encargará de que la persona no toque nada...”

Anónimo

DATOS DE CONTACTO

Tutor: Tte. Ing. Yadir Martínez Vergara

UCID, Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: yvergara@uci.cu

AGRADECIMIENTOS

Con la realización de este trabajo quisiera agradecer:

A la Revolución, a Fidel y a la UCI, por darme la oportunidad de formar parte de este proyecto futuro y formarme como profesional y oficial de las fuerzas armadas.

A mis padres y a mi hermanita por haber estado siempre ahí cuando los necesité, por su amor, su cariño y su confianza.

A mi tía Marisol y al Polo por su apoyo incondicional y a todos mis familiares en general

A mis hermanos del barrio, de la primaria, de la secundaria: Willian y el Pirata, gracias por estar aquí en este momento tan especial para mí.

A mi tutor el Tte. Ing. Yadir Martínez Vergara por su apoyo brindado.

A mis hermanos y hermanas de la universidad, a todos los del viejo grupo 1301, a Chavelys, Yanelys, la perica (Milay), la Rubia (Dayanis), Sergio, Gabriel, Chanel, Leordan, David, Dariel, Osnel, Nelly, Rafa, Alejandro, Lisbey, Eliska, Yuniel, Yula, Yaité, Ana Yensi, Daily, Maidel, Pedroso, Yeriuska, que me disculpen si olvido alguno...

A Katia y Annelisse, que a pesar del corto tiempo de conocernos me han ayudado y soportado durante todo este año.

Y a todas mis amistades que de una forma u otra me ayudaron para que este sueño se hiciera realidad.

DEDICATORIA

Este trabajo va dedicado con mucho amor y cariño a mis padres por haberme guiado sabiamente, sin su amor, dedicación y apoyo no hubiese sido posible. Todo lo que he logrado en la vida, se los debo a ustedes, los quiero...

A mi abuelo Indio, mi abuela Mirella y mi abuela Salvadora, que ya no se encuentran físicamente, los quiero mucho. A mi hermanita, que aunque casi siempre esté fajado con ella, la quiero con el alma. A mis familiares, a mis amistades, a toda la gente que me aprecia y a todos los que han estado ahí cuando los necesité.

RESUMEN

El tema de las competencias y su relación con el trabajo se inicia al término de la Segunda Guerra Mundial en que se buscó con insistencia mejorar el rendimiento y aumentar la producción. Posteriormente, en la década del 70, David McLelland, psicólogo de la universidad de Harvard, resaltó la importancia de “verificar competencias en lugar de la inteligencia”, desarrollando así una nueva forma de comprender el desempeño de las personas en las diferentes actividades laborales: la competencia define el desempeño en relación con el trabajo que se realiza y la forma en que éste se hace.

En los últimos años, las investigaciones centradas en el aspecto humano dentro de las organizaciones han demostrado que, aquellas que prestan mayor atención a las personas logran mejores resultados. Esto ha traído como consecuencia que el concepto de competencia se haya hecho presente con increíble fuerza en la mayoría de las organizaciones.

En el presente trabajo se realiza un estudio de los distintos procesos que se llevan a cabo sobre la gestión de evaluaciones en el país, particularizando en el proceso de configuración de la evaluación. Para ello se propone el diseño y la implementación de un componente que permita la configuración de los procesos evaluativos.

Palabras claves: competencias, capital humano, evaluación del desempeño, configuración de procesos evaluativos.

TABLA DE CONTENIDOS

INTRODUCCIÓN	1
CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA. ESTADO DEL ARTE.....	4
1.1. INTRODUCCIÓN	4
1.2. ESTUDIO DE LA TESIS PRECEDENTE.....	4
1.2.1. ESTUDIO DEL CAPÍTULO 1	4
1.2.1.1. PROCESO DE EVALUACIÓN DEL MINISTERIO DE TRABAJO Y SEGURIDAD SOCIAL.....	6
1.2.1.2. PROCESO DE EVALUACIÓN DE LOS CUADROS DE LAS FAR.....	6
1.2.1.3. TECNOLOGÍAS Y HERRAMIENTAS PROPUESTAS A UTILIZAR	7
1.2.1.4. INCONFORMIDADES ENCONTRADAS	8
1.2.2. ESTUDIO DEL CAPÍTULO 2	9
1.2.2.1. PROCESOS DE NEGOCIOS QUE INTERVIENEN.....	9
1.2.2.2. VISTA GENERAL DE CAMBIOS EN EL PROCESO DE NEGOCIO “DEFINIR PROCESO EVALUATIVO”	10
1.2.2.3. VISTA GENERAL DE CAMBIOS EN EL PROCESO DE NEGOCIO “PREPARAR PROCESO EVALUATIVO”	12
1.3. PROPUESTA DE SOLUCIÓN	14
1.4. PROCESO DE DESARROLLO Y GESTIÓN DE PROYECTOS DE SOFTWARE	14
1.4.1. FASES DEL CICLO DE VIDA	15
1.4.2. MODELO DE DESARROLLO DE SOFTWARE	17
1.5. HERRAMIENTAS, LENGUAJES Y TECNOLOGÍAS A UTILIZAR.....	18
1.5.1. MARCO DE TRABAJO SAUXE 1.5.4.....	18
1.5.2. VENTAJAS DEL NUEVO MARCO DE TRABAJO:	18
1.5.3. FRAMEWORK DEL LADO DEL CLIENTE	19
1.5.4. FRAMEWORK PARA DESARROLLO DE APLICACIONES WEB.....	20
1.5.5. SISTEMA DE MAPEO OBJETO–RELACIONAL	21
1.5.6. ZEND EXT FRAMEWORK	22
1.5.7. ESTILO ARQUITECTÓNICO MODELO-VISTA-CONTROLADOR.....	22
1.6. CONCLUSIONES	24
CAPÍTULO 2: DISEÑO DEL SISTEMA.	25
2.1. INTRODUCCIÓN	25
2.2. ESTÁNDARES DE CÓDIGO.....	25
2.3. PATRONES DE DISEÑO	26

2.3.1.	PATRONES GRASP	27
2.3.1.1.	PATRÓN CONTROLADOR	27
2.3.1.2.	PATRÓN EXPERTO.....	28
2.3.1.3.	PATRÓN CREADOR.....	28
2.3.1.4.	PATRÓN DE ALTA COHESIÓN.....	29
2.3.1.5.	BAJO ACOPLAMIENTO	29
2.3.2.	PATRONES GoF.....	30
2.4.	DIAGRAMA DE CLASES WEB.....	30
2.4.1.	DIAGRAMA DE CLASES GENÉRICO DE LAS VISTAS.....	31
2.4.2.	DESCRIPCIÓN DE LAS CLASES GENÉRICAS DE LA VISTA	31
2.4.3.	DIAGRAMA DE CLASES GENÉRICO	32
2.4.4.	DESCRIPCIÓN DE LAS CLASES	32
2.4.5.	DIAGRAMA DE CLASES DEL DISEÑO GESTIONAR PROCESO EVALUATIVO.....	34
2.4.6.	DESCRIPCIÓN DE LAS CLASES	34
2.4.7.	DIAGRAMA DE SECUENCIA LISTAR DEFINIR PROCESO EVALUATIVO	37
2.4.8.	DIAGRAMA DE SECUENCIA ADICIONAR DEFINIR PROCESO EVALUATIVO.....	37
2.4.9.	DIAGRAMA DE SECUENCIA MODIFICAR DEFINIR PROCESO EVALUATIVO	38
2.4.10.	DIAGRAMA DE SECUENCIA ELIMINAR DEFINIR PROCESO EVALUATIVO.....	38
2.5.	DIAGRAMA DE SECUENCIA ASOCIAR RECOMENDACIÓN	39
2.6.	DIAGRAMA DE SECUENCIA ASOCIAR SEÑALAMIENTO	39
2.7.	DISEÑO DE LA BASE DE DATOS	40
2.8.	DIAGRAMA ENTIDAD-RELACIÓN DE LA BASE DE DATOS	40
2.9.	DESCRIPCIÓN DE LAS TABLAS DE LA BASE DE DATOS	40
2.10.	TRATAMIENTO DE ERRORES	41
2.10.1.	VALIDACIÓN EN EL SERVIDOR	41
2.10.2.	EXCEPCIONES.....	43
2.11.	SEGURIDAD.....	44
2.11.1.	SEGURIDAD DEL SISTEMA.....	44
2.11.2.	INYECCIÓN SQL.....	45
2.11.3.	XSS (CROSS-SITE SCRIPTING)	46
2.12.	PROPUESTA DEL SISTEMA	46
2.13.	CONCEPCIÓN DE LA AYUDA	48
2.14.	CONCLUSIONES PARCIALES.....	48

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA.	49
3.1. INTRODUCCIÓN	49
3.2. DIAGRAMA DE COMPONENTES.....	49
3.3. MATRIZ DE INTEGRACIÓN DE COMPONENTES INTERNOS.....	50
3.4. MODELOS DE PRUEBA	52
3.4.1. PRUEBAS DE CAJA NEGRA.....	52
3.4.2. DISEÑO DE CASOS DE PRUEBA	53
3.5. CONCLUSIONES PARCIALES	58
CONCLUSIONES GENERALES.....	59
RECOMENDACIONES.....	60
BIBLIOGRAFÍA	61
REFERENCIAS BIBLIOGRÁFICAS	62
GLOSARIO DE TÉRMINOS.....	63

ÍNDICE DE FIGURAS

FIGURA 1 ESQUEMA DE REPRESENTACIÓN DE LA MIGRACIÓN DE LA METODOLOGÍA DE DESARROLLO DE <i>SOFTWARE</i>	8
FIGURA 2 MODELO CONCEPTUAL DE DEFINIR PROCESO EVALUATIVO.	12
FIGURA 3 MODELO CONCEPTUAL DE PREPARAR PROCESO EVALUATIVO.....	14
FIGURA 4 ESQUEMA DE LAS FASES DEL PROCESO DE DESARROLLO Y GESTIÓN DE PROYECTOS DE <i>SOFTWARE</i>	16
FIGURA 5 DIAGRAMA GENÉRICO DE LAS CLASES DE LA VISTA.....	31
FIGURA 6 DIAGRAMA DE CLASES GENÉRICO.	32
FIGURA 7 DIAGRAMA DE CLASE DEL DISEÑO GESTIONAR DEFINIR PROCESO.	34
FIGURA 8 DIAGRAMA DE SECUENCIA LISTAR DEFINIR PROCESO EVALUATIVO	37
FIGURA 9 DIAGRAMA DE SECUENCIA ADICIONAR DEFINIR PROCESO EVALUATIVO	37
FIGURA 10 DIAGRAMA DE SECUENCIA MODIFICAR DEFINIR PROCESO EVALUATIVO	38
FIGURA 11 DIAGRAMA DE SECUENCIA ELIMINAR DEFINIR PROCESO EVALUATIVO.....	38
FIGURA 12 DIAGRAMA DE SECUENCIA ASOCIAR RECOMENDACIÓN	39
FIGURA 13 DIAGRAMA DE SECUENCIA ASOCIAR SEÑALAMIENTO.....	39
FIGURA 14 DIAGRAMA ENTIDAD-RELACIÓN DE LA BASE DE DATOS.....	40
FIGURA 15 EJEMPLO DE ACCIONES QUE PUEDE TENER UNA FUNCIONALIDAD.....	45
FIGURA 16 EJEMPLO DEL USO DE LA AYUDA.	48
FIGURA 17 DIAGRAMA DE COMPONENTES.	50

ÍNDICE DE TABLAS

TABLA 1 DESCRIPCIÓN DE LA CLASE EXT-BASE.JS	31
TABLA 2 DESCRIPCIÓN DE LA CLASE EXT-ALL.JS.....	31
TABLA 3 DESCRIPCIÓN DE LA CLASE GENÉRICA GESTX.PHTML	31
TABLA 4 DESCRIPCIÓN DE LA CLASE GENÉRICA GESTX.JS.....	31
TABLA 5 DESCRIPCIÓN DE LA CLASE ZEND_CONTROLLER_ACTION.PHP	32
TABLA 6 DESCRIPCIÓN DE LA CLASE ZEND_CONTROLLER_SECURE.PHP	33
TABLA 7 DESCRIPCIÓN DE LA CLASE GENÉRICA GESTXCONTROLLER.PHP	33
TABLA 8 DESCRIPCIÓN DE LA CLASE ZEND_VIEW.PHP	33
TABLA 9 DESCRIPCIÓN DE LA CLASE ZEND_CONTROLLER_FRONT.PHP	33
TABLA 10 DESCRIPCIÓN DE LA CLASE INDEX.PHP	33
TABLA 11 DESCRIPCIÓN DE LA CLASE ZENDEXT_MODEL.PHP	33
TABLA 12 DESCRIPCIÓN DE LA CLASE GENÉRICA GESTXMODEL.PHP.....	33
TABLA 13 DESCRIPCIÓN DE LA CLASE BASECLASE.PHP	33
TABLA 14 DESCRIPCIÓN DE LA CLASE CLASE.PHP	33
TABLA 15 DESCRIPCIÓN DE LA CLASE DEFINIRPROCESO.JS.....	34
TABLA 16 DESCRIPCIÓN DE LA CLASE DEFINIRPROCESOCONTROLLER.PHP.....	35
TABLA 17 DESCRIPCIÓN DE LA CLASE NOMEVALDEFINIRPROCESOMODEL.PHP	36
TABLA 18 DESCRIPCIÓN DE LA CLASE DATEVALDEFPROCESOEVALUATIVO.PHP.....	36
TABLA 19 DESCRIPCIÓN DE LA TABLA DATEVALDEFPROCESOEVALUATIVO.....	41
TABLA 20 MATRIZ DE INTEGRACIÓN INTERNA.....	51
TABLA 21 MATRIZ DE INTEGRACIÓN EXTERNA.....	52
TABLA 22 DISEÑOS DE CASO DE PRUEBA GESTIONAR SEÑALAMIENTO.....	54
TABLA 23 DISEÑO DE CASO DE PRUEBA GESTIONAR TIPO DE SEÑALAMIENTO	54
TABLA 24 DISEÑOS DE CASO DE PRUEBA GESTIONAR RECOMENDACIÓN.....	54
TABLA 25 DISEÑO DE CASO DE PRUEBA GESTIONAR TIPO DE RECOMENDACIÓN	54
TABLA 26 DISEÑO DE CASO DE PRUEBA GESTIONAR TIPO DE PROCESO	55
TABLA 27 DISEÑO DE CASO DE PRUEBA GESTIONAR PERÍODO	55
TABLA 28 DISEÑOS DE CASO DE PRUEBA GESTIONAR COMISIÓN DE EVALUACIÓN	55
TABLA 29 DISEÑO DE CASO DE PRUEBA GESTIONAR EVALUADORES.....	55
TABLA 30 DISEÑO DE CASO DE PRUEBA GESTIONAR EVALUADOS	56
TABLA 31 DISEÑO DE CASO DE PRUEBA GESTIONAR PREPARAR PROCESO EVALUATIVO.....	56
TABLA 32 DISEÑO DE CASO DE PRUEBA GESTIONAR DEFINIR PROCESO EVALUATIVO	56

TABLA 33 JUEGO DE DATOS DEL ESCENARIO ADICIONAR PROCESO EVALUATIVO.	57
TABLA 34 JUEGO DE DATOS DEL ESCENARIO MODIFICAR PROCESO EVALUATIVO.	57
TABLA 35 JUEGO DE DATOS DEL ESCENARIO ELIMINAR PROCESO EVALUATIVO.....	57
TABLA 36 JUEGO DE DATOS DEL ESCENARIO ASOCIAR RECOMENDACIÓN.....	58
TABLA 37 JUEGO DE DATOS DEL ESCENARIO ASOCIAR SEÑALAMIENTO.	58

Introducción

Hoy en día, debido al alto desarrollo alcanzado en la rama de la computación, la gran mayoría de las empresas se encuentran en algún grado informatizadas, convirtiéndose los sistemas computacionales en un recurso imprescindible para todas las ramas de la sociedad. Estos proporcionan la infraestructura suficiente y necesaria para la gestión de la información, elimina el trabajo engorroso de realizarlo a mano, evita que se comentan errores por el agotamiento del cerebro humano y permite obtener una información actualizada de todos los procesos que se encuentren digitalizados, obteniéndose un estricto control y seguimiento de los mismos.

Uno de estos sistemas computacionales indispensables para una entidad y que proporcionaría un ahorro de tiempo y esfuerzos extraordinarios es la evaluación del desempeño la cual consiste en la medición sistemática e integral del grado de eficiencia de la labor realizada por los trabajadores en la actividad que ejecutan, durante un período de tiempo, y de su potencial de desarrollo en el ámbito de la empresa, identifica los tipos de insuficiencias y problemas del personal evaluado, sus fortalezas, posibilidades y capacidades, los caracteriza y constituye la base para la elaboración del plan de desarrollo, acorde a las necesidades. Repercute sobre la permanencia, promoción, formación, reconocimiento moral e idoneidad.

Actualmente en nuestro país, los directivos evalúan el desempeño de sus subordinados de manera ineficiente y engorrosa. Al mismo tiempo, los pasos a seguir por los evaluadores se hacen cada vez más complejos: siempre que se realiza un análisis de una persona en un período de tiempo determinado, se tiene que realizar una búsqueda en todo su expediente, mediante complejos métodos de evaluación y un gran esfuerzo humano se lleva a cabo el proceso de desempeño alcanzado por dicha individuo.

Teniendo en cuenta lo expuesto anteriormente el **problema a resolver** queda definido de la siguiente manera:

¿Cómo facilitar el proceso configuración y gestión del proceso evaluativo de una entidad?

El **objeto de estudio** del presente trabajo es el proceso de gestión del capital humano. Derivándose como **campo de acción** el proceso de evaluación de desempeño.

Se plantea como **Idea a defender** lo siguiente: Si se realiza la implementación del componente Configuración del Proceso Evaluativo se facilitará la gestión y configuración de los procesos evaluativos que se llevan a cabo en una entidad.

Basado en la idea a defender anteriormente expuesta, se define como **objetivo general** de la presente investigación: implementar un componente dinámico para el módulo Evaluación de Desempeño que facilite la gestión y configuración del proceso evaluativo de una entidad.

Del cual se derivan los siguientes **objetivos específicos**:

1. Obtener el diseño teórico definitivo de la investigación.
2. Obtener un estudio del estado del arte de los principales procesos evaluativos que se llevan a cabo en el país haciendo énfasis en el proceso de configuración de la evaluación.
3. Obtener el diseño del componente Configuración del Proceso Evaluativo.
4. Obtener la implementación del componente Configuración del Proceso Evaluativo.
5. Obtener la validación del componente Configuración del Proceso Evaluativos.

El documento está estructurado de la siguiente forma:

- **Capítulo 1: Fundamentación teórica. Estado del arte:** Se desarrolla un estudio de los principales sistemas de gestión de Capital Humano existentes, tanto en Cuba como en el resto del mundo. Se desarrolla una valoración crítica del diseño propuesto por el analista, un análisis de posibles implementaciones ya existentes y que puedan ser reutilizadas. Además, se describen los lenguajes, las herramientas y metodologías a utilizar para el desarrollo del componente.
- **Capítulo 2: Diseño del sistema:** Se desarrollan los diagramas de clases del diseño así como sus respectivos diagramas de secuencia por escenario. Se muestra el diagrama entidad-relación de la base de datos. Se brinda una breve panorámica del tratamiento de errores del

sistema además de cómo se ve reflejada la seguridad en el mismo. Se explica cómo se ven reflejados los patrones de diseño en el sistema y cómo se concibe la ayuda.

- **Capítulo 3: Implementación y validación de la solución propuesta:** Se desarrolla una búsqueda de las pruebas de unidades que permitan validar la solución propuesta. Se realiza una descripción de la misma teniendo en cuenta: tipo de prueba, objetivo de la prueba y alcance. Se describen los valores utilizados y se evalúa la ejecución de la prueba y los resultados obtenidos.

Capítulo 1 Fundamentación Teórica

Capítulo 1: Fundamentación teórica. Estado del arte.

1.1. Introducción

En el presente capítulo se realiza un análisis detallado de la tesis precedente, en el cual se verifica la necesidad de efectuar los cambios pertinentes con el objetivo de garantizar el confort del cliente. Se brinda una propuesta de solución la cual refleja las principales características o funcionalidades que presentará el componente. Se ofrece una breve descripción del “Proceso de Desarrollo y Gestión de Proyectos de *Software*” y cómo este guía el progreso de la aplicación, además se muestran las características fundamentales del modelo de desarrollo de *software* a utilizar. Se lleva a cabo un estudio crítico del estado del arte de las principales herramientas, lenguajes y tecnologías a emplear, así como una valoración de los que son más factibles para darle solución al problema en cuestión.

1.2. Estudio de la tesis precedente

Con el objetivo de comprobar las bases de la investigación y así determinar si se realizan o no cambios que influyan en el diseño o implementación del componente, se decide realizar un análisis de la tesis precedente. El mismo deberá comprender los temas tratados en los capítulos 1 y 2 para luego al finalizar cada uno de ellos efectuar una valoración crítica justificando los cambios pertinentes.

1.2.1. Estudio del capítulo 1

El primer capítulo de la tesis a la que se le está dando continuidad, comienza con la exposición del concepto de evaluación, fundamental para garantizar un mejor dominio del problema en cuestión donde, según lo establecido en la Resolución No.21 de la Gaceta Oficial de Cuba:

“La evaluación es una actividad cotidiana y sistemática, que deben realizar los jefes con sus subordinados, como parte esencial del sistema de trabajo con los cuadros y constituye un extraordinario instrumento de dirección y educación, que les permite valorar el comportamiento y resultados obtenidos por el oficial en el cumplimiento de sus múltiples misiones y tareas, así como el desarrollo multilateral alcanzado por el mismo. Dichos resultados permiten valorar sus aciertos, deficiencias y causas de las mismas, determinando, a su vez, cómo han sido aplicados los principios de la política de cuadros y trazar las medidas para el trabajo futuro.” [1]

Capítulo 1 Fundamentación Teórica

La evaluación es un proceso mediante el cual, por una parte, el individuo conoce cómo la institución valora los resultados de su trabajo, sus cualidades profesionales y personales, su correspondencia con las responsabilidades que desempeña, sus potencialidades de desarrollo, y que se quiere con él en un futuro inmediato. Por otro lado, la institución conoce cómo piensa el evaluado sobre sí mismo y su entorno, si siente que se le está reconociendo su trabajo, si están siendo atendidas sus necesidades, su nivel de compromiso en erradicar las dificultades y señalamientos indicados, y al mismo tiempo tiene la oportunidad de transmitirle confianza, seguridad, y de exhortarlo a mejores desempeños.

Luego de estudiar los conceptos básicos, se presentan los principales procesos de evaluación existentes en el país:

- Proceso de Evaluación del PCC¹
- Proceso de Evaluación de la UJC²
- Proceso de Evaluación de los Cuadros de las FAR³
- Proceso de Evaluación de la CTC⁴
- Proceso de Evaluación del Ministerio de Trabajo y Seguridad Social
- Proceso de Evaluación del MININT⁵

Posteriormente se realiza un estudio más profundo de dos de estos procesos evaluativos:

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

¹ Partido Comunista de Cuba

² Unión de Jóvenes Comunistas

³ Fuerzas Armadas Revolucionarias

⁴ Central de Trabajadores de Cuba

⁵ Ministerio del Interior

Capítulo 1 Fundamentación Teórica

1.2.1.1. Proceso de Evaluación del Ministerio de Trabajo y Seguridad Social

“La evaluación del desempeño se realiza por 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.” [2]

Este procedimiento para llevar a cabo evaluaciones es uno de los más comunes, empleado en casi todas las entidades. Entre las principales características que presenta este proceso evaluativo se encuentra que:

La administración está obligada a poner en conocimiento de los trabajadores los indicadores fundamentales, así como los adicionales, antes del comienzo del período de evaluación.

El trabajador tiene derecho a discutir la evaluación con su jefe y manifestar sus opiniones. En caso de inconformidad con el resultado de la evaluación del desempeño, el trabajador puede restablecer reclamación escrita ante el jefe inmediato superior al que lo evaluó, quién de conjunto con el dirigente de la organización sindical a ese nivel, analizará la reclamación y tomará la decisión de:

- Mantener como firme el criterio del jefe que evaluó.
- Solicitar al jefe que evaluó que rectifique la evaluación realizada.
- Proceder directamente a rectificar la evaluación dando razón al trabajador.

El jefe inmediato superior tendrá un período máximo de quince días hábiles para tomar la decisión.

1.2.1.2. Proceso de Evaluación de los Cuadros de las FAR

Los cuadros preparan y establecen las evaluaciones sobre la base del esquema y del plan de evaluación, que a su vez son elaborados por ellos.

“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 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. No obstante estos modelos pueden ser modificados en el tiempo.” [2]

Capítulo 1 Fundamentación Teórica

De los estudiados, se escoge este proceso evaluativo con toda intención ya que es uno de los pocos que tiene sus propias particularidades, unas ya vistas en la cita antes mencionada y otras se pueden apreciar a continuación.

Los cuadros deben ser evaluados periódicamente y por diversas causas (propuesta de movimiento, tiempo de estancia en el mismo cargo, cumplimiento del período de instrucción).

Cada dirigente es el encargado de evaluar a los cuadros que le están directamente subordinados, para lo cual puede crear comisiones de análisis de la evaluación.

Después de realizadas las evaluaciones, se crea el plan de desarrollo y los diferentes tipos de listados de candidatos, que deben ser aprobados por el jefe de la nomenclatura del cargo en el caso del listado de candidatos a cargo (cargo, estímulo, ascenso, curso, entre otros).

1.2.1.3. Tecnologías y herramientas propuestas a utilizar

Tras la explicación de estos procesos, se realiza un estudio de las tecnologías y herramientas actuales a utilizar, estas fueron tomadas de la arquitectura del proyecto Cuadros del UCID⁶. Se selecciona el Modelo Orientado a Componentes del Proyecto ERP⁷-Cuba como metodología de desarrollo y *Visual Paradigm for UML*⁸ 6.1 Enterprise Edition como herramienta para el modelado usando BPMN⁹ y UML, PHP¹⁰ 5.2.4 para la implementación del lado del servidor y HTML¹¹ 4.0, *JavaScript* y CSS¹² 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*

⁶ Unidad de Compatibilización e Integración para la Defensa

⁷ *Enterprise Resource Planque Brasil ning*(Planificación de Recursos Empresariales)

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

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

¹⁰ *Hypertext Pre-processor*(Pre-procesador de Hipertexto)

¹¹ HTML: *Hypertext Markup Language*(Lenguaje de Marcas de Hipertexto)

¹² CSS: *Cascading Style Sheets*(Hojas de Estilo en Cascada)

Capítulo 1 Fundamentación Teórica

for Eclipse 6.0.0 como IDE de desarrollo, PostgreSQL 8.3 como SGBD¹³, pgAdmin III v1.8.2 como herramienta de administración para el SGBD y como generador de reportes dinámicos la Plataforma de Ayuda a la toma de decisiones en Sistemas Inteligentes. Todo en completa armonía bajo el estilo arquitectónico MVC¹⁴.

1.2.1.4. Inconformidades encontradas

Luego de un análisis en detalle de este capítulo, se encuentra como único cambio necesario para que el desarrollo del sistema se ajuste a las políticas establecidas por el centro UCID, la metodología de desarrollo a utilizar, es decir, se decide sustituir la metodología Modelo Orientado a Componentes del Proyecto ERP-Cuba por el Proceso de Desarrollo y Gestión de Proyectos de *Software* de centro UCID.

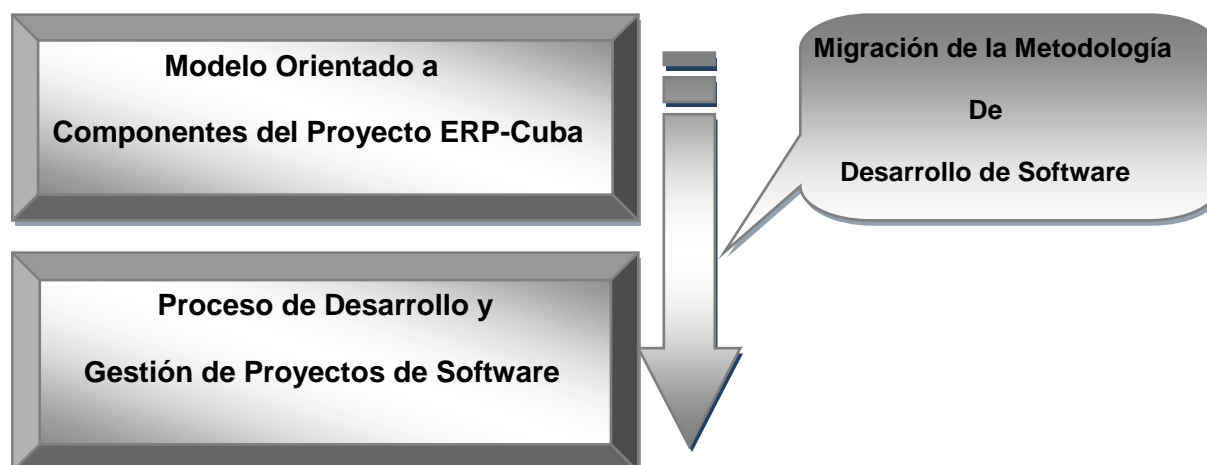


Figura 1 Esquema de representación de la migración de la Metodología de Desarrollo de *Software*.

¹³ SGBD: Sistema Gestor de Base de Datos

¹⁴ MVC: Modelo Vista Controlador

Capítulo 1 Fundamentación Teórica

1.2.2. Estudio del capítulo 2

El capítulo inicia con un estudio crítico de las aplicaciones usadas en la actualidad que pudieran satisfacer las necesidades o simplemente aportar ideas para el desarrollo del sistema. Las herramientas tratadas fueron:

- **Lavasoft:** No utiliza tecnologías web ni de software libre y su carácter estático es un obstáculo para crear una aplicación genérica capaz de humanizar el proceso de gestión de las evaluaciones.
- **EVAL 360:** Es un software propietario, por lo que no 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 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.
- Configuración básica para la gestión del proceso evaluativo.
- Recuperaciones dinámicas de estadísticas del proceso evaluativo.
- Integración entre subsistemas como el ERP-Cuba.

Como se puede apreciar, estas aplicaciones de una manera u otra, no cumplen con las expectativas previstas.

1.2.2.1. Procesos de negocios que intervienen

Luego se identifican y describen los procesos de negocios que intervienen en la evaluación de los cuadros de las FAR, los cuales se presentan a continuación:

- Preparar Proceso Evaluativo
- Realizar Evaluación

Capítulo 1 Fundamentación Teórica

- Resumir Proceso Evaluativo

Después de varias confrontaciones con los clientes se determina agregar un nuevo proceso de negocio “Definir Proceso Evaluativo”, que sería el que diera inicio, seguido por “Preparar Proceso Evaluativo”, luego “Realizar Evaluación”, y para concluir “Resumir Proceso Evaluativo”. La incorporación de este nuevo proceso de negocio trae como consecuencia la toma de una serie de decisiones que le dan un vuelco a los requisitos funcionales propuestos, los mismos se verán a continuación de manera general y más argumentada en el próximo capítulo.

Debido a que la investigación se encuentra enmarcada en los procesos de negocios “Definir Proceso Evaluativo” y “Preparar Proceso Evaluativo”, los próximos análisis estarán orientados entorno a estos, conformando así, el componente Configuración del Proceso Evaluativo.

1.2.2.2. Vista general de cambios en el proceso de negocio “Definir proceso evaluativo”

Por su necesidad, se decide mantener intactos los requisitos definidos durante la tesis precedente y simplemente agregar nuevos requisitos que garanticen el correcto funcionamiento y cumplimientos de los objetivos del proceso de negocio tratado. Entre los requisitos que se adicionan se encuentra gestionar “Proceso evaluativo”, el cual durante la tesis anterior pertenecía al proceso de negocio “Preparar Proceso Evaluativo” y ahora pertenece al proceso evaluativo tratado en esta sección. Los requisitos que se agregan son:

- Gestionar “Proceso Evaluativo”

En este requisito se gestiona todo el trabajo administrativo con los procesos evaluativos como la creación, modificación y eliminación de los mismos. Dicho proceso tiene como objetivo definir un proceso evaluativo. En la gestión de los procesos evaluativos se insertan un conjunto de datos necesarios.

- Gestionar “Tipo de Proceso Evaluativo”

Capítulo 1 Fundamentación Teórica

En este requisito se gestiona todo el trabajo administrativo con los tipos de procesos evaluativos como la creación, modificación y eliminación de los mismos. Dicho proceso tiene como objetivo definir un tipo de proceso evaluativo. En la gestión de los tipos de procesos evaluativos se insertan un conjunto de datos necesarios.

- Gestionar “Tipo de Recomendación”

En este requisito se gestiona todo el trabajo administrativo con los tipos de recomendaciones como la creación, modificación y eliminación de los mismos. Dicho proceso tiene como objetivo definir un tipo de recomendación. En la gestión de los tipos de recomendaciones se insertan un conjunto de datos necesarios.

- Gestionar “Tipo de Señalamiento”

En este requisito se gestiona todo el trabajo administrativo con los tipos de señalamientos como la creación, modificación y eliminación de los mismos. Dicho proceso tiene como objetivo definir un tipo de señalamiento. En la gestión de los tipos de señalamientos se insertan un conjunto de datos necesarios.

- Gestionar “Recomendación”

En este requisito se gestiona todo el trabajo administrativo con las recomendaciones como la creación, modificación y eliminación de los mismos. Dicho proceso tiene como objetivo definir una recomendación. En la gestión de las recomendaciones se insertan un conjunto de datos necesarios.

- Gestionar “Señalamiento”

En este requisito se gestiona todo el trabajo administrativo con los señalamientos como la creación, modificación y eliminación de los mismos. Dicho proceso tiene como objetivo definir un señalamiento. En la gestión de los señalamientos se insertan un conjunto de datos necesarios.

- Gestionar “Período evaluativo”

Capítulo 1 Fundamentación Teórica

En este requisito se gestiona todo el trabajo administrativo con los períodos evaluativos como la creación, modificación y eliminación de los mismos. Dicho proceso tiene como objetivo definir un período evaluativo. En la gestión de los períodos evaluativos se insertan un conjunto de datos necesarios.

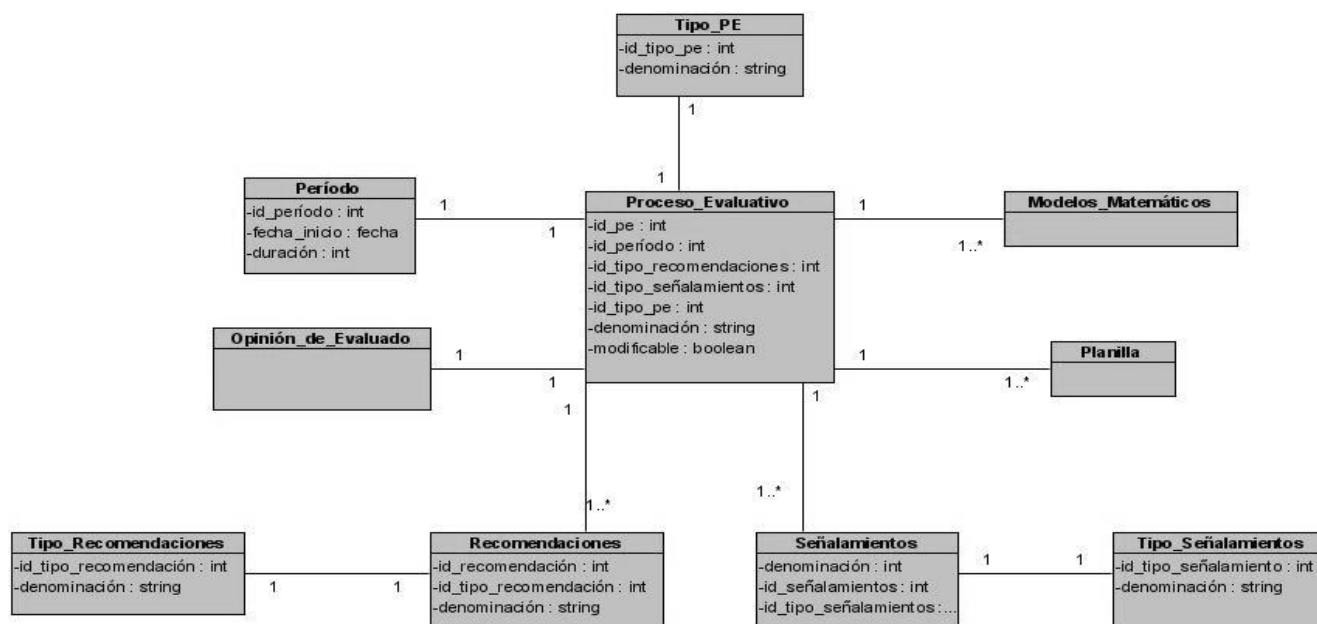


Figura 2 Modelo Conceptual de Definir Proceso Evaluativo.

1.2.2.3. Vista general de cambios en el proceso de negocio “Preparar proceso evaluativo”

Anteriormente, el proceso de negocio solo estaba compuesto por el requisito “Gestionar Proceso Evaluativo”, ahora éste pasa a formar parte del proceso de negocio “Definir Proceso Evaluativo” y el Proceso de negocio en cuestión, queda estructurado de la siguiente manera:

- Gestionar “Configurador del Proceso Evaluativo”

En este requisito se gestiona todo el trabajo administrativo con el configurador proceso evaluativo como la creación, modificación y eliminación de los mismos. Dicho proceso tiene como objetivo definir el proceso evaluativo así como las comisiones, evaluados y evaluadores que contendrá. En la gestión del configurador proceso evaluativo se insertan un conjunto de datos necesarios.

Capítulo 1 Fundamentación Teórica

- Gestionar “Comisión de Evaluación”

En este requisito se gestiona todo el trabajo administrativo con la comisión de evaluación como la creación, modificación y eliminación de las mismas. Dicho proceso tiene como objetivo definir la comisión de evaluación. En la gestión de las comisiones de evaluaciones se insertan un conjunto de datos necesarios.

- Gestionar “Evaluadores”

En este requisito se gestiona todo el trabajo administrativo con la gestión de evaluadores como la creación, modificación y eliminación de los mismos. Dicho proceso tiene como objetivo poder definir los evaluadores. En la gestión de los evaluadores se insertan un conjunto de datos necesarios.

- Gestionar “Evaluados asociados a los evaluadores”

En este requisito se gestiona todo el trabajo administrativo con la gestión de los evaluados asociados a los evaluadores como la creación, modificación y eliminación de los mismos. Dicho proceso tiene como objetivo poder definir los evaluados asociados a los evaluadores. En la gestión de los evaluados se insertan un conjunto de datos necesarios.

Capítulo 1 Fundamentación Teórica

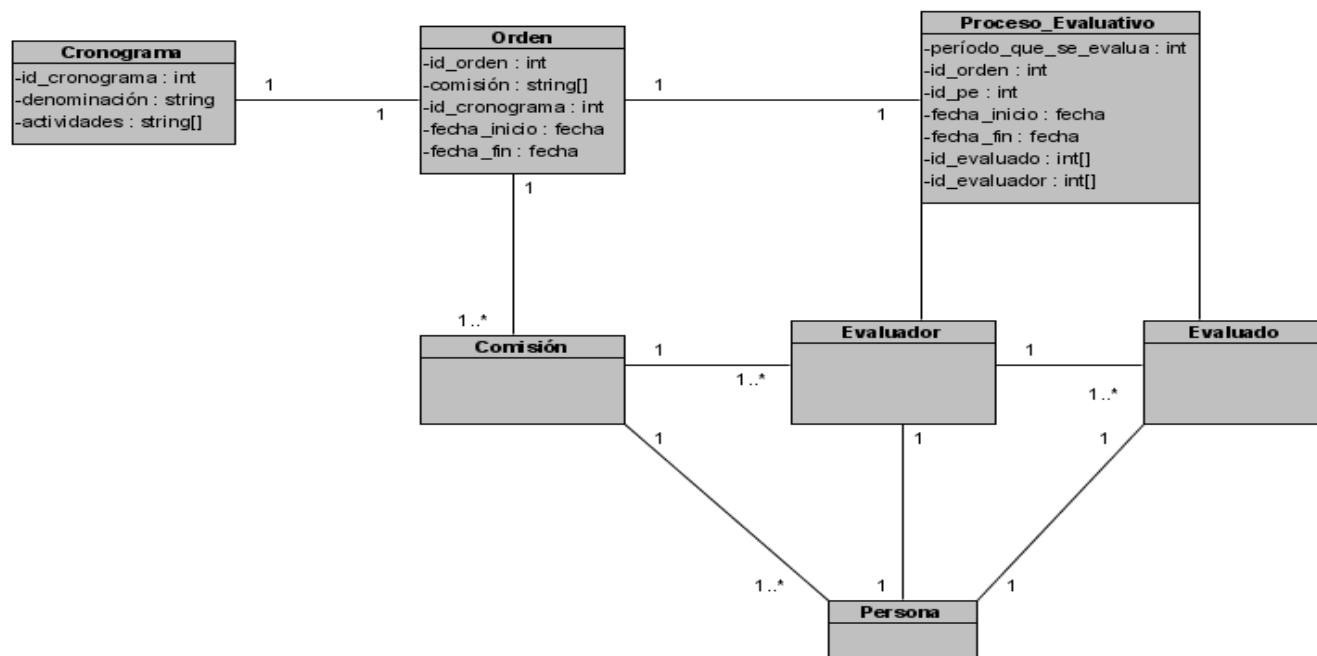


Figura 3 Modelo Conceptual de Preparar Proceso Evaluativo

1.3. Propuesta de solución

Después de realizar el estudio de los Sistemas de Gestión de Recursos Humanos utilizados en el mundo y en Cuba, se llega a la conclusión de que los mismos presentan una serie de características que de una manera u otra, no cumplen con todas las expectativas previstas, lo que ocasiona limitaciones que impiden controlar adecuadamente todos los procesos de la Evaluación de Desempeño y específicamente la configuración de procesos evaluativos. Por tal motivo, se decide elaborar una aplicación web de fácil uso, multiplataforma, que reúna todos los requisitos propuestos por el cliente y lleve un control de la configuración de un proceso evaluativo, es decir, el mismo debe ser capaz de definir y preparar procesos de gestión de las evaluaciones dinámicamente.

1.4. Proceso de Desarrollo y Gestión de Proyectos de Software

Diversas tendencias y metodologías de desarrollo de *software* han aparecido en años recientes, buscando resolver los problemas que proyectos más tradicionales, no han conseguido enfrentar. Entre ellas están los *framework* de proyectos, las metodologías ágiles y los modelos de medición de madurez. Junto con estos marcos de trabajo, ciertas estrategias específicas han permitido a los

Capítulo 1 Fundamentación Teórica

equipos de desarrollo producir un *software* más robusto, predecible, reutilizable y de fácil mantenimiento.

1.4.1. Fases del Ciclo de vida

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 riesgos 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.

Capítulo 1 Fundamentación Teórica

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 4 Esquema de las fases del Proceso de Desarrollo y Gestión de Proyectos de Software.

1.4.2. Modelo de desarrollo de software

El modelo de desarrollo de software propuesto describe la secuencia de actividades de alto nivel para la construcción y desarrollo de soluciones. Se logra con la combinación entre los modelos “Basado en componentes”, el “Iterativo e incremental” y “Orientado a procesos”. Se emplearán las técnicas de prototipo, si son requeridas, para los requerimientos del usuario de los que no existe una visión clara por parte de estos, con el objetivo de desarrollar una definición mejorada de los requisitos del usuario para el sistema.

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, hasta obtener el producto de software completo en la última iteración. En cada una de estas iteraciones se obtiene como resultado un incremento.

Desarrollo basado en componentes: Permite 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 proceso de negocio 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 a utilizar

Las 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 1.5.4

El marco de trabajo Sauxe 1.5.4 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-Ext, 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 estilo arquitectónico Modelo Vista Controlador. También tiene como propósito insertar la programación orientada a aspectos así como la inversión de controles. El *framework* en su nueva configuración define que la conexión a la base de datos será configurada en un archivo de tipo XML¹⁵ 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, este solamente deberá programar la obtención de los campos de la presentación y dentro del Sauxe el Doctrine es el responsable de que los mismos sean guardados en la base de datos.

1.5.2. Ventajas del nuevo marco de trabajo:

El antiguo método de *try* y *catch* para el lanzamiento de excepciones desaparece, en el nuevo marco con registrar las excepciones en el manager de excepciones, el *framework* será capaz de realizar un tratamiento óptimo de las mismas.

¹⁵ *Extensible Markup Language*(Lenguaje de marcado extendido)

Capítulo 1 Fundamentación Teórica

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.

Permite convertir estructuras de datos PHP a JSON¹⁶ y viceversa, para su utilización en aplicaciones AJAX¹⁷.

Facilidad de validaciones *pre* y *post* condición.

1.5.3. **Framework del lado del cliente**

ExtJS¹⁸ “Es un *framework* JavaScript del lado del cliente para el desarrollo de aplicaciones Web. Tiene un sistema dual de licencia: Comercial y Open Source.” [3] ExtJS posee una librería inmensa que permite configurar las interfaces Web usando tecnologías como AJAX, DHTML¹⁹ y DOM²⁰ de manera semejante a aplicaciones de escritorio, incluye la mayoría de los controles de los formularios Web basándose en contenedores para mostrar datos y elementos semejantes a la programación de escritorio 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 una acción del usuario, dándole la libertad de cargar información sin que el cliente lo note, 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

¹⁶ JavaScript Object Notation(Notación de objetos de JavaScript)

¹⁷ Asynchronous JavaScript And XML(Javascript asíncrono y XML)

¹⁸ Extend JavaScript library(Librería de JavaScript extendida)

¹⁹ HTML Dinámico(del inglés *Dynamic HTML*)

²⁰ Document Object Model

Capítulo 1 Fundamentación Teórica

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.

1.5.4. *Framework* para desarrollo de aplicaciones *web*

Zend Framework “Es utilizado para desarrollo de aplicaciones Web y servicios Web con PHP. Brinda soluciones para construir sitios web modernos, robustos y seguros. Además, es Open Source y trabaja con PHP 5.” [4] *Zend Framework* utiliza el patrón MVC como base de su funcionamiento. Es fácilmente integrable a las aplicaciones debido a su composición y a que contiene diferentes clases de gran utilidad, como por ejemplo en la búsqueda dinámica de ficheros a incluir o utilizar. Cuenta con un importante mecanismo de manejo de controladores y vistas.

Dentro de sus principales características están:

- Proporciona los componentes que forma la infraestructura del patrón MVC.
- Proporciona una capa de acceso a base de datos, construida sobre PDO²¹ pero ampliándola con diferentes características.
- Proporciona mecanismos de filtrado y validación de entradas de datos.
- Permite convertir estructuras de datos PHP a JSON y viceversa, para su utilización en aplicaciones AJAX.
- Proporciona capacidades de búsqueda sobre documentos y contenidos.

²¹ PHP Data Objects es una extensión que provee una capa de abstracción de acceso a datos para PHP 5, con lo cual se consigue hacer uso de las mismas funciones para hacer consultas y obtener datos de distintos manejadores de bases de datos.

Capítulo 1 Fundamentación Teórica

- Permite consumir y proveer 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.

1.5.5. Sistema de mapeo objeto–relacional

Doctrine es un potente y completo sistema ORM²² para PHP5.2+ con un DBAL *database abstraction layer* (Capa de abstracción de base de datos) incorporado, el mismo cuenta con disímiles funcionalidades, una de ellas es que da la posibilidad 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. Su principal ventaja radica en poder acceder a la base de datos utilizando la programación orientada a objetos, debido a que doctrine utiliza el patrón Active Record para manejar la base de datos, tiene su propio lenguaje de consultas y trabaja de manera rápida y eficiente.

“El patrón “Active Record es una extensión del patrón Domain Model (“Modelo de Dominio”), que se entiende como una clase o un grupo de clases que representan a “objetos” o responsabilidades particulares en la aplicación” [5], de forma general es un enfoque al problema de acceder a los datos de una base de datos. Una fila en la tabla de la base de datos (o vista) 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. Cuando se modifican los atributos del objeto, se actualiza la fila de la base de datos. Para una aplicación que despliega un catálogo de productos, por ejemplo, cada producto podría ser una instancia de la clase “Producto” (el catálogo mismo podría ser, a su vez, un objeto que contiene muchos objetos “Producto”).

²² *Object-Relational mapping*(mapeo objeto-relacional)

Capítulo 1 Fundamentación Teórica

1.5.6. Zend Ext Framework

Es un *framework* “*open source*” desarrollado por el grupo de arquitectura de la UCID, está diseñado para PHP 5 y con buenas capacidades de ampliación. Es elaborado a partir de Zend Framework cumpliendo con todas sus características. Este trae de novedoso un controlador vertical para el control de las acciones realizada por las vistas hacia el controlador, un motor de reglas para las validaciones en el servidor. Se le incluyó el IoC para la comunicación entre los módulos o componentes, la integración con el ORM Doctrine Framework para trabajo en la capa de abstracción a base de datos, el ExtJS Framework para el desarrollo de las vistas y un controlador de trazas para controlar las acciones del sistema (acción, excepciones, rendimiento, integración, y excepción de integración).

1.5.7. Estilo arquitectónico Modelo-Vista-Controlador

Uno de los elementos bases del proceso de desarrollo de software es diseñar la arquitectura de *software*. Esencialmente sobre ella se sustentan todos los mecanismos de diseño y representaciones de la estructura general de la aplicación a desarrollar. De la cohesión, utilidad y flexibilidad de los componentes de la arquitectura dependerán la calidad final y la utilidad del *software*. La correcta definición del estilo arquitectónico a utilizar, patrones y mecanismos de diseño es la raíz de lo anteriormente descrito.

Según lo planteado en el libro *The Unified Modeling Language user guide* (en español guía de usuario del Lenguaje de Modelado Unificado) se define como arquitectura:

“...el conjunto de decisiones significativas sobre la organización del sistema de software, la selección de los elementos estructurales y sus interfaces, con los que se compone el sistema, junto con su comportamiento tal como se especifica en las colaboraciones entre esos elementos, la composición de esos elementos estructurales y de comportamiento en subsistemas progresivamente más amplios, y el estilo de arquitectura que guía esta organización -estos elementos y sus interfaces, sus colaboraciones, y su composición”.[6]

Modelo Vista Controlador es un estilo arquitectónico basado en un patrón de diseño que plantea la separación de diferentes clases en dependencia de la función que realizan de modo tal que sea

Capítulo 1 Fundamentación Teórica

posible manejar dinámicamente la forma en que se procesan solicitudes y se gestiona la manera en que se muestran resultados al usuario final. En otras palabras separa la presentación del dominio de la aplicación. Debido a su gran aplicación en el mundo de las aplicaciones web se dice que *“MVC es particularmente apropiada para aplicaciones webs interactivas, aplicaciones donde un usuario web interactúa con un sitio web.”*[7]

Modelo: El modelo representa el almacenamiento de los datos y las reglas de negocio que rigen el acceso y actualización de estos datos. Esta es la representación específica de la información con la cual el sistema opera. Administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado y responde a instrucciones de cambiar el estado. El acceso a datos está implementado por clases php denominadas [Dat/Eval][requisito_x] ó [Dat/Nom][requisito_x] encargadas de la lectura de datos y las [requisito_x]Model se especializan en las funcionalidades de insertar, modificar y eliminar.

Vista: Representa el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario. Las vistas son las encargadas de enviar las peticiones de los usuarios a las clases controladoras y representar los resultados obtenidos, dichas vistas están implementadas por ficheros JavaScript: [requisito_x].js (encargada de la lógica de interfaz), ext_base.js, ext_all.js y ucid_all.js estos tres últimos serán descritos más adelante en el diagrama de clases genérico de las vistas. Todos estos ficheros son llamados desde un fichero [requisito_x].phtml que es el encargado de representar la interfaz de usuario.

Controlador: El controlador traduce las interacciones con el punto de vista en las acciones a realizar por el modelo. Controla el flujo entre la vista y el modelo (los datos). Responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y en la vista. Las clases php controladoras tendrán el nombre de [requisito_x]Controller y son las encargadas de recibir las peticiones de los usuarios desde la vista y hacer llamadas a las funciones necesarias para dar cumplimientos a las mismas, son las clases responsables de mediar entre la capa modelo y la vista.

Capítulo 1 Fundamentación Teórica

1.6. Conclusiones

Durante el transcurso del presente capítulo se realizó un estudio de los capítulos 1 y 2 de la tesis precedente donde se analizaron cuestiones tales como estado del arte, fundamentación teórica de los lenguajes, tecnologías y herramientas a emplear, arquitectura de diseño y proceso de desarrollo concluyendo con la necesidad de agregar y modificar algunos aspectos, vitales para el resultado final de la aplicación, que garantice el confort del cliente. También se aprobó la propuesta de herramientas y tecnologías a utilizar mencionadas en la tesis precedente siempre teniendo en cuenta no salir del ámbito de las propuestas de herramientas del marco de trabajo Sauxe 1.5.4 garantizando así cumplir con las políticas de desarrollo de centro UCID.

Capítulo 2: Diseño del sistema.

2.1. Introducción

En este capítulo se tratan varios aspectos ya más inmersos en el desarrollo del componente. Se comienza realizando un profundo análisis de las definiciones de diseños aplicadas así como la influencia de estos sobre el componente. Se pretende mostrar cómo está aprovechada la arquitectura y las posibilidades que proporcionan los *framework* y las librerías utilizadas en la programación de la aplicación, la representación de los diagramas de clases del diseño y sus respectivos diagramas de secuencias, con el objetivo de facilitar la comprensión del funcionamiento de los componentes implementados. Se expone el diagrama entidad-relación en el cual se puede observar cómo están relacionadas las tablas de la base de datos y se describe a detalle el tratamiento de errores en el sistema y cómo se garantiza la seguridad en el componente, se brinda una propuesta de solución que describe el diseño de las interfaces, se muestra cómo está concebida la ayuda en el componente, se realiza una descripción de las clases implementadas.

2.2. Estándares de código

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. *“Un estándar de programación no solo busca definir la nomenclatura de las variables, objetos, métodos y funciones, sino que también tiene que ver con el orden y legibilidad del código escrito.*

Siguiendo esta idea, se definen tres partes principales dentro de un estándar de programación:

- **Convención de nomenclatura:** *Cómo nombrar variables, funciones, métodos, entre otros.*
- **Convenciones de legibilidad de código:** *Cómo organizar el código.*
- **Convenciones de documentación:** *Cómo establecer comentarios, archivos de ayuda, entre otros.”*[8]

Capítulo 2 Diseño del Sistema

Los estándares de codificación permiten una mejor integración entre las líneas de producción y establece pautas que conlleven a lograr un código más legible y reutilizable, de tal forma que permita escalabilidad a lo largo del tiempo.

Notación húngara: *“Esta convención se basa en definir prefijos para cada tipo de datos y según el ámbito de las variables. También es conocida como notación REDDICK (por el nombre de su creador). La idea de esta notación es la de dar mayor información al nombre de la variable, método o función definiendo en ella un prefijo que identifique su tipo de dato y ámbito.”* [8]

Notación PascalCasing: *“Es como la notación húngara pero sin prefijos. En este caso, los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula.”* [8]

Notación CamelCasing: *“Es parecido al Pascal-Casing con la excepción que la letra inicial del identificador no debe estar en mayúscula.”*[8] Esta notación se utilizó para el nombre de las funciones y el nombre de los atributos.

Para la implementación del componente se siguió la política estándar de diseño a cumplir por todas las aplicaciones desarrolladas en el centro UCID. Es importante que toda el área implicada siga la misma línea, que se cumplan con las reglas y las responsabilidades del trabajo en cuestión. La misma no podrá modificarse. Para llevar a cabo una modificación deberá dirigirse al responsable dentro del equipo de arquitectura que atiende el tema. De considerarse y materializarse el cambio con previa aprobación por la dirección del centro, se pondrá en vigor el documento con las nuevas modificaciones.

2.3. Patrones de diseño

Definidos como una solución reutilizable para un problema que comúnmente ocurre durante el diseño del software, los patrones *“son plantillas o descripciones para saber cómo resolver un problema que puede ser utilizado en contextos distintos”* [9]. Ellos en sí forman un lenguaje que puede ser utilizado para describir soluciones clásicas al diseño de problemas comunes orientados a objetos; permiten analizar sistemas de objetos como entidades encapsuladas, lo cual da a conocer la estrecha relación

con la programación orientada a objetos. Pueden aumentar la rapidez del proceso de desarrollo proporcionando paradigmas sólidos. Un diseño efectivo de software requiere que se consideren problemas que no puedan ser detectados hasta que llega la Implementación; lo cual conlleva a mejorar la legibilidad para los programadores y arquitectos.

2.3.1. Patrones GRASP

En diseño orientado a objetos, GRASP acrónimo de *General Responsibility Assignment Software Patterns*, en español patrones generales de software para asignación de responsabilidades se considera que más que patrones propiamente dichos, son una serie de buenas prácticas de aplicación recomendable en el diseño de software.

Los patrones de diseños empleados en la implementación del componente fueron seleccionados del estándar de diseño para las aplicaciones de gestión creado en el centro UCID.

2.3.1.1. 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.

2.3.1.2. 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: Una de las formas que el sistema implementa este patrón a través de servicios; ejemplo de su aplicación: la clase `definirprocesoController.php` al necesitar datos del nomenclador `tipoproceso` y `periodoevaluativo`. Este obtiene la información mediante la creación de objetos de las clases `NomEvalTipoproceso.php` y `NomEvalperiodoevaluativo.php`, las cuales contienen la información necesaria para realizar las tareas .encomendadas.

2.3.1.3. 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 `DatEval[]Controller` ó `NomEval[]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 clases `DatEval[]` ó `NomEval[]` en caso de necesitar cargarlos.

2.3.1.4. 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 los procesos evaluativos se relacionan con los tipo de procesos evaluativos y con los períodos evaluativos, no se decide la implementación de una clase general sino la creación de clases para cada proceso de gestión (`definirprocesoevaluativoController.php`, `tipoprocesoController.php` y `peridodoController.php`) evitando la sobrecarga, compresión y reutilización de funcionalidades.

2.3.1.5. Bajo acoplamiento

Problema: ¿Cómo dar soporte a una mínima dependencia y a un aumento de la reutilización?

Solución: Una clase con bajo acoplamiento no depende de “muchas otras” clases. Las clases con alto acoplamiento recurren a muchas clases y no es conveniente, además son más difíciles de mantener, entender y reutilizar.

Aplicación: Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

2.3.2. Patrones GoF

Por motivos de que se usan varios *frameworks* de desarrollo y estos a su vez implementan independientemente sus propios patrones de diseños, a continuación se presentará una breve descripción de los principales patrones empleados por cada una de las capas.

Vista: Implementa el patrón *Decorator* (del español envoltorio) en la clase *Zend_View*, encargada de asignarle responsabilidades a objetos de manera dinámica y configurarlos con nuevos atributos. También se implementa el patrón *Facade* (del español Fachada) el cual proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de utilizar.

Controlador: *Zend Framework* tiene implementado el patrón *Front Controller* implica que todas las solicitudes son dirigidas a un único script PHP que se encarga de instanciar al controlador frontal y redirigir las llamadas. Además tiene una instancia única del controlador frontal disponible mediante el patrón *Singleton* (traducido al español Instancia Única) para lograr una vía de entrada única a las solicitudes.

Modelo: Debido a que *Doctrine* utiliza el patrón *Active Record* para manejar la base de datos puede acceder a la misma utilizando la programación orientada a objetos. Además, implementa el patrón *Abstract Factory* (en español fábrica abstracta) el cual permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando, es decir, crea instancias de clases concretas a partir de una superclase abstracta.

2.4. Diagrama de clases web

El diagrama de clase es el diagrama principal de diseño y análisis para un sistema. 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.

2.4.1. Diagrama de clases genérico de las vistas

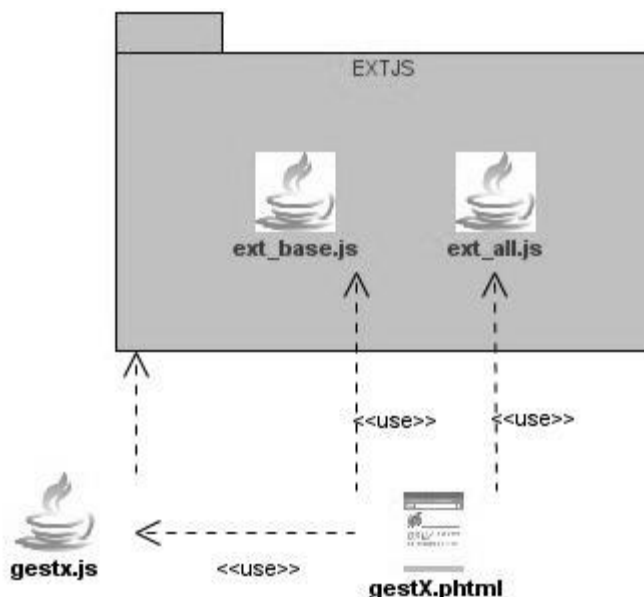


Figura 5 Diagrama genérico de las clases de la vista.

2.4.2. Descripción de las clases genéricas de la vista

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. Está incluida en el paquete original.

Tabla 1 Descripción de la clase ext-base.js

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.

Tabla 2 Descripción de la clase ext-all.js

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

Tabla 3 Descripción de la clase genérica gestX.phtml

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

Tabla 4 Descripción de la clase genérica gestx.js

2.4.3. Diagrama de clases genérico

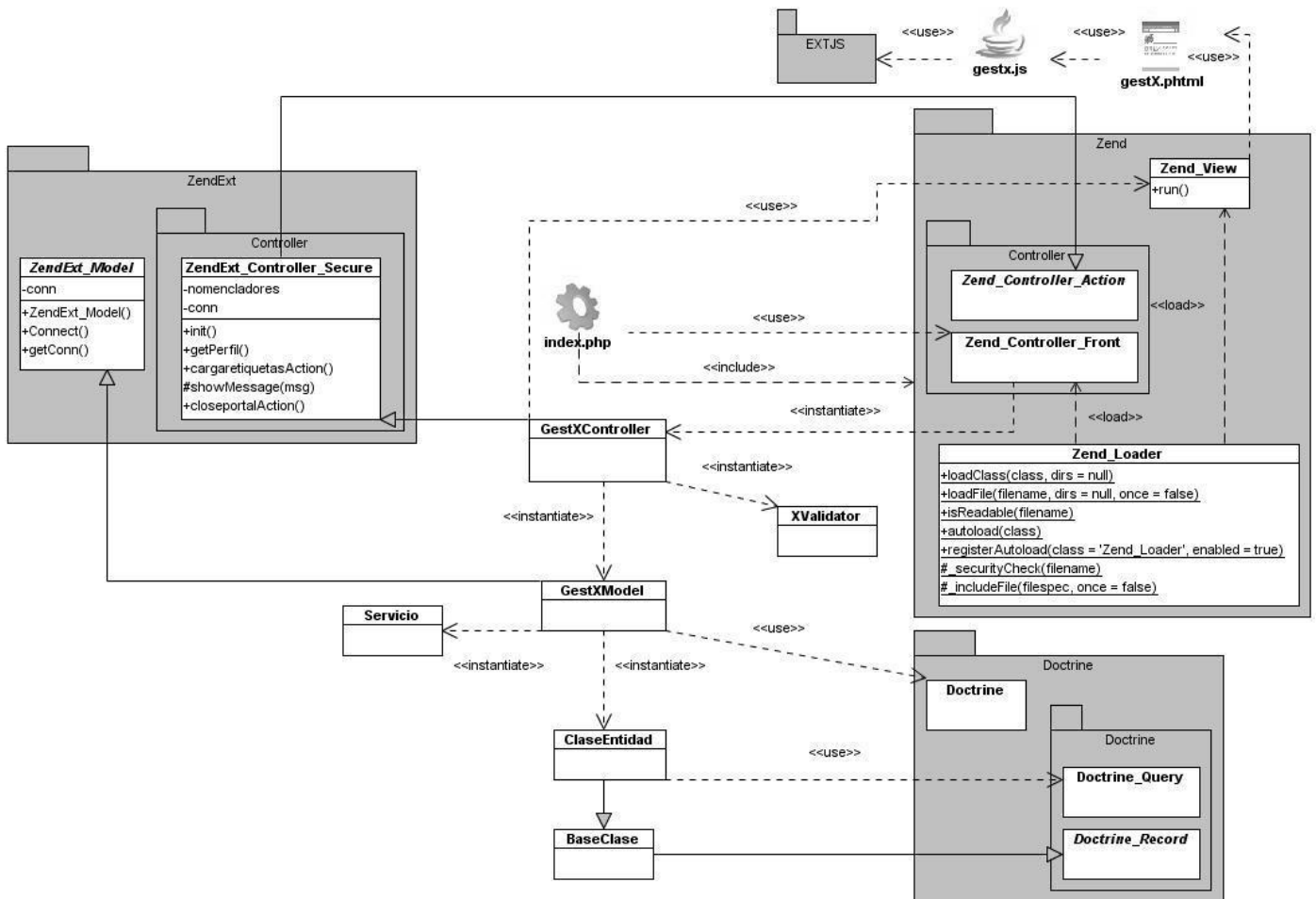


Figura 6 Diagrama de clases genérico.

2.4.4. Descripción de las clases

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

Tabla 5 Descripción de la clase Zend_Controller_Action.php

Nombre:	Zend_Controller_Secure.php
Tipo de clase:	controladora
Descripción:	De está clase deben heredar todas las controladoras. Es un controlador de acciones

Capítulo 2 Diseño del Sistema

	personalizado e integrado a seguridad.
--	----------------------------------------

Tabla 6 Descripción de la clase Zend_Controller_Secure.php

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

Tabla 7 Descripción de la clase genérica GestXController.php

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

Tabla 8 Descripción de la clase Zend_View.php

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 <i>index.php</i> .

Tabla 9 Descripción de la clase Zend_Controller_Front.php

Nombre: Index.php	
Tipo de clase: servidora	
Descripción:	Constituye el único punto de acceso a la aplicación, conjuntamente con la clase <i>Zend Loader</i> y <i>Zend_Controller_Front</i> se encarga del funcionamiento de la aplicación, atención a solicitudes y respuestas.

Tabla 10 Descripción de la clase *Index.php*

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

Tabla 11 Descripción de la clase ZendExt_model.php

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.

Tabla 12 Descripción de la clase genérica GestXModel.php

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 <i>doctrine_record</i> .

Tabla 13 Descripción de la clase BaseClase.php

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

Tabla 14 Descripción de la clase Clase.php

Capítulo 2 Diseño del Sistema

2.4.5. Diagrama de clases del diseño gestionar proceso evaluativo

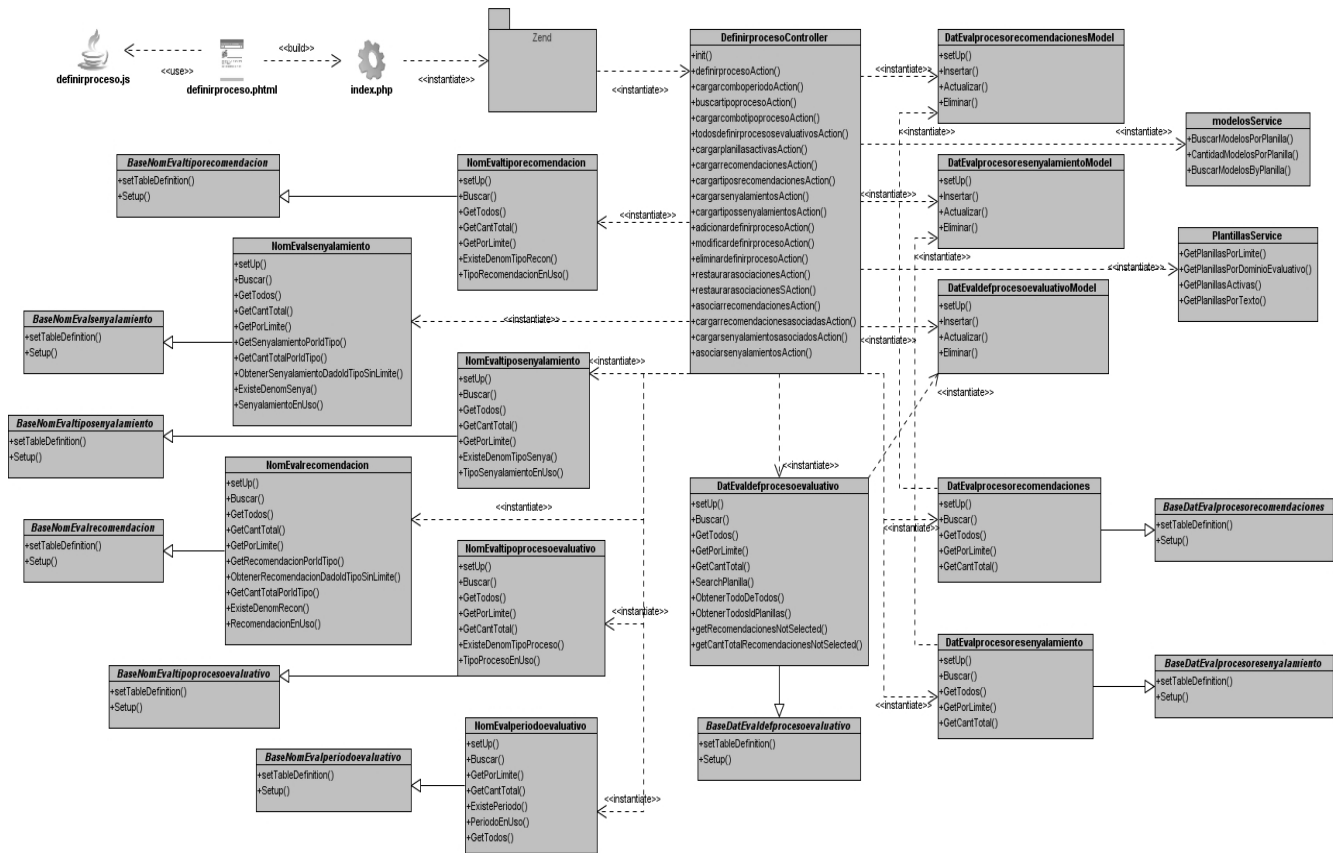


Figura 7 Diagrama de clase del diseño Gestionar definir proceso.

2.4.6. Descripción de las clases

Nombre: definirproceso.js	
Tipo de clase: interfaz.	
Para cada responsabilidad:	
Nombre:	cargarAyuda()
Descripción:	Carga en una nueva ventana la ayuda de la funcionalidad.
Nombre:	cargarInterfaz()
Descripción:	Función principal, encargada de mostrar la interfaz.
Nombre:	cargarVentanaGestionarDefinirproceso()
Descripción:	Función encargada de gestionar la adición y modificación de procesos evaluativos.
Nombre:	cargarVentanaAsociarDefinirproceso()
Descripción:	Función encargada de gestionar la asociación de señalamientos y recomendaciones a los procesos evaluativos.
Nombre:	renderImage()
Descripción:	Cambia los valores true o false por imágenes que ilustren su significado.

Tabla 15 Descripción de la clase definirproceso.js

Capítulo 2 Diseño del Sistema

Nombre: definirprocesoController.php	
Tipo de clase: controladora	
Para cada responsabilidad:	
Nombre:	init()
Descripción:	Inicializa el controlador.
Nombre:	adicionadefinirprocesoAction()
Descripción:	Crea un objeto de la tabla de la base de datos DatEvaldefprocesoevaluativo y ejecuta la función del modelo para insertarlo en la base de datos.
Nombre:	modificardefinirprocesoAction()
Descripción:	Recoge los valores a modificar entrados en el formulario y ejecuta la función del modelo encargado de modificar los datos existentes en la tabla de la base de datos DatEvaldefprocesoevaluativo.
Nombre:	eliminardefinirprocesoAction()
Descripción:	Ejecuta la función del modelo que se encarga de eliminar el valor enviado por el formulario.
Nombre:	cargarcomboperiodoAction()
Descripción:	Carga de la base datos los períodos.
Nombre:	cargarcombopropocesoAction()
Descripción:	Carga de la base datos los tipos de procesos evaluativos.
Nombre:	todosdefinirprocesoevaluativosAction()
Descripción:	Carga de la base datos toda la información a mostrar en el contenedor principal (Grid).
Nombre:	cargarplanillasactivasAction()
Descripción:	Cargas las planillas activas y sus respectivos modelos matemáticos.
Nombre:	cargarrecomendacionesAction()
Descripción:	Cargar las recomendaciones a asociar.
Nombre:	cargartiposrecomendacionesAction()
Descripción:	Cargar los tipos de recomendaciones a asociar.
Nombre:	cargarsenalamientosAction()
Descripción:	Cargar los señalamientos a asociar.
Nombre:	cargartiposseñalamientosAction()
Descripción:	Cargar los tipos de señalamientos a asociar.
Nombre:	restaurarasociacionesAction()
Descripción:	Restablece las recomendaciones seleccionadas previamente asociadas.
Nombre:	restaurarasociacionesSAction()
Descripción:	Restablece los señalamientos seleccionados previamente asociados.
Nombre:	asociarrecomendacionesAction()
Descripción:	Asocia las recomendaciones seleccionadas.
Nombre:	cargarrecomendacionesasociadasAction()
Descripción:	Carga de la base dato las recomendaciones asociadas.
Nombre:	cargarsenalamientosasociadosAction()
Descripción:	Carga de la base dato los señalamientos asociados.
Nombre:	asociarsenalamientosAction()
Descripción:	Asocia los señalamientos seleccionados.

Tabla 16 Descripción de la clase definirprocesoController.php

Nombre: NomEvaldefinirprocesoModel.php	
Tipo de clase: modelo	

Capítulo 2 Diseño del Sistema

Para cada responsabilidad:	
Nombre:	setUp()
Descripción:	Controlador de la clase.
Nombre:	Insertar(\$val)
Descripción:	Inserta en la tabla DatEvaldefprocesoevaluativo la información contenido en \$val.
Nombre:	Actualizar(\$val)
Descripción:	Modifica en la tabla DatEvaldefprocesoevaluativo la tupla correspondiente con el objeto pasado por parámetro.
Nombre:	Eliminar(\$idval)
Descripción:	Elimina de la tabla DatEvaldefprocesoevaluativo la tupla cuyo identificador es igual al pasado por parámetros.

Tabla 17 Descripción de la clase NomEvaldefinirprocesoModel.php

Nombre: DatEvaldefprocesoevaluativo.php	
Tipo de clase: modelo	
Para cada responsabilidad:	
Nombre:	setUp()
Descripción:	Controlador de la clase.
Nombre:	Buscar(\$idval)
Descripción:	Devuelve la tupla cuyo id es igual al pasado por parámetro.
Nombre:	GetTodos()
Descripción:	Devuelve todas las tuplas de la tabla
Nombre:	GetCantTotal()
Descripción:	Devuelve la cantidad total de tuplas que tiene la tabla
Nombre:	GetPorLimite(\$limite = 15,\$inicio = 0)
Descripción:	Devuelve una cantidad de tuplas igual a la pasada por \$limite y se comenzara a por la tupla igual a inicio.
Nombre:	ObtenerTodoDeTodos()
Descripción:	Recoge de la base de datos todos los valores de las tablas DatEvaldefprocesoevaluativo, NomEvalperiooevaluativo y NomEvaltipoprocesoevaluativo.

Tabla 18 Descripción de la clase DatEvaldefprocesoevaluativo.php

2.4.7. Diagrama de secuencia listar definir proceso evaluativo

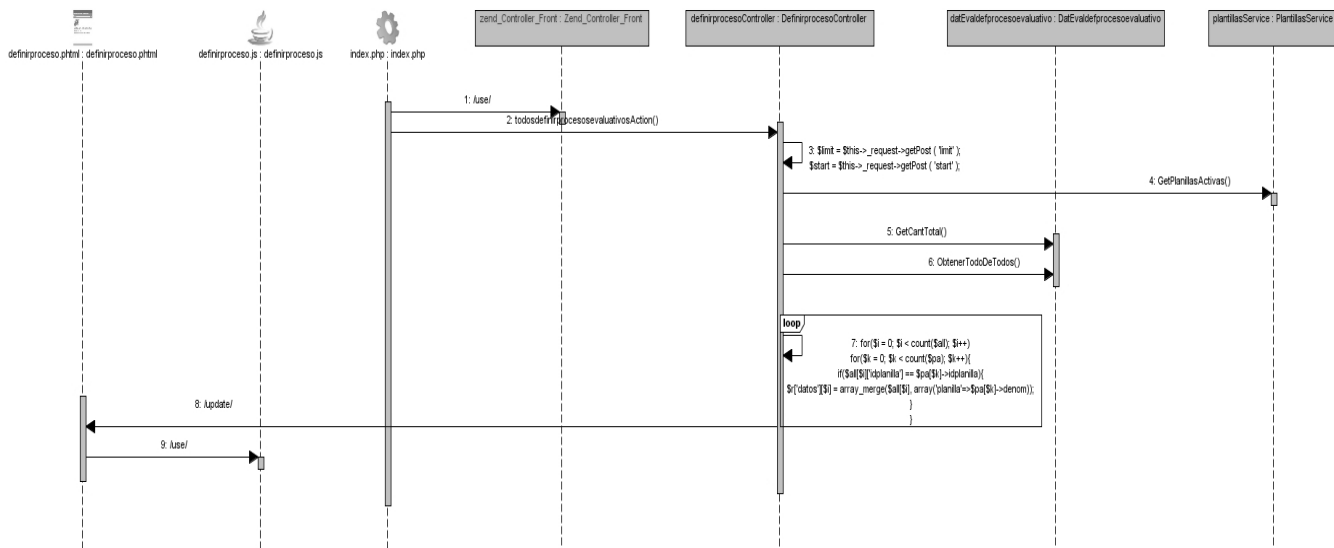


Figura 8 Diagrama de secuencia listar definir proceso evaluativo

2.4.8. Diagrama de secuencia Adicionar definir proceso evaluativo

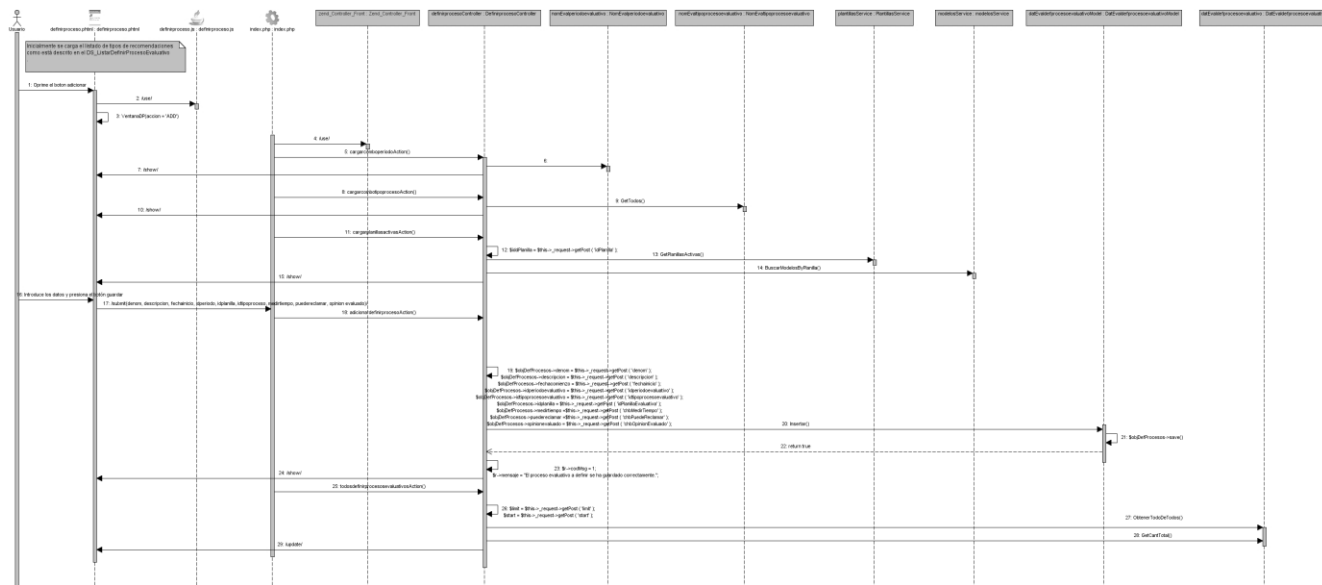


Figura 9 Diagrama de secuencia Adicionar definir proceso evaluativo

2.4.9. Diagrama de secuencia modificar definir proceso evaluativo

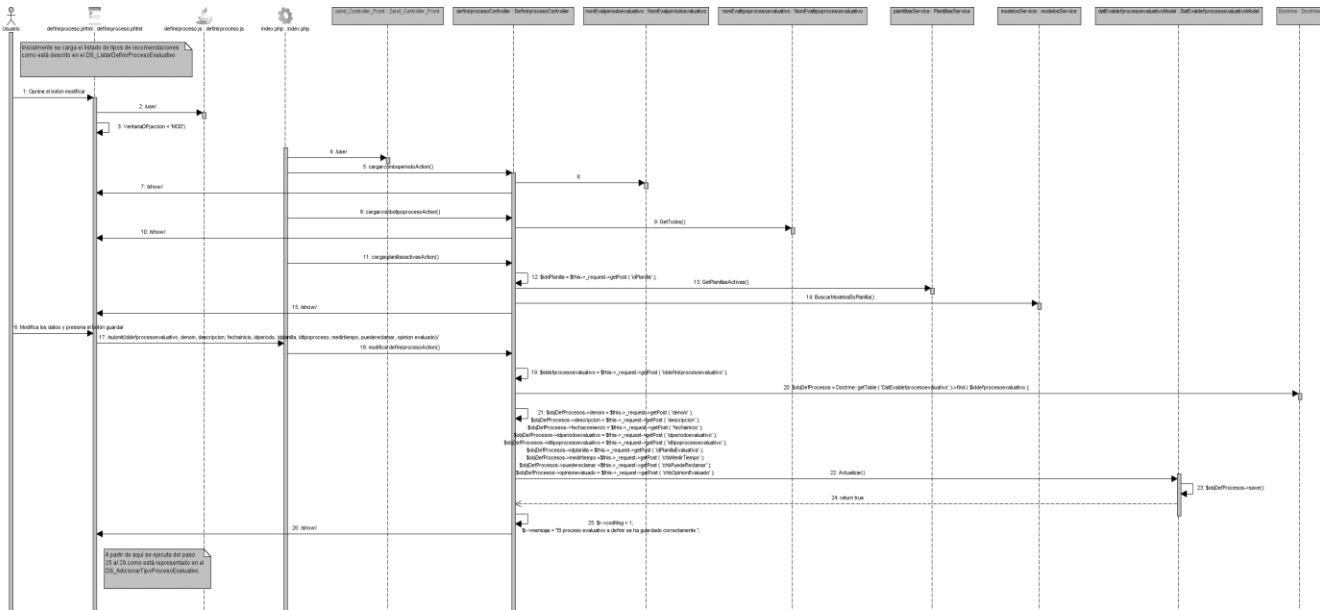


Figura 10 Diagrama de secuencia modificar definir proceso evaluativo

2.4.10. Diagrama de secuencia eliminar definir proceso evaluativo

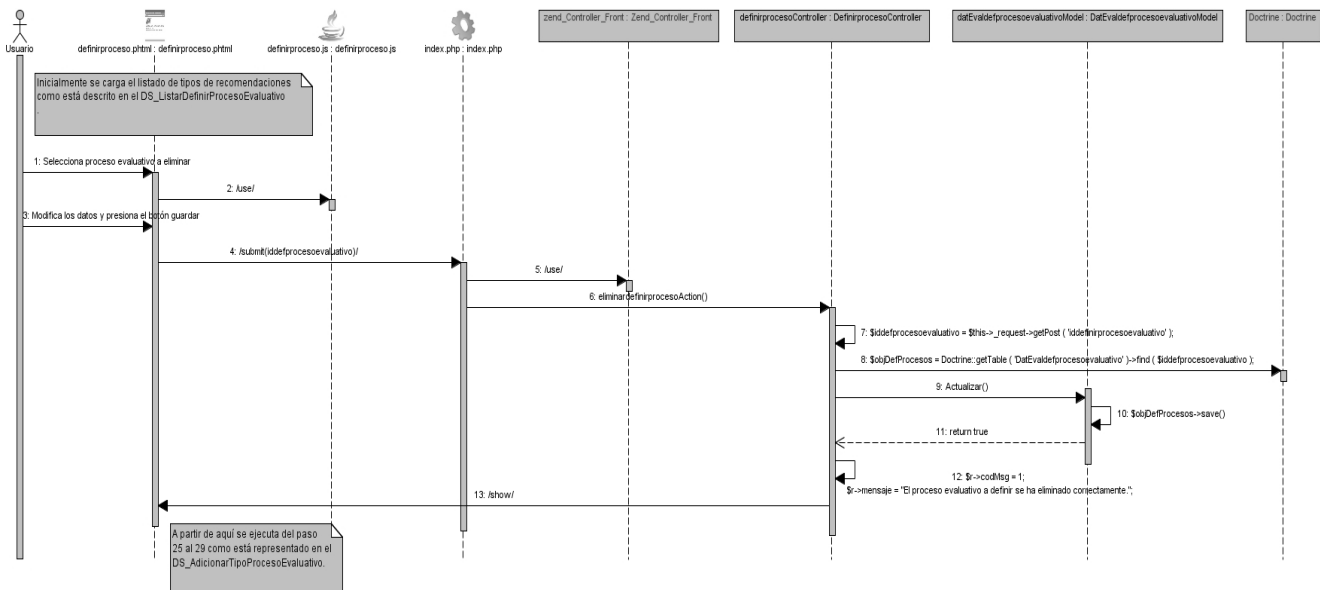


Figura 11 Diagrama de secuencia eliminar definir proceso evaluativo

Capítulo 2 Diseño del Sistema

Para ver los restantes diagramas de clases del diseño con sus respectivos diagramas de secuencia además de sus descripciones de clases dirigirse a ([Anexo 1](#)).

2.7. Diseño de la base de datos

Uno de los elementos fundamentales de los sistemas de gestión de información es la persistencia de los datos y la forma en la que se organizan. Los organismos constantemente consultan y realizan análisis de la información con la que se maneja, por lo que se hace necesario una forma de que los datos perduren a través del tiempo sin sufrir daño alguno logrando así una mejor distribución entre ellos. El diagrama de base de datos que se plantea a continuación puede estar sujeto a cambios debido a que el sistema se le va a agregar más funcionalidades en una posterior etapa.

2.8. Diagrama entidad-relación de la base de datos

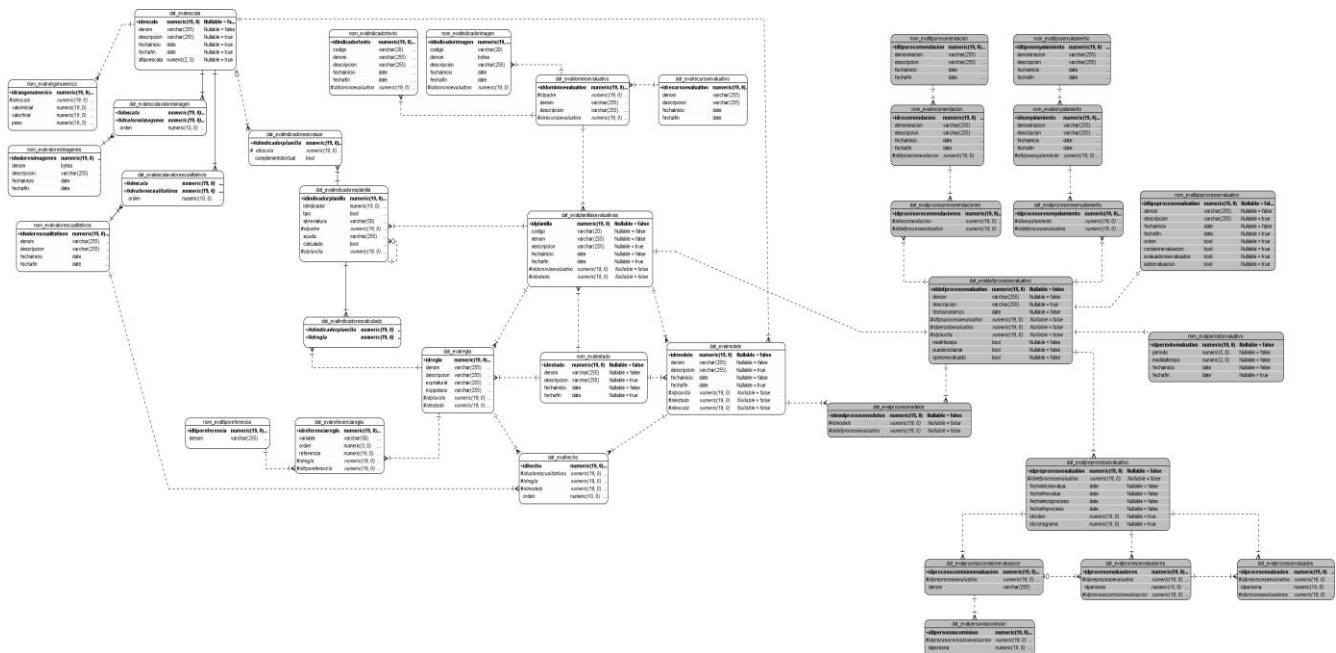


Figura 14 Diagrama entidad-relación de la base de datos

2.9. Descripción de las tablas de la base de datos

Nombre: DatEvaldefprocesoevaluativo		
Descripción:		
Atributo	Tipo	Descripción

Capítulo 2 Diseño del Sistema

iddefprocesoevaluativo	numeric(19,0)	Identificador de la tabla
denom	varchar(255)	Denominación del proceso evaluativo.
descripcion	varchar(255)	Descripción del proceso evaluativo.
fechacomienzo	date	Fecha de inicio del proceso evaluativo
idperiodoevaluativo	numeric(19,0)	Llave foránea e identificador de la tabla Nomevalperiodoevaluativo
idplanilla	numeric(19,0)	Llave foránea e identificador de la tabla Datevalplanillaevaluativa
idtipoprocesoevaluativo	numeric(19,0)	Llave foránea e identificador de la tabla Nomevaltipoprocesoevaluativo
medirtiempo	boolean	Valor que especifica si se medirá el tiempo o no.
opinionevaluado	boolean	Valor que especifica si el evaluado podrá opinar o no.
puedereclamar	boolean	Valor que especifica si el evaluado podrá reclamar o no.

Tabla 19 Descripción de la tabla DatEvaldefprocesoevaluativo

Para ver restantes descripciones de tablas de la base de datos ([Anexo 2](#)).

2.10. Tratamiento de errores

El tratamiento de errores es uno de los elementos de suma importancia a la hora de implementar una aplicación, la misma se realiza con el objetivo de tratar el lanzamiento de excepciones. Dentro del marco de trabajo se encuentran definidas en estos momentos una serie de excepciones donde cada una tiene un código, un tipo y el idioma en que se muestra el mensaje (para el inglés con las siglas en y español con las siglas es), hay que aclarar que el mensaje tiene a su vez un nombre y una descripción. Las mismas se encuentran almacenadas en la carpeta “../comun/recursos/xml”.

2.10.1. Validación en el servidor

La validación en el servidor consiste en validar los campos que serán introducidos en inserciones a la base de datos, se comprueba que están correctos antes de introducirlos en las tablas. Este componente que crea el equipo de arquitectura se muestra en el caso de estudio a través del ejemplo de modificar una fila en el XML que se identifica por *validation* y se ubica en la carpeta de recursos comunes del proyecto ya que en la nueva estructura los XML se ubicarán todos en dicho directorio. Para la acción de modificar las etiquetas del fichero serían:

```
<validador>
```

Capítulo 2 Diseño del Sistema

```
<procesoevaluativo> Subsistema
<senalamientoController> Clase control
<modificarsenalamientoAction> Método donde se van a validar los campos.
<precondition> Indica que se va a ejecutar antes que el método.
    <validator class="SenyalamientoValidator"
        method="ExisteSenyalamiento" error="EV008"/>
```

En `<validator class>` se indica la clase almacenada en la carpeta `validators` donde se encuentra el método que se va a cargar antes de `modificarsenalamiento`. Las clases de validación deben estar contenidas dentro del componente en la carpeta `validators` debido a que el `framework` al leer el XML busca la clase en dicho directorio. En `precondición` se especifica cual va a ser la validación que se va a realizar de ejecutar el método `modificarsenalamiento`. Puede haber tantas `precondiciones` como sean necesarias y lo mismo sucede con las `postcondition` que no son más que las validaciones que se ejecutarán después del método en la `controller` en cuestión. Se especifica el código del error que se va a lanzar en caso de que la `precondición` no se cumpla. Si el método devuelve falso se dispara la excepción registrada en el xml de `exception` con el código que se especifica (`error="EV008"`).

```
<validator class="SenyalamientoValidator" method=" validarConfirmacionClave " error="EV009"/>
</precondition> Fin de la primera precondición.
<precondition>
<validator class="SenyalamientoValidator"method=" validarExistenciaSenyalamiento
"error="EV010"/>
</precondition>
<param name="idsenalamiento" type="enterospos" not_null="true" null_error="EV001"
type_error="EV002"/>
```

Validación de los parámetros, se indica el identificador del parámetro (`name="idsenalamiento"`) el código del tipo que debe ser (este código se registra en el XML de expresiones regulares, `expressions`. XML con la expresión regular indicando los caracteres que puede soportar el tipo de dato) en el caso de id de la persona debe ser un entero mayor que cero, si el parámetro puede o no ser nulo, o sea `not_null` se indica en `true` si no puede ser vacío el campo y `false` si se permite entrar el campo vacío. Se especifica el código de la excepción que se va a lanzar en el caso de que `not_null=true` y sin

embargo el parámetro está vacío, de la forma `null_error="EV001"` y la excepción que se va a lanzar en el caso de que el parámetro no sea del tipo requerido, `type_error="EV002"`.

```
</modificarsenyalamientoAction>  
</SenyalamientoController>  
</procesoevaluativo>  
</validador>
```

2.10.2. Excepciones

Con la implementación y puesta en marcha del manejador de excepciones no será necesario realizar el lanzamiento de excepciones en cada funcionalidad de las clases, solamente se deberá registrar el método y la excepción en el xml que lleva por título `managerexception.xml` de la siguiente manera:

```
<? xml version="1.0" encoding="UTF-8"?>  
<exceptions>  
<procesoevaluativo> Componente donde se registra la excepción.  
<SenyalamientoController> Clase donde se realiza el tratamiento de excepciones.  
<todossenyalamientoAction>  
<exception type="Doctrine_Excepcion" code="EGU001" />  
</todossenyalamientoAction>  
<ModificarsenyalamientoAction> Método donde puede ocurrir.  
<excepcion type="Doctrine_Excepcion" code="EGU004" />
```

Se especifica que cuando ocurra una excepción de Doctrine (`type="Doctrine_Excepcion"`) se realizará su tratamiento mediante el xml `exception` donde se encuentra registrada una excepción cuyo código es "EGU003" (`code="EGU004"`).

```
</modificarsenyalamientoAction>
```

Termina la especificación o el tratamiento de las excepciones para el método `modificarsenyalamiento`.

```
</SenyalamientoController>
```

Capítulo 2 Diseño del Sistema

Termina la especificación o el tratamiento de las excepciones de la clase controladora. Aclarar que si dentro de la clase se quieren tratar las excepciones de otros métodos se realizan las etiquetas de cada uno antes de cerrar la de la clase control (</SenyalamientoController>).

</procesoevaluativo>

Termina la especificación o el tratamiento de las excepciones del componente. Aclarar que si dentro del componente se quieren indicar las excepciones que pueden ocurrir en otras clases control antes de cerrar la etiqueta del subsistema se especifica el tratamiento de igual forma a la anteriormente mostrada.

</exceptions> Termina la especificación de todas las excepciones.

Es muy importante tener en cuenta cuando se dispara una excepción en el *framework* a través de los xml llamados *managerexception* y *exception* realizará todo el tratamiento que conlleva.

2.11. Seguridad

La seguridad de manera general consiste en garantizar que los recursos del sistema de información (material informático o programas) de una organización sean utilizados de la manera que se decidió y, que el acceso a la información allí contenida, así como su modificación, sólo sea posible a las personas que se encuentren acreditadas y dentro de los límites de su autorización. Es importante aclarar que es prácticamente imposible tener un sistema seguro al 100% en todas las circunstancias.

2.11.1. Seguridad del sistema

La seguridad es un factor de peso en la implementación de cualquier aplicación, esta verifica que la información no pueda ser visualizada por personal no autorizado, o utilizada con fines impropios o contrarios a las políticas del cliente. En la actualidad no se concibe un software de ningún tipo sin pensar en un mecanismo de seguridad que lo respalde.

En el subsistema la seguridad está garantizada mediante el consumo de servicios pertenecientes a la línea de seguridad del Sauxe. La estructura de la jerarquía del nivel de seguridad está establecida de tal manera que un usuario pueda acceder solo a las funcionalidades y acciones que se les ha asignado. De manera global se puede decir que un usuario tiene asociado uno o varios roles y cada

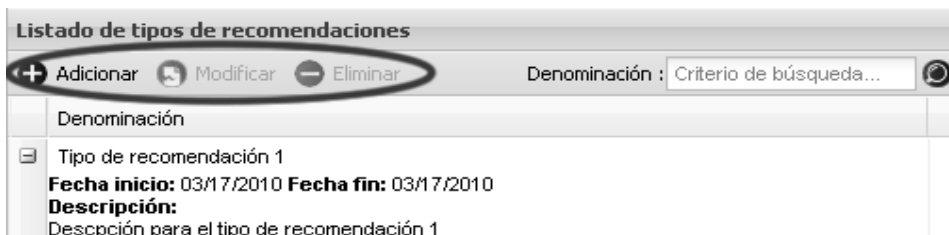


Figura 15 Ejemplo de acciones que puede tener una funcionalidad.

uno de estos trae consigo funcionalidades y acciones, donde las funcionalidades son las interfaces terminales que responden a cada uno de los requisitos funcionales ([Anexo 3](#)) y las acciones no son más que métodos implementados dentro de esta o las facilidades que te permite realizar la funcionalidad (Figura 15).

2.11.2. Inyección SQL

La Inyección de SQL (Lenguaje de Consulta Estructurado) es un ataque en el cual el código malicioso es insertado en cadenas de texto que luego son pasadas al servidor o instancia de SQL para ejecución [10]. Es inminente que los procedimientos que construyen las declaraciones deban ser revisados ya que el servidor SQL ejecutará todas las consultas válidas sintácticamente que reciba. Los ataques suelen ser muy complejos e inteligentes y principalmente dirigidos hacia los datos más vulnerables. Inicialmente la inyección SQL consiste en la inserción directa de código a variables que toman valores de acuerdo con lo que el usuario introduce y que luego son concatenados con comandos SQL y luego ejecutados. El proceso de inyección es prematuramente terminar una cadena de texto y luego anexar un comando. Esto tiene una muy cercana relación con los ataques que son dirigidos al almacenamiento de datos o metadatos en las tablas, cuando las cadenas son guardadas y subsecuentemente atadas a un dinámico comando SQL, entonces es cuando el código malicioso es ejecutado. Por lo tanto, se deben tomar medidas y planear la forma en que se validará la integridad del sistema en contra de este tipo de ataques ya que hay que tener en cuenta las habilidades que pueden tener los hackers: Implementar múltiples capas de validación. Las precauciones que se toman contra los usuarios maliciosos pueden resultar ser inefectivos contra atacantes determinados.

Una buena práctica es validar las entradas en la interfaz del usuario y así todos los puntos y capas por donde pasan los valores. Validaciones de datos en una aplicación del lado del cliente pueden prevenir inyecciones simples de scripts. De todas formas, si en la próxima capa de validaciones se asume que la entrada ya se ha validado, cualquier usuario malicioso que pueda sobrepasar un cliente puede tener

Capítulo 2 Diseño del Sistema

acceso no restringido al sistema. La solicitud de servicios online como viajes y transportes por parte de los usuarios, por muy básica que sea, es muy vulnerable.

2.11.3. XSS (Cross-site Scripting)

Cross-site Scripting (Ataques a navegadores mediante scripts) o XSS es una vulnerabilidad usualmente encontrada en aplicaciones web en la cual se usan códigos scripts para atacar a otros usuarios [10]. Pueden ser utilizadas para sobrepasar controles de acceso y corromper el comportamiento del navegador en general. El ejemplo más común de ataques XSS es en una entrada de Blog o Fórum donde el usuario final puede adicionar contenido a la aplicación web para que otros la consulten. En una entrada donde se permitan comentarios cualquier usuario puede escribir contenido dentro del campo de comentarios que luego va a ser insertado en la base de datos o, más allá de esto, el código malicioso escrito puede re-direccionar la página mostrada a otro sitio incluyendo los datos de la sesión y las cookies registradas.

Hay que llevar a cabo políticas de seguridad en la etapa de desarrollo de las aplicaciones e ir construyendo la conciencia de utilizar al máximo las funciones y utilidades que brinda PHP para el reconocimiento y filtrado de la información manipulada ya que los ataques no solo pueden ser a través de campos de entrada o selección, sino también por vías mucho más complejas de manipular que son las peticiones y las cabeceras; los atacantes aprovechan la dependencia de aplicaciones de manejar datos confidenciales y de solicitudes que son enviadas a través de la red y con la ayuda de software y scripts que generan código y lo adjuntan en el momento preciso en que la información se encuentra “en el aire”. La gestión de Reservaciones e Información que realiza el Sistema Integrado de Transportación incluye validaciones contra este tipo de ataques y tiene máxima prioridad ya que estos módulos tienen estrecha relación el usuario final y son los encargados de presentar contenidos que pueden variar en dependencia de lo solicitado.

2.12. Propuesta del Sistema

Uno de los primeros pasos que se realiza en una entidad para llevar a cabo la evaluación de desempeño es definir un proceso evaluativo y acto seguido prepararlo. Para ello el sistema le dará al usuario según sus privilegios la opción de acceder al menú que contendrá las funcionalidades para la

Capítulo 2 Diseño del Sistema

gestión de dicho proceso. Para poder definir un nuevo proceso evaluativo primeramente, es necesario haber definido previamente una serie de datos gestionados por los nomencladores. Estos son: las recomendaciones, los tipos de recomendaciones, los señalamientos, los tipos de señalamientos, los tipos de procesos evaluativos y los períodos.

La recomendación no es más que una observación o sugerencia que le proporciona un evaluador a la persona que está evaluando en ese momento. Un proceso evaluativo puede tener tantas recomendaciones como el especialista en evaluaciones estime conveniente y a su vez estas se encuentran organizadas por el tipo de recomendación, el cual tiene como función principal clasificar a las recomendaciones para gestionarlas de manera más ágil ([Anexo 4](#)). El principio de funcionamiento del señalamiento es muy similar al de las recomendaciones con la diferencia de que estas, tienen un carácter negativo, es decir, son indicaciones que se proporciona al evaluado para que mejore en determinados aspectos y al igual que las recomendaciones presentan un tipo de señalamiento que funciona como una especie de clasificador, con los mismo objetivos que el nomenclador tipo de recomendación pero orientado hacia los señalamientos ([Anexo 5](#)). El nomenclador período es el encargado de definir con que sistematicidad se realizará este tipo de evaluación, dígame cada tres años, anual o una frecuencia definida por el especialista en evaluaciones. Y por último el nomenclador tipo de proceso evaluativo, el cual tiene una importancia considerable, debido a que en el se define parte de la estructura que va a tener el proceso evaluativo a la hora de ser preparado. Cada proceso evaluativo va a tener un tipo de proceso en cual va a caracterizarlo es decir, especificar si este va a tener comisión, evaluados, evaluadores y orden. ([Anexo 6](#)). Una vez que se tienen estos nomencladores se puede pasar a definir el proceso evaluativo el cual tiene como principal objetivo detallar como éste va a estar estructurado ([Anexo 7](#)).

Para preparar un proceso evaluativo no es necesario, como en el caso anterior de definir proceso evaluativo gestionar nomencladores. Cuando se prepara un proceso evaluativo primeramente se debe gestionar y seleccionar la o las comisiones, los evaluadores y los evaluados. Los evaluados se asocian a los evaluadores y estos a su vez a las comisiones, es decir, una comisión va a atender a cierto número de evaluadores y cada uno de estos estará a cargo de un o más evaluados. Una vez seleccionado estos datos se procede a la gestión de la preparación del proceso evaluativo, el cual consiste en especificar la fecha de inicio y de fin del proceso evaluativo y del período que se evalúa,

para esto se tiene que tener en cuenta que dos o más procesos evaluativos no pueden coincidir en un mismo período de evaluación.

2.13. Concepción de la ayuda

En la mayoría de las aplicaciones existentes en el mercado, al presionarse las teclas “Esc” o “F1” se accede a una serie de informaciones sobre el programa en cuestión y cómo manejarlo, que se denominan genéricamente ayuda. Suelen ser un resumen de las instrucciones recogidas en los manuales que se adjuntan con todo programa. Los archivos de ayuda son una fuente de información valiosa para los usuarios de la aplicación. En el sistema la ayuda se concibe por funcionalidades, de tal manera que el usuario desde cualquier interfaz pueda auxiliarse en el empleo de esta y su finalidad (Figura 16).

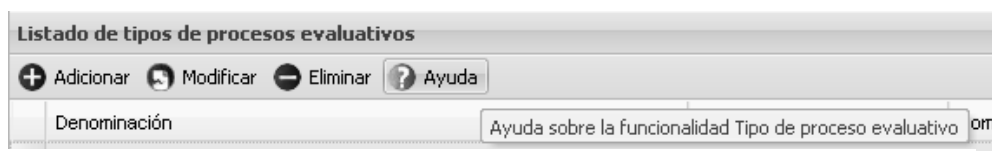


Figura 16 Ejemplo del uso de la ayuda.

2.14. Conclusiones parciales

Con el desarrollo de este capítulo se conocieron los diagramas de clases del diseño y los diagramas de secuencia correspondientes, lo que proporcionó una temprana idea de la complejidad de la solución. Se explicaron las definiciones de diseño aplicadas para el desarrollo de la aplicación, teniendo en cuenta la arquitectura del sistema. Todo lo anterior permitió la implementación de las interfaces. La descripción de clases importantes permitió tener una visión para la puesta en práctica de patrones GRASP y GoF. Se describió el uso de tratamiento de errores en la aplicación y cómo se ve reflejada la seguridad en el sistema. Finalmente, se expone una propuesta de solución la cual describe el flujo de eventos para definir y preparar un proceso evaluativo auxiliándose de las interfaces diseñadas adjuntadas en los anexos.

Capítulo 3 Implementación y Validación

Capítulo 3: Implementación y validación de la solución propuesta.

3.1. Introducción

En el presente capítulo, a través de la representación del diagrama de componentes y la matriz de integración se exhibe la interrelación tanto interna como externa de los componentes del sistema de manera general. Se desarrolla una búsqueda de las pruebas de unidades que permitan validar la solución propuesta, se realiza una descripción del mismo teniendo en cuenta: tipo de pruebas, objetivo de la prueba y alcance, se describen los valores utilizados y se evalúa la ejecución de la prueba y los resultados obtenidos.

3.2. Diagrama de componentes

“Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación” [11]. En la Figura 17 se muestra el diagrama de componentes.

Un diagrama de componentes representa los segmentos de aplicaciones, controladores embebidos y otros elementos que conformarán un sistema. Un diagrama de componentes tiene un nivel de abstracción mucho más elevado que un diagrama de clase. Usualmente un componente se implementa por una o más clases en tiempo de ejecución. Eventualmente un componente puede comprender una gran porción de un sistema.

Capítulo 3 Implementación y Validación

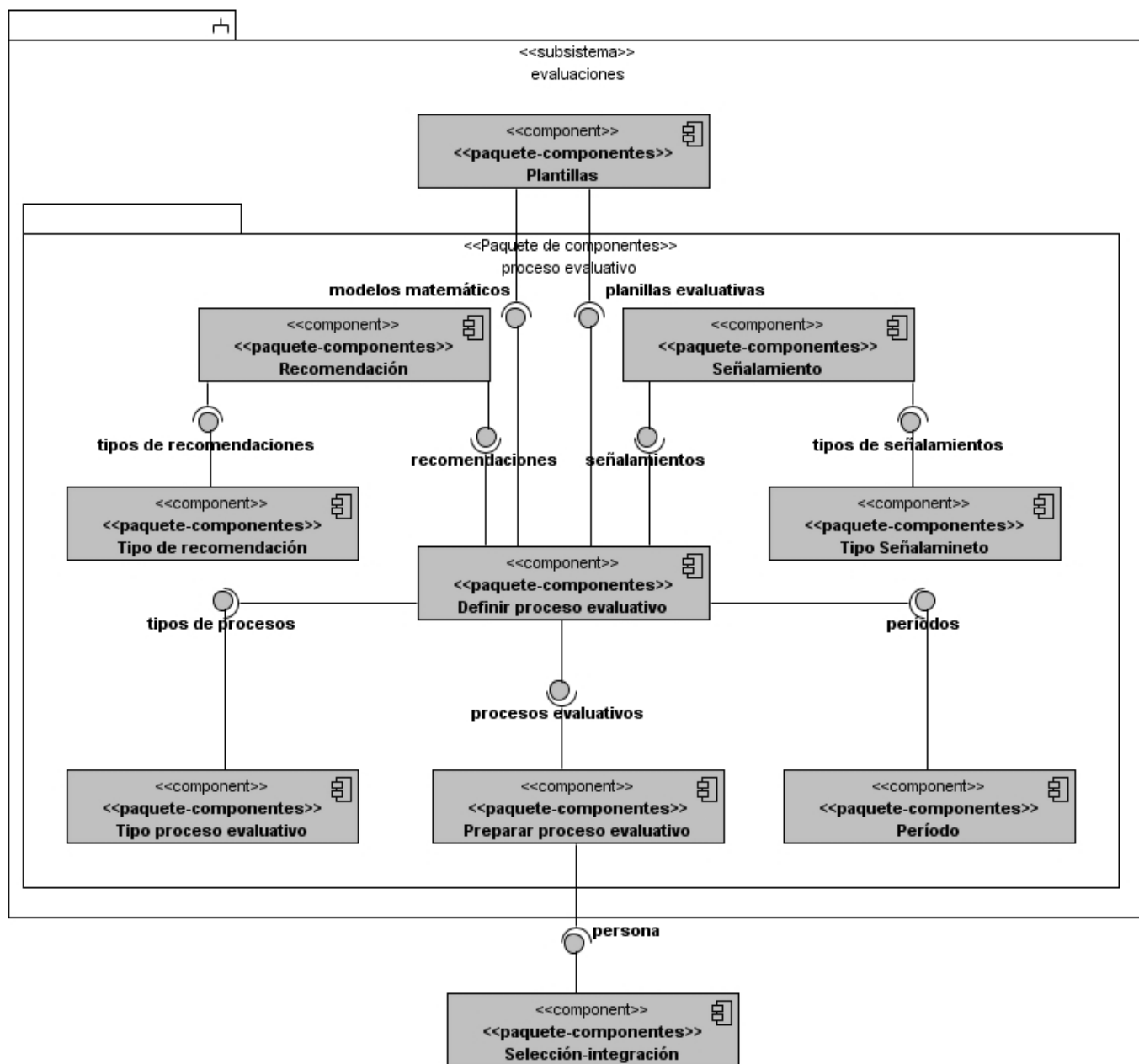


Figura 17 Diagrama de componentes.

3.3. Matriz de Integración de Componentes Internos

La matriz de integración de componentes contiene todos los componentes definidos en el subsistema, de forma matricial. En las intercepciones se especifican los servicios que consume el componente con

Capítulo 3 Implementación y Validación

respecto a su correspondiente. Existen dos matrices de integración de componentes, Interna y Externa.

Interna: en esta se especifica la integración entre los componentes internos del sistema.

Externa: en esta se especifica la integración entre los componentes externos del sistema.

Componentes Internos	Componentes Internos								
	Planillas evaluativas	Preparar proceso evaluativo	Tipo proceso evaluativo	Período evaluativo	Tipo de recomendación	Tipo de señalamiento	Recomendación	Señalamiento	Definir proceso evaluativo
Definir procesos evaluativos	-Planillas activas. - Modelos matemáticos	modelos matemáticos	tipo proceso	- período	tipo recomendación	tipo señalamiento	recomendación	señalamiento	---
Señalamiento	---	---	---	---	---	tipo señalamiento	---	---	---
Recomendación	---	---	---	---	tipo recomendación	---	---	---	---
Preparar proceso evaluativo	---	---	---	---	---	---	---	---	definir proceso

Tabla 20 Matriz de integración interna

Como se puede apreciar, la matriz muestra cómo se relacionan los componentes internos del sistema donde en la segunda fila se encuentran todos los componentes que brindan servicios o funcionalidades en la primera columna los componentes internos que consumen o precisan del servicio brindado. Ejemplo de esto se puede observar cuando el componente “Preparar proceso evaluativo” requiere una funcionalidad de “Definir proceso evaluativo”. A continuación se muestra la matriz de integración de componentes externos.

Componentes Internos	Componentes Externos
	Selección integración
Preparar proceso evaluativo	-Todas Personas por limite -Personas dado arreglo de Id -Cantidad total personas

Capítulo 3 Implementación y Validación

	- Todas Personas
--	------------------

Tabla 21 Matriz de integración externa

Al igual que la matriz de componentes internos, esta muestra la relación entre componentes, en este caso entre los internos y los externos. Como se presenta el componente “Preparar proceso evaluativo” a través de los servicios “Todas Personas”, “Cantidad Personas”, “Personas dado arreglo de id” y “Todas Personas por límites” obtiene el listado de todas las personas del componente “Selección integración” que se encuentran registradas en el sistema y mostrarlos de acuerdo con el servicio que se requiera, en su correspondiente contenedor.

3.4. Modelos de prueba

Cualquier producto de ingeniería puede ser probado de varias formas entre las que se encuentra conociendo la funcionalidad específica para la cual fue diseñado el producto, se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa.

3.4.1. Pruebas de caja negra

Se refiere a las pruebas que se llevan a cabo sobre la interfaz del *software*, por lo que los casos de prueba pretenden demostrar que las funciones del *software* son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del *software*.

La prueba de caja negra se centra principalmente en los requisitos funcionales del *software*. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

La prueba de caja negra no es una alternativa a las técnicas de prueba de la caja blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la caja blanca, por lo que se puede decir, que estas no son excluyentes, sino complementarias.

Capítulo 3 Implementación y Validación

Entre las principales características que presenta la prueba de caja negra están que:

- Verifican las especificaciones funcionales y no consideran la estructura interna del programa.
- Es hecha sin el conocimiento interno del producto.
- No validan funciones ocultas por tanto los errores asociados a ellas no serán encontrados.

En otras palabras, la prueba de la caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software.

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas se encuentran:

Técnica de la Partición de Equivalencia: Técnica 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.

Técnica del Análisis de Valores Límites: El análisis de valores límite (AVL) 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.

3.4.2. Diseño de casos de prueba

Los casos de prueba son un conjunto de condiciones o variables bajo las cuales el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio.

Capítulo 3 Implementación y Validación

La prueba es una actividad en la cual, un sistema o componente, es ejecutado bajo unas condiciones o requerimientos específicos, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Gestionar señalamiento	El requisito tiene 4 escenarios y es el encargado de gestionar señalamiento.	Listar señalamiento	
		Adicionar señalamiento	Aceptar, Aplicar y Cancelar
		Modificar señalamiento	Aceptar y Cancelar
		Eliminar señalamiento	Aceptar y cancelar

Tabla 22 Diseños de caso de prueba Gestionar Señalamiento

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Gestionar tipo de señalamiento	El requisito tiene 4 escenarios y es el encargado de gestionar tipo de señalamiento.	Listar tipo de señalamiento	
		Adicionar tipo de señalamiento	Aceptar, Aplicar y
		Modificar tipo de señalamiento	Aceptar y Cancelar
		Eliminar tipo de señalamiento	Aceptar y cancelar

Tabla 23 Diseño de caso de prueba Gestionar Tipo de señalamiento

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Gestionar recomendación	El requisito tiene 3 escenarios y es el encargado de Listar recomendación.	Adicionar recomendación	Aceptar, Aplicar y Cancelar
		Modificar recomendación	Aceptar y Cancelar
		Eliminar recomendación	Aceptar y cancelar
		Listar recomendación	

Tabla 24 Diseños de caso de prueba Gestionar recomendación

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Gestionar tipo de recomendación	El requisito tiene 3 escenarios y es el encargado de Listar tipo de recomendación.	Adicionar tipo de recomendación	Aceptar, Aplicar y
		Modificar tipo de recomendación	Aceptar y Cancelar
		Listar tipo de reconocimiento	
		Eliminar tipo de recomendación	Aceptar y cancelar

Tabla 25 Diseño de caso de prueba Gestionar Tipo de recomendación

Capítulo 3 Implementación y Validación

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Gestionar tipo de proceso.	El requisito tiene 4 escenarios y es el encargado de Gestionar tipo de proceso.	Listar tipo de proceso	
		Adicionar tipo de proceso	Aceptar, Aplicar y Cancelar
		Modificar tipo de proceso	Aceptar v Cancelar
		Eliminar tipo de proceso	Aceptar y cancelar

Tabla 26 Diseño de caso de prueba Gestionar tipo de proceso

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Gestionar período	El requisito tiene 4 escenarios y es el encargado de gestionar períodos.	Listar período	
		Adicionar período	Aceptar, Aplicar y Cancelar
		Modificar período	Aceptar y Cancelar
		Eliminar período	Aceptar y cancelar

Tabla 27 Diseño de caso de prueba Gestionar Período

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Gestionar comisión de evaluación	El requisito tiene 4 escenarios y es el encargado de gestionar las comisiones.	Listar comisión	
		Adicionar comisión	Aceptar, Aplicar y Cancelar
		Modificar comisión	Aceptar y Cancelar
		Eliminar comisión	Aceptar y cancelar

Tabla 28 Diseños de caso de prueba Gestionar comisión de evaluación

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Gestionar evaluadores	El requisito tiene 3 escenarios y es el encargado de gestionar evaluadores.	Listar evaluadores	
		Adicionar evaluadores	Aceptar, Aplicar y Cancelar
		Eliminar evaluadores	Aceptar y cancelar

Tabla 29 Diseño de caso de prueba Gestionar Evaluadores

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Gestionar evaluados	El requisito tiene 3 escenarios y es el	Listar evaluadores	
		Adicionar evaluadores	Aceptar, Aplicar y Cancelar

Capítulo 3 Implementación y Validación

	encargado de gestionar	Eliminar evaluadores	Aceptar y cancelar
--	------------------------	----------------------	--------------------

Tabla 30 Diseño de caso de prueba Gestionar evaluados

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Preparar proceso evaluativo	El requisito tiene 4 escenarios y es el encargado de gestionar preparar proceso evaluativos.	Listar preparar proceso	
		Adicionar preparar proceso	Aceptar, Aplicar y Cancelar
		Modificar preparar proceso	Aceptar y Cancelar
		Eliminar preparar proceso	Aceptar y cancelar

Tabla 31 Diseño de caso de prueba Gestionar preparar proceso evaluativo

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Definir proceso evaluativo	El requisito tiene 6 escenarios y es el encargado de gestionar definir procesos evaluativos.	Listar preparar proceso	
		Adicionar preparar proceso	Aceptar, Aplicar y Cancelar
		Modificar preparar proceso	Aceptar y Cancelar
		Eliminar preparar proceso	Aceptar y cancelar
		Asociar Recomendación	Asociar, restaurar, cancelar
		Asociar Señalamiento	Asociar, restaurar, cancelar

Tabla 32 Diseño de caso de prueba Gestionar definir proceso evaluativo

Id del escenario	Escenario	Denominación	Fecha inicio	Período	Tipo proceso	planilla	Respuesta del sistema	Resultado de la prueba
1.1	Adicionar definir proceso evaluativo	V(852mod 459)	V(5/05/2010)	V(852mod 459)	V(852mod 459)	V(852mod 459)	Se adiciona el proceso y se muestra el mensaje de confirmación.	Satisfactorio
1.2	Adicionar definir proceso evaluativo incorrectamente.	V(vacío)	V(5/05/2010)	V(852mod 459)	V(852mod 459)	V(852mod 459)	Se muestra un mensaje de error indicando que hay campos	Satisfactorio

Capítulo 3 Implementación y Validación

							requeridos vacíos.	
1.3	Cancelar operación.	NA	NA	NA	NA	NA	El sistema cancela las operaciones.	Satisfactorio

Tabla 33 Juego de datos del escenario Adicionar proceso evaluativo.

Id del escenario	Escenario	Denominación	Fecha inicio	Período	Tipo proceso	planilla	Respuesta del sistema	Resultado de la prueba
2.1	Modificar definir proceso evaluativo	V(852mod459)	V(5/05/2010)	V(852mod459)	V(852mod459)	V(852mod459)	Se modifica el proceso y se muestra el mensaje de confirmación.	Satisfactorio
2.2	Modificar definir proceso evaluativo incorrectam	V(vacío)	V(5/05/2010)	V(852mod459)	V(852mod459)	V(852mod459)	Se muestra un mensaje de error indicando que hay campos requeridos vacíos.	Satisfactorio
2.3	Cancelar operación.	NA	NA	NA	NA	NA	El sistema cancela las operaciones	Satisfactorio

Tabla 34 Juego de datos del escenario Modificar proceso evaluativo.

Id del escenario	Escenario	Respuesta del sistema	Resultado de la prueba
3.1	Eliminar definir proceso evaluativo	Se elimina el proceso y se muestra el mensaje de confirmación.	Satisfactorio
3.2	Eliminar definir proceso evaluativo incorrectamente.	Se muestra un mensaje de error indicando que no hay seleccionado ningún proceso evaluativo	Satisfactorio
3.3	Cancelar operación.	El sistema cancela las operaciones	Satisfactorio

Tabla 35 Juego de datos del escenario Eliminar proceso evaluativo.

Capítulo 3 Implementación y Validación

Id del escenario	Escenario	Respuesta del sistema	Resultado de la prueba
4.1	Asociar recomendación	Se asocia una o varias recomendaciones a un proceso evaluativo y se muestra el mensaje de confirmación.	Satisfactorio
4.2	Asociar recomendación incorrectamente.	Se muestra un mensaje de error indicando que no hay seleccionado ninguna recomendación	Satisfactorio
4.3	Cancelar operación.	El sistema cancela las operaciones	Satisfactorio

Tabla 36 Juego de datos del escenario Asociar recomendación.

Id del escenario	Escenario	Respuesta del sistema	Resultado de la prueba
5.1	Asociar señalamiento	Se asocia uno o varios señalamientos a un proceso evaluativo y se muestra el mensaje de confirmación.	Satisfactorio
5.2	Asociar señalamiento incorrectamente.	Se muestra un mensaje de error indicando que no hay seleccionado ningún señalamiento.	Satisfactorio
5.3	Cancelar operación.	El sistema cancela las operaciones.	Satisfactorio

Tabla 37 Juego de datos del escenario Asociar señalamiento.

Restantes escenarios ([Anexo 8](#)).

3.5. Conclusiones parciales

En este capítulo se desarrolló el diagrama de componentes y la matriz de integración de los mismos ya sea de componentes externos como internos lo cual refleja la combinación de estos, los diferentes servicios que se consumen entre sí, es decir, cómo está estructurado el módulo completo de evaluaciones al cual pertenece el componente desarrollado. Además, se realizaron las pruebas de unidad que permitieron validar la solución propuesta, se realizaron descripciones del mismo teniendo en cuenta: tipo de prueba, objetivo de la prueba y alcance. Se describieron los valores utilizados y se evaluó la ejecución de la prueba así como los resultados obtenidos.

Conclusiones generales

Con la culminación de la presente investigación, queda confirmada la necesidad de la implementación de un componente dentro del subsistema “Evaluación de desempeño” que permita configurar los procesos evaluativos en una entidad. La implementación del componente permitió el ahorro de una serie de procesos engorrosos que consumían tiempo y esfuerzos y que daban un alto margen al fraude, además de que se resolvió de forma eficiente uno de los problemas que más afecta a Cuba en la actualidad, los temas relacionados con la gestión de las evaluaciones, así como la adaptación de la solución a las nuevas concepciones de informatización del país. De manera general, con la elaboración de este trabajo se logró:

- Realizar un estudio del estado del arte de sistemas similares al propuesto donde se llegó a la conclusión de que sus características no eran las apropiadas y deseadas para aplicar a la solución descrita en este trabajo
- Dar cumplimiento de forma satisfactoria al objetivo general, implementándose el Componente de Configuración de los Proceso de Evaluativos en una entidad.
- La realización de un sistema potencialmente navegable, confiable y eficiente.
- Utilizar las herramientas, lenguajes y el proceso de desarrollo, necesarios para implementar la aplicación con la calidad requerida, haciendo uso de software libre.
- Realizar pruebas unitarias de caja negra a la solución.

Recomendaciones

Con el objetivo de mejorar la solución planteada se proponen las siguientes recomendaciones:

Continuar un desarrollo iterativo e incremental de los componentes Señalamientos, Recomendaciones, Tipos de Procesos, Períodos, Definir Proceso Evaluativo y Preparar Proceso Evaluativo, obteniendo como resultado un incremento en cada iteración que permita perfeccionar el funcionamiento de dichos componentes.

Incorporar a los componentes de Definir Proceso Evaluativo y Preparar Proceso Evaluativo un mayor nivel de complejidad que le permita configurar procesos de gestión de la evaluación dinámicamente.

Seguir perfeccionando la aplicación de acuerdo con las nuevas necesidades que van surgiendo en el transcurso del tiempo.

Bibliografía

1. **Cuba, Gaceta oficial de.** Resolución 21. 2007. Vol. Apartado segundo.
2. **Herrera Martínez, Rolando y Fernández Montiel, Joan Manuel.** *Sistema informático para la gestión de evaluaciones*. Ciudad de la Habana : s.n., 2009.
3. YeRu\$h@. *ExtJS 2.0*. [En línea] [Citado el: 2010 de Enero de 28.] <http://yerusha.wordpress.com/2008/01/10/extjs-20/>.
4. **Leopoldo Magaña, Carlos.** Zend Framework, una introducción. [En línea] 2005-2008. [Citado el: 29 de Enero de 2010.] <http://www.carlosleopoldo.com/post/zend-framework-una-introduccion/>.
5. **Celis, Ismael.** ESTADOBETA desarrollo web con estándares. *Active Record*. [En línea] 2005. [Citado el: 6 de Febrero de 2010.] <http://www.estadobeta.com/2006/05/02/active-record/>.
6. **Medinilla, Nelson.** Proyecto Práctico de Construcción de un Sistema Software. *Resumen*. [En línea] 6 de Febrero de 2010. http://is.ls.fi.upm.es/udis/docencia/proyecto/docs/Confuso_Modelo_Vista_Controlador.doc.
7. **Sergio, Carlos y otros.** E Boletín N°8 Rama de Estudiantes de IEEE-UNED. [En línea] 2007. [Citado el: 5 de Febrero de 2010.] http://www.ieec.uned.es/ieeee/investigacion/ieeee_dieec/sb/boletin_8_Octubre_2007.pdf
8. **Costa, Helkin R Coello.** [En línea] 27 de Noviembre de 2002. [Citado el: 20 de Marzo de 2010.] http://www.informatizate.net/articulos/dime_como_programas_y_te_dire_quien_eres_82004.html.
9. **Beck, Keny.** *Implementation Patterns*. s.l. : Addison Wesley, 2007. ISBN: 978-0321413093
10. *Seguridad en Aplicaciones Web*. **Racciatti, Hernán Marcelo.** 96, s.l. : @RROBA, 2005.
11. **Pressman, R. S.** 1998. Ingeniería de software. Un enfoque práctico. 1998.
12. **UCID.** *Estandar para el diseño de interfaces v1.1*. 2008.
13. **UCID.** *Ayuda al usuario del caso de estudio en el nuevo marco de trabajo*. 2008.
14. **Proenza Y.** *Revista Atix. Diseavanzado de aplicaciones web. ExtJS-Zend Framework-Doctrine*. 2009
15. **Santiago D, Dondero M y Urquiza S.** Un framework orientado a objetos para la implementación de métodos discretos. *Mecánica Computacional Vol XXVII*. 2008

Referencias bibliográficas

1. **Cuba, Gaceta oficial de.** Resolución 21. 2007. Vol. Apartado segundo.
2. **Herrera Martínez, Rolando y Fernández Montiel, Joan Manuel.** *Sistema informático para la gestión de evaluaciones*. Ciudad de la Habana : s.n., 2009.
3. **YeRu\$h@.** *ExtJS 2.0.* [En línea] [Citado el: 2010 de Enero de 28.] [http://yerusha.wordpress.com/2008/01/10/extjs-20/..](http://yerusha.wordpress.com/2008/01/10/extjs-20/)
4. **Leopoldo Magaña, Carlos.** Zend Framework, una introducción. [En línea] 2005-2008. [Citado el: 29 de Enero de 2010.] [http://www.carlosleopoldo.com/post/zend-framework-una-introduccion/..](http://www.carlosleopoldo.com/post/zend-framework-una-introduccion/)
5. **Celis, Ismael.** ESTADOBETA desarrollo web con estándares. *Active Record.* [En línea] 2005. [Citado el: 6 de Febrero de 2010.] [http://www.estadobeta.com/2006/05/02/active-record/.](http://www.estadobeta.com/2006/05/02/active-record/)
6. **Sergio, Carlos y otros.** E Boletín N°8 Rama de Estudiantes de IEEE-UNED. [En línea] 2007. [Citado el: 5 de Febrero de 2010.] http://www.ieec.uned.es/ieee/investigacion/ieee_dieec/sb/boletin_8_Octubre_2007.pdf
7. **Booch, G, Rumbaugh, J, and Jacobson, I.** 1999. The Unified Modeling Language User Guide. 1999.
8. **Costa, Helkin R Coello.** [En línea] 27 de Noviembre de 2002. [Citado el: 20 de Marzo de 2010.] [http://www.informatizate.net/articulos/dime_como_programas_y_te_dire_quien_eres_23082004.html.](http://www.informatizate.net/articulos/dime_como_programas_y_te_dire_quien_eres_23082004.html)
9. **Beck, Keny.** *Implementation Patterns.* s.l. : Addison Wesley, 2007. ISBN: 978-0321413093
10. *Seguridad en Aplicaciones Web.* **Racciatti, Hernán Marcelo.** 96, s.l. : @RROBA, 2005.
11. **Pressman, R. S.** 1998. Ingeniería de software. Un enfoque práctico. 1998.

Glosario de términos

[A]

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]

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.

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.

[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.

Encapsulamiento: Significa reunir a todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión de los componentes del sistema. Algunos autores confunden este concepto con el principio de ocultación, principalmente porque se suelen emplear conjuntamente.

[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).

IoC: Inversión de control (*Inversion of Control* en inglés) es un método de programación en el que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales, en los que la interacción se expresa de forma imperativa haciendo llamadas a procedimientos (*procedure calls*) o funciones.

[M]

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.

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.

[N]

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

[S]

Software: Es un programa o aplicación de 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.

Servidor: En informática, un servidor es una computadora que, formando parte de una red, provee servicios a otras computadoras denominadas clientes.

[O]

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

Open source: Código abierto es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas las cuales destacan en el llamado software libre.

[P]

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

[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.

[W]

Web: En informática, la World Wide Web, cuya traducción podría ser *Red Global Mundial* o "Red de Amplitud Mundial", es un sistema de documentos de hipertexto enlazados y accesibles a través de Internet.